MariaDB Server

Documentation

Snapshot 2026-08-15
4002 pages

© MariaDB. Content licensed CC BY-SA / GNU FDL. The authoritative, continuously updated version of this documentation is at mariadb.com/docs.

Contents

MariaDB Server Documentation

Quickstart Guides

- Installing MariaDB Server Guide
- Adding & Changing Data Guide
- Essential Queries Guide
- Basics Guide
- Altering Tables Guide
- Connecting to MariaDB Guide
- Troubleshooting Connection Issues Guide
- Doing Time Guide
- Importing Data Guide
- Essentials of an Index Guide
- Getting Started with Indexes Guide
- Joining Tables with JOIN Clauses
- Advanced Joins
- Configuring MariaDB for Remote Client Access Guide
- Getting Data Guide
- Basic SQL Statements Guide
- Basic SQL Debugging Guide
- MariaDB String Functions Guide
- Restoring Data from Dump Files Guide
- Changing Times in MariaDB
- System & Status Variables Guide
- Making Backups with mariadb-dump Guide
- A MariaDB Primer Guide
- Creating & Using Views Guide
- Database Applications
  - Introduction & Background
  - Database Design
  - Application Code
  - Canary Testing

Server Usage

- Basics
  - Basic Queries
- Backup & Restore
  - Backup and Restore Overview
  - Forming a Backup Strategy
  - Backup Optimization
  - MariaDB Enterprise Backup
  - mariadb-backup
    - mariadb-backup Overview
    - mariadb-backup Options
    - Full Backup and Restore (mariadb-backup)
    - Incremental Backup and Restore (mariadb-backup)
    - Partial Backup and Restore (mariadb-backup)
    - Point-In-Time Recovery (PITR, mariadb-backup)
    - Restoring Individual Databases From a Full Backup (mariadb-backup)
    - Restoring Individual Tables and Partitions (mariadb-backup)
    - Setting up a Replica (mariadb-backup)
    - Files Backed Up by mariadb-backup
    - Files Created by mariadb-backup
    - Using Encryption and Compression Tools With mariadb-backup
    - Encrypted Files Backup (mariadb-backup)
    - How mariadb-backup Works
    - mariadb-backup and BACKUP STAGE
  - Point-In-Time Recovery (InnoDB Log Archiving)
  - Replication as a Backup Solution
  - Backup and Restore via dbForge Studio
- Tables
  - Copying Tables Between Databases and Servers
  - ALTER TABLE
  - CREATE TABLE
  - DROP TABLE
  - Data Types
  - Storage Engines
- Partitioning Tables
  - Partitioning Overview
  - Partition Pruning and Selection
  - Partition Maintenance
  - Partitioning Types
    - Partitioning Types Overview
    - HASH Partitioning Type
    - KEY Partitioning Type
    - LINEAR HASH Partitioning Type
    - LINEAR KEY Partitioning Type
    - LIST Partitioning Type
    - RANGE COLUMNS and LIST COLUMNS Partitioning Types
    - RANGE Partitioning Type
  - Partitioning Limitations
  - Partitions Files
  - Partitions Metadata
- Stored Routines
  - Stored Procedures
    - Stored Procedure Overview
    - CREATE PROCEDURE
    - ALTER PROCEDURE
    - DROP PROCEDURE
  - Stored Functions
    - Stored Function Overview
    - Stored Aggregate Functions
    - Stored Routine Privileges
    - DROP FUNCTION
    - Stored Function Limitations
  - Binary Logging of Stored Routines
  - Stored Routine Limitations
  - DBMS_OUTPUT
  - Storage Engines Overview
  - Choosing the Right Storage Engine
  - ARCHIVE
  - ARIA
    - Aria Storage Engine
    - Aria Group Commit
    - Aria Status Variables
    - Aria Storage Formats
    - Aria System Variables
    - Aria Two-step Deadlock Detection
    - Aria FAQ
    - The Aria Name
  - BLACKHOLE
  - CONNECT
    - Introduction to the CONNECT Engine
    - Using CONNECT
      - Using CONNECT - Condition Pushdown
      - Using CONNECT - Exporting Data From MariaDB
      - Using CONNECT - General Information
      - Using CONNECT - Importing File Data Into MariaDB Tables
      - Using CONNECT - Indexing
      - Using CONNECT - Partitioning and Sharding
      - Using CONNECT - Virtual and Special Columns
    - Installing CONNECT

- CONNECT Create Table Options
- CONNECT Data Types
- CONNECT Table Types
  - CONNECT Table Types Overview
  - CONNECT BIN Table Type
  - CONNECT CSV and FMT Table Types
  - CONNECT DBF Table Type
  - CONNECT DOS and FIX Table Types
  - CONNECT - External Table Types
  - CONNECT - Files Retrieved Using Rest Queries
  - CONNECT INI Table Type
  - CONNECT JDBC Table Type: Accessing Tables from Another DBMS
  - CONNECT JSON Table Type
  - CONNECT MONGO Table Type: Accessing Collections from MongoDB
  - CONNECT MYSQL Table Type: Accessing MySQL/MariaDB Tables
  - CONNECT - NoSQL Table Types
  - CONNECT OCCUR Table Type
  - CONNECT ODBC Table Type: Accessing Tables From Another DBMS
  - CONNECT PIVOT Table Type
  - CONNECT PROXY Table Type
  - CONNECT Table Types - Catalog Tables
  - CONNECT Table Types - Data Files
  - CONNECT Table Types - OEM: Implemented in an External LIB
  - CONNECT Table Types - Special "Virtual" Tables
  - CONNECT Table Types - VIR
  - CONNECT TBL Table Type: Table List
  - CONNECT - Using the TBL and MYSQL Table Types Together
  - CONNECT VEC Table Type
  - CONNECT XCOL Table Type
  - CONNECT XML Table Type
  - CONNECT Zipped File Tables
  - Inward and Outward Tables
- CONNECT Security
- CONNECT System Variables
- Adding the REST Feature as a Library Called by an OEM Table
- Compiling JSON UDFs in a Separate Library
- Making the GetRest Library
- OEM Table Example
- Current Status of the CONNECT Handler
- JSON Sample Files

CSV
- CSV Overview
- Checking and Repairing CSV Tables

DuckDB

FederatedX
- About FederatedX
- Differences Between FederatedX and Federated

InnoDB
- InnoDB Introduction
- InnoDB File Format
- InnoDB Limitations
- AUTO_INCREMENT Handling in InnoDB
- Binary Log Group Commit and InnoDB Flushing Performance
- InnoDB Asynchronous I/O
- InnoDB Buffer Pool
- InnoDB Change Buffering
- InnoDB Data Scrubbing
- InnoDB Doublewrite Buffer
- InnoDB Flush Method
- InnoDB Lock Modes
- InnoDB Log Archiving
- InnoDB Monitors
- InnoDB Page Compression
- InnoDB Page Flushing
- InnoDB Purge
- InnoDB Undo Log
- InnoDB Redo Log
- InnoDB Strict Mode
- InnoDB System Variables
- InnoDB Versions
- InnoDB Architecture for MariaDB Enterprise Server
  - MariaDB Enterprise Server InnoDB Background Thread Pool
  - MariaDB Enterprise Server InnoDB I/O Threads
- InnoDB Online DDL
  - InnoDB Online DDL Overview
  - InnoDB Online DDL Operations with the INPLACE Alter Algorithm
  - InnoDB Online DDL Operations with the INSTANT Alter Algorithm
  - InnoDB Online DDL Operations with the NOCOPY Alter Algorithm
  - Instant ADD COLUMN for InnoDB
- InnoDB Row Formats
  - InnoDB Row Formats Overview
  - InnoDB COMPACT Row Format
  - InnoDB COMPRESSED Row Format
  - InnoDB DYNAMIC Row Format
  - InnoDB REDUNDANT Row Format
  - Troubleshooting Row Size Too Large Errors with InnoDB
- InnoDB Tablespaces
  - InnoDB File-Per-Table Tablespaces
  - InnoDB System Tablespaces
  - InnoDB Temporary Tablespaces
- InnoDB Troubleshooting
  - InnoDB Troubleshooting Overview
  - InnoDB Data Dictionary Troubleshooting
  - InnoDB Recovery Modes
- MariaDB Enterprise Server InnoDB Operations
  - Configure the InnoDB Buffer Pool
  - Configure the InnoDB I/O Threads
  - Configure the InnoDB Purge Threads
  - Configure the InnoDB Redo Log
  - Configure the InnoDB Undo Log
  - Schema Changes
    - InnoDB Schema Changes
  - InnoDB - Unmaintained
    - About XtraDB
    - Using InnoDB Instead of XtraDB

MEMORY

MERGE

Mroonga
- Mroonga Overview
- About Mroonga
- Mroonga Status Variables
- Mroonga System Variables
- Mroonga User-Defined Functions
  - Creating Mroonga User-Defined Functions
  - last_insert_grn_id
  - mroonga_command
  - mroonga_escape
  - mroonga_highlight_html
  - mroonga_normalize
  - mroonga_snippet
  - mroonga_snippet_html

MyISAM
- MyISAM Overview

- MyISAM Index Storage Space
- MyISAM Storage Formats
- MyISAM System Variables
- MyRocks
  - About MyRocks for MariaDB
  - Building MyRocks in MariaDB
  - Differences Between MyRocks Variants
  - Getting Started with MyRocks
  - Loading Data Into MyRocks
  - MyRocks Bloom Filters
  - MyRocks CHECK TABLE
  - MyRocks Data Compression
  - MyRocks Group Commit with Binary log
  - MyRocks Index-Only Scans
  - MyRocks and Replication
  - MyRocks START TRANSACTION WITH CONSISTENT SNAPSHOT
  - MyRocks Column Families
  - MyRocks Performance Troubleshooting
  - Optimizer Statistics in MyRocks
  - MyRocks Transactional Isolation
  - MyRocks Status Variables
  - MyRocks System Variables
- OQGRAPH
  - OQGRAPH Overview
  - Installing OQGRAPH
  - Compiling OQGRAPH
  - Building OQGRAPH Under Windows
  - OQGRAPH Examples
- PERFORMANCE_SCHEMA
- SEQUENCE Storage Engine
- S3 Storage Engine
  - Using the S3 Storage Engine
  - S3 Status Variables
  - S3 System Variables
  - S3 Engine Internals
  - Testing Connections to S3
  - aria_s3_copy
- SphinxSE
  - About SphinxSE
  - Installing Sphinx
  - Installing and Testing SphinxSE with MariaDB
  - Configuring Sphinx
- Spider
  - Spider Storage Engine Overview
  - Spider Storage Engine Introduction
    - MariaDB Enterprise Spider Schema Design
    - MariaDB Enterprise Spider Operations
      - Federated MariaDB Enterprise Spider Topology Operations
      - Spider Federated Overview
      - Federated MariaDB Enterprise Spider Topology Backup and Restore
      - Federated MariaDB Enterprise Spider Topology Migrate Tables
      - ODBC MariaDB Enterprise Spider Topology Operations
      - Use Spider ODBC to Connect to Oracle
      - Sharded MariaDB Enterprise Spider Topology Operations
      - Spider Sharded Overview
      - Sharded MariaDB Enterprise Spider Topology Add a Shard
      - Sharded MariaDB Enterprise Spider Topology Backup and Restore
  - Spider Differences Between SpiderForMySQL and MariaDB
  - Spider Feature Matrix
  - Spider Use Cases
  - Spider Installation
  - Spider Cluster Management
  - Spider Case Studies
  - Spider Benchmarks
  - Information Schema SPIDER_WRAPPER_PROTOCOLS Table
  - Spider Storage Engine Core Concepts
  - Spider System Variables
  - Spider Table Parameters
  - Spider Functions
    - SPIDER_BG_DIRECT_SQL
    - SPIDER_COPY_TABLES
    - SPIDER_DIRECT_SQL
    - SPIDER_FLUSH_TABLE_MON_CACHE
  - Spider FAQ
- VIDEX Storage Engine
- Converting Tables from MyISAM to InnoDB
- Machine Learning with MindsDB
- FEDERATED Storage Engine
- Triggers & Events
  - Event Scheduler
    - Events Overview
    - ALTER EVENT
    - Event Limitations
  - Triggers
    - Trigger Overview
    - CREATE TRIGGER
    - Trigger Limitations
    - Triggers and Implicit Locks
- User-Defined Functions
  - User-Defined Functions Overview
  - Creating User-Defined Functions
  - CREATE FUNCTION UDF
  - DROP FUNCTION UDF
  - User-Defined Functions Calling Sequences
  - User-Defined Functions Security
- Views
  - ALTER VIEW
  - CREATE VIEW
  - DROP VIEW
  - Information Schema VIEWS Table
  - Inserting and Updating with Views
  - View Algorithms

Server Management

- Deployment
  - MariaDB Package Repository Setup and Usage
  - General Deployment Instructions
    - Best Practices
    - Deployment Methods
  - Installing Enterprise Server
    - Customer Download Token
    - Installing MariaDB Enterprise Server
    - Installing Primary/Replica
    - Enterprise Server with ColumnStore (Local Storage)
    - Enterprise Server with ColumnStore (Object Storage)
  - Installing Community Server
    - MariaDB ID Sign Up
    - MariaDB Binary Packages
      - GPG
      - Installing MariaDB .deb Files
      - Installing MariaDB RPM Files
        - About the MariaDB RPM Files
        - Installing MariaDB with yum/dnf
        - Installing MariaDB with zypper
        - Checking MariaDB RPM Package Signatures
        - Installing MariaDB With the rpm Tool
        - MariaDB for DirectAdmin Using RPMs

- MariaDB Installation via RPMs on CentOS 7
- Troubleshooting MariaDB Installs on RHEL / CentOS
- Why Source RPMs (SRPMs) Aren't Packaged For Some Platforms
- Installing MariaDB Alongside MySQL
- Installing MariaDB Binary Tarballs
- Installing MariaDB MSI Packages on Windows
- Installing MariaDB on macOS
- Installing MariaDB on macOS (PKG)
- Installing MariaDB Windows ZIP Packages
- Deploy with Package Tarballs
- Deploy with Local Package Repository Mirrors
- Installing System Tables
- Installing System Tables on Unix
- Installing System Tables on Windows
- Troubleshooting Installation Issues
- Error: symbol mysql_get_server_name, version libmysqlclient_16 not defined
- Installation issues on Windows
- Installation Issues with PHP5
- Installing on an Old Linux Version
- Installation Issues on Debian and Ubuntu
- apt-upgrade Fails, But the Database is Running
- Differences in MariaDB in Debian (and Ubuntu)
- Installing MariaDB on IBM Cloud

Compiling MariaDB From Source
- Compiling MariaDB From Source: The Master Guide
- Building MariaDB on Windows
- Cross-compiling MariaDB
- Compiling MariaDB with Extra Modules/Options
- Specifying Which Plugins to Build
- Using MariaDB with TCMalloc or jemalloc

Configuring MariaDB
- Configuring MariaDB with Option Files
- MariaDB Environment Variables
- Advanced Configuration
- Atomic Write Support
- Configuring Linux for MariaDB
- Configuring MariaDB for Optimal Performance
- Configuring Swappiness
- Fusion-io
- Fusion-io Introduction

Upgrading MariaDB
- Platform Specific Upgrade Guides
- Upgrading on Linux
- Upgrading Between Major MariaDB Versions
- Upgrading Between Minor MariaDB Versions
- Upgrading MariaDB on Windows
- Upgrading from MySQL to MariaDB
- MariaDB Community Server Upgrade Paths
- Upgrading from MariaDB 11.8 to MariaDB 12.3
- Upgrading from MariaDB 11.4 to MariaDB 11.8
- Upgrading from MariaDB 11.3 to MariaDB 11.4
- Upgrading from MariaDB 10.11 to MariaDB 11.4
- Upgrading from MariaDB 10.6 to MariaDB 10.11
- Upgrading from MariaDB 10.5 to MariaDB 10.6
- Archived Guides (Unmaintained CS Versions)
- Upgrading from MariaDB 11.2 to MariaDB 11.3
- Upgrading from MariaDB 11.1 to MariaDB 11.2
- Upgrading from MariaDB 11.0 to MariaDB 11.1
- Upgrading from MariaDB 10.11 to MariaDB 11.0
- Upgrading from MariaDB 10.7 to MariaDB 10.8
- Upgrading from MariaDB 10.6 to MariaDB 10.7
- Upgrading from MariaDB 10.4 to MariaDB 10.5
- Upgrading from MariaDB 10.3 to MariaDB 10.4
- Upgrading from MariaDB 10.2 to MariaDB 10.3
- Upgrading from MariaDB 10.1 to MariaDB 10.2
- Upgrading from MariaDB 10.0 to MariaDB 10.1
- Upgrading from MariaDB 5.3 to MariaDB 5.5
- General Upgrade Information
- MariaDB Enterprise Server Upgrade Paths
- MariaDB Enterprise Server 11.8
- Upgrade to MariaDB Enterprise Server 11.8
- Upgrade MariaDB Enterprise Server from 11.8.X to 11.8.Y
- Upgrading from MariaDB Enterprise Server 10.6 to 11.8
- Upgrade from MariaDB Community Server 10.6 to MariaDB Enterprise Server 11.8
- MariaDB Enterprise Server 11.4
- Upgrade to MariaDB Enterprise Server 11.4
- Upgrade MariaDB Enterprise Server from 11.4.X to 11.4.Y
- Upgrade MariaDB Enterprise Server from 10.6 to 11.4
- Upgrade from MariaDB Community Server to MariaDB Enterprise Server 11.4
- MariaDB Enterprise Server 10.6
- Upgrade to MariaDB Enterprise Server 10.6
- Upgrade MariaDB Enterprise Server from 10.6.X to 10.6.Y
- Upgrade from MariaDB Community Server to MariaDB Enterprise Server 10.6

Downgrading MariaDB

Migrating to MariaDB
- Differences Between MariaDB and Other DBMSs
- DDL Export Guide
- Migrating to MariaDB from MySQL
- MySQL to MariaDB Migration: The Master Guide
- MySQL to MariaDB Compatibility Matrix
- Migrating Galera Cluster
- Migrating from MySQL to MariaDB Galera Cluster (Node-by-Node In-Place)
- MySQL to MariaDB Migrator
- Installation and First Run
- Migration Modes
- Application User Migration
- Environment Variables
- Migrate with Serial Streaming Copy
- Migrate with Offline Copy
- Migrate with Parallel Restartable Streaming Copy
- Migrate with Replication
- Migrating to MariaDB from PostgreSQL
- Migrating to MariaDB from Oracle
- Oracle XE 11.2. and MariaDB 10.1 integration on Ubuntu 14.04 and Debian systems
- Migrating to MariaDB from SQL Server
- MariaDB Backups Overview for SQL Server Users
- Understanding MariaDB Architecture
- MariaDB Features Not Available in SQL Server
- SQL Server Features Not Available in MariaDB
- SQL Server Features Implemented Differently in MariaDB
- SQL Server and MariaDB Types Comparison
- Syntax Differences between MariaDB and SQL Server
- MariaDB Authorization and Permissions for SQL Server Users
- MariaDB Replication Overview for SQL Server Users
- MariaDB Transactions and Isolation Levels for SQL Server Users
- Moving Data Between SQL Server and MariaDB
- Repairing MariaDB Tables for SQL Server Users
- Setting Up MariaDB for Testing for SQL Server Users
- Migrating From MariaDB RDS to MariaDB Cloud

MariaDB on Amazon RDS

Automated Deployment & Administration

Starting & Stopping

- Starting and Stopping Overview
- launchd (macOS)
- systemd (Linux)
  - Starting MariaDB on systemd
  - Configuring MariaDB for systemd
- sysVinit
- mariadbd-safe
- mariadbd-multi
- mariadbd Options
- mariadbd
- mysql.server
- Running MariaDB From the Build Directory
- Running Multiple MariaDB Server Processes
- Specifying Permissions for Schema (Data) Directories and Tables
- Switching Between Different Installed MariaDB Versions
- What to Do if MariaDB Doesn't Start
- Why to Automate MariaDB Deployments and Management
- Comparison Between Automation Systems
- Automating MariaDB Tasks with Events
- Automating Upgrades with MariaDB.Org Downloads REST API
- HashiCorp Vault and MariaDB
- Orchestrator Overview
- Ansible and MariaDB
  - Ansible Overview for MariaDB Users
  - Deploying Docker Containers with Ansible
  - Deploying to Remote Servers with Ansible
  - Existing Ansible Modules and Roles for MariaDB
  - Installing MariaDB .deb Files with Ansible
  - Managing Secrets in Ansible
  - Running mariadb-tzinfo-to-sql with Ansible
- Puppet and MariaDB
  - Puppet Overview
  - Deploying Docker Containers with Puppet
  - Existing Puppet Modules for MariaDB
  - Puppet hiera Configuration System
  - Bolt Examples
- MariaDB Containers
  - Adding Plugins to the MariaDB Docker Official Image
  - Benefits of Managing MariaDB Containers with Orchestration Software
  - Container Backup and Restoration
  - Container Security
  - Creating a Custom Container Image
  - Deploy MariaDB Enterprise Server with Docker
  - MariaDB Enterprise Docker Registry for MariaDB Enterprise Server
  - Docker and AWS EC2
  - Docker and Google Cloud
  - Docker and Microsoft Azure
  - Docker Official Image Frequently Asked Questions
  - Installing and Using MariaDB via Docker
  - MariaDB Container Cheat Sheet
  - MariaDB Enterprise Server Docker Official Image Environment Variables
  - Setting Up a LAMP Stack with Docker Compose
  - Using Healthcheck
  - MariaDB Enterprise Container Image Configuration Reference
- Kubernetes and MariaDB
  - Kubernetes Operators for MariaDB
  - Kubernetes Overview for MariaDB Users
- Vagrant and MariaDB
  - Creating a Vagrantfile
  - Vagrant Overview for MariaDB Users
  - Vagrant Security

Server Monitoring & Logs

- Overview of MariaDB Logs
- Securing MariaDB Logs
- Binary Log
  - Overview of the Binary Log
  - Activating the Binary Log
  - Binary Log Formats
  - Using and Maintaining the Binary Log
  - Row Binlog Events
  - InnoDB-Based Binary Log
  - Compressing Events to Reduce Size of the Binary Log
  - Flashback
  - Group Commit for the Binary Log
  - Relay Log
  - Selectively Skipping Replication of Binlog Events
- Error Log
- General Query Log
- MyISAM Log
- Rotating Logs on Unix and Linux
- Slow Query Log
  - Slow Query Log Overview
  - EXPLAIN in the Slow Query Log
  - log_slow_always_query_time System Variable
- SQL Error Log Plugin
- Transaction Coordinator Log
  - Transaction Coordinator Log Overview
  - Heuristic Recovery with the Transaction Coordinator Log
- Writing Logs Into Tables

Variables and Modes

- Options, System & Status Variables
- Server Status Variables
- Server System Variables
- NEW_MODE
- OLD_MODE
- SQL_MODE

Security

Securing MariaDB

- Running MariaDB as root
- MariaDB on SELinux

Encryption

- TLS and Cryptography Libraries
- Data-in-Transit Encryption
  - Secure Connections Overview
  - Securing Connections for Client and Server
  - Zero-Configuration SSL
  - Replication with Secure Connections
  - Enabling TLS on MariaDB Server
  - Certificate Creation with OpenSSL
  - SSL/TLS System Variables
  - Using TLSv1.3
  - Post-Quantum Cryptography in MariaDB TLS
- Data-at-Rest Encryption
  - Data-at-Rest Encryption (TDE) Fundamentals
  - Managing Binary Log Encryption
  - Aria Encryption
    - Aria: Encryption Overview
    - Aria: Enabling Encryption
    - Aria: Encryption Keys
    - Aria: Disabling Encryption
  - InnoDB Encryption
    - InnoDB: Encryption Overview
    - InnoDB: Enabling Encryption
    - InnoDB: Encryption Keys
    - InnoDB: Disabling Encryption
    - InnoDB: Background Encryption Threads

- InnoDB: Encryption Troubleshooting
- Key Management and Encryption Plugins
- Encryption Key Management
- File Key Management Encryption Plugin
- AWS Key Management Encryption Plugin
- Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Advanced Usage
- Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Setup Guide
- Hashicorp Key Management Plugin
- Uninstall Key Management Plugins
- User Account Management
- Roles
- Roles Overview
- System Users, Roles, Privileges
- Account Locking
- Authentication
- Incrementing of the access_denied_errors Status Variable
- User Password Expiry
- Catalogs
- Catalogs Overview
- Starting with Catalogs
- Catalog-Specific Functions and Variables
- Catalog Status Variables
- Connecting to a Server Configured for Catalogs
- CREATE CATALOG
- SHOW CREATE CATALOG
- SHOW CATALOGS
- USE CATALOG
- DROP CATALOG
- Authentication with Enterprise Server
- Authentication for MariaDB Enterprise Server
- Authentication with gssapi
- Limiting Size of Created Disk Temporary Files and Tables
- Limiting Size of Created Disk Temporary Files and Tables Overview
- max_tmp_session_space_usage System Variable
- max_tmp_total_space_usage System Variable
- Security Vulnerabilities (CVE) Fixed in MariaDB
- Security Vulnerabilities (CVE) Fixed in MariaDB Enterprise Server
- Security Vulnerabilities (CVE) Fixed in MariaDB Community Server
- Security Vulnerabilities (CVE) Fixed in Oracle MySQL That Did Not Exist in MariaDB

Architecture

- Topologies
- Topologies Overview
- Compatibility
- Columnstore Object Storage
- Step 1: Prepare ColumnStore Nodes
- Step 2: Configure Shared Local Storage
- Step 3: Install MariaDB Enterprise Server
- Step 4: Start and Configure MariaDB Enterprise Server
- Step 5: Test MariaDB Enterprise Server
- Step 6: Install MariaDB MaxScale
- Step 7: Start and Configure MariaDB MaxScale
- Step 8: Test MariaDB MaxScale
- Step 9: Import Data
- ColumnStore Shared Local Storage
- Step 1: Prepare ColumnStore Nodes
- Step 2: Configure Shared Local Storage
- Step 3: Install MariaDB Enterprise Server
- Step 4: Start and Configure MariaDB Enterprise Server
- Step 5: Test MariaDB Enterprise Server
- Step 6: Install MariaDB MaxScale
- Step 7: Start and Configure MariaDB MaxScale
- Step 8: Test MariaDB MaxScale
- Step 9: Import Data
- Galera Cluster
- Step 1: Install MariaDB Enterprise Server
- Step 2: Start and Configure MariaDB Enterprise Server
- Step 3: Test MariaDB Enterprise Server
- Step 4: Install MariaDB MaxScale
- Step 5: Start and Configure MariaDB MaxScale
- Step 6: Test MariaDB MaxScale
- HTAP
- Step 1: Prepare ColumnStore Node
- Step 2: Install MariaDB Enterprise Server
- Step 3: Start and Configure MariaDB Enterprise Server
- Step 4: Test MariaDB Enterprise Server
- Step 1: Install MariaDB Enterprise Server
- Step 2: Start and Configure MariaDB Enterprise Server on Primary Server
- Step 3: Start and Configure MariaDB Enterprise Server on Replica Servers
- Step 4: Test MariaDB Enterprise Server
- Step 5: Install MariaDB MaxScale
- Step 6: Start and Configure MariaDB MaxScale
- Step 7: Test MariaDB MaxScale
- Single Node Topologies
- Community Server
- Community Server with ColumnStore
- Step 1: Prepare Systems for Enterprise ColumnStore Nodes
- Step 2: Install Enterprise ColumnStore
- Step 3: Start and Configure Enterprise ColumnStore
- Step 4: Test Enterprise ColumnStore
- Step 5: Bulk Import of Data
- Step 1: Prepare Systems for Enterprise ColumnStore Nodes
- Step 2: Install Enterprise ColumnStore
- Step 3: Start and Configure Enterprise ColumnStore
- Step 4: Test Enterprise ColumnStore
- Step 5: Bulk Import of Data
- Spider Storage Engine
- MariaDB Enterprise Spider Topologies
- Federated MariaDB Enterprise Spider Topology
- ODBC MariaDB Enterprise Spider Topology
- Sharded MariaDB Enterprise Spider Topology
- Spider Federated
- Step 1: Install Enterprise Spider
- Step 2: Configure Spider Node and Data Node
- Step 3: Test Spider Federated Topology
- Spider Sharded
- Step 1: Install Enterprise Spider
- Step 2: Configure Spider Node and Data Nodes
- Step 3: Test Spider Sharded Topology
- Server Constraints
- Server Constraints Overview
- AUTO_INCREMENT Constraints
- FOREIGN KEY Constraints
- NOT NULL Constraints
- PRIMARY KEY Constraints
- UNIQUE Constraints

Clients & Utilities

- Administrative Tools
- dbdeployer
- innochecksum
- mariadb-access
- mariadb-admin
- mariadb-conv
- mariadb-setpermission

- mariadb-show
- mariadb-plugin
- mariadb-report
- mariadb-find-rows
- mariadb-tzinfo-to-sql
- mariadb-waitpid
- my_print_defaults
- mariadb-embedded
- perror
- replace
- resolve_stack_dump

Aria Clients and Utilities
- aria_chk
- aria_pack
- aria_read_log

Backup, Restore and Import Clients
- mariadb-dump
- mariadb-hotcopy
- mariadb-import
- dbForge Studio

Deployment Tools
- mariadb-install-db
- mariadb-upgrade
- mariadb-secure-installation

Graphical and Enhanced Clients
- Adminer
- AdminNeo
- Beekeeper Studio
- SQL Diagnostic Manager & SQLyog
- Database Workbench
- dbForge Data Compare
- dbForge Data Generator
- dbForge Documenter
- dbForge Edge
- dbForge Fusion
- dbForge Query Builder
- dbForge Schema Compare
- dbForge Studio
- DbSchema
- DbVisualizer
- DeZign for Databases
- ERBuilder Data Modeler
- DBeaver
- OmniDB
- Sequel Pro
- HeidiSQL
- KS DB Merge Tools
- LibreOffice Base
- Luna Modeler
- mycli
- Navicat
- ocelotgui
- PgManage
- phpMyAdmin
- Querious
- SB Data Generator
- SQLPro Studio
- SQLyog
- TablePlus
- TOAD Edge
- Valentina Studio
- MariaDB Direct Query Adapter For Microsoft Power BI

Logging Tools
- mariadb-binlog
  - Using mariadb-binlog
  - mariadb-binlog Options
  - Annotate_rows_log_event
  - mysqlbinlog
- mariadb-dumpslow

mariadb Client
- mariadb Command-Line Client
- mysql Command-Line Client

Networking Tools
- resolveip

Table Tools
- mariadb-check
- mariadb-convert-table-format
- mariadb-fix-extensions

Testing Tools
- mariadb-slap
- mariadb-stress-test
- mariadb-test
  - mariadb-test Overview
  - Installing MinIO for Usage With mariadb-test-run
  - mariadb-test Auxiliary Files
  - mariadb-test-run.pl Options
  - New Features for mysqltest in MariaDB
  - Pausing mariadb-test-run.pl
  - The Debug Sync Facility

MyISAM Clients and Utilities
- Memory and Disk Use With myisamchk
- MyISAM Database Management using GUI Client
- myisam_ftdump
- myisamchk Table Information
- myisamchk
- myisamlog
- myisampack

Server & Client Software
- About MariaDB Software
- Applications Supporting MariaDB
- About the MariaDB Foundation
- Client Libraries
  - Connect and Query
  - Proxy Protocol Support
  - Client/Server Protocol
- Downloads
  - MariaDB Source Code
  - MariaDB RPM Packages
  - Mirror Sites for MariaDB
  - Where to Download MariaDB

Analyzing Tools
- EXPLAIN Analyzer
- EXPLAIN Analyzer API

HA & Performance

Optimization and Tuning
- Buffers, Caches and Threads
  - Query Cache
  - Thread Command Values
  - Thread Pool
    - Thread Pool in MariaDB
    - Thread Groups in the Unix Implementation of the Thread Pool
    - Thread Pool System and Status Variables
    - Thread Pool in MariaDB 5.1 - 5.3
  - Thread States
    - Delayed Insert Connection Thread States
    - Delayed Insert Handler Thread States
    - Event Scheduler Thread States
    - Galera Cluster Thread States

- General Thread States
- Master Thread States
- Query Cache Thread States
- Replica I/O Thread States
- Replica Connection Thread States
- Replica SQL Thread States

MariaDB Internal Optimizations
- Fair Choice Between Range and Index_merge Optimizations
- Multi Range Read Optimization

Operating System Optimizations
- Filesystem Optimizations
- Storage I/O: Buffering and Persistence

Optimization and Indexes
- Building the best INDEX for a given SELECT
- Compound (Composite) Indexes
- Foreign Keys
- Ignored Indexes
- Index Statistics
- Latitude/Longitude Indexing
- Primary Keys with Nullable Columns
- Storage Engine Index Types
- Full-Text Indexes
  - Full-Text Index Overview
  - Full-Text Index Stopwords

Optimizer Hints
- SELECT Modifier Hints
- Expanded New-Style Optimizer Hints
- Index-Level Hints
- Query Block Naming
- Table-Level Hints

Compression
- Compression Plugins
- Storage-Engine Independent Column Compression

Optimizing Data Structure
- Numeric vs String Fields
- Optimizing MEMORY Tables
- Optimizing String and Character Fields

Optimizing Tables
- OPTIMIZE TABLE
- Defragmenting InnoDB Tablespaces
- Entity-Attribute-Value Implementation
- IP Range Table Performance

Optimizing Queries
- Aborting Statements that Exceed a Certain Time to Execute
- Big DELETEs
- Charset Narrowing Optimization
- Data Sampling: Techniques for Efficiently Finding a Random Row
- Data Warehousing High Speed Ingestion
- Data Warehousing Summary Tables
- Data Warehousing Techniques
- DISTINCT removal in aggregate functions
- Equality propagation optimization
- Filesort with Small LIMIT Optimization
- FORCE INDEX
- Groupwise Max in MariaDB
- GUID/UUID Performance
- hash_join_cardinality optimizer_switch Flag
- How to Quickly Insert Data Into MariaDB
- IGNORE INDEX
- Index Condition Pushdown
- Index Hints: How to Force Query Plans
- index_merge sort_intersection
- LIMIT ROWS EXAMINED
- MariaDB 5.3 Optimizer Debugging
- not_null_range_scan Optimization
- optimizer_switch
- optimizer_adjust_secondary_key_costs
- optimizer_join_limit_pref_ratio Optimization
- Optimizing for "Latest News"-style Queries
- Pagination Optimization
- Pivoting in MariaDB
- Query Limits and Timeouts
- reorder_outer_joins
- Rollup Unique User Counts
- Rowid Filtering Optimization
- Sargable DATE and YEAR
- Sargable UPPER
- USE INDEX
- Virtual Column Support in the Optimizer
- Optimization Strategies
  - Duplicate Weedout Strategy
  - FirstMatch Strategy
  - Improvements to ORDER BY Optimization
  - LooseScan Strategy
  - Semi-join Materialization Strategy
- Optimizations for Derived Tables
  - Condition Pushdown into Derived Table Optimization
  - Derived Table Merge Optimization
  - Derived Table with Key Optimization
  - Lateral Derived Optimization
  - Split Materialized Optimization
- Statistics for Optimizing Queries
  - Engine-Independent Table Statistics
  - Histogram-Based Statistics
  - InnoDB Persistent Statistics
  - Slow Query Log Extended Statistics
  - User Statistics
- Subquery Optimizations
  - Condition Pushdown Into IN subqueries
  - Conversion of Big IN Predicates Into Subqueries
  - EXISTS-to-IN Optimization
  - Non-semi-join Subquery Optimizations
  - Optimizing GROUP BY and DISTINCT Clauses in Subqueries
  - Semi-join Subquery Optimizations
  - Subquery Cache
  - Subquery Optimizations Map
  - Table Pullout Optimization
- Table Elimination
  - Table Elimination External Resources
  - Table Elimination in MariaDB
  - Table Elimination in Other Databases
  - Table Elimination User Interface
  - What is Table Elimination?

Query Optimizer
- Block-Based Join Algorithms
- Condition Selectivity Computation Internals
- Extended Keys
- MIN/MAX optimization
- Notes When an Index Cannot Be Used
- Optimizer Debugging With GDB
- Optimizer Development
- optimizer_max_sel_arg_weight
- Range Optimizer
- The Optimizer Cost Model from MariaDB 11.0
- Optimizer Trace
  - Basic Optimizer Trace Example
  - How to Collect Large Optimizer Traces
  - Optimizer Trace for Developers

- Optimizer Trace Guide
- Optimizer Trace Overview
- Optimizer Trace Resources
- System and Status Variables
  - Big Query Settings
  - Handling Too Many Connections
  - MariaDB Optimization for MySQL Users
  - Optimizing key_buffer_size
  - Optimizing table_open_cache
  - OQGRAPH System and Status Variables
  - Sample my.cnf Files
  - Segmented Key Cache
  - Semisynchronous Replication Plugin Status Variables
  - InnoDB Server Status Variables
  - Setting Innodb Buffer Pool Size Dynamically
  - Sphinx Status Variables
  - Spider Status Variables
  - SQL Error Log System Variables and Options
  - SSL/TLS Status Variables
  - System and Status Variables Added By Major Release
    - Community Server
      - System Variables Added in MariaDB 12.3
      - Status Variables Added in MariaDB 12.3
      - System Variables Added in MariaDB 12.2
      - Status Variables Added in MariaDB 12.2
      - System Variables Added in MariaDB 12.1
      - Status Variables Added in MariaDB 12.1
      - System Variables Added in MariaDB 12.0
      - System Variables Added in MariaDB 11.8
      - Status Variables Added in MariaDB 11.8
      - System Variables Added in MariaDB 11.4
      - Status Variables Added in MariaDB 11.4
      - System Variables Added in MariaDB 10.11
      - System Variables Added in MariaDB 10.6
      - Status Variables Added in MariaDB 10.6
    - Enterprise Server
      - System Variables Added in Enterprise Server 12.3
      - Status Variables Added in Enterprise Server 12.3
      - System Variables Added in Enterprise Server 11.8
      - Status Variables Added in Enterprise Server 11.8
      - System Variables Added in Enterprise Server 11.4
    - System and Status Variables Added By Major Unmaintained Release
      - System Variables Added in MariaDB 11.7
      - System Variables Added in MariaDB 11.6
      - System Variables Added in MariaDB 11.5
      - Status Variables Added in MariaDB 11.5
      - System Variables Added in MariaDB 11.3
      - System Variables Added in MariaDB 11.2
      - System Variables Added in MariaDB 11.1
      - System Variables Added in MariaDB 11.0
      - Status Variables Added in MariaDB 11.0
      - System Variables Added in MariaDB 10.10
      - System Variables Added in MariaDB 10.9
      - System Variables Added in MariaDB 10.8
      - System Variables Added in MariaDB 10.5
      - Status Variables Added in MariaDB 10.5
      - System Variables Added in MariaDB 10.4
      - Status Variables Added in MariaDB 10.4
      - System Variables Added in MariaDB 10.3
      - Status Variables Added in MariaDB 10.3
      - System Variables Added in MariaDB 10.2
      - Status Variables Added in MariaDB 10.2
      - System Variables Added in MariaDB 10.1
      - Status Variables Added in MariaDB 10.1
      - System Variables Added in MariaDB 10.0
      - Status Variables Added in MariaDB 10.0
      - System Variables Added in MariaDB 5.5
- Replication
  - Replication Overview
  - Choosing a Replication Strategy
  - Replication Statements
  - Setting Up Replication
  - Global Transaction ID
  - Read-Only Replicas
  - Multi-Source Replication
  - Multi-Master Ring Replication
  - Delayed Replication
  - Parallel Replication
  - Semisynchronous Replication
  - Row-based Replication With No Primary Key
  - Unsafe Statements for Statement-Based Replication
  - Replication Filters
  - Replication Threads
  - Replication and Foreign Keys
  - Replication and Binary Log System Variables
  - Replication and Binary Log Status Variables
  - Binlog Event Checksum Interoperability
  - Binlog Event Checksums
  - Changing a Replica to Become the Primary
  - Replication When the Primary and Replica Have Different Table Definitions
  - START TRANSACTION ... WITH CONSISTENT SNAPSHOT
  - Restricting Speed of Reading Binlog from Primary by a Replica
  - Running Triggers on the Replica for Row-based Events
  - Conflict Detection and Resolution (CDR) Triggers
- Benchmarking
- MariaDB Memory Allocation
- Hardware Optimization
- Connection Redirection Mechanism in the MariaDB Client/Server Protocol

Reference

- SQL Structure
  - SQL Language Structure
    - Binary Literals
    - Date and Time Literals
    - Hexadecimal Literals
    - Identifier Case Sensitivity
    - Identifier Names
    - Identifier Qualifiers
    - Identifier to File Name Mapping
    - Numeric Literals
    - Reserved Words
    - Boolean Literals
    - Schema Qualifiers
    - String Literals
    - Table Value Constructors
    - User-Defined Variables
  - Joins, Subqueries, and Set
  - Geometry
    - Geometry Hierarchy
    - Geometry Types
    - GIS Features
    - GIS Resources
    - SPATIAL INDEX
  - NoSQL
    - Dynamic Columns
    - Dynamic Column API
    - HANDLER

- HANDLER
- HANDLER for MEMORY Tables
- HandlerSocket
  - HandlerSocket Installation
  - HandlerSocket Client Libraries
  - HandlerSocket Configuration Options
  - HandlerSocket External Resources
- Operators
  - Operator Precedence
  - Arithmetic Operators
    - Addition Operator (+)
    - Subtraction Operator (-)
    - Multiplication Operator (*)
    - Division Operator (/)
    - Modulo Operator (%)
  - Assignment Operators
    - Assignment Operator (:=)
    - Assignment Operator (=)
  - Comparison Operators
    - BETWEEN AND
    - COALESCE
    - =
    - >=
    - >
    - GREATEST
    - IN
    - INTERVAL
    - IS JSON
    - IS NOT NULL
    - IS NOT
    - IS NULL
    - IS
    - ISNULL
    - LEAST
    - <=
    - <
    - NOT BETWEEN
    - !=
    - NOT IN
    - <=>
  - Logical Operators
    - &&
    - !
    - ||
    - XOR
- Sequences
  - Sequence Overview
  - ALTER SEQUENCE
  - CREATE SEQUENCE
  - DROP SEQUENCE
  - SEQUENCE Functions
    - LASTVAL
    - NEXT VALUE FOR
    - NEXTVAL
    - PREVIOUS VALUE FOR
    - SETVAL
- Temporal Tables
  - Application-Time Periods
  - Bitemporal Tables
  - System-Versioned Tables
- Vectors
  - Vector Overview
  - CREATE TABLE with Vectors
  - Vector System Variables
  - Vector Framework Integrations
  - Vector Functions
  - VECTOR
  - Optimizing Hybrid Search Query with Reciprocal Rank Fusion (RRF)
- SQL Statements
  - Comment Syntax
  - Account Management
    - ALTER USER
    - CREATE ROLE
    - CREATE USER
    - DENY
    - DROP ROLE
    - DROP USER
    - GRANT
    - RENAME USER
    - REVOKE
    - SET DEFAULT ROLE
    - SET PASSWORD
    - SET SESSION AUTHORIZATION
    - SET ROLE
  - Administrative Statements
    - BINLOG
    - CACHE INDEX
    - DESCRIBE
    - HELP Command
    - KILL
    - PURGE BINARY LOGS
    - RESET
    - SHUTDOWN
    - USE [DATABASE]
    - ANALYZE and EXPLAIN Statements
      - ANALYZE FORMAT=JSON
      - ANALYZE FORMAT=JSON Examples
      - ANALYZE: Interpreting rows and filtered members
      - ANALYZE Statement
      - EXPLAIN ANALYZE
      - EXPLAIN FORMAT=JSON
      - EXPLAIN
      - Using Buffer UPDATE Algorithm
    - BACKUP Statements
      - BACKUP LOCK
      - BACKUP STAGE
      - Storage Snapshots and BACKUP STAGE Statements
    - FLUSH Statements
      - FLUSH
      - FLUSH QUERY CACHE
      - FLUSH TABLES FOR EXPORT
    - Plugin Statements
      - INSTALL PLUGIN
      - INSTALL SONAME
      - UNINSTALL PLUGIN
      - UNINSTALL SONAME
    - Replication Statements
      - CHANGE MASTER TO
      - RESET MASTER
      - RESET REPLICA
      - SET GLOBAL SQL_SLAVE_SKIP_COUNTER
      - START REPLICA
      - STOP REPLICA
    - SET Statements
      - SET
      - SET CHARACTER SET
      - SET NAMES
      - SET PATH

- SET SQL_LOG_BIN
- SET STATEMENT
- SET TRANSACTION
- SET Variable

SHOW Statements
- About SHOW
- Extended SHOW
- SHOW ANALYZE
- SHOW AUTHORS
- SHOW BINARY LOGS
- SHOW BINLOG EVENTS
- SHOW MASTER STATUS
- SHOW CHARACTER SET
- SHOW CLIENT_STATISTICS
- SHOW COLLATION
- SHOW COLUMNS
- SHOW CONTRIBUTORS
- SHOW CREATE DATABASE
- SHOW CREATE EVENT
- SHOW CREATE FUNCTION
- SHOW CREATE PACKAGE BODY
- SHOW CREATE PACKAGE
- SHOW CREATE PROCEDURE
- SHOW CREATE SEQUENCE
- SHOW CREATE SERVER
- SHOW CREATE TABLE
- SHOW CREATE TRIGGER
- SHOW CREATE USER
- SHOW CREATE VIEW
- SHOW DATABASES
- SHOW ENGINE INNODB STATUS
- SHOW ENGINE
- SHOW ENGINES
- SHOW ERRORS
- SHOW EVENTS
- SHOW EXPLAIN
- SHOW FUNCTION CODE
- SHOW FUNCTION STATUS
- SHOW GRANTS
- SHOW INDEX_STATISTICS
- SHOW INDEX
- SHOW LOCALES
- SHOW OPEN TABLES
- SHOW PACKAGE BODY STATUS
- SHOW PACKAGE STATUS
- SHOW PLUGINS SONAME
- SHOW PLUGINS
- SHOW PRIVILEGES
- SHOW PROCEDURE CODE
- SHOW PROCEDURE STATUS
- SHOW PROCESSLIST
- SHOW PROFILE
- SHOW PROFILES
- SHOW QUERY_RESPONSE_TIME
- SHOW RELAYLOG EVENTS
- SHOW REPLICA HOSTS
- SHOW REPLICA STATUS
- SHOW STATUS
- SHOW TABLE_STATISTICS
- SHOW TABLE STATUS
- SHOW TABLES
- SHOW TRIGGERS
- SHOW USER_STATISTICS
- SHOW USER_VARIABLES
- SHOW VARIABLES
- SHOW WARNINGS
- SHOW WSREP_MEMBERSHIP
- SHOW WSREP_STATUS

Data Definition (DDL)
- Atomic DDL
- CONSTRAINT
- RENAME TABLE
- Renaming Databases
- ALTER
  - ALTER DATABASE
  - ALTER FUNCTION
  - ALTER LOGFILE GROUP
  - ALTER SERVER
    - Online Schema Change
  - ALTER TABLESPACE
- CREATE
  - CREATE DATABASE
  - CREATE EVENT
  - CREATE FUNCTION
  - CREATE INDEX
  - CREATE LOGFILE GROUP
  - CREATE PACKAGE
  - CREATE PACKAGE BODY
  - CREATE SERVER
  - CREATE TABLESPACE
  - Generated Columns
  - Invisible Columns
  - Silent Column Changes
- DROP
  - DROP DATABASE
  - DROP EVENT
  - DROP INDEX
  - DROP LOGFILE GROUP
  - DROP PACKAGE BODY
  - DROP PACKAGE
  - DROP SERVER
  - DROP TABLESPACE
  - DROP TRIGGER

Data Manipulation (DML)
- Changing & Deleting Data
  - DELETE
  - HIGH_PRIORITY and LOW_PRIORITY
  - REPLACE
  - REPLACE...RETURNING
  - UPDATE
- Inserting & Loading Data
  - Concurrent Inserts
  - IGNORE
  - INSERT
  - INSERT - Default & Duplicate Values
  - INSERT DELAYED
  - INSERT IGNORE
  - INSERT ON DUPLICATE KEY UPDATE
  - INSERT SELECT
  - INSERT...RETURNING
  - LOAD Data into Tables or Index
    - LOAD DATA INFILE
    - LOAD INDEX
    - LOAD XML
- Selecting Data
  - DUAL
  - FOR UPDATE
  - GROUP BY

- LIMIT
- LOCK IN SHARE MODE
- ORDER BY
- PROCEDURE
- SELECT
- SELECT INTO DUMPFILE
- SELECT INTO OUTFILE
- SELECT ... OFFSET ... FETCH
- SELECT WITH ROLLUP
- Common Table Expressions (CTE)
  - Recursive Common Table Expressions Overview
  - Non-Recursive Common Table Expressions Overview
  - WITH
- Joins
  - JOIN Syntax
  - Comma vs JOIN
- Set Operations
  - Precedence Control in Table Operations
  - EXCEPT
  - INTERSECT
  - MINUS
  - UNION
- Subqueries
  - Subqueries With ALL
  - Subqueries With ANY and SOME
  - Subqueries with IN and NOT IN
  - Subqueries With EXISTS
  - Subqueries With JOINs
  - Subqueries in a FROM Clause (Derived Tables)
  - Row Subqueries
  - Scalar Subqueries
  - Subquery Limitations

Geometry Statements
- Geometry Constructors
  - BUFFER
  - CONVEXHULL
  - GEOMETRYCOLLECTION
  - LINESTRING
  - MULTILINESTRING
  - MULTIPOINT
  - MULTIPOLYGON
  - POINT
  - PointOnSurface
  - POLYGON
  - ST_BUFFER
  - ST_CONVEXHULL
  - ST_INTERSECTION
  - ST_POINTONSURFACE
  - ST_SYMDIFFERENCE
  - ST_UNION
  - ST_AsGeoJSON
  - ST_GeomFromGeoJSON
- Geometry Properties
  - BOUNDARY
  - DIMENSION
  - ENVELOPE
  - GeometryN
  - GeometryType
  - IsClosed
  - IsEmpty
  - IsRing
  - IsSimple
  - NumGeometries
  - SRID
  - ST_BOUNDARY
  - ST_DIMENSION
  - ST_ENVELOPE
  - ST_GEOMETRYN
  - ST_GEOMETRYTYPE
  - ST_ISCLOSED
  - ST_ISEMPTY
  - ST_IsRing
  - ST_IsSimple
  - ST_NUMGEOMETRIES
  - ST_RELATE
  - ST_SRID
- Geometry Relations
  - CONTAINS
  - CROSSES
  - DISJOINT
  - EQUALS
  - INTERSECTS
  - OVERLAPS
  - ST_CONTAINS
  - ST_CROSSES
  - ST_DIFFERENCE
  - ST_DISJOINT
  - ST_DISTANCE
  - ST_DISTANCE_SPHERE
  - ST_EQUALS
  - ST_INTERSECTS
  - ST_LENGTH
  - ST_OVERLAPS
  - ST_TOUCHES
  - ST_WITHIN
  - TOUCHES
  - WITHIN
- LineString Properties
  - ENDPOINT
  - GLENGTH
  - NumPoints
  - PointN
  - STARTPOINT
  - ST_ENDPOINT
  - ST_NUMPOINTS
  - ST_POINTN
  - ST_STARTPOINT
- MBR (Minimum Bounding Rectangle)
  - MBR Definition
  - MBRContains
  - MBRCoveredBy
  - MBRDisjoint
  - MBREqual
  - MBRIntersects
  - MBROverlaps
  - MBRTouches
  - MBRWithin
- Miscellaneous GIS functions
  - ST_Collect
  - ST_GeoHash
  - ST_IsValid
  - ST_LatFromGeoHash
  - ST_LongFromGeoHash
  - ST_PointFromGeoHash
  - ST_Simplify
  - ST_Validate
- Point Properties
  - X

- Y
- ST_X
- ST_Y
- Polygon Properties
  - CENTROID
  - AREA
  - ExteriorRing
  - InteriorRingN
  - NumInteriorRings
  - ST_AREA
  - ST_CENTROID
  - ST_ExteriorRing
  - ST_InteriorRingN
  - ST_NumInteriorRings
- WKB
  - AsBinary
  - AsWKB
  - GeomCollFromWKB
  - GeometryCollectionFromWKB
  - GeometryFromWKB
  - GeomFromWKB
  - LineFromWKB
  - LineStringFromWKB
  - MLineFromWKB
  - MPointFromWKB
  - MPolyFromWKB
  - MultiLineStringFromWKB
  - MultiPointFromWKB
  - MultiPolygonFromWKB
  - PointFromWKB
  - PolyFromWKB
  - PolygonFromWKB
  - ST_AsBinary
  - ST_AsWKB
  - ST_GeomCollFromWKB
  - ST_GeometryCollectionFromWKB
  - ST_GeometryFromWKB
  - ST_GeomFromWKB
  - ST_LineFromWKB
  - ST_LineStringFromWKB
  - ST_MPointFromWKB
  - ST_MPolyFromWKB
  - ST_MultiPointFromWKB
  - ST_MultiPolygonFromWKB
  - ST_PointFromWKB
  - ST_PolyFromWKB
  - ST_PolygonFromWKB
  - Well-Known Binary (WKB) Format
- WKT
  - WKT Definition
  - AsText
  - AsWKT
  - GeomCollFromText
  - GeometryCollectionFromText
  - GeometryFromText
  - GeomFromText
  - LineFromText
  - LineStringFromText
  - MLineFromText
  - MPointFromText
  - MPolyFromText
  - MultiLineStringFromText
  - MultiPointFromText
  - MultiPolygonFromText
  - PointFromText
  - PolyFromText
  - PolygonFromText
  - ST_AsText
  - ST_ASWKT
  - ST_GeomCollFromText
  - ST_GeometryCollectionFromText
  - ST_GeometryFromText
  - ST_GeomFromText
  - ST_LineFromText
  - ST_LineStringFromText
  - ST_MPointFromText
  - ST_MPolyFromText
  - ST_MultiLineStringFromText
  - ST_MultiPointFromText
  - ST_MultiPolygonFromText
  - ST_PointFromText
  - ST_PolyFromText
  - ST_PolygonFromText
  - ST_MLineFromText
- Prepared Statements
  - DEALLOCATE / DROP PREPARE
  - EXECUTE Statement
  - EXECUTE IMMEDIATE
  - PREPARE Statement
- Programmatic & Compound Statements
  - BEGIN END
  - CASE Statement
  - DECLARE CONDITION
  - DECLARE HANDLER
  - DECLARE TYPE
  - DECLARE Variable
  - FOR
  - GOTO
  - IF
  - ITERATE
  - Labels
  - LEAVE
  - LOOP
  - REPEAT LOOP
  - RESIGNAL
  - RETURN
  - SELECT INTO
  - SIGNAL
  - WHILE
  - Using Compound Statements Outside of Stored Programs
  - Cursors
    - Cursor Overview
    - CLOSE
    - DECLARE CURSOR
    - FETCH
    - OPEN
  - Diagnostics
    - Diagnostics Area
    - GET DIAGNOSTICS
    - SQLSTATE
- Stored Routine Statements
  - CALL
  - DO
- Table Statements
  - ANALYZE TABLE
  - CHECK TABLE
  - CHECK VIEW
  - CHECKSUM TABLE

- REPAIR TABLE
- REPAIR VIEW
- TRUNCATE TABLE
- Transactions
  - COMMIT
  - LOCK TABLES
  - Metadata Locking
  - READ COMMITTED
  - READ UNCOMMITTED
  - REPEATABLE READ
  - ROLLBACK
  - SAVEPOINT
  - SERIALIZABLE
  - SQL statements Causing an Implicit Commit
  - START TRANSACTION
  - Transaction Timeouts
  - UNLOCK TABLES
  - WAIT and NOWAIT
  - XA Transactions
- SQL Functions
  - Functions & Operators
  - Aggregate Functions
    - AVG
    - BIT_AND
    - BIT_OR
    - BIT_XOR
    - COUNT DISTINCT
    - COUNT
    - GROUP_CONCAT
    - JSON_ARRAYAGG
    - JSON_OBJECTAGG
    - MAX
    - MIN
    - STD
    - STDDEV
    - STDDEV_POP
    - STDDEV_SAMP
    - SUM
    - VAR_POP
    - VAR_SAMP
    - VARIANCE
  - Control Flow Functions
    - CASE OPERATOR
    - DECODE_ORACLE
    - IF Function
    - IFNULL
    - NULLIF
    - NVL
    - NVL2
  - Date & Time Functions
    - Date and Time Units
    - ADD_MONTHS
    - ADDDATE
    - ADDTIME
    - CONVERT_TZ
    - CURDATE
    - CURRENT_DATE
    - CURRENT_TIME
    - CURRENT_TIMESTAMP
    - CURTIME
    - DATE FUNCTION
    - DATE_ADD
    - DATE_FORMAT
    - DATE_SUB
    - DATEDIFF
    - DAY
    - DAYNAME
    - DAYOFMONTH
    - DAYOFWEEK
    - DAYOFYEAR
    - EXTRACT
    - FORMAT_PICO_TIME
    - FROM_DAYS
    - FROM_UNIXTIME
    - GET_FORMAT
    - HOUR
    - LAST_DAY
    - LOCALTIME
    - LOCALTIMESTAMP
    - MAKEDATE
    - MAKETIME
    - MICROSECOND
    - Microseconds in MariaDB
    - MINUTE
    - MONTH
    - MONTHNAME
    - MONTHS_BETWEEN
    - NOW
    - PERIOD_ADD
    - PERIOD_DIFF
    - QUARTER
    - SEC_TO_TIME
    - SECOND
    - STR_TO_DATE
    - SUBDATE
    - SUBTIME
    - SYSDATE
    - TIME
    - TIME_FORMAT
    - TIME_TO_SEC
    - TIMEDIFF
    - TIMESTAMP
    - TIMESTAMPADD
    - TIMESTAMPDIFF
    - TO_DATE
    - TO_DAYS
    - TO_SECONDS
    - TRUNC
    - UNIX_TIMESTAMP
    - UTC_DATE
    - UTC_TIME
    - UTC_TIMESTAMP
    - WEEK
    - WEEKDAY
    - WEEKOFYEAR
    - YEAR
    - YEARWEEK
  - Numeric Functions
    - ABS
    - ACOS
    - ASIN
    - ATAN
    - ATAN2
    - CEIL
    - CEILING
    - CONV
    - COS
    - COT

- CRC32
- CRC32C
- DEGREES
- DIV
- EXP
- FLOOR
- LN
- LOG
- LOG10
- LOG2
- MOD
- OCT
- PI
- POW
- POWER
- RADIANS
- RAND
- ROUND
- SIGN
- SIN
- SQRT
- TAN
- TO_NUMBER
- TRUNCATE
- XXH3
- XXH32

Pseudo Columns

- _rowid

Secondary Functions

- Bit Functions and Operators
  - BIT_COUNT
  - ~
  - |
  - ^
  - &
  - Parentheses
  - <<
  - >>
  - TRUE FALSE
- Encryption, Hashing and Compression Functions
  - AES_DECRYPT
  - AES_ENCRYPT
  - COMPRESS
  - DECODE
  - DES_DECRYPT
  - DES_ENCRYPT
  - ENCODE
  - ENCRYPT
  - KDF
  - MD5
  - OLD_PASSWORD
  - PASSWORD
  - RANDOM_BYTES
  - SHA1
  - SHA2
  - UNCOMPRESS
  - UNCOMPRESSED_LENGTH
- Information Functions
  - BENCHMARK
  - BINLOG_GTID_POS
  - CHARSET
  - COERCIBILITY
  - COLLATION
  - CONNECTION_ID
  - CURRENT_ROLE
  - CURRENT_USER
  - DATABASE
  - DECODE_HISTOGRAM
  - DEFAULT
  - FOUND_ROWS
  - LAST_INSERT_ID
  - LAST_VALUE
  - PROCEDURE ANALYSE
  - ROW_COUNT
  - ROWNUM
  - SCHEMA
  - SESSION_USER
  - SYSTEM_USER
  - USER
  - VERSION
- Miscellaneous Functions
  - FORMAT_BYTES
  - GET_LOCK
  - INET6_ATON
  - INET6_NTOA
  - INET_ATON
  - INET_NTOA
  - IS_FREE_LOCK
  - IS_IPV4
  - IS_IPV4_COMPAT
  - IS_IPV4_MAPPED
  - IS_IPV6
  - IS_USED_LOCK
  - MASTER_GTID_WAIT
  - MASTER_POS_WAIT
  - NAME_CONST
  - OLD_VALUE
  - RELEASE_ALL_LOCKS
  - RELEASE_LOCK
  - SLEEP
  - SYS_GUID
  - UUID
  - UUID_SHORT
  - UUID_v4
  - UUID_v7
  - VALUES / VALUE

Special Functions

- Dynamic Columns Functions
  - COLUMN_ADD
  - COLUMN_CHECK
  - COLUMN_CREATE
  - COLUMN_DELETE
  - COLUMN_EXISTS
  - COLUMN_GET
  - COLUMN_JSON
  - COLUMN_LIST
- Galera Functions
  - WSREP_LAST_SEEN_GTID
  - WSREP_LAST_WRITTEN_GTID
  - WSREP_SYNC_WAIT_UPTO_GTID
- Geographic Functions
- JSON Functions
  - JSONPath Expressions
  - Differences between JSON_QUERY and JSON_VALUE
  - JSON Arrow Operators
  - JSON_ARRAY
  - JSON_ARRAY_APPEND
  - JSON_ARRAY_INSERT

- JSON_ARRAY_INTERSECT
- JSON_COMPACT
- JSON_CONTAINS
- JSON_CONTAINS_PATH
- JSON_DEPTH
- JSON_DETAILED
- JSON_EQUALS
- JSON_EXISTS
- JSON_EXTRACT
- JSON_INSERT
- JSON_KEY_VALUE
- JSON_KEYS
- JSON_LENGTH
- JSON_LOOSE
- JSON_MERGE
- JSON_MERGE_PATCH
- JSON_MERGE_PRESERVE
- JSON_NORMALIZE
- JSON_OBJECT
- JSON_OBJECT_FILTER_KEYS
- JSON_OBJECT_TO_ARRAY
- JSON_OVERLAPS
- JSON_PRETTY
- JSON_QUERY
- JSON_QUOTE
- JSON_REMOVE
- JSON_REPLACE
- JSON_SCHEMA_VALID
- JSON_SEARCH
- JSON_SET
- JSON_TABLE
- JSON_TYPE
- JSON_UNQUOTE
- JSON_VALID
- JSON_VALUE
- JSON Validation Function

- Window Functions
  - Window Functions Overview
  - Window Frames
  - ColumnStore Window Functions
  - CUME_DIST
  - DENSE_RANK
  - FIRST_VALUE
  - LAG
  - LEAD
  - MEDIAN
  - NTH_VALUE
  - NTILE
  - PERCENT_RANK
  - PERCENTILE_CONT
  - PERCENTILE_DISC
  - RANK
  - ROW_NUMBER

- String Functions
  - ASCII
  - BIN
  - BINARY Operator
  - BIT_LENGTH
  - CAST
  - CHAR Function
  - CHAR_LENGTH
  - CHARACTER_LENGTH
  - CHR
  - CONCAT
  - CONCAT_WS
  - CONVERT
  - ELT
  - EXPORT_SET
  - EXTRACTVALUE
  - FIELD
  - FIND_IN_SET
  - FORMAT
  - FROM_BASE64
  - HEX
  - INSERT Function
  - INSTR
  - LCASE
  - LEFT
  - LENGTH
  - LENGTHB
  - LIKE
  - LOAD_FILE
  - LOCATE
  - LOWER
  - LPAD
  - LTRIM
  - MAKE_SET
  - MATCH AGAINST
  - MID
  - NATURAL_SORT_KEY
  - NOT LIKE
  - NOT REGEXP
  - OCTET_LENGTH
  - ORD
  - POSITION
  - QUOTE
  - REPEAT Function
  - REPLACE Function
  - REVERSE
  - RIGHT
  - RPAD
  - RTRIM
  - SFORMAT
  - SOUNDEX
  - SOUNDS LIKE
  - SPACE
  - STRCMP
  - SUBSTR
  - SUBSTRING
  - SUBSTRING_INDEX
  - TO_BASE64
  - TO_CHAR
  - TRIM
  - TRIM_ORACLE
  - Type Conversion
  - UCASE
  - UNHEX
  - UPDATEXML
  - UPPER
  - WEIGHT_STRING
  - Regular Expressions Functions
    - Regular Expressions Overview
    - PCRE - Perl Compatible Regular Expressions
    - REGEXP
    - REGEXP_INSTR
    - REGEXP_REPLACE
    - REGEXP_SUBSTR
    - RLIKE

- VEC_DISTANCE
- VEC_DISTANCE_COSINE
- VEC_DISTANCE_EUCLIDEAN
- VEC_FromText
- VEC_ToText
- Data Type Storage Requirements
- AUTO_INCREMENT
- AUTO_INCREMENT FAQ
- NULL Values
- ROW TYPE OF
- SERIAL
- TYPE OF
- Numeric Data Types
  - Numeric Data Type Overview
  - BIGINT
  - BIT
  - BOOL
  - BOOLEAN
  - DEC, NUMERIC, FIXED
  - DEC
  - DECIMAL
  - DOUBLE PRECISION
  - DOUBLE
  - FIXED
  - FLOAT
  - FLOAT4
  - FLOAT8
  - Floating-point Accuracy
  - INT
  - INT1
  - INT2
  - INT3
  - INT4
  - INT8
  - INTEGER
  - MEDIUMINT
  - MIDDLEINT
  - NUMBER
  - NUMERIC
  - REAL
  - SMALLINT
  - TINYINT
- Date and Time Data Types
  - DATE
  - DATETIME
  - SQL_TSI_YEAR
  - TIME
  - TIMESTAMP
  - YEAR Data Type
- String Data Types
  - BLOB and TEXT Data Types
  - BINARY
  - BLOB
  - CHAR
  - CHARACTER
  - CHAR BYTE
  - CHAR VARYING
  - CLOB
  - ENUM
  - INET4
  - INET6
  - JSON Data Type
  - LONG and LONG VARCHAR
  - LONG CHAR VARYING
  - LONG CHARACTER VARYING
  - LONG VARBINARY
  - LONG VARCHAR
  - LONG VARCHARACTER
  - LONGBLOB
  - LONGTEXT
  - MEDIUMBLOB
  - MEDIUMTEXT
  - NATIONAL CHAR
  - NATIONAL CHAR VARYING
  - NATIONAL CHARACTER VARYING
  - NATIONAL CHARACTER
  - NATIONAL VARCHAR
  - NATIONAL VARCHARACTER
  - NCHAR
  - NCHAR VARCHAR
  - NCHAR VARCHARACTER
  - NCHAR VARYING
  - RAW
  - ROW
  - SET Data Type
  - TEXT
  - TINYBLOB
  - TINYTEXT
  - UUID Data Type
  - VARBINARY
  - VARCHAR
  - VARCHAR2
  - VARCHARACTER
  - XMLTYPE
  - Character Sets and Collations
    - Character Set and Collation Overview
    - Setting Character Sets and Collations
    - Supported Character Sets and Collations
    - Unicode
    - Internationalization and Localization
      - Coordinated Universal Time
      - Locales Plugin
      - Server Locale
      - Setting the Language for Error Messages
      - Time Zones
- Plugins
  - Plugin Overview
  - Plugin Maturity
  - MariaDB Enterprise Audit
  - MariaDB Community Audit Plugin
    - Audit Plugin Overview
    - Audit Plugin Installation
    - Audit Plugin Configuration
    - Audit Plugin Log Settings
    - Audit Plugin Location and Rotation of Logs
    - Audit Plugin Log Format
    - Audit Plugin Options and System Variables
    - Audit Plugin Status Variables
    - Audit Plugin Versions
  - Authentication Plugins
    - Pluggable Authentication Overview
    - Authentication Plugin - caching_sha2_password
    - Authentication Plugin - ed25519
    - Authentication Plugin - GSSAPI
    - Authentication Plugin - mysql_native_password
    - Authentication Plugin - mysql_old_password
    - Authentication Plugin - Named Pipe
    - Authentication Plugin - PARSEC

- Authentication Plugin - SHA-256
- Authentication Plugin - Unix Socket
- Authentication with Pluggable Authentication Modules (PAM)
  - Authentication Plugin - PAM
  - Configuring PAM Authentication and User Mapping with LDAP Authentication
  - Configuring PAM Authentication and User Mapping with Unix Authentication
  - User and Group Mapping with PAM
  - Authentication with JWT, OIDC, and OAuth 2.0 via PAM
- MariaDB Replication & Cluster Plugins
  - WSREP_INFO Plugin
  - wsrep_provider
- Other Plugins
  - Disks Plugin
  - Feedback Plugin
  - inet4
  - METADATA_LOCK_INFO Plugin
  - MYSQL_JSON
  - mhnsw
  - online_alter_log
  - Query Cache Information Plugin
  - Query Response Time Plugin
  - User Variables Plugin
- Password Validation Plugins
  - Password Validation Plugin Overview
  - Cracklib Password Check Plugin
  - Password Reuse Check Plugin
  - password_reuse_check_interval Variable
  - Simple Password Check Plugin

System Tables

- Information Schema
  - TIME_MS column in INFORMATION_SCHEMA.PROCESSLIST
  - Information Schema Tables
    - Information Schema ALL_PLUGINS Table
    - Information Schema APPLICABLE_ROLES Table
    - Information Schema CATALOG Table
    - Information Schema CHARACTER_SETS Table
    - Information Schema CHECK_CONSTRAINTS Table
    - Information Schema CLIENT_STATISTICS Table
    - Information Schema COLLATION_CHARACTER_SET_APPLICABILITY Table
    - Information Schema COLLATIONS Table
    - Information Schema COLUMN_PRIVILEGES Table
    - Information Schema COLUMNS Table
    - Information Schema DISKS Table
    - Information Schema ENABLED_ROLES Table
    - Information Schema ENGINES Table
    - Information Schema EVENTS Table
    - Information Schema FEEDBACK Table
    - Information Schema FILE_KEY_MANAGEMENT_KEYS Table
    - Information Schema HASHICORP_KEY_MANAGEMENT_CACHE Table
    - Information Schema GEOMETRY_COLUMNS Table
    - Information Schema GLOBAL_STATUS and SESSION_STATUS Tables
    - Information Schema GLOBAL_VARIABLES and SESSION_VARIABLES Tables
    - Information Schema INDEX_STATISTICS Table
    - Information Schema KEY_CACHES Table
    - Information Schema KEY_COLUMN_USAGE Table
    - Information Schema KEY_PERIOD_USAGE Table
    - Information Schema KEYWORDS Table
    - Information Schema LOCALES Table
    - Information Schema METADATA_LOCK_INFO Table
    - Information Schema MROONGA_STATS Table
    - Information Schema OPTIMIZER_TRACE Table
    - Information Schema PARAMETERS Table
    - Information Schema PARTITIONS Table
    - Information Schema PERIODS Table
    - Information Schema PLUGINS Table
    - Information Schema PROCESSLIST Table
    - Information Schema PROFILING Table
    - Information Schema QUERY_CACHE_INFO Table
    - Information Schema QUERY_RESPONSE_TIME Table
    - Information Schema REFERENTIAL_CONSTRAINTS Table
    - Information Schema ROUTINES Table
    - Information Schema SCHEMA_PRIVILEGES Table
    - Information Schema SCHEMATA Table
    - Information Schema SEQUENCES Table
    - Information Schema SLAVE_STATUS Table
    - Information Schema SPATIAL_REF_SYS Table
    - Information Schema SPIDER_ALLOC_MEM Table
    - Information Schema SQL_FUNCTIONS Table
    - Information Schema STATISTICS Table
    - Information Schema SYSTEM_VARIABLES Table
    - Information Schema TABLE_CONSTRAINTS Table
    - Information Schema TABLE_PRIVILEGES Table
    - Information Schema TABLE_STATISTICS Table
    - Information Schema TABLES Table
    - Information Schema TABLESPACES Table
    - Information Schema THREAD_POOL_GROUPS Table
    - Information Schema THREAD_POOL_QUEUES Table
    - Information Schema THREAD_POOL_STATS Table
    - Information Schema THREAD_POOL_WAITS Table
    - Information Schema TRIGGERED_UPDATE_COLUMNS
    - Information Schema TRIGGERS Table
    - Information Schema USER_PRIVILEGES Table
    - Information Schema USER_STATISTICS Table
    - Information Schema USER_VARIABLES Table
    - Information Schema USERS Table
    - Information Schema WSREP_BF_ABORTS
    - Information Schema WSREP_CERT_KEYS
    - Information Schema WSREP_CERT_KEYS_HISTORY
    - Information Schema WSREP_CONNECTIONS
    - Information Schema WSREP_MEMBERSHIP Table
    - Information Schema WSREP_STATUS Table
    - Information Schema WSREP_THD_STATE
    - Information Schema WSREP_THD_STATE_HISTORY
    - Information Schema InnoDB Tables
      - Information Schema INNODB_CMP_PER_INDEX and INNODB_CMP_PER_INDEX_RESET Tables
      - Information Schema INNODB_BUFFER_PAGE Table
      - Information Schema INNODB_BUFFER_PAGE_LRU Table
      - Information Schema INNODB_BUFFER_POOL_PAGES Table
      - Information Schema INNODB_BUFFER_POOL_PAGES_BLOB Table
      - Information Schema INNODB_BUFFER_POOL_PAGES_INDEX Table
      - Information Schema INNODB_BUFFER_POOL_STATS Table
      - Information Schema INNODB_CHANGED_PAGES Table
      - Information Schema INNODB_CMP and INNODB_CMP_RESET Tables
      - Information Schema INNODB_CMPMEM and INNODB_CMPMEM_RESET Tables
      - Information Schema INNODB_FT_BEING_DELETED Table
      - Information Schema INNODB_FT_CONFIG Table
      - Information Schema INNODB_FT_DEFAULT_STOPWORD Table
      - Information Schema INNODB_FT_DELETED Table

- Information Schema INNODB_FT_INDEX_CACHE Table
- Information Schema INNODB_FT_INDEX_TABLE Table
- Information Schema INNODB_LOCK_WAITS Table
- Information Schema INNODB_LOCKS Table
- Information Schema INNODB_METRICS Table
- Information Schema INNODB_MUTEXES Table
- Information Schema INNODB_SYS_COLUMNS Table
- Information Schema INNODB_SYS_DATAFILES Table
- Information Schema INNODB_SYS_FIELDS Table
- Information Schema INNODB_SYS_FOREIGN Table
- Information Schema INNODB_SYS_FOREIGN_COLS Table
- Information Schema INNODB_SYS_INDEXES Table
- Information Schema INNODB_SYS_SEMAPHORE_WAITS Table
- Information Schema INNODB_SYS_TABLES Table
- Information Schema INNODB_SYS_TABLESPACES Table
- Information Schema INNODB_SYS_TABLESTATS Table
- Information Schema INNODB_SYS_VIRTUAL Table
- Information Schema INNODB_TABLESPACES_ENCRYPTION Table
- Information Schema INNODB_TABLESPACES_SCRUBBING Table
- Information Schema INNODB_TRX Table
- Information Schema TEMP_TABLES_INFO Table
- Information Schema INNODB_UNDO_LOGS Table

- Information Schema MyRocks Tables
  - Information Schema ROCKSDB_CF_OPTIONS Table
  - Information Schema ROCKSDB_CFSTATS Table
  - Information Schema ROCKSDB_COMPACTION_STATS Table
  - Information Schema ROCKSDB_DBSTATS Table
  - Information Schema ROCKSDB_DDL Table
  - Information Schema ROCKSDB_DEADLOCK Table
  - Information Schema ROCKSDB_GLOBAL_INFO Table
  - Information Schema ROCKSDB_INDEX_FILE_MAP Table
  - Information Schema ROCKSDB_LOCKS Table
  - Information Schema ROCKSDB_PERF_CONTEXT Table
  - Information Schema ROCKSDB_PERF_CONTEXT_GLOBAL Table
  - Information Schema ROCKSDB_SST_PROPS Table
  - Information Schema ROCKSDB_TRX Table
- Information Schema XtraDB Tables
  - Information Schema CHANGED_PAGE_BITMAPS Table
  - Information Schema XTRADB_INTERNAL_HASH_TABLES Table
  - Information Schema XTRADB_READ_VIEW Table
  - Information Schema XTRADB_RSEG Table

- Performance Schema
  - Performance Schema Digests
  - Performance Schema Overview
  - Performance Schema Status Variables
  - Performance Schema System Variables
  - Performance Schema Tables
    - List of Performance Schema Tables
    - Performance Schema accounts Table
    - Performance Schema cond_instances Table
    - Performance Schema events_stages_current Table
    - Performance Schema events_stages_history Table
    - Performance Schema events_stages_history_long Table
    - Performance Schema events_stages_summary_by_account_by_event_name Table
    - Performance Schema events_stages_summary_by_host_by_event_name Table
    - Performance Schema events_stages_summary_by_thread_by_event_name Table
    - Performance Schema events_stages_summary_by_user_by_event_name Table
    - Performance Schema events_stages_summary_global_by_event_name Table
    - Performance Schema events_statements_current Table
    - Performance Schema events_statements_history Table
    - Performance Schema events_statements_history_long Table
    - Performance Schema events_statements_summary_by_account_by_event_name Table
    - Performance Schema events_statements_summary_by_digest Table
    - Performance Schema events_statements_summary_by_host_by_event_name Table
    - Performance Schema events_statements_summary_by_program Table
    - Performance Schema events_statements_summary_by_thread_by_event_name Table
    - Performance Schema events_statements_summary_by_user_by_event_name Table
    - Performance Schema events_statements_summary_global_by_event_name Table
    - Performance Schema events_transactions_current Table
    - Performance Schema events_transactions_history Table
    - Performance Schema events_transactions_history_long Table
    - Performance Schema events_transactions_summary_by_account_by_event_name Table
    - Performance Schema events_transactions_summary_by_host_by_event_name Table
    - Performance Schema events_transactions_summary_by_thread_by_event_name Table
    - Performance Schema events_transactions_summary_by_user_by_event_name Table
    - Performance Schema events_transactions_summary_global_by_event_name Table
    - Performance Schema events_waits_current Table
    - Performance Schema events_waits_history Table
    - Performance Schema events_waits_history_long Table
    - Performance Schema events_waits_summary_by_account_by_event_name Table
    - Performance Schema events_waits_summary_by_host_by_event_name Table
    - Performance Schema events_waits_summary_by_instance Table
    - Performance Schema events_waits_summary_by_thread_by_event_name Table
    - Performance Schema events_waits_summary_by_user_by_event_name Table
    - Performance Schema events_waits_summary_global_by_event_name Table
    - Performance Schema file_instances Table
    - Performance Schema file_summary_by_event_name Table
    - Performance Schema file_summary_by_instance Table
    - Performance Schema global_status Table
    - Performance Schema host_cache Table
    - Performance Schema hosts Table
    - Performance Schema memory_summary_by_account_by_event_name Table
    - Performance Schema memory_summary_by_host_by_event_name Table
    - Performance Schema memory_summary_by_thread_by_event_name Table
    - Performance Schema memory_summary_by_user_by_event_name Table
    - Performance Schema memory_summary_global_by_event_name Table
    - Performance Schema metadata_locks Table
    - Performance Schema mutex_instances Table

- Performance Schema objects_summary_global_by_type Table
- Performance Schema performance_timers Table
- Performance Schema prepared_statements_instances Table
- Performance Schema replication_applier_configuration Table
- Performance Schema replication_applier_status Table
- Performance Schema replication_applier_status_by_coordinator Table
- Performance Schema replication_applier_status_by_worker Table
- Performance Schema replication_connection_configuration Table
- Performance Schema rwlock_instances Table
- Performance Schema session_account_connect_attrs Table
- Performance Schema session_connect_attrs Table
- Performance Schema session_status Table
- Performance Schema setup_actors Table
- Performance Schema setup_consumers Table
- Performance Schema setup_instruments Table
- Performance Schema setup_objects Table
- Performance Schema setup_timers Table
- Performance Schema socket_instances Table
- Performance Schema socket_summary_by_event_name Table
- Performance Schema socket_summary_by_instance Table
- Performance Schema status_by_account Table
- Performance Schema status_by_host Table
- Performance Schema status_by_thread Table
- Performance Schema status_by_user Table
- Performance Schema table_handles Table
- Performance Schema table_io_waits_summary_by_index_usage Table
- Performance Schema table_io_waits_summary_by_table Table
- Performance Schema table_lock_waits_summary_by_table Table
- Performance Schema threads Table
- Performance Schema users Table
- Performance Schema user_variables_by_thread Table

Sys Schema

- Sys Schema sys_config Table
- Sys Schema Stored Functions
  - extract_schema_from_file_name
  - extract_table_from_file_name
  - format_bytes
  - format_path
  - format_statement
  - format_time
  - list_add
  - list_drop
  - ps_is_account_enabled
  - ps_is_consumer_enabled
  - ps_is_instrument_default_enabled
  - ps_is_instrument_default_timed
  - ps_is_thread_instrumented
  - ps_thread_account
  - ps_thread_id
  - ps_thread_stack
  - ps_thread_trx_info
  - quote_identifier
  - sys_get_config
  - version_major
  - version_minor
  - version_patch
- Sys Schema Stored Procedures
  - create_synonym_db
  - optimizer_switch Helper Functions
  - ps_trace_statement_digest
  - ps_trace_thread
  - ps_truncate_all_tables
  - statement_performance_analyzer
  - table_exists
- Sys Schema Views
  - host_summary and x$host_summary Sys Schema Views
  - host_summary_by_file_io and x$host_summary_by_file_io Sys Schema Views
  - host_summary_by_file_io_type and x$host_summary_by_file_io_type Sys Schema Views
  - host_summary_by_stages and x$host_summary_by_stages Sys Schema Views
  - host_summary_by_statement_type and x$host_summary_by_statement_type Sys Schema Views
  - host_summary_by_statement_latency and x$host_summary_by_statement_latency Sys Schema Views
  - innodb_buffer_stats_by_schema and x$innodb_buffer_stats_by_schema Sys Schema Views
  - innodb_buffer_stats_by_table and x$innodb_buffer_stats_by_table Sys Schema Views
  - innodb_lock_waits and x$innodb_lock_waits Sys Schema Views
  - io_by_thread_by_latency and x$io_by_thread_by_latency Sys Schema Views
  - io_global_by_file_by_bytes and x$io_global_by_file_by_bytes Sys Schema Views
  - io_global_by_file_by_latency and x$io_global_by_file_by_latency Sys Schema Views
  - io_global_by_wait_by_bytes and x$io_global_by_wait_by_bytes Sys Schema Views
  - io_global_by_wait_by_latency and x$io_global_by_wait_by_latency Sys Schema Views
  - latest_file_io and x$latest_file_io Sys Schema Views
  - memory_by_host_by_current_bytes and x$memory_by_host_by_current_bytes Views
  - metrics Sys Schema View
  - privileges_by_table_by_level Sys Schema View
  - schema_auto_increment_columns Sys Schema View
  - schema_object_overview Sys Schema View

The mysql Database Tables

- mysql.column_stats Table
- mysql.columns_priv Table
- mysql.db Table
- mysql.event Table
- mysql.func Table
- mysql.general_log Table
- mysql.global_priv Table
- mysql.gtid_slave_pos Table
- mysql.help_category Table
- mysql.help_keyword Table
- mysql.help_relation Table
- mysql.help_topic Table
- mysql.index_stats Table
- mysql.innodb_index_stats
- mysql.innodb_table_stats
- mysql.password_reuse_check_history Table
- mysql.plugin Table
- mysql.proc Table
- mysql.procs_priv Table
- mysql.proxies_priv Table
- mysql.roles_mapping Table
- mysql.servers Table
- mysql.slow_log Table
- mysql.table_stats Table
- mysql.tables_priv Table
- mysql.time_zone Table
- mysql.time_zone_leap_second Table

- mysql.time_zone_name Table
- mysql.time_zone_transition Table
- mysql.time_zone_transition_type Table
- mysql.transaction_registry Table
- mysql.user Table
- Spider mysql Database Tables
  - mysql.spider_link_failed_log Table
  - mysql.spider_link_mon_servers Table
  - mysql.spider_table_crd Table
  - mysql.spider_table_position_for_recovery Table
  - mysql.spider_table_sts Table
  - mysql.spider_tables Table
  - mysql.spider_xa Table
  - mysql.spider_xa_failed_log Table
  - mysql.spider_xa_member Table

Error Codes
- MariaDB Error Code Reference
- Operating System Error Codes
- MariaDB Error Codes 1000 to 1099
  - Error 1000: hashchk
  - Error 1001: isamchk
  - Error 1002: NO
  - Error 1003: YES
  - Error 1004: Can't create file
  - Error 1005: Can't create table
  - Error 1006: Can't create database
  - Error 1007: Can't create database, database exists
  - Error 1008: Can't drop database; database doesn't exist
  - Error 1009: Error dropping database (can't delete)
  - Error 1010: Error dropping database (can't rmdir)
  - Error 1011: Error on delete of
  - Error 1012: Can't read record in system table
  - Error 1013: Can't get status of
  - Error 1014: Can't get working directory
  - Error 1015: Can't lock file
  - Error 1016: Can't open file
  - Error 1017: Can't find file
  - Error 1018: Can't read dir of
  - Error 1019: Can't change dir to
  - Error 1020: Record has changed since last read in table
  - Error 1021: Disk full (%s); waiting for someone to free some space
  - Error 1022: Can't write; duplicate key in table
  - Error 1023: Error on close of
  - Error 1024: Error reading file
  - Error 1025: Error on rename of
  - Error 1026: Error writing file
  - Error 1027: is locked against change
  - Error 1028: Sort aborted
  - Error 1029: View doesn't exist for
  - Error 1030: Got error from storage engine
  - Error 1031: Table storage engine doesn't have this option
  - Error 1032: Can't find record in
  - Error 1033: Incorrect information in file
  - Error 1034: Incorrect key file for table; try to repair it
  - Error 1035: Old key file for table; repair it!
  - Error 1036: Table is read only
  - Error 1037: Out of memory; restart server and try again
  - Error 1038: Out of sort memory, consider increasing server sort buffer size
  - Error 1039: Unexpected EOF found when reading file
  - Error 1040: Too many connections
  - Error 1041: Out of memory
  - Error 1042: Can't get hostname for your address
  - Error 1043: Bad handshake
  - Error 1044: Access denied for user
  - Error 1045: Access denied for user (using password)
  - Error 1046: No database selected
  - Error 1047: Unknown command
  - Error 1048: Column cannot be null
  - Error 1049: Unknown database
  - Error 1050: Table already exists
  - Error 1051: Unknown table
  - Error 1052: Column is ambiguous
  - Error 1053: Server shutdown in progress
  - Error 1054: Unknown column
  - Error 1055: isn't in GROUP BY
  - Error 1056: Can't group on
  - Error 1057: Statement has sum functions and columns in same statement
  - Error 1058: Column count doesn't match value count
  - Error 1059: Identifier name is too long
  - Error 1060: Duplicate column name
  - Error 1061: Duplicate key name
  - Error 1062: Duplicate entry for key
  - Error 1063: Incorrect column specifier for column
  - Error 1064: near at line
  - Error 1065: Query was empty
  - Error 1066: Not unique table/alias
  - Error 1067: Invalid default value for
  - Error 1068: Multiple primary key defined
  - Error 1069: Too many keys specified; max keys allowed
  - Error 1070: Too many key parts specified; max parts allowed
  - Error 1071: Specified key was too long; max key length is
  - Error 1072: Key column doesn't exist in table
  - Error 1073: BLOB column can't be used in key specification with the used table type
  - Error 1074: Column length too big for column; use BLOB or TEXT instead
  - Error 1075: Incorrect table definition; there can be only one auto column and it must be defined as
  - Error 1076: ready for connections
  - Error 1077: Normal shutdown
  - Error 1078: Got signal. Aborting!
  - Error 1079: Shutdown complete
  - Error 1080: Forcing close of thread
  - Error 1081: Can't create IP socket
  - Error 1082: Table has no index like the one used in CREATE INDEX; recreate the table
  - Error 1083: Field separator argument is not what is expected; check the manual
  - Error 1084: You can't use fixed rowlength with BLOBs; please use 'fields terminated by'
  - Error 1085: The file must be in the database directory or be readable by all
  - Error 1086: File already exists
  - Error 1087: Records Deleted Skipped Warnings
  - Error 1088: Records Duplicates
  - Error 1089: Incorrect prefix key; the used key part isn't a string, the used length is longer than t
  - Error 1090: You can't delete all columns with ALTER TABLE; use DROP TABLE instead
  - Error 1091: Can't DROP; check that column/key exists
  - Error 1092: Records Duplicates Warnings
  - Error 1093: You can't specify target table for update in FROM clause
  - Error 1094: Unknown thread id
  - Error 1095: You are not owner of thread
  - Error 1096: No tables used
  - Error 1097: Too many strings for column and SET
  - Error 1098: Can't generate a unique log-filename

- Error 1099: Table was locked with a READ lock and can't be updated

MariaDB Error Codes 1100 to 1199

- Error 1100: Table was not locked with LOCK TABLES
- Error 1101: You should never see it
- Error 1102: Incorrect database name
- Error 1103: Incorrect table name
- Error 1104: The SELECT would examine more than MAX_JOIN_SIZE rows
- Error 1105: Unknown error
- Error 1106: Unknown procedure
- Error 1107: Incorrect parameter count to procedure
- Error 1108: Incorrect parameters to procedure
- Error 1109: Unknown table
- Error 1110: Column specified twice
- Error 1111: Invalid use of group function
- Error 1112: Table uses an extension that doesn't exist in this MariaDB version
- Error 1113: A table must have at least 1 column
- Error 1114: The table is full
- Error 1115: Unknown character set
- Error 1116: Too many tables; MariaDB can only use tables in a join
- Error 1117: Too many columns
- Error 1118: Row size too large
- Error 1119: Thread stack overrun
- Error 1120: Cross dependency found in OUTER JOIN; examine your ON conditions
- Error 1121: Table handler doesn't support NULL in given index
- Error 1122: Can't load function
- Error 1123: Can't initialize function
- Error 1124: No paths allowed for shared library
- Error 1125: Function already exists
- Error 1126: Can't open shared library
- Error 1127: Can't find symbol in library
- Error 1128: Function is not defined
- Error 1129: Host is blocked because of many connection errors
- Error 1130: Host is not allowed to connect to this MariaDB server
- Error 1131: You are using MariaDB as an anonymous user
- Error 1132: You must have privileges to update tables in the mysql database
- Error 1133: Can't find any matching row in the user table
- Error 1134: Rows matched Changed Warnings
- Error 1135: Can't create a new thread
- Error 1136: Column count doesn't match value count at row
- Error 1137: Can't reopen table
- Error 1138: Invalid use of NULL value
- Error 1139: Got error from regexp
- Error 1140: Mixing of GROUP columns with no GROUP columns
- Error 1141: There is no such grant defined for user on host
- Error 1142: command denied to user for table
- Error 1143: command denied to user for column in table
- Error 1144: Illegal GRANT/REVOKE command
- Error 1145: The host or user argument to GRANT is too long
- Error 1146: Table doesn't exist
- Error 1147: There is no such grant defined for user on host on table
- Error 1148: The used command is not allowed with this MariaDB version
- Error 1149: You have an error in your SQL syntax
- Error 1150: Delayed insert thread couldn't get requested lock for table
- Error 1151: Too many delayed threads in use
- Error 1152: Aborted connection to db user
- Error 1153: Got a packet bigger than 'max_allowed_packet' bytes
- Error 1154: Got a read error from the connection pipe
- Error 1155: Got an error from fcntl()
- Error 1156: Got packets out of order
- Error 1157: Couldn't uncompress communication packet
- Error 1158: Got an error reading communication packets
- Error 1159: Got timeout reading communication packets
- Error 1160: Got an error writing communication packets
- Error 1161: Got timeout writing communication packets
- Error 1162: Result string is longer than 'max_allowed_packet' bytes
- Error 1163: The used table type doesn't support BLOB/TEXT columns
- Error 1164: The used table type doesn't support AUTO_INCREMENT columns
- Error 1165: INSERT DELAYED can't be used with table because it is locked
- Error 1166: Incorrect column name
- Error 1167: The used storage engine can't index column
- Error 1168: Unable to open underlying table which is differently defined or of non-MyISAM type or do
- Error 1169: Can't write, because of unique constraint, to table
- Error 1170: BLOB/TEXT column used in key specification without a key length
- Error 1171: All parts of a PRIMARY KEY must be NOT NULL
- Error 1172: Result consisted of more than one row
- Error 1173: This table type requires a primary key
- Error 1174: This version of MariaDB is not compiled with RAID support
- Error 1175: You are using safe update mode and you tried to update a table without a WHERE that uses
- Error 1176: Key doesn't exist in table
- Error 1177: Can't open table
- Error 1178: The storage engine for the table doesn't support
- Error 1179: You are not allowed to execute this command in a transaction
- Error 1180: Got error during COMMIT
- Error 1181: Got error during ROLLBACK
- Error 1182: Got error during FLUSH_LOGS
- Error 1183: Got error during CHECKPOINT
- Error 1184: Aborted connection to db user host
- Error 1185: You should never see it
- Error 1186: Binlog closed, cannot RESET MASTER
- Error 1187: Failed rebuilding the index of dumped table
- Error 1188: Error from master
- Error 1189: Net error reading from master
- Error 1190: Net error writing to master
- Error 1191: Can't find FULLTEXT index matching the column list
- Error 1192: Can't execute the given command because you have active locked tables or an active trans
- Error 1193: Unknown system variable
- Error 1194: Table is marked as crashed and should be repaired
- Error 1195: Table is marked as crashed and last (automatic?) repair failed
- Error 1196: Some non-transactional changed tables couldn't be rolled back
- Error 1197: Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage
- Error 1198: This operation cannot be performed with a running slave; run STOP SLAVE first
- Error 1199: This operation requires a running slave; configure slave and do START SLAVE

MariaDB Error Codes 1200 to 1299

- Error 1200: The server is not configured as slave
- Error 1201: Could not initialize master info structure
- Error 1202: Could not create slave thread; check system resources
- Error 1203: User already has more than 'max_user_connections' active connections

Error 1204: You may only use constant expressions with SET
Error 1205: Lock wait timeout exceeded; try restarting transaction
Error 1206: The total number of locks exceeds the lock table size
Error 1207: Update locks cannot be acquired during a READ UNCOMMITTED transaction
Error 1208: DROP DATABASE not allowed while thread is holding global read lock
Error 1209: CREATE DATABASE not allowed while thread is holding global read lock
Error 1210: Incorrect arguments to
Error 1211: is not allowed to create new users
Error 1212: Incorrect table definition; all MERGE tables must be in the same database
Error 1213: Deadlock found when trying to get lock; try restarting transaction
Error 1214: The used table type doesn't support FULLTEXT indexes
Error 1215: Cannot add foreign key constraint
Error 1216: Cannot add or update a child row: a foreign key constraint fails
Error 1217: Cannot delete or update a parent row: a foreign key constraint fails
Error 1218: Error connecting to master
Error 1219: Error running query on master
Error 1220: Error when executing command
Error 1221: Incorrect usage of
Error 1222: The used SELECT statements have a different number of columns
Error 1223: Can't execute the query because you have a conflicting read lock
Error 1224: Mixing of transactional and non-transactional tables is disabled
Error 1225: Option used twice in statement
Error 1226: User has exceeded the resource
Error 1227: Access denied; you need (at least one of) the privilege(s) for this operation
Error 1228: Variable is a SESSION variable and can't be used with SET GLOBAL
Error 1229: Variable is a GLOBAL variable and should be set with SET GLOBAL
Error 1230: Variable doesn't have a default value
Error 1231: Variable can't be set to the value of
Error 1232: Incorrect argument type to variable
Error 1233: Variable can only be set, not read
Error 1234: Incorrect usage/placement
Error 1235: This version of MariaDB doesn't yet support
Error 1236: Got fatal error from master when reading data from binary log
Error 1237: Slave SQL thread ignored the query because of replicate-*-table rules
Error 1238: Variable is a
Error 1239: Incorrect foreign key definition
Error 1240: Key reference and table reference don't match
Error 1241: Operand should contain column(s)
Error 1242: Subquery returns more than 1 row
Error 1243: Unknown prepared statement handler given
Error 1244: Help database is corrupt or does not exist
Error 1245: Cyclic reference on subqueries
Error 1246: Converting column from to
Error 1247: Reference not supported
Error 1248: Every derived table must have its own alias
Error 1249: Select was reduced during optimization
Error 1250: Table from one of the SELECTs cannot be used
Error 1251: Client does not support authentication protocol requested by server
Error 1252: All parts of a SPATIAL index must be NOT NULL
Error 1253: COLLATION is not valid for CHARACTER SET
Error 1254: Slave is already running
Error 1255: Slave already has been stopped
Error 1256: Uncompressed data size too large
Error 1257: ZLIB: Not enough memory
Error 1258: ZLIB: Not enough room in the output buffer
Error 1259: ZLIB: Input data corrupted
Error 1260: Row was cut by GROUP_CONCAT()
Error 1261: Row doesn't contain data for all columns
Error 1262: Row was truncated; it contained more data than there were input columns
Error 1263: Column set to default value; NULL supplied to NOT NULL columnError 1263: Column set to d
Error 1264: Out of range value for column
Error 1265: Data truncated for column
Error 1266: Using storage engine for table
Error 1267: Illegal mix of collations for operation
Error 1268: Cannot drop one or more of the requested users
Error 1269: Can't revoke all privileges for one or more of the requested users
Error 1270: Illegal mix of collations for operation
Error 1271: Illegal mix of collations for operation
Error 1272: Variable is not a variable component (can't be used as XXXX.variable_name)
Error 1273: Unknown collation
Error 1274: SSL parameters in CHANGE MASTER are ignored because this MariaDB slave was compiled with
Error 1275: Server is running in --secure-auth mode, but has a password in the old format
Error 1276: Field or reference of SELECT was resolved in SELECT
Error 1277: Incorrect parameter or combination of parameters for START SLAVE UNTIL
Error 1278: It is recommended to use --skip-slave-start when doing step-by-step replication with STA
Error 1279: SQL thread is not to be started so UNTIL options are ignored
Error 1280: Incorrect index name
Error 1281: Incorrect catalog name
Error 1282: Query cache failed to set size; new query cache size is
Error 1283: Column cannot be part of FULLTEXT index
Error 1284: Unknown key cache
Error 1285: MariaDB is started in --skip-name-resolve mode
Error 1286: Unknown storage engine
Error 1287: is deprecated and will be removed in a future release. Please use instead
Error 1288: The target table of the is not updatable
Error 1289: The feature is disabled; you need MariaDB built with to have it working
Error 1290: The MariaDB server is running with the option so it cannot execute this statement
Error 1291: Column has duplicated value
Error 1292: Truncated incorrect value
Error 1293: Incorrect table definition; there can be only one TIMESTAMP column with CURRENT_TIMESTAM
Error 1294: Invalid ON UPDATE clause
Error 1295: This command is not supported in the prepared statement protocol yet
Error 1296: Got error from
Error 1297: Got temporary error from
Error 1298: Unknown or incorrect time zone
Error 1299: Invalid TIMESTAMP value in column at row

MariaDB Error Codes 1300 to 1399

Error 1300: Invalid character string
Error 1301: Result was larger than max_allowed_packet - truncated
Error 1302: Conflicting declarations

Error 1303: Can't create from within another stored routine
Error 1304: already exists
Error 1305: does not exist
Error 1306: Failed to DROP
Error 1307: Failed to CREATE
Error 1308: with no matching label
Error 1309: Redefining label
Error 1310: End-label without match
Error 1311: Referring to uninitialized variable
Error 1312: PROCEDURE can't return a result set in the given context
Error 1313: RETURN is only allowed in a FUNCTION
Error 1314: is not allowed in stored procedures
Error 1315: The update log is deprecated and replaced by the binary log
Error 1316: The update log is deprecated and replaced by the binary log; SET SQL_LOG_UPDATE has been
Error 1317: Query execution was interrupted
Error 1318: Incorrect number of arguments; expected, got
Error 1319: Undefined CONDITION
Error 1320: No RETURN found in FUNCTION
Error 1321: FUNCTION ended without RETURN
Error 1322: Cursor statement must be a SELECT
Error 1323: Cursor SELECT must not have INTO
Error 1324: Undefined CURSOR
Error 1325: Cursor is already open
Error 1326: Cursor is not open
Error 1327: Undeclared variable
Error 1328: Incorrect number of FETCH variables
Error 1329: No data - zero rows fetched, selected, or processed
Error 1330: Duplicate parameter
Error 1331: Duplicate variable
Error 1332: Duplicate condition
Error 1333: Duplicate cursor
Error 1334: Failed to ALTER
Error 1335: Subquery value not supported
Error 1336: is not allowed in stored function or trigger
Error 1337: Variable or condition declaration after cursor or handler declaration
Error 1338: Cursor declaration after handler declaration
Error 1339: Case not found for CASE statement
Error 1340: Configuration file is too big
Error 1341: Malformed file type header in file
Error 1342: Unexpected end of file while parsing comment
Error 1343: Error while parsing parameter
Error 1344: Unexpected end of file while skipping unknown parameter
Error 1345: EXPLAIN/SHOW can not be issued; lacking privileges for underlying table
Error 1346: File has unknown type in its header
Error 1347: '%s.%s' is not %s
Error 1348: Column is not updatable
Error 1349: View's SELECT contains a subquery in the FROM clause
Error 1350: View's SELECT contains clause
Error 1351: View's SELECT contains a variable or parameter
Error 1352: View's SELECT refers to a temporary table'
Error 1353: View's SELECT and view's field list have different column counts
Error 1354: View merge algorithm can't be used here for now
Error 1355: View being updated does not have complete key of underlying table in it
Error 1356: View references invalid table(s) or column(s) or function(s) or definer/invoker of view
Error 1357: Can't drop or alter from within another stored routine
Error 1358: GOTO is not allowed in a stored procedure handler
Error 1359: Trigger already exists
Error 1360: Trigger does not exist
Error 1361: Trigger's is view or temporary table
Error 1362: Updating of row is not allowed in trigger
Error 1363: There is no row in trigger
Error 1364: Field doesn't have a default value
Error 1365: Division by 0
Error 1366: Incorrect value for column at row
Error 1367: Illegal value found during parsing
Error 1368: CHECK OPTION on non-updatable view
Error 1369: CHECK OPTION failed
Error 1370: command denied to user for routine
Error 1371: Failed purging old relay logs
Error 1372: Password hash should be a %d-digit hexadecimal number
Error 1373: Target log not found in binlog index
Error 1374: I/O error reading log index file
Error 1375: Server configuration does not permit binlog purge
Error 1376: Failed on fseek()
Error 1377: Fatal error during log purge
Error 1378: A purgeable log is in use, will not purge
Error 1379: Unknown error during log purge
Error 1380: Failed initializing relay log position
Error 1381: You are not using binary logging
Error 1382: The syntax is reserved for purposes internal to the MariaDB server
Error 1383: WSAStartup Failed
Error 1384: Can't handle procedures with different groups yet
Error 1385: Select must have a group with this procedure
Error 1386: Can't use ORDER clause with this procedure
Error 1387: Binary logging and replication forbid changing the global server
Error 1388: Can't map file
Error 1389: Wrong magic in
Error 1390: Prepared statement contains too many placeholders
Error 1391: Key part length cannot be 0
Error 1392: View text checksum failed
Error 1393: Can not modify more than one base table through a join view
Error 1394: Can not insert into join view without fields list
Error 1395: Can not delete from join view
Error 1396: Operation failed for
Error 1397: XAER_NOTA: Unknown XID
Error 1398: XAER_INVAL: Invalid arguments (or unsupported command)
Error 1399: XAER_RMFAIL: The command cannot be executed when global transaction is in the state

MariaDB Error Codes 1400 to 1499

Error 1400: XAER_OUTSIDE: Some work is done outside global transaction
Error 1401: XAER_RMERR: Fatal error occurred in the transaction branch - check your data for consist
Error 1402: XA_RBROLLBACK: Transaction branch was rolled back
Error 1403: There is no such grant defined for user on host on routine
Error 1404: Failed to grant EXECUTE and ALTER ROUTINE privileges
Error 1405: Failed to revoke all privileges to dropped routine
Error 1406: Data too long for column at row
Error 1407: Bad SQLSTATE
Error 1408: ready for connections. Version: socket: port:
Error 1409: Can't load value from file with fixed size rows to variable
Error 1410: You are not allowed to create a user with GRANT
Error 1411: Incorrect value for function

Error 1412: Table definition has changed, please retry transaction
Error 1413: Duplicate handler declared in the same block
Error 1414: OUT or INOUT argument for routine is not a variable or NEW pseudo-variable in BEFORE tri
Error 1415: Not allowed to return a result set from
Error 1416: Cannot get geometry object from data you send to the GEOMETRY field
Error 1417: A routine failed and has neither NO SQL nor READS SQL DATA in its declaration
Error 1418: This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration an
Error 1419: You do not have the SUPER privilege and binary logging is enabled
Error 1420: You can't execute a prepared statement which has an open cursor associated with it
Error 1421: The statement has no open cursor
Error 1422: Explicit or implicit commit is not allowed in stored function or trigger
Error 1423: Field of view underlying table doesn't have a default value
Error 1424: Recursive stored functions and triggers are not allowed.
Error 1425: Too big scale specified for column
Error 1426: Too big precision specified for column
Error 1427: For float(M,D, double(M,D or decimal(M,D, M must be >= D
Error 1428: You can't combine write-locking of system tables with other tables or lock types
Error 1429: Unable to connect to foreign data source
Error 1430: There was a problem processing the query on the foreign data source
Error 1431: The foreign data source you are trying to reference does not exist
Error 1432: Can't create federated table. The data source connection string is not in the correct format
Error 1433: The data source connection string is not in the correct format
Error 1434: Can't create federated table. Foreign data src error
Error 1435: Trigger in wrong schema
Error 1436: Thread stack overrun
Error 1437: Routine body is too long
Error 1438: Cannot drop default keycache
Error 1439: Display width out of range for column
Error 1440: XAER_DUPID: The XID already exists
Error 1441: Datetime function: field overflow
Error 1442: Can't update table in stored function/trigger because it is already used by statement wh
Error 1443: The definition of table prevents operation on table
Error 1444: The prepared statement contains a stored routine call that refers to that same statement
Error 1445: Not allowed to set autocommit from a stored function or trigger
Error 1446: Definer is not fully qualified
Error 1447: View has no definer information (old table format)
Error 1448: You need the SUPER privilege for creation view with definer
Error 1449: The user specified as a definer does not exist
Error 1450: Changing schema from is not allowed.
Error 1451: Cannot delete or update a parent row: a foreign key constraint fails
Error 1452: Cannot add or update a child row: a foreign key constraint fails
Error 1453: Variable must be quoted with ..., or renamed
Error 1454: No definer attribute for trigger
Error 1455: has an old format, you should re-create the object(s)
Error 1456: Recursive limit (as set by the max_sp_recursion_depth variable) was exceeded for routine
Error 1457: Failed to load routine
Error 1458: Incorrect routine name
Error 1459: Table upgrade required. Please do "REPAIR TABLE" or dump/reload to fix it!
Error 1460: AGGREGATE is not supported for stored functions
Error 1461: Can't create more than max_prepared_stmt_count statements
Error 1462: contains view recursion
Error 1463: Non-grouping field is used in clause
Error 1464: The used table type doesn't support SPATIAL indexes
Error 1465: Triggers can not be created on system tables
Error 1466: Leading spaces are removed from name
Error 1467: Failed to read auto-increment value from storage engine
Error 1468: user name
Error 1469: host name
Error 1470: String is too long for (should be no longer than)
Error 1471: The target table of the is not insertable-into
Error 1472: Table is differently defined or of non-MyISAM type or doesn't exist
Error 1473: Too high level of nesting for select
Error 1474: Name has become ''
Error 1475: First character of the FIELDS TERMINATED string is ambiguous
Error 1476: The foreign server you are trying to create already exists
Error 1477: The foreign server name you are trying to reference does not exist. Data source error
Error 1478: Table storage engine does not support the create option
Error 1479: Syntax error: PARTITIONING requires definition of VALUES for each partition
Error 1480: Only PARTITIONING can use VALUES in partition definition
Error 1481: MAXVALUE can only be used in last partition definition
Error 1482: Subpartitions can only be hash partitions and by key
Error 1483: Must define subpartitions on all partitions if on one partition
Error 1484: Wrong number of partitions defined, mismatch with previous setting
Error 1485: Wrong number of subpartitions defined, mismatch with previous setting
Error 1486: Constant/Random expression in (sub)partitioning function is not allowed
Error 1487: Expression in RANGE/LIST VALUES must be constant
Error 1488: Field in list of fields for partition function not found in table
Error 1489: List of fields is only allowed in KEY partitions
Error 1490: The partition info in the frm file is not consistent with what can be written into the f
Error 1491: The function returns the wrong type
Error 1492: For partitions each partition must be defined
Error 1493: VALUES LESS THAN value must be strictly increasing for each partition
Error 1494: VALUES value must be of same type as partition function
Error 1495: Multiple definition of same constant in list partitioning
Error 1496: Partitioning can not be used stand-alone in query
Error 1497: The mix of handlers in the partitions is not allowed in this version of MariaDB
Error 1498: For the partitioned engine it is necessary to define all
Error 1499: Too many partitions (including subpartitions) were defined

MariaDB Error Codes 1500 to 1599

Error 1500: It is only possible to mix RANGE/LIST partitioning with HASH/KEY partitioning for subpar

Error 1501: Failed to create specific handler file

Error 1502: A BLOB field is not allowed in partition function

Error 1503: must include all columns in the table's partitioning function

Error 1504: Number of = 0 is not an allowed value

Error 1505: Partition management on a not partitioned table is not possible

Error 1506: Foreign key clause is not yet supported in conjunction with partitioning

Error 1507: Error in list of partitions to

Error 1508: Cannot remove all partitions, use DROP TABLE instead

Error 1509: COALESCE PARTITION can only be used on HASH/KEY partitions

Error 1510: REORGANIZE PARTITION can only be used to reorganize partitions not to change their numbe

Error 1511: REORGANIZE PARTITION without parameters can only be used on auto-partitioned tables

Error 1512: PARTITION can only be used on RANGE/LIST partitions

Error 1513: Trying to Add partition(s) with wrong number of subpartitions

Error 1514: At least one partition must be added

Error 1515: At least one partition must be coalesced

Error 1516: More partitions to reorganize than there are partitions

Error 1517: Duplicate partition name

Error 1518: It is not allowed to shut off binlog on this command

Error 1519: When reorganizing a set of partitions they must be in consecutive order

Error 1520: Reorganize of range partitions cannot change total ranges except for last partition

Error 1521: Partition function not supported in this version for this handler

Error 1522: Partition state cannot be defined from CREATE/ALTER TABLE

Error 1523: The handler only supports 32 bit integers in VALUES

Error 1524: Plugin is not loaded

Error 1525: Incorrect value

Error 1526: Table has no partition for value

Error 1527: It is not allowed to specify more than once

Error 1528: Failed to create

Error 1529: Failed to drop

Error 1530: The handler doesn't support autoextend of tablespaces

Error 1531: A size parameter was incorrectly specified, either number or on the form 10M

Error 1532: The size number was correct but we don't allow the digit part to be more than 2 billion

Error 1533: Failed to alter

Error 1534: Writing one row to the row-based binary log failed

Error 1535: Table definition on master and slave does not match

Error 1536: Slave running with --log-slave-updates must use row-based binary logging to be able to r

Error 1537: Event already exists

Error 1538: Failed to store event. Error code from storage engine.

Error 1539: Unknown event

Error 1540: Failed to alter event

Error 1541: Failed to drop

Error 1542: INTERVAL is either not positive or too big

Error 1543: ENDS is either invalid or before STARTS

Error 1544: Event execution time is in the past. Event has been disabled

Error 1545: Failed to open mysql.event

Error 1546: No datetime expression provided

Error 1547: Column count is wrong. Expected found. The table is probably corrupted

Error 1548: Cannot load from mysql. The table is probably corrupted

Error 1549: Failed to delete the event from mysql.event

Error 1550: Error during compilation of event's body

Error 1551: Same old and new event name

Error 1552: Data for column too long

Error 1553: Cannot drop index needed in a foreign key constraint

Error 1554: The syntax is deprecated and will be removed in MariaDB

Error 1555: You can't write-lock a log table. Only read access is possible

Error 1556: You can't use locks with log tables

Error 1557: Upholding foreign key constraints for table, entry, key would lead to a duplicate entry

Error 1558: Column count is wrong. Expected found. Created with MariaDB now running. Please use mysq

Error 1559: Cannot switch out of the row-based binary log format when the session has open temporary

Error 1560: Cannot change the binary logging format inside a stored function or trigger

Error 1561: You should never see it

Error 1562: Cannot create temporary table with partitions

Error 1563: Partition constant is out of partition function domain

Error 1564: This partition function is not allowed

Error 1565: Error in DDL log

Error 1566: Not allowed to use NULL value in VALUES LESS THAN

Error 1567: Incorrect partition name

Error 1568: Transaction isolation level can't be changed while a transaction is in progress

Error 1569: ALTER TABLE causes auto_increment resequencing, resulting in duplicate entry for key

Error 1570: Internal scheduler error

Error 1571: Error during starting/stopping of the scheduler

Error 1572: Engine cannot be used in partitioned tables

Error 1573: Cannot activate log

Error 1574: The server was not built with row-based replication

Error 1575: Decoding of base64 string failed

Error 1576: Recursion of EVENT DDL statements is forbidden when body is present

Error 1577: Cannot proceed because system tables used by Event Scheduler were found damaged at serve

Error 1578: Only integers allowed as number here

Error 1579: This storage engine cannot be used for log tables

Error 1580: You cannot '%s' a log table if logging is enabled

Error 1581: Cannot rename. When logging enabled, rename to/from log table must rename two tables

Error 1582: Incorrect parameter count in the call to native function

Error 1583: Incorrect parameters in the call to native function

Error 1584: Incorrect parameters in the call to stored function

Error 1585: This function has the same name as a native function

Error 1586: Duplicate entry for key

Error 1587: Too many files opened, please execute the command again

Error 1588: Event execution time is in the past and ON COMPLETION NOT PRESERVE is set

Error 1589: Event execution time is in the past and ON COMPLETION NOT PRESERVE is set

Error 1590: The incident occurred on the master

Error 1591: Table has no partition for some existing values

Error 1592: Unsafe statement written to the binary log using statement format since BINLOG_FORMAT =

Error 1593: Fatal error

Error 1594: Relay log read failure

Error 1595: Relay log write failure

Error 1596: Failed to create

Error 1597: Master command failed

- Error 1598: Binary logging not possible
- Error 1599: View has no creation context

MariaDB Error Codes 1600 to 1699

- Error 1600: Creation context of view is invalid
- Error 1601: Creation context of stored routine is invalid
- Error 1602: Corrupted TRG file for table
- Error 1603: Triggers for table have no creation context
- Error 1604: Trigger creation context of table is invalid
- Error 1605: Creation context of event is invalid
- Error 1606: Cannot open table for trigger
- Error 1607: Cannot create stored routine. Check warnings
- Error 1608: You should never see it
- Error 1609: The BINLOG statement of type was not preceded by a format description BINLOG statement.
- Error 1610: Corrupted replication event was detected
- Error 1611: Invalid column reference in LOAD DATA
- Error 1612: Being purged log was not found
- Error 1613: XA_RBTIMEOUT: Transaction branch was rolled back: took too long
- Error 1614: XA_RBDEADLOCK: Transaction branch was rolled back: deadlock was detected
- Error 1615: Prepared statement needs to be re-prepared
- Error 1616: DELAYED option not supported for table
- Error 1617: The master info structure does not exist
- Error 1618: option ignored
- Error 1619: Built-in plugins cannot be deleted
- Error 1620: Plugin is busy and will be uninstalled on shutdown
- Error 1621: variable is read-only. Use SET to assign the value
- Error 1622: Storage engine does not support rollback for this statement
- Error 1623: Unexpected master's heartbeat data
- Error 1624: The requested value for the heartbeat period is either negative or exceeds the maximum a
- Error 1625: You should never see it
- Error 1626: Error in parsing conflict function
- Error 1627: Write to exceptions table failed
- Error 1628: Comment for table is too long
- Error 1629: Comment for field is too long
- Error 1630: FUNCTION does not exist. Check the 'Function Name Parsing and Resolution' section in the
- Error 1631: Database
- Error 1632: Table
- Error 1633: Partition
- Error 1634: Subpartition
- Error 1635: Temporary
- Error 1636: Renamed
- Error 1637: Too many active concurrent transactions
- Error 1638: Non-ASCII separator arguments are not fully supported
- Error 1639: debug sync point wait timed out
- Error 1640: debug sync point hit limit reached
- Error 1641: Duplicate condition information item
- Error 1642: Unhandled user-defined warning condition
- Error 1643: Unhandled user-defined not found condition
- Error 1644: Unhandled user-defined exception condition
- Error 1645: RESIGNAL when handler not active
- Error 1646: SIGNAL/RESIGNAL can only use a CONDITION defined with SQLSTATE
- Error 1647: Data truncated for condition item
- Error 1648: Data too long for condition item
- Error 1649: Unknown locale
- Error 1650: The requested server id clashes with the slave startup option --replicate-same-server-id
- Error 1651: Query cache is disabled; restart the server with query_cache_type=1 to enable it
- Error 1652: Duplicate partition field name
- Error 1653: Inconsistency in usage of column lists for partitioning
- Error 1654: Partition column values of incorrect type
- Error 1655: Too many fields
- Error 1656: Cannot use MAXVALUE as value in VALUES IN
- Error 1657: Cannot have more than one value for this type of partitioning
- Error 1658: Row expressions in VALUES IN only allowed for multi-field column partitioning
- Error 1659: Field is of a not allowed type for this type of partitioning
- Error 1660: The total length of the partitioning fields is too large
- Error 1661: Cannot execute statement: impossible to write to binary log since both row-incapable eng
- Error 1662: Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = ROW an
- Error 1663: Cannot execute statement: impossible to write to binary log since statement is unsafe, s
- Error 1664: Cannot execute statement: impossible to write to binary log since statement is in row format
- Error 1665: Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEM
- Error 1666: Cannot execute statement: impossible to write to binary log since statement is in row format
- Error 1667: Cannot execute statement: impossible to write to binary log since more than one engine i
- Error 1668: The statement is unsafe because it uses a LIMIT clause
- Error 1669: The statement is unsafe because it uses INSERT DELAYED. This is unsafe because the times
- Error 1670: The statement is unsafe because it uses the general log, slow query log, or performance_
- Error 1671: Statement is unsafe because it invokes a trigger or a stored function that inserts into
- Error 1672: Statement is unsafe because it uses a UDF which may not return the same value on the sla
- Error 1673: Statement is unsafe because it uses a system variable that may have a different value on
- Error 1674: Statement is unsafe because it uses a system function that may return a different value
- Error 1675: Statement is unsafe because it accesses a non-transactional table after accessing a tran
- Error 1676: Statement
- Error 1677: Column of table cannot be converted from type to type
- Error 1678: Can't create conversion table for table
- Error 1679: Cannot modify @@session.binlog_format inside a transaction
- Error 1680: The path specified is too long
- Error 1681: is deprecated and will be removed in a future release.
- Error 1682: Native table has the wrong structure
- Error 1683: Invalid performance_schema usage
- Error 1684: Table was skipped since its definition is being modified by concurrent DDL statement
- Error 1685: Cannot modify @@session.binlog_direct_non_transactional_updates inside a transaction
- Error 1686: Cannot change the binlog direct flag inside a stored function or trigger
- Error 1687: A SPATIAL index may only contain a geometrical type column
- Error 1688: Comment for index is too long
- Error 1689: Wait on a lock was aborted due to a pending exclusive lock
- Error 1690: value is out of range in
- Error 1691: A variable of a non-integer based type in LIMIT clause
- Error 1692: Mixing self-logging and non-self-logging engines in a statement is unsafe.
- Error 1693: Statement accesses nontransactional table as well as transactional or temporary table, a

Error 1694: Cannot modify @@session.sql_log_bin inside a transaction

Error 1695: Cannot change the sql_log_bin inside a stored function or trigger

Error 1696: Failed to read from the .par file

Error 1697: VALUES value for partition must have type INT

Error 1698: Access denied for user

Error 1699: SET PASSWORD has no significance for users authenticating via plugins

MariaDB Error Codes 1700 to 1799

Error 1700: GRANT with IDENTIFIED WITH is illegal because the user already exists

Error 1701: Cannot truncate a table referenced in a foreign key constraint

Error 1702: Plugin is force_plus_permanent and can not be unloaded

Error 1703: The requested value for the heartbeat period is less than 1 millisecond

Error 1704: The requested value for the heartbeat period exceeds the value of slave_net_timeout seco

Error 1705: Multi-row statements required more than 'max_binlog_stmt_cache_size' bytes of storage; i

Error 1706: Primary key/partition key update is not allowed since the table is updated both as and

Error 1707: Table rebuild required. Please do "ALTER TABLE FORCE" or dump/reload to fix it!

Error 1708: The value of should be no less than the value of

Error 1709: Index column size too large. The maximum column size is bytes.

Error 1710: Trigger has an error in its body

Error 1711: Unknown trigger has an error in its body

Error 1712: Index is corrupted

Error 1713: Undo log record is too big

Error 1714: INSERT IGNORE... SELECT is unsafe because the order in which rows are retrieved by the S

Error 1715: INSERT... SELECT... ON DUPLICATE KEY UPDATE is unsafe because the order in which rows ar

Error 1716: REPLACE... SELECT is unsafe because the order in which rows are retrieved by the SELECT

Error 1717: CREATE... IGNORE SELECT is unsafe because the order in which rows are retrieved by the S

Error 1718: CREATE... REPLACE SELECT is unsafe because the order in which rows are retrieved by the

Error 1719: UPDATE IGNORE is unsafe because the order in which rows are updated determines which (if

Error 1720: You should never see it

Error 1721: You should never see it

Error 1722: Statements writing to a table with an auto-increment column after selecting from another

Error 1723: CREATE TABLE... SELECT... on a table with an auto-increment column is unsafe because the

Error 1724: INSERT... ON DUPLICATE KEY UPDATE on a table with more than one UNIQUE KEY is unsafe

Error 1725: Table is being used in foreign key check.

Error 1726: Storage engine does not support system tables

Error 1727: INSERT into autoincrement field which is not the first part in the composed primary key

Error 1728: Cannot load from. The table is probably corrupted

Error 1729: The requested value for the master delay exceeds the maximum

Error 1730: Only Format_description_log_event and row events are allowed in BINLOG statements

Error 1731: Non matching attribute between partition and table

Error 1732: Table to exchange with partition is partitioned

Error 1733: Table to exchange with partition is temporary

Error 1734: Subpartitioned table, use subpartition instead of partition

Error 1735: Unknown partition in table

Error 1736: Tables have different definitions

Error 1737: Found a row that does not match the partition

Error 1738: Option binlog_cache_size is greater than max_binlog_cache_size; setting binlog_cache_siz

Error 1739: Cannot use access on index due to type or collation conversion on field

Error 1740: Table to exchange with partition has foreign key references

Error 1741: Key value was not found in table

Error 1742: Data for column too long

Error 1743: Replication event checksum verification failed while reading from network.

Error 1744: Replication event checksum verification failed while reading from a log file

Error 1745: Option binlog_stmt_cache_size is greater than max_binlog_stmt_cache_size

Error 1746: Can't update table while is being created

Error 1747: PARTITION () clause on non partitioned table

Error 1748: Found a row not matching the given partition set

Error 1749: partition doesn't exist

Error 1750: Failure while changing the type of replication repository

Error 1751: The creation of some temporary tables could not be rolled back.

Error 1752: Some temporary tables were dropped, but these operations could not be rolled back.

Error 1753: is not supported in multi-threaded slave mode

Error 1754: The number of modified databases exceeds the maximum

Error 1755: Cannot execute the current event group in the parallel mode

Error 1756: %s

Error 1757: FULLTEXT index is not supported for partitioned tables

Error 1758: Invalid condition number

Error 1759: Sending passwords in plain text without SSL/TLS is extremely insecure

Error 1760: Storing MariaDB user name or password information in the master.info repository is not s

Error 1761: Foreign key constraint for table, record would lead to a duplicate entry in table

Error 1762: Foreign key constraint for table, record would lead to a duplicate entry in a child table

Error 1763: Setting authentication options is not possible when only the Slave SQL Thread is being s

Error 1764: The table does not have FULLTEXT index to support this query

Error 1765: The system variable cannot be set in stored functions or triggers

Error 1766: The system variable cannot be set when there is an ongoing transaction

Error 1767: The system variable @@SESSION.GTID_NEXT has the value which is not listed in @@SESSION.G

Error 1768: The system variable @@SESSION.GTID_NEXT cannot change inside a transaction

Error 1769: The statement 'SET' cannot invoke a stored function

Error 1770: The system variable @@SESSION.GTID_NEXT cannot be 'AUTOMATIC' when @@SESSION.GTID_NEXT_L

Error 1771: Skipping transaction because it has already been executed and logged

Error 1772: Malformed GTID set specification

Error 1773: Malformed GTID set encoding

Error 1774: Malformed GTID specification

Error 1775: Impossible to generate Global Transaction Identifier: the integer component reached the

Error 1776: Parameters MASTER_LOG_FILE, MASTER_LOG_POS, RELAY_LOG_FILE and RELAY_LOG_POS cannot be s

Error 1777: CHANGE MASTER TO MASTER_AUTO_POSITION = 1 can only be executed when @@GLOBAL.GTID_MODE =

Error 1778: Cannot execute statements with implicit commit inside a transaction when @@SESSION.GTID_

Error 1779: GTID_MODE = ON or GTID_MODE = UPGRADE_STEP_2 requires DISABLE_GTID_UNSAFE_STATEMENTS = 1

Error 1780: @@GLOBAL.GTID_MODE = ON or UPGRADE_STEP_1 or UPGRADE_STEP_2 requires --log-bin and --log

Error 1781: @@SESSION.GTID_NEXT cannot be set to UUID:NUMBER when @@GLOBAL.GTID_MODE = OFF

Error 1782: @@SESSION.GTID_NEXT cannot be set to ANONYMOUS when @@GLOBAL.GTID_MODE = ON

Error 1783: @@SESSION.GTID_NEXT_LIST cannot be set to a non-NULL value when @@GLOBAL.GTID_MODE = OFF

Error 1784: Found a Gtid_log_event or Previous_gtids_log_event when @@GLOBAL.GTID_MODE = OFF

Error 1785: When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, updates to non-transactional tables can only

Error 1786: CREATE TABLE ... SELECT is forbidden when @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1

Error 1787: When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, the statements CREATE TEMPORARY TABLE and DR

Error 1788: The value of @@GLOBAL.GTID_MODE can only change one step at a time

Error 1789: The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the master

Error 1790: @@SESSION.GTID_NEXT cannot be changed by a client that owns a GTID

Error 1791: Unknown EXPLAIN format name

Error 1792: Cannot execute statement in a READ ONLY transaction

Error 1793: Comment for table partition is too long

Error 1794: Slave is not configured or failed to initialize properly. You must at least set --server

Error 1795: InnoDB presently supports one FULLTEXT index creation at a time

Error 1796: Cannot create FULLTEXT index on temporary InnoDB table

Error 1797: Column is of wrong type for an InnoDB FULLTEXT index

Error 1798: Index '%s' is of wrong type for an InnoDB FULLTEXT index

Error 1799: Creating index required more than 'innodb_online_alter_log_max_size' bytes of modificati

MariaDB Error Codes 1800 to 1899

Error 1800: Unknown ALGORITHM

Error 1801: Unknown LOCK type

Error 1802: CHANGE MASTER cannot be executed when the slave was stopped with an error or killed in M

Error 1803: Cannot recover after SLAVE errored out in parallel execution mode

Error 1804: Cannot clean up worker info tables

Error 1805: Column count of is wrong. Expected, found. The table is probably corrupted

Error 1806: Slave must silently retry current transaction

Error 1807: There is a foreign key check running on table

Error 1808: Schema mismatch

Error 1809: Table in system tablespace

Error 1810: IO Read error

Error 1811: IO Write error

Error 1812: Tablespace is missing for table

Error 1813: Tablespace for table exists. Please DISCARD the tablespace before IMPORT

Error 1814: Tablespace has been discarded for table

Error 1815: Internal error

Error 1816: ALTER TABLE IMPORT TABLESPACE failed with error

Error 1817: Index corrupt

Error 1818: YEAR(%lu) column type is deprecated. Creating YEAR(4) column instead.

Error 1819: Your password does not satisfy the current policy requirements

Error 1820: You must SET PASSWORD before executing this statement

Error 1821: Failed to add the foreign key constraint. Missing index for constraint in the foreign ta

Error 1822: Failed to add the foreign key constraint. Missing index for constraint in the referenced

Error 1823: Failed to add the foreign key constraint to system tables

Error 1824: Failed to open the referenced table

Error 1825: Failed to add the foreign key constraint on table. Incorrect options in FOREIGN KEY cons

Error 1826: Duplicate foreign key constraint name

Error 1827: The password hash doesn't have the expected format

Error 1828: Cannot drop column: needed in a foreign key constraint

Error 1829: Cannot drop column: needed in a foreign key constraint of table

Error 1830: Column cannot be NOT NULL: needed in a foreign key constraint SET NULL

Error 1831: Duplicate index defined on the table. This is deprecated and will be disallowed in a fut

Error 1832: Cannot change column: used in a foreign key constraint

Error 1833: Cannot change column: used in a foreign key constraint of table

Error 1834: Cannot delete rows from table which is parent in a foreign key constraint of table

Error 1835: Malformed communication packet.

Error 1836: Running in read-only mode

Error 1837: When GTID_NEXT is set to a GTID, you must explicitly set it again after a COMMIT or ROLL

Error 1838: The system variable cannot be set in stored procedures

Error 1839: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_MODE = ON

Error 1840: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty

Error 1841: @@GLOBAL.GTID_PURGED can only be set when there are no ongoing transactions (not even in

Error 1842: @@GLOBAL.GTID_PURGED was changed from to

Error 1843: @@GLOBAL.GTID_EXECUTED was changed from to

Error 1844: Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEM

Error 1845: is not supported for this operation. Try

Error 1846: is not supported. Reason: Try

Error 1847: COPY algorithm requires a lock

Error 1848: Partition specific operations do not yet support LOCK/ALGORITHM

Error 1849: Columns participating in a foreign key are renamed

Error 1850: Cannot change column type INPLACE

Error 1851: Adding foreign keys needs foreign_key_checks=OFF

Error 1852: Creating unique indexes with IGNORE requires COPY algorithm to remove duplicate rows

Error 1853: Dropping a primary key is not allowed without also adding a new primary key

Error 1854: Adding an auto-increment column requires a lock

Error 1855: Cannot replace hidden FTS_DOC_ID with a user-visible one

Error 1856: Cannot drop or rename FTS_DOC_ID

Error 1857: Fulltext index creation requires a lock

Error 1858: sql_slave_skip_counter can not be set when the server is running with @@GLOBAL.GTID_MODE

Error 1859: Duplicate entry for key

Error 1860: Long database name and identifier for object resulted in path length exceeding maximum c

Error 1861: cannot silently convert NULL values, as required in this SQL_MODE

Error 1862: Your password has expired. To log in you must change it using a client that supports exp

Error 1863: Found a row in wrong partition

Error 1864: Cannot schedule event, relay-log name, position to Worker thread because its size exceed

Error 1865: Cannot CREATE FULLTEXT INDEX WITH PARSER on InnoDB table

Error 1866: The binary log file is logically corrupted

Error 1867: file was not purged because it was being read by thread(s), purged only out of

Error 1868: file was not purged because it is the active log file

Error 1869: Auto-increment value in UPDATE conflicts with internally generated values

Error 1870: Row events are not logged for statements that modify BLACKHOLE tables in row format

Error 1871: Slave failed to initialize master info structure from the repository

Error 1872: Slave failed to initialize relay log info structure from the repository

Error 1873: Access denied trying to change to user (using password). Disconnecting.

Error 1874: InnoDB is in read only mode

Error 1875: STOP SLAVE command execution is incomplete: Slave SQL thread got the stop signal

Error 1876: STOP SLAVE command execution is incomplete: Slave IO thread got the stop signal

Error 1877: Operation cannot be performed. The table is missing, corrupt or contains bad data

Error 1878: Temporary file write failure

Error 1879: Upgrade index name failed, please use create index(alter table) algorithm copy to rebuild

Error 1880: ''

MariaDB Error Codes 1900 to 1999

Error 1900: ''

Error 1901: Function or expression cannot be used in the clause of

Error 1902: ''

Error 1903: Primary key cannot be defined upon a generated column

Error 1904: Key/Index cannot be defined on a virtual generated column

Error 1905: Cannot define foreign key with clause on a generated column

Error 1906: The value specified for generated column in table has been ignored

Error 1907: This is not yet supported for generated columns

Error 1908: ''

Error 1909: ''

Error 1910: storage engine does not support generated columns

Error 1911: Unknown option

Error 1912: Incorrect value for option

Error 1913: You should never see it

Error 1914: You should never see it

Error 1915: You should never see it

Error 1916: Got overflow when converting to. Value truncated

Error 1917: Truncated value when converting to

Error 1918: Encountered illegal value when converting to

Error 1919: Encountered illegal format of dynamic column string

Error 1920: Dynamic column implementation limit reached

Error 1921: Illegal value used as argument of dynamic column function

Error 1922: Dynamic column contains unknown character set

Error 1923: At least one of the 'in_to_exists' or 'materialization' optimizer_switch flags must be '

Error 1924: Query cache is disabled (resize or similar command in progress); repeat this command lat

Error 1925: Query cache is globally disabled and you can't enable it only for this session

Error 1926: View ORDER BY clause ignored because there is other ORDER BY clause already

Error 1927: Connection was killed

Error 1928: You should never see it

Error 1929: Cannot modify @@session.skip_replication inside a transaction

Error 1930: Cannot modify @@session.skip_replication inside a stored function or trigger

Error 1931: Query execution was interrupted. The query exceeded. The query result may be incomplete

Error 1932: Table doesn't exist in engine

Error 1933: Target is not running an EXPLAINable command

Error 1934: Connection conflicts with existing connection

Error 1935: Master

Error 1936: Can't %s SLAVE

Error 1937: SLAVE started

Error 1938: SLAVE stopped

Error 1939: Engine failed to discover table with

Error 1940: Failed initializing replication GTID state

Error 1941: Could not parse GTID list

Error 1942: Could not update replication slave gtid state

Error 1943: GTID conflict (duplicate domain id)

Error 1944: Failed to open

Error 1945: Connecting slave requested to start from GTID, which is not in the master's binlog

Error 1946: Failed to load replication slave GTID position from table

Error 1947: Specified GTID conflicts with the binary log which contains a more recent GTID

Error 1948: Specified value for @@gtid_slave_pos contains no value for replication domain

Error 1949: START SLAVE UNTIL master_gtid_pos requires that slave is using GTID

Error 1950: An attempt was made to binlog GTID which would create an out-of-order sequence number wi

Error 1951: The binlog on the master is missing the GTID requested by the slave

Error 1952: Unexpected GTID received from master after reconnect

Error 1953: Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a transaction

Error 1954: Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a stored function

Error 1955: Connecting slave requested to start from GTID which is not in the master's binlog

Error 1956: This operation is not allowed if any GTID has been logged to the binary log

Error 1957: Unknown query id

Error 1958: Bad base64 data as position

Error 1959: Invalid role specification

Error 1960: The current user is invalid

Error 1961: Cannot grant role to

Error 1962: Cannot revoke role from

Error 1963: Cannot change @@slave_parallel_threads while another change is in progress

Error 1964: Commit failed due to failure of an earlier commit on which this one depends

Error 1965: is a view

- Error 1966: When using GTID, @@sql_slave_skip_counter can not be used
- Error 1967: The definition for table is too big
- Error 1968: Plugin already installed
- Error 1969: Query execution was interrupted (max_statement_time exceeded)
- Error 1970: does not support subqueries or stored functions
- Error 1971: The system variable cannot be set in SET STATEMENT
- Error 1972: You should never see it
- Error 1973: Can't create user; it already exists
- Error 1974: Can't drop user; it doesn't exist
- Error 1975: Can't create role; it already exists
- Error 1976: Can't drop role; it doesn't exist
- Error 1977: Cannot convert character to
- Error 1978: Incorrect default value for column
- Error 1979: You are not owner of query
- Error 1980: Engine-independent statistics are not collected for column
- Error 1981: Aggregate function of SELECT belongs to SELECT
- Error 1982: option ignored for InnoDB partition

MariaDB Error Codes 3000 to 3099

- Error 3000: File is corrupted
- Error 3001: Query partially completed on the master (error on master) and was aborted
- Error 3002: Query caused different errors on master and slave
- Error 3003: Storage engine for table is not loaded
- Error 3004: GET STACKED DIAGNOSTICS when handler not active
- Error 3005: is no longer supported. The statement was converted to
- Error 3006: Statement is unsafe because it uses a fulltext parser plugin which may not return the sa
- Error 3007: Cannot DISCARD/IMPORT tablespace associated with temporary table
- Error 3008: Foreign key cascade delete/update exceeds max depth of
- Error 3009: Column count of is wrong
- Error 3010: Trigger does not have CREATED attribute
- Error 3011: Referenced trigger for the given action time and event type does not exist
- Error 3012: EXPLAIN FOR CONNECTION command is supported only for SELECT/UPDATE/INSERT/DELETE/REPLACE
- Error 3013: Invalid size for column
- Error 3014: Table storage engine found required create option missing
- Error 3015: Out of memory in storage engine
- Error 3016: The password for anonymous user cannot be expired
- Error 3017: This operation cannot be performed with a running slave sql thread
- Error 3018: Cannot create FULLTEXT index on materialized subquery
- Error 3019: Undo Log error
- Error 3020: Invalid argument for logarithm
- Error 3021: This operation cannot be performed with a running slave io thread
- Error 3022: This operation may not be safe when the slave has temporary tables
- Error 3023: CHANGE MASTER TO with a MASTER_LOG_FILE clause but no MASTER_LOG_POS clause may not be s
- Error 3024: Query execution was interrupted, maximum statement execution time exceeded
- Error 3025: Select is not a read only statement, disabling timer
- Error 3026: Duplicate entry
- Error 3027: mode no longer has any effect. Use STRICT_ALL_TABLES or STRICT_TRANS_TABLES instead.
- Error 3028: Expression of ORDER BY contains aggregate function and applies to a UNION
- Error 3029: Expression of ORDER BY contains aggregate function and applies to the result of a non-ag
- Error 3030: Slave worker has stopped after at least one previous worker encountered an error when sl
- Error 3031: slave_preserve_commit_order is not supported
- Error 3032: The server is currently in offline mode
- Error 3033: Binary geometry function given two geometries of different srids
- Error 3034: Calling geometry function with unsupported types of arguments
- Error 3035: Unknown GIS error occurred in function
- Error 3036: Unknown exception caught in GIS function
- Error 3037: Invalid GIS data provided to function
- Error 3038: The geometry has no data in function
- Error 3039: Unable to calculate centroid because geometry is empty in function
- Error 3040: Geometry overlay calculation error: geometry data is invalid in function
- Error 3041: Geometry turn info calculation error: geometry data is invalid in function
- Error 3042: Analysis procedures of intersection points interrupted unexpectedly in function
- Error 3043: Unknown exception thrown in function
- Error 3044: Memory allocation error
- Error 3045: Domain error
- Error 3046: Length error
- Error 3047: Invalid argument error
- Error 3048: Out of range error
- Error 3049: Overflow error error
- Error 3050: Range error
- Error 3051: Underflow error
- Error 3052: Logic error
- Error 3053: Runtime error
- Error 3054: Unknown exception
- Error 3055: Geometry byte string must be little endian
- Error 3056: The password provided for the replication user exceeds the maximum length of 32 characte
- Error 3057: Incorrect user-level lock name
- Error 3058: Deadlock found when trying to get user-level lock
- Error 3059: REPLACE cannot be executed as it requires deleting rows that are not in the view
- Error 3060: Do not support online operation on table with GIS index

MariaDB Error Codes 4000 to 4099

- Error 4000: COM_MULTI can't return a result set in the given context
- Error 4001: Command is not allowed for COM_MULTI
- Error 4002: WITH column list and SELECT field list have different column counts
- Error 4003: Too many WITH elements in WITH clause
- Error 4004: Duplicate query name in WITH clause
- Error 4005: No anchors for recursive WITH element
- Error 4006: Unacceptable mutual recursion with anchored table
- Error 4007: Reference to recursive WITH table in materialized derived
- Error 4008: Restrictions imposed on recursive definitions are violated for table
- Error 4009: Window specification with name is not defined
- Error 4010: Multiple window specifications with the same name
- Error 4011: Window specification referencing another one cannot contain partition list
- Error 4012: Referenced window specification already contains order list
- Error 4013: Referenced window specification cannot contain window frame

Error 4014: Unacceptable combination of window frame bound specifications

Error 4015: Window function is allowed only in SELECT list and ORDER BY clause

Error 4016: Window function is not allowed in window specification

Error 4017: Window frame is not allowed with

Error 4018: No order list in window specification for

Error 4019: RANGE-type frame requires ORDER BY clause with single sort key

Error 4020: Integer is required for ROWS-type frame

Error 4021: Numeric datatype is required for RANGE-type frame

Error 4022: Frame exclusion is not supported yet

Error 4023: This window function may not have a window frame

Error 4024: Argument of NTILE must be greater than 0

Error 4025: CONSTRAINT failed for

Error 4026: Expression in the clause is too big

Error 4027: Got an error evaluating stored expression

Error 4028: Got an error when calculating default value for

Error 4029: Expression for field is referring to uninitialized field

Error 4030: Only one DEFAULT partition allowed

Error 4031: Referenced trigger for the given action time and event type does not exist

Error 4032: Default/ignore value is not supported for such parameter usage

Error 4033: Only row based replication supported for bulk operations

Error 4034: Uncompress the compressed binlog failed

Error 4035: Broken JSON string in argument to function at position

Error 4036: Character disallowed in JSON in argument to function at position

Error 4037: Unexpected end of JSON text in argument to function

Error 4038: Syntax error in JSON text in argument to function at position

Error 4039: Incorrect escaping in JSON text in argument to function at position

Error 4040: Limit on JSON nested structures depth is reached in argument to function at position

Error 4041: Unexpected end of JSON path in argument to function

Error 4042: Syntax error in JSON path in argument to function at position

Error 4043: Limit on JSON path depth is reached in argument to function at position

Error 4044: Wildcards in JSON path not allowed in argument to function

Error 4045: JSON path should end with an array identifier in argument to function

Error 4046: Argument 2 to function must be "one" or "all".

Error 4047: CREATE TEMPORARY TABLE is not allowed with ROW_FORMAT=COMPRESSED or KEY_BLOCK_SIZE.

Error 4048: Incorrect GeoJSON format specified for st_geomfromgeojson function

Error 4049: Incorrect GeoJSON format - too few points for linestring specified

Error 4050: Incorrect GeoJSON format - polygon not closed

Error 4051: Path expression '$' is not allowed in argument to function

Error 4052: A slave with the same server_uuid/server_id as this slave has connected to the master

Error 4053: Flashback does not support

Error 4054: Keys are out order during bulk load

Error 4055: Bulk load rows overlap existing rows

Error 4056: Can't execute updates on master with binlog_format != ROW

Error 4057: MyRocks supports only READ COMMITTED and REPEATABLE READ isolation levels

Error 4058: When unique checking is disabled in MyRocks, INSERT,UPDATE,LOAD statements with clauses

Error 4059: Can't execute updates when you started a transaction with START TRANSACTION WITH CONSIST

Error 4060: This transaction was rolled back and cannot be committed. Only supported operation is to

Error 4061: MyRocks currently does not support ROLLBACK TO SAVEPOINT if modifying rows

Error 4062: Only REPEATABLE READ isolation level is supported for START TRANSACTION WITH CONSISTENT

Error 4063: Unsupported collation on string indexed column

Error 4064: Table '%s' does not exist, but metadata information exists inside MyRocks. This is a sig

Error 4065: Column family flag is different from an existing flag

Error 4066: TTL duration in MyRocks must be an unsigned non-null 64-bit integer

Error 4067: Status error received from RocksDB

Error 4068: Status error received from RocksDB

Error 4069: TTL support is currently disabled when table has a hidden PK

Error 4070: TTL column in MyRocks must be an unsigned non-null 64-bit integer, exist inside the table

Error 4071: The per-index column family option has been deprecated

Error 4072: MyRocks failed creating new key definitions during alter

Error 4073: MyRocks failed populating secondary key during alter

Error 4074: Window functions can not be used as arguments to group functions

Error 4075: OK packet too large

Error 4076: Incorrect GeoJSON format - empty 'coordinates' array

Error 4077: MyRocks doesn't currently support collations with "No pad" attribute

Error 4078: Illegal parameter data types for operation

Error 4079: Illegal parameter data type for operation

Error 4080: Incorrect parameter count to cursor

Error 4081: Unknown structured system variable or ROW routine variable

Error 4082: Row variable does not have a field

Error 4083: END identifier does not match

Error 4084: Sequence has run out

Error 4085: Sequence values are conflicting

Error 4086: Sequence table structure is invalid

Error 4087: Sequence access error

Error 4088: Sequences requires binlog_format mixed or row

Error 4089: is not a SEQUENCE

Error 4090: is not a SEQUENCE

Error 4091: Unknown SEQUENCE

Error 4092: Unknown VIEW

Error 4093: Wrong INSERT into a SEQUENCE. One can only do single table INSERT into a sequence object

Error 4094: At line in

Error 4095: Subroutine is declared in the package specification but is not defined in the package bo

Error 4096: Subroutine has a forward declaration but is not defined

Error 4097: Compressed column can't be used in key specification

Error 4098: Unknown compression method

Error 4099: The used table value constructor has a different number of values

MariaDB Error Codes 4100 to 4199

Error 4100: Field reference can't be used in table value constructor

Error 4101: Numeric datatype is required for function

Error 4102: Argument to the function is not a constant for a partition

Error 4103: Argument to the function does not belong to the range [0,1]

Error 4104: function only accepts arguments that can be converted to numerical types

Error 4105: Aggregate specific instruction (FETCH GROUP NEXT ROW) used in a wrong context

Error 4106: Aggregate specific instruction(FETCH GROUP NEXT ROW) missing from the aggregate function

Error 4107: Limit only accepts integer values

Error 4108: Invisible column must have a default value

Error 4109: Rows matched: Changed: Inserted: Warnings:

Error 4110: must be of type for system-versioned table

Error 4111: Transaction-precise system versioning for is not supported

Error 4112: You should never see it

Error 4113: Wrong partitioning type, expected type

Error 4114: Versioned table last HISTORY partition is out of, need more HISTORY partitions

Error 4115: Maybe missing parameters

Error 4116: Can only drop oldest partitions when rotating by INTERVAL

Error 4117: You should never see it

Error 4118: Partition contains non-historical data

Error 4119: Not allowed for system-versioned. Change @@system_versioning_alter_history to proceed wi

Error 4120: Not allowed for system-versioned. Change to/from native system versioning engine is not

Error 4121: SYSTEM_TIME range selector is not allowed

Error 4122: Conflicting FOR SYSTEM_TIME clauses in WITH RECURSIVE

Error 4123: Table must have at least one versioned column

Error 4124: Table is not system-versioned

Error 4125: Wrong parameters for: missing

Error 4126: PERIOD FOR SYSTEM_TIME must use columns

Error 4127: Wrong parameters for partitioned: wrong value for

Error 4128: Wrong partitions for: must have at least one HISTORY and exactly one last CURRENT

Error 4129: TRX_ID not found in mysql.transaction_registry

Error 4130: Can not change system versioning field

Error 4131: Can not DROP SYSTEM VERSIONING for table partitioned BY SYSTEM_TIME

Error 4132: System-versioned tables in the database are not supported

Error 4133: Transaction registry is disabled

Error 4134: Duplicate ROW column

Error 4135: Table is already system-versioned

Error 4136: You should never see it

Error 4137: System-versioned tables do not support

Error 4138: Transaction-precise system-versioned tables do not support partitioning by ROW START or

Error 4139: The index file for table is full

Error 4140: The column cannot be changed more than once in a single UPDATE statement

Error 4141: Row with no elements is not allowed in table value constructor in this context

Error 4142: SYSTEM_TIME partitions in table does not support historical query

Error 4143: index does not support this operation

Error 4144: Changing table options requires the table to be rebuilt

Error 4145: Can't execute the command as you have a BACKUP STAGE active

Error 4146: You must start backup with "BACKUP STAGE START"

Error 4147: Backup stage is same or before current backup stage

Error 4148: Backup stage failed

Error 4149: Unknown backup stage: Stage should be one of START, FLUSH, BLOCK_DDL, BLOCK_COMMIT or EN

Error 4150: User is blocked because of too many credential errors; unblock with 'FLUSH PRIVILEGES'

Error 4151: Access denied, this account is locked

Error 4152: Application-time period table cannot be temporary

Error 4153: Fields of PERIOD FOR have different types

Error 4154: Cannot specify more than one application-time period

Error 4155: Period field cannot be

Error 4156: Period is not found in table

Error 4157: Column used in period specified in update SET list

Error 4158: Can't DROP CONSTRAINT. Use DROP PERIOD for this

Error 4159: Specified key part was too long; max key part length is

Error 4160: Comment for database is too long

Error 4161: Unknown data type

Error 4162: Operator does not exists

Error 4163: Table history row start is later than row end

Error 4164: STARTS is later than query time, first history partition may exceed INTERVAL value

Error 4165: DDL-statement is forbidden as table storage engine does not support Galera replication

Error 4166: The used command is not allowed because the MariaDB server or client has disabled the lo

Error 4167: No secure transports are configured, unable to set --require_secure_transport=ON

Error 4168: Slave SQL thread ignored the '%s' because table is shared

Error 4169: AUTO_INCREMENT column cannot be used in the UNIQUE index

Error 4170: Key cannot explicitly include column

Error 4171: Key cannot have WITHOUT OVERLAPS

Error 4172: is not allowed in this context

Error 4173: Engine does not support rollback. Changes where committed during rollback call

Error 4174: A primary key cannot be marked as IGNORE

Error 4175: SKIP LOCKED makes this statement unsafe

Error 4176: Field can't be set for JSON_TABLE

Error 4177: Every table function must have an alias

Error 4178: Can't store an array or an object in the scalar column of JSON_TABLE

Error 4179: Can't store multiple matches of the path in the column of JSON_TABLE

Error 4180: FETCH ... WITH TIES requires ORDER BY clause to be present

Error 4181: Dropped orphan trigger, originally created for table

Error 4182: Storage engine is disabled

Error 4183: SFORMAT error

Error 4184: Convert partition is not supported for subpartitioned table

Error 4185: MariaDB tried to use the %s, but its provider plugin is not loaded

Error 4186: Failed to parse histogram for table at offset

Error 4187: OUT or INOUT argument for function is not allowed here

Error 4188: Replicated query table can not be temporary

Error 4189: Versioned table: adding HISTORY partition(s) failed

Error 4190: is implicitly changing the value of from to

Error 4191: CHANGE MASTER TO option is missing requirement

Error 4192: Slave log event execution was interrupted (slave_max_statement_time exceeded)

Error 4193: Invalid value for keyword

Error 4194: keyword is not supported

Error 4195: Variable schema is not supported

Error 4196: Pseudo thread id should not be modified by the client as it will be overwritten

Error 4197: Wrong number of columns

Error 4198: Sequence tables cannot have any keys

Error 4199: Sequence tables cannot have any constraints

MariaDB Error Codes 4200 to 4299

Error 4200: ORDER BY

- Error 4201: The variable is ignored
- Error 4202: Incorrect column name count for derived table
- Error 4203: Values were longer than max_sort_length. Sorting used only the first bytes
- Error 4204: Invalid binary vector format
- Error 4205: Invalid vector format at offset
- Error 4206: Cannot determine distance type for VEC_DISTANCE, index is not found
- Error 4221: Unresolved table name for hint
- Error 4222: Unresolved index name for hint
- Error 4223: Optimizer hints are not supported inside view definitions
- Error 4224: Hint is ignored as malformed
- Error 4225: Optimizer hints at the INSERT part of a INSERT..SELECT statement are not supported
- Error 4226: Slave SQL thread aborted
- Error 4265: LIST partition type does not support INTERVAL
- Error 4266: RANGE COLUMN partition by a DATE with INTERVAL smaller than date
- Error 4267: MAXVALUE is not allowed in range partitioning with interval
- Error 4268: Range partition table %s.%s: adding INTERVAL partition(s) failed
- Error 4269: Failed to parse saved optimizer context: %s at offset %d.
- Error 4270: Failed to match the stats from replay context with the optimizer stats: %s doesn't exist in %s

Client/Server Protocol
- Protocol Data Types
- MariaDB Protocol Differences with MySQL
- 0 - Packet
- 1 - Connecting
  - Connecting
  - Connecting via sha256_password
  - Connecting via caching_sha2_password
- 2 - Text Protocol
  - COM_CHANGE_USER
  - COM_CREATE_DB
  - COM_DEBUG
  - COM_DROP_DB
  - COM_FIELD_LIST
  - COM_INIT_DB
  - COM_PING
  - COM_PROCESSLIST
  - COM_PROCESS_KILL
  - COM_QUERY
  - COM_QUIT
  - COM_RESET_CONNECTION
  - COM_SET_OPTION
  - COM_SHUTDOWN
  - COM_SLEEP
  - COM_STATISTICS
- 3 - Binary Protocol (Prepared Statements)
  - COM_STMT_BULK_EXECUTE
  - COM_STMT_CLOSE
  - COM_STMT_EXECUTE
  - COM_STMT_FETCH
  - COM_STMT_PREPARE
  - COM_STMT_RESET
  - COM_STMT_SEND_LONG_DATA
  - Server Response Packets (Binary Protocol)
    - PACKET_BINDATA
- 4 - Server Response Packets
  - EOF_Packet
  - ERR_Packet
  - LOCAL_INFILE_Packet
  - OK_Packet
  - Result Set Packets
  - Result Set Row
- Replication Protocol
  - 1-Binlog Events
  - 2-Binlog Event Header
  - 3-Binlog Network Stream
  - 4-Semi-Sync Replication
  - 5-Replica Registration
  - ANNOTATE_ROWS_EVENT
  - BEGIN_LOAD_QUERY_EVENT
  - BINLOG_CHECKPOINT_EVENT
  - COM_BINLOG_DUMP
  - COM_REGISTER_SLAVE
  - EXECUTE_LOAD_QUERY_EVENT
  - Fake GTID_LIST Event
  - Fake ROTATE_EVENT
  - FORMAT_DESCRIPTION_EVENT
  - GTID_EVENT
  - GTID_LIST_EVENT
  - HEARTBEAT_LOG_EVENT
  - INTVAR_EVENT
  - QUERY_EVENT
  - RAND_EVENT
  - ROTATE_EVENT
  - ROWS_EVENT_V1/V2, ROWS_COMPRESSED_EVENT_V1
  - START_ENCRYPTION_EVENT
  - STOP_EVENT
  - TABLE_MAP_EVENT
  - USER_VAR_EVENT
  - XA_PREPARE_LOG_EVENT
  - XID_EVENT

Product Development
- MariaDB Quality Development Rules
- Uploading Package to PPA
- Debugging & Troubleshooting
  - Troubleshooting Overview
  - MariaDB Crashes
  - MariaDB Error Messages
  - Debugging Memory Usage
  - Compiling MariaDB for Debugging
  - Creating a Trace File
  - Debugging a Running Server (on Linux)
  - Debugging MariaDB With a Debugger
  - Enabling Core Dumps
  - Producing a Full Stack Trace for mariadbd
  - Using procmon to Trace mysqld.exe Filesystem Access
  - Limitations/Differences with a MariaDB Server Compiled for Debugging
- Plugin Development
  - Writing Plugins for MariaDB
  - Encryption Plugin Development
  - Audit Plugin Development
  - Authentication Plugin Development
  - Information Schema Plugins: SHOW and FLUSH Statements
  - Replication & Cluster Plugin Development
  - Password Validation Plugin Development
  - Storage Engine Development
    - Engine-defined New Table/Field/Index Attributes
    - Table Discovery
- Server Development
  - MariaDB Server Roadmap
  - Working with GitHub
  - Quality Assurance
    - Code Coverage

- Code Coverage With dgcov
- QA Tests
- QA Metrics
- QA Tools
- Optimizer Quality
- QA - Aria Recovery
- Worklog Quality Checklist Template
- Random Query Generator Tests
- RQG Extensions for MariaDB
- Benchmarks and Long Running Tests
  - Benchmark Results
  - sysbench v0.5 - Single Five Minute Runs on T500 Laptop
  - sysbench v0.5 - Three Times Five Minutes Runs on work with 5.1.42
  - sysbench v0.5 - 3x Five Minute Runs on work with 5.2-wl86
  - sysbench v0.5 - 3x Five Minute Runs on work with 5.1 vs. 5.2-wl86
  - sysbench v0.5 - 3x 15 Minute Runs on perro with 5.2-wl86 a
  - sysbench v0.5 - 3x 15 Minute Runs on perro with 5.2-wl86 b
  - Select Random Ranges and Select Random Point
  - Sysbench Results
  - sysbench v0.5 - Single Five Minute Runs on perro
  - sysbench v0.5 - Single Five Minute Runs on work
  - Threadpool Benchmarks
  - Benchmarks
  - Benchmark Builds
  - Benchmarking Aria
  - DBT3 Automation Scripts
  - DBT3 Benchmark Results InnoDB
  - DBT3 Benchmark Results MyISAM
  - DBT3 Example Preparation Time
  - MariaDB 5.3 - Asynchronous I/O on Windows with InnoDB
  - MariaDB 5.3/MySQL 5.5 Windows performance patches
  - mariadb-tools
  - Performance of MEMORY Tables
  - Recommended Settings for Benchmarks
  - RQG Performance Comparisons
  - run-sql-bench.pl
  - Segmented Key Cache Performance
  - sysbench Benchmark Setup
- InnoDB Upgrade Tests
  - 10.0.32 Release Upgrade Tests
  - 10.0.34 Release Upgrade Tests
  - 10.0.36 Release Upgrade Tests
  - 10.1.22 Pre-release Upgrade Tests
  - 10.1.25 Release Upgrade Tests
  - 10.1.26 Release Upgrade Tests
  - 10.1.30 Release Upgrade Tests
  - 10.1.32 Release Upgrade Tests
  - 10.1.36 Release Upgrade Tests
  - 10.2.12 Release Upgrade Tests
  - 10.2.13 Release Upgrade Tests
  - 10.2.14 Release Upgrade Tests
  - 10.2.17 Release Upgrade Tests
  - 10.2.18 Release Upgrade Tests
  - 10.2.5 Pre-release Upgrade Tests
  - 10.2.7 Release Upgrade Tests
  - 10.3.10 Release Upgrade Tests
  - 10.3.3-beta Release Upgrade Tests
  - 10.3.4-beta Release Upgrade Tests
  - 10.3.5-gamma Release Upgrade Tests
  - 10.3.6-gamma Release Upgrade Tests
  - 10.3.9 Release Upgrade Tests
- QA Datasets
  - DBT-3 Dataset
  - DBT-3 Queries
  - OpenStreetMap Dataset
  - osmdb06.sql
- MariaDB Development Tools
  - Jira - Status and Transition Definitions
  - Jira - Project Planning and Tracking
  - Profiling with Linux perf tool
  - Worklog
  - Buildbot
    - About Buildbot
    - Branch Protection Using Buildbot
    - Buildbot Database Schema
    - Buildbot Development
    - Buildbot ToDo
    - Package Testing with Buildbot and KVM
    - Buildbot runvm
    - Buildbot Setup
    - Buildbot Setup Notes
    - Buildbot Setup for macOS
    - Buildbot Setup for Windows
    - Buildbot Setup for BSD
    - Buildbot Setup for Solaris Sparc
    - Buildbot Setup for Solaris x86
    - Buildbot Setup for Ubuntu-Debian
    - Buildbot Setup for VM Host
    - Setup of a Buildbot Build Slave
    - Buildbot Setup for Virtual Machines
    - Buildbot Setup for Virtual Machines - FreeBSD 9.2
    - Buildbot Setup for Virtual Machines - General Principles
    - Buildbot Setup for Virtual Machines - Additional Steps
    - Install cmake on Build VMs
    - Installing Correct Libraries for PAM and readline
    - Installing the Boost Library Needed for the OQGraph Storage Engine
    - Installing VM Images for Testing .deb Upgrade Between Versions
    - Resizing a Virtual Machine Image
    - Update Debian 4 Mirrors for Buildbot VMs
    - Buildbot Setup for Virtual Machines - Debian
    - Buildbot Setup for Virtual Machines - Debian 4 amd64
    - Buildbot Setup for Virtual Machines - Debian 4 i386
    - Buildbot Setup for Virtual Machines - Debian 5 amd64
    - Buildbot Setup for Virtual Machines - Debian 5 i386
    - Buildbot Setup for Virtual Machines - Debian 6 amd64
    - Buildbot Setup for Virtual Machines - Debian 6 i386
    - Buildbot Setup for Virtual Machines - Red Hat
    - Buildbot Setup for Virtual Machines - Centos 5 amd64
    - Buildbot Setup for Virtual Machines - CentOS 5 i386
    - Buildbot Setup for Virtual Machines - CentOS 6.2
    - Buildbot Setup for Virtual Machines - Fedora 16
    - Buildbot Setup for Virtual Machines - Fedora 17
    - Buildbot Setup for Virtual Machines - Fedora 18
    - Buildbot Setup for Virtual Machines - Fedora 19
    - Buildbot Setup for Virtual Machines - Fedora 20
    - Buildbot Setup for Virtual Machines - Red Hat 6 x86
    - Buildbot Setup for Virtual Machines - Ubuntu
    - Buildbot Setup for Virtual Machines - Ubuntu 10.04 (alpha), i386 and amd64
    - Buildbot Setup for Virtual Machines - Ubuntu 10.10 "maverick"
    - Buildbot Setup for Virtual Machines - Ubuntu 11.04 "natty"
    - Buildbot Setup for Virtual Machines - Ubuntu 11.10 "oneiric"
    - Buildbot Setup for Virtual Machines - Ubuntu 12.04 "precise"
    - Buildbot Setup for Virtual Machines - Ubuntu 12.10 "quantal"
    - Buildbot Setup for Virtual Machines - Ubuntu 13.04 "raring"
    - Buildbot Setup for Virtual Machines - Ubuntu 13.10 "saucy"

- Buildbot Setup for Virtual Machines - Ubuntu 14.04 "trusty"
- Buildbot Setup for Virtual Machines - Ubuntu 8.04, 8.10, and 9.10
- Buildbot Setup for Virtual Machines - Ubuntu 9.04 amd64
- Buildbot Setup for Virtual Machines - Ubuntu 9.04 i386
- Using Git With MariaDB
- Using Git
- Configuring Git to Send Commit Notices

MariaDB Internals
- Merging into MariaDB
- Creating a New Merge Tree
- Merging with a Merge Tree

MariaDB Source Code Internals
- Stored Procedure Internals
- Connection Memory Usage

Using MariaDB with Your Programs (API)
- Progress Reporting
- libmysqld
- Embedded MariaDB Interface
- The mariadb-test and mariadb-test-embedded Programs
- Non-Blocking Client Library
- Non-Blocking Operation in the Client Library
- Non-Blocking API Reference
- Using the Non-Blocking Library

MariaDB Server Documentation

> **Note:**
>
> 📄 Read offline: mariadb-server.pdf (~6,200 pages, 80 MB). The PDF is a point-in-time snapshot; this site is always current.

See Quickstart Guides.

Get started quickly with MariaDB Server using these quickstart guides. Follow step-by-step instructions to install, configure, and begin using MariaDB for your projects.

See Server Usage.

Learn how to effectively use MariaDB Server. This section covers SQL statements, built-in functions, client utilities, and best practices for daily database operations.

See Server Management.

Effectively managing your MariaDB Server is key to ensuring its reliability, performance, and security. This section serves as your central hub for all aspects of MariaDB Server management.

See Security.

Secure your MariaDB Server. This section provides comprehensive guidance on user management, encryption, authentication, auditing, and other crucial security measures.

See Architecture.

Understand MariaDB Server's architecture. Explore its components, storage engines, and how they interact to provide a high-performance, reliable database solution.

See Clients & Utilities.

Discover MariaDB Server's clients and utilities. This section guides you through tools for connecting, managing, and interacting with your database, from command-line clients to graphical interfaces.

See HA & Performance.

Optimize MariaDB Server for high availability and performance. Learn about replication, clustering, load balancing, and configuration tuning for robust and efficient database solutions.

See Reference.

Access the comprehensive MariaDB Server reference. Find detailed documentation on SQL syntax, data types, functions, system variables, and other technical specifications.

This page is licensed: CC BY-SA / Gnu FDL

Quickstart Guides

See Installing MariaDB Server Guide.

Official MariaDB Server install guide: Linux apt/dnf/yum commands, mariadb-secure-installation setup, systemctl status/start checks, Windows .msi installer.

See Adding & Changing Data Guide.

This guide provides a walkthrough of the INSERT, UPDATE, and DELETE statements, demonstrating how to add, modify, and remove data in tables.

See Essential Queries Guide.

Learn how to perform essential SQL operations such as creating tables, inserting data, and using aggregate functions like MAX, MIN, and AVG.

See Basics Guide.

Complete MariaDB basics guide: connect with mariadb -u/-p/-h, CREATE DATABASE/USE, CREATE TABLE with AUTO_INCREMENT, INSERT/SELECT/UPDATE/DELETE commands.

See Altering Tables Guide.

Learn how to modify existing table structures using the ALTER TABLE statement, including adding columns, changing types, and managing indexes.

See Connecting to MariaDB Guide.

This guide details how to connect to a MariaDB server using the command-line client, covering options for host, user, password, and protocol.

See Troubleshooting Connection Issues Guide.

Diagnose and fix common MariaDB Server connection problems, such as “Can’t connect to local server” and access-denied errors, with step-by-step troubleshooting.

See Doing Time Guide.

Understand how to work with date and time values in MariaDB, including data types like DATETIME and TIMESTAMP, and useful temporal functions.

See Importing Data Guide.

Learn how to efficiently import data into MariaDB tables from external files using the LOAD DATA INFILE statement.

See Essentials of an Index Guide.

This guide provides a conceptual overview of database indexes, explaining their purpose, different types, and when to use them for optimization.

See Getting Started with Indexes Guide.

Definitive MariaDB indexes guide: PRIMARY KEY, UNIQUE INDEX, INDEX, FULLTEXT types, CREATE/ALTER TABLE syntax, CREATE INDEX, SHOW INDEX, and EXPLAIN.

See Joining Tables with JOIN Clauses.

This guide introduces the different types of JOINs (INNER, LEFT, RIGHT, CROSS) and demonstrates how to combine data from multiple tables.

See Advanced Joins.

Explore complex join scenarios. This guide covers filtering joined data with WHERE clauses, handling dates, and aggregating results from multiple tables for deeper analysis.

See Configuring MariaDB for Remote Client Access Guide.

Configure MariaDB Server to accept remote connections by setting bind-address, granting privileges for remote users, and opening the right firewall rules.

See Getting Data Guide.

This guide explains the SELECT statement in detail, covering how to retrieve, filter, limit, and sort data from your MariaDB database.

See Basic SQL Statements Guide.

A quick reference for core SQL statements including DDL (CREATE, DROP), DML (INSERT, UPDATE, DELETE), and TCL (COMMIT, ROLLBACK) commands.

See Basic SQL Debugging Guide.

This guide offers conventions and practical tips for designing SQL queries that are easier to read, understand, and debug.

See MariaDB String Functions Guide.

This guide goes through several built-in string functions in MariaDB, grouping them by similar features, and providing examples of how they might be used.

See Restoring Data from Dump Files Guide.

Restore MariaDB data from mariadb-dump backup files using the mariadb client, including how to selectively restore a single table.

See Changing Times in MariaDB.

This guide explores MariaDB functions for performing calculations and modifications on date and time values, like DATE_ADD and DATE_SUB.

See System & Status Variables Guide.

This guide indicates where the various system and status variables of MariaDB Server are found.

See Making Backups with mariadb-dump Guide.

Create logical backups of MariaDB databases with the mariadb-dump utility, covering how to back up all databases, specific databases, or individual tables.

See A MariaDB Primer Guide.

A beginner-friendly primer on using the mariadb command-line client to log in, create databases, and execute basic SQL commands.

See Creating & Using Views Guide.

Discover how to create and use views to simplify complex queries, restrict data access, and present a specific perspective of your data.

See Database Applications.

This section offers advice on writing and maintaining applications that use databases, covering schema design, code practices, and testing.

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB Server Guide

This guide provides step-by-step instructions for installing MariaDB Server on various operating systems, including package updates and security settings.

For Linux (Ubuntu/Debian/Red Hat-based distributions)

The most common way to install MariaDB on Linux is through your system's package manager.

Steps:

1. Update Package List:

   Before installing, it's a good practice to update your package index.

   - For Debian/Ubuntu:Bash

     ```
     sudo apt update
     ```

   - For Red Hat/CentOS/Fedora:Bash

     ```
     sudo yum update # For older systems
     sudo dnf update # For newer systems
     ```

2. Install MariaDB Server:

   Install the MariaDB server and client packages.

   - For Debian/Ubuntu:Bash

     ```
     sudo apt install mariadb-server mariadb-client galera-4
     ```

   - For Red Hat/CentOS/Fedora:Bash

     ```
     sudo dnf install mariadb mariadb-server
     ```

3. Secure the Installation:

   After installation, run the security script to set a root password, remove anonymous users, and disable remote root login.

   ```
   sudo mariadb-secure-installation
   ```

   Follow the prompts to configure your security settings.

4. Start and Verify the Service:

   MariaDB typically starts automatically after installation. You can check its status and manually start it if needed.

   - Check status:

     ```
     sudo systemctl status mariadb
     ```

   - Start service (if not running):Bash

     ```
     sudo systemctl start mariadb
     ```

   - Verify installation by connecting as root:Bash

     ```
     mariadb -u root -p
     ```

Enter the root password you set during the secure installation.

For Windows

For Windows, MariaDB provides an `.msi` installer for a straightforward graphical installation.

Steps:

1. Download MariaDB:

   Visit the MariaDB downloads page to get the latest `.msi` installer.

2. Run the Installer:

   Double-click the downloaded `.msi` file to start the installation wizard.

3. Follow On-Screen Instructions:

   The installer will guide you through the process, including:

   - Accepting the end-user license agreement.
   - Selecting features and the installation directory.
   - Setting a password for the `root` user.
   - Configuring MariaDB as a service and setting the port (default is 3306).
   - Optionally, enabling UTF8 as the default server character set.

Important Notes:

- **Firewall:** Ensure your firewall is configured to allow connections to MariaDB on the appropriate port (default 3306) if you need remote access.
- **Root Password:** Always set a strong root password during the secure installation step.
- **Further Configuration:** For production environments, you may need to adjust further settings in the MariaDB configuration files (e.g., `my.cnf` on Linux).

Additional Resources:

- Get Started with MariaDB
- How To Install MariaDB on Ubuntu 22.04 - DigitalOcean
- Install MariaDB - MariaDBTutorial.com

This page is licensed: CC BY-SA / Gnu FDL

Adding & Changing Data Guide

This guide explains how to add new data and modify existing data in MariaDB using `INSERT`, `REPLACE`, and `UPDATE` statements. Learn about various options for handling duplicates, managing statement priorities, inserting data from other tables, and performing conditional updates.

Adding Data with `INSERT`

The `INSERT` statement is used to add new rows to a table.

Basic Syntax:

If providing values for all columns in their defined order:

```
INSERT table1 VALUES('value1','value2','value3');
```

The number of values must match the number of columns in `table1`.

Specifying Columns:

It's good practice to specify the columns you are inserting data into, which also allows you to insert columns in any order or omit columns that have default values or allow NULL.

```
INSERT INTO table1 (col3, col1) VALUES('value_for_col3', 'value_for_col1');
```

- The `INTO` keyword is optional but commonly used for readability.
- If a column is not listed and is an `AUTO_INCREMENT` key, its value will be generated. For other omitted columns, their `DEFAULT` value will be used, or `NULL` if allowed. You can explicitly insert a default value using the `DEFAULT` keyword in the `VALUES` list for a specific column.

Multiple Row Inserts:

Insert multiple rows in a single statement for efficiency:

```
INSERT INTO table2 (id_col, data_col1, data_col2) VALUES
  ('id1', 'text_a', 'text_b'),
  ('id2', 'text_c', 'text_d'),
  ('id3', 'text_e', 'text_f');
```

The `VALUES` keyword is used only once, with each row's values enclosed in parentheses and separated by commas.

Handling Duplicates with INSERT IGNORE:

If you attempt to insert a row that would cause a duplicate value in a PRIMARY KEY or UNIQUE index, an error normally occurs, and the row (and potentially subsequent rows in a multi-row insert) might not be inserted.

Using IGNORE tells MariaDB to discard the duplicate row(s) and continue inserting any other valid rows without generating an error.

```
INSERT IGNORE INTO table2 (unique_id_col, data_col) VALUES
  ('id1', 'some_data'),        -- Will be inserted if new
  ('id2', 'other_data'),       -- Will be inserted if new
  ('id1', 'duplicate_data');  -- Will be ignored if 'id1' already exists or was just inserted
```

Managing `INSERT` Priority and Behavior

LOW_PRIORITY:

An INSERT statement normally takes priority over SELECT statements, potentially locking the table and making other clients wait. LOW_PRIORITY makes the INSERT wait until no other clients are reading from the table.

```
INSERT LOW_PRIORITY INTO table1 VALUES('value1','value2','value3');
```

- Once the `LOW_PRIORITY` insert begins, it will lock the table as usual. New read requests that arrive while it's waiting will be processed before it.

DELAYED:

`INSERT DELAYED` lets the server queue the insert request and return control to the client immediately; the rows are written when the table is not in use, and inserts from many clients are batched together.

```
INSERT DELAYED INTO table1 VALUES('value1','value2','value3');
```

- `DELAYED` works only with non-transactional storage engines — `MyISAM`, `MEMORY`, `ARCHIVE`, `BLACKHOLE`, non-transactional `Aria`, and `OQGRAPH`. Using it with another engine returns an error.
- It is controlled by the `max_delayed_threads` system variable; setting that to `0` disables `DELAYED`.

- Because control returns before the rows are written, there is no confirmation of success, and queued rows held in memory are lost if the server stops before they are written. See INSERT DELAYED for the full list of limitations.

Inserting Data from Another Table (`INSERT...SELECT`)

You can insert rows into a table based on data retrieved from another table (or tables) using `INSERT ... SELECT`.

```
INSERT INTO softball_team (last_name, first_name, telephone)
  SELECT name_last, name_first, tel_home
  FROM company_database.employees
  WHERE is_on_softball_team = 'Y';
```

- The columns in the `INSERT INTO softball_team (...)` list must correspond in number and general data type compatibility to the columns in the `SELECT` list.
- `INSERT...SELECT` statements generally cannot operate on the exact same table as both the target and the source directly without mechanisms like temporary tables or certain subquery structures.

Replacing Data with `REPLACE`

The `REPLACE` statement works like `INSERT`, but if a new row has the same value as an existing row for a `PRIMARY KEY` or a `UNIQUE` index, the existing row is deleted before the new row is inserted. If no such conflict exists, it acts like a normal `INSERT`.

```
REPLACE LOW_PRIORITY INTO table2 (id_col, data_col1, data_col2) VALUES
  ('id1', 'new_text_a', 'new_text_b'), -- If 'id1' exists, old row is deleted, this is inserted
  ('id2', 'new_text_c', 'new_text_d'), -- If 'id2' doesn't exist, this is inserted
  ('id3', 'new_text_e', 'new_text_f');
```

- Flags like `LOW_PRIORITY` work similarly to `INSERT`.
- `REPLACE` also supports the `REPLACE ... SELECT` syntax.
- Because `REPLACE` performs a delete then an insert, any columns in the table not specified in the `REPLACE` statement will receive their default values for the newly inserted row, not values from the old row.

Modifying Data with `UPDATE`

Use the `UPDATE` statement to change data in existing rows.

Basic Syntax:

```
UPDATE table3
SET col1 = 'new_value_a', col2 = 'new_value_b'
WHERE id_column < 100;
```

- The `SET` clause specifies which columns to modify and their new values.
- The `WHERE` clause is crucial; it determines which rows are updated. **Without a `WHERE` clause, all rows in the table will be updated.**
- `LOW_PRIORITY` and `IGNORE` (to ignore errors like unique key violations during update, allowing other valid row updates to proceed) can also be used with `UPDATE`.

Using Current Column Values in an Update:

You can use a column's current value in the calculation for its new value.

```
UPDATE table5
SET event_date = DATE_ADD(event_date, INTERVAL 1 DAY)
WHERE DAYOFWEEK(event_date) = 1; -- Example: Add 1 day if event_date is a Sunday
```

ORDER BY and LIMIT with UPDATE:

You can control the order in which rows are updated and limit the number of rows affected (for single-table updates).

```
UPDATE LOW_PRIORITY table3
SET col1 = 'updated_text_a', col2 = 'updated_text_b'
WHERE status_column = 'pending'
ORDER BY creation_date DESC
LIMIT 10;
```

This updates the 10 most recently created ‘pending’ rows.

Multi-Table UPDATE:

You can update rows in one table based on values from another table by joining them.

```
UPDATE products p
JOIN stock_levels s ON p.product_id = s.product_id
SET p.stock_count = s.current_stock
WHERE s.warehouse_id = 'WHA';
```

- Here, `products.stock_count` is updated using values from `stock_levels`.
- `ORDER BY` and `LIMIT` are generally not allowed with multi-table `UPDATE` statements in this direct join form.

Conditional Inserts or Updates (`INSERT ... ON DUPLICATE KEY UPDATE`)

This powerful feature allows you to `INSERT ... ON DUPLICATE KEY UPDATE` a new row, but if a duplicate key (Primary or Unique) conflict occurs, it performs an `UPDATE` on the existing row instead.

```
INSERT INTO table1 (id, col1, col2, status_column)
VALUES ('1012', 'some_text', 'other_text', 'new')
ON DUPLICATE KEY UPDATE status_column = 'old', col2 = VALUES(col2);
```

- If `id` ‘1012’ does not exist, the row is inserted with `status_column = 'new'`.
- If `id` ‘1012’ already exists, the existing row is updated: `status_column` is set to ‘old’, and `col2` is updated with the value that *would have been inserted* for `col2` (using `VALUES(col2)`).
- The `IGNORE` keyword can be used with `INSERT ... ON DUPLICATE KEY UPDATE` to ignore errors that might occur during the `UPDATE` part if the update itself causes a problem (though this is less common). If `IGNORE` is used with `INSERT` and `ON DUPLICATE KEY UPDATE` is also present, `IGNORE` only applies to the `INSERT` part, not the `UPDATE` part.

Further Data Modification Methods

Beyond these SQL statements, MariaDB offers bulk methods for adding data, such as:

- `LOAD DATA INFILE`: For importing data from text files.
- `mariadb-import` utility: A command-line tool that uses `LOAD DATA INFILE`. These are covered in the Importing Data Guide.

This page is licensed: CC BY-SA / Gnu FDL

Essential Queries Guide

The Essential Queries Guide offers a concise collection of commonly-used SQL queries. It’s designed to help developers and database administrators quickly find syntax and examples for typical database operations, from table creation and data insertion to effective data retrieval and manipulation.

Creating a Table

To create new tables:

```
CREATE TABLE t1 ( a INT );
```

```
CREATE TABLE t2 ( b INT );
```

```
CREATE TABLE student_tests (
 name CHAR(10), test CHAR(10),
 score TINYINT, test_date DATE
);
```

For more details, see the official CREATE TABLE documentation.

Inserting Records

To add data into your tables:

```
INSERT INTO t1 VALUES (1), (2), (3);
```

```
INSERT INTO t2 VALUES (2), (4);
```

```
INSERT INTO student_tests
 (name, test, score, test_date) VALUES
 ('Chun', 'SQL', 75, '2012-11-05'),
 ('Chun', 'Tuning', 73, '2013-06-14'),
 ('Esben', 'SQL', 43, '2014-02-11'),
 ('Esben', 'Tuning', 31, '2014-02-09'),
 ('Kaolin', 'SQL', 56, '2014-01-01'),
 ('Kaolin', 'Tuning', 88, '2013-12-29'),
 ('Tatiana', 'SQL', 87, '2012-04-28'),
 ('Tatiana', 'Tuning', 83, '2013-09-30');
```

For more information, see INSERT.

Using AUTO_INCREMENT

The `AUTO_INCREMENT` attribute automatically generates a unique identity for new rows.

Create a table with an `AUTO_INCREMENT` column:

```
CREATE TABLE student_details (
 id INT NOT NULL AUTO_INCREMENT, name CHAR(10),
 date_of_birth DATE, PRIMARY KEY (id)
);
```

When inserting, omit the `id` field; it will be automatically generated:

```
INSERT INTO student_details (name,date_of_birth) VALUES
 ('Chun', '1993-12-31'),
 ('Esben','1946-01-01'),
 ('Kaolin','1996-07-16'),
 ('Tatiana', '1988-04-13');
```

Verify the inserted records:

```
SELECT * FROM student_details;
```

```
+----+---------+---------------+
| id | name    | date_of_birth |
+----+---------+---------------+
|  1 | Chun    | 1993-12-31    |
|  2 | Esben   | 1946-01-01    |
|  3 | Kaolin  | 1996-07-16    |
|  4 | Tatiana | 1988-04-13    |
+----+---------+---------------+
```

For more details, see the AUTO_INCREMENT documentation.

Querying from Two Tables on a Common Value (JOIN)

To combine rows from two tables based on a related column:

```
SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.b;
```

This type of query is a join. For more details, consult the documentation on JOINS.

Finding the Maximum Value

To find the maximum value in a column:

```
SELECT MAX(a) FROM t1;
```

```
+--------+
| MAX(a) |
+--------+
|      3 |
+--------+
```

See the MAX() function documentation. For a grouped example, refer to *Finding the Maximum Value and Grouping the Results* below.

Finding the Minimum Value

To find the minimum value in a column:

```
SELECT MIN(a) FROM t1;
```

```
+--------+
| MIN(a) |
+--------+
|      1 |
+--------+
```

See the MIN() function documentation.

Finding the Average Value

To calculate the average value of a column:

```
SELECT AVG(a) FROM t1;
```

```
+--------+
| AVG(a) |
+--------+
| 2.0000 |
+--------+
```

See the AVG() function documentation.

Finding the Maximum Value and Grouping the Results

To find the maximum value within groups:

```
SELECT name, MAX(score) FROM student_tests GROUP BY name;
```

```
+---------+------------+
| name    | MAX(score) |
+---------+------------+
| Chun    |         75 |
| Esben   |         43 |
| Kaolin  |         88 |
| Tatiana |         87 |
+---------+------------+
```

Further details are available in the MAX() function documentation.

Ordering Results

To sort your query results (e.g., in descending order):

```
SELECT name, test, score FROM student_tests
 ORDER BY score DESC; -- Use ASC for ascending order
```

```
+---------+--------+-------+
| name    | test   | score |
+---------+--------+-------+
| Kaolin  | Tuning |    88 |
| Tatiana | SQL    |    87 |
| Tatiana | Tuning |    83 |
| Chun    | SQL    |    75 |
| Chun    | Tuning |    73 |
| Kaolin  | SQL    |    56 |
| Esben   | SQL    |    43 |
| Esben   | Tuning |    31 |
+---------+--------+-------+
```

For more options, see the ORDER BY documentation.

Finding the Row with the Minimum of a Particular Column

To find the entire row containing the minimum value of a specific column across all records:

```
SELECT name, test, score FROM student_tests
 WHERE score = (SELECT MIN(score) FROM student_tests);
```

```
+-------+--------+-------+
| name  | test   | score |
+-------+--------+-------+
| Esben | Tuning |    31 |
+-------+--------+-------+
```

Finding Rows with the Maximum Value of a Column by Group

To retrieve the full record for the maximum value within each group (e.g., highest score per student):

```
SELECT name, test, score FROM student_tests st1
 WHERE score = (SELECT MAX(st2.score) FROM student_tests st2 WHERE st1.name = st2.name);
```

```
+---------+--------+-------+
| name    | test   | score |
+---------+--------+-------+
| Chun    | SQL    |    75 |
| Esben   | SQL    |    43 |
| Kaolin  | Tuning |    88 |
| Tatiana | SQL    |    87 |
+---------+--------+-------+
```

Calculating Age

Use the `TIMESTAMPDIFF` function to calculate age from a birth date.

To see the current date (optional, for reference):

```
SELECT CURDATE() AS today;
```

```
+------------+
| today      |
+------------+
| 2014-02-17 | -- Example output; actual date will vary
+------------+
```

To calculate age as of a specific date (e.g., '2014-08-02'):

```
SELECT name, date_of_birth, TIMESTAMPDIFF(YEAR, date_of_birth, '2014-08-02') AS age
  FROM student_details;
```

```
+---------+---------------+------+
| name    | date_of_birth | age  |
+---------+---------------+------+
| Chun    | 1993-12-31    |   20 |
| Esben   | 1946-01-01    |   68 |
| Kaolin  | 1996-07-16    |   18 |
| Tatiana | 1988-04-13    |   26 |
+---------+---------------+------+
```

To calculate current age, replace the specific date string (e.g., '2014-08-02') with CURDATE().

See the TIMESTAMPDIFF() documentation for more.

Using User-Defined Variables

User-defined variablescan store values for use in subsequent queries within the same session.

Example: Set a variable for the average score and use it to filter results.

```
SELECT @avg_score := AVG(score) FROM student_tests;
```

```
+-------------------------+
| @avg_score:= AVG(score) |
+-------------------------+
|            67.000000000 |
+-------------------------+
```

```
SELECT * FROM student_tests WHERE score > @avg_score;
```

```
+---------+--------+-------+------------+
| name    | test   | score | test_date  |
+---------+--------+-------+------------+
| Chun    | SQL    |    75 | 2012-11-05 |
| Chun    | Tuning |    73 | 2013-06-14 |
| Kaolin  | Tuning |    88 | 2013-12-29 |
| Tatiana | SQL    |    87 | 2012-04-28 |
| Tatiana | Tuning |    83 | 2013-09-30 |
+---------+--------+-------+------------+
```

Example: Add an incremental counter to a result set.

```
SET @count = 0;
```

```
SELECT @count := @count + 1 AS counter, name, date_of_birth FROM student_details;
```

```
+---------+---------+---------------+
| counter | name    | date_of_birth |
+---------+---------+---------------+
|       1 | Chun    | 1993-12-31    |
|       2 | Esben   | 1946-01-01    |
|       3 | Kaolin  | 1996-07-16    |
|       4 | Tatiana | 1988-04-13    |
+---------+---------+---------------+
```

See User-defined Variables for more.

View Tables in Order of Size

To list all tables in the current database, ordered by their size (data + index) in megabytes:

```
SELECT table_schema AS `DB`, table_name AS `TABLE`,
  ROUND(((data_length + index_length) / 1024 / 1024), 2) `Size (MB)`
  FROM information_schema.TABLES
  WHERE table_schema = DATABASE() -- This clause restricts results to the current database
  ORDER BY (data_length + index_length) DESC;
```

```
+--------------------+---------------------------------------+-----------+
| DB                 | Table                                 | Size (MB) | -- Example Output
+--------------------+---------------------------------------+-----------+
| your_db_name       | some_large_table                      |      7.05 |
| your_db_name       | another_table                         |      6.59 |
...
+--------------------+---------------------------------------+-----------+
```

Removing Duplicates

To remove duplicate rows based on specific column values, while keeping one instance (e.g., the instance with the highest `id`).

This example assumes `id` is a unique primary key and duplicates are identified by the values in column `f1`. It keeps the row with the maximum `id` for each distinct `f1` value.

Setup sample table and data:

```
CREATE TABLE t (id INT, f1 VARCHAR(2));
```

```
INSERT INTO t VALUES (1,'a'), (2,'a'), (3,'b'), (4,'a');
```

To delete duplicate rows, keeping the one with the highest `id` for each group of `f1` values:

```
DELETE t_del FROM t AS t_del
INNER JOIN (
    SELECT f1, MAX(id) AS max_id
    FROM t
    GROUP BY f1
    HAVING COUNT(*) > 1 -- Identify groups with actual duplicates
) AS t_keep ON t_del.f1 = t_keep.f1 AND t_del.id < t_keep.max_id;
```

This query targets rows for deletion (`t_del`) where their `f1` value matches an `f1` in a subquery (`t_keep`) that has duplicates, and their `id` is less than the maximum `id` found for that `f1` group.

Verify results after deletion:

```
SELECT * FROM t;
```

```
+------+------+
| id   | f1   |
+------+------+
|    3 | b    |
|    4 | a    |
+------+------+
```

This page is licensed: CC BY-SA / Gnu FDL

Basics Guide

The quickstart guide walks you through connecting to a MariaDB server, creating your initial database and table structures, and performing fundamental data operations. It's designed for new users or anyone needing a quick refresher on essential MariaDB commands and basic syntax.

Connecting to MariaDB Server

To interact with the MariaDB server, use a client program. The default command-line client is `mariadb` .

Connect to MariaDB in monitor mode from the Linux command-line:

```
mariadb -u root -p -h localhost
```

Common options:

- `-u username` : Specifies the MariaDB user (e.g., `root`). This is not the OS user.
- `-p` : Prompts for the password. If no password is set, press [Enter].
- `-h hostname_or_IP` : Specifies the server's hostname or IP address if the client is on a different machine than the server. Often not needed if connecting locally.

If logged into Linux as `root` , you might only need:

```
mariadb -p
```

To exit the `mariadb` monitor, type `quit` or `exit` and press [Enter].

Creating a Database Structure

First, create and select a database.

```
CREATE DATABASE bookstore;
USE bookstore;
```

This creates a database named `bookstore` and sets it as the default for subsequent operations.

Next, create tables to hold data.

```
CREATE TABLE books (
    isbn CHAR(20) PRIMARY KEY,
    title VARCHAR(50),
    author_id INT,
    publisher_id INT,
    year_pub CHAR(4),
    description TEXT
);
```

This statement creates a `books` table with six columns:

- `isbn`: `CHAR(20)`, the primary key for unique identification.
- `title`: `VARCHAR(50)`, a variable-width string for the book title.
- `author_id`, `publisher_id`: `INT`, for storing numeric IDs.
- `year_pub`: `CHAR(4)`, a fixed-width string for the publication year.
- `description`: `TEXT`, for longer descriptive text (up to 65,535 bytes).

To view the structure of a created table:

```
DESCRIBE books;
```

```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| isbn         | char(20)    | NO   | PRI | NULL    |       |
| title        | varchar(50) | YES  |     | NULL    |       |
| author_id    | int(11)     | YES  |     | NULL    |       |
| publisher_id | int(11)     | YES  |     | NULL    |       |
| year_pub     | char(4)     | YES  |     | NULL    |       |
| description  | text        | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

To modify an existing table, use the `ALTER TABLE` statement (see ALTER TABLE documentation). To delete a table and all its data (irreversibly), use `DROP TABLE table_name;` (see DROP TABLE documentation).

Example of another table, `authors`, using `AUTO_INCREMENT` for the primary key:

```
CREATE TABLE authors (
    author_id INT AUTO_INCREMENT PRIMARY KEY,
    name_last VARCHAR(50),
    name_first VARCHAR(50),
    country VARCHAR(50)
);
```

The `author_id` will automatically generate a unique number for each new author.

SQL Syntax Notes

- SQL statements typically end with a semicolon (`;`) or `\G`.
- Statements can span multiple lines; execution occurs after the terminating character and [Enter].
- To cancel a partially typed statement in the `mariadb` client, enter `\c` and press [Enter].
- SQL reserved words (e.g., `CREATE`, `SELECT`) are often written in uppercase for readability but are case-insensitive in MariaDB.
- Database and table names are case-sensitive on Linux systems (as they map to directories and files) but generally not on Windows. Column names are case-insensitive.

- Using lowercase for table and column names is a common convention.

Entering Data

Use the `INSERT` statement (see INSERT documentation) to add new rows to a table.

```
INSERT INTO authors (name_last, name_first, country)
VALUES('Kafka', 'Franz', 'Czech Republic');
```

Since `author_id` in the `authors` table is `AUTO_INCREMENT` (see AUTO_INCREMENT documentation), its value is assigned automatically. If not all columns are being supplied with data, the column names must be listed, followed by their corresponding values in the `VALUES` clause.

To insert data for a book, referencing `author_id` `1` (assuming Kafka's `author_id` became `1`):

```
INSERT INTO books (title, author_id, isbn, year_pub)
VALUES('The Castle', '1', '0805211063', '1998');
```

Multiple rows can be inserted with a single `INSERT` statement:

```
INSERT INTO books (title, author_id, isbn, year_pub)
VALUES('The Trial', '1', '0805210407', '1995'),
      ('The Metamorphosis', '1', '0553213695', '1995'),
      ('America', '1', '0805210644', '1995');
```

Retrieving Data

Use the SELECT statement (see SELECT documentation) to query data from tables.

To retrieve all book titles:

```
SELECT title FROM books;
```

To limit the number of rows returned (e.g., to 5) using `LIMIT` (see LIMIT documentation):

```
SELECT title FROM books LIMIT 5;
```

To retrieve data from multiple tables, use a `JOIN` (see JOIN documentation). This example lists book titles and author last names by joining `books` and `authors` on their common `author_id` column:

```
SELECT title, name_last
FROM books
JOIN authors USING (author_id);
```

To filter results, use the `WHERE` clause. This example finds books by 'Kafka' and renames the `title` column[1] in the output to 'Kafka Books' using `AS` (an alias):

```
SELECT title AS 'Kafka Books'
FROM books
JOIN authors USING (author_id)
WHERE name_last = 'Kafka';
```

```
+-------------------+
| Kafka Books       |
+-------------------+
| The Castle        |
| The Trial         |
| The Metamorphosis |
| America           |
+-------------------+
```

Changing & Deleting Data

To modify existing data, use the `UPDATE` statement (see UPDATE documentation). Always use a `WHERE` clause to specify which rows to update.

```
UPDATE books
SET title = 'Amerika'
WHERE isbn = '0805210644';
```

This changes the `title` for the book with the specified `isbn`. Multiple columns can be updated by separating `column = value` assignments with commas within the `SET` clause.

To remove rows from a table, use the `DELETE` statement (see DELETE documentation). Use `WHERE` to specify which rows to delete.

```
DELETE FROM books
WHERE author_id = '2034'; -- Assuming '2034' is the author_id to be deleted
```

This deletes all books associated with `author_id` ‘2034’.

This page is licensed: CC BY-SA / Gnu FDL

Altering Tables Guide

This guide provides essential instructions for modifying existing table structures. Learn how to add, drop, and change columns, manage indexes and default values, and rename tables, along with key precautions for these operations when working with your database.

Before You Begin: Backup Your Tables

Before making any structural changes to a table, especially if it contains data, **always create a backup**. The `mariadb-dump` utility is a common and effective tool for this.

Example: Backing up a single table

Suppose you have a database db1 and a table clients. Its initial structure is:

```
DESCRIBE clients;
```

```
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| cust_id     | int(11)     |      | PRI | 0       |       |
| name        | varchar(25) | YES  |     | NULL    |       |
| address     | varchar(25) | YES  |     | NULL    |       |
| city        | varchar(25) | YES  |     | NULL    |       |
| state       | char(2)     | YES  |     | NULL    |       |
| zip         | varchar(10) | YES  |     | NULL    |       |
| client_type | varchar(4)  | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

To back up the `clients` table from the command-line:

```
mariadb-dump --user='your_username' --password='your_password' --add-locks db1 clients > clients.sql
```

- Replace `'your_username'` and `'your_password'` with your actual MariaDB credentials.
- `--add-locks`: Locks the table during the backup and unlocks it afterward.
- `db1 clients`: Specifies the database and then the table.
- `> clients.sql`: Redirects the output to a file named `clients.sql`.

Restoring from a backup

If you need to restore the table:

```
mariadb --user='your_username' --password='your_password' db1 < clients.sql
```

This command uses the `mariadb` client to execute the SQL in `clients.sql`, which will typically drop the existing table (if it exists) and recreate it from the backup. Ensure no critical data has been added to the live table since the backup if you intend to overwrite it.

For the examples that follow, we'll assume structural changes are being made, sometimes on an empty table for simplicity, but the backup step is always recommended for tables with data.

Adding Columns

Use the `ALTER TABLE` statement with the `ADD COLUMN` clause.

Add a column to the end of the table:

To add a status column with a fixed width of two characters:

```
ALTER TABLE clients
ADD COLUMN status CHAR(2);
```

Add a column after a specific existing column:

To add address2 (varchar 25) after the address column:

```
ALTER TABLE clients
ADD COLUMN address2 VARCHAR(25) AFTER address;
```

Add a column to the beginning of the table:

```
ALTER TABLE clients
ADD COLUMN new_first_column VARCHAR(50) FIRST;
```

(Assuming `new_first_column` is the one to be added at the beginning).

After additions, the table structure might look like (excluding `new_first_column` for consistency with original example flow):

```
DESCRIBE clients;
```

```
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| cust_id     | int(11)     |      | PRI | 0       |       |
| name        | varchar(25) | YES  |     | NULL    |       |
| address     | varchar(25) | YES  |     | NULL    |       |
| address2    | varchar(25) | YES  |     | NULL    |       |
| city        | varchar(25) | YES  |     | NULL    |       |
| state       | char(2)     | YES  |     | NULL    |       |
| zip         | varchar(10) | YES  |     | NULL    |       |
| client_type | varchar(4)  | YES  |     | NULL    |       |
| status      | char(2)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

Changing Column Definitions

Use `ALTER TABLE` with `CHANGE` or `MODIFY`.

Change column type (e.g., to ENUM):

The status column name is specified twice even if not changing the name itself when using CHANGE.

```
ALTER TABLE clients
CHANGE status status ENUM('AC','IA');
```

Change column name and keep type:

To change status to active while keeping the ENUM definition:

```
ALTER TABLE clients
CHANGE status active ENUM('AC','IA');
```

When using `CHANGE`, the current column name is followed by the new column name and the complete type definition.

Modify column type or attributes without renaming:

Use MODIFY if you are only changing the data type or attributes, not the name.

```
ALTER TABLE clients
MODIFY address1 VARCHAR(40); -- Assuming 'address1' is an existing column
```

Complex Changes (e.g., ENUM migration with data):

Changing ENUM values in a table with existing data requires careful steps to prevent data loss. This typically involves:

1. Temporarily modifying the ENUM to include both old and new values.
2. Updating existing rows to use the new values.
3. Modifying the ENUM again to remove the old values.

Example of changing `address` to `address1` (40 chars) and preparing `active` ENUM for new values ‘yes’,‘no’ from ‘AC’,‘IA’:

```
ALTER TABLE clients
    CHANGE address address1 VARCHAR(40),
    MODIFY active ENUM('yes','no','AC','IA'); -- Temporarily include all
```

Then, update the data:

```
UPDATE clients
SET active = 'yes'
WHERE active = 'AC';

UPDATE clients
SET active = 'no'
WHERE active = 'IA';
```

Finally, restrict the ENUM to new values:

```
ALTER TABLE clients
MODIFY active ENUM('yes','no');
```

Dropping Columns

To remove a column and its data (this action is permanent and irreversible without a backup):

```
ALTER TABLE clients
DROP COLUMN client_type;
```

Managing Default Values

Set a default value for a column:

If most clients are in ‘LA’, set it as the default for the state column:

```
ALTER TABLE clients
ALTER state SET DEFAULT 'LA';
```

Remove a default value from a column:

This reverts the default to its standard (e.g., NULL if nullable, or determined by data type).

```
ALTER TABLE clients
ALTER state DROP DEFAULT;
```

This `DROP DEFAULT` does not delete existing data in the column.

Managing Indexes

Indexes are separate objects from columns. Modifying an indexed column often requires managing its index.

View existing indexes on a table with `SHOW INDEX`:

The \G displays results in a vertical format, which can be easier to read for wide output.

```
SHOW INDEX FROM clients\G
```

Example output:

```
*************************** 1. row ***************************
           Table: clients
      Non_unique: 0
        Key_name: PRIMARY
    Seq_in_index: 1
     Column_name: cust_id
       Collation: A
     Cardinality: 0
        Sub_part: NULL
          Packed: NULL
         Comment:
```

Changing an indexed column (e.g., Primary Key):

Attempting to CHANGE a column that is part of a PRIMARY KEY without addressing the key might result in an error like “Multiple primary key defined”. The index must be dropped first, then the column changed, and the key re-added.

```
ALTER TABLE clients
    DROP PRIMARY KEY,
    CHANGE cust_id client_id INT PRIMARY KEY;
```

The order is important: `DROP PRIMARY KEY` first.

Changing a column with another index type (e.g., UNIQUE):

If cust_id had a UNIQUE index named cust_id_unique_idx (Key_name from SHOW INDEX):

```
ALTER TABLE clients
    DROP INDEX cust_id_unique_idx, -- Use the actual Key_name
    CHANGE cust_id client_id INT UNIQUE;
```

If the `Key_name` is the same as the `Column_name` (e.g. for a single column `UNIQUE` key defined on `cust_id` where `cust_id` is also its `Key_name`):

```
ALTER TABLE clients
    DROP INDEX cust_id, -- If cust_id is the Key_name
    CHANGE cust_id client_id INT UNIQUE;
```

Changing index type and handling duplicates (e.g., INDEX to UNIQUE):

If changing from an index type that allows duplicates (like a plain INDEX) to one that doesn’t (UNIQUE), and duplicate data exists, the operation will fail. To force the change and remove duplicates (use with extreme caution):

```
ALTER IGNORE TABLE clients
    DROP INDEX cust_id_idx, -- Assuming cust_id_idx is the name of the old INDEX
    CHANGE cust_id client_id INT UNIQUE;
```

The `IGNORE` keyword causes rows with duplicate key values (for the new `UNIQUE` key) to be deleted. Only the first encountered row is kept.

Renaming and Shifting Tables

Rename a table:

To change the name of clients to client_addresses:

```
RENAME TABLE clients TO client_addresses;
```

Move a table to another database (can be combined with renaming):

To move client_addresses to a database named db2:

```
RENAME TABLE client_addresses TO db2.client_addresses;
```

Re-sort data within a table (MyRocks/Aria, not typically InnoDB):

For some storage engines (excluding InnoDB where tables are ordered by the primary key), you can physically reorder rows. This does not usually apply to InnoDB unless the ORDER BY columns form the primary key.

```
ALTER TABLE client_addresses
ORDER BY city, name;
```

After this, `SELECT * FROM client_addresses` (without an `ORDER BY` clause) might return rows in this new physical order, until further data modifications occur.

Key Considerations

- **Backup First:** Always back up tables before making structural alterations, especially on production systems.
- **Data Integrity:** Be mindful of how changes (e.g., type changes, ENUM modifications, dropping columns) can affect existing data. Test changes in a development environment.
- **Irreversible Actions:** Operations like `DROP COLUMN` or `DROP TABLE` are generally irreversible without restoring from a backup. There's typically no confirmation prompt.
- **Indexes:** Understand that indexes are distinct from columns. Modifying indexed columns often requires separate steps to manage the associated indexes.
- **Performance:** `ALTER TABLE` operations on large tables can be time-consuming and resource-intensive, potentially locking the table and impacting application performance. Plan these operations during maintenance windows if possible.

This page is licensed: CC BY-SA / Gnu FDL

Connecting to MariaDB Guide

This guide details the parameters for connecting to a MariaDB server using client programs like `mariadb`. Learn about default connection behaviors and how to use various command-line options to customize your connection, including secure TLS configurations.

While the examples focus on the `mariadb` command-line client, the concepts apply to other clients like graphical interfaces or backup utilities (e.g., `mariadb-dump`). If you are completely new to MariaDB, refer to A MariaDB Primer first.

Default Connection Behavior

When a connection parameter is not explicitly provided, a default value is used. To connect using only default values with the `mariadb` client:

```
mariadb
```

In this scenario, the following defaults typically apply:

- **Host name:** `localhost`
- **User name:** Your Unix login name (on Unix-like systems) or `ODBC` (on Windows).
- **Password:** No password is sent.
- **Database:** The client connects to the server but not to a specific database by default.
- **Socket:** The default socket file is used for connection.

Overriding Defaults

You can override these defaults by specifying parameters. For example:

```
mariadb -h 166.78.144.191 -u username -ppassword database_name
```

In this example:

- `-h 166.78.144.191`: Specifies the host IP address instead of `localhost`.
- `-u username`: Specifies `username` as the MariaDB user.
- `-ppassword`: Specifies `password` as the password.

    - **Note:** For passwords, there must be no space between `-p` and the password value.
    - **Security Warning:** Providing a password directly on the command line is insecure as it can be visible to other users on the system. It's more secure to use `-p` without the password value, which will prompt you to enter it.

- `database_name`: This is the name of the database to connect to, provided as the first argument after all options.
- The connection will use the default TCP/IP port (usually 3306).

Connection Parameters

The following are common connection parameters:

host

- `--host=name`
- `-h name`

Connects to the MariaDB server on the given host.

Default: localhost.

MariaDB typically does not permit remote logins by default; see Configuring MariaDB for Remote Client Access.

password

- `--password[=passwd]`
- `-p[passwd]`

Specifies the password for the MariaDB account.

- **Security Best Practice:** For improved security, use the `-p` or `--password` option without providing the password value directly on the command line. You will be prompted to enter it, preventing it from being visible in command history or process lists.

pipe

- `--pipe`
- `-W`

(Windows only) Connects to the server using a named pipe, if the server was started with the `--enable-named-pipe` option.

port

- `--port=num`
- `-P num`

Specifies the TCP/IP port number for the connection.

Default: 3306.

protocol

- `--protocol=name`

Specifies the connection protocol. Possible values (case-insensitive): `TCP`, `SOCKET`, `PIPE`, `MEMORY`. The default protocol is typically the most efficient for the operating system (e.g., `SOCKET` on Unix).

- **TCP:** TCP/IP connection (local or remote). Available on all OS.

- **SOCKET:** Unix socket file connection (local server on Unix systems only). If `--socket` is not specified, the default is `/tmp/mysql.sock`.
- **PIPE:** Named-pipe connection (local or remote). Windows only.
- **MEMORY:** Shared-memory connection (local server on Windows systems only).

shared-memory-base-name

- `--shared-memory-base-name=name`

(Windows only) Specifies the shared-memory name for connecting to a local server started with the –shared-memory option. The value is case-sensitive.

Default: MARIADB.

socket

- `--socket=name`
- `-S name`

For connections to `localhost`:

- On Unix: Specifies the Unix socket file to use. *Default:* `/tmp/mysql.sock`.
- On Windows: Specifies the name (case-insensitive) of the named pipe if the server was started with `--enable-named-pipe`. *Default:* `MARIADB`.

user

- `--user=name`
- `-u name`

Specifies the MariaDB user name for the connection.

Default: Your Unix login name (on Unix-like systems) or ODBC (on Windows).

See the GRANT command for information on creating MariaDB user accounts.

TLS Options

These options control the use of TLS (Transport Layer Security) for secure connections. For comprehensive details, see Secure Connections Overview and TLS System Variables.

- **`--ssl`**: Enable TLS for the connection. Automatically enabled if other `--ssl-*` flags are used. Disable with `--skip-ssl`.
- **`--ssl-ca=name`**: CA (Certificate Authority) file in PEM format. (Implies `--ssl`).
- **`--ssl-capath=name`**: Directory containing CA certificates in PEM format. (Implies `--ssl`).
- **`--ssl-cert=name`**: Client X.509 certificate in PEM format. (Implies `--ssl`).
- **`--ssl-cipher=name`**: Specific TLS cipher(s) to use for the connection. (Implies `--ssl`).
- **`--ssl-key=name`**: Client X.509 private key in PEM format. (Implies `--ssl`).
- **`--ssl-crl=name`**: Certificate Revocation List (CRL) file in PEM format. (Implies `--ssl`).
- **`--ssl-crlpath=name`**: Directory containing CRL files. (Implies `--ssl`).
- **`--ssl-verify-server-cert`**: Verifies the server's certificate "Common Name" against the hostname used for connecting. Disabled by default.

Option Files

Connection parameters and other options can also be set in option files (configuration files), which most MariaDB clients read upon startup. To see which option files a client reads and the option groups it recognizes, typically run the client with the `--help` option.

See Also

- A MariaDB Primer
- mariadb client
- Clients and Utilities
- Configuring MariaDB for Remote Client Access
- `--skip-grant-tables` allows you to start MariaDB without `GRANT`. This is useful if you lost your root password.

This page is licensed: CC BY-SA / Gnu FDL

Troubleshooting Connection Issues Guide

The guide helps diagnose and resolve common issues encountered when connecting to a MariaDB server. Identify causes for errors like 'Can't connect to local server' or access denied messages, and learn steps to effectively troubleshoot these connection problems.

If you are completely new to MariaDB and relational databases, you may want to start with A MariaDB Primer. Also, ensure you understand the connection parameters discussed in the Connection Parameters Guide.

Server Not Running or Incorrect Location

Symptoms

You receive errors like client error[1] 2002 or client error 2003:

```
ERROR 2002 (HY000): Can't connect to local MySQL server through
  socket '/var/run/mysqld/mysqld.sock' (2 "No such file or directory")
```

```
mariadb -u someuser -p --port=3307 --protocol=tcp
ERROR 2003 (HY000): Can't connect to MySQL server on 'localhost'
  (111 "Connection refused")
```

Causes & Solutions

Server Not Running

The MariaDB server process may not be running. Verify with `mariadb-admin`:

```
$ mariadb-admin status
mariadb-admin: connect to server at 'localhost' failed
error: 'Can't connect to local server through socket '/tmp/mysql.sock' (2)'
Check that mariadbd is running and that the socket: '/tmp/mysql.sock' exists!
```

If the server isn't running, start it using the method applicable to your environment.

Incorrect Parameters

The server is running, but not on the specified host, port, socket, pipe, or protocol. Verify your connection parameters.

- **Socket File Mismatch (Unix):** The socket file path might be non-standard or inconsistent between server and client configurations.
  - Check your configuration file. Ensure the `socket` option has the identical value for both server and client.
  - To find the running Unix socket file, try this command:

    ```
    $ netstat -ln | grep mysqld
    ```

    Example output:

    ```
    unix  2      [ ACC ]     STREAM     LISTENING     33209505 /var/run/mysqld/mysqld.sock
    ```
- See also: Troubleshooting Installation Issues.

Unable to Connect from a Remote Location

Symptoms

You can connect locally, but not from a remote machine, possibly seeing errors like this:

```
$ mariadb --host=myhost --protocol=tcp --port=3306 test
ERROR 2002 (HY000): Can't connect to MySQL server on 'myhost' (115)
```

You can use `telnet` (if available) to test basic network connectivity to the port:

```
$ telnet myhost 3306
```

A "Connection refused" message from `telnet` indicates a network or firewall issue, or that MariaDB is not listening for TCP/IP connections or on that specific interface/port.

The `perror` utility can interpret OS[2] error codes:

```
$ perror 115
```

Example output:

```
OS error code 115: Operation now in progress
```

Causes & Solutions

By default, MariaDB often does not accept remote TCP/IP connections, or is bound only to `localhost` (`127.0.0.1`).

Solution: See Configuring MariaDB for Remote Client Access for detailed instructions on how to enable remote connections by adjusting the `bind-address` server variable and ensuring user accounts are configured correctly for remote hosts.

Authentication Problems

Symptoms

Connection is established, but authentication fails (for instance, "Access denied for user…").

Causes & Solutions

- **Unix Socket Authentication:** On Unix-like systems, the `unix_socket` authentication plugin is enabled by default for local connections via the Unix socket file. This plugin uses operating system user credentials.

    - See the `unix_socket` authentication plugin documentation for connection instructions and how to switch to password-based authentication if needed.
    - For an overview of authentication changes in MariaDB 10.4, see Authentication from MariaDB 10.4.
- **Incorrect Username/Host Combination:** Authentication is specific to a `username@host` combination. For example, `'user1'@'localhost'` is distinct from `'user1'@'166.78.144.191'`. Ensure the user account exists for the host from which you are connecting.
    - See GRANT for details on granting permissions.
- **Password Hashing:** When setting or changing passwords using `SET PASSWORD`, ensure the `PASSWORD()` function is used if the server expects hashed passwords.
    - Example: `SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('newpass');`
    - Rather than: `SET PASSWORD FOR 'bob'@'%.loc.gov' = 'newpass';` (which might store the password as plain text, potentially leading to issues depending on the authentication plugin).

Problems Exporting Query Results or Loading Data

Symptoms

You can run regular queries, but get authentication or permission errors when using `SELECT ... INTO OUTFILE`, `SELECT ... INTO DUMPFILE`, or `LOAD DATA INFILE`.

Causes & Solutions

- These operations require the `FILE` privilege on the server.
- **Solution:** Grant the necessary `FILE` privilege to the user. See the GRANT article.

Access Denied to a Specific Database

Symptoms

You can connect to the MariaDB server, but attempting to issue the `USE` command or query a specific database results in an error:

```
USE test;
ERROR 1044 (42000): Access denied for user 'youruser'@'yourhost' to database 'test'
```

Or, connecting with `mariadb -u user -p db1` works, but `mariadb -u user -p db2` fails for `db2`.

Causes & Solutions

- The user account has not been granted sufficient privileges for that particular database.
- **Solution:** Grant the required privileges (e.g., `SELECT`, `INSERT`, etc.) on the specific database to the user. See GRANT.

Issues Due to Option Files or Environment Variables

Symptoms

Unexpected connection behavior, or parameter usage that you didn't explicitly provide on the command line.

Causes & Solutions

- Option files (for instance, `my.cnf`, `.my.ini`) or environment variables (for instance, `MYSQL_HOST`) might be supplying incorrect parameters, or overriding connection parameters.
- **Troubleshooting:**

- Check the values in any option files read by your client. See Configuring MariaDB with Option Files and the documentation for the specific client you are using (listed under Clients and Utilities).
- You can often suppress the reading of default option files by using a `--no-defaults` option (if supported by the client):

  ```
  $ mariadb --no-defaults ...
  ```

Unable to Connect / Lost Root Password

Symptoms

You cannot connect to a running server because the root (or other administrative) password is lost or unknown.

Causes & Solutions

- **Solution:** You can start the MariaDB server with the `--skip-grant-tables` option. This bypasses the privilege system, granting full access. **Use this with extreme caution and only temporarily.**
  1. Stop the MariaDB server.
  2. Restart the server manually from the command line, adding the `--skip-grant-tables` option.
  3. Connect to the server (no password is required for `root@localhost`).
  4. Change the password for the account you're connecting with (for example, `root`):

     ```
     SET PASSWORD FOR 'root'@'localhost' = PASSWORD('your_new_strong_password');
     ```

  5. Stop the server, and restart it normally (without `--skip-grant-tables`).

> **Note:**
>
> Before doing this, particularly if you cannot connect to a freshly installed MariaDB server, see if the next solution can solve your problem.

Quick Fix: Access Denied for 'root'@'localhost'?

Starting with MariaDB 10.4, the default security model for Linux installations uses the `unix_socket` authentication plugin. This means the MariaDB `root` user is tied to your system's `root` user.

- **The Problem:** If you try to connect using `mariadb -u root -p`, the server may reject you because it is looking for your operating system identity, not a password.
- **The Solution:** Instead of a password, use `sudo`:

  ```
  sudo mariadb
  ```

- **Why this works:** The server recognizes you have `sudo` (administrative) privileges on the machine and automatically logs you into the MariaDB `root` account without requiring a separate database password.

> **Warning:**
>
> Do not use this as a permanent solution.
>
> Rather than that, use it as a one-off, to be able to connect to the MariaDB Server at all. Once logged in, create a proper user, like `'myuser'@'localhost'`, or even `'myadmin'@'localhost'`. Then, grant the necessary privileges to that user. An administrative user, comparable to `root`, should have privileges to access every object in your database, by running this query:
>
> `GRANT ALL ON *.* to 'myadmin'@'localhost' IDENTIFIED BY '(your_password)' WITH GRANT OPTION`
>
> When done, log out, then log in again, using your newly created user. This is now possible without using the `sudo` workaround:
>
> `mariadb --user myadmin --password` (specify *your_password* when prompted)

`localhost` vs. `%` Wildcard Host Issues

Symptoms

You've created a user like `'melisa'@'%'` but cannot log in as melisa when connecting from `localhost`.

```
-- User created with '%' host
CREATE USER 'melisa'@'%' IDENTIFIED BY 'password';

-- Checking users in mysql.user table
SELECT user, host FROM mysql.user WHERE user='melisa' OR user='';
```

Example output showing the problem:

```
+--------+-----------+
| user   | host      |
+--------+-----------+
| melisa | %         |
|        | localhost | -- An anonymous user for localhost
+--------+-----------+
```

Causes & Solutions

- The MariaDB user authentication prioritizes more specific host matches. If an anonymous user (`''@'localhost'`) exists, it can take precedence over `'melisa'@'%'` when connecting from `localhost`.
- **Solutions:**
  1. Create a specific user for localhost:

     ```
     CREATE USER 'melisa'@'localhost' IDENTIFIED BY 'password_for_melisa_localhost';
     GRANT ALL PRIVILEGES ON yourdatabase.* TO 'melisa'@'localhost'; -- Grant necessary privileges
     ```

  2. Remove the anonymous user for localhost (use with caution):

     ```
     DROP USER ''@'localhost';
     ```

     Ensure this doesn't break other intended anonymous access, if any.

MariaDB Authentication Tutorial

https://www.youtube.com/watch?v=aWFG4uLbimM

MariaDB authentication from MariaDB 10.4

In this video tutorial, the MariaDB team explains the fundamental changes to the security model introduced in version 10.4, specifically regarding how the `root` user and local connections are handled.

Core Topics Covered:

- The “No Password” Default: Explains why, in MariaDB 10.4 and later, the `root` user does not have a password by default on many Linux distributions.
- Unix Socket Authentication: A walkthrough of the `unix_socket` plugin. This plugin allows the OS-level `root` user to log in to the MariaDB `root` account without a password, as security is verified by the operating system identity.
- The `mysql.global_priv` table: Introduction of the new table that replaces the old `mysql.user` table for storing privileges, and how this change simplifies managing multiple authentication methods for a single user.
- Switching Authentication Methods: Practical steps on how to move from socket-based authentication back to traditional password-based authentication (using the `mysql_native_password` plugin) if your environment requires it.

Key Takeaway for Troubleshooting:

If you are receiving an Access Denied error while trying to log in as `root` despite using a password you believe is correct, the video demonstrates that your server is likely expecting Unix Socket authentication. In this case, you should use `sudo mariadb` rather than `mariadb -u root -p`.

See Also

- CREATE USER
- GRANT
- Authentication
- Authentication from MariaDB 10 4 (video • 20 minutes • 2020)
- Error 1698: Access denied for user

This page is licensed: CC BY-SA / Gnu FDL

Doing Time Guide

This guide covers effective ways to work with date and time information in MariaDB. Learn about temporal data types, essential functions for recording current date/time, extracting specific parts, and formatting your date/time values for display or analysis.

Temporal Data Types

While dates and times can be stored as character strings, using specific temporal data types allows you to leverage MariaDB’s built-in functions for manipulation and formatting.

- **`DATE`**: For dates only. Format: `YYYY-MM-DD`.
- **`TIME`**: For time only. Format: `HHH:MM:SS` (hours can range beyond 24).
- **`DATETIME`**: For combined date and time. Format: `YYYY-MM-DD HH:MM:SS`.
- **`TIMESTAMP`**: Similar to `DATETIME`, but with a more limited range and automatic update capabilities (not covered here). Range typically from `1970-01-01 00:00:01` UTC to `2038-01-19 03:14:07` UTC. From MariaDB 11.5 (64-bit), this range extends to `2106-02-07`.
- **`YEAR`**: For years only. Format: `YY` or `YYYY`.

Recording Current Date and Time

MariaDB provides several functions to get the current date and time.

Current Date:

Use `CURRENT_DATE` (no parentheses) or `CURDATE()` (with parentheses).

```
INSERT INTO billable_work (doctor_id, patient_id, session_date)
VALUES ('1021', '1256', CURRENT_DATE);
```

To see the ID of the last inserted row (if the primary key is `AUTO_INCREMENT`):

```
SELECT rec_id, doctor_id, patient_id, session_date
FROM billable_work
WHERE rec_id = LAST_INSERT_ID();
```

```
+--------+-----------+------------+--------------+
| rec_id | doctor_id | patient_id | session_date |
+--------+-----------+------------+--------------+
|   2462 | 1021      | 1256       | 2025-05-28   | -- Example date
+--------+-----------+------------+--------------+
```

Current Time:

Use `CURRENT_TIME` or `CURTIME()`.

```
UPDATE billable_work
SET session_time = CURTIME()
WHERE rec_id = '2462';

SELECT patient_id, session_date, session_time
FROM billable_work
WHERE rec_id = '2462';
```

```
+------------+--------------+--------------+
| patient_id | session_date | session_time |
+------------+--------------+--------------+
| 1256       | 2025-05-28   | 13:03:22     | -- Example time
+------------+--------------+--------------+
```

Current Date and Time (Timestamp):

Use `CURRENT_TIMESTAMP`, `NOW()`, or `SYSDATE()`. These functions return the current date and time in `YYYY-MM-DD HH:MM:SS` format, suitable for `DATETIME` or `TIMESTAMP` columns.

Extracting Date and Time Parts

Extracting from `DATE` types:

- `YEAR(date_column)`: Extracts the year.
- `MONTH(date_column)`: Extracts the month number (1-12).
- `DAYOFMONTH(date_column)`: Extracts the day of the month (1-31). Also `DAY()`.

```
SELECT
    MONTH(session_date) AS Month,
    DAYOFMONTH(session_date) AS Day,
    YEAR(session_date) AS Year
FROM billable_work
WHERE rec_id = '2462';
```

```
+-------+------+------+
| Month | Day  | Year |
+-------+------+------+
|     5 |   28 | 2025 | -- Example output
+-------+------+------+
```

(The `AS` keyword is used to provide an alias for the output column name.)

Day of the Week:

- `DAYOFWEEK(date_column)`: Returns the weekday index (1=Sunday, 2=Monday, …, 7=Saturday).
- `WEEKDAY(date_column)`: Returns the weekday index (0=Monday, 1=Tuesday, …, 6=Sunday).

Example using `IF()` to determine a billing rate based on the day of the week (Saturday = day 7 for `DAYOFWEEK`):

```
SELECT
    patient_id AS 'Patient ID',
    session_date AS 'Date of Session',
    IF(DAYOFWEEK(session_date) = 7, 1.5, 1.0) AS 'Billing Rate'
FROM billable_work
WHERE rec_id = '2462';
```

The `IF(condition, value_if_true, value_if_false)` function allows conditional logic.

Other Date Part Functions:

- `DAYOFYEAR(date_column)`: Returns the day of the year (1-366).
- `QUARTER(date_column)`: Returns the quarter of the year (1-4).

Example: Selecting sessions in a specific quarter (e.g., Q2):

```
SELECT patient_id, session_date
FROM billable_work
WHERE QUARTER(session_date) = 2;
```

User variables can be used for dynamic queries:

```
SET @target_quarter := 2;
SELECT patient_id, COUNT(*) AS num_sessions
FROM billable_work
WHERE QUARTER(session_date) = @target_quarter AND doctor_id = '1021'
GROUP BY patient_id;
```

Extracting from `TIME` types:

- `HOUR(time_column)`: Extracts the hour.
- `MINUTE(time_column)`: Extracts the minute.
- `SECOND(time_column)`: Extracts the second.

```
SELECT
    HOUR(session_time) AS Hour,
    MINUTE(session_time) AS Minute,
    SECOND(session_time) AS Second
FROM billable_work
WHERE rec_id = '2462';
```

```
+------+--------+--------+
| Hour | Minute | Second |
+------+--------+--------+
|   13 |     03 |     22 | -- Example output
+------+--------+--------+
```

Using `EXTRACT()` for `DATETIME` or `TIMESTAMP` types:

The `EXTRACT(unit FROM datetime_column)` function extracts a specified `unit` from a date/time value.
Common units: `YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, `SECOND`.
Combined units: `YEAR_MONTH`, `DAY_HOUR`, `HOUR_MINUTE`, etc.

```
SELECT
    patient_name AS Patient,
    EXTRACT(HOUR FROM appointment) AS Hour,
    EXTRACT(MINUTE FROM appointment) AS Minute
FROM billable_work
JOIN patients ON billable_work.patient_id = patients.patient_id
WHERE doctor_id = '1021'
  AND EXTRACT(MONTH FROM appointment) = 5
  AND EXTRACT(DAY FROM appointment) = 28;
```

(For details on joining tables, refer to relevant SQL documentation or a guide like "Essential Queries Guide".)

Using a combined unit:

```
SELECT
    patient_name AS Patient,
    EXTRACT(HOUR_MINUTE FROM appointment) AS AppointmentHM
FROM billable_work
JOIN patients ON billable_work.patient_id = patients.patient_id
WHERE doctor_id = '1021';
```

Output for `HOUR_MINUTE` might be like `1303` (for 13:03).

Formatting Dates and Times for Display

Wordier Date Formats:

- `MONTHNAME(date_column)`: Returns the full name of the month (e.g., 'May').
- `DAYNAME(date_column)`: Returns the full name of the day (e.g., 'Wednesday').

Example using `CONCAT()` to combine parts:

```
SELECT
    patient_name AS Patient,
    CONCAT(
        DAYNAME(appointment), ' - ',
        MONTHNAME(appointment), ' ',
        DAYOFMONTH(appointment), ', ',
        YEAR(appointment)
    ) AS Appointment
FROM billable_work
JOIN patients ON billable_work.patient_id = patients.patient_id
WHERE doctor_id = '1021' AND DATE(appointment) = '2025-05-28'
LIMIT 1;
```

```
+------------------+------------------------------+
| Patient          | Appointment                  |
+------------------+------------------------------+
| Michael Zabalaoui | Wednesday - May 28, 2025     | -- Example
+------------------+------------------------------+
```

Using `DATE_FORMAT(datetime_column, format_string)`:
This function provides extensive formatting options.
Syntax: `DATE_FORMAT(date_value, 'format_options_and_literals')`.

```
SELECT
    patient_name AS Patient,
    DATE_FORMAT(appointment, '%W - %M %e, %Y') AS Appointment
FROM billable_work
JOIN patients ON billable_work.patient_id = patients.patient_id
WHERE doctor_id = '1021' AND DATE_FORMAT(appointment, '%c') = 5 -- Filter by month 5 (May)
LIMIT 1;
```

Common format specifiers:

- `%W`: Full weekday name
- `%M`: Full month name
- `%e`: Day of the month, numeric (1-31)
- `%d`: Day of the month, 2 digits (01-31)
- `%Y`: Year, 4 digits
- `%y`: Year, 2 digits
- `%c`: Month, numeric (1-12)
- `%r`: Time in 12-hour format (hh:mm:ss AM/PM)
- `%T`: Time in 24-hour format (hh:mm:ss)
- `%H`: Hour (00-23)
- `%h` or `%I`: Hour (01-12)
- `%i`: Minutes (00-59)
- `%s` or `%S`: Seconds (00-59)
- `%p`: AM or PM

Example with time:

```
SELECT
    DATE_FORMAT(appointment, '%W - %M %e, %Y at %r') AS Appointment
FROM billable_work
LIMIT 1;
```

```
+-----------------------------------------------+
| Appointment                                   |
+-----------------------------------------------+
| Wednesday - May 28, 2025 at 01:03:22 PM       | -- Example
+-----------------------------------------------+
```

For a complete list of options, see the official DATE_FORMAT() documentation.

Using `TIME_FORMAT(time_column, format_string)`:
Similar to `DATE_FORMAT()`, but uses only time-related format options.

```
SELECT
    patient_name AS Patient,
    TIME_FORMAT(appointment, '%l:%i %p') AS AppointmentTime
FROM billable_work
JOIN patients ON billable_work.patient_id = patients.patient_id
WHERE doctor_id = '1021'
  AND DATE(appointment) = CURDATE();
```

```
+------------------+-----------------+
| Patient          | AppointmentTime |
+------------------+-----------------+
| Michael Zabalaoui |     1:03 PM     | -- Example
+------------------+-----------------+
```

Here, `%l` is hour (1-12) and `%p` adds AM/PM.

Tips for Effective Date/Time Handling

- **Use Appropriate Data Types:** Choose temporal data types (`DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, `YEAR`) over string types for date/time data to leverage built-in functions and ensure data integrity.

- **Leverage Built-in Functions:** MariaDB offers a rich set of functions for date/time manipulation. Use them within your SQL queries to avoid complex logic in your application code.
- **Test Queries:** When dealing with complex date/time logic or formatting, test your SQL statements directly in a MariaDB client (like the `mariadb` command-line tool) to verify results before embedding them in applications.
- **Be Aware of Time Zones:** `TIMESTAMP` values are stored in UTC and converted to/from the session's time zone, while `DATETIME` values are stored "as is" without time zone conversion. Understand how your server and session time zones are configured if working with data across different regions. (Time zone handling is a more advanced topic not fully covered here).

See Also

- Date & Time Functions — the full reference for every function used above
- DATE_FORMAT() and TIME_FORMAT()
- EXTRACT()

This page is licensed: CC BY-SA / Gnu FDL

Importing Data Guide

This guide introduces methods and tools for efficiently importing bulk data into MariaDB. Learn to prepare your data, use `LOAD DATA INFILE` and the `mariadb-import` utility, handle common data import challenges, and manage potential constraints.

Preparing Your Data File

The most common approach for bulk importing is to use a delimited text file.

1. **Export Source Data:** Load your data in its original software (e.g., MS Excel, MS Access) and export it as a delimited text file.
   - **Delimiter:** Use a character not commonly found in your data to separate fields. The pipe symbol (`|`) is often a good choice. Tab (`\t`) is also common.
   - **Record Separator:** Use line feeds (`\n`) to separate records.
2. **Align Columns (Recommended for Simplicity):** Ideally, the order and number of columns in your text file should match the target MariaDB table.
   - If the table has extra columns not in your file, they will be filled with their default values (or `NULL`).
   - If your file has extra columns not in the table, you'll need to specify which file columns to load (see "Mapping File Columns to Table Columns" below) or remove them from the text file.
3. **Clean Data:** Remove any header rows or footer information from the text file unless you plan to skip them during import (see `IGNORE N LINES` below).
4. **Upload File:** Transfer the text file to a location accessible by the MariaDB server.
   - Use ASCII mode for FTP transfers to ensure correct line endings.
   - For security, upload data files to non-public directories on the server.

Using `LOAD DATA INFILE`

The `LOAD DATA INFILE` statement is a powerful SQL command for importing data from text files. Ensure the MariaDB user has the `FILE` privilege.

Basic Syntax:

First, connect to MariaDB using the mariadb client and select your target database:

```
USE sales_dept; -- Or your database name
```

Then, load the data:

```
LOAD DATA INFILE '/tmp/prospects.txt'
INTO TABLE prospect_contact
FIELDS TERMINATED BY '|';
```

- Replace `/tmp/prospects.txt` with the actual path to your data file on the server. On Windows, paths use forward slashes (e.g., `'C:/tmp/prospects.txt'`).
- `prospect_contact` is the target table. You can also specify `database_name.table_name`.
- `FIELDS TERMINATED BY '|'` specifies the field delimiter. For tab-delimited, use `'\t'`.
- The default record delimiter is the line feed (`\n`).

Specifying Line Terminators and Enclosing Characters:

If your file has custom line endings or fields enclosed by characters (e.g., quotes):

```
LOAD DATA INFILE '/tmp/prospects.txt'
INTO TABLE prospect_contact
FIELDS TERMINATED BY '|' ENCLOSED BY '"'
LINES STARTING BY '"' TERMINATED BY '"\r\n';
```

- `ENCLOSED BY '"'`: Specifies that fields are enclosed in double quotes.
- `LINES STARTING BY '"'`: Indicates each line starts with a double quote.
- `TERMINATED BY '"\r\n'`: Indicates each line ends with a double quote followed by a Windows-style carriage return and line feed.
- To specify a single quote as an enclosing character, you can escape it or put it within double quotes: `ENCLOSED BY '\''` or `ENCLOSED BY "'"`.

Handling Duplicate Rows

When importing data, you might encounter records with primary key values that already exist in the target table.

- **Default Behavior:** MariaDB attempts to import all rows. If duplicates are found and the table has a primary or unique key that would be violated, an error occurs, and subsequent rows may not be imported.
- **`REPLACE`:** If you want new data from the file to overwrite existing rows with the same primary key:SQL

  ```
  LOAD DATA INFILE '/tmp/prospects.txt'
  REPLACE INTO TABLE prospect_contact
  FIELDS TERMINATED BY '|';
  ```

- **`IGNORE`:** If you want to keep existing rows and skip importing duplicate records from the file:SQL

  ```
  LOAD DATA INFILE '/tmp/prospects.txt'
  IGNORE INTO TABLE prospect_contact
  FIELDS TERMINATED BY '|';
  ```

Importing into Live Tables

If the target table is actively being used, importing data can lock it, preventing access.

- **`LOW_PRIORITY`:** To allow other users to read from the table while the load operation is pending, use `LOW_PRIORITY`. The load will wait until no other clients are reading the table.SQL

  ```
  LOAD DATA LOW_PRIORITY INFILE '/tmp/prospects.txt'
  INTO TABLE prospect_contact
  FIELDS TERMINATED BY '|';
  ```

Without `LOW_PRIORITY` or `CONCURRENT`, the table is typically locked for the duration of the import.

Advanced `LOAD DATA INFILE` Options

Binary Line Endings:

If your file has Windows CRLF line endings and was uploaded in binary mode, you can specify the hexadecimal value:

```
LOAD DATA INFILE '/tmp/prospects.txt'
INTO TABLE prospect_contact
FIELDS TERMINATED BY '|'
LINES TERMINATED BY 0x0d0a; -- 0x0d is carriage return, 0x0a is line feed
```

Note: No quotes around the hexadecimal value.

Skipping Header Lines:

To ignore a certain number of lines at the beginning of the file (e.g., a header row):

SQL

```
LOAD DATA INFILE '/tmp/prospects.txt'
INTO TABLE prospect_contact
FIELDS TERMINATED BY '|'
IGNORE 1 LINES; -- Skips the first line
```

Handling Escaped Characters:

If fields are enclosed by quotes and contain embedded quotes that are escaped by a special character (e.g., # instead of the default backslash \):

```
LOAD DATA INFILE '/tmp/prospects.txt'
INTO TABLE prospect_contact
FIELDS TERMINATED BY '|'
    ENCLOSED BY '"'
    ESCAPED BY '#'
IGNORE 1 LINES;
```

Mapping File Columns to Table Columns:

If the order or number of columns in your text file differs from the target table, you can specify the column mapping at the end of the LOAD DATA INFILE statement.

Assume prospect_contact table has: (row_id INT AUTO_INCREMENT, name_first VARCHAR, name_last VARCHAR, telephone VARCHAR).

And prospects.txt has columns in order: Last Name, First Name, Telephone.

```
LOAD DATA INFILE '/tmp/prospects.txt'
INTO TABLE prospect_contact
FIELDS TERMINATED BY '|' -- Or your actual delimiter, e.g., 0x09 for tab
ENCLOSED BY '"'
ESCAPED BY '#'
IGNORE 1 LINES
(name_last, name_first, telephone);
```

- MariaDB will map data from the file's first column to `name_last`, second to `name_first`, and third to `telephone`.
- The `row_id` column in the table, not being specified in the list, will be filled by its default mechanism (e.g., `AUTO_INCREMENT` or `DEFAULT` value, or `NULL`).

Using the `mariadb-import` Utility

The `mariadb-import` utility (known as `mysqlimport` before MariaDB 10.5) is a command-line program that acts as a wrapper for `LOAD DATA INFILE`. It's useful for scripting imports.

Syntax:

```
mariadb-import --user='your_username' --password='your_password' \
    --fields-terminated-by='|' --lines-terminated-by='\r\n' \
    --replace --low-priority --fields-enclosed-by='"' \
    --fields-escaped-by='#' --ignore-lines='1' --verbose \
    --columns='name_last,name_first,telephone' \
    sales_dept '/tmp/prospect_contact.txt'
```

- This command is run from the system shell, not within the `mariadb` client.
- Lines are continued with `\` for readability here; it can be a single line.
- `--password`: If the password value is omitted, you'll be prompted.
- The database name (`sales_dept`) is specified before the file path.
- **File Naming:** `mariadb-import` expects the text file's name (without extension) to match the target table name (e.g., `prospect_contact.txt` for table `prospect_contact`). If your file is `prospects.txt` and table is `prospect_contact`, you might need to rename the file or import into a temporary table named `prospects` first.
- `--verbose`: Shows progress information.
- You can list multiple text files to import into correspondingly named tables.

Dealing with Web Hosting Restraints

Some web hosts disable `LOAD DATA INFILE` or `mariadb-import` for security reasons. A workaround involves using `mariadb-dump`:

1. **Prepare Data Locally:** Prepare your delimited text file (e.g., `prospects.txt`).
2. **Local Import:** If you have a local MariaDB server, import the text file into a local table (e.g., `local_db.prospect_contact`) using `LOAD DATA INFILE` as described above.
3. **Local Export with `mariadb-dump`:** Export the data from your local table into an SQL file containing `INSERT` statements.

   ```
   mariadb-dump --user='local_user' --password='local_pass' --no-create-info local_db prospect_contact > /tmp/prospects.sql
   ```

   - `--no-create-info` (or `-t`): Prevents the `CREATE TABLE` statement from being included, outputting only `INSERT` statements. This is useful if the table already exists on the remote server.
4. **Upload SQL File:** Upload the generated `.sql` file (e.g., `prospects.sql`) to your web server (in ASCII mode).
5. **Remote Import of SQL File:** Log into your remote server's shell and import the SQL file using the `mariadb` client:

   ```
   mariadb --user='remote_user' --password='remote_pass' remote_sales_dept < /tmp/prospects.sql
   ```

Handling Duplicates with mariadb-dump Output:

mariadb-dump does not have a REPLACE flag like LOAD DATA INFILE. If the target table might contain duplicates:

- Open the `.sql` file generated by `mariadb-dump` in a text editor.
- Perform a search and replace operation to change all occurrences of `INSERT INTO` to `REPLACE INTO`. The syntax for `INSERT` and `REPLACE` (for the data values part) is similar enough that this often works. Test thoroughly.

Key Considerations

- **Flexibility:** MariaDB provides powerful and flexible options for data importing. Understanding the details of `LOAD DATA INFILE` and `mariadb-import` can save significant effort.
- **Data Validation:** While these tools are efficient for bulk loading, they may not perform extensive data validation beyond basic type compatibility. Cleanse and validate your data as much as possible before importing.
- **Character Sets:** Ensure your data file's character set is compatible with the target table's character set to avoid data corruption. You can specify character sets in `LOAD DATA INFILE`.
- **Other Tools/Methods:** For very complex transformations or ETL (Extract, Transform, Load) processes, dedicated ETL tools or scripting languages (e.g., Python, Perl with database modules) might be more suitable, though they are beyond the scope of this guide.

This page is licensed: CC BY-SA / Gnu FDL

Essentials of an Index Guide

An index on `Last_Name` organizes the records by surname, enhancing search efficiency without altering the original table order. Indices can be created for any column, such as ID or first name, to enable quick lookups based on different criteria.

Imagine you've created a table with the following rows:

```
+----+------------+-----------+-------------------------+---------------------------+--------------+
| ID | First_Name | Last_Name | Position                | Home_Address              | Home_Phone   |
+----+------------+-----------+-------------------------+---------------------------+--------------+
|  1 | Mustapha   | Mond      | Chief Executive Officer | 692 Promiscuous Plaza     | 326-555-3492 |
|  2 | Henry      | Foster    | Store Manager           | 314 Savage Circle         | 326-555-3847 |
|  3 | Bernard    | Marx      | Cashier                 | 1240 Ambient Avenue       | 326-555-8456 |
|  4 | Lenina     | Crowne    | Cashier                 | 281 Bumblepuppy Boulevard | 328-555-2349 |
|  5 | Fanny      | Crowne    | Restocker               | 1023 Bokanovsky Lane      | 326-555-6329 |
|  6 | Helmholtz  | Watson    | Janitor                 | 944 Soma Court            | 329-555-2478 |
+----+------------+-----------+-------------------------+---------------------------+--------------+
```

Now, imagine you've been asked to return the home phone of Fanny Crowne. Without indexes, the only way to do it is to go through every row until you find the matching first name and surname. Now imagine there are millions of records and you can see that, even for a speedy database server, this is highly inefficient.

The answer is to sort the records. If they were stored in alphabetical order by surname, even a human could quickly find a record amongst a large number. But we can't sort the entire record by surname. What if we want to also look a record by ID, or by first name? The answer is to create separate indexes for each column we wish to sort by. An index simply contains the sorted data (such as surname), and a link to the original record.

For example, an index on Last_Name:

```
+-----------+----+
| Last_Name | ID |
+-----------+----+
| Crowne    |  4 |
| Crowne    |  5 |
| Foster    |  2 |
| Marx      |  3 |
| Mond      |  1 |
| Watson    |  6 |
+-----------+----+
```

and an index on Position

```
+-------------------------+----+
| Position                | ID |
+-------------------------+----+
| Cashier                 |  3 |
| Cashier                 |  4 |
| Chief Executive Officer |  1 |
| Janitor                 |  6 |
| Restocker               |  5 |
| Store Manager           |  2 |
+-------------------------+----+
```

would allow you to quickly find the phone numbers of all the cashiers, or the phone number of the employee with the surname Marx, very quickly.

Where possible, you should create an index for each column that you search for records by, to avoid having the server read every row of a table.

See CREATE INDEX and Getting Started with Indexes for more information.

This page is licensed: CC BY-SA / Gnu FDL

Getting Started with Indexes Guide

This guide explains the different types of indexes in MariaDB, their characteristics, and how they are used. Learn to create and manage Primary Keys, Unique Indexes, and Plain Indexes, along with key considerations for choosing and maintaining effective indexes for optimal query performance.

In MariaDB, the terms `KEY` and `INDEX` are generally used interchangeably in SQL statements. For a gentler conceptual overview, see The Essentials of an Index.

Index Types Overview

There are four main kinds of indexes:

- **Primary Keys:** Unique and not NULL.
- **Unique Indexes:** Must be unique but can contain NULL values.
- **Plain Indexes (or Regular Indexes):** Not necessarily unique.
- **Full-Text Indexes:** Used for full-text searching capabilities.

Primary Key

A primary key uniquely identifies each record in a table. Its values must be unique, and it cannot contain `NULL` values. Each table can have only one primary key.

InnoDB Considerations:

- In InnoDB tables, the primary key is included as a suffix in all other indexes. Therefore, keeping the primary key compact (e.g., using an appropriate integer type) is important for performance and storage efficiency.
- If a table has no explicitly defined primary key and no `UNIQUE` indexes, InnoDB automatically creates an invisible 6-byte clustered index.

Using `AUTO_INCREMENT`: The `AUTO_INCREMENT` attribute is commonly used with numeric primary keys to automatically generate a unique ID for each new row.

```
CREATE TABLE `Employees` (
  `ID` TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `First_Name` VARCHAR(25) NOT NULL,
  `Last_Name` VARCHAR(25) NOT NULL,
  `Position` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`ID`)
);
```

Note: The column defined as a primary key (or part of it) must be explicitly declared as `NOT NULL`.

Adding a Primary Key to an Existing Table: Use `ALTER TABLE`. You cannot create a primary key with `CREATE INDEX`.

```
ALTER TABLE Employees ADD PRIMARY KEY(ID);
```

Finding Tables Without Primary Keys: This query uses the `information_schema` database to find tables lacking primary keys:

```
SELECT t.TABLE_SCHEMA, t.TABLE_NAME
FROM information_schema.TABLES AS t
LEFT JOIN information_schema.KEY_COLUMN_USAGE AS c
ON t.TABLE_SCHEMA = c.CONSTRAINT_SCHEMA
   AND t.TABLE_NAME = c.TABLE_NAME
   AND c.CONSTRAINT_NAME = 'PRIMARY'
WHERE t.TABLE_SCHEMA NOT IN ('information_schema', 'performance_schema', 'mysql', 'sys')
  AND c.CONSTRAINT_NAME IS NULL;
```

Unique Index

A unique index ensures that all values in the indexed column (or combination of columns) are unique. However, unlike a primary key, columns in a unique index can store `NULL` values.

Each key value uniquely identifies a row, but not every row needs to be represented if `NULL`s are allowed.

Behavior (MariaDB 10.5+):

- If the index type is not specified, `UNIQUE` typically creates a BTREE index, usable by the optimizer.
- If a key exceeds the maximum length for the storage engine and the engine supports long unique indexes, a HASH key might be created to enforce uniqueness.

Creating Unique Indexes: During table creation:

```
CREATE TABLE `Employees` (
  `ID` TINYINT(3) UNSIGNED NOT NULL,
  `Employee_Code` VARCHAR(25) NOT NULL,
  `First_Name` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `UK_EmpCode` (`Employee_Code`) -- Naming the unique key is good practice
);
```

After table creation using `ALTER TABLE`:

```
ALTER TABLE Employees ADD UNIQUE `UK_HomePhone` (`Home_Phone`);
```

After table creation using `CREATE UNIQUE INDEX`:

```
CREATE UNIQUE INDEX `IX_Position` ON Employees(Position);
```

Multi-Column Unique Indexes: An index can span multiple columns. MariaDB can use the leftmost part(s) of such an index if it cannot use the whole index (except for HASH indexes).

```
CREATE TABLE t1 (a INT NOT NULL, b INT, UNIQUE (a,b));
INSERT INTO t1 VALUES (1,1), (2,2);
INSERT INTO t1 VALUES (2,1); -- Valid: (2,1) is unique, though '2' in 'a' and '1' in 'b' are not
individually unique here.
SELECT * FROM t1;
```

```
+---+------+
| a | b    |
+---+------+
| 1 |    1 |
| 2 |    1 |
| 2 |    2 |
+---+------+
```

`NULL` Values in Unique Indexes: A `UNIQUE` constraint allows multiple `NULL` values because in SQL, `NULL` is never equal to another `NULL`.

```
INSERT INTO t1 VALUES (3,NULL), (3, NULL); -- Both rows are inserted
SELECT * FROM t1;
```

```
+---+------+
| a | b    |
+---+------+
| 1 |    1 |
| 2 |    1 |
| 2 |    2 |
| 3 | NULL |
| 3 | NULL |
+---+------+
```

Verification:

```
SELECT (3, NULL) = (3, NULL);
```

```
+-----------------------+
| (3, NULL) = (3, NULL) |
+-----------------------+
|                     0 | -- 0 means false
+-----------------------+
```

Conditional Uniqueness with Virtual Columns: You can enforce uniqueness over a subset of rows using unique indexes on virtual columns. This example ensures `user_name` is unique for 'Active' or 'On-Hold' users, but allows duplicate names for 'Deleted' users:

```
CREATE TABLE Table_1 (
  user_name VARCHAR(10),
  status ENUM('Active', 'On-Hold', 'Deleted'),
  del CHAR(0) AS (IF(status IN ('Active', 'On-Hold'), '', NULL)) PERSISTENT,
  UNIQUE(user_name, del)
);
```

Trailing Pad Characters: If a unique index is on a column where trailing pad characters are stripped or ignored (e.g., `CHAR` vs `VARCHAR` behavior), inserts where values differ only by the number of trailing pad characters can result in duplicate-key errors.

Long Keys and HASH Indexes (MariaDB 10.4+): For engines like InnoDB, `UNIQUE` can be used with various column types and numbers. If a key's length exceeds the engine's maximum, a HASH key may be created.

```
-- Example table definition (simplified for brevity)
CREATE TABLE t_long_keys (
  a INT PRIMARY KEY,
  b BLOB,
  c1 VARCHAR(1000),
  UNIQUE KEY `uk_b` (b),
  UNIQUE KEY `uk_c1` (c1)
) ENGINE=InnoDB;

-- SHOW CREATE TABLE might reveal 'USING HASH' for uk_b or uk_c1 if they exceed length limits
SHOW CREATE TABLE t_long_keys\G
```

Example output snippet showing `USING HASH`:

```
...
  UNIQUE KEY `uk_b` (`b`) USING HASH,
...
```

Plain Indexes (Regular Indexes)

Plain indexes do not enforce uniqueness; they are primarily used to speed up data retrieval.

```
CREATE TABLE t2 (a INT NOT NULL, b INT, INDEX `idx_a_b` (a,b));
INSERT INTO t2 VALUES (1,1), (2,2), (2,2); -- Duplicate (2,2) is allowed
SELECT * FROM t2;
```

```
+---+------+
| a | b    |
+---+------+
| 1 |    1 |
| 2 |    2 |
| 2 |    2 |
+---+------+
```

Full-Text Indexes

Full-text indexes are used for performing full-text searches on text data. For details, see the Full-Text Indexes documentation.

Choosing Indexes

- **Index for Queries:** Add indexes that match the `WHERE` clauses, `JOIN` conditions, and `ORDER BY` clauses of your application's queries.
- **Avoid Over-Indexing:** Extra indexes consume storage and can slow down `INSERT`, `UPDATE`, and `DELETE` operations.
- **Impact of Table Size:** Indexes provide more significant speed-ups on large tables (larger than buffer sizes) than on very small tables.
- **Use `EXPLAIN`:** Analyze your queries with the `EXPLAIN` statement to determine if indexes are being used effectively and identify columns that might benefit from indexing.
- **`LIKE '%word%'`:** Queries using a leading wildcard in a `LIKE` clause (e.g., `LIKE '%word%'`) typically cannot use standard BTREE indexes effectively and may result in full table scans unless a full-text index is used.
- **Delayed Writes:** For tables with many reads and writes, consider storage engine options or server configurations related to delayed writes to potentially improve performance by batching disk I/O. (This is an advanced topic.)
- **Creating Indexes on Existing Tables:** Use `CREATE INDEX index_name ON table_name (column_list);`
- **Large Tables:** For very large tables, it's often faster to load data into the table first and then create indexes, rather than creating indexes on an empty table and then loading data.

Viewing Indexes

- **`SHOW INDEX FROM table_name;`** : Displays information about all indexes on a table.

```
SHOW INDEX FROM Employees;
```

- **`SHOW CREATE TABLE table_name;`** : Shows the `CREATE TABLE` statement, which includes definitions for all indexes.SQL

```
SHOW CREATE TABLE Employees\G
```

When to Remove an Index

Remove an index if:

- It is rarely or never used. Unused indexes still incur overhead during data modification operations.
- **Identifying Unused Indexes:**
  - If user statistics are enabled, query the `information_schema.INDEX_STATISTICS` table.
  - If the slow query log is enabled and the `log_queries_not_using_indexes` server system variable is `ON`, queries performing full table scans will be logged, which can indicate missing or ineffective indexes.

This page is licensed: CC BY-SA / Gnu FDL

Joining Tables with JOIN Clauses

This guide offers a simple, hands-on introduction to three basic `JOIN` types in MariaDB: `INNER JOIN`, `CROSS JOIN`, and `LEFT JOIN`. Use these examples to understand how different joins combine data from multiple tables based on specified conditions.

Setup: Example Tables and Data

First, create and populate two simple tables, `t1` and `t2`, to use in the `JOIN` examples:

```
CREATE TABLE t1 ( a INT );
CREATE TABLE t2 ( b INT );

INSERT INTO t1 VALUES (1), (2), (3);
INSERT INTO t2 VALUES (2), (4);
```

JOIN Examples and Output

Below are examples of different `JOIN` types using the tables `t1` and `t2`.

INNER JOIN

An `INNER JOIN` produces a result set containing only rows that have a match in both tables for the specified join condition(s).

```
SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.b;
```

Output:

```
+------+------+
| a    | b    |
+------+------+
|    2 |    2 |
+------+------+
1 row in set (0.00 sec)
```

Explanation: Only the row where `t1.a` (value 2) matches `t2.b` (value 2) is returned.

CROSS JOIN

A `CROSS JOIN` produces a result set in which every row from the first table is joined to every row in the second table. This is also known as a Cartesian product.

```
SELECT * FROM t1 CROSS JOIN t2;
```

Output:

```
+------+------+
| a    | b    |
+------+------+
|    1 |    2 |
|    2 |    2 |
|    3 |    2 |
|    1 |    4 |
|    2 |    4 |
|    3 |    4 |
+------+------+
6 rows in set (0.00 sec)
```

Explanation: Each of the 3 rows in `t1` is combined with each of the 2 rows in `t2`, resulting in 3 * 2 = 6 rows. *Note:* In MariaDB, the `CROSS` keyword can often be omitted if no `ON` clause is present (e.g., `SELECT * FROM t1 JOIN t2;` or `SELECT * FROM t1, t2;` would also produce a Cartesian product).

LEFT JOIN (t1 LEFT JOIN t2)

A `LEFT JOIN` (or `LEFT OUTER JOIN`) produces a result set with all rows from the "left" table (`t1` in this case). If a match is found in the "right" table (`t2`), the corresponding columns from the right table are included. If no match is found, these columns are filled with `NULL`.

```
SELECT * FROM t1 LEFT JOIN t2 ON t1.a = t2.b;
```

Output:

```
+------+------+
| a    | b    |
+------+------+
|    1 | NULL |
|    2 |    2 |
|    3 | NULL |
+------+------+
3 rows in set (0.00 sec)
```

Explanation: All rows from `t1` are present. For `t1.a = 1` and `t1.a = 3`, there are no matching `t2.b` values, so `b` is `NULL`. For `t1.a = 2`, a match is found (`t2.b = 2`), so `b` is `2`.

> Starting with MariaDB 12.3, multiple independent `LEFT JOIN` operations can be reordered by the optimizer for better performance if the reorder_outer_joins flag is enabled in optimizer_switch. It is also recommended to set `optimizer_prune_level=0` to prevent heuristic pruning from eliminating optimal join orders.

RIGHT JOIN (t1 RIGHT JOIN t2)

A `RIGHT JOIN` (or `RIGHT OUTER JOIN`) produces a result set with all rows from the "right" table (`t2` in this case). If a match is found in the "left" table (`t1`), the corresponding columns from the left table are included. If no match is found, these columns are filled with `NULL`.

```
SELECT * FROM t1 RIGHT JOIN t2 ON t1.a = t2.b;
```

Output:

```
+------+------+
| a    | b    |
+------+------+
|    2 |    2 |
| NULL |    4 |
+------+------+
2 rows in set (0.00 sec)
```

LEFT JOIN (t2 LEFT JOIN t1) - Simulating a RIGHT JOIN

This example uses a `LEFT JOIN` but with `t2` as the left table. This effectively demonstrates how a `RIGHT JOIN` would behave if `t1` were the left table and `t2` the right. A `RIGHT JOIN` includes all rows from the "right" table and `NULL`s for non-matching "left" table columns.

```
SELECT * FROM t2 LEFT JOIN t1 ON t1.a = t2.b;
```

Output:

```
+------+------+
| b    | a    |
+------+------+
|    2 |    2 |
|    4 | NULL |
+------+------+
2 rows in set (0.00 sec)
```

Explanation: All rows from `t2` are present. For `t2.b = 2`, a match is found (`t1.a = 2`), so `a` is `2`. For `t2.b = 4`, there is no matching `t1.a` value, so `a` is `NULL`.

Older (Implicit) JOIN Syntax

The first two `SELECT` statements (`INNER JOIN` and `CROSS JOIN`) are sometimes written using an older, implicit join syntax:

- **Implicit INNER JOIN:**

  ```
  SELECT * FROM t1, t2 WHERE t1.a = t2.b;
  ```

  This is equivalent to `SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.b;`.

- **Implicit CROSS JOIN (Cartesian Product):**

  ```
  SELECT * FROM t1, t2;
  ```

  This is equivalent to `SELECT * FROM t1 CROSS JOIN t2;`.

While this syntax works, the explicit `JOIN` syntax (`INNER JOIN`, `LEFT JOIN`, etc.) with an `ON` clause is generally preferred for clarity and to better distinguish join conditions from filtering conditions (`WHERE` clause).

Understanding JOIN Types Summary

- **`INNER JOIN`**: Returns rows only when there is a match in both tables based on the join condition.
- **`CROSS JOIN`**: Returns the Cartesian product of the two tables (all possible combinations of rows).
- **`LEFT JOIN` (Outer Join)**: Returns all rows from the left table, and the matched rows from the right table. If there is no match, `NULL` is returned for columns from the right table.

- **`RIGHT JOIN` (Outer Join)**: Returns all rows from the right table, and the matched rows from the left table. If there is no match, `NULL` is returned for columns from the left table. (The example `SELECT * FROM t2 LEFT JOIN t1 ...` shows this behavior from `t1`'s perspective).

Joining Multiple Tables

`JOIN` clauses can be concatenated (chained) to retrieve results from three or more tables by progressively joining them.

See Also

- More Advanced Joins
- JOIN Syntax
- Comma vs JOIN
- Joins, Subqueries and SET

The initial version of this article was copied, with permission, from Introduction_to_Joins on 2012-10-05.

This page is licensed: CC BY-SA / Gnu FDL

Advanced Joins

This article is a continuation of the Joining Tables with JOIN Clauses Guide. If you're getting started with `JOIN` statements, review that page first.

The Employee Database

Let us begin by using an example employee database of a fairly small family business, which does not anticipate expanding in the future.

First, we create the table that will hold all of the employees and their contact information:

```
CREATE TABLE `Employees` (
  `ID` TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `First_Name` VARCHAR(25) NOT NULL,
  `Last_Name` VARCHAR(25) NOT NULL,
  `Position` VARCHAR(25) NOT NULL,
  `Home_Address` VARCHAR(50) NOT NULL,
  `Home_Phone` VARCHAR(12) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM;
```

Next, we add a few employees to the table:

```
INSERT INTO `Employees` (`First_Name`, `Last_Name`, `Position`, `Home_Address`, `Home_Phone`)
  VALUES
  ('Mustapha', 'Mond', 'Chief Executive Officer', '692 Promiscuous Plaza', '326-555-3492'),
  ('Henry', 'Foster', 'Store Manager', '314 Savage Circle', '326-555-3847'),
  ('Bernard', 'Marx', 'Cashier', '1240 Ambient Avenue', '326-555-8456'),
  ('Lenina', 'Crowne', 'Cashier', '281 Bumblepuppy Boulevard', '328-555-2349'),
  ('Fanny', 'Crowne', 'Restocker', '1023 Bokanovsky Lane', '326-555-6329'),
  ('Helmholtz', 'Watson', 'Janitor', '944 Soma Court', '329-555-2478');
```

Now, we create a second table, containing the hours which each employee clocked in and out during the week:

```
CREATE TABLE `Hours` (
  `ID` TINYINT(3) UNSIGNED NOT NULL,
  `Clock_In` DATETIME NOT NULL,
  `Clock_Out` DATETIME NOT NULL
) ENGINE=MyISAM;
```

Finally, although it is a lot of information, we add a full week of hours for each of the employees into the second table that we created:

```
INSERT INTO `Hours`
  VALUES
  ('1', '2005-08-08 07:00:42', '2005-08-08 17:01:36'),
  ('1', '2005-08-09 07:01:34', '2005-08-09 17:10:11'),
  ('1', '2005-08-10 06:59:56', '2005-08-10 17:09:29'),
  ('1', '2005-08-11 07:00:17', '2005-08-11 17:00:47'),
  ('1', '2005-08-12 07:02:29', '2005-08-12 16:59:12'),
  ('2', '2005-08-08 07:00:25', '2005-08-08 17:03:13'),
  ('2', '2005-08-09 07:00:57', '2005-08-09 17:05:09'),
  ('2', '2005-08-10 06:58:43', '2005-08-10 16:58:24'),
  ('2', '2005-08-11 07:01:58', '2005-08-11 17:00:45'),
  ('2', '2005-08-12 07:02:12', '2005-08-12 16:58:57'),
  ('3', '2005-08-08 07:00:12', '2005-08-08 17:01:32'),
  ('3', '2005-08-09 07:01:10', '2005-08-09 17:00:26'),
  ('3', '2005-08-10 06:59:53', '2005-08-10 17:02:53'),
  ('3', '2005-08-11 07:01:15', '2005-08-11 17:04:23'),
  ('3', '2005-08-12 07:00:51', '2005-08-12 16:57:52'),
  ('4', '2005-08-08 06:54:37', '2005-08-08 17:01:23'),
  ('4', '2005-08-09 06:58:23', '2005-08-09 17:00:54'),
  ('4', '2005-08-10 06:59:14', '2005-08-10 17:00:12'),
  ('4', '2005-08-11 07:00:49', '2005-08-11 17:00:34'),
  ('4', '2005-08-12 07:01:09', '2005-08-12 16:58:29'),
  ('5', '2005-08-08 07:00:04', '2005-08-08 17:01:43'),
  ('5', '2005-08-09 07:02:12', '2005-08-09 17:02:13'),
  ('5', '2005-08-10 06:59:39', '2005-08-10 17:03:37'),
  ('5', '2005-08-11 07:01:26', '2005-08-11 17:00:03'),
  ('5', '2005-08-12 07:02:15', '2005-08-12 16:59:02'),
  ('6', '2005-08-08 07:00:12', '2005-08-08 17:01:02'),
  ('6', '2005-08-09 07:03:44', '2005-08-09 17:00:00'),
  ('6', '2005-08-10 06:54:19', '2005-08-10 17:03:31'),
  ('6', '2005-08-11 07:00:05', '2005-08-11 17:02:57'),
  ('6', '2005-08-12 07:02:07', '2005-08-12 16:58:23');
```

Working with the Employee Database

Now that we have a cleanly structured database to work with, let us begin this tutorial by stepping up one notch from the last tutorial and filtering our information a little.

Filtering by Name

Earlier in the week, an anonymous employee reported that Helmholtz came into work almost four minutes late; to verify this, we will begin our investigation by filtering out employees whose first names are “Helmholtz”:

```
SELECT
  `Employees`.`First_Name`,
  `Employees`.`Last_Name`,
  `Hours`.`Clock_In`,
  `Hours`.`Clock_Out`
FROM `Employees`
INNER JOIN `Hours` ON `Employees`.`ID` = `Hours`.`ID`
WHERE `Employees`.`First_Name` = 'Helmholtz';
```

The result looks like this:

```
+------------+-----------+---------------------+---------------------+
| First_Name | Last_Name | Clock_In            | Clock_Out           |
+------------+-----------+---------------------+---------------------+
| Helmholtz  | Watson    | 2005-08-08 07:00:12 | 2005-08-08 17:01:02 |
| Helmholtz  | Watson    | 2005-08-09 07:03:44 | 2005-08-09 17:00:00 |
| Helmholtz  | Watson    | 2005-08-10 06:54:19 | 2005-08-10 17:03:31 |
| Helmholtz  | Watson    | 2005-08-11 07:00:05 | 2005-08-11 17:02:57 |
| Helmholtz  | Watson    | 2005-08-12 07:02:07 | 2005-08-12 16:58:23 |
+------------+-----------+---------------------+---------------------+
5 rows in set (0.00 sec)
```

This is obviously more information than we care to trudge through, considering we only care about when he arrived past 7:00:59 on any given day within this week; thus, we need to add a couple more conditions to our `WHERE` clause.

Filtering by Name, Date and Time

In the following example, we will filter out all of the times which Helmholtz clocked in that were before 7:01:00 and during the work week that lasted from the 8th to the 12th of August:

```
SELECT
  `Employees`.`First_Name`,
  `Employees`.`Last_Name`,
  `Hours`.`Clock_In`,
  `Hours`.`Clock_Out`
FROM `Employees`
INNER JOIN `Hours` ON `Employees`.`ID` = `Hours`.`ID`
WHERE `Employees`.`First_Name` = 'Helmholtz'
AND DATE_FORMAT(`Hours`.`Clock_In`, '%Y-%m-%d') >= '2005-08-08'
AND DATE_FORMAT(`Hours`.`Clock_In`, '%Y-%m-%d') <= '2005-08-12'
AND DATE_FORMAT(`Hours`.`Clock_In`, '%H:%i:%S') > '07:00:59';
```

The result looks like this:

```
+------------+-----------+---------------------+---------------------+
| First_Name | Last_Name | Clock_In            | Clock_Out           |
+------------+-----------+---------------------+---------------------+
| Helmholtz  | Watson    | 2005-08-09 07:03:44 | 2005-08-09 17:00:00 |
| Helmholtz  | Watson    | 2005-08-12 07:02:07 | 2005-08-12 16:58:23 |
+------------+-----------+---------------------+---------------------+
2 rows in set (0.00 sec)
```

By merely adding a few more conditions, we eliminated all of the irrelevant information; Helmholtz was late to work on the 9th and the 12th of August.

Displaying Total Work Hours per Day

Suppose you would like to—based on the information stored in both of our tables in the employee database—develop a quick list of the total hours each employee has worked for each day recorded; a simple way to estimate the time each employee worked per day is exemplified below:

```
SELECT
  `Employees`.`ID`,
  `Employees`.`First_Name`,
  `Employees`.`Last_Name`,
  `Hours`.`Clock_In`,
  `Hours`.`Clock_Out`,
DATE_FORMAT(`Hours`.`Clock_Out`, '%T')-DATE_FORMAT(`Hours`.`Clock_In`, '%T')
AS 'Total_Hours'
FROM `Employees`
INNER JOIN `Hours` ON `Employees`.`ID` = `Hours`.`ID`;
```

The result (limited to 10 rows) looks like this:

```
+----+------------+-----------+---------------------+---------------------+-------------+
| ID | First_Name | Last_Name | Clock_In            | Clock_Out           | Total_Hours |
+----+------------+-----------+---------------------+---------------------+-------------+
|  1 | Mustapha   | Mond      | 2005-08-08 07:00:42 | 2005-08-08 17:01:36 |          10 |
|  1 | Mustapha   | Mond      | 2005-08-09 07:01:34 | 2005-08-09 17:10:11 |          10 |
|  1 | Mustapha   | Mond      | 2005-08-10 06:59:56 | 2005-08-10 17:09:29 |          11 |
|  1 | Mustapha   | Mond      | 2005-08-11 07:00:17 | 2005-08-11 17:00:47 |          10 |
|  1 | Mustapha   | Mond      | 2005-08-12 07:02:29 | 2005-08-12 16:59:12 |           9 |
|  2 | Henry      | Foster    | 2005-08-08 07:00:25 | 2005-08-08 17:03:13 |          10 |
|  2 | Henry      | Foster    | 2005-08-09 07:00:57 | 2005-08-09 17:05:09 |          10 |
|  2 | Henry      | Foster    | 2005-08-10 06:58:43 | 2005-08-10 16:58:24 |          10 |
|  2 | Henry      | Foster    | 2005-08-11 07:01:58 | 2005-08-11 17:00:45 |          10 |
|  2 | Henry      | Foster    | 2005-08-12 07:02:12 | 2005-08-12 16:58:57 |           9 |
+----+------------+-----------+---------------------+---------------------+-------------+
10 rows in set (0.00 sec)
```

See Also

- Joining Tables with JOIN Clauses Guide
- JOIN Syntax
- Comma vs JOIN
- Joins, Subqueries and SET

The first version of this article was copied, with permission, from More_Advanced_Joins on 2012-10-05.

This page is licensed: CC BY-SA / Gnu FDL

Configuring MariaDB for Remote Client Access Guide

This guide explains how to configure your MariaDB server to accept connections from remote hosts. Learn to adjust crucial network settings like `bind-address`, grant appropriate user privileges for remote connections, and configure essential firewall rules.

Understanding Key Network Directives

Two main configuration directives control MariaDB's network accessibility:

- **`skip-networking`**: If this directive is enabled, MariaDB will not listen for TCP/IP connections at all. All interaction must be through local mechanisms like Unix sockets or named pipes.
- **`bind-address`**: This directive specifies the IP address the server listens on.
  - By default, for security, many MariaDB packages bind to `127.0.0.1` (localhost). This means the server will only accept connections originating from the server machine itself via the loopback interface. Remote connections will fail.
  - If `bind-address` is set to `127.0.0.1`, attempting to connect from another host, or even from the same host using a non-loopback IP address, will result in errors like:

    ```
    ERROR 2002 (HY000): Can't connect to MySQL server on 'myhost' (115)
    ```

    A `telnet myhost 3306` test would likely show "Connection refused."
  - To allow connections from other hosts, you must either comment out the `bind-address` directive (making MariaDB listen on all available network interfaces, i.e., `0.0.0.0` for IPv4), or set it to a specific public IP address of the server.
  - **MariaDB 10.11 and later:** `bind-address` can accept multiple comma-separated IP addresses, allowing the server to listen on specific interfaces while excluding others.

Connecting via `localhost` typically works even if `bind-address` is `127.0.0.1` (using the loopback interface):

```
./client/mariadb --host=localhost --protocol=tcp --port=3306 test
```

Locating the MariaDB Configuration File

To change these network settings, you need to edit MariaDB's configuration file (often named `my.cnf` or `my.ini`).

- See Configuring MariaDB with my.cnf for comprehensive details.
- **Common Locations:**
  - `/etc/my.cnf` (Unix/Linux/BSD)
  - `/etc/mysql/my.cnf` (Common on Debian/Ubuntu)
  - `$MYSQL_HOME/my.cnf` (Unix/Linux/BSD, where `$MYSQL_HOME` is MariaDB's base directory)
  - `SYSCONFDIR/my.cnf` (Compile-time specified system configuration directory)
  - `DATADIR\my.ini` (Windows, in the data directory)
  - `~/.my.cnf` (User-specific configuration file)
- **Identifying Loaded Files:** To see which configuration files your `mariadbd` server instance reads and in what order, execute:

  Bash

  ```
  mariadbd --help --verbose
  ```

  Look for a line similar to: `Default options are read from the following files in the given order: /etc/my.cnf /etc/mysql/my.cnf ~/.my.cnf`

Modifying the Configuration File for Remote Access

1. **Open the File:** Use a text editor to open the primary configuration file identified (e.g., `/etc/mysql/my.cnf`).
2. **Locate `[mysqld]` Section:** Find the section starting with `[mysqld]`.
3. **Adjust Directives:**
   - If `skip-networking` is present and enabled (not commented out with `#`), comment it out or set it to `0`:Ini, TOML

     ```
     #skip-networking
     ```

     orIni, TOML

     ```
     skip-networking=0
     ```

   - If `bind-address = 127.0.0.1` (or another loopback/specific IP that's too restrictive) is present:
     - To listen on all available IPv4 interfaces: Comment it out entirely (`#bind-address = 127.0.0.1`) or set `bind-address = 0.0.0.0`.
     - To listen on a specific public IP address of your server: `bind-address = <your_server_public_ip>`.
     - Alternatively, to effectively disable binding to a specific address and listen on all, you can add `skip-bind-address`. Example changes:

```
[mysqld]
...
#skip-networking
#bind-address = 127.0.0.1
...
```

Or, to be explicit for listening on all interfaces if `bind-address` was previously restrictive:

```
[mysqld]
bind-address = 0.0.0.0
```

4. **Save and Restart:** Save the configuration file and restart the MariaDB server service.
   - See Starting and Stopping MariaDB for instructions.
5. **Verify Settings (Optional):** You can check the options `mariadbd` is effectively using by running:

```
./sql/mariadbd --print-defaults  # Adjust path to mariadbd if necessary
```

Look for the effective `bind-address` value or the absence of `skip-networking`. If multiple `[mysqld]` sections or `skip-bind-address` are used, the last specified prevailing value is typically what counts.

Granting User Privileges for Remote Connections

Configuring the server to listen for remote connections is only the first step. You must also grant privileges to user accounts to connect from specific remote hosts. MariaDB user accounts are defined as `'username'@'hostname'`.

1. **Connect to MariaDB:**

```
mariadb -u root -p
```

2. **View Existing Remote Users (Optional):**SQL

```
SELECT User, Host FROM mysql.user
WHERE Host <> 'localhost' AND Host <> '127.0.0.1' AND Host <> '::1';
```

3. **Grant Privileges:** Use the `GRANT` statement to allow a user to connect from a remote host or a range of hosts.
   - **Syntax Elements:**
     - Privileges (e.g., `ALL PRIVILEGES`, `SELECT, INSERT, UPDATE`)
     - Database/tables (e.g., `database_name.*` for all tables in a database, `*.*` for all databases)
     - Username
     - Host (IP address, hostname, or subnet with wildcards like `%`)
     - Password (using `IDENTIFIED BY 'password'`)
   - **Example: Grant `root`-like access from a specific LAN subnet:** It's highly discouraged to allow `root` access from all hosts (`'root'@'%'`) directly to the internet. Instead, restrict it to trusted networks if necessary.

     SQL

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'192.168.100.%'
  IDENTIFIED BY 'my-very-strong-password' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

     This allows the `root` user to connect from any IP address in the `192.168.100.x` subnet. Replace `'my-very-strong-password'` with a strong, unique password.

- For creating less privileged users or more granular permissions, see the GRANT documentation.

Configuring Your Firewall

Even if MariaDB is configured for remote access, a firewall on the server (software or hardware) might block incoming connections on MariaDB's port (default is `3306`).

- **RHEL/CentOS 7 Example (using `firewall-cmd`):**

  ```
  sudo firewall-cmd --add-port=3306/tcp --permanent
  sudo firewall-cmd --reload
  ```

  The first command adds the rule, the second makes it persist after reboots and applies the changes. Consult your OS/firewall documentation for specific commands.

Important Considerations and Reverting Changes

- **Security:** Opening MariaDB to remote connections, especially to the internet, increases security risks. Always use strong passwords, grant minimal necessary privileges, and restrict host access as much as possible. Consider using TLS/SSL for encrypted connections (see Secure Connections Overview).
- **Reverting:** To disable remote access and revert to a more secure local-only setup:
  1. Edit your MariaDB configuration file.
  2. Ensure `skip-networking` is not enabled (or is `0`).
  3. Set `bind-address = 127.0.0.1` explicitly, or remove any `skip-bind-address` directive if you previously added it to listen on all interfaces. The goal is to have `bind-address=127.0.0.1` as the effective setting.
  4. Restart the MariaDB server.
  5. Review and revoke any unnecessary remote `GRANT` privileges.

The initial version of this article was copied, with permission, from Remote_Clients_Cannot_Connect on 2012-10-30.

This page is licensed: CC BY-SA / Gnu FDL

Getting Data Guide

This guide explains how to retrieve data from MariaDB using the `SELECT` statement, progressing from basic syntax to more involved queries. Learn to select specific columns, limit results, filter with `WHERE`, sort with `ORDER BY`, join tables, and use various helpful options and functions.

Setup: Creating and Populating Example Tables

To follow the examples, first create and populate the `books` and `authors` tables:

```
CREATE OR REPLACE TABLE books (
    isbn CHAR(20) PRIMARY KEY,
    title VARCHAR(50),
    author_id INT,
    publisher_id INT,
    year_pub CHAR(4),
    description TEXT
);

CREATE OR REPLACE TABLE authors (
    author_id INT AUTO_INCREMENT PRIMARY KEY,
    name_last VARCHAR(50),
    name_first VARCHAR(50),
    country VARCHAR(50)
);

INSERT INTO authors (name_last, name_first, country) VALUES
  ('Kafka', 'Franz', 'Czech Republic'),
  ('Dostoevsky', 'Fyodor', 'Russia');

INSERT INTO books (title, author_id, isbn, year_pub) VALUES
 ('The Trial', 1, '0805210407', '1995'),
 ('The Metamorphosis', 1, '0553213695', '1995'),
 ('America', 2, '0805210644', '1995'), -- Note: Original data had author_id 2 for 'America', Dostoevsky
is author_id 2.
 ('Brothers Karamozov', 2, '0553212168', ''),
 ('Crime & Punishment', 2, '0679420290', ''),
 ('Crime & Punishment', 2, '0553211757', ''),
 ('Idiot', 2, '0192834118', ''),
 ('Notes from Underground', 2, '067973452X', '');
```

Basic Data Retrieval

Selecting All Columns:

Use * to select all columns from a table.

```
SELECT * FROM books;
```

Output (example):

```
+------------+-----------------------+-----------+--------------+----------+-------------+
| isbn       | title                 | author_id | publisher_id | year_pub | description |
+------------+-----------------------+-----------+--------------+----------+-------------+
| 0192834118 | Idiot                 |         2 |         NULL |          | NULL        |
| 0553211757 | Crime & Punishment    |         2 |         NULL |          | NULL        |
... (other rows)
| 0805210644 | America               |         2 |         NULL | 1995     | NULL        |
+------------+-----------------------+-----------+--------------+----------+-------------+
8 rows in set (0.001 sec)
```

Selecting Specific Columns:

List the column names separated by commas.

```
SELECT isbn, title, author_id FROM books;
```

Output (example):

```
+------------+-----------------------+-----------+
| isbn       | title                 | author_id |
+------------+-----------------------+-----------+
| 0192834118 | Idiot                 |         2 |
| 0553211757 | Crime & Punishment    |         2 |
... (other rows)
+------------+-----------------------+-----------+
8 rows in set (0.001 sec)
```

Limiting the Number of Rows with `LIMIT`:

- To get the first `N` rows:

  ```
  SELECT isbn, title, author_id FROM books LIMIT 5;
  ```

  Output (example):

  ```
  +------------+--------------------+-----------+
  | isbn       | title              | author_id |
  +------------+--------------------+-----------+
  | 0192834118 | Idiot              |         2 |
  | 0553211757 | Crime & Punishment |         2 |
  | 0553212168 | Brothers Karamozov |         2 |
  | 0553213695 | The Metamorphosis  |         1 |
  | 0679420290 | Crime & Punishment |         2 |
  +------------+--------------------+-----------+
  5 rows in set (0.001 sec)
  ```

- To get `N` rows starting from an offset (offset is 0-indexed):SQL

  ```
  SELECT isbn, title, author_id FROM books LIMIT 5, 10; -- Skip 5 rows, show next 10 (or fewer if less
  remain)
  ```

  Output (example, assuming only 3 more rows exist after offset 5):

  ```
  +------------+-----------------------+-----------+
  | isbn       | title                 | author_id |
  +------------+-----------------------+-----------+
  | 067973452X | Notes from Underground |         2 |
  | 0805210407 | The Trial             |         1 |
  | 0805210644 | America               |         2 |
  +------------+-----------------------+-----------+
  3 rows in set (0.001 sec)
  ```

Filtering and Ordering Results

Filtering with `WHERE`:

Use the `WHERE` clause to specify conditions for row selection.

```
SELECT isbn, title
FROM books
WHERE author_id = 2
LIMIT 5;
```

Output (example):

```
+------------+------------------------+
| isbn       | title                  |
+------------+------------------------+
| 0192834118 | Idiot                  |
| 0553211757 | Crime & Punishment     |
| 0553212168 | Brothers Karamozov     |
| 0679420290 | Crime & Punishment     |
| 067973452X | Notes from Underground |
+------------+------------------------+
5 rows in set (0.000 sec)
```

Ordering with `ORDER BY`:

Use `ORDER BY column_name [ASC|DESC]` to sort the result set.

```
SELECT isbn, title
FROM books
WHERE author_id = 2
ORDER BY title ASC
LIMIT 5;
```

Output (example):

```
+------------+--------------------+
| isbn       | title              |
+------------+--------------------+
| 0805210644 | America            |
| 0553212168 | Brothers Karamozov |
| 0553211757 | Crime & Punishment |
| 0679420290 | Crime & Punishment |
| 0192834118 | Idiot              |
+------------+--------------------+
5 rows in set (0.001 sec)
```

- `ASC` (ascending) is the default order. `DESC` is for descending order.
- You can order by multiple columns: `ORDER BY col1 ASC, col2 DESC`.
- **Clause Order:** `SELECT ... FROM ... WHERE ... ORDER BY ... LIMIT ...`. MariaDB generally processes `WHERE`, then `ORDER BY`, then `LIMIT`.

Working with Multiple Tables (JOINs) and Functions

Joining Tables:

Use JOIN to combine rows from two or more tables based on a related column.

```
SELECT isbn, title, CONCAT(name_first, ' ', name_last) AS author
FROM books
JOIN authors USING (author_id) -- Assumes 'author_id' column exists in both tables
WHERE name_last = 'Dostoevsky'
ORDER BY title ASC
LIMIT 5;
```

Output (example):

```
+------------+--------------------+-------------------+
| isbn       | title              | author            |
+------------+--------------------+-------------------+
| 0805210644 | America            | Fyodor Dostoevsky |
| 0553212168 | Brothers Karamozov | Fyodor Dostoevsky |
| 0553211757 | Crime & Punishment | Fyodor Dostoevsky |
| 0679420290 | Crime & Punishment | Fyodor Dostoevsky |
| 0192834118 | Idiot              | Fyodor Dostoevsky |
+------------+--------------------+-------------------+
5 rows in set (0.00 sec)
```

- Alternative `JOIN` syntax: `... JOIN authors ON books.author_id = authors.author_id ...`. For more on joins, see the JOIN Syntax documentation or a "Basic Joins Guide".
- **`CONCAT(str1, str2, ...)`:** Concatenates strings.
- **`AS alias_name`:** Assigns an alias to an output column.

Pattern Matching with LIKE:

Use LIKE in the WHERE clause for pattern matching. % is a wildcard for zero or more characters.

```
SELECT isbn, title, CONCAT(name_first, ' ', name_last) AS author
FROM books
JOIN authors USING (author_id)
WHERE name_last LIKE 'Dostoevsk%'
ORDER BY title ASC
LIMIT 5;
```

Output (example, same as above if only Dostoevsky matches):

```
+------------+--------------------+-------------------+
| isbn       | title              | author            |
+------------+--------------------+-------------------+
| 0805210644 | America            | Fyodor Dostoevsky |
| 0553212168 | Brothers Karamozov | Fyodor Dostoevsky |
| 0553211757 | Crime & Punishment | Fyodor Dostoevsky |
| 0679420290 | Crime & Punishment | Fyodor Dostoevsky |
| 0192834118 | Idiot              | Fyodor Dostoevsky |
+------------+--------------------+-------------------+
5 rows in set (0.001 sec)
```

SELECT Statement Modifiers

Place these modifiers immediately after the `SELECT` keyword.

- **`ALL` vs `DISTINCT`:**
  - `ALL` (default): Returns all rows that meet the criteria.
  - `DISTINCT`: Returns only unique rows for the selected columns. If multiple identical rows are found for the specified columns, only the first one is displayed.

```
SELECT DISTINCT title
FROM books
JOIN authors USING (author_id)
WHERE name_last = 'Dostoevsky'
ORDER BY title;
```

Output (example, showing one "Crime & Punishment"):

```
+-----------------------+
| title                 |
+-----------------------+
| America               |
| Brothers Karamozov    |
| Crime & Punishment    |
| Idiot                 |
| Notes from Underground |
+-----------------------+
```

- `HIGH_PRIORITY`:

  Gives the `SELECT` statement higher priority over concurrent data modification statements (use with caution as it can impact write performance).

  SQL

```
SELECT DISTINCT HIGH_PRIORITY title
FROM books
JOIN authors USING (author_id)
WHERE name_last = 'Dostoevsky'
ORDER BY title;
```

- `SQL_CALC_FOUND_ROWS` and `FOUND_ROWS()`:

  To find out how many rows a query would have returned without a `LIMIT` clause, use `SQL_CALC_FOUND_ROWS` in your `SELECT` statement, and then execute `SELECT FOUND_ROWS()`; immediately after.

```
SELECT SQL_CALC_FOUND_ROWS isbn, title
FROM books
JOIN authors USING (author_id)
WHERE name_last = 'Dostoevsky'
ORDER BY title -- Order before limit to ensure consistent FOUND_ROWS() for a given query logic
LIMIT 5;
```

  Output (example for the first query):

```
+------------+-----------------------+
| isbn       | title                 |
+------------+-----------------------+
| 0805210644 | America               |
| 0553212168 | Brothers Karamozov    |
| 0553211757 | Crime & Punishment    |
| 0679420290 | Crime & Punishment    |
| 0192834118 | Idiot                 |
+------------+-----------------------+
5 rows in set (0.001 sec)
```

  Then, to get the total count:

```
SELECT FOUND_ROWS();
```

  Output (example, if 6 Dostoevsky books in total):

```
+--------------+
| FOUND_ROWS() |
+--------------+
|            6 |
+--------------+
1 row in set (0.000 sec)
```

  The value from `FOUND_ROWS()` is temporary and specific to the current session.

See Also

- SELECT — the full statement reference, including `LIMIT`, `ORDER BY`, `DISTINCT`, and `GROUP BY`
- JOIN Syntax
- A MariaDB Primer and MariaDB Basics

This page is licensed: CC BY-SA / Gnu FDL

Basic SQL Statements Guide

This guide provides a quick overview of essential SQL statements in MariaDB, categorized by their function in data definition, data manipulation, and transaction control. Find brief descriptions and links to detailed documentation for each statement, along with a simple illustrative example sequence.

(If you need a basic tutorial on how to use the MariaDB database server and execute simple commands, see A MariaDB Primer. Also see Essential Queries Guide for examples of commonly-used queries.)

Defining How Your Data Is Stored

These statements are part of the SQL Data Definition Language - DDL.

CREATE DATABASE

Used to create a new, empty database.

DROP DATABASE

Used to completely destroy an existing database.

USE

Used to select a default database for subsequent statements.

CREATE TABLE

Used to create a new table, which is where your data is actually stored.

ALTER TABLE

Used to modify an existing table's definition (e.g., add/remove columns, change types).

DROP TABLE

Used to completely destroy an existing table and all its data.

DESCRIBE (or `DESC`)

Shows the structure of a table (columns, data types, etc.).

Manipulating Your Data

These statements are part of the SQL Data Manipulation Language - DML.

- **SELECT**: Used when you want to read (or select) your data from one or more tables.
- **INSERT**: Used when you want to add (or insert) new rows of data into a table.
- **UPDATE**: Used when you want to change (or update) existing data in a table.
- **DELETE**: Used when you want to remove (or delete) existing rows of data from a table.
- **REPLACE**: Works like `INSERT`, but if an old row in the table has the same value as a new row for a `PRIMARY KEY` or a `UNIQUE` index, the old row is deleted before the new row is inserted.
- **TRUNCATE TABLE**: Used to quickly remove all data from a table, resetting any `AUTO_INCREMENT` values. It is faster than `DELETE` without a `WHERE` clause for emptying a table.

Transactions

These statements are part of the SQL Transaction Control Language - TCL.

- **START TRANSACTION** (or `BEGIN`): Used to begin a new transaction, allowing multiple SQL statements to be treated as a single atomic unit.
- **COMMIT**: Used to save all changes made during the current transaction, making them permanent.
- **ROLLBACK**: Used to discard all changes made during the current transaction, reverting the database to its state before the transaction began.

A Simple Example Sequence

This example demonstrates several of the statements in action:

```
-- Create a new database
CREATE DATABASE mydb;

-- Select the new database to use
USE mydb;

-- Create a new table
CREATE TABLE mytable (
    id INT PRIMARY KEY,
    name VARCHAR(20)
);

-- Insert some data
INSERT INTO mytable VALUES (1, 'Will');
INSERT INTO mytable VALUES (2, 'Marry');
INSERT INTO mytable VALUES (3, 'Dean');

-- Select specific data
SELECT id, name FROM mytable WHERE id = 1;

-- Update existing data
UPDATE mytable SET name = 'Willy' WHERE id = 1;

-- Select all data to see changes
SELECT id, name FROM mytable;

-- Delete specific data
DELETE FROM mytable WHERE id = 1;

-- Select all data again
SELECT id, name FROM mytable;

-- Drop the database (removes the database and its tables)
DROP DATABASE mydb;
```

Common Query: Counting Rows

To count the number of records in a table:

```
SELECT COUNT(*) FROM mytable; -- Or SELECT COUNT(1) FROM mytable;
```

(Note: This query would typically be run on an existing table, for example, before it or its database is dropped.)

The first version of this article was copied, with permission, from Basic_SQL_Statements on 2012-10-05.

This page is licensed: CC BY-SA / Gnu FDL

Basic SQL Debugging Guide

Designing Queries

Following a few conventions makes finding errors in queries a lot easier, especially when you ask for help from people who might know SQL, but know nothing about your particular schema. A query easy to read is a query easy to debug. Use whitespace to group clauses within the query. Choose good table and field aliases to add clarity, not confusion. Choose the syntax that supports the query's meaning.

Using Whitespace

A query hard to read is a query hard to debug. White space is free. New lines and indentation make queries easy to read, particularly when constructing a query inside a scripting language, where variables are interspersed throughout the query.

There is a syntax error in the following. How fast can you find it?

```
SELECT u.id, u.name, alliance.ally FROM users u JOIN alliance ON
(u.id=alliance.userId) JOIN team ON (alliance.teamId=team.teamId
WHERE team.teamName='Legionnaires' AND u.online=1 AND ((u.subscription='paid'
AND u.paymentStatus='current') OR u.subscription='free') ORDER BY u.name;
```

Here's the same query, with correct use of whitespace. Can you find the error faster?

```
SELECT
    u.id
    , u.name
    , alliance.ally
FROM
    users u
    JOIN alliance ON (u.id = alliance.userId)
    JOIN team ON (alliance.teamId = team.teamId
WHERE
    team.teamName = 'Legionnaires'
    AND u.online = 1
    AND (
        (u.subscription = 'paid' AND u.paymentStatus = 'current')
        OR
        u.subscription = 'free'
    )
ORDER BY
    u.name;
```

Even if you don't know SQL, you might still have caught the missing ')' following team.teamId.

The exact formatting style you use isn't so important. You might like commas in the select list to follow expressions, rather than precede them. You might indent with tabs or with spaces. Adherence to some particular form is not important. Legibility is the only goal.

Table and Field Aliases

Aliases allow you to rename tables and fields for use within a query. This can be handy when the original names are very long, and is required for self joins and certain subqueries. However, poorly chosen aliases can make a query harder to debug, rather than easier. Aliases should reflect the original table name, not an arbitrary string.

Bad:

```
SELECT *
FROM
    financial_reportQ_1 AS a
    JOIN sales_renderings AS b ON (a.salesGroup = b.groupId)
    JOIN sales_agents AS c ON (b.groupId = c.group)
WHERE
    b.totalSales > 10000
    AND c.id != a.clientId
```

As the list of joined tables and the `WHERE` clause grow, it becomes necessary to repeatedly look back to the top of the query to see to which table any given alias refers.

Better:

```
SELECT *
FROM
    financial_report_Q_1 AS frq1
    JOIN sales_renderings AS sr ON (frq1.salesGroup = sr.groupId)
    JOIN sales_agents AS sa ON (sr.groupId = sa.group)
WHERE
    sr.totalSales > 10000
    AND sa.id != frq1.clientId
```

Each alias is just a little longer, but the table initials give enough clues that anyone familiar with the database only need see the full table name once, and can generally remember which table goes with which alias while reading the rest of the query.

Placing JOIN Conditions

The manual warns against using the `JOIN` condition (that is, the ON clause) for restricting rows. Some queries, particularly those using implicit joins, take the opposite extreme - all join conditions are moved to the `WHERE` clause. In consequence, the table relationships are mixed with the business logic.

Bad:

```
SELECT *
FROM
    family,
    relationships
WHERE
    family.personId = relationships.personId
    AND relationships.relation = 'father'
```

Without digging through the WHERE clause, it is impossible to say what links the two tables.

Better:

```
SELECT *
FROM
    family
    JOIN relationships ON (family.personId = relationships.personId)
WHERE
    relationships.relation = 'father'
```

The relation between the tables is immediately obvious. The WHERE clause is left to limit rows in the result set.

Compliance with such a restriction negates the use of the comma operator to join tables. It is a small price to pay. Queries should be written using the explicit JOIN keyword anyway, and the two should never be mixed (unless you like rewriting all your queries every time a new version changes operator precedence).

Finding Syntax Errors

Syntax errors are among the easiest problems to solve. MariaDB provides an error message showing the exact point where the parser became confused. Check the query, including a few words before the phrase shown in the error message. Most syntax and parsing errors are obvious after a second look, but some are more elusive, especially when the error text seems empty, points to a valid keyword, or seems to error on syntax that appears exactly correct.

Interpreting the Empty Error

Most syntax errors are easy to interpret. The error generally details the exact source of the trouble. A careful look at the query, with the error message in mind, often reveals an obvious mistake, such as misspelled field names, a missing '`AND`', or an extra closing parenthesis. Sometimes the error is a little less helpful. A frequent, less-than-helpful message:

```
ERROR 1064: You have an error in your SQL syntax; check the manual that corresponds to your
MariaDB server version for the right syntax to use near ' ' at line 1
```

The empty ' ' can be disheartening. Clearly there is an error, but where? A good place to look is at the end of the query. The ' ' suggests that the parser reached the end of the statement while still expecting some syntax token to appear.

Check for missing closers, such as ' and):

```
SELECT * FROM someTable WHERE field = 'value
```

Look for incomplete clauses, often indicated by an exposed comma:

```
SELECT * FROM someTable WHERE field = 1 GROUP BY id,
```

Checking for Keywords

MariaDB allows table and field names and aliases that are also reserved words. To prevent ambiguity, such names must be enclosed in backticks (`):

```
SELECT * FROM actionTable WHERE `DELETE` = 1;
```

If the syntax error is shown near one of your identifiers, check if it appears on the reserved word list.

A text editor with color highlighting for SQL syntax helps to find these errors. When you enter a field name, and it shows up in the same color as the SELECT keyword, you know something is amiss. Some common culprits:

- DESC is a common abbreviation for “description” fields. It means “descending” in a MariaDB ORDER clause.
- DATE, TIME, and TIMESTAMP are all common field names. They are also field types.
- ORDER appears in sales applications. MariaDB uses it to specify sorting for results.

Some keywords are so common that MariaDB makes a special allowance to use them unquoted. My advice: don’t. If it’s a keyword, quote it.

Version Specific Syntax

As MariaDB adds new features, the syntax must change to support them. Most of the time, old syntax will work in newer versions of MariaDB. One notable exception is the change in precedence of the comma operator relative to the JOIN keyword in version 5.0. A query that used to work, such as

```
SELECT * FROM a, b JOIN c ON a.x = c.x;
```

will now fail.

More common, however, is an attempt to use new syntax in an old version. Web hosting companies are notoriously slow to upgrade MariaDB, and you may find yourself using a version several years out of date. The result can be very frustrating when a query that executes flawlessly on your own workstation, running a recent installation, fails completely in your production environment.

This query fails in any version of MySQL prior to 4.1, when subqueries were added to the server:

```
SELECT * FROM someTable WHERE someId IN (SELECT id FROM someLookupTable);
```

This query fails in some early versions of MySQL, because the JOIN syntax did not originally allow an ON clause:

```
SELECT * FROM tableA JOIN tableB ON tableA.x = tableB.y;
```

Always check the installed version of MariaDB, and read the section of the manual relevant for that version. The manual usually indicates exactly when particular syntax became available for use.

The initial version of this article was copied, with permission, from Basic_Debugging on 2012-10-05.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB String Functions Guide

MariaDB has many built-in functions that can be used to manipulate strings of data. With these functions, one can format data, extract certain characters, or use search expressions. Good developers should be aware of the string functions that are available. Therefore, in this article we will go through several string functions, grouping them by similar features, and provide examples of how they might be used.

Formatting

There are several string functions that are used to format text and numbers for nicer display. A popular and very useful function for pasting together the contents of data fields with text is the CONCAT() function. As an example, suppose that a table called contacts has a column for each sales contact's first name and another for the last name. The following SQL statement would put them together:

```
SELECT CONCAT(name_first, ' ', name_last)
AS Name
FROM contacts;
```

This statement will display the first name, a space, and then the last name together in one column. The AS clause will change the column heading of the results to Name.

A less used concatenating function is CONCAT_WS(). It will put together columns with a separator between each. This can be useful when making data available for other programs. For instance, suppose we have a program that will import data, but it requires the fields to be separated by vertical bars. We could just export the data, or we could use a SELECT statement like the one that follows in conjunction with an interface written with an API language like Perl:

```
SELECT CONCAT_WS('|', col1, col2, col3)
FROM table1;
```

The first element above is the separator. The remaining elements are the columns to be strung together.

If we want to format a long number with commas every three digits and a period for the decimal point (e.g., 100,000.00), we can use the function FORMAT() like so:

```
SELECT CONCAT('$', FORMAT(col5, 2))
FROM table3;
```

In this statement, the CONCAT() will place a dollar sign in front of the numbers found in the `col5` column, which are formatted with commas by FORMAT(). The `2` within the FORMAT() stipulates two decimal places.

Occasionally, one will want to convert the text from a column to either all upper-case letters or all lower-case letters. In the example that follows, the output of the first column is converted to upper-case and the second to lower-case:

```
SELECT UCASE(col1),
LCASE(col2)
FROM table4;
```

When displaying data in forms, it's sometimes useful to pad the data displayed with zeros or dots or some other filler. This can be necessary when dealing with VARCHAR columns where the width varies to help the user to see the column limits. There are two functions that may be used for padding: LPAD() and RPAD().

```
SELECT RPAD(part_nbr, 8, '.') AS 'Part Nbr.',
LPAD(description, 15, '_') AS Description
FROM catalog;
```

In this SQL statement, dots are added to the right end of each part number. So a part number of "H200" will display as "H200….", but without the quotes. Each part's description will have under-scores preceding it. A part with a description of "brass hinge" will display as "brass hinge".

If a column is a CHAR data-type, a fixed width column, then it may be necessary to trim any leading or trailing spaces from displays. There are a few functions to accomplish this task. The LTRIM() function will eliminate any leading spaces to the left. So "`H200`" becomes "`H200`". For columns with trailing spaces, spaces on the right, RTRIM() will work: "`H500`" becomes "`H500`". A more versatile trimming function, though, is TRIM(). With it one can trim left, right or both. Below are a few examples:

```
SELECT TRIM(LEADING '.' FROM col1),
TRIM(TRAILING FROM col2),
TRIM(BOTH '_' FROM col3),
TRIM(col4)
FROM table5;
```

In the first TRIM() clause, the padding component is specified; the leading dots are to be trimmed from the output of `col1`. The trailing spaces are trimmed off of `col2`—space is the default. Both leading and trailing under-scores are trimmed from `col3` above. Unless specified, BOTH is the default. So leading and trailing spaces are trimmed from `col4` in the statement here.

Extracting

When there is a need to extract specific elements from a column, MariaDB has a few functions that can help. Suppose a column in the table contacts contains the telephone numbers of sales contacts, including the area-codes, but without any dashes or parentheses. The area-code of each could be extracted for sorting with the LEFT() and the telephone number with the RIGHT() function.

```
SELECT LEFT(telephone, 3) AS area_code,
RIGHT(telephone, 7) AS tel_nbr
FROM contacts
ORDER BY area_code;
```

In the LEFT() function above, the column telephone is given along with the number of characters to extract, starting from the first character on the left in the column. The RIGHT() function is similar, but it starts from the last character on the right, counting left to capture, in this statement, the last seven characters. In the SQL statement above,

area_code is reused to order the results set. To reformat the telephone number, it are necessary to use the SUBSTRING() function.

```
SELECT CONCAT('(', LEFT(telephone, 3), ') ',
SUBSTRING(telephone, 4, 3), '-',
MID(telephone, 7)) AS 'Telephone Number'
FROM contacts
ORDER BY LEFT(telephone, 3);
```

In this SQL statement, the CONCAT() function is employed to assemble some characters and extracted data to produce a common display for telephone numbers (e.g., (504) 555-1234). The first element of the CONCAT() is an opening parenthesis. Next, a LEFT() is used to get the first three characters of telephone, the area-code. After that a closing parenthesis, along with a space is added to the output. The next element uses the SUBSTRING() function to extract the telephone number's prefix, starting at the fourth position, for a total of three characters. Then a dash is inserted into the display. Finally, the function MID() extracts the remainder of the telephone number, starting at the seventh position. The functions MID() and SUBSTRING() are interchangeable and their syntax are the same. By default, for both functions, if the number of characters to capture isn't specified, then it's assumed that the remaining ones are to be extracted.

Manipulating

There are a few functions in MariaDB that can help in manipulating text. One such function is REPLACE(). With it every occurrence of a search parameter in a string can be replaced. For example, suppose we wanted to replace the title *Mrs.* with *Ms.* in a column containing the person's title, but only in the output. The following SQL would do the trick:

```
SELECT CONCAT(REPLACE(title, 'Mrs.', 'Ms.'),
' ', name_first, ' ', name_last) AS Name
FROM contacts;
```

We're using the ever handy CONCAT() function to put together the contact's name with spaces. The REPLACE() function extracts each title and replaces *Mrs.* with *Ms.*, where applicable. Otherwise, for all other titles, it displays them unchanged.

If we want to insert or replace certain text from a column (but not all of its contents), we could use the INSERT() function in conjunction with the LOCATE() function. For example, suppose another contacts table has the contact's title and full name in one column. To change the occurrences of Mrs. to Ms., we could not use REPLACE() since the title is embedded in this example. Instead, we would do the following:

```
SELECT INSERT(name, LOCATE(name, 'Mrs.'), 4, 'Ms.')
FROM contacts;
```

The first element of the INSERT() function is the column. The second element which contains the LOCATE() is the position in the string that text is to be inserted. The third element is optional; it states the number of characters to overwrite. In this case, Mrs. which is four characters is overwritten with Ms. (the final element), which is only three. Incidentally, if 0 is specified, then nothing is overwritten, text is inserted only. As for the LOCATE() function, the first element is the column and the second the search text. It returns the position within the column where the text is found. If it's not found, then 0 is returned. A value of 0 for the position in the INSERT() function negates it and returns the value of name unchanged.

On the odd chance that there is a need to reverse the content of a column, there's the REVERSE() function. You would just place the column name within the function. Another minor function is the REPEAT() function. With it a string may be repeated in the display:

```
SELECT REPEAT(col1, 2)
FROM table1;
```

The first component of the function above is the string or column to be repeated. The second component states the number of times it's to be repeated.

Expression Aids

The function CHAR_LENGTH() is used to determine the number of characters in a string. This could be useful in a situation where a column contains different types of information of specific lengths. For instance, suppose a column in a table for a college contains identification numbers for students, faculty, and staff. If student identification numbers have eight characters while others have less, the following will count the number of student records:

```
SELECT COUNT(school_id)
AS 'Number of Students'
FROM table8
WHERE CHAR_LENGTH(school_id)=8;
```

The COUNT() function above counts the number of rows that meet the condition of the `WHERE` clause.

In a SELECT statement, an ORDER BY clause can be used to sort a results set by a specific column. However, if the column contains IP addresses, a simple sort may not produce the desired results:

```
SELECT ip_address
FROM computers WHERE server='Y'
ORDER BY ip_address LIMIT 3;

+-------------+
| ip_address  |
+-------------+
| 10.0.1.1    |
| 10.0.11.1   |
| 10.0.2.1    |
+-------------+
```

In the limited results above, the IP address 10.0.2.1 should be second. This happens because the column is being sorted lexically and not numerically. The function INET_ATON() will solve this sorting problem.

```
SELECT ip_address
FROM computers WHERE server='Y'
ORDER BY INET_ATON(ip_address) LIMIT 3;
```

Basically, the INET_ATON() function will convert IP addresses to regular numbers for numeric sorting. For instance, if we were to use the function in the list of columns in a SELECT statement, instead of the `WHERE` clause, the address 10.0.1.1 would return 167772417, 10.0.11.1 will return 167774977, and 10.0.2.1 the number 167772673. As a complement to INET_ATON(), the function INET_NTOA() will translate these numbers back to their original IP addresses.

MariaDB is fairly case insensitive, which usually is fine. However, to be able to check by case, the STRCMP() function can be used. It converts the column examined to a string and makes a comparison to the search parameter.

```
SELECT col1, col2
FROM table6
WHERE STRCMP(col3, 'text')=0;
```

If there is an exact match, the function STRCMP() returns 0. So if `col3` here contains "Text", it won't match. Incidentally, if `col3` alphabetically is before the string to which it's compared, a `-1` are returned. If it's after it, a `1` is returned.

When you have list of items in one string, the SUBSTRING_INDEX() can be used to pull out a sub-string of data. As an example, suppose we have a column which has five elements, but we want to retrieve just the first two elements. This SQL statement will return them:

```
SELECT SUBSTRING_INDEX(col4, '|', 2)
FROM table7;
```

The first component in the function above is the column or string to be picked apart. The second component is the delimiter. The third is the number of elements to return, counting from the left. If we want to grab the last two elements, we would use a negative two to instruct MariaDB to count from the right end.

Conclusion

There are more string functions available in MariaDB. A few of the functions mentioned here have aliases or close alternatives. There are also functions for converting between ASCII, binary, hexi-decimal, and octal strings. And there are also string functions related to text encryption and decryption that were not mentioned. However, this article has given you a good collection of common string functions that will assist you in building more powerful and accurate SQL statements.

This page is licensed: CC BY-SA / Gnu FDL

Restoring Data from Dump Files Guide

This guide explains how to restore your MariaDB data from backup files created with `mariadb-dump`. Learn the basic restoration process using the `mariadb` client and a specific technique for selectively restoring a single table while minimizing data loss on other tables.

It's important to understand that `mariadb-dump` is used for creating backup (dump) files, while the `mariadb` client utility is used for restoring data from these files. The dump file contains SQL statements that, when executed, recreate the database structure and/or data.

Basic Restoration Process

To restore a dump file, you direct the `mariadb` client to execute the SQL statements contained within the file.

```
mariadb --user your_username --password < /path/to/your/backupfile.sql
```

- Replace `your_username` with your MariaDB username and `/path/to/your/backupfile.sql` with the actual path to your dump file.
- You will be prompted for the password for `your_username`.
- The `<` symbol is a standard input (STDIN) redirect, feeding the contents of `backupfile.sql` to the `mariadb` client.
- Often, the dump file itself contains `CREATE DATABASE IF NOT EXISTS` and `USE database_name;` statements, so a specific database doesn't always need to be named on the command line during restore. If your dump file restores to a specific database, ensure that user has permissions to it. If the dump file does *not* specify a database, you might need to create the database first and then run:

  ```
  mariadb --user your_username --password your_database_name < /path/to/your/backupfile.sql
  ```

Important Considerations Before Restoring

- **Data Overwriting:** Restoring a dump file will execute the SQL statements within it. If the dump file contains `DROP TABLE` and `CREATE TABLE` statements (common for full backups), existing tables with the same names will be dropped and recreated, leading to loss of any data added or changed since the backup was made.
- **Backup Age:** If your dump file is several days old, restoring it entirely could revert all data in the affected tables/databases to that older state. This can be disastrous if only a small portion of data was lost and the rest has been actively updated.

Always ensure you understand the contents of the dump file and the potential impact before initiating a restore, especially on a production system. Consider testing the restore on a non-production environment first if possible.

Restoring a Single Table Selectively

If only one table has been lost or corrupted and your backup file contains an entire database (or multiple tables), a full restore might overwrite recent, valid data in other tables. Here's a method to restore only a specific table using a temporary user with restricted privileges:

1. **Create a Temporary User:** Create a MariaDB user specifically for this restore operation.
2. **Grant Limited Privileges:**
   - Grant this temporary user the minimal privileges needed for the dump file to execute up to the point of restoring your target table. This might be `SELECT` on all tables in the database if the dump file checks other tables, or simply the ability to `USE` the database.
   - Then, grant `ALL PRIVILEGES` (or specific necessary privileges like `CREATE`, `DROP`, `INSERT`, `SELECT`) *only* on the specific table you want to restore.

   Example SQL to create a temporary user and grant permissions (replace placeholders):

   ```
   -- Connect to MariaDB as an administrative user (e.g., root)
   CREATE USER 'admin_restore_temp'@'localhost' IDENTIFIED BY 'its_very_secure_pwd';

   -- Grant general SELECT on the database (might be needed if dump file structure requires it)
   -- Or, if not needed, ensure the user can at least USE the database.
   GRANT SELECT ON your_database_name.* TO 'admin_restore_temp'@'localhost';

   -- Grant full privileges ONLY on the table to be restored
   GRANT ALL PRIVILEGES ON your_database_name.table_to_restore TO 'admin_restore_temp'@'localhost';

   FLUSH PRIVILEGES;
   ```

3. Restore Using the Temporary User and `--force`:

   Use the mariadb client with the temporary user and the –force option. The –force option tells MariaDB to continue executing statements in the dump file even if some SQL errors occur. Errors will occur for operations on tables where admin_restore_temp lacks permissions, but operations on table_to_restore (where permissions were granted) should succeed.

   Bash

   ```
   mariadb --user admin_restore_temp --password --force your_database_name <
   /path/to/your/fulldumpfile.sql
   ```

   You will be prompted for the password of `admin_restore_temp`.

4. **Verify Restoration:** Check that `table_to_restore` has been correctly restored.
5. **Clean Up:** Drop the temporary user once the restoration is confirmed:

   ```
   DROP USER 'admin_restore_temp'@'localhost';
   ```

This method helps to isolate the restore operation to the intended table, protecting other data from being inadvertently reverted to an older state.

This page is licensed: CC BY-SA / Gnu FDL

Changing Times in MariaDB

This guide explores MariaDB functions for performing calculations and modifications on date and time values. Learn to use functions like `DATE_ADD`, `DATE_SUB`, `TIME_TO_SEC`, and `SEC_TO_TIME` to accurately add or subtract intervals and manage date/time changes that cross midnight or month/year boundaries.

(For foundational knowledge on date and time data types and basic retrieval, please refer to the "Date and Time Handling Guide".)

Calculating Time Across Midnight

When adding hours to a `TIME` value, calculations might exceed 24 hours. For example, if a task is entered at 23:00 and is promised 2 hours later, a simple addition can be problematic.

Consider an `INSERT` statement for a `tickets` table with `entered` and `promised` `TIME` columns:

```
-- Example: Calculating a promised time 2 hours (7200 seconds) from current time
INSERT INTO tickets (client_id, urgency, trouble, ticket_date, entered, promised)
VALUES ('some_client', 'ASAP', 'Issue details',
        CURDATE(), CURTIME(),
        SEC_TO_TIME(TIME_TO_SEC(CURTIME()) + 7200));
```

- `TIME_TO_SEC(time)` converts a time value to seconds.
- `SEC_TO_TIME(seconds)` converts seconds back to a time format (`HHH:MM:SS`).

If `CURTIME()` is `23:00:00` (82,800 seconds), `82800 + 7200 = 90000` seconds. `SEC_TO_TIME(90000)` would result in `25:00:00`. While MariaDB can store this, it doesn't represent a standard clock time for the next day.

Modulo Arithmetic for Time Rollover:

To handle time wrapping around the 24-hour clock (86,400 seconds in a day) for TIME columns, use the modulo operator (%):

```
-- Corrected calculation for 'promised' TIME, wraps around 24 hours
SEC_TO_TIME((TIME_TO_SEC(CURTIME()) + 7200) % 86400)
```

If current time is 23:00, `(82800 + 7200) % 86400` becomes `90000 % 86400`, which is `3600` seconds. `SEC_TO_TIME(3600)` correctly results in `01:00:00`.

Tracking Date Changes with Time: Using `DATETIME`

The modulo arithmetic above gives the correct time of day but doesn't indicate if the promised time falls on the next calendar day. For calculations where the date might change, it's essential to use `DATETIME` (or `TIMESTAMP`) data types.

If your table initially used separate `DATE` and `TIME` columns (e.g., `ticket_date`, `entered_time`, `promised_time`), you would typically alter the table to use `DATETIME` columns (e.g., `entered_datetime`, `promised_datetime`) to store both date and time information accurately. This often involves:

1. Adding new `DATETIME` columns.
2. Populating them by combining the old date and time columns (e.g., using `CONCAT(ticket_date, ' ', entered_time)`).
3. Dropping the old separate date and time columns. *(Always back up your data before such structural changes.)*

With `DATETIME` columns, `NOW()` can be used to get the current date and time.

Adding Durations with `DATE_ADD`

The `DATE_ADD(date, INTERVAL expr unit)` function is the most robust way to add a duration to a date, time, or datetime value. It correctly handles rollovers across days, months, and years.

- `date`: A `DATE`, `DATETIME`, or `TIME` value.
- `expr`: The value of the interval to add.
- `unit`: The unit of the interval (e.g., `HOUR`, `MINUTE`, `DAY`, `MONTH`, `YEAR`, etc.).

Adding Hours (handles date change):

If entered and promised are DATETIME columns:

```
INSERT INTO tickets (client_id, urgency, trouble, entered, promised)
VALUES ('some_client', 'ASAP', 'Issue details',
        NOW(),
        DATE_ADD(NOW(), INTERVAL 2 HOUR));
```

If `NOW()` is `2025-06-03 23:00:00`, `promised` will correctly be `2025-06-04 01:00:00`.

Adding Combined Hours and Minutes:

Use HOUR_MINUTE as the unit. The expr is a string 'hours:minutes'.

```
-- Add 2 hours and 30 minutes
DATE_ADD(NOW(), INTERVAL '2:30' HOUR_MINUTE)
```

If `NOW()` is `2025-06-03 23:00:00`, this results in `2025-06-04 01:30:00`.

Date Calculations Across Months and Years with `DATE_ADD`

`DATE_ADD` also correctly handles date changes across month and year boundaries, including leap years.

Adding Days:

```
-- Add 5 days
DATE_ADD(NOW(), INTERVAL 5 DAY)
```

If `NOW()` is `2025-02-27`, this would result in `2025-03-04` (assuming 2025 is not a leap year).

Adding Combined Days and Hours:

Use DAY_HOUR as the unit. The expr is a string 'days hours'.

```
-- Add 2 days and 6 hours
DATE_ADD(NOW(), INTERVAL '2 6' DAY_HOUR)
```

Adding Combined Years and Months:

Use YEAR_MONTH as the unit. The expr is a string 'years-months'.

```
-- Add 1 year and 2 months
DATE_ADD(NOW(), INTERVAL '1-2' YEAR_MONTH) -- Note: Original text used '1 2', '1-2' is common for
YEAR_MONTH
```

If `NOW()` is `2025-09-15 23:00:00`, this results in `2026-11-15 23:00:00`. This type of interval typically does not affect the day or time components directly, only the year and month.

Subtracting Durations

Using DATE_ADD with a Negative Interval:

You can subtract durations by providing a negative value for expr.

```
-- Subtract 5 days
DATE_ADD(NOW(), INTERVAL -5 DAY)
```

Using DATE_SUB(date, INTERVAL expr unit):

This function is specifically for subtracting durations.

```
-- Subtract 5 days
DATE_SUB(NOW(), INTERVAL 5 DAY)
```

Note: With `DATE_SUB`, `expr` is positive for subtraction. A negative `expr` would result in addition.

This page is licensed: CC BY-SA / Gnu FDL

System & Status Variables Guide

System variables and status variables serve two distinct roles in MariaDB: System variables are the “knobs” you turn to change how the server behaves, while status variables are the “gauges” you watch to see how the server is performing.

Because MariaDB is modular, these variables are often grouped by their specific functional area or plugin.

What you do with System Variables

System variables are configuration settings. You use them to define the environment, set resource limits, and tune performance.

- **Configuration & Tuning:** You modify these to optimize the server for your workload. For example, you might increase `innodb_buffer_pool_size` to allow the database to cache more data in memory, reducing disk I/O.
- **Controlling Behavior:** You use them to toggle features on or off. Setting `read_only` to `1` ensures no data can be modified (useful for maintenance or replicas).
- **Enforcing Constraints:** You set limits to prevent resource exhaustion. For example, `max_connections` prevents the server from being overwhelmed by too many simultaneous users.
- **Session Personalization:** Many variables can be set at the Session level. This means you can change how the server behaves just for your current connection – like changing the `sql_mode` to be more or less strict – without affecting other users.

What you use Status Variables for

Status variables are read-only counters and metrics. You use them to monitor the health and activity of the server.

- **Performance Monitoring:** You check status variables to see if your system tuning is working. If you see high numbers for `Select_full_join`, it tells you that your queries are missing indexes.
- **Health Checks:** You use them to identify bottlenecks. If `Aborted_connects` is high, you might have network issues or a client with the wrong password attempting to connect repeatedly.
- **Resource Tracking:** They tell you how much of your allocated resources are actually being used. Comparing `Max_used_connections` against your system variable `max_connections` helps you decide if you need to scale up.
- **Capacity Planning:** By monitoring variables like `Bytes_sent` and `Bytes_received` over time, you can forecast when you will need to upgrade your hardware or network bandwidth.

Comparison at a Glance

Feature	System Variables	Status Variables
Analogy	Steering wheel / Gas pedal	Speedometer / Fuel gauge
Action	Set (change the value)	View (read the value)
Purpose	Control and configuration	Monitoring and diagnostics
SQL Statement	`SET GLOBAL ...` or `SET SESSION ...`	`SHOW STATUS ...`

Pages Documenting Variables

Here is the comprehensive list of pages that document variables.

Core Server Variables

- Server System Variables (The main list for general server configuration)
- Server Status Variables (Real-time monitoring metrics)
- Performance Schema System Variables

Storage Engine Specific Variables

- Aria System Variables
- Aria Status Variables
- InnoDB System Variables
- MyISAM System Variables
- CONNECT System Variables
- Spider System Variables
- Spider Status Variables

Replication & Performance Variables

- Replication and Binary Log System Variables
- Replication and Binary Log Status Variables
- Thread Pool System and Status Variables

Plugin Specific Variables

- Audit Plugin Options and System Variables
- Audit Plugin Status Variables
- Encryption Plugin System Variables (for instance, File Key Management)

Index / Master List

- Full list of MariaDB options, system and status variables
  Note: This page acts as a directory that links back to the detailed pages above.

This page is licensed: CC BY-SA / Gnu FDL

Making Backups with mariadb-dump Guide

This guide explains how to use the `mariadb-dump` utility to create essential backup (dump) files of your MariaDB data. Learn to effectively back up all databases, specific databases, or individual tables, ensuring your data is protected and can be restored when needed.

About `mariadb-dump`

`mariadb-dump` is a command-line utility included with MariaDB for creating logical backups of your databases. It was previously known as `mysqldump`, which often still works as a symbolic link.

Key Advantages:

- **No Server Shutdown:** Backups can be performed while the MariaDB server is running.
- **SQL Output:** It generates a `.sql` file (a "dump file") containing SQL statements (`CREATE TABLE`, `INSERT`, etc.) necessary to reconstruct the databases and data.

All `mariadb-dump` commands are executed from your system's command-line shell, not within the `mariadb` client.

Backing Up All Databases

To export all databases managed by your MariaDB server:

```
mariadb-dump --user=admin_backup --password --lock-tables --all-databases >
/data/backup/dbs_alldatabases.sql
```

- `--user=admin_backup`: Specifies the MariaDB user performing the backup (this user needs appropriate privileges, typically at least `SELECT` and `LOCK TABLES`).
- `--password`: Prompts for the user's password. For use in scripts where prompting is not possible, you can use `--password=yourpassword` (note the absence of a space and the security implication of having the password in a script or command history).
- `--lock-tables` (or `-x`): Locks all tables across all databases before starting the backup to ensure data consistency. The lock is released once the dump is complete for each table. For transactional tables like InnoDB, using `--single-transaction` is often preferred as it provides a consistent snapshot without prolonged locking of all tables.
- `--all-databases` (or `-A`): Specifies that all databases should be dumped.
- `> /data/backup/dbs_alldatabases.sql`: Redirects the output (the SQL statements) to the specified file. Ensure the path exists and the user running the command has write permissions.

Commonly Used Option for Efficiency:

- `--extended-insert` (or `-e`): Creates `INSERT` statements that include multiple rows per statement. This generally results in a smaller dump file and faster restores. This option is often enabled by default but can be explicitly stated.

Example with long options and password in script:

```
mariadb-dump --user=admin_backup --password=yoursecurepassword --lock-tables --extended-insert --all-
databases > /data/backup/dbs_alldatabases.sql
```

Backing Up a Single Database

Backing up databases individually can result in smaller, more manageable dump files and allow for more flexible backup schedules.

```
mariadb-dump --user=admin_backup --password --lock-tables --extended-insert --databases
your_database_name > /data/backup/your_database_name.sql
```

- `--databases` (or `-B`): Followed by the name of the database to dump.
- To back up multiple specific databases, list their names separated by spaces after the `--databases` option:

```
mariadb-dump --user=admin_backup --password --lock-tables --extended-insert --databases db1_name
db2_name > /data/backup/selected_databases.sql
```

Backing Up Specific Tables

For very large databases, or if only certain tables change frequently, you might back up individual tables.

```
mariadb-dump --user=admin_backup --password --lock-tables --extended-insert your_database_name
table_name1 table_name2 > /data/backup/your_database_name_selected_tables.sql
```

- First, specify the database name (`your_database_name`).
- Then, list one or more table names (`table_name1` , `table_name2`) separated by spaces.
- Note that the `--databases` option is *not* used when dumping specific tables in this manner.

Important Considerations and Best Practices

- **User Privileges:** The MariaDB user specified with `--user` needs at least `SELECT` privileges for the tables being dumped. `LOCK TABLES` privilege is needed if using `--lock-tables` . `RELOAD` or `FLUSH_TABLES` might be needed for options like `--flush-logs` or `--master-data` . For `--single-transaction` , `PROCESS` and `RELOAD` might be required. A user with global `SELECT` , `LOCK TABLES` , `SHOW VIEW` , `EVENT` , and `TRIGGER` privileges is often used for backups.
- **Consistency with InnoDB:** For databases primarily using InnoDB tables, consider using the `--single-transaction` option instead of `--lock-tables` . This option starts a transaction before dumping and reads data from a consistent snapshot without locking the tables for extended periods, allowing concurrent reads and writes.Bash

  ```
  mariadb-dump --user=admin_backup --password --single-transaction --extended-insert --databases
  your_innodb_database > /data/backup/your_innodb_database.sql
  ```
- **Practice Makes Perfect:** `mariadb-dump` is powerful but can have many options. Practice using it on a test database or server to become comfortable with its usage and to verify that your backup strategy works as expected.
- **Test Your Backups:** Regularly test your backup files by restoring them to a non-production environment to ensure they are valid and can be used for recovery.
- **Restoration:** To learn how to restore data from these dump files, see the “Data Restoration Guide”.
- **Security:** Store backup files in a secure location. If passwords are included in scripts, ensure the script files have restricted permissions.

This page is licensed: CC BY-SA / Gnu FDL

A MariaDB Primer Guide

This primer offers a quick jump-start for beginners using an existing MariaDB database via the `mariadb` command-line client. Learn how to log in, understand basic database concepts, and perform essential SQL operations like creating tables, inserting data, and retrieving or modifying records.

Logging into MariaDB

To begin, log into your MariaDB server from your system’s command-line:

```
mariadb -u user_name -p -h ip_address db_name
```

- Replace `user_name` with your MariaDB username.
- Replace `ip_address` with the hostname or IP address of your MariaDB server. If you are accessing MariaDB from the same server you're logged into (locally = `localhost`), you can usually omit the `-h ip_address` part.
- Replace `db_name` with the name of the database you wish to access (for instance, `test`). Some setups may have a `test` database by default; others might not, or it might have been removed (for instance, by `mariadb-secure-installation`). If unsure, or if you want to connect without selecting a specific database initially, you can omit `db_name`.

You are prompted to enter your password. If your login is successful, you see a prompt similar to this:

```
MariaDB [test]>
```

"MariaDB" indicates you are connected to a MariaDB server. The name within the brackets (for instance, `test`) is your current default database. If no database was specified or successfully connected to, it shows `[(none)]`.

Understanding Database Basics and Setup

SQL (Structured Query Language): This is the language used to interact with MariaDB. An SQL statement that requests data is called a query.

Tables: Databases store information in tables, which are structured like spreadsheets with rows and columns, but are much more efficient for data management.

Example Setup:

If the test database is empty or doesn't exist, you can run the following SQL statements to create and populate tables for the examples in this primer. Copy and paste these into the mariadb client prompt.

```
CREATE DATABASE IF NOT EXISTS test;
USE test;

CREATE TABLE IF NOT EXISTS books (
  BookID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  Title VARCHAR(100) NOT NULL,
  SeriesID INT,
  AuthorID INT
);

CREATE TABLE IF NOT EXISTS authors (
  id INT NOT NULL PRIMARY KEY AUTO_INCREMENT
  -- You would typically add more columns like name, etc.
);

CREATE TABLE IF NOT EXISTS series (
  id INT NOT NULL PRIMARY KEY AUTO_INCREMENT
  -- You would typically add more columns like series_name, etc.
);

INSERT INTO books (Title, SeriesID, AuthorID) VALUES
  ('The Fellowship of the Ring', 1, 1),
  ('The Two Towers', 1, 1),
  ('The Return of the King', 1, 1),
  ('The Sum of All Men', 2, 2),
  ('Brotherhood of the Wolf', 2, 2),
  ('Wizardborn', 2, 2),
  ('The Hobbbit', 0, 1); -- Note: "Hobbbit" is intentionally misspelled for a later example
```

- **Semicolons (`;`):** The `mariadb` client allows complex SQL statements over multiple lines. It sends the statement to the server for execution only after you type a semicolon (`;`) and press [Enter].

Exploring Your Database Structure

Listing Tables:

To see the tables in your current database:

```
SHOW TABLES;
```

Output (example):

```
+----------------+
| Tables_in_test |
+----------------+
| authors        |
| books          |
| series         |
+----------------+
3 rows in set (0.00 sec)
```

Describing a Table:

To get information about the columns in a table (like their names and types):

```
DESCRIBE books;
```

Output (example):

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| BookID   | int(11)      | NO   | PRI | NULL    | auto_increment |
| Title    | varchar(100) | NO   |     | NULL    |                |
| SeriesID | int(11)      | YES  |     | NULL    |                |
| AuthorID | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

The `Field` column lists the column names, which you'll need to retrieve specific data.

Retrieving Data (SELECT)

To retrieve data from a table, use the `SELECT` statement.

```
SELECT * FROM books;
```

- The asterisk (`*`) is a wildcard meaning "all columns." **Output (example):**

```
+--------+---------------------------+----------+----------+
| BookID | Title                     | SeriesID | AuthorID |
+--------+---------------------------+----------+----------+
|      1 | The Fellowship of the Ring |        1 |        1 |
|      2 | The Two Towers             |        1 |        1 |
|      3 | The Return of the King     |        1 |        1 |
|      4 | The Sum of All Men         |        2 |        2 |
|      5 | Brotherhood of the Wolf    |        2 |        2 |
|      6 | Wizardborn                 |        2 |        2 |
|      7 | The Hobbbit                |        0 |        1 |
+--------+---------------------------+----------+----------+
7 rows in set (0.00 sec)
```

Adding Data (INSERT)

To add new rows to a table, use the `INSERT` statement.

```
INSERT INTO books (Title, SeriesID, AuthorID)
VALUES ("Lair of Bones", 2, 2);
```

- After `INSERT INTO table_name`, list the columns you are providing data for in parentheses.
- The `VALUES` keyword is followed by a list of values in parentheses, in the same order as the listed columns.
  Output:

```
Query OK, 1 row affected (0.00 sec)
```

You can run `SELECT * FROM books;` again to see the newly added row.

Modifying Data (UPDATE)

To change existing data in a table, use the `UPDATE` statement. Let's correct the spelling of "The Hobbbit".

```
UPDATE books
SET Title = "The Hobbit"
WHERE BookID = 7;
```

- `SET Title = "The Hobbit"` specifies the column to change and its new value.
- `WHERE BookID = 7` is crucial; it specifies *which row(s)* to update. Without a `WHERE` clause, `UPDATE` would change *all* rows in the table. **Output:**

```
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Run `SELECT * FROM books WHERE BookID = 7;` to see the correction.

Using MariaDB involves understanding SQL syntax. It doesn't allow for typing mistakes or clauses in the wrong order, but with practice, it becomes straightforward.

See Also

- MariaDB Basics

This page is licensed: CC BY-SA / Gnu FDL

Creating & Using Views Guide

This guide introduces SQL Views in MariaDB, virtual tables based on the result-set of a stored query. Learn how views simplify complex queries, enhance data security by restricting access, and provide an abstraction layer over your database tables through practical examples.

Prerequisites

- A basic understanding of SQL, particularly `JOIN` operations. (You may want to refer to More Advanced Joins.)
- Access to a MariaDB database.
- Privileges to `CREATE TABLE` and `CREATE VIEW`.

Setup: Example Employee Database

First, we'll create and populate two tables, `Employees` and `Hours`, to use in our examples. If you have already completed a tutorial using this database structure (e.g., from a "More Advanced Joins" guide), you might be able to skip this setup.

Employees Table:

```
CREATE TABLE `Employees` (
  `ID` TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `First_Name` VARCHAR(25) NOT NULL,
  `Last_Name` VARCHAR(25) NOT NULL,
  `Position` VARCHAR(25) NOT NULL,
  `Home_Address` VARCHAR(50) NOT NULL,
  `Home_Phone` VARCHAR(12) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM;

INSERT INTO `Employees` (`First_Name`, `Last_Name`, `Position`, `Home_Address`, `Home_Phone`)
VALUES
  ('Mustapha', 'Mond', 'Chief Executive Officer', '692 Promiscuous Plaza', '326-555-3492'),
  ('Henry', 'Foster', 'Store Manager', '314 Savage Circle', '326-555-3847'),
  ('Bernard', 'Marx', 'Cashier', '1240 Ambient Avenue', '326-555-8456'),
  ('Lenina', 'Crowne', 'Cashier', '281 Bumblepuppy Boulevard', '328-555-2349'),
  ('Fanny', 'Crowne', 'Restocker', '1023 Bokanovsky Lane', '326-555-6329'),
  ('Helmholtz', 'Watson', 'Janitor', '944 Soma Court', '329-555-2478');
```

Hours Table:

```
CREATE TABLE `Hours` (
  `ID` TINYINT(3) UNSIGNED NOT NULL,
  `Clock_In` DATETIME NOT NULL,
  `Clock_Out` DATETIME NOT NULL
) ENGINE=MyISAM;

INSERT INTO `Hours`
VALUES ('1', '2005-08-08 07:00:42', '2005-08-08 17:01:36'),
  ('1', '2005-08-09 07:01:34', '2005-08-09 17:10:11'),
  ('1', '2005-08-10 06:59:56', '2005-08-10 17:09:29'),
  ('1', '2005-08-11 07:00:17', '2005-08-11 17:00:47'),
  ('1', '2005-08-12 07:02:29', '2005-08-12 16:59:12'),
  ('2', '2005-08-08 07:00:25', '2005-08-08 17:03:13'),
  ('2', '2005-08-09 07:00:57', '2005-08-09 17:05:09'),
  ('2', '2005-08-10 06:58:43', '2005-08-10 16:58:24'),
  ('2', '2005-08-11 07:01:58', '2005-08-11 17:00:45'),
  ('2', '2005-08-12 07:02:12', '2005-08-12 16:58:57'),
  ('3', '2005-08-08 07:00:12', '2005-08-08 17:01:32'),
  ('3', '2005-08-09 07:01:10', '2005-08-09 17:00:26'),
  ('3', '2005-08-10 06:59:53', '2005-08-10 17:02:53'),
  ('3', '2005-08-11 07:01:15', '2005-08-11 17:04:23'),
  ('3', '2005-08-12 07:00:51', '2005-08-12 16:57:52'),
  ('4', '2005-08-08 06:54:37', '2005-08-08 17:01:23'),
  ('4', '2005-08-09 06:58:23', '2005-08-09 17:00:54'),
  ('4', '2005-08-10 06:59:14', '2005-08-10 17:00:12'),
  ('4', '2005-08-11 07:00:49', '2005-08-11 17:00:34'),
  ('4', '2005-08-12 07:01:09', '2005-08-12 16:58:29'),
  ('5', '2005-08-08 07:00:04', '2005-08-08 17:01:43'),
  ('5', '2005-08-09 07:02:12', '2005-08-09 17:02:13'),
  ('5', '2005-08-10 06:59:39', '2005-08-10 17:03:37'),
  ('5', '2005-08-11 07:01:26', '2005-08-11 17:00:03'),
  ('5', '2005-08-12 07:02:15', '2005-08-12 16:59:02'),
  ('6', '2005-08-08 07:00:12', '2005-08-08 17:01:02'),
  ('6', '2005-08-09 07:03:44', '2005-08-09 17:00:00'),
  ('6', '2005-08-10 06:54:19', '2005-08-10 17:03:31'),
  ('6', '2005-08-11 07:00:05', '2005-08-11 17:02:57'),
  ('6', '2005-08-12 07:02:07', '2005-08-12 16:58:23');
```

Building a Complex Query (Example: Employee Tardiness)

Let's say Human Resources needs a report on employees who are late (clock in after 7:00:59 AM) and do not make up the time at the end of their shift (work less than 10 hours and 1 minute).

Initial Query (Helmholtz's Lateness):

This query finds instances where Helmholtz was late within a specific week:

```
SELECT
  `Employees`.`First_Name`,
  `Employees`.`Last_Name`,
  `Hours`.`Clock_In`,
  `Hours`.`Clock_Out`
FROM `Employees`
INNER JOIN `Hours` ON `Employees`.`ID` = `Hours`.`ID`
WHERE `Employees`.`First_Name` = 'Helmholtz'
AND DATE_FORMAT(`Hours`.`Clock_In`, '%Y-%m-%d') >= '2005-08-08'
AND DATE_FORMAT(`Hours`.`Clock_In`, '%Y-%m-%d') <= '2005-08-12'
AND DATE_FORMAT(`Hours`.`Clock_In`, '%H:%i:%S') > '07:00:59';
```

Output:

```
+------------+-----------+---------------------+---------------------+
| First_Name | Last_Name | Clock_In            | Clock_Out           |
+------------+-----------+---------------------+---------------------+
| Helmholtz  | Watson    | 2005-08-09 07:03:44 | 2005-08-09 17:00:00 |
| Helmholtz  | Watson    | 2005-08-12 07:02:07 | 2005-08-12 16:58:23 |
+------------+-----------+---------------------+---------------------+
```

Refined Query (Policy Violators):

This query identifies all employees who were late and whose shift duration was less than 10 hours and 1 minute (601 minutes).

```
SELECT
  `Employees`.`First_Name`,
  `Employees`.`Last_Name`,
  `Hours`.`Clock_In`,
  `Hours`.`Clock_Out`,
  (601 - TIMESTAMPDIFF(MINUTE, `Hours`.`Clock_In`, `Hours`.`Clock_Out`)) AS Difference -- Corrected
Difference Calculation
FROM `Employees`
INNER JOIN `Hours` USING (`ID`) -- Simplified JOIN condition
WHERE DATE_FORMAT(`Hours`.`Clock_In`, '%Y-%m-%d') BETWEEN '2005-08-08' AND '2005-08-12'
  AND TIME(`Hours`.`Clock_In`) > '07:00:59'
  AND TIMESTAMPDIFF(MINUTE, `Hours`.`Clock_In`, `Hours`.`Clock_Out`) < 601;
```

Output of Refined Query (example structure):

```
+------------+-----------+---------------------+---------------------+------------+
| First_Name | Last_Name | Clock_In            | Clock_Out           | Difference |
+------------+-----------+---------------------+---------------------+------------+
| Mustapha   | Mond      | 2005-08-12 07:02:29 | 2005-08-12 16:59:12 |          4 |
... (other rows matching the criteria)
+------------+-----------+---------------------+---------------------+------------+
```

Creating and Using a View

The refined query is becoming complex. Storing this query logic in application code makes it harder to manage and means changes to table structures require application code changes. Views can simplify this.

A view is a virtual table based on the result-set of a stored query.

Creating the Employee_Tardiness View:

We use the refined query to create a view. SQL SECURITY INVOKER means the view runs with the permissions of the user querying it.

```
CREATE SQL SECURITY INVOKER VIEW Employee_Tardiness AS
SELECT
  `Employees`.`First_Name`,
  `Employees`.`Last_Name`,
  `Hours`.`Clock_In`,
  `Hours`.`Clock_Out`,
  (601 - TIMESTAMPDIFF(MINUTE, `Hours`.`Clock_In`, `Hours`.`Clock_Out`)) AS Difference
FROM `Employees`
INNER JOIN `Hours` USING (`ID`)
WHERE DATE_FORMAT(`Hours`.`Clock_In`, '%Y-%m-%d') BETWEEN '2005-08-08' AND '2005-08-12'
  AND TIME(`Hours`.`Clock_In`) > '07:00:59'
  AND TIMESTAMPDIFF(MINUTE, `Hours`.`Clock_In`, `Hours`.`Clock_Out`) < 601;
```

Querying the View:

Now, retrieving the tardiness data is much simpler:

```
SELECT * FROM Employee_Tardiness;
```

This will produce the same results as the complex “Refined Query” above.

You can also apply further conditions when querying the view:

```
SELECT * FROM Employee_Tardiness WHERE Difference >= 5;
```

Output (example structure, showing those at least 5 minutes short):

```
+------------+-----------+---------------------+---------------------+------------+
| First_Name | Last_Name | Clock_In            | Clock_Out           | Difference |
+------------+-----------+---------------------+---------------------+------------+
| Mustapha   | Mond      | 2005-08-12 07:02:29 | 2005-08-12 16:59:12 |          5 |
... (other rows where Difference >= 5)
+------------+-----------+---------------------+---------------------+------------+
```

Other Benefits and Uses of Views

- **Simplifying Complex Queries:** As demonstrated, views hide complex joins and calculations.
- **Restricting Data Access (Column-Level Security):** Views can expose only a subset of columns from underlying tables, preventing users or applications from seeing sensitive information (e.g., `Home_Address`, `Home_Phone` were not included in our `Employee_Tardiness` view).
- **Implementing Row-Level Security:** A view can include a `WHERE` clause that filters rows based on the user querying it or other criteria, effectively providing row-level access control. For updatable views, defining them with `WITH CHECK OPTION` (or the similar effect of a `CASCADE` clause mentioned in original text, usually `WITH CASCADED CHECK OPTION`) can ensure that `INSERT`s or `UPDATE`s through the view adhere to the view’s `WHERE` clause conditions.
- **Preemptive Optimization:** Complex, frequently used queries can be defined as views with optimal join strategies and indexing considerations. Other users or applications query the already optimized view, reducing the risk of running inefficient ad-hoc queries.
- **Abstracting Table Structures:** Views provide a consistent interface to applications even if the underlying table structures change (e.g., tables are normalized, split, or merged). The view definition can be updated to map to the new structure, while applications continue to query the unchanged view.

Summary of View Advantages

Views offer a powerful way to:

- **Simplify data access:** Make complex queries easier to write and understand.
- **Abstract database logic:** Separate application code from the complexities of the database schema.

- **Enhance security:** Control access to specific rows and columns.
- **Improve maintainability:** Changes to underlying tables can often be managed by updating the view definition without altering application queries.

See Also

- Views — the full reference for `CREATE VIEW`, `ALTER VIEW`, `DROP VIEW`, and updatable views
- More Advanced Joins

This page is licensed: CC BY-SA / Gnu FDL

Database Applications

> By Anders Karlsson, Principal Sales Engineer at MariaDB Plc — 24 minutes read

This document offers guidance on creating and maintaining database applications with minimal downtime and effort, covering aspects from database schema design to application code.

Here's a summary of key areas and advice covered; find the full guide on the subsequent pages.

- Database Design: A well-designed database is crucial. Standardization in naming conventions (e.g., `orders_t`), data types, character sets (preferably UTF-8 for full Unicode support, `utf8mb4`), and collations is highly recommended to ensure ease of maintenance.
- Data Types:
  - Choosing appropriate types: Consider if a number will be computed; if not, a string might be better (e.g., `VARCHAR` for product codes with leading zeros).
  - Text/String: Use `VARCHAR` and be generous with sizing, as schema upgrades for length extensions are undesirable. UTF-8 (specifically `utf8mb4`) is generally preferred for character sets, and consistency in collations simplifies maintenance.
  - Numeric: Use `BIGINT` for auto-generated primary keys to prevent overflow issues. Avoid `FLOAT` and `DOUBLE` for monetary values due to rounding issues; `DECIMAL` is more accurate.
  - Temporal: Understand the differences between `DATETIME` (stores time as is) and `TIMESTAMP` (affected by client-side time zones) before using them.
  - Other: Be cautious with `ENUM` and `SET` types, as adding values requires schema alteration.
- Schema Objects:
  - Views: Excellent for hiding complexity and supporting different schema versions. Naming views with version strings (`orders_v_1`) can be beneficial. They can also be used to reference data in newer schemas during migration.
- Application Code:
  - Separation of Concerns: While complete separation of application logic and database logic is difficult, practices like using ORMs and stored procedures can help.
  - Object Relational Mappers (ORM): Tools like Hibernate allow applications to be less reliant on specific database schema details, aiding maintenance, though performance and complex operations might still require attention.
  - Stored Procedures and Functions: Isolate database logic from application logic, making maintenance easier by centralizing complex SQL operations in the database layer.
  - Best Practices in Application SQL:
    - Avoid `SELECT *`: Explicitly list columns to prevent issues if table schema changes (column order or addition).
    - Avoid `INSERT` without column names: Always specify column names in `INSERT` statements to avoid errors when table schema changes.

    - Processing of column data: Be consistent; processing in the application can sometimes make SQL more readable.
    - Use of reserved words: Avoid using SQL reserved words for schema objects and column names, even if quoting them makes them valid.
    - Relying on non-explicit assumptions: Never assume row ordering without an `ORDER BY` clause. The order of returned rows is otherwise undetermined. Be cautious with `LIMIT` in `UPDATE` or `DELETE` without `ORDER BY`.
- Code and Schema Standardization: Adhering to internal standards for data types, column names, and database interaction improves maintainability and code readability.
- Complex SQL: Break down very complex SQL (especially `SELECT JOIN`s) into multiple statements or use temporary tables to improve readability and maintainability.
- Canary Testing:
    - Database Naming: Utilize the MariaDB `database` concept (similar to schema) as a namespace to allow different schemas to coexist, avoiding hard-coding database names.
    - Views: Create separate databases for new versions with views referencing data in older or newer schemas to manage transitions.
    - Replication: Use MariaDB replication, particularly statement-based replication (SBR), for canary testing by replicating from the production system to a new server with the updated schema.
    - Invisible Columns: A MariaDB feature allowing columns to exist without being exposed by default, useful in scenarios where old applications use `SELECT *` or `INSERT` without column names, allowing new columns to be added without breaking existing code.

In This Guide

See Introduction & Background.

This section provides an introduction to developing database-backed applications with MariaDB, discussing maintenance, upgrades, and separation of concerns.

See Database Design.

Learn about best practices for database schema design, including naming conventions, choosing appropriate data types, and using views to abstract complexity.

See Application Code.

This guide covers application-side considerations, such as using ORMs, stored procedures, and writing robust SQL that handles schema changes gracefully.

See Canary Testing.

Explore strategies for safely testing schema and application changes using canary deployments, replication, and features like invisible columns.

This page is licensed: CC BY-SA / Gnu FDL

Introduction & Background

Introduction

When designing database-based applications, one aspect that might always be considered at first is how you are supposed to maintain these applications; here, we mean maintenance in the sense of application code and database schemas being upgraded and how this can be done with minimum downtime and effort.

This document aims to look at some important aspects of this, from database schema design to application code. Most of these are not strict rules, but rather aspects to consider when working with applications. The document does not cover general application code aspects, only the aspects that deal with the database part of applications.

Background to Relational Database Applications

Relational database systems, more or less all of them using the SQL query language, have been around since the early 1980s, and things have changed a lot over that time. One aspect that applications using relational databases introduced was the separation of the application, dealing with logic, presentation, user interaction and similar aspects on one hand and the database structure and design on the other.

With relational databases came the relational database modelling and eventually the different normal forms of database design. Much of this has relaxed these days, but there are still things to consider here.

The introduction of logic in the database layer, such as stored procedures, triggers and other aspects, is somewhat blurring the line between code and data, but the general rules still hold.

This page is licensed: CC BY-SA / Gnu FDL

Database Design

Overview

The design of the database is a key here; a well-designed database with appropriate relationships, naming, etc., is still a solid foundation to build a structure on, but as things evolve, it will change with the application. The task is to implement necessary changes, but also to consider future enhancements in the design of the database.

Standardization

To make a database schema easy to maintain, it is best to adhere to some kind of naming standard. What this standard is has less importance than ensuring that it is adhered to. Some like to prefix a name with the type indicator, some like to suffix it, and others ignore this; whichever is your preference, stick to it.

```
CREATE TABLE `orders_t`(`order_id` BIGINT NOT NULL PRIMARY KEY,
   `order_date` DATETIME NOT NULL,
   `customer_id` BIGINT NOT NULL,
   FOREIGN KEY(`customer_id`) REFERENCES `customer_t`(`customer_id`));
```

In addition, sticking to a standard for data types to use, including character sets and collations, is a good practice.

Database Data Types

The data types used really need careful consideration. There are performance aspects here, but in addition, the ease of upgrading the schema should also be considered. Any data type has limitations in and of themselves, and in many cases, limits are introduced as part of a column definition, such as the maximum length of string types and the precision of numeric datatypes.

Choosing an Appropriate Data Type

In some cases, a suitable data type is obvious; in other cases, this is not the case. Take, for example, a product code consisting of 8 digits; is it best stored as an `INTEGER` or as a `VARCHAR(8)`? The former will be more compact in storage and will be faster, the latter less so. Also, are leading zeros significant? If this is the case, then an `INTEGER` is likely not a good option. Actually, in general, if you are not computing a number, it is likely better to be stored as a string.

Text / String Data Types

The most common text data types are variable length, so in general, there is no need to be conservative when sizing a `VARCHAR`, for example. In MariaDB, there is a limit on the total row size of a table, and in this calculation, the maximum size of `VARCHAR` is used, so this might be a reason not to extend this too much. In some cases, you know the maximum length of a `VARCHAR` column, and in that case, it should be used, of course, but be careful, as things might change over time, say some code of some kind might be extended in the future.

In other situations, you cannot really tell what the maximum might be, say a name, then make sure that there is ample space available, having to upgrade a schema just because you have to extend the size of a `VARCHAR` column is unnecessary.

Character Sets and Collations

One issue that needs to be considered is what character set to use for text strings. In most cases, it is recommended that UTF-8 be used, but note that MariaDB has a few options here; either you can use 3-byte UTF-8, called utf8mb3, which allows for 2-byte Unicode or 4-byte, called utf8mb4, which allows full Unicode support. When you use UTF-8, though, remember that storage of strings might be longer. MariaDB ensures that space is allocated as appropriate, but the max potential length of a `VARCHAR(8)` string using utf8mb3 will be 24 bytes, and when it comes to calculating the maximum row size, this is calculated as 24, not 8, bytes.

Collations determine how a string is sorted, for example, when an `ORDER BY` is used and when indexes are created. You really do want to avoid upgrading a schema to change the collation of a column in a table, so be careful here.

Best Practices With Regard to String Data Types

It is recommended that you allow as much space as possible when you don't know the maximum length of a column in a table. It is recommended to use UTF-8 for string data; it does have some drawbacks, but in the end, it is the best general-use character set. It might be useful to use a single-byte character set in some cases, say when using some alphanumeric code item such as a product code, but mostly you are best off with UTF-8 even here, as mixing different data types makes things unnecessarily complicated.

Similar things can be said for collations; there are few reasons to deviate from using just a single collation for a specific application database from the point of view of creating a schema that can be easily maintained at least. From a performance point of view, there are things to consider when determining which collation to use.

Numeric Data Types

Numeric data types come in several flavors, from a high level we are looking at integer types, floating point types and fixed-point types. In MariaDB, all numeric types are fixed-size storage. From the point of view of schema maintenance, the first thing to look for when it comes to creating a database schema that will need less maintenance is to ensure that values fit in the types used. `BIGINT` instead of `INTEGER` is often a good idea, in particular when used for auto-generated primary keys.

Another thing to watch out for is `FLOAT` and `DOUBLE`, including aliases for these, as they are floating-point, which means there might be rounding issues. Using these types for monetary values might not always be a good idea.

An alternative to `FLOAT` and `DOUBLE` is to use the fixed-point `DECIMAL` type, which is exact. Note also that in `SQL_MODE=Oracle` in MariaDB, the Oracle type `NUMERIC` is an alias for `DECIMAL`.

Temporal Data Types

MariaDB has a range of temporal types, from the standard `DATETIME` and `TIMESTAMP` to the more specialised `DATE`, `YEAR` and `TIME`. The by far most used ones are `DATETIME` and `TIMESTAMP`, and there are some things to consider when it comes to maintenance. `DATETIME` and `TIMESTAMP` both store similar values, but there is a difference in that `DATETIME` stores the time as it is, whereas `TIMESTAMP` takes the time zone on the client side into effect.

From a schema maintenance point of view, make sure that you understand how these two types work before using them.

Other Data Types

Most database systems on the market have types that are specific to that system in particular, in MariaDB, which includes `ENUM` and `SET` types. Some types are little used, like BIT and smaller variations of `INTEGER` types. As for `ENUM` and `SET`, these have some useful attributes in that they, on the one hand, require limited storage, but to extend them with additional values, the schema has to be altered, which might be an issue in many cases.

Schema Objects

Beyond this, there are other objects to handle when it comes to schema and application maintenance. Here, we will look at some of them from the point of view of creating a schema that allows itself to be maintained with limited effort.

Views

Using views has several advantages, and although using views might have performance issues, they provide many advantages when it comes to maintaining a database schema. When considering that, this assumes that the views are created with performance and maintenance in mind.

Very complex queries that might need to be maintained can well be put in views. It may also be advantageous to use views to support different versions of a schema over time; adding a version string to VIEW names might well be a good idea.

```
CREATE VIEW orders_v_1
AS
SELECT o.`order_id`, o.`order_date` FROM `orders_t` o;
```

See Also

- Data types
- Character sets and collations
- Views

This page is licensed: CC BY-SA / Gnu FDL

Application Code

Although relational database applications strive to separate application code from database data, this is only possible to a limited extent. That said, there are means to work with database and application design that make maintaining database schema and applications easier. The task at hand then, on the application side of things, is to make the application easy to maintain and as flexible as possible when it comes to integrating with the database on one hand, and on the other hand ensure that the measures taken don't take a toll on features, functionality or performance.

Object Relational Mappers (ORM)

Using an ORM, such as Hibernate for Java applications, makes it possible to build applications that do not rely on the lowest detail of the database schema. On the other hand, performance is sometimes an issue and, in some cases, complex relational operations might still need to be hard-coded. Despite this, hibernate is often a good way to create applications and database schemas that are easy to maintain.

Stored Procedures and Functions

Using database stored procedures is another way of isolating database logic from application logic. One advantage here is that if applications call stored procedures instead of issuing SQL statements, then we can keep a clear differentiator between the database and backend processing and application-level processing. A disadvantage is that the connection between the backend logic and the structure of the data is still there, it is just somewhere else than in the application, but this still makes maintenance easier in many ways.

It is also common not to drive this too far, i.e. complex SQL and code that truly belongs in the backend can be in the database, whereas more basic SQL and simple SELECTs can be kept as they are in the application.

Views

Views are useful to separate complex processing, such as advanced JOIN operations, from the application. In some cases, VIEWs introduce performance issues, but these cases are rare.

Application Code

As for database code / SQL in applications, there are several best practices to stick to.

SELECT *

Avoid `SELECT *` in applications; these have several bad effects, such as assuming in the application what columns are in a table, no more and no less, and in what order. This is not a good idea. In addition, selecting more columns than necessary does affect performance and may cause the optimizer to use a non-optimal path. Note that `SELECT *` makes two assumptions: first that it assumes which columns exist in a table, and secondly, the order in which these columns are defined. Getting any of these wrong can break things unnecessarily, and using this construct in application code makes schema changes more difficult.

INSERT Without Column Names

Just as in the case of `SELECT *`, an `INSERT` without column names, such as this …

```
INSERT INTO `orders_t` VALUES(1, '2025-06-01 12:00:00');
```

… is a bad idea. The proper way to write an SQL statement like this is:

```
INSERT INTO `orders_t`(`order_id`, `order_date`)
  VALUES(1, '2025-06-01 12:00:00');
```

Not doing this may cause errors in the application if the table is changed, such as when columns are added or the order of the columns is changed.

Processing of Column Data

When data is returned from a `SELECT` statement, it is sometimes tempting to put processing of values in the SQL statement, like this:

```
SELECT YEAR(`order_date`) AS `order_year` FROM `orders_t`;
```

This is not incorrect in any way and is perfectly valid, but sometimes it is better to do this processing in the application. The best is to be consistent and the issue with this kind of construct is that it sometimes makes the SQL less readable than necessary.

Use of Reserved Words in the Schema

It is possible to use reserved words in names of schema objects and in the SQL itself, as here:

```
CREATE TABLE `order`(`order_id` INTEGER NOT NULL PRIMARY KEY);
```

And here:

```
SELECT order_id AS `order` FROM orders_t;
```

Both of these are valid constructs, but are not really recommended. The keyword order used above is likely among the most likely reserved words to use in a schema, but there are more. It is best to avoid them completely, even though quoting them makes the schema and SQL syntax valid.

Relying on Nonexplicit Assumptions

This is an issue that sometimes comes into play, where an application assumes data is processed in a particular order or in a particular way. This really should never be done in an application. One of the best examples is the ordering of rows retrieved; one should not even rely on data being returned in the order of a `PRIMARY KEY` or some index. Example:

```
MariaDB> SELECT `order_date` FROM `orders_t`;
+---------------------+
| order_date          |
+---------------------+
| 2024-05-17 18:01:01 |
| 2025-12-12 18:44:08 |
| 2025-12-12 18:44:17 |
| 2025-09-09 14:57:47 |
+---------------------+
4 rows in set (0.000 sec)

MariaDB> ALTER TABLE `orders_t` ADD KEY(`order_date`);
Query OK, 0 rows affected (0.034 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB> SELECT `order_date` FROM `orders_t`;
+---------------------+
| order_date          |
+---------------------+
| 2024-05-17 18:01:01 |
| 2025-09-09 14:57:47 |
| 2025-12-12 18:44:08 |
| 2025-12-12 18:44:17 |
+---------------------+
4 rows in set (0.003 sec)
```

In the example, the order of the rows returned changed after the index was created, as the index can be used to fetch the data, and when the optimizer does that, it processes the index in order, which means the data is returned in the order of the index. Relying on this ordering in the application is not a good idea, though. If you require data to be returned in a particular order, then you have to use an `ORDER BY` clause; if you don't, the row ordering should be treated as being undetermined.

Another example is the `LIMIT` clause. When using `LIMIT` with a `SELECT`, the rows that are returned are undetermined unless an `ORDER BY` clause is provided. When using a `LIMIT` clause with an `UPDATE` or `DELETE` statement, it is even more important to understand that ordering is not fixed unless an `ORDER BY` statement is used. In general, I'd be careful with using the `LIMIT` clause for `UPDATE` and `DELETE` unless there is a good reason to do so.

There are other assumptions than row ordering, but it is one of the most common issues.

Code and Schema Standardization

Following some internal standards is really helpful when it comes to building applications that can be maintained. Standardizing how to determine the data type, the column name and how to interact with the database schema is a good first step in making an application easy to maintain over time. This also helps in making application code easy to read, which is also helpful.

Complex SQL

There are situations where the application-level SQL gets really complex, sometimes too complex, which in turn makes the code hard to maintain. In those cases, it is sometimes useful to break up a very complex SQL `SELECT` into multiple statements. In particular, complex `SELECT JOIN` queries are troublesome in this respect. If a `JOIN` is not a straight equi-join but a more complex one, for example, joining of a part of a database column, say the `YEAR` part of a `DATETIME` field, then things get really complex. An alternative is sometimes to use temporary tables, which are very efficient in MariaDB, instead of a single very complex `SELECT`.

See Also

- Stored procedures and functions
- Views
- SELECT
- INSERT
- Reserved words
- JOIN queries\

This page is licensed: CC BY-SA / Gnu FDL

Canary Testing

For canary testing new versions of an application, there are some tools to work with, but it has to be said that this is hardly ever 100% clear. If we assume that the advice above has been followed to some extent, then the following are some tools to work with.

Database Naming

In a MariaDB instance, there is the concept of a database, which is similar to a schema. A database is a kind of namespace, which means that an object, say a table, in one database may have the same name as an object in some other database in the same instance. Given this, databases are a key to allowing somewhat different schemas to coexist, which in turn means that it is good practice not to hard-code the database name in applications or schema objects, unless this makes sense.

Views

Views are an excellent way of hiding complexities, and a good way of dealing with this is to have a separate database for a new version with `VIEW`s referencing the actual data in some other database.

```
MariaDB> USE prod_v2;
MariaDB> CREATE VIEW `orders_t` AS
  SELECT `order_id`, `order_date` FROM `prod_v1`.`orders_t`;
```

When adding a new version of the schema/application, it is sometimes necessary to migrate the data to the new schema and have a `VIEW` in the “old” schema referencing the new one. A view is not always necessary in this case, but any added columns have to be handled by a trigger or by a sensible default.

```
MariaDB> USE prod_v2;
MariaDB> CREATE TABLE `orders_t`(
  `order_id` INTEGER NOT NULL PRIMARY KEY,
  `order_date` DATETIME NOT NULL,
  `customer_id` INTEGER NOT NULL DEFAULT 0);
MariaDB> INSERT INTO `orders_t` SELECT `order_id`, `order_date`
  FROM `prod_v1`.`orders_t`;

MariaDB> USE prod_v1;
MariaDB> RENAME TABLE `orders_t` TO `orders_t_v1`;
MariaDB> CREATE VIEW `orders_t` AS
  SELECT `order_id`, `order_date` FROM `prod_v2`.`orders_t`;
```

Replication

A good way of dealing with canary testing with MariaDB is to use replication, which works even in cases with schemas that are different, to an extent, if statement-based replication (SBR) is used. Using this, a new version is built on one server and a new schema is installed there. This is then set to replicate from the current production system. This is not likely to work for all cases, but in many cases, this is a useful tool to allow for canary testing.

Invisible Columns

Hidden or invisible columns in MariaDB are a feature that allows a column to exist in a table without being exposed by default, for instance a `SELECT *` or an `INSERT` without column names will not touch this column. If the advice given in the previous sections of the document is followed, then this should not be necessary much, but in some cases, it is still a useful feature.

For example, in the example above with the orders_t table, if it is the case that the current application actually does `INSERT` without specifying the columns to insert into, then a new column, say customer_id, cannot be added. We can then use an invisible column to support this application, including a new version that does use column names in the `INSERT` and also use the customer_id column:

```
MariaDB> USE prod_v1

MariaDB> ALTER TABLE `orders_t`
  ADD `customer_id` INTEGER NOT NULL DEFAULT 0 INVISIBLE;
MariaDB> SELECT * FROM `orders_t`;
+----------+---------------------+
| order_id | order_date          |
+----------+---------------------+
|        1 | 2024-05-17 18:01:01 |
|        2 | 2025-12-12 18:44:08 |
+----------+---------------------+
2 rows in set (0.004 sec)

MariaDB> INSERT INTO `orders_t`
  (`order_id`, `order_date`, `customer_id`) VALUES(3, NOW(), 2);
```

See Also

- Database
- Views
- Replication
- Invisible columns

This page is licensed: CC BY-SA / Gnu FDL

Server Usage

See Basics.

Grasp the basics of using MariaDB Server. This section introduces fundamental concepts, common SQL commands, and essential operations to get you started with your database.

See Backup & Restore.

Learn to back up and restore MariaDB Server databases. This section covers essential strategies and tools to ensure data safety and quick recovery from potential data loss.

See Tables.

Manage tables in MariaDB Server. This section details creating, altering, and dropping tables, along with understanding data types and storage engines for optimal database design.

See Partitioning Tables.

Optimize large tables in MariaDB Server with partitioning. Learn how to divide tables into smaller, manageable parts for improved performance, easier maintenance, and scalability.

See Stored Routines.

Automate tasks in MariaDB Server with stored routines. Learn to create and manage stored procedures and functions for enhanced database efficiency and code reusability.

See Storage Engines.

Understand MariaDB Server's storage engines. Explore the features and use cases of InnoDB, Aria, MyISAM, and other engines to choose the best option for your specific data needs.

See Triggers & Events.

Automate database actions with triggers and events in MariaDB Server. Learn to define automatic responses to data modifications and schedule tasks for efficient database management.

See User-Defined Functions.

Extend MariaDB Server's capabilities with user-defined functions (UDFs). Learn how to create and implement custom functions to perform specialized operations directly within your SQL queries.

See Views.

Learn to use views in MariaDB Server. This section explains how to create virtual tables from query results, simplifying complex queries and enhancing data security and abstraction.

This page is licensed: CC BY-SA / Gnu FDL

Basics

See Basic Queries.

This guide covers the fundamentals of creating database structures, inserting data, and retrieving information using the default MariaDB client.

Note:

The introductory SQL tutorials now live in the Quickstart Guides:

- A MariaDB Primer
- Basic SQL Statements
- Essential Queries
- Basic SQL Debugging
- String Functions

This page is licensed: CC BY-SA / Gnu FDL

Basic Queries

Connecting to MariaDB

MariaDB is a database system, a database server. To interface with the MariaDB server, you can use a client program, or you can write a program or script with one of the popular programming languages (e.g., PHP) using an API (Application Programming Interface) to interface with the MariaDB server. For the purposes of this article, we will focus on using the default client that comes with MariaDB called `mariadb`. With this client, you can either enter queries from the command-line, or you can switch to a terminal, that is to say, monitor mode. To start, we'll use the latter.

From the Linux command-line, you would enter the following to log in as the root user and to enter monitor mode:

```
mariadb -u root -p -h localhost
```

The `-u` option is for specifying the user name. You would replace `root` here if you want to use a different user name. This is the MariaDB user name, not the Linux user name. The password for the MariaDB user `root` will probably be different from the Linux user `root`. Incidentally, it's not a good security practice to use the `root` user unless you have a specific administrative task to perform for which only `root` has the needed privileges.

The `-p` option above instructs the `mariadb` client to prompt you for the password. If the password for the `root` user hasn't been set yet, then the password is blank and you would just hit [Enter] when prompted. The `-h` option is for specifying the host name or the IP address of the server. This would be necessary if the client is running on a different machine than the server. If you've secure-shelled into the server machine, you probably won't need to use the host option. In fact, if you're logged into Linux as `root`, you won't need the user option—the `-p` is all you'll need. Once you've entered the line above along with the password when prompted, you are logged into MariaDB through the client. To exit, type quit or exit and press [Enter].

Creating a Structure

In order to be able to add and to manipulate data, you first have to create a database structure. Creating a database is simple. You would enter something like the following from within the mariadb client:

```
CREATE DATABASE bookstore;

USE bookstore;
```

This very minimal, first SQL statement will create a sub-directory called bookstore on the Linux filesystem in the directory which holds your MariaDB data files. It won't create any data, obviously. It'll just set up a place to add tables, which will in turn hold data. The second SQL statement above will set this new database as the default database. It will remain your default until you change it to a different one or until you log out of MariaDB.

The next step is to begin creating tables. This is only a little more complicated. To create a simple table that will hold basic data on books, we could enter something like the following:

```
CREATE TABLE books (
isbn CHAR(20) PRIMARY KEY,
title VARCHAR(50),
author_id INT,
publisher_id INT,
year_pub CHAR(4),
description TEXT );
```

This SQL statement creates the table books with six fields, or rather columns. The first column (isbn) is an identification number for each row—this name relates to the unique identifier used in the book publishing business. It has a fixed-width character type of 20 characters. It are the primary key column on which data are indexed. The column data type for the book title is a variable width character column of fifty characters at most. The third and fourth columns are used for identification numbers for the author and the publisher. They are integer data types. The fifth column is used for the publication year of each book. The last column is for entering a description of each book. It's a TEXT data type, which means that it's a variable width column and it can hold up to 65535 bytes of data for each row. There are several other data types that may be used for columns, but this gives you a good sampling.

To see how the table we created looks, enter the following SQL statement:

```
DESCRIBE books;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| isbn        | char(20)    | NO   | PRI | NULL    |       |
| title       | varchar(50) | YES  |     | NULL    |       |
| author_id   | int(11)     | YES  |     | NULL    |       |
| publisher_id | int(11)    | YES  |     | NULL    |       |
| year_pub    | char(4)     | YES  |     | NULL    |       |
| description | text        | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

To change the settings of a table, you can use the ALTER TABLE statement. I'll cover that statement in another article. To delete a table completely (including its data), you can use the DROP TABLE statement, followed by the table name. Be careful with this statement since it's not reversible.

The next table we'll create for our examples is the authors table to hold author information. This table will save us from having to enter the author's name and other related data for each book written by each author. It also helps to ensure consistency of data: there's less chance of inadvertent spelling deviations.

```
CREATE TABLE authors
(author_id INT AUTO_INCREMENT PRIMARY KEY,
name_last VARCHAR(50),
name_first VARCHAR(50),
country VARCHAR(50) );
```

We'll join this table to the books table as needed. For instance, we would use it when we want a list of books along with their corresponding authors' names. For a real bookstore's database, both of these tables would probably have more columns. There would also be several more tables. For the examples that follow, these two tables as they are enough.

Minor Items

Before moving on to the next step of adding data to the tables, let me point out a few minor items that I've omitted mentioning. SQL statements end with a semi-colon (or a \G). You can spread an SQL statement over multiple lines. However, it won't be passed to the server by the client until you terminate it with a semi-colon and hit [Enter]. To cancel an SQL statement once you've started typing it, enter `\c` and press [Enter].

As a basic convention, reserved words are printed in all capital letters. This isn't necessary, though. MariaDB is case-insensitive with regards to reserved words. Database and table names, however, are case-sensitive on Linux. This is because they reference the related directories and files on the filesystem. Column names aren't case sensitive since they're not affected by the filesystem, per se. As another convention, we use lower-case letters for structural names (e.g., table names). It's a matter of preference for deciding on names.

Entering Data

The primary method for entering data into a table is to use the INSERT statement. As an example, let's enter some information about an author into the authors table. We'll do that like so:

```
INSERT INTO authors
(name_last, name_first, country)
VALUES('Kafka', 'Franz', 'Czech Republic');
```

This will add the name and country of the author Franz Kafka to the authors table. We don't need to give a value for the author_id since that column was created with the AUTO_INCREMENT option. MariaDB will automatically assign an identification number. You can manually assign one, especially if you want to start the count at a higher number than 1 (e.g., 1000). Since we are not providing data for all of the columns in the table, we have to list the columns for which we are giving data and in the order that the data is given in the set following the VALUES keyword. This means that we could give the data in a different order.

For an actual database, we would probably enter data for many authors. We'll assume that we've done that and move on to entering data for some books. Below is an entry for one of Kafka's books:

```
INSERT INTO books
(title, author_id, isbn, year_pub)
VALUES('The Castle', '1', '0805211063', '1998');
```

This adds a record for Kafka's book, *The Castle*. Notice that we mixed up the order of the columns, but it still works because both sets agree. We indicate that the author is Kafka by giving a value of 1 for the author_id. This is the value that was assigned by MariaDB when we entered the row for Kafka earlier. Let's enter a few more books for Kafka, but by a different method:

```
INSERT INTO books
(title, author_id, isbn, year_pub)
VALUES('The Trial', '1', '0805210407', '1995'),
('The Metamorphosis', '1', '0553213695', '1995'),
('America', '1', '0805210644', '1995');
```

In this example, we've added three books in one statement. This allows us to give the list of column names once. We also give the keyword `VALUES` only once, followed by a separate set of values for each book, each contained in parentheses and separated by commas. This cuts down on typing and speeds up the process. Either method is fine and both have their advantages. To be able to continue with our examples, let's assume that data on thousands of books has been entered. With that behind us, let's look at how to retrieve data from tables.

Retrieving Data

The primary method of retrieving data from tables is to use a SELECT statement. There are many options available with the SELECT statement, but you can start simply. As an example, let's retrieve a list of book titles from the books table:

```
SELECT title
FROM books;
```

This will display all of the rows of books in the table. If the table has thousands of rows, MariaDB will display thousands. To limit the number of rows retrieved, we could add a LIMIT clause to the SELECT statement like so:

```
SELECT title
FROM books
LIMIT 5;
```

This will limit the number of rows displayed to five. To be able to list the author's name for each book along with the title, you will have to join the books table with the authors table. To do this, we can use the JOIN clause like so:

```
SELECT title, name_last
FROM books
JOIN authors USING (author_id);
```

Notice that the primary table from which we're drawing data is given in the `FROM` clause. The table to which we're joining is given in the JOIN clause along with the commonly named column (i.e., author_id) that we're using for the join.

To retrieve the titles of only books written by Kafka based on his name (not the author_id), we would use the `WHERE` clause with the SELECT statement. This would be entered like the following:

```
SELECT title AS 'Kafka Books'
FROM books
JOIN authors USING (author_id)
WHERE name_last = 'Kafka';

+------------------+
| Kafka Books      |
+------------------+
| The Castle       |
| The Trial        |
| The Metamorphosis |
| America          |
+------------------+
```

This statement will list the titles of Kafka books stored in the database. Notice that I've added the `AS` parameter next to the column name title to change the column heading in the results set to Kafka Books. This is known as an alias. Looking at the results here, we can see that the title for one of Kafka's books is incorrect. His book Amerika is spelled above with a "c" in the table instead of a "k". This leads to the next section on changing data.

Changing & Deleting Data

In order to change existing data, a common method is to use the UPDATE statement. When changing data, though, we need to be sure that we change the correct rows. In our example, there could be another book with the title *America* written by a different author. Since the key column isbn has only unique numbers and we know the ISBN number for the book that we want to change, we can use it to specify the row.

```
UPDATE books
SET title = 'Amerika'
WHERE isbn = '0805210644';
```

This will change the value of the title column for the row specified. We could change the value of other columns for the same row by giving the column = value for each, separated by commas.

If we want to delete a row of data, we can use the DELETE statement. For instance, suppose that our fictitious bookstore has decided no longer to carry books by John Grisham. By first running a SELECT statement, we determine the identification number for the author to be 2034. Using this author identification number, we could enter the following:

```
DELETE FROM books
WHERE author_id = '2034';
```

This statement will delete all rows from the table books for the author_id given. To do a clean job of it, we'll have to do the same for the authors table. We would just replace the table name in the statement above; everything else would be the same.

Conclusion

This is a very basic primer for using MariaDB. Hopefully, it gives you the idea of how to get started with MariaDB. Each of the SQL statements mentioned here have several more options and clauses each. We will cover these statements and others in greater detail in other articles. For now, though, you can learn more about them from experimenting and by further reading of the documentation online documentation.

This page is licensed: CC BY-SA / Gnu FDL

Backup & Restore

See Backup and Restore Overview.

Complete MariaDB backup and recovery guide. Complete resource for backup methods, mariabackup usage, scheduling, and restoration for production use.

See Forming a Backup Strategy.

Learn how to design a robust backup strategy tailored to your business needs, balancing recovery time objectives and data retention policies.

See Backup Optimization.

Discover techniques to optimize your backup processes, including multithreading, incremental backups, and leveraging storage snapshots.

See MariaDB Enterprise Backup.

This page details MariaDB Enterprise Backup, an enhanced version of mariadb-backup with enterprise-specific optimizations and support.

See mariadb-backup.

Get an overview of MariaDB Backup. This section introduces the hot physical backup tool, explaining its capabilities for efficient and consistent backups of your MariaDB Server.

See Point-In-Time Recovery (InnoDB Log Archiving).

Perform point-in-time recovery in MariaDB by replaying archived InnoDB write-ahead logs at startup to restore the server to a specific Log Sequence Number (LSN).

See Replication as a Backup Solution.

Explore how to use replication as part of your backup strategy, allowing you to offload backup tasks to a replica server to reduce load on the primary.

See Backup and Restore via dbForge Studio.

Learn how to use dbForge Studio, a GUI tool, to perform backup and restore operations for MariaDB databases visually.

This page is licensed: CC BY-SA / Gnu FDL

Backup and Restore Overview

This article briefly discusses the main ways to backup MariaDB. For detailed descriptions and syntax, see the individual pages. More detail is in the process of being added.

Logical vs Physical Backups

Logical backups consist of the SQL statements necessary to restore the data, such as CREATE DATABASE, CREATE TABLE and INSERT.

Physical backups are performed by copying the individual data files or directories.

The main differences are as follows:

- logical backups are more flexible, as the data can be restored on other hardware configurations, MariaDB versions or even on another DBMS, while physical backups cannot be imported on significantly different hardware, a different DBMS, or potentially even a different MariaDB version.
- logical backups can be performed at the level of database and table, while physical databases are the level of directories and files. In the MyISAM and InnoDB storage engines, each table has an equivalent set of files.
- logical backups are larger in size than the equivalent physical backup.
- logical backups takes more time to both backup and restore than the equivalent physical backup.
- log files and configuration files are not part of a logical backup

Backup Tools

`mariadb-backup`

The mariadb-backup program is a physical online backup tool and a fork of Percona XtraBackup with added support for compression and data-at-rest encryption.

Storage Engines and Backup Types

MariaDB Backup creates a file-level backup of data from the MariaDB Server data directory. This backup includes temporal data, and the encrypted and unencrypted tablespaces of supported storage engines (e.g., InnoDB, MyRocks, Aria).

MariaDB Server implements:

- Full backups, which contain all data in the database.
- Incremental backups, which contain modifications since the last backup.
- Partial backups, which contain a subset of the tables in the database.

Backup support is specific to storage engines. All supported storage engines enable full backup. The InnoDB storage engine additionally supports incremental backup.

Non-Blocking Backups

A feature of MariaDB Backup and MariaDB Server, non-blocking backups minimize workload impact during backups. When MariaDB Backup connects to MariaDB Server, staging operations are initiated to protect data during read.

Non-blocking backup functionality differs from historical backup functionality in the following ways:

- MariaDB Backup includes optimizations to backup staging, including DDL statement tracking, which reduces lock-time during backups.
- MariaDB Backup in MariaDB Community Server 10.4 and later will block writes, log tables, and statistics.
- Older releases used `FLUSH TABLES WITH READ LOCK`, which closed open tables and only allowed tables to be reopened with a read lock during the duration of backups.

Understanding Recovery

Full backups produced using MariaDB Server are not initially point-in-time consistent, and an attempt to restore from a raw full backup will cause InnoDB to crash to protect the data. Incremental backups contain only the changes since the last backup and cannot be used standalone to perform a restore.

To restore from a backup, you first need to prepare the backup for point-in-time consistency using the `--prepare` command:

- Running `--prepare` on a *full backup* synchronizes the tablespaces, ensuring they are point-in-time consistent.
- Running `--prepare` on an *incremental backup* synchronizes the tablespaces and applies the updated data into the previous full backup.
- Running `--prepare` on data used for a *partial restore* requires the `--export` option to create the necessary `.cfg` files.

Restore Requires Empty Data Directory

For MariaDB Backup to safely restore data from full and incremental backups, the data directory must be empty. When MariaDB Backup restores from a backup using `--copy-back` or `--move-back`, it copies or moves the backup files into the MariaDB Server data directory.

Creating the Backup User

When MariaDB Backup performs a backup operation, it connects to the running MariaDB Server to manage locks and backup staging that prevent the Server from writing to a file while being read. It is recommended that a dedicated user be created and authorized to perform backups:

```
CREATE USER 'mariabackup'@'localhost' IDENTIFIED BY 'mbu_passwd';
GRANT RELOAD, PROCESS, LOCK TABLES, BINLOG MONITOR
      ON * TO 'mariabackup'@'localhost';
```

> **Note:**
>
> While MariaDB Backup requires a user for backup operations, no user is required for restore operations since restores occur while MariaDB Server is not running.

`mariadb-dump`

mariadb-dump (previously mysqldump) performs a logical backup. It is the most flexible way to perform a backup and restore, and a good choice when the data size is relatively small.

For large datasets, the backup file can be large, and the restore time lengthy.

`mariadb-dump` dumps the data into SQL format (it can also dump into other formats, such as CSV or XML) which can then easily be imported into another database. The data can be imported into other versions of MariaDB, MySQL, or even another DBMS entirely, assuming there are no version or DBMS-specific statements in the dump.

mariadb-dump dumps triggers along with tables, as these are part of the table definition. However, stored procedures, views, and events are not, and need extra parameters to be recreated explicitly (for example, `--routines` and `--events`). Procedures and functions are however also part of the system tables (for example mysql.proc).

InnoDB Logical Backups

InnoDB uses the buffer pool, which stores data and indexes from its tables in memory. This buffer is very important for performance. If InnoDB data doesn't fit the memory, it is important that the buffer contains the most frequently accessed data. However, last accessed data is candidate for insertion into the buffer pool. If not properly configured,

when a table scan happens, InnoDB may copy the whole contents of a table into the buffer pool. The problem with logical backups is that they always imply full table scans.

An easy way to avoid this is by increasing the value of the innodb_old_blocks_time system variable. It represents the number of milliseconds that must pass before a recently accessed page can be put into the "new" sublist in the buffer pool. Data which is accessed only once should remain in the "old" sublist. This means that they will soon be evicted from the buffer pool. Since during the backup process the "old" sublist is likely to store data that is not useful, one could also consider resizing it by changing the value of the innodb_old_blocks_pct system variable.

It is also possible to explicitly dump the buffer pool on disk before starting a logical backup, and restore it after the process. This will undo any negative change to the buffer pool which happens during the backup. To dump the buffer pool, the innodb_buffer_pool_dump_now system variable can be set to `ON`. To restore it, the innodb_buffer_pool_load_now system variable can be set to `ON`.

`mariadb-dump` Examples

Backing up a single database

```
mariadb-dump db_name > backup-file.sql
```

Restoring or loading the database

```
mariadb db_name < backup-file.sql
```

See the mariadb-dump page for detailed syntax and examples.

`mariadb-hotcopy`

mariadb-hotcopy performs a physical backup, and works only for backing up MyISAM and ARCHIVE tables. It can only be run on the same machine as the location of the database directories.

`mariadb-hotcopy` Examples

```
mariadb-hotcopy db_name [/path/to/new_directory]
mariadb-hotcopy db_name_1 ... db_name_n /path/to/new_directory
```

Percona XtraBackup

Percona XtraBackup is **not supported** in MariaDB. mariadb-backup is the recommended backup method to use instead of Percona XtraBackup. See Percona XtraBackup Overview: Compatibility with MariaDB for more information.

Percona XtraBackup is a tool for performing fast, hot backups. It was designed specifically for XtraDB/InnoDB databases, but can be used with any storage engine (although not with encryption and compression). It is not included with MariaDB.

Filesystem Snapshots

Some filesystems, like Veritas, support snapshots. During the snapshot, the table must be locked. The proper steps to obtain a snapshot are:

- From the mariadb client, execute FLUSH TABLES WITH READ LOCK. The client must remain open.
- From a shell, execute `mount vxfs snapshot`
- The client can execute UNLOCK TABLES.
- Copy the snapshot files.
- From a shell, unmount the snapshot with `umount snapshot`.

LVM

Widely-used physical backup method, using a Perl script as a wrapper.

> **Warning:**
>
> **LVM snapshots are not a standalone DBMS backup solution.**
> LVM operates at the block level, meaning it is "database-blind." It captures a crash-consistent state, identical to a sudden power failure, ignoring data cached in RAM. Without flushing buffers and locking tables, snapshots risk torn pages and permanent corruption.
> Furthermore, the Copy-on-Write (CoW) mechanism significantly degrades production performance. Snapshots also exist on the same physical disks; they are not true backups and offer no protection against hardware failure. Always use application-aware tools (like mariadb-backup) to ensure data integrity.

dbForge Studio for MySQL

Besides the system utilities, it is possible to use third-party GUI tools to perform backup and restore operations. In this context, it is worth mentioning dbForge Studio for MySQL, a feature-rich database IDE that is fully compatible with MariaDB and delivers extensive backup functionality.

The backup and restore module of the Studio allows precise configuration and management of full and partial backups up to particular database objects. The feature of scheduling regular backups offers specific settings to handle errors and keep a log of them. Additionally, settings and configurations can be saved for later reuse.

These operations are wizard-aided allowing users to set up all tasks in a visual mode.

See Also

- Streaming MariaDB backups in the cloud • blog post • 2015 • 5 minutes read

This page is licensed: CC BY-SA / Gnu FDL

Forming a Backup Strategy

Overview

The strategy applied when implementing data backups depends on business needs.

Data backup strategy depends on business needs. Business needs can be evaluated by performing a data inventory, determining data recovery objectives, considering the replication environment, and considering encryption requirements. Also critical is a backup storage strategy and testing backup and recovery procedures.

Data Inventory

Backup strategy requirements flow from the understanding you build by performing a data inventory. A data inventory is established by asking questions such as:

1. What data is housed in the databases?
2. What business purpose does this data serve?
3. How long does the data needed to be retained in order to meet this business purpose?
4. Are there any legal or regulatory requirements, which would limit the length of data retention?

Recovery Objectives

Data recovery requirements are often defined in terms of Recovery Point Objective (RPO) and Recovery Time Objective (RTO). RTO and RPO are considered in the context of the data identified in the data inventory.

Recovery Point Objective (RPO) defines the maximum amount of data a business is willing to lose. For example, a business can define a RPO of 24 hours.

Recovery Time Objective (RTO) defines how quickly a business needs to restore service in the event of a fault. For example, a business can define a RTO of 8 hours.

Backup strategy plays a substantial role in achieving RPO and RTO.

Achieving RPO

RPO depends on completion of backups, which provide a viable recovery point. Since RPO is measured at backup completion, not backup initiation, backup jobs must be scheduled at an interval smaller than the RPO.

Techniques for achieving RPO include:

- Frequent incremental backups and less frequent full backups.
- Performing backups in conjunction with replication and clustering to eliminate impact on production workloads, allowing a higher backup frequency.
- Automated monitoring of backup status.
- Automated testing of backups.

Achieving RTO

The RTO window typically commences at the point when a decision is made by the business to recover from backups, not at the start of an incident.

Techniques for achieving RTO include:

- Leveraging information produced during incident response, which can reduce the set of data to restore from backups, or identify specific data validation requirements dependent on the nature of the incident.
- Having fast access to backup data. Performance requirements of backup infrastructure should be understood for both backup and restoration workloads.
- Using delayed replication, either within the same data center or to a different data center, can provide shorter path to recovery. This is particularly true when coupled with robust application monitoring, which allows intervention before the window of delay elapses.
- Applying written and tested recovery procedures, which designate the systems and commands to be used during recovery.
- Performing drills and exercises that periodically test recovery procedures to confirm readiness.

Replication Considerations

MariaDB Enterprise Server supports several implementations of replication, which accurately duplicates data from one Server to one or more other Servers. The use of a dedicated replica as a source for backups can minimize workload impact.

MariaDB Enterprise Cluster implements virtually synchronous replication, where each Server instance contains a replica of all of the data for the Cluster. Backups can be performed from any node in the Cluster.

Encryption Considerations

MariaDB Enterprise Server supports encryption on disk (data-at-rest encryption) and on the network (data-in-transit encryption).

MariaDB Enterprise Backup copies tablespaces from disk. When data-at-rest encryption is enabled, backups contain encrypted data.

MariaDB Enterprise Backup supports TLS encryption for communications with MariaDB Enterprise Server. To enable TLS encryption, set TLS options from the command-line or in the configuration file:

```
mariadb-backup --backup \
      --target-dir=/data/backups/full \
      --user=mariadb-backup \
      --password=mbu_passwd \
      --ssl-ca=/etc/my.cnf.d/certs/ca.pem \
      --ssl-cert=/etc/my.cnf.d/certs/client-cert.pem \
      --ssl-key=/etc/my.cnf.d/certs/client-key.pem
```

Backup Storage Considerations

How backups are stored can impact backup viability. Backup storage also presents separate risks. These risks need to be carefully considered:

- Backup data should always be stored separately from the system being backed up, and separate from the system used for recovery.
- Backup data should be subject to equal or more controls than data in production databases. For example, backup data should generally be encrypted even where a decision has bee made that a production database will not use data-at-rest encryption.
- Business requirements may define a need for offsite storage of backups as a means of guaranteeing delivery on RPO. In these cases you should also consider onsite storage of backups as a means of guaranteeing delivery on RTO.
- Retention requirements and the run-rate of new data production can aid in capacity planning.

Backup Testing

Testing has been identified as a critical success factor for the successful operation of data systems.

Backups should be tested. Recovery using backups and recovery procedures should be tested.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Backup Optimization

Overview

Backup and restore implementations can help overcome specific technical challenges that would otherwise pose a barrier to meeting business requirements.

Each of these practices represents a trade-off. Understand risks before implementing any of these practices.

Scheduling of Restore Preparation

Technical challenge: restore time

Trade-off: increased ongoing overhead for backup processing

Backup data can be prepared for restore any time after it is produced and before it is used for restore. To expedite recovery, incremental backups can be pre-applied to the prior full backup to enable faster recovery. This may be done at the expense of recovery points, or at the expense of storage by maintaining copies of unmerged full and incremental backup directories.

Moving Restored Data

Technical challenge: disk space limitations

Trade-off: modification of backup directory contents

Suggested method for moving restored data is to use `--copy-back` as this method provides added safety. Where you might instead optimize for disk space savings, system resources, and time you may choose to instead use

MariaDB Enterprise Backup's `--move-back` option. Speed benefits are only present when backup files are on the same disk partition as the destination data directory.

The `--move-back` option will result in the removal of all data files from the backup directory, so it is best to use this option only when you have an additional copy of your backup data in another location.

To restore from a backup by moving files, use the `--move-back` option:

```
mariadb-backup --move-back --target-dir=/data/backups/full
```

Multithreading

Technical challenge:: CPU bottlenecks

Trade-off: Increased workload during backups

MariaDB Enterprise Backup is a multi-threaded application that by default runs on a single thread. In cases where you have a host with multiple cores available, you can specify the number of threads you want it to use for parallel data file transfers using the `--parallel` option:

```
mariadb-backup --backup \
      --target-dir=/data/backups/full \
      --user=mariadb-backup \
      --password=mbu_passwd \
      --parallel=12
```

Incrementing an Incremental Backup

Technical challenge: Backup resource overhead, backup duration

Trade-off: Increased restore complexity, restore process duration

Under normal operation an incremental backup is taken against an existing full backup. This allows you to further shorten the amount of time MariaDB Enterprise Backup locks MariaDB Enterprise Server while copying tablespaces. You can then apply the changes in the increment to the full backup with a `--prepare` operation at leisure, without disrupting database operations.

MariaDB Enterprise Backup also supports incrementing from an incremental backup. In this operation, the `--incremental-basedir` option points not to the full backup directory but rather to the previous incremental backup.

```
mariadb-backup --backup \
      --incremental-basedir=/data/backups/inc1 \
      --target-dir=/data/backups/inc2 \
      --user=mariadb-backup \
      --password=mbu_passwd
```

In preparing a backup to restore the data directory, apply the chain of incremental backups to the full backup in order. That is, first `inc1/, then inc2/`, and so on:

```
mariadb-backup --prepare \
      --target-dir=/data/backups/full \
      --incremental-dir=/data/backups/inc1
```

```
mariadb-backup --prepare \
      --target-dir=/data/backups/full \
      --incremental-dir=/data/backups/inc2
```

Continue to apply all the incremental changes until you have applied all available to the backup. Then restore as usual:

```
mariadb-backup --copy-back --target-dir=/data/backups/full
chown -R mysql:mysql /var/lib/mysql
```

Start MariaDB Enterprise Server on the restored data directory.

Storage Snapshots

Technical challenge: Backup resource overhead, backup duration.

Trade-off: Limited to platforms with volume-level snapshots, may require crash recovery.

While MariaDB Enterprise Backups produces file-level backups, users on storage solutions may prefer to instead perform volume-level snapshots to minimize resource impact. This storage capability exists with some SAN, NAS, and volume manager platforms.

Snapshots occur point-in-time, so no preparation step is needed to ensure data is internally consistent. Snapshots occur while tablespaces are open, and a restored snapshot may need to undergo crash recovery.

Just as traditional full, incremental, and partial backups should be tested, so too should recovery from snapshots be tested on an ongoing basis.

Taking Snapshots

CURRENT

MariaDB Server includes advanced backup functionality to reduce the impact of backup operations:

1. Connect with a client and issue a `BACKUP STAGE START` statement and then a `BACKUP STAGE BLOCK_COMMIT` statement.
2. Take the snapshot.
3. Issue a `BACKUP STAGE END` statement.
4. Once the backup has been completed, remove all files which begin with the `#sql prefix`. These files are generated when `ALTER TABLE` occurs during a staged backup.
5. Retrieve, copy, or store the snapshot as is typical for your storage platform and as per business requirements to make the backup durable. This may require mounting the snapshot in some manner.

< 10.11

It is recommended to briefly prevent writes while snapshotting. Specific commands vary depending on storage platform, business requirements, and setup, but a general approach is to:

1. Connect with a client and issue a `FLUSH TABLES WITH READ LOCK` statement, leaving the client connected.
2. Take the snapshot.
3. Issue an `UNLOCK TABLES` statement, to remove the read lock.
4. Retrieve, copy, or store the snapshot as is typical for your storage platform and as per business requirements to make the backup durable. This may require mounting the snapshot in some manner.

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Backup

Overview

Regular and reliable backups are essential to successful recovery of mission critical applications. MariaDB Enterprise Server backup and restore operations are performed using MariaDB Enterprise Backup, an enterprise-build of MariaDB Backup.

MariaDB Enterprise Backup is compatible with MariaDB Enterprise Server.

Storage Engines and Backup Types

MariaDB Backup creates a file-level backup of data from the MariaDB Community Server data directory. This backup includes temporal data, and the encrypted and unencrypted tablespaces of supported storage engines (e.g., InnoDB, MyRocks, Aria).

MariaDB Enterprise Server implements:

- Full backups, which contain all data in the database.
- Incremental backups, which contain modifications since the last backup.
- Partial backups, which contain a subset of the tables in the database.

Backup support is specific to storage engines. All supported storage engines enable full backup. The InnoDB storage engine additionally supports incremental backup.

Note: MariaDB Enterprise Backup does not support backups of MariaDB ColumnStore. Backup of MariaDB ColumnStore can be performed using MariaDB ColumnStore Tools. Backup of data ingested to MariaDB ColumnStore can also occur pre-ingestion, such as in the case of HTAP where backup could occur of transactional data in MariaDB Enterprise Server, and restore of data to MariaDB ColumnStore would then occur through reprocessing..

Nonblocking Backups

A feature of MariaDB Enterprise Backup and MariaDB Enterprise Server, non-blocking backups minimize workload impact during backups. When MariaDB Enterprise Backup connects to MariaDB Enterprise Server, staging operations are initiated to protect data during read.

Non-blocking backup functionality differs from historical backup functionality in the following ways:

- MariaDB Enterprise Backup in MariaDB Enterprise Server includes enterprise-only optimizations to backup staging, including DDL statement tracking, which reduces lock-time during backups.
- MariaDB Backup in MariaDB Community Server 10.4 and later will block writes, log tables, and statistics.
- Older MariaDB Community Server releases used FLUSH TABLES WITH READ LOCK, which closed open tables and only allowed tables to be reopened with a read lock during the duration of backups.

Understanding Recovery

MariaDB Enterprise Backup creates complete or incremental backups of MariaDB Enterprise Server data, and is also used to restore data from backups produced using MariaDB Enterprise Backup.

Preparing Backups for Recovery

Full backups produced using MariaDB Enterprise Server are not initially point-in-time consistent, and an attempt to restore from a raw full backup will cause InnoDB to crash to protect the data.

Incremental backups produced using MariaDB Enterprise Backup contain only the changes since the last backup and cannot be used standalone to perform a restore.

To restore from a backup, you first need to prepare the backup for point-in-time consistency using the `--prepare` command:

- Running the `--prepare` command on a full backup synchronizes the tablespaces, ensuring that they are point-in-time consistent and ready for use in recovery.
- Running the `--prepare` command on an incremental backup synchronizes the tablespaces and also applies the updated data into the previous full backup, making it a complete backup ready for use in recovery.

- Running the `--prepare` command on data that is to be used for a partial restore (when restoring only one or more selected tables) requires that you also use the `--export` option to create the necessary `.cfg` files to use in recovery.

Restore Requires Empty Data Directory

When MariaDB Enterprise Backup restores from a backup, it copies or moves the backup files into the MariaDB Enterprise Server data directory, as defined by the datadir system variable.

For MariaDB Backup to safely restore data from full and incremental backups, the data directory must be empty. One way to achieve this is to move the data directory aside to a unique directory name:

1. Make sure that the Server is stopped.
2. Move the data directory to a unique name (e.g., `/var/lib/mysql-2020-01-01`) OR remove the old data directory (depending on how much space you have available).
3. Create a new (empty) data directory (e.g., `mkdir /var/lib/mysql)`.
4. Run MariaDB Backup to restore the databases into that directory.
5. Change the ownership of all the restored files to the correct system user (e.g., `chown -R mysql:mysql /var/lib/mysql`).
6. Start MariaDB Enterprise Server, which now uses the restored data directory.
7. When ready, and if you have not already done so, delete the old data directory to free disk space.

Creating the Backup User

When MariaDB Backup performs a backup operation, it not only copies files from the data directory but also connects to the running MariaDB Enterprise Server.

This connection to MariaDB Enterprise Server is used to manage locks that prevent the Server from writing to a file while being read for a backup.

MariaDB Backup establishes this connection based on the user credentials specified with the `--user` and `--password` options when performing a backup.

It is recommended that a dedicated user be created and authorized to perform backups.

> **CURRENT**
>
> MariaDB Backup requires this user to have the `RELOAD, PROCESS, LOCK TABLES,` and `REPLICATION CLIENT` privileges.
>
> ```
> CREATE USER 'mariadb-backup'@'localhost'
> IDENTIFIED BY 'mbu_passwd';
>
> GRANT RELOAD, PROCESS, LOCK TABLES, BINLOG MONITOR
> ON *.*
> TO 'mariadb-backup'@'localhost';
> ```
>
> In the above example, MariaDB Backup would run on the local system that runs MariaDB Enterprise Server. Where backups may be run against a remote server, the user authentication and authorization should be adjusted.
>
> While MariaDB Backup requires a user for backup operations, no user is required for restore operations since restores occur while MariaDB Enterprise Server is not running.
>
> **< 10.5**
>
> MariaDB Backup requires this user to have the `RELOAD, PROCESS, LOCK TABLES,` and `REPLICATION CLIENT` privileges.

```
CREATE USER 'mariadb-backup'@'localhost'
IDENTIFIED BY 'mbu_passwd';

GRANT RELOAD, PROCESS, LOCK TABLES, REPLICATION CLIENT
ON *.*
TO 'mariadb-backup'@'localhost';
```

In the above example, MariaDB Backup would run on the local system that runs MariaDB Enterprise Server. Where backups may be run against a remote server, the user authentication and authorization should be adjusted.

While MariaDB Backup requires a user for backup operations, no user is required for restore operations since restores occur while MariaDB Enterprise Server is not running.

Full Backup and Restore

Full backups performed with MariaDB Backup contain all table data present in the database.

When performing a full backup, MariaDB Backup makes a file-level copy of the MariaDB Enterprise Server data directory. This backup omits log data such as the binary logs (binlog), error logs, general query logs, and slow query logs.

Performing Full Backups

When you perform a full backup, MariaDB Backup writes the backup to the `--target-dir` path. The directory must be empty or non-existent and the operating system user account must have permission to write to that directory. A database user account is required to perform the backup.

The version of `mariadb-backup` or `mariadb-backup` should be the same version as the MariaDB Enterprise Server version. When the version does not match the server version, errors can sometimes occur, or the backup can sometimes be unusable.

To create a backup, execute mariadb-backup or mariadb-backup with the `--backup` option, and provide the database user account credentials using the `--user` and `--password` options:

```
sudo mariadb-backup --backup \
      --target-dir=/data/backups/full \
      --user=mariadb-backup \
      --password=mbu_passwd
```

Subsequent to the above example, the backup is now available in the designated `--target-dir` path.

Preparing a Full Backup for Recovery

A raw full backup is not point-in-time consistent and must be prepared before it can be used for a restore. The backup can be prepared any time after the backup is created and before the backup is restored. However, MariaDB recommends preparing a backup immediately after taking the backup to ensure that the backup is consistent.

The backup should be prepared with the same version of MariaDB Backup that was used to create the backup.

To prepare the backup, execute mariadb-backup or mariadb-backup with the `--prepare` option:

```
sudo mariadb-backup --prepare \
   --use-memory=34359738368 \
   --target-dir=/data/backups/full
```

For best performance, the `--use-memory` option should be set to the server's `innodb_buffer_pool_size` value.

Restoring from Full Backups

Once a full backup has been prepared to be point-in-time consistent, MariaDB Backup is used to copy backup data to the MariaDB Enterprise Server data directory.

To restore from a full backup:

1. Stop the MariaDB Enterprise Server
2. Empty the data directory
3. Restore from the “full” directory using the `--copy-back` option:

```
mariadb-backup --copy-back --target-dir=/data/backups/full
```

MariaDB Backup writes to the data directory as the current user, which can be changed using `sudo`. To confirm that restored files are properly owned by the user that runs MariaDB Enterprise Server, run a command like this (adapted for the correct user/group):

```
chown -R mysql:mysql /var/lib/mysql
```

Once this is done, start MariaDB Enterprise Server:

```
sudo systemctl start mariadb
```

When the Server starts, it works from the restored data directory.

Incremental Backup and Restore

Full backups of large data-sets can be time-consuming and resource-intensive. MariaDB Backup supports the use of incremental backups to minimize this impact.

While full backups are resource-intensive at time of backup, the resource burden around incremental backups occurs when preparing for restore. First, the full backup is prepared for restore, then each incremental backup is applied.

Performing Incremental Backups

When you perform an incremental backup, MariaDB Backup compares a previous full or incremental backup to what it finds on MariaDB Community Server. It then creates a new backup containing the incremental changes.

Incremental backup is supported for InnoDB tables. Tables using other storage engines receive full backups even during incremental backup operations.

To increment a full backup, use the `--incremental-basedir` option to indicate the path to the full backup and the `--target-dir` option to indicate where you want to write the incremental backup:

```
mariadb-backup --backup \
      --incremental-basedir=/data/backups/full \
      --target-dir=/data/backups/inc1 \
      --user=mariadb-backup \
      --password=mbu_passwd
```

In this example, MariaDB Backup reads the `/data/backups/full directory`, and MariaDB Enterprise Server then creates an incremental backup in the `/data/backups/inc1` directory.

Preparing an Incremental Backup

An incremental backup must be applied to a prepared full backup before it can be used in a restore operation. If you have multiple full backups to choose from, pick the nearest full backup prior to the incremental backup that you want

to restore. You may also want to back up your full-backup directory, as it are modified by the updates in the incremental data.

If your full backup directory is not yet prepared, run this to make it consistent:

```
mariadb-backup --prepare --target-dir=/data/backups/full
```

Then, using the prepared full backup, apply the first incremental backup's data to the full backup in an incremental preparation step:

```
mariadb-backup --prepare \
      --target-dir=/data/backups/full \
      --incremental-dir=/data/backups/inc1
```

Once the incremental backup has been applied to the full backup, the full backup directory contains the changes from the incremental backup (that is, the inc1/ directory). Feel free to remove inc1/ to save disk space.

Restoring from Incremental Backups

Once you have prepared the full backup directory with all the incremental changes you need (as described above), stop the MariaDB Community Server, Empty its data directory, and restore from the original full backup directory using the –copy-back option:

```
mariadb-backup --copy-back --target-dir=/data/backups/full
```

MariaDB Backup writes files into the data directory using either the current user or root (in the case of a sudo operation), which may be different from the system user that runs the database. Run the following to recursively update the ownership of the restored files and directories:

```
chown -R mysql:mysql /var/lib/mysql
```

Then, start MariaDB Enterprise Server. When the Server starts, it works from the restored data directory.

Partial Backup and Restore

In a partial backup, MariaDB Backup copies a specified subset of tablespaces from the MariaDB Enterprise Server data directory. Partial backups are useful in establishing a higher frequency of backups on specific data, at the expense of increased recovery complexity. In selecting tablespaces for a partial backup, please consider referential integrity.

Performing a Partial Backup

Command-line options can be used to narrow the set of databases or tables to be included within a backup:

Option	Description
`--databases`	List of databases to include
`--databases-exclude`	List of databases to omit from the backup
`--databases-file`	Path to file listing the databases to include
`--tables`	List of tables to include
`--tables-exclude`	List of tables to exclude
`--tables-file`	Path to file listing the tables to include

For example, you may wish to produce a partial backup, which excludes a specific database:

```
mariadb-backup --backup \
      --target-dir=/data/backups/part \
      --user=mariadb-backup \
      --password=mbu_passwd \
      --database-exclude=test
```

Partial backups can also be incremental:

```
mariadb-backup --backup \
      --incremental-basedir=/data/backups/part \
      --target-dir=/data/backups/part_inc1 \
      --user=mariadb-backup \
      --password=mbu_passwd  \
      --database-exclude=test
```

Preparing a Backup Before a Partial Restore

As with full and incremental backups, partial backups are not point-in-time consistent. A partial backup must be prepared before it can be used for recovery.

A partial restore can be performed from a full backup or partial backup.

The preparation step for either partial or full backup restoration requires the use of transportable tablespaces for InnoDB. As such, each prepare operation requires the –export option:

```
mariadb-backup --prepare --export --target-dir=/data/backups/part
```

When using a partial incremental backup for restore, the incremental data must be applied to its prior partial backup data before its data is complete. If performing partial incremental backups, run the prepare statement again to apply the incremental changes onto the partial backup that served as the base.

```
mariadb-backup --prepare --export \
      --target-dir=/data/backups/part \
      --incremental-dir=/data/backups/part_inc1
```

Performing a Partial Restore

Unlike full and incremental backups, you cannot restore partial backups directly using MariaDB Backup. Further, as a partial backup does not contain a complete data directory, you cannot restore MariaDB Community Server to a startable state solely with a partial backup.

To restore from a partial backup, you need to prepare a table on the MariaDB Community Server, then manually copy the files into the data directory.

The details of the restore procedure depend on the characteristics of the table:

- Partial Restore Non-partitioned Tables
- Partial Restore Partitioned Tables
- Partial Restore of Tables with Full-Text Indexes

As partial restores are performed while the server is running, not stopped, care should be taken to prevent production workloads during restore activity.

Note: You can also use data from a full backup in a partial restore operation if you have prepared the data using the `--export` option as described above.

Partial Restore Nonpartitioned Tables

To restore a non-partitioned table from a backup, first create a new table on MariaDB Community Server to receive the restored data. It should match the specifications of the table you're restoring.

Be extra careful if the backup data is from a server with a different version than the restore server, as some differences (such as a differing `ROW_FORMAT`) can cause an unexpected result.

1. Create an empty table for the data being restored:

```
CREATE TABLE test.address_book (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(255),
   email VARCHAR(255));
```

2. Modify the table to discard the tablespace:

```
ALTER TABLE test.address_book DISCARD TABLESPACE;
```

3. You can copy (or move) the files for the table from the backup to the data directory:

```
# cp /data/backups/part_inc1/test/address_book.* /var/lib/mysql/test
```

4. Use a wildcard to include both the .ibd and .cfg files. Then, change the owner to the system user running MariaDB Community Server:

```
# chown mysql:mysql /var/lib/mysql/test/address_book.*
```

5. Lastly, import the new tablespace:

```
ALTER TABLE test.address_book IMPORT TABLESPACE;
```

MariaDB Community Server looks in the data directory for the tablespace you copied in, then imports it for use. If the table is encrypted, it also looks for the encryption key with the relevant key ID that the table data specifies.

6. Repeat this step for every table you wish to restore.

Partial Restore Partitioned Tables

Restoring a partitioned table from a backup requires a few extra steps compared to restoring a non-partitioned table.

To restore a partitioned table from a backup, first create a new table on MariaDB Community Server to receive the restored data. It should match the specifications of the table you're restoring, including the partition specification.

Be extra careful if the backup data is from a server with a different version than the restore server, as some differences (such as a differing ROW_FORMAT) can cause an unexpected result.

1. Create an empty table for the data being restored:

```
CREATE TABLE test.students (
   id INT PRIMARY KEY AUTO_INCREMENT
   name VARCHAR(255),
   email VARCHAR(255),
   graduating_year YEAR)
PARTITION BY RANGE (graduating_year) (
   PARTITION p9 VALUES LESS THAN 2019
   PARTITION p1 VALUES LESS THAN MAXVALUE
);
```

2. Then create a second empty table matching the column specification, but without partitions. This is your working table:

```
CREATE TABLE test.students_work AS
SELECT * FROM test.students WHERE NULL;
```

3. For each partition you want to restore, discard the working table's tablespace:

```
ALTER TABLE test.students_work DISCARD TABLESPACE;
```

4. Then, copy the table files from the backup, using the new name:

```
# cp /data/backups/part_inc1/test/students.ibd /var/lib/mysql/test/students_work.ibd
# cp /data/backups/part_inc1/test/students.cfg /var/lib/mysql/test/students_work.cfg
```

5. Change the owner to that of the user running MariaDB Community Server:

```
# chown mysql:mysql /var/lib/mysql/test/students_work.*
```

6. Import the copied tablespace:

```
ALTER TABLE test.students_work IMPORT TABLESPACE;
```

7. Lastly, exchange the partition, copying the tablespace from the working table into the partition file for the target table:

```
ALTER TABLE test.students EXCHANGE PARTITION p0 WITH TABLE test.students_work;
```

8. Repeat the above process for each partition until you have them all exchanged into the target table. Then delete the working table, as it's no longer necessary:

```
DROP TABLE test.students_work;
```

This restores a partitioned table.

Partial Restore of Tables with Full-Text Indexes

When restoring a table with a full-text search (FTS) index, InnoDB may throw a schema mismatch error.

In this case, to restore the table, it is recommended to:

- Remove the corresponding .cfg file.
- Restore data to a table without any secondary indexes including FTS.
- Add the necessary secondary indexes to the restored table.

For example, to restore table t1 with FTS index from database db1:

1. In the MariaDB shell, drop the table you are going to restore:

```
DROP TABLE IF EXISTS db1.t1;
```

2. Create an empty table for the data being restored:

```
CREATE TABLE db1.t1(f1 CHAR(10)) ENGINE=INNODB;
```

3. Modify the table to discard the tablespace:

```
ALTER TABLE db1.t1 DISCARD TABLESPACE;
```

4. In the operating system shell, copy the table files from the backup to the data directory of the corresponding database:

```
$ sudo cp /data/backups/part/db1/t1.* /var/lib/mysql/db1
```

5. Remove the .cfg file from the data directory:

```
$ sudo rm /var/lib/mysql/db1/t1.cfg
```

6. Change the owner of the newly copied files to the system user running MariaDB Community Server:

```
$ sudo chown mysql:mysql /var/lib/mysql/db1/t1.*
```

7. In the MariaDB shell, import the copied tablespace:

```
ALTER TABLE db1.t1 IMPORT TABLESPACE;
```

8. Verify that the data has been successfully restored:

```
SELECT * FROM db1.t1;
```

```
+--------+
| f1     |
+--------+
| ABC123 |
+--------+
```

9. Add the necessary secondary indexes:

```
ALTER TABLE db1.t1 FORCE, ADD FULLTEXT INDEX f_idx(f1);
```

10. The table is now fully restored:

```
SHOW CREATE TABLE db1.t1\G
```

```
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `f1` char(10) DEFAULT NULL,
  FULLTEXT KEY `f_idx` (`f1`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Point-in-Time Recoveries

Point-in-time recovery (PITR) is documented here.

This page is: Copyright © 2026 MariaDB. All rights reserved.

mariadb-backup

See mariadb-backup Overview.

An introduction to the `mariadb-backup` utility, detailing its features, installation process, and support for hot online backups of InnoDB tables.

See mariadb-backup Options.

A comprehensive reference for all command-line options available in mariadb-backup, covering backup, prepare, and restore operations.

See Full Backup and Restore (mariadb-backup).

Learn how to perform and restore full physical backups of MariaDB databases using the mariadb-backup tool, ensuring consistent data recovery.

See Incremental Backup and Restore (mariadb-backup).

This guide explains how to create and apply incremental backups with `mariadb-backup`, saving storage space and reducing backup time.

See Point-In-Time Recovery (PITR, mariadb-backup).

Explains how to restore (recover) to a specific point in time. Point-in-time recovery is often referred to as PITR.

See Partial Backup and Restore (mariadb-backup).

Back up specific databases or tables. This guide explains how to filter your backup to include only the data you need.

See Restoring Individual Databases From a Full Backup (mariadb-backup).

Restore a single database from a full backup. Learn the procedure to extract and recover a specific database schema from a larger backup set.

See Restoring Individual Tables and Partitions (mariadb-backup).

Restore specific tables from a backup. Learn the process of importing individual `.ibd` files to recover specific tables without restoring the whole database.

See Setting up a Replica (mariadb-backup).

Initialize a replication slave using a backup. This guide shows how to use mariadb-backup to provision a new replica from a master server.

See Files Backed Up by mariadb-backup.

List of file types included in a backup. Understand which data files, logs, and configuration files are preserved during the backup process.

See Files Created by mariadb-backup.

Reference of files generated during backup. This page explains the purpose of metadata files created by the `mariadb-backup`.

See Using Encryption and Compression Tools With mariadb-backup.

Secure and compress backup streams. Learn to pipe backup output to tools like GPG and GZIP for encryption and storage efficiency.

See How mariadb-backup Works.

Deep dive into backup mechanics. Understand how the tool handles redo logs, locking, and file copying to ensure consistent backups.

See mariadb-backup and BACKUP STAGE.

Understand backup locking stages. This page explains how mariadb-backup uses `BACKUP STAGE` statements to minimize locking during operation.

See mariadb-backup SST Method.

Configure State Snapshot Transfers for Galera Cluster. Learn to use `mariadb-backup` for non-blocking data transfer when a new node joins a cluster.

See Manual SST of Galera Cluster Node With mariadb-backup.

Perform a manual node provision. This guide details the steps to manually backup a donor and restore it to a joiner node in a Galera Cluster.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-backup Overview

> **Note:**
>
> mariadb-backup was previously called mariabackup.

mariadb-backup is an open source tool provided by MariaDB for performing physical online backups of InnoDB, Aria and MyISAM tables. For InnoDB, “hot online” backups are possible. It was originally forked from Percona XtraBackup 2.3.8. It is available on Linux and Windows.

This tool provides a production-quality, nearly non-blocking method for performing full backups on running systems. While partial backups with mariadb-backup are technically possible, they require many steps and cannot be restored directly onto existing servers containing other data.

Supported Features

`mariadb-backup` supports all of the main features of Percona XtraBackup 2.3.8, plus:

- Backup/Restore of tables using Data-at-Rest Encryption.
- Backup/Restore of tables using InnoDB Page Compression.
- mariadb-backup SST method with Galera Cluster.
- Microsoft Windows support.
- Backup/Restore of tables using the MyRocks storage engine. See Files Backed up by mariadb-backup: MyRocks Data Files for more information.

Supported Features in MariaDB Enterprise Backup

> **CURRENT**
>
> MariaDB Backup supports some additional features, such as:

- Minimizes locks during the backup to permit more concurrency and to enable faster backups.
  - This relies on the usage of `BACKUP STAGE` commands and DDL logging.
  - This includes no locking during the copy phase of `ALTER TABLE` statements, which tends to be the longest phase of these statements.
- Provides optimal backup support for all storage engines that store things on local disk.

< 10.11.8

MariaDB Backup does **not** support some additional features.

Backup Types

mariadb-backup supports various type of backups (and restores from those backups), documented on separate pages:

- Full backup and restore
- Incremental backup and restore
- Partial backup and restore
- Restoring individual databases, tables, and partitions
- Point-in-time recovery (PITR)

Installing `mariadb-backup`

Installing on Linux

The `mariadb-backup` executable is included in binary tarballs on Linux.

Installing with a Package Manager

mariadb-backup can also be installed via a package manager on Linux. Many Linux distributions provide MariaDB software "out of the box", including `mariadb-backup`. If your Linux distribution doesn't, however, you can install using a MariaDB repository.

In order to do so, your system needs to be configured to install from one of the MariaDB repositories.

You can configure your package manager to install it from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script.

You can also configure your package manager to install it from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool.

Installing with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using yum or dnf. Starting with RHEL 8 and Fedora 22, `yum` has been replaced by `dnf`, which is the next major version of `yum`. However, `yum` commands still work on many systems that use `dnf`. For example:

```
sudo yum install MariaDB-backup
```

Installing with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant DEB package from MariaDB's repository using apt-get. For example:

```
sudo apt-get install mariadb-backup
```

Installing with zypper

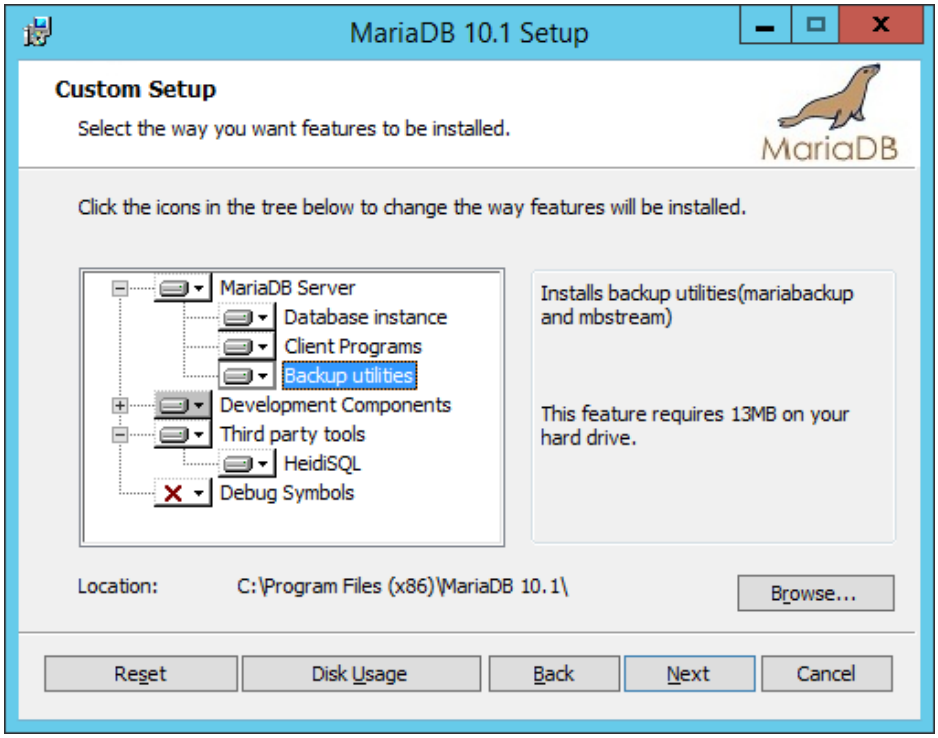
On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using zypper. For example:

```
sudo zypper install MariaDB-backup
```

Installing on Windows

The `mariadb-backup` executable is included in MSI and ZIP packages on Windows.

When using the Windows MSI installer, `mariadb-backup` can be installed by selecting *Backup utilities*:



MariaDB MSI Installer showing the Backup utilities install option

Usage

The command to use `mariadb-backup` and the general syntax is:

```
mariadb-backup <options>
```

For in-depth explanations on how to use `mariadb-backup`, see:

- Full Backup and Restore with mariadb-backup
- Incremental Backup and Restore with mariadb-backup
- Partial Backup and Restore with mariadb-backup
- Restoring Individual Tables and Partitions with mariadb-backup
- Setting up a Replica with mariadb-backup
- Using Encryption and Compression Tools With mariadb-backup

Options

Options supported by mariadb-backup can be found on the mariadb-backup Options page.

Warning:

`mariadb-backup` will currently silently ignore unknown command-line options, so be extra careful about accidentally including typos in options or accidentally using options from later `mariadb-backup` versions. The reason for this is that `mariadb-backup` currently treats command-line options and options from option files equivalently. When it reads from these option files, it has to read a lot of options from the server option groups read by mariadbd. However, `mariadb-backup` does not know about many of the options that it normally reads in these option groups. If `mariadb-backup` raised an error or warning when it encountered an unknown option, then this process would generate a large amount of log messages under normal use. Therefore, `mariadb-backup` is designed to silently ignore the unknown options instead. See MDEV-18215 about that.

Option Files

In addition to reading options from the command-line, mariadb-backup can also read options from option files.

The following options relate to how MariaDB command-line tools handles option files. They must be given as the first argument on the command-line:

Option	Description
`--print-defaults`	Print the program argument list and exit.
`--no-defaults`	Don't read default options from any option file.
`--defaults-file=#`	Only read default options from the given option file.
`--defaults-extra-file=#`	Read this file after the global files are read.
`--defaults-group-suffix=#`	In addition to the default option groups, also read option groups with this suffix.

Server Option Groups

mariadb-backup reads server options from the following option groups from option files:

Group	Description
`[mariadb-backup]`	Options read by `mariadb-backup`.
`[mariadb-backup]`	Options read by `mariadb-backup`.
`[xtrabackup]`	Options read by `mariadb-backup` and Percona XtraBackup.
`[server]`	Options read by MariaDB Server.
`[mysqld]`	Options read by `mariadbd`, which includes both MariaDB Server and MySQL Server (where it is called `mysqld`).
`[mysqld-X.Y]`	Options read by a specific version of mysqld, which includes both MariaDB Server and MySQL Server. For example: `[mysqld-10.6]`.
`[mariadb]`	Options read by MariaDB Server.
`[mariadb-X.Y]`	Options read by a specific version of MariaDB Server. For example: `[mariadb-10.6]`.
`[mariadbd]`	Options read by MariaDB Server. Available from MariaDB 10.5.4.
`[mariadbd-X.Y]`	Options read by a specific version of MariaDB Server. For example: `[mariadbd-10.6]`. Available from MariaDB 10.5.4.

Group	Description
`[client-server]`	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
`[galera]`	Options read by MariaDB Server, but only if it is compiled with Galera Cluster support. All builds on Linux are compiled with Galera Cluster support. When using one of these builds, options from this option group are read even if the Galera Cluster functionality is not enabled.

Client Option Groups

mariadb-backup reads client options from the following option groups from option files:

Group	Description
`[mariadb-backup]`	Options read by mariadb-backup. Available starting with MariaDB 10.1.31 and MariaDB 10.2.13.
`[mariadb-backup]`	Options read by mariadb-backup. Available starting with MariaDB 10.4.14 and MariaDB 10.5.4
`[xtrabackup]`	Options read by mariadb-backup and Percona XtraBackup.
`[client]`	Options read by all MariaDB and MySQL client programs, which includes both MariaDB and MySQL clients. For example, mysqldump.
`[client-server]`	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients. Available starting with MariaDB 10.1.38, MariaDB 10.2.22, and MariaDB 10.3.13.
`[client-mariadb]`	Options read by all MariaDB client programs. Available starting with MariaDB 10.1.38, MariaDB 10.2.22, and MariaDB 10.3.13.

Backup History Table

`mariadb-backup` can optionally track your backup operations in a database table. This provides a centralized audit log and allows you to automate incremental backups by referencing the logical name of the previous backup instead of managing file paths.

Table Location and Schema Changes (MariaDB 10.11):

- **MariaDB 10.11 and later:** The history table is `mysql.mariadb_backup_history` and uses the **InnoDB** storage engine (transactional).
- **MariaDB 10.10 and earlier:** The history table is `PERCONA_SCHEMA.xtrabackup_history` and uses the **CSV** storage engine.

> **Note:**
>
> On the first run after upgrading to MariaDB 10.11, `mariadb-backup` will attempt to migrate the old CSV table to the new InnoDB table. This requires specific privileges (see below).

Authentication and Privileges

`mariadb-backup` needs to authenticate with the database server when it performs a backup operation (i.e. when the –backupoption is specified). For most use cases, the user account that performs the backup needs to have the following global privileges on the database server.

> **CURRENT**
>
> The required privileges are:

```
CREATE USER 'mariadb-backup'@'localhost' IDENTIFIED BY 'mypassword';
GRANT RELOAD, PROCESS, LOCK TABLES, BINLOG MONITOR ON *.* TO 'mariadb-backup'@'localhost';
```

< 10.5

The required privileges are:

```
CREATE USER 'mariadb-backup'@'localhost' IDENTIFIED BY 'mypassword';
GRANT RELOAD, PROCESS, LOCK TABLES, REPLICATION CLIENT ON *.* TO 'mariadb-backup'@'localhost';
```

If your database server is also using the MyRocks storage engine, then the user account that performs the backup will also need the `SUPER` global privilege. This is because `mariadb-backup` creates a checkpoint of this data by setting the rocksdb_create_checkpoint system variable, which requires this privilege. See MDEV-20577 for more information.

`CONNECTION ADMIN` is also required where –kill-long-queries-timeout is greater than `0`, and –no-lock isn't applied in order to KILL queries.

`REPLICA MONITOR` (or alias `SLAVE MONITOR`) is also required where –galera-info or –slave-info is specified.

To use the –history option(or the incremental history options), the backup user requires specific privileges on the history table.

CURRENT

The user needs `INSERT` to create history records and `SELECT` to read them for incremental backups:

```
GRANT SELECT, INSERT, CREATE, ALTER ON mysql.mariadb_backup_history TO 'mariadb-backup'@'localhost';
```

< MARIADB 10.10

The user needs privileges on the legacy `PERCONA_SCHEMA`:

```
GRANT SELECT, INSERT, CREATE, ALTER ON PERCONA_SCHEMA.xtrabackup_history TO 'mariadb-
backup'@'localhost';
```

For Upgrading to 10.11 (One-Time Migration)

If upgrading from an older version, `mariadb-backup` will attempt to migrate the old table to the new location on the first run. The backup user needs privileges to move and modify the old table:

```
GRANT DROP, ALTER, RENAME ON PERCONA_SCHEMA.xtrabackup_history TO 'mariadb-backup'@'localhost';
GRANT CREATE ON PERCONA_SCHEMA TO 'mariadb-backup'@'localhost';
```

Alternatively, you can perform this migration manually before running the backup:

```
RENAME TABLE PERCONA_SCHEMA.xtrabackup_history TO mysql.mariadb_backup_history;
ALTER TABLE mysql.mariadb_backup_history ENGINE=InnoDB;
```

The user account information can be specified with the –user and –password command-line options. For example:

```
mariadb-backup --backup \
   --target-dir=/var/mariadb/backup/ \
   --user=mariadb-backup --password=mypassword
```

The user account information can also be specified in a supported client option group in an option file. For example:

```
[mariadb-backup]
user=mariadb-backup
password=mypassword
```

`mariadb-backup` does not need to authenticate with the database server when preparing or restoring a backup.

File System Permissions

`mariadb-backup` has to read MariaDB's files from the file system. Therefore, when you run `mariadb-backup` as a specific operating system user, you should ensure that user account has sufficient permissions to read those files.

If you are using Linux and if you installed MariaDB with a package manager, then MariaDB's files will probably be owned by the `mysql` user and the `mysql` group.

Using `mariadb-backup` with Data-at-Rest Encryption

`mariadb-backup` supports Data-at-Rest Encryption.

mariadb-backup will query the server to determine which key management and encryption plugin is being used, and then it will load that plugin itself, which means that `mariadb-backup` needs to be able to load the key management and encryption plugin's shared library.

mariadb-backup will also query the server to determine which encryption keys it needs to use.

In other words, `mariadb-backup` is able to figure out a lot of encryption-related information on its own, so normally one doesn't need to provide any extra options to backup or restore encrypted tables.

`mariadb-backup` backs up encrypted and unencrypted tables as they are on the original server. If a table is encrypted, then the table will remain encrypted in the backup. Similarly, if a table is unencrypted, then the table will remain unencrypted in the backup.

The primary reason that mariadb-backup needs to be able to encrypt and decrypt data is that it needs to apply InnoDB redo log records to make the data consistent when the backup is prepared. As a consequence, mariadb-backup does not perform many encryption or decryption operations when the backup is initially taken. MariaDB performs more encryption and decryption operations when the backup is prepared. This means that some encryption-related problems (such as using the wrong encryption keys) may not become apparent until the backup is prepared.

Using `mariadb-backup` for Galera SSTs

The `mariadb-backup` SST method uses the `mariadb-backup` utility for performing SSTs. See mariadb-backup SST method for more information.

Files Backed up by `mariadb-backup`

`mariadb-backup` backs up many different files in order to perform its backup operation. See Files Backed up by mariadb-backup for a list of these files.

Files Created by `mariadb-backup`

`mariadb-backup` creates several different types of files during the backup and prepare phases. See Files Created by mariadb-backup for a list of these files.

Binary Logs

`mariadb-backup` can store the binary log position in the backup. See –binlog-info. This can be used for point-in-time recovery and to use the backup to setup a slave with the correct binlog position.

Known Issues

No Default Data Directory

`mariadb-backup` defaults to the server's default `datadir` value. See MDEV-12956 for more information.

Too Many Open Files

If `mariadb-backup` uses more file descriptors than the system is configured to allow, then users can see errors like the following:

```
2019-02-12 09:48:38 7ffff7fdb820  InnoDB: Operating system error number 23 in a file operation.
InnoDB: Error number 23 means 'Too many open files in system'.
InnoDB: Some operating system error numbers are described at
InnoDB: http://dev.mysql.com/doc/refman/5.6/en/operating-system-error-codes.html
InnoDB: Error: could not open single-table tablespace file ./db1/tab1.ibd
InnoDB: We do not continue the crash recovery, because the table may become
InnoDB: corrupt if we cannot apply the log records in the InnoDB log to it.
InnoDB: To fix the problem and start mysqld:
InnoDB: 1) If there is a permission problem in the file and mysqld cannot
InnoDB: open the file, you should modify the permissions.
InnoDB: 2) If the table is not needed, or you can restore it from a backup,
InnoDB: then you can remove the .ibd file, and InnoDB will do a normal
InnoDB: crash recovery and ignore that table.
InnoDB: 3) If the file system or the disk is broken, and you cannot remove
InnoDB: the .ibd file, you can set innodb_force_recovery > 0 in my.cnf
InnoDB: and force InnoDB to continue crash recovery here.
```

`mariadb-backup` throws an error and aborts when this error is encountered. See MDEV-19060 for more information.

When this error is encountered, one solution is to explicitly specify a value for the –open-files-limit option either on the command line or in one of the supported server option group s in an option file. For example:

```
[mariadb-backup]
open_files_limit=65535
```

An alternative solution is to set the soft and hard limits for the user account that runs `mariadb-backup` by adding new limits to /etc/security/limits.conf. For example, if `mariadb-backup` is run by the `mysql` user, then you could add lines like the following:

```
mysql soft nofile 65535
mysql hard nofile 65535
```

After the system is rebooted, the above configuration should set new open file limits for the `mysql` user, and the user's `ulimit` output should look like the following:

```
ulimit -Sn
65535
ulimit -Hn
65535
```

See Also

- mariadb-dump/mysqldump
- How to backup with MariaDB (video)
- MariaDB point-in-time recovery (video)
- mariadb-backup and Restic (video)

This page is licensed: CC BY-SA / Gnu FDL

mariadb-backup Options

> **Note:**
>
> mariadb-backup was previously called mariabackup.

`mariadb-backup` Options (mariabackup)

Use this page as a reference for **`mariadb-backup` / `mariabackup` command-line options**. It focuses on the options (flags) you use for **physical (file-based) MariaDB backups**, including **hot online backups** for InnoDB.

Quick Reference (Most Searched Options)

- Take a physical backup: `--backup` + `--target-dir`
- Prepare a backup: `--prepare` (or legacy `--apply-log`)
- Restore a backup: `--copy-back` or `--move-back`
- Incremental backups: `--incremental-basedir` + `--incremental-dir`
- Replication/Galera metadata: `--slave-info`, `--binlog-info`, `--galera-info`
- Stream output (pipes to gzip/gpg/etc): `--stream` + `--extra-lsndir`

Common Command Patterns

Full backup (physical):

```
mariadb-backup --backup --target-dir=/backups/full \
  --user=mariadb-backup --password=...
```

Prepare (make files consistent for restore):

```
mariadb-backup --prepare --target-dir=/backups/full
```

Restore:

```
mariadb-backup --copy-back --target-dir=/backups/full
```

Incremental backup (delta against an existing base backup):

```
mariadb-backup --backup --target-dir=/backups/inc1 \
  --incremental-basedir=/backups/full
```

Related Pages

- mariadb-backup Overview
- Full Backup and Restore (mariadb-backup)
- Incremental Backup and Restore (mariadb-backup)
- Using Encryption and Compression Tools With mariadb-backup

Options

`--apply-log`

Prepares an existing backup to restore to the MariaDB Server. This is only valid in `innobackupex` mode, which can be enabled with the –innobackupex option.

Files that `mariadb-backup` generates during –backup operations in the target directory are not ready for use on the Server. Before you can restore the data to MariaDB, you first need to prepare the backup.

In the case of full backups, the files are not point in time consistent, since they were taken at different times. If you try to restore the database without first preparing the data, InnoDB rejects the new data as corrupt. Running `mariadb-backup` with the –prepare command readies the data so you can restore it to MariaDB Server. When working with incremental backups, you need to use the `--prepare` command and the –incremental-dir option to update the base backup with the deltas from an incremental backup.

```
mariadb-backup --innobackupex --apply-log
```

Once the backup is ready, you can use the –copy-back or the –move-back commands to restore the backup to the server.

`--apply-log-only`

If this option is used when preparing a backup, then only the redo log apply stage are performed, and other stages of crash recovery are ignored. This option is used with incremental backups.

> **Important:**
>
> **Note:** This option is not needed or supported anymore.

`--backup`

Backs up your databases.

Using this command option, `mariadb-backup` performs a backup operation on your database or databases. The backups are written to the target directory, as set by the –target-dir option.

```
mariadb-backup --backup
      --target-dir /path/to/backup \
      --user user_name --password user_passwd
```

`mariadb-backup` can perform full and incremental backups. A full backup creates a snapshot of the database in the target directory. An incremental backup checks the database against a previously taken full backup, (defined by the –incremental-basedir option) and creates delta files for these changes.

In order to restore from a backup, you first need to run `mariadb-backup` with the `--prepare` option, to make a full backup point-in-time consistent or to apply incremental backup deltas to base. Then you can run `mariadb-backup` again with either the –copy-back or –move-back commands to restore the database.

For more information, see Full Backup and Restore and Incremental Backup and Restore.

`--binlog-info`

Defines how `mariadb-backup` retrieves the binary log coordinates from the server.

```
--binlog-info[=OFF | ON | LOCKLESS | AUTO]
```

The `--binlog-info` option supports the following retrieval methods. When no retrieval method is provided, it defaults to `AUTO`.

Option	Description
`OFF`	Disables the retrieval of binary log information
`ON`	Enables the retrieval of binary log information, performs locking where available to ensure consistency
`LOCKLESS`	Unsupported option
`AUTO`	Enables the retrieval of binary log information using `ON` or `LOCKLESS` where supported

Using this option, you can control how `mariadb-backup` retrieves the server's binary log coordinates corresponding to the backup.

When enabled, whether using `ON` or `AUTO`, `mariadb-backup` retrieves information from the binlog during the backup process. When disabled with `OFF`, `mariadb-backup` runs without attempting to retrieve binary log information. You may find this useful when you need to copy data without metadata like the binlog or replication coordinates.

```
mariadb-backup --binlog-info --backup
```

Currently, the `LOCKLESS` option depends on features unsupported by MariaDB Server. See the description of the xtrabackup_binlog_pos_innodb file for more information. If you attempt to run `mariadb-backup` with this option, then it causes the utility to exit with an error.

`--close-files`

Defines whether you want to close file handles.

Using this option, you can tell `mariadb-backup` that you want to close file handles. Without this option, `mariadb-backup` keeps files open in order to manage DDL operations. When working with particularly large tablespaces, closing the file can make the backup more manageable. However, it can also lead to inconsistent backups. Use at your own risk.

```
mariadb-backup --close-files --prepare
```

`--compress`

> **Warning:**
>
> This option was deprecated as it relies on the no longer maintained QuickLZ library. It are removed in a future release - versions supporting this function will not be affected. It is recommended to instead backup to a stream (stdout), and use a 3rd party compression library to compress the stream, as described in Using Encryption and Compression Tools With mariadb-backup.

Defines the compression algorithm for backup files.

```
--compress[=compression_algorithm]
```

The `--compress` option only supports the now deprecated `quicklz` algorithm.

Option	Description
`quicklz`	Uses the QuickLZ compression algorithm

```
mariadb-backup --compress --backup
```

If a backup is compressed using this option, then `mariadb-backup` will record that detail in the xtrabackup_info file.

`--compress-chunk-size`

> **Warning:**
>
> Deprecated, for details see the –compress option.

Defines the working buffer size for compression threads.

```
--compress-chunk-size=#
```

`mariadb-backup` can perform compression operations on the backup files before writing them to disk. It can also use multiple threads for parallel data compression during this process. Using this option, you can set the chunk size each thread uses during compression. It defaults to `64K`.

```
mariadb-backup --backup --compress \
     --compress-threads=12 --compress-chunk-size=5M
```

To further configure backup compression, see the –compress and –compress-threads options.

`--compress-threads`

> **Warning:**
>
> Deprecated, for details see the –compress option.

Defines the number of threads to use in compression.

```
--compress-threads=#
```

`mariadb-backup` can perform compression operations on the backup files before writing them to disk. Using this option, you can define the number of threads you want to use for this operation. You may find this useful in speeding up the compression of particularly large databases. It defaults to single-threaded.

```
mariadb-backup --compress --compress-threads=12 --backup
```

To further configure backup compression, see the –compress and –compress-chunk-size options.

`--copy-back`

Restores the backup to the data directory.

Using this command, `mariadb-backup` copies the backup from the target directory to the data directory, as defined by the `--datadir` option. You must stop the MariaDB Server before running this command. The data directory must be empty. If you want to overwrite the data directory with the backup, use the `--force-non-empty-directories` option.

Bear in mind, before you can restore a backup, you first need to run mariadb-backup with the –prepare option. In the case of full backups, this makes the files point-in-time consistent. With incremental backups, this applies the deltas to the base backup. Once the backup is prepared, you can run `--copy-back` to apply it to MariaDB Server.

```
mariadb-backup --copy-back --force-non-empty-directories
```

Running the `--copy-back` command copies the backup files to the data directory. Use this command if you want to save the backup for later. If you don't want to save the backup for later, use the `--move-back` option.

`--core-file`

Defines whether to write a core file.

Using this option, you can configure `mariadb-backup` to dump its core to file in the event that it encounters fatal signals. You may find this useful for review and debugging purposes.

```
mariadb-backup --core-file --backup
```

`--databases`

Defines the databases and tables you want to back up.

```
--databases="database[.table][ database[.table] ...]"
```

Using this option, you can define the specific database or databases you want to back up. In cases where you have a particularly large database or otherwise only want to back up a portion of it, you can optionally also define the tables on the database.

```
mariadb-backup --backup \
      --databases="example.table1 example.table2"
```

In cases where you want to back up most databases on a server or tables on a database, but not all, you can set the specific databases or tables you don't want to back up using the `--databases-exclude` option.

If a backup is a partial backup, then mariadb-backup will record that detail in the `xtrabackup_info` file.

In `innobackupex` mode, which can be enabled with the `--innobackupex` option, the `--databases` option can be used as described above, or it can be used to refer to a file, just as the –databases-file option can in the normal mode.

`--databases-exclude`

Defines the databases you don't want to back up.

```
--databases-exclude="database[.table][ database[.table] ...]"
```

Using this option, you can define the specific database or databases you want to exclude from the backup process. You may find it useful when you want to back up most databases on the server or tables on a database, but would like to exclude a few from the process.

```
mariadb-backup --backup \
      --databases="example" \
      --databases-exclude="example.table1 example.table2"
```

To include databases in the backup, see the `--databases` option.

If a backup is a partial backup, then `mariadb-backup` records that detail in the `xtrabackup_info` file.

`--databases-file`

Defines the path to a file listing databases and/or tables you want to back up.

```
--databases-file="/path/to/database-file"
```

Format the databases file to list one element per line, with the following syntax:

```
database[.table]
```

In cases where you need to back up a number of databases or specific tables in a database, you may find the syntax for the `--databases` and `--databases-exclude` options a little cumbersome. Using this option you can set the path to a file listing the databases or databases and tables you want to back up.

For instance, listing the databases and tables for a backup in a file called `main-backup`:

```
cat main-backup
```

```
example1
example2.table1
example2.table2
```

```
mariadb-backup --backup --databases-file=main-backup
```

If a backup is a partial backup, `mariadb-backup` records that detail in the `xtrabackup_info` file.

`-h, --datadir`

Defines the path to the database root.

```
--datadir=PATH
```

Using this option, you can define the path to the source directory. This is the directory that `mariadb-backup` reads for the data it backs up. It should be the same as the MariaDB Server `datadir` system variable.

```
mariadb-backup --backup -h /var/lib64/mysql
```

`--debug-sleep-before-unlock`

This is a debug-only option used by the Xtrabackup test suite.

`--decompress`

Deprecated, for details see the `--compress` option.

This option requires that you have the `qpress` utility installed on your system.

Defines whether you want to decompress previously compressed backup files.

When you run mariadb-backup with the `--compress` option, it compresses the subsequent backup files, using the QuickLZ algorithm. Using this option, `mariadb-backup` decompresses the compressed files from a previous backup.

For instance, run a backup with compression:

```
mariadb-backup --compress --backup
```

Then, decompress the backup:

```
mariadb-backup --decompress
```

You can enable the decryption of multiple files at a time using the `--parallel` option. By default, mariadb-backup does not remove the compressed files from the target directory. To delete these files, use the `--remove-original` option.

`--debug-sync`

Defines the debug sync point. This option is only used by the `mariadb-backup` test suite.

`--defaults-extra-file`

Defines the path to an extra default option file.

```
--defaults-extra-file=/path/to/config
```

Using this option, you can define an extra default option file for `mariadb-backup`. Unlike `--defaults-file`, this file is read after the default option files are read, allowing you to only overwrite the existing defaults.

```
mariadb-backup --backup \
      --defaults-file-extra=addition-config.cnf \
      --defaults-file=config.cnf
```

`--defaults-file`

Defines the path to the default option file.

```
--defaults-file=/path/to/config
```

Using this option, you can define a default option file for `mariadb-backup`. Unlike the `--defaults-extra-file` option, when this option is provided, it completely replaces all default option files.

```
mariadb-backup --backup \
     --defaults-file=config.cnf
```

`--defaults-group`

Defines the option group to read in the option file.

```
--defaults-group="name"
```

In situations where you find yourself using certain `mariadb-backup` options consistently every time you call it, you can set the options in an option file. The `--defaults-group` option defines what option group `mariadb-backup` reads for its options.

Options you define from the command-line can be set in the configuration file using minor formatting changes. For instance, if you find yourself perform compression operations frequently, you might set `--compress-threads` and `--compress-chunk-size` options in this way:

```
[mariadb-backup]
compress_threads = 12
compress_chunk_size = 64K
```

Now whenever you run a backup with the `--compress` option, it always performs the compression using 12 threads and 64K chunks.

```
mariadb-backup --compress --backup
```

See mariadb-backup Overview: Server Option Groups and mariadb-backup Overview: Client Option Groups for a list of the option groups read by `mariadb-backup` by default.

`--encrypted-backup`

When this option is used with `--backup`, if `mariadb-backup` encounters a page that has a non-zero `key_version` value, then mariadb-backup assumes that the page is encrypted.

Use `--skip-encrypted-backup` instead to allow mariadb-backup to copy unencrypted tables that were originally created before MySQL 5.1.48.

`--export`

If this option is provided during the `--prepare` stage, then it tells `mariadb-backup` to create `.cfg` files for each InnoDB file-per-table tablespace. These `.cfg` files are used to import transportable tablespaces in the process of restoring partial backups and restoring individual tables and partitions.

The `--export` option could require rolling back incomplete transactions that had modified the table. This will likely create a “new branch of history” that does not correspond to the server that had been backed up, which makes it impossible to apply another incremental backup on top of such additional changes. The option should only be applied when doing a `--prepare` of the last incremental.

```
mariadb-backup --prepare --export
```

`mariadb-backup` did not support the `--export` option. See MDEV-13466 about that. In earlier versions of MariaDB, this means that mariadb-backup could not create `.cfg` files for InnoDB file-per-table tablespaces during the `--prepare` stage. You can still import file-per-table tablespaces without the `.cfg` files in many cases, so it may still be possible in those versions to restore partial backups or to restore individual tables and partitions with just the `.ibd` files. If you have a full backup and you need to create `.cfg` files for InnoDB file-per-table tablespaces, then you can do so by preparing the backup as usual without the `--export` option, and then restoring the backup, and then starting the server. At that point, you can use the server’s built-in features to copy the transportable tablespaces.

`--extra-lsndir`

Saves an extra copy of the `xtrabackup_checkpoints` and `xtrabackup_info` files into the given directory.

```
--extra-lsndir=PATH
```

When using the `--backup` option, `mariadb-backup` produces a number of backup files in the target directory. Using this option, you can have `mariadb-backup` produce additional copies of the `xtrabackup_checkpoints` and `xtrabackup_info` files in the given directory.

```
mariadb-backup --extra-lsndir=extras/ --backup
```

This is especially useful when using `--stream` for streaming output, e.g. for compression and/or encryption using external tools in combination with incremental backups, as the `xtrabackup_checkpoints` file necessary to determine the LSN to continue the incremental backup from is still accessible without uncompressing / decrypting the backup file first. Pass in the `--extra-lsndir` of the previous backup as `--incremental-basedir` .

`--force-non-empty-directories`

Allows `--copy-back` or `--move-back` options to use non-empty target directories.

When using `mariadb-backup` with the `--copy-back` or `--move-back` options, they normally require a non-empty target directory to avoid conflicts. Using this option with either of command allows `mariadb-backup` to use a non-empty directory.

```
mariadb-backup --force-non-empty-directories --copy-back
```

Bear in mind that this option does not enable overwrites. When copying or moving files into the target directory, if `mariadb-backup` finds that the target file already exists, it fails with an error.

`--ftwrl-wait-query-type`

Defines the type of query allowed to complete before `mariadb-backup` issues the global lock.

```
--ftwrl-wait-query-type=[ALL | UPDATE | SELECT]
```

The `--ftwrl-wait-query-type` option supports the following query types. The default value is `ALL` .

Option	Description
ALL	Waits until all queries complete before issuing the global lock
SELECT	Waits until SELECT statements complete before issuing the global lock
UPDATE	Waits until UPDATE statements complete before issuing the global lock

When `mariadb-backup` runs, it issues a global lock to prevent data from changing during the backup process. When it encounters a statement in the process of executing, it waits until the statement is finished before issuing the global lock. Using this option, you can modify this default behavior to ensure that it waits only for certain query types, such as for `SELECT` and `UPDATE` statements.

```
mariadb-backup --backup  \
      --ftwrl-wait-query-type=UPDATE
```

`--ftwrl-wait-threshold`

Defines the minimum threshold for identifying long-running queries for FTWRL.

```
--ftwrl-wait-threshold=#
```

When `mariadb-backup` runs, it issues a global lock to prevent data from changing during the backup process and ensure a consistent record. If it encounters statements still in the process of executing, it waits until they complete before setting the lock. Using this option, you can set the threshold at which mariadb-backup engages FTWRL[3]. When it `--ftwrl-wait-timeout` is not 0 and a statement has run for at least the amount of time given this argument, mariadb-backup waits until the statement completes or until the `--ftwrl-wait-timeout` expires before setting the global lock and starting the backup.

```
mariadb-backup --backup \
     --ftwrl-wait-timeout=90 \
     --ftwrl-wait-threshold=30
```

`--ftwrl-wait-timeout`

Defines the timeout to wait for queries before trying to acquire the global lock. The global lock refers to `BACKUP STAGE BLOCK_COMMIT`. The global lock refers to `FLUSH TABLES WITH READ LOCK` (FTWRL).

```
--ftwrl-wait-timeout=#
```

When `mariadb-backup` runs, it acquires a global lock to prevent data from changing during the backup process and ensure a consistent record. If it encounters statements still in the process of executing, it can be configured to wait until the statements complete before trying to acquire the global lock.

If the `--ftwrl-wait-timeout` is set to 0, `mariadb-backup` tries to acquire the global lock immediately without waiting. This is the default value.

If the `--ftwrl-wait-timeout` is set to a non-zero value, then `mariadb-backup` waits for the configured number of seconds until trying to acquire the global lock.

`mariadb-backup` exits if it can't acquire the global lock after waiting for the configured number of seconds.

```
mariadb-backup --backup \
      --ftwrl-wait-query-type=UPDATE \
      --ftwrl-wait-timeout=5
```

The `--ftwrl-wait-timeout` option specifies the maximum time that `mariadb-backup` will wait to obtain the global lock required to begin a consistent backup.

> **FROM MARIADB 10.4**
>
> this lock is acquired with `BACKUP STAGE BLOCK_COMMIT`.
>
> **BEFORE 10.4**
>
> this lock is acquired with `FLUSH TABLES WITH READ LOCK (FTWRL)`.

If the lock cannot be obtained within the configured timeout, the backup process fails.

This option helps avoid failures caused by long-running MariaDB queries that block backup locks.

Example Errors

When the timeout is not set appropriately, backups may fail with messages such as:

```
Unable to obtain lock. Please try again later.
```

or

```
FATAL ERROR: failed to execute query BACKUP STAGE START:
Lock wait timeout exceeded; try restarting transaction
```

Example log excerpt:

```
[00] 2022-02-08 15:43:25 Unable to obtain lock. Please try again later.
[00] 2022-02-08 15:43:25 Error on BACKUP STAGE START query execution
mariabackup: Stopping log copying thread.
```

Originally, `mariadb-backup` could wait indefinitely for the lock. Starting with the fix for MDEV-20230:

- The `--ftwrl-wait-timeout` option also ensures mariadb backup exits gracefully if the lock cannot be obtained within the timeout period.
- This prevents backups from hanging when lock acquisition is blocked by long-running queries.

When to Use

Use `--ftwrl-wait-timeout` when:

- Your workload includes long-running queries (for example, `ALTER TABLE` or large `INSERT` batches).
- Backups sometimes fail with lock wait timeout errors.
- You want `mariadb-backup` to either wait longer for the lock or exit cleanly if it cannot be obtained.

`--galera-info`

Defines whether you want to back up information about a Galera Cluster node's state.

When this option is used, mariadb-backup creates an additional file called `xtrabackup_galera_info`, which records information about a Galera Cluster node's state. It records the values of the wsrep_local_state_uuid and wsrep_last_committed status variables.

You should only use this option when backing up a Galera Cluster node. If the server is not a Galera Cluster node, then this option has no effect.

This option, when enabled and used with GTID replication, will rotate the binary logs at backup time.

```
mariadb-backup --backup --galera-info
```

`--history`

CURRENT

Defines whether you want to track backup history in the `mysql.mariadb_backup_history` table.

```
--history[=name]
```

When using this option, `mariadb-backup` records its operation in a table on the MariaDB Server. Passing a name to this option allows you group backups under arbitrary terms for later processing and analysis.

```
mariadb-backup --backup --history=backup_all
```

Information is written to `mysql.mariadb_backup_history`.

`mariadb-backup` also records this in the mariadb_backup_info file.

< 10.11

Defines whether you want to track backup history in the `PERCONA_SCHEMA.xtrabackup_history` table.

```
--history[=name]
```

When using this option, `mariadb-backup` records its operation in a table on the MariaDB Server. Passing a name to this option allows you group backups under arbitrary terms for later processing and analysis.

```
mariadb-backup --backup --history=backup_all
```

Information is written to `PERCONA_SCHEMA.xtrabackup_history`.

`mariadb-backup` also records this in the xtrabackup_info file.

`-H, --host`

Defines the hostname for the MariaDB Server you want to back up.

```
--host=name_or_ip-address
```

This option defines the hostname or IP address to use when **connecting to a local MariaDB Server over TCP/IP**. By default, `mariadb-backup` attempts to connect to `localhost`.

> **Warning:**
>
> **The mariadb-backup client cannot create backups from a remote server.** Therefore, this option does not allow you to back up a remote server. `mariadb-backup` must always be run on the same server where the database files reside. The `--host` option is used only to establish the client connection for managing locks and retrieving metadata. The actual data files are always read from the local filesystem. Attempting to use this option to back up a remote host results in a backup of the local machine's data, associated with the remote machine's binary log coordinates.

```
mariadb-backup --backup \
      --host="192.168.0.33"
```

`--include`

This option is a regular expression to be matched against table names in *`databasename.tablename`* format. It is equivalent to the `--tables` option. This is only valid in `innobackupex` mode, which can be enabled with the `--innobackupex` option.

`--incremental`

Defines whether you want to take an increment backup, based on another backup. This is only valid in `innobackupex` mode, which can be enabled with the `--innobackupex` option.

```
mariadb-backup --innobackupex --incremental
```

Using this option with the `--backup` option makes the operation incremental rather than a complete overwrite. When this option is specified, either the `--incremental-lsn` or `--incremental-basedir` options can also be given. If neither option is given, `--incremental-basedir` is used by default, set to the first timestamped backup directory in the backup base directory.

```
mariadb-backup --innobackupex --backup --incremental \
     --incremental-basedir=/data/backups \
     --target-dir=/data/backups
```

If a backup is a incremental backup, then `mariadb-backup` records that detail in the `xtrabackup_info` file.

`--incremental-basedir`

Defines whether you want to take an incremental backup, based on another backup.

```
--incremental-basedir=PATH
```

Using this option with the `--backup` option makes the operation incremental rather than a complete overwrite. `mariadb-backup` only copies pages from `.ibd` files if they are newer than the backup in the specified directory.

```
mariadb-backup --backup \
     --incremental-basedir=/data/backups \
     --target-dir=/data/backups
```

If a backup is a incremental backup, then `mariadb-backup` records that detail in the `xtrabackup_info` file.

`--incremental-dir`

Defines whether you want to take an incremental backup, based on another backup.

```
--increment-dir=PATH
```

Using this option with `--prepare` command option makes the operation incremental rather than a complete overwrite. mariadb-backup will apply `.delta` files and log files into the target directory.

```
mariadb-backup --prepare \
      --increment-dir=backups/
```

If a backup is a incremental backup, then `mariadb-backup` records that detail in the `xtrabackup_info` file.

`--incremental-force-scan`

Defines whether you want to force a full scan for incremental backups.

When using `mariadb-backup` to perform an incremental backup, this option forces it to also perform a full scan of the data pages being backed up, even when there's bitmap data on the changes. MariaDB does not support changed page bitmaps, so this option is useless in those versions. See MDEV-18985 for more information.

```
mariadb-backup --backup \
     --incremental-basedir=/path/to/target \
     --incremental-force-scan
```

`--incremental-history-name`

Defines a logical name for the backup.

```
--incremental-history-name=name
```

`mariadb-backup` can store data about its operations on the MariaDB Server. Using this option, you can define the logical name it uses in identifying the backup.

```
mariadb-backup --backup \
     --incremental-history-name=morning_backup
```

The table it uses by default is named `mysql.mariadb_backup_history`. Prior to MariaDB 10.11, the default table was `PERCONA_SCHEMA.xtrabackup_history`.

`mariadb-backup` also records this in the `xtrabackup_info` file.

`--incremental-history-uuid`

Defines a UUID for the backup.

```
--incremental-history-uuid=name
```

`mariadb-backup` can store data about its operations on the MariaDB Server. Using this option, you can define the UUID it uses in identifying a previous backup to increment from. It checks `--incremental-history-name`, `--incremental-basedir`, and `--incremental-lsn`. If mariadb-backup fails to find a valid lsn, it generates an error.

```
mariadb-backup --backup \
      --incremental-history-uuid=main-backup012345678
```

The table it uses is named `PERCONA_SCHEMA.xtrabackup_history`, but expect that name to change in future releases. See MDEV-19246 for more information.

Table Name and Schema Changes (MariaDB 10.11):

- MariaDB 10.11 and later: Uses mysql.mariadb_backup_history (InnoDB).
- MariaDB 10.10 and earlier: Uses PERCONA_SCHEMA.xtrabackup_history (CSV).

`mariadb-backup` also records this in the `xtrabackup_info` file.

`--incremental-lsn`

Defines the sequence number for incremental backups.

```
--incremental-lsn=name
```

Using this option, you can define the sequence number (LSN) value for `--backup` operations. During backups, `mariadb-backup` only copies `.ibd` pages newer than the specified values.

> **Warning:**
>
> Incorrect LSN values can make the backup unusable. It is impossible to diagnose this issue.

`--innobackupex`

> **Note:**
>
> Deprecated option.

Use to enable `innobackupex` mode, which is a compatibility mode.

```
mariadb-backup --innobackupex
```

`--innodb`

This option has no effect. Set only for MySQL option compatibility.

`--innodb-adaptive-hash-index`

Enables InnoDB Adaptive Hash Index.

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option you can explicitly enable the InnoDB Adaptive Hash Index. This feature is enabled by default for mariadb-backup. If you want to disable it, use `--skip-innodb-adaptive-hash-index`.

```
mariadb-backup --backup \
      --innodb-adaptive-hash-index
```

`--innodb-autoextend-increment`

Defines the increment in megabytes for auto-extending the size of tablespace file.

```
--innodb-autoextend-increment=36
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can set the increment in megabytes for automatically extending the size of tablespace data file in InnoDB.

```
mariadb-backup --backup \
     --innodb-autoextend-increment=35
```

`--innodb-buffer-pool-filename`

Using this option has no effect. It is available to provide compatibility with the MariaDB Server.

`--innodb-buffer-pool-size`

Defines the memory buffer size InnoDB uses the cache data and indexes of the table.

```
--innodb-buffer-pool-size=124M
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can configure the buffer pool for InnoDB operations.

```
mariadb-backup --backup \
      --innodb-buffer-pool-size=124M
```

`--innodb-checksum-algorithm`

`innodb_checksum_algorithm` has been removed.

`--innodb-data-file-path`

Defines the path to individual data files.

```
--innodb-data-file-path=/path/to/file
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option you can define the path to InnoDB data files. Each path is appended to the `--innodb-data-home-dir` option.

```
mariadb-backup --backup \
     --innodb-data-file-path=ibdata1:13M:autoextend \
     --innodb-data-home-dir=/var/dbs/mysql/data
```

`--innodb-data-home-dir`

Defines the home directory for InnoDB data files.

```
--innodb-data-home-dir=PATH
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option you can define the path to the directory containing InnoDB data files. You can specific the files using the `--innodb-data-file-path` option.

```
mariadb-backup --backup \
     --innodb-data-file-path=ibdata1:13M:autoextend \
     --innodb-data-home-dir=/var/dbs/mysql/data
```

`--innodb-doublewrite`

Enables doublewrites for InnoDB tables.

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. When using this option, `mariadb-backup` improves fault tolerance on InnoDB tables with a doublewrite buffer. By default, this feature is enabled. Use this option to explicitly enable it. To disable doublewrites, use the `--skip-innodb-doublewrite` option.

```
mariadb-backup --backup \
     --innodb-doublewrite
```

`--innodb-encrypt-log`

Defines whether you want to encrypt InnoDB logs.

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can tell mariadb-backup that you want to encrypt logs from its InnoDB activity.

`--innodb-file-io-threads`

Defines the number of file I/O threads in InnoDB.

```
--innodb-file-io-threads=#
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can define the number of file I/O threads mariadb-backup uses on InnoDB tables.

```
mariadb-backup --backup \
     --innodb-file-io-threads=5
```

`--innodb-file-per-table`

Defines whether you want to store each InnoDB table as an `.ibd` file.

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option causes mariadb-backup to store each InnoDB table as an `.ibd` file in the target directory.

`--innodb-flush-method`

Defines the data flush method. Ignored from MariaDB 11.0. For the OS-level mechanisms behind these flag names, see Storage I/O: Buffering and Persistence.

```
--innodb-flush-method=fdatasync
                     | O_DSYNC
                     | O_DIRECT
                     | O_DIRECT_NO_FSYNC
                     | ALL_O_DIRECT
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can define the data flush method `mariadb-backup` uses with InnoDB tables.

```
mariadb-backup --backup \
      --innodb-flush-method==_DIRECT_NO_FSYNC
```

`--innodb-io-capacity`

Defines the number of IOP's the utility can perform.

```
--innodb-io-capacity=#
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can limit the I/O activity for InnoDB background tasks. It should be set around the number of I/O operations per second that the system can handle, based on drive or drives being used.

```
mariadb-backup --backup \
     --innodb-io-capacity=200
```

`--innodb-log-buffer-size`

The size of the buffer that will be used for reading log during `mariadb-backup --prepare`. Ignored when using `--innodb-log-file-mmap`.

`--innodb-log-checksums`

Defines whether to include checksums in the InnoDB logs.

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can explicitly set `mariadb-backup` to include checksums in the InnoDB logs. The feature is enabled by default. To disable it, use the `--skip-innodb-log-checksums` option.

```
mariadb-backup --backup \
      --innodb-log-checksums
```

`--innodb-log-checkpoint-now`

At the start of a backup, instruct the server to write out all modified pages to the data files, to minimize the size of the `ib_logfile0` that needs to be copied.

```
mariadb-backup --backup \
      --innodb-log-checkpoint-now
```

`--innodb-log-file-mmap`

> **Note:**
>
> This variable is available from MariaDB 11.4.4 and 10.11.10.

When this option is enabled, `mariadb-backup` reads the `ib_logfile0` via a memory mapping, rather than by reading into a separately allocated buffer of `--innodb-log-buffer-size`.

`--innodb-log-files-in-group`

This option has no functionality in `mariadb-backup`. It exists for MariaDB Server compatibility.

`--innodb-log-group-home-dir`

Defines the path to InnoDB log files.

```
--innodb-log-group-home-dir=PATH
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can define the path to InnoDB log files.

```
mariadb-backup --backup \
     --innodb-log-group-home-dir=/path/to/logs
```

`--innodb-max-dirty-pages-pct`

Defines the percentage of dirty pages allowed in the InnoDB buffer pool.

```
--innodb-max-dirty-pages-pct=#
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can define the maximum percentage of dirty, (that is, unwritten) pages that mariadb-backup allows in the InnoDB buffer pool.

```
mariadb-backup --backup \
     --innodb-max-dirty-pages-pct=80
```

`--innodb-open-files`

Defines the number of files kept open at a time.

```
--innodb-open-files=#
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can set the maximum number of files InnoDB keeps open at a given time during backups.

```
mariadb-backup --backup \
      --innodb-open-files=10
```

`--innodb-page-size`

Defines the universal page size.

```
--innodb-page-size=#
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can define the universal page size in bytes for `mariadb-backup`.

```
mariadb-backup --backup \
     --innodb-page-size=16k
```

`--innodb-read-io-threads`

Defines the number of background read I/O threads in InnoDB.

```
--innodb-read-io-threads=#
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can set the number of I/O threads MariaDB uses when reading from InnoDB.

```
mariadb-backup --backup \
      --innodb-read-io-threads=4
```

`--innodb-undo-directory`

Defines the directory for the undo tablespace files.

```
--innodb-undo-directory=PATH
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can define the path to the directory where you want MariaDB to store the undo tablespace on InnoDB tables. The path can be absolute.

```
mariadb-backup --backup \
     --innodb-undo-directory=/path/to/innodb_undo
```

`--innodb-undo-tablespaces`

Defines the number of undo tablespaces to use.

```
--innodb-undo-tablespaces=#
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can define the number of undo tablespaces you want to use during the backup.

```
mariadb-backup --backup \
      --innodb-undo-tablespaces=10
```

`--innodb-use-native-aio`

Defines whether you want to use native AI/O.

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can enable the use of the native asynchronous I/O subsystem. It is only available on Linux operating systems.

```
mariadb-backup --backup \
      --innodb-use-native-aio
```

`--innodb-write-io-threads`

Defines the number of background write I/O threads in InnoDB.

```
--innodb-write-io-threads=#
```

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can set the number of background write I/O threads `mariadb-backup` uses.

```
mariadb-backup --backup \
     --innodb-write-io-threads=4
```

`--kill-long-queries-timeout`

Defines the timeout for blocking queries.

```
--kill-long-queries-timeout=#
```

When `mariadb-backup` runs, it issues a `FLUSH TABLES WITH READ LOCK` statement. It then identifies blocking queries. Using this option you can set a timeout in seconds for these blocking queries. When the time runs out, `mariadb-backup` kills the queries.

The default value is `0`, which causes `mariadb-backup` to not attempt killing any queries.

```
mariadb-backup --backup \
      --kill-long-queries-timeout=10
```

`--kill-long-query-type`

Defines the query type the utility can kill to unblock the global lock.

```
--kill-long-query-type=ALL | UPDATE | SELECT
```

When `mariadb-backup` encounters a query that sets a global lock, it can kill the query in order to free up MariaDB Server for the backup. Using this option, you can choose the types of query it kills: `SELECT`, `UPDATE`, or both set with `ALL`. The default is `ALL`.

```
mariadb-backup --backup \
      --kill-long-query-type=UPDATE
```

`--lock-ddl-per-table`

Prevents DDL for each table to be backed up by acquiring MDL lock on that.

> **Note:**
>
> Unless the `--no-lock` option is also specified, conflicting DDL queries are killed at the end of backup This is done to avoid a deadlock between `FLUSH TABLE WITH READ LOCK`, user's DDL query (`ALTER`, `RENAME`), and MDL lock on table.

`--log`

This option has no functionality. It is set to ensure compatibility with MySQL.

`--log-bin`

Defines the base name for the log sequence.

```
--log-bin[=name]
```

Using this option you, you can set the base name for `mariadb-backup` to use in log sequences.

`--log-copy-interval`

Defines the copy interval between checks done by the log copying thread.

```
--log-copy-interval=#
```

Using this option, you can define the copy interval mariadb-backup uses between checks done by the log copying thread. The given value is in milliseconds.

```
mariadb-backup --backup \
      --log-copy-interval=50
```

`--log-innodb-page-corruption`

Continue backup if InnoDB corrupted pages are found. The pages are logged in `innodb_corrupted_pages` and backup is finished with error. `--prepare` will try to fix corrupted pages. If `innodb_corrupted_pages` exists after –prepare in base backup directory, backup still contains corrupted pages and can not be considered as consistent.

`--move-back`

Restores the backup to the data directory.

Using this command, `mariadb-backup` moves the backup from the target directory to the data directory, as defined by the `--datadir` option. You must stop the MariaDB Server before running this command. The data directory must be empty. If you want to overwrite the data directory with the backup, use the `--force-non-empty-directories` option.

Bear in mind, before you can restore a backup, you first need to run `mariadb-backup` with the `--prepare` option. In the case of full backups, this makes the files point-in-time consistent. With incremental backups, this applies the deltas to the base backup. Once the backup is prepared, you can run `--move-back` to apply it to MariaDB Server.

```
mariadb-backup --move-back \
      --datadir=/var/mysql
```

Running the `--move-back` command moves the backup files to the data directory. Use this command if you don't want to save the backup for later. If you do want to save the backup for later, use the `--copy-back` option.

`--mysqld`

Used internally to prepare a backup.

`--no-backup-locks`

`mariadb-backup` locks the database by default when it runs. This option disables support for Percona Server's backup locks.

When backing up Percona Server, mariadb-backup would use backup locks by default. To be specific, backup locks refers to the `LOCK TABLES FOR BACKUP` and `LOCK BINLOG FOR BACKUP` statements. This option can be used to disable support for Percona Server's backup locks. This option has no effect when the server does not support Percona's backup locks.

Deprecated and has no effect from MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5 and MariaDB 11.2.4 as MariaDB now always uses backup locks for better performance. See MDEV-32932.

```
mariadb-backup --backup --no-backup-locks
```

`--no-lock`

Disables table locks with the `FLUSH TABLE WITH READ LOCK` statement.

Using this option causes mariadb-backup to disable table locks with the `FLUSH TABLE WITH READ LOCK` statement. Only use this option if:

- You are not executing DML statements on non-InnoDB tables during the backup. This includes the `mysql` database system tables (which are MyISAM).
- You are not executing any DDL statements during the backup.
- You are *not* using the file `xtrabackup_binlog_info`, which is not consistent with the data when `--no-lock` is used. Use the file `xtrabackup_binlog_pos_innodb` instead.
- All tables you're backing up use the InnoDB storage engine.

```
mariadb-backup --backup --no-lock
```

If you're considering `--no-lock` due to backups failing to acquire locks, this may be due to incoming replication events preventing the lock. Consider using the `--safe-slave-backup` option to momentarily stop the replica thread. This alternative may help the backup to succeed without resorting to `--no-lock`.

The `--no-lock` option only provides a consistent backup if the user ensures that no DDL or non-transactional table updates occur during the backup. The `--no-lock` option is not supported by MariaDB plc.

`--no-timestamp`

This option prevents creation of a time-stamped subdirectory of the `BACKUP-ROOT-DIR` given on the command line. When it is specified, the backup is done in `BACKUP-ROOT-DIR` instead. This is only valid in `innobackupex` mode, which can be enabled with the `--innobackupex` option.

`--no-version-check`

Disables version check.

Using this option, you can disable mariadb-backup version check.

```
mariadb-backup --backup --no-version-check
```

`--open-files-limit`

Defines the maximum number of file descriptors.

```
--open-files-limit=#
```

Using this option, you can define the maximum number of file descriptors mariadb-backup reserves with `setrlimit()`.

```
mariadb-backup --backup \
      --open-files-limit=
```

`--parallel`

Defines the number of threads to use for parallel data file transfer.

```
--parallel=#
```

Using this option, you can set the number of threads mariadb-backup uses for parallel data file transfers. By default, it is set to 1.

`-p, --password`

Defines the password to use to connect to MariaDB Server.

```
--password=passwd
```

When you run `mariadb-backup`, it connects to MariaDB Server in order to access and back up the databases and tables. Using this option, you can set the password mariadb-backup uses to access the server. To set the user, use the `--user` option.

```
mariadb-backup --backup \
      --user=root \
      --password=root_password
```

`--plugin-dir`

Defines the directory for server plugins.

```
--plugin-dir=PATH
```

Using this option, you can define the path `mariadb-backup` reads for MariaDB Server plugins. It only uses it during the `--prepare` phase to load the encryption plugin. It defaults to the `plugin_dir` server system variable.

```
mariadb-backup --backup \
      --plugin-dir=/var/mysql/lib/plugin
```

`--plugin-load`

The option has been removed.

`-P, --port`

Defines the server port to connect to.

```
--port=#
```

When you run `mariadb-backup`, it connects to MariaDB Server in order to access and back up your databases and tables. Using this option, you can set the port the utility uses to access the server over TCP/IP. To set the host, see the `--host` option. Use `mysql --help` for more details.

```
mariadb-backup --backup \
      --host=192.168.11.1 \
      --port=3306
```

`--prepare`

Prepares an existing backup to restore to the MariaDB Server.

Files that `mariadb-backup` generates during `--backup` operations in the target directory are not ready for use on the Server. Before you can restore the data to MariaDB, you first need to prepare the backup.

In the case of full backups, the files are not point in time consistent, since they were taken at different times. If you try to restore the database without first preparing the data, InnoDB rejects the new data as corrupt. Running mariadb-backup with the `--prepare` command readies the data so you can restore it to MariaDB Server. When working with incremental backups, you need to use the `--prepare` command and the `--incremental-dir` option to update the base backup with the deltas from an incremental backup.

```
mariadb-backup --prepare
```

Once the backup is ready, you can use the `--copy-back` or the `--move-back` options to restore the backup to the server.

`--print-defaults`

Prints the utility argument list, then exits.

Using this argument, MariaDB prints the argument list to stdout and then exits. You may find this useful in debugging to see how the options are set for the utility.

```
mariadb-backup --print-defaults
```

`--print-param`

Prints the MariaDB Server options needed for `copy-back`.

Using this option, `mariadb-backup` prints to stdout the MariaDB Server options that the utility requires to run the `--copy-back` command option.

```
mariadb-backup --print-param
```

`--rollback-xa`

By default, mariadb-backup will not commit or rollback uncommitted XA transactions, and when the backup is restored, any uncommitted XA transactions must be manually committed using `XA COMMIT` or manually rolled back using `XA ROLLBACK`.

MariaDB starting with 10.5

mariadb-backup's `--rollback-xa` option is not present because the server has more robust ways of handling uncommitted XA transactions.

This is an experimental option. Do not use this option in older versions. Older implementation can cause corruption of InnoDB data.

`--rsync`

Defines whether to use rsync.

During normal operation, mariadb-backup transfers local non-InnoDB files using a separate call to `cp` for each file. Using this option, you can optimize this process by performing this transfer with rsync, instead.

```
mariadb-backup --backup --rsync
```

This option is not compatible with the `--stream` option.

Deprecated and has no effect from MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5 and MariaDB 11.2.4 as rsync will not work on tables that are in use. See MDEV-32932.

`--safe-slave-backup`

Stops replica SQL threads for backups.

When running `mariadb-backup` on a server that uses replication, you may occasionally encounter locks that block backups. Using this option, it stops replica SQL threads and waits until the `Slave_open_temp_tables` in the `SHOW STATUS` statement is zero. If there are no open temporary tables, the backup runs, otherwise the SQL thread starts and stops until there are no open temporary tables.

```
mariadb-backup --backup \
      --safe-slave-backup \
      --safe-slave-backup-timeout=500
```

The backup fails if the `Slave_open_temp_tables` doesn't reach zero after the timeout period set by the `--safe-slave-backup-timeout` option.

`--safe-slave-backup-timeout`

Defines the timeout for replica backups.

```
--safe-slave-backup-timeout=#
```

When running mariadb-backup on a server that uses replication, you may occasionally encounter locks that block backups. With the `--safe-slave-backup` option, it waits until the `Slave_open_temp_tables` in the `SHOW STATUS` statement reaches zero. Using this option, you set how long it waits. It defaults to 300.

```
mariadb-backup --backup \
      --safe-slave-backup \
      --safe-slave-backup-timeout=500
```

`--secure-auth`

Refuses client connections to servers using the older protocol.

Using this option, you can set it explicitly to refuse client connections to the server when using the older protocol, from before 4.1.1. This feature is enabled by default. Use the `--skip-secure-auth` option to disable it.

```
mariadb-backup --backup --secure-auth
```

`--skip-innodb-adaptive-hash-index`

Disables InnoDB Adaptive Hash Index.

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option you can explicitly disable the InnoDB Adaptive Hash Index. This feature is enabled by default for mariadb-backup. If you want to explicitly enable it, use `--innodb-adaptive-hash-index`.

```
mariadb-backup --backup \
      --skip-innodb-adaptive-hash-index
```

`--skip-innodb-doublewrite`

Disables doublewrites for InnoDB tables.

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. When doublewrites are enabled, InnoDB improves fault tolerance with a doublewrite buffer. By default this feature is turned on. Using this option you can disable it for mariadb-backup. To explicitly enable doublewrites, use the `--innodb-doublewrite` option.

```
mariadb-backup --backup \
     --skip-innodb-doublewrite
```

`--skip-innodb-log-checksums`

Defines whether to exclude checksums in the InnoDB logs.

`mariadb-backup` initializes its own embedded instance of InnoDB using the same configuration as defined in the configuration file. Using this option, you can set mariadb-backup to exclude checksums in the InnoDB logs. The feature is enabled by default. To explicitly enable it, use the `--innodb-log-checksums` option.

`--skip-secure-auth`

Refuses client connections to servers using the older protocol.

Using this option, you can set it accept client connections to the server when using the older protocol, from before 4.1.1. By default, it refuses these connections. Use the `--secure-auth` option to explicitly enable it.

```
mariadb-backup --backup --skip-secure-auth
```

`--slave-info`

Prints the binary log position and the name of the primary server.

If the server is a replica, then this option causes `mariadb-backup` to print the hostname of the replica's replication primary and the binary log file and position of the replica's SQL thread to `stdout`.

This option also causes `mariadb-backup` to record this information as a `CHANGE MASTER` statement that can be used to set up a new server as a replica of the original server's primary after the backup has been restored. This information are written to the `xtrabackup_slave_info` file.

`mariadb-backup` does **not** check if GTIDs are being used in replication. It takes a shortcut and assumes that if the `gtid_slave_pos` system variable is non-empty, then it writes the `CHANGE MASTER` statement with the `MASTER_USE_GTID` option set to `slave_pos`. Otherwise, it writes the `CHANGE MASTER` statement with the `MASTER_LOG_FILE` and `MASTER_LOG_POS` options using the primary's binary log file and position. See MDEV-19264 for more information.

```
mariadb-backup --slave-info
```

`-S, --socket`

Defines the socket for connecting to local database.

```
--socket=name
```

Using this option, you can define the UNIX domain socket you want to use when connecting to a local database server. The option accepts a string argument. For more information, see the `mysql --help` command.

```
mariadb-backup --backup \
      --socket=/var/mysql/mysql.sock
```

`--ssl`

Enables TLS. By using this option, you can explicitly configure `mariadb-backup` to encrypt its connection with TLS when communicating with the server. You may find this useful when performing backups in environments where security is extra important or when operating over an insecure network.

TLS is also enabled even without setting this option when certain other TLS options are set. For example, see the descriptions of the following options:

- --ssl-ca
- --ssl-capath
- --ssl-cert
- --ssl-cipher
- --ssl-key

`--ssl-ca`

Defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use the absolute path, not a relative path. For example:

```
--ssl-ca=/etc/my.cnf.d/certificates/ca.pem
```

This option is usually used with other TLS options. For example:

```
mariadb-backup --backup \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem
```

See Secure Connections Overview: Certificate Authorities (CAs) for more information.

This option implies the `--ssl` option.

`--ssl-capath`

Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. For example:

```
--ssl-capath=/etc/my.cnf.d/certificates/ca/
```

This option is usually used with other TLS options. For example:

```
mariadb-backup --backup \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --ssl-capath=/etc/my.cnf.d/certificates/ca/
```

The directory specified by this option needs to be run through the openssl rehash command.

See Secure Connections Overview: Certificate Authorities (CAs) for more information

This option implies the `--ssl` option.

`--ssl-cert`

Defines a path to the X509 certificate file to use for TLS. This option requires that you use the absolute path, not a relative path. For example:

```
--ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem
```

This option is usually used with other TLS options. For example:

```
mariadb-backup --backup \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem
```

This option implies the `--ssl` option.

`--ssl-cipher`

Defines the list of permitted ciphers or cipher suites to use for TLS. For example:

```
--ssl-cipher=name
```

This option is usually used with other TLS options. For example:

```
mariadb-backup --backup \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --ssl-cipher=TLSv1.2
```

To determine if the server restricts clients to specific ciphers, check the ssl_cipher system variable.

This option implies the `--ssl` option.

`--ssl-crl`

Defines a path to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. For example:

```
--ssl-crl=/etc/my.cnf.d/certificates/crl.pem
```

This option is usually used with other TLS options. For example:

```
mariadb-backup --backup \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --ssl-crl=/etc/my.cnf.d/certificates/crl.pem
```

See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information.

This option is only supported if `mariadb-backup` was built with OpenSSL. If `mariadb-backup` was built with yaSSL, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

`--ssl-crlpath`

Defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. For example:

```
--ssl-crlpath=/etc/my.cnf.d/certificates/crl/
```

This option is usually used with other TLS options. For example:

```
mariadb-backup --backup \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --ssl-crlpath=/etc/my.cnf.d/certificates/crl/
```

The directory specified by this option needs to be run through the openssl rehash command.

See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information.

This option is only supported if mariadb-backup was built with OpenSSL. If `mariadb-backup` was built with yaSSL, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

`--ssl-key`

Defines a path to a private key file to use for TLS. This option requires that you use the absolute path, not a relative path. For example:

```
--ssl-key=/etc/my.cnf.d/certificates/client-key.pem
```

This option is usually used with other TLS options. For example:

```
mariadb-backup --backup \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem
```

This option implies the `--ssl` option.

`--ssl-verify-server-cert`

Enables server certificate verification. This option is disabled by default.

This option is usually used with other TLS options. For example:

```
mariadb-backup --backup \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --ssl-verify-server-cert
```

`--stream`

Streams backup files to stdout.

```
--stream=xbstream
```

Using this command option, you can set `mariadb-backup` to stream the backup files to `stdout` in the given format. Currently, the supported format is `xbstream`.

```
mariadb-backup --stream=xbstream > backup.xb
```

To extract all files from the xbstream archive into a directory use the `mbstream` utility

```
mbstream  -x < backup.xb
```

If a backup is streamed, then `mariadb-backup` records the format in the `xtrabackup_info` file.

`--tables`

Defines the tables you want to include in the backup.

```
--tables=REGEX
```

Using this option, you can define what tables you want `mariadb-backup` to back up from the database. The table values are defined using Regular Expressions (regex[4]). To define the tables you want to exclude from the backup, see the `--tables-exclude` option.

```
mariadb-backup --backup \
     --databases=example \
     --tables=nodes_* \
     --tables-exclude=nodes_tmp
```

In the example, *`nodes_*`* matches tables named *`nodes`*, *`nodes_`*, *`nodes__`*, and so forth, because `*` means *zero or more occurrences of the previous character* (`_`).

If instead you want to back up all tables whose names start with *`nodes`*, the regular expression is `^nodes.`, and to exclude tables starting with *`nodes_tmp`*, the expression is *`^nodes_tmp.`*. (Notice the trailing period (`.`); it means *zero or more occurrences of characters following* *`nodes`*.) The command looks like this:

```
mariadb-backup --backup \
     --databases=example \
     --tables=^nodes. \
     --tables-exclude=^nodes_tmp.
```

In that example, some of the tables included via the `--tables` option are excluded by `--tables-excludes`. That works because `--tables-exclude` takes precedence over `--tables`.

You can specify multiple table name regex[5] patterns as a comma-separated list, for both the `--tables` and the `--tables-exclude` options.

The following command backs up all tables in the *`test1`* and *`test2`* databases, except the *`exclude_table`* table in the *`test2`* database, and stores the backup files under */`path/to/backups/`*:

```
mariadb-backup --backup \
     --tables=test1[.].*,test2[.].* \
     --tables-exclude=^test2[.]exclude_table
     --target-dir=/path/to/backups/
```

Warning:

The `--databases` and `--databases-exclude` options, if used, take precedence over `--tables` and `--tables-exclude`. That is, they can filter out tables, which are then not “visible” to the latter mentioned options.

If a backup is a partial backup, `mariadb-backup` records that detail in the `xtrabackup_info` file.

`--tables-exclude`

Defines the tables you want to exclude from the backup.

```
--tables-exclude=REGEX
```

Using this option, you can define what tables you want `mariadb-backup` to exclude from the backup. The table values are defined using Regular Expressions. To define the tables you want to include from the backup, see the `--tables` option.

> **Note:**
>
> See the `--tables` option for examples and hints regarding regular expressions.

If a backup is a partial backup, `mariadb-backup` records that detail in the `xtrabackup_info` file.

`--tables-file`

Defines path to file with tables for backups.

```
--tables-file=/path/to/file
```

Using this option, you can set a path to a file listing the tables you want to back up. `mariadb-backup` iterates over each line in the file. The format is `database.table`.

```
mariadb-backup --backup \
     --databases=example \
     --tables-file=/etc/mysql/backup-file
```

If a backup is a partial backup, then mariadb-backup will record that detail in the `xtrabackup_info` file.

`--target-dir`

Defines the destination directory.

```
--target-dir=/path/to/target
```

Using this option you can define the destination directory for the backup. `mariadb-backup` writes all backup files to this directory. mariadb-backup will create the directory, if it does not exist (but it does not create the full path recursively, i.e. at least parent directory if the `--target-dir` must exist.

```
mariadb-backup --backup \
       --target-dir=/data/backups
```

`--throttle`

Defines the limit for I/O operations per second in IOS values.

```
--throttle=#
```

Using this option, you can set a limit on the I/O operations mariadb-backup performs per second in IOS values. It is only used during the `--backup` option.

`--tls-version`

This option accepts a comma-separated list of TLS protocol versions. A TLS protocol version is only enabled if it is present in this list. All other TLS protocol versions will not be permitted. For example:

```
--tls-version="TLSv1.2,TLSv1.3"
```

This option is usually used with other TLS options. For example:

```
mariadb-backup --backup \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --tls-version="TLSv1.2,TLSv1.3"
```

See Secure Connections Overview: TLS Protocol Versions for more information.

`-t, --tmpdir`

Defines path for temporary files.

```
--tmpdir=/path/tmp[;/path/tmp...]
```

Using this option, you can define the path to a directory `mariadb-backup` uses in writing temporary files. If you want to use more than one, separate the values by a semicolon (that is, `;`). When passing multiple temporary directories, it cycles through them using round-robin.

```
mariadb-backup --backup \
     --tmpdir=/data/tmp;/tmp
```

`--use-memory`

Defines the buffer pool size that is used during the prepare stage.

```
--use-memory=124M
```

Using this option, you can define the buffer pool size for `mariadb-backup`. Use it instead of `buffer_pool_size`.

```
mariadb-backup --prepare \
      --use-memory=124M
```

`--user`

Defines the username for connecting to the MariaDB Server.

```
--user=name
-u name
```

When `mariadb-backup` runs, it connects to the specified MariaDB Server to get its backups. Using this option, you can define the database user used for authentication. Starting from MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3, MariaDB 11.3.2, MariaDB 11.4.1, if the `--user` option is omitted, the user name is detected from the OS.

```
mariadb-backup --backup \
      --user=root \
      --password=root_passwd
```

`--verbose`

Displays verbose output.

```
mariadb-backup --verbose
```

`--version`

Prints the `mariadb-backup` version information to `stdout`.

```
mariadb-backup --version
```

This page is licensed: CC BY-SA / Gnu FDL

Full Backup and Restore (mariadb-backup)

> **Note:**
>
> mariadb-backup was previously called mariabackup.

When using mariadb-backup, you have the option of performing a full or an incremental backup. Full backups create a complete backup of the database server in an empty directory while incremental backups update a previous backup with whatever changes to the data have occurred since the backup. This page documents how to perform full backups.

> **Note:**
>
> For a complete list of `mariadb-backup` options, see this page.
>
> For a detailed description of `mariadb-backup` functionality, see this page.

Backing up the Database Server

In order to back up the database, you need to run mariadb-backup with the `--backup` option to tell it to perform a backup and with the `--target-dir` option to tell it where to place the backup files. When taking a full backup, the target directory must be empty or it must not exist.

To take a backup, run the following command:

```
$ mariadb-backup --backup \
   --target-dir=/var/mariadb/backup/ \
   --user=mariadb-backup --password=mypassword
```

The time the backup takes depends on the size of the databases or tables you're backing up. You can cancel the backup if you need to, as the backup process does not modify the database.

`mariadb-backup` writes the backup files the target directory. If the target directory doesn't exist, it creates it. If the target directory exists and contains files, it raises an error and aborts.

Here is an example backup directory:

```
$ ls /var/mariadb/backup/

aria_log.0000001  mysql                    xtrabackup_checkpoints
aria_log_control  performance_schema       xtrabackup_info
backup-my.cnf     test                     xtrabackup_logfile
ibdata1           xtrabackup_binlog_info
```

Using the Backup History Feature

You can optionally use the `--history` option to record metadata about your full backup in the database. This creates a centralized log and allows future incremental backups to reference this full backup by name instead of by directory path.

```
$ mariadb-backup --backup \
   --target-dir=/var/mariadb/backup/ \
   --user=mariadb-backup --password=mypassword \
   --history=full_backup_weekly
```

- Privileges: The backup user requires `INSERT`, `CREATE`, and `ALTER` privileges on the history table (`mysql.mariadb_backup_history` in MariaDB 10.11+, or `PERCONA_SCHEMA.xtrabackup_history` in older versions).
- Failure Case: If the user lacks privileges, the backup will complete the file copy process but will fail at the final step with an `INSERT command denied` error.

Preparing the Backup for Restoration

The data files that `mariadb-backup` creates in the target directory are not point-in-time consistent, given that the data files are copied at different times during the backup operation. If you try to restore from these files, InnoDB notices the inconsistencies and crashes to protect you from corruption

Before you can restore from a backup, you first need to **prepare** it to make the data files consistent. You can do so with the `--prepare` option.

```
$ mariadb-backup --prepare \
   --target-dir=/var/mariadb/backup/
```

Backup Preparation Steps

1. Run `mariadb-backup --backup`. You must use a version of `mariadb-backup` that is compatible with the server version you are planning to upgrade from. For instance, when upgrading from MariaDB 10.4 to 10.5, you must use the 10.4 version of `mariadb-backup`, Another example: When upgrading from MariaDB 10.6 to 10.11, you must use the 10.6 version of `mariadb-backup`.
2. Run `mariadb-backup --prepare`, again using a compatible version of `mariadb-backup`, as described in the previous step.

Restoring the Backup

Once the backup is complete and you have prepared the backup for restoration (previous step), you can restore the backup using either the `--copy-back` or the `--move-back` options. The `--copy-back` option allows you to keep the original backup files. The `--move-back` option actually moves the backup files to the datadir, so the original backup files are lost.

- First, stop the MariaDB Server process.
- Then, ensure that `datadir` is empty.
- Then, run `mariadb-backup` with one of the options mentioned above:

```
$ mariadb-backup --copy-back \
   --target-dir=/var/mariadb/backup/
```

- Then, you may need to fix the file permissions.

When `mariadb-backup` restores a database, it preserves the file and directory privileges of the backup. However, it writes the files to disk as the user and group restoring the database. As such, after restoring a backup, you may need to adjust the owner of the data directory to match the user and group for the MariaDB Server, typically `mysql` for both. For example, to recursively change ownership of the files to the `mysql` user and group, you could execute:

```
$ chown -R mysql:mysql /var/lib/mysql/
```

- Finally, start the MariaDB Server process.

Restoring with Other Tools

Once a full backup is prepared, it is a fully functional MariaDB data directory. Therefore, as long as the MariaDB Server process is stopped on the target server, you can technically restore the backup using any file copying tool, such as `cp` or `rsync`. For example, you could also execute the following to restore the backup:

```
$ rsync -avrP /var/mariadb/backup /var/lib/mysql/
$ chown -R mysql:mysql /var/lib/mysql/
```

This page is licensed: CC BY-SA / Gnu FDL

Incremental Backup and Restore (mariadb-backup)

> **Note:**
>
> mariadb-backup was previously called mariabackup.

When using `mariadb-backup`, you have the option of performing a full or incremental backup. Full backups create a complete copy in an empty directory while incremental backups update a previous backup with new data. This page documents incremental backups.

InnoDB pages contain log sequence numbers, or LSN's. Whenever you modify a row on any InnoDB table on the database, the storage engine increments this number. When performing an incremental backup, `mariadb-backup` checks the most recent LSN for the backup against the LSN's contained in the database. It then updates any of the backup files that have fallen behind.

> **Note:**
>
> For a complete list of `mariadb-backup` options, see this page.
>
> For a detailed description of `mariadb-backup` functionality, see this page.

Backing up the Database Server

In order to take an incremental backup, you first need to take a full backup. In order to back up the database, you need to run `mariadb-backup` with the `--backup` option to tell it to perform a backup and with the `--target-dir` option to tell it where to place the backup files. When taking a full backup, the target directory must be empty or it must not exist.

To take a backup, run the following command:

```
$ mariadb-backup --backup \
   --target-dir=/var/mariadb/backup/ \
   --user=mariadb-backup --password=mypassword
```

This backs up all databases into the target directory `/var/mariadb/backup`. If you look in that directory at the `mariadb_backup_checkpoints` file, you can see the LSN data provided by InnoDB.

For example:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 1635102
last_lsn = 1635102
recover_binlog_info = 0
```

Backing up the Incremental Changes

Once you have created a full backup on your system, you can also back up the incremental changes as often as you would like.

In order to perform an incremental backup, you need to run `mariadb-backup` with the `--backup` option to tell it to perform a backup and with the `--target-dir` option to tell it where to place the incremental changes. The target directory must be empty. You also need to run it with the `--incremental-basedir` option to tell it the path to the full backup taken above. For example:

```
$ mariadb-backup --backup \
   --target-dir=/var/mariadb/inc1/ \
   --incremental-basedir=/var/mariadb/backup/ \
   --user=mariadb-backup --password=mypassword
```

This command creates a series of delta files that store the incremental changes in `/var/mariadb/inc1`. You can find a similar `mariadb_backup_checkpoints` file in this directory, with the updated LSN values.

> **Note:**
>
> In MariaDB 11.0 and earlier, this file is named `xtrabackup_checkpoints`. The new name is recognized as a fallback for backups taken with older versions.

For example:

```
backup_type = incremental
from_lsn = 1635102
to_lsn = 1635114
last_lsn = 1635114
recover_binlog_info = 0
```

To perform additional incremental backups, you can then use the target directory of the previous incremental backup as the incremental base directory of the next incremental backup. For example:

```
$ mariadb-backup --backup \
   --target-dir=/var/mariadb/inc2/ \
   --incremental-basedir=/var/mariadb/inc1/ \
   --user=mariadb-backup --password=mypassword
```

Using the Backup History Table

Alternatively, you can use the backup history table to manage your backup chain. This allows you to reference the previous backup by a logical name instead of a directory path.

1. **Create the Base Backup:** Take a full backup using the `--history` option.

   ```
   mariadb-backup --backup --target-dir=/full \
     --history=full_backup_1
   ```

2. **Create the Incremental Backup:** Use `--incremental-history-name` to specify the base backup's name. It is recommended to use `--history` again to record this new incremental backup.

   ```
   mariadb-backup --backup --target-dir=/inc1 \
     --incremental-history-name=full_backup_1 \
     --history=inc_backup_1
   ```

Note:

Privileges

Requires `SELECT` on the history table to find the base backup, and `INSERT` to record the new one.

Tip:

You can also use `--incremental-history-uuid` if you prefer to reference the unique ID generated by `mariadb-backup`.

Combining With `--stream` Output

When using `--stream`, for instance for compression or encryption using external tools, the `mariadb_backup_checkpoints` file containing the information where to continue from on the next incremental backup will also be part of the compressed/encrypted backup file, and so not directly accessible by default.

A directory containing an extra copy of the file can be created using the `--extra-lsndir=..` option though, and this directory can then be passed to the next incremental backup `--incremental-basedir=...`, for example:

```
# initial full backup
$ mariadb-backup --backup --stream=mbstream \
  --user=mariadb-backup --password=mypassword \
  --extra-lsndir=backup_base | gzip > backup_base.gz

# incremental backup
$ mariadb-backup --backup --stream=mbstream \
  --incremental-basedir=backup_base \
  --user=mariadb-backup --password=mypassword \
  --extra-lsndir=backup_inc1 | gzip > backup-inc1.gz
```

Preparing the Backup

Following the above steps, you have three backups in `/var/mariadb`: The first is a full backup, the others are increments on this first backup. In order to restore a backup to the database, you first need to apply the incremental backups to the base full backup. This is done using the `--prepare` command option.

Perform the following process:

First, prepare the base backup:

```
$ mariadb-backup --prepare \
   --target-dir=/var/mariadb/backup
```

Running this command brings the base full backup, that is, `/var/mariadb/backup`, into sync with the changes contained in the InnoDB redo log collected while the backup was taken.

Then, apply the incremental changes to the base full backup:

```
$ mariadb-backup --prepare \
   --target-dir=/var/mariadb/backup \
   --incremental-dir=/var/mariadb/inc1
```

Running this command brings the base full backup, that is, `/var/mariadb/backup`, into sync with the changes contained in the first incremental backup.

For each remaining incremental backup, repeat the last step to bring the base full backup into sync with the changes contained in that incremental backup.

Restoring the Backup

Once you've applied all incremental backups to the base, you can restore the backup using either the `--copy-back` or the `--move-back` options. The `--copy-back` option allows you to keep the original backup files. The `--move-back` option actually moves the backup files to the `datadir`, so the original backup files are lost.

- First, stop the MariaDB Server process.
- Then, ensure that the `datadir` is empty.
- Then, run `mariadb-backup` with one of the options mentioned above:

```
$ mariadb-backup --copy-back \
   --target-dir=/var/mariadb/backup/
```

- Then, you may need to fix the file permissions.

When `mariadb-backup` restores a database, it preserves the file and directory privileges of the backup. However, it writes the files to disk as the user and group restoring the database. As such, after restoring a backup, you may need to adjust the owner of the data directory to match the user and group for the MariaDB Server, typically `mysql` for both. For example, to recursively change ownership of the files to the `mysql` user and group, you could execute:

```
$ chown -R mysql:mysql /var/lib/mysql/
```

- Finally, start the MariaDB Server process.

This page is licensed: CC BY-SA / Gnu FDL

Partial Backup and Restore (mariadb-backup)

> **Note:**
>
> mariadb-backup was previously called mariabackup.

When using `mariadb-backup`, you have the option of performing partial backups. Partial backups allow you to choose which databases or tables to backup, as long as the table or partition involved is in an InnoDB file-per-table tablespace.This page documents how to perform partial backups.

> **Note:**
>
> For a complete list of `mariadb-backup` options, see this page.
>
> For a detailed description of `mariadb-backup` functionality, see this page.

Backing up the Database Server

Just like with full backups, in order to back up the database, you need to run mariadb-backup with the `--backup` option to tell it to perform a backup and with the `--target-dir` option to tell it where to place the backup files. The target directory must be empty or not exist.

For a partial backup, there are a few other arguments that you can provide as well:

- To tell it which databases to backup, you can provide the `--databases` option.
- To tell it which databases to exclude from the backup, you can provide the `--databases-exclude` option.
- To tell it to check a file for the databases to backup, you can provide the `--databases-file` option.
- To tell it which tables to back up, you can use the `--tables` option.
- To tell it which tables to exclude from the backup, you can provide the `--tables-exclude` option.
- To tell it to check a file for specific tables to backup, you can provide the `--tables-file` option.

The non-file partial backup options support regex in the database and table names.

For example, to take a backup of any database that starts with the string `app1_` and any table in those databases that start with the string `tab_`, run the following command:

```
$ mariadb-backup --backup \
   --target-dir=/var/mariadb/backup/ \
   --databases='app1 app2' --tables='tab_[0-9]+' \
   --user=mariadb-backup --password=mypassword
```

Using –history with Partial Backups

You can use the `--history` option with a partial backup to log the operation in the history table for auditing purposes.

```
mariadb-backup --backup --databases="db1" \
  --target-dir=/backup --history=partial_db1
```

Warning:

You cannot use a partial backup as the base for an incremental backup history chain. The `--incremental-history-name` option is incompatible with partial backups because restoring partial incrementals requires specific preparation steps (`--export`) that the history feature does not automate.

Note:

`mariadb-backup` cannot back up a subset of partitions from a partitioned table. Backing up a partitioned table is an all-or-nothing selection. See MDEV-17132 about that. If you need to backup a subset of partitions, one possibility is that instead of using `mariadb-backup` , you can export the file-per-table tablespaces of the partitions.

The time the backup takes depends on the size of the databases or tables you're backing up. You can cancel the backup if you need to, as the backup process does not modify the database.

`mariadb-backup` writes the backup files to the target directory. If the target directory doesn't exist, then it creates it. If the target directory exists and contains files, then it raises an error and aborts.

Preparing the Backup

Just like with full backups, the data files that `mariadb-backup` creates in the target directory are not point-in-time consistent, given that the data files are copied at different times during the backup operation. If you try to restore from these files, InnoDB notices the inconsistencies and crashes to protect you from corruption. In fact, for partial backups, the backup is not even a completely functional MariaDB data directory, so InnoDB would raise more errors than it would for full backups. This point will also be very important to keep in mind during the restore process.

Before you can restore from a backup, you first need to **prepare** it to make the data files consistent. You can do so with the `--prepare` command option.

Partial backups rely on InnoDB's transportable tablespaces. For MariaDB to import tablespaces like these, InnoDB looks for a file with a `.cfg` extension. For `mariadb-backup` to create these files, you also need to add the `--export` option during the prepare step.

For example, you might execute the following command:

```
$ mariadb-backup --prepare --export \
   --target-dir=/var/mariadb/backup/
```

If this operation completes without error, then the backup is ready to be restored.

`mariadb-backup` did not support the `--export` option. See MDEV-13466 about that. This means that `mariadb-backup` could not create `.cfg` files for InnoDB file-per-table tablespaces during the `--prepare` stage. You can still import file-per-table tablespaces without the `.cfg` files in many cases, so it may still be possible in those versions to restore partial backups or to restore individual tables and partitions with just the `.ibd` files. If you have a full backup and you need to create `.cfg` files for InnoDB file-per-table tablespaces, then you can do so by preparing the backup as usual without the `--export` option, and then restoring the backup, and then starting the server. At that point, you can use the server's built-in features to copy the transportable tablespaces.

Restoring the Backup

The restore process for partial backups is quite different than the process for full backups. A partial backup is not a completely functional data directory. The data dictionary in the InnoDB system tablespace will still contain entries for the databases and tables that were not included in the backup.

Rather than using the `--copy-back` or the `--move-back`, each individual InnoDB file-per-table tablespace file will have to be manually imported into the target server. The process that is used to import the file will depend on whether partitioning is involved.

Restoring Individual Non-Partitioned Tables

To restore individual non-partitioned tables from a backup, find the `.ibd` and `.cfg` files for the table in the backup, and then import them using the Importing Transportable Tablespaces for Non-partitioned Tables process.

Restoring Individual Partitions and Partitioned Tables

To restore individual partitions or partitioned tables from a backup, find the `.ibd` and `.cfg` files for the partitions in the backup, and then import them using the Importing Transportable Tablespaces for Partitioned Tables process.

Restoring Individual Tables with Full-Text Indexes

When restoring a table with a full-text search (FTS) index, InnoDB may throw a schema mismatch error.

In this case, to restore the table, it is recommended to:

- Remove the corresponding `.cfg` file.
- Restore data to a table without any secondary indexes including FTS.
- Add the necessary secondary indexes to the restored table.

For example, to restore table `t1` with FTS index from database `db1`:

1. In the MariaDB Command-Line Client, drop the table you are going to restore:

```
DROP TABLE IF EXISTS db1.t1;
```

2. Create an empty table for the data being restored:

```
CREATE TABLE db1.t1(f1 CHAR(10)) ENGINE=INNODB;
```

3. Modify the table to discard the tablespace:

```
ALTER TABLE db1.t1 DISCARD TABLESPACE;
```

4. In the operating system shell, copy the table files from the backup to the data directory of the corresponding database:

```
$ sudo cp /data/backups/part/db1/t1.* /var/lib/mysql/db1
```

5. Remove the `.cfg` file from the data directory:

```
$ sudo rm /var/lib/mysql/db1/t1.cfg
```

6. Change the owner of the newly copied files to the system user running MariaDB Server:

```
$ sudo chown mysql:mysql /var/lib/mysql/db1/t1.*
```

7. In the MariaDB Command-Line Client, import the copied tablespace:

```
ALTER TABLE db1.t1 IMPORT TABLESPACE;
```

8. Verify that the data has been successfully restored:

```
SELECT * FROM db1.t1;
```

9. Add the necessary secondary indexes:

```
ALTER TABLE db1.t1 FORCE, ADD FULLTEXT INDEX f_idx(f1);
```

10. The table is now fully restored:

```
SHOW CREATE TABLE db1.t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `f1` char(10) DEFAULT NULL,
  FULLTEXT KEY `f_idx` (`f1`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

This page is licensed: CC BY-SA / Gnu FDL

Point-In-Time Recovery (PITR, mariadb-backup)

Recovering from a backup can restore the data directory at a specific point in time, but it does not restore the binary log. In a point-in-time recovery, start by restoring the data directory from a full or incremental backup, then use the mysqlbinlog utility to restore the binary log data to a specific point in time.

> **Warning:**
>
> Run the following commands as root unless indicated otherwise.

> **Step 1**
>
> **Find the binary log position to restore to.**
>
> When MariaDB Backup runs on a MariaDB Server with binary logs enabled (which is a prerequisite for PITR), it stores binary log information in the `mariadb_backup_binlog_info` (or `xtrabackup_binlog_info` in older releases) file. Consult this file to find the name of the binary log position to use. In the following example, the log position is 321:

```
cat /data/backups/full/mariadb_backup_binlog_info

mariadb-node4.00001     321
```

Step 2

Configure a new data directory.

Update the configuration file (for instance, `my.cnf`) to use a new data directory.

```
[mysqld]
datadir=/var/lib/mysql_new
```

Step 3

Restore the backup.

Restore from the backup as explained here.

Step 4

Start the database server.

Start MariaDB Server.

```
systemctl start mariadb
```

Step 5

Create a script using mysqlbinlog.

Use the mysqlbinlog utility to create an SQL script, using the binary log file in the *old* data directory, the start position in the `xtrabackup_binlog_info` file, and the date and time you want to restore to. Issue the following command *as a regular user*:

```
$ mysqlbinlog --start-position=321 \
      --stop-datetime="2019-06-28 12:00:00" \
      /var/lib/mysql/mariadb-node4.00001 \
      > mariadb-binlog.sql
```

Step 6

Run the script.

In the *new* data directory, run the script created in the previous step:

```
$ mariadb < mariadb-binlog.sql
```

See Also

- Point-In-Time Recovery (InnoDB log archiving) — an alternative PITR procedure that replays archived InnoDB write-ahead logs instead of binary logs. Available from MariaDB 13.0.

Warning:

`mariadb-backup` does not support the `innodb_log_archive =ON` log format and fails when the server is running with `innodb_log_archive=ON`. Use the InnoDB log archiving PITR procedure instead in that configuration.

This page is licensed: CC BY-SA / Gnu FDL

Restoring Individual Databases From a Full Backup (mariadb-backup)

Individual Database Restores with mariadb-backup from Full Backup

> **Note:**
>
> mariadb-backup was previously called mariabackup.

This method is to solve a flaw with mariadb-backup; it cannot do single database restores from a full backup easily. There is a blog post that details a way to do this, but it's a manual process which is fine for a few tables but if you have hundreds or even thousands of tables then it would be impossible to do quickly.

We can't just move the data files to the datadir as the tables are not registered in the engines, so the database will error. Currently, the only effective method is to a do full restore in a test database and then dump the database that requires restoring or running a partial backup.

This has only been tested with InnoDB. Also, if you have stored procedures or triggers then these will need to be deleted and recreated.

Some of the issues that this method overcomes:

- Tables not registered in the InnoDB engine so will error when you try to select from a table if you move the data files into the datadir
- Tables with foreign keys need to be created without keys, otherwise it will error when you discard the tablespace

Single Node

Below is the process to perform a single database restore.

Firstly, we will need the table structure from a mariadb-dump backup with the –no-data option. I recommend this is done at least once per day or every six hours via a cronjob. As it is just the structure, it are very fast.

```
mariadb-dump -u root -p --all-databases --no-data > nodata.sql
```

Using SED to return only the table structure we require, then use vim or another text editor to make sure nothing is left.

```
sed -n '/Current Database: `DATABASENAME`/, /Current Database:/p' nodata.sql > trimednodata.sql
vim trimednodata.sql
```

Prepare the backup with any incremental-backup-and-restores that you have, and then run the following on the full backup folder using the `--export` option to generate files with .cfg extensions which InnoDB will look for.

```
mariadb-backup --prepare --export --target-dir=/media/backups/fullbackupfolder
```

Once we have done these steps, we can then import the table structure. If you have used the `--all-databases` option, then you will need to either use SED or open it in a text editor and export out tables that you require. You will also need to log in to the database and create the database if the dump file doesn't. Run the following command below:

```
mysql -u root -p schema_name < nodata.sql
```

Once the structure is in the database, we have now registered the tables to the engine. Next, we will run the following statements in the information_schema database, to export statements to import/discard table spaces and drop and create foreign keys which we will use later. (edit the `CONSTRAINT_SCHEMA` and `TABLE_SCHEMA` `WHERE` clause to the database you are restoring. Also, add the following lines after your `SELECT` and before the `FROM` to have MariaDB export the files to the OS)

```
SELECT ...
INTO OUTFILE '/tmp/filename.SQL'
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
FROM ...
```

The following are the statements that we will need later.

```
USE information_schema;
SELECT concat("ALTER TABLE ",table_name," DISCARD TABLESPACE;")  AS discard_tablespace
FROM information_schema.tables
WHERE TABLE_SCHEMA="DATABASENAME";

SELECT concat("ALTER TABLE ",table_name," IMPORT TABLESPACE;") AS import_tablespace
FROM information_schema.tables
WHERE TABLE_SCHEMA="DATABASENAME";

SELECT
CONCAT ("ALTER TABLE ", rc.CONSTRAINT_SCHEMA, ".",rc.TABLE_NAME," DROP FOREIGN KEY ",
rc.CONSTRAINT_NAME,";") AS drop_keys
FROM REFERENTIAL_CONSTRAINTS AS rc
WHERE CONSTRAINT_SCHEMA = 'DATABASENAME';

SELECT
CONCAT ("ALTER TABLE ",
KCU.CONSTRAINT_SCHEMA, ".",
KCU.TABLE_NAME,"
ADD CONSTRAINT ",
KCU.CONSTRAINT_NAME, "
FOREIGN KEY ", "
(`",KCU.COLUMN_NAME,"`)", "
REFERENCES `",REFERENCED_TABLE_NAME,"`
(`",REFERENCED_COLUMN_NAME,"`)" ,"
ON UPDATE " ,(SELECT UPDATE_RULE FROM REFERENTIAL_CONSTRAINTS WHERE CONSTRAINT_NAME = KCU.CONSTRAINT_NAME
AND CONSTRAINT_SCHEMA = KCU.CONSTRAINT_SCHEMA),"
ON DELETE ",(SELECT DELETE_RULE FROM REFERENTIAL_CONSTRAINTS WHERE CONSTRAINT_NAME = KCU.CONSTRAINT_NAME
AND CONSTRAINT_SCHEMA = KCU.CONSTRAINT_SCHEMA),";") AS add_keys
FROM KEY_COLUMN_USAGE AS KCU
WHERE KCU.CONSTRAINT_SCHEMA = 'DATABASENAME'
AND KCU.POSITION_IN_UNIQUE_CONSTRAINT >= 0
AND KCU.CONSTRAINT_NAME NOT LIKE 'PRIMARY';
```

Once we have run those statements, and they have been exported to a Linux directory or copied from a GUI interface.

Run the `ALTER DROP KEYS` statements in the database.

```
ALTER TABLE schemaname.tablename DROP FOREIGN KEY key_name;
...
```

Once completed, run the `DROP TABLE SPACE` statements in the database:

```
ALTER TABLE test DISCARD TABLESPACE;
...
```

Exit out the database and change into the directory of the full backup location. Run the following commands to copy all the `.cfg` and `.ibd` files to the datadir such as `/var/lib/mysql/testdatabase` (change the `datadir` location if needed). Learn more about files that `mariadb-backup` generates with files-created-by-mariadb-backup.

```
cp *.cfg /var/lib/mysql
cp *.ibd /var/lib/mysql
```

After moving the files, it is very important that MySQL is the owner of the files, otherwise it won't have access to them and will error when we import the tablespaces.

```
sudo chown -R mysql:mysql /var/lib/mysql
```

Run the import table spaces statements in the database.

```
ALTER TABLE test IMPORT TABLESPACE;
...
```

Run the add key statements in the database

```
ALTER TABLE schmeaname.tablename ADD CONSTRAINT key_name FOREIGN KEY (`column_name`) REFERENCES
`foreign_table` (`colum_name`) ON UPDATE NO ACTION ON DELETE NO ACTION;
...
```

We have successfully restored a single database. To test that this has worked, we can do a basic check on some tables.

```
USE DATABASE
SELECT * FROM test LIMIT 10;
```

Replica nodes

If you have a primary-replica set up, it would be best to follow the sets above for the primary node and then either take a full mariadb-dump or take a new full mariadb-backup and restore this to the replica. You can find more information about restoring a replica with mariadb-backup in Setting up a Replica with mariadb-backup

After running the below command, copy to the replica and use the `LESS` linux command to grab the change master statement. Remember to follow this process: Stop replica > restore data > run `CHANGE MASTER` statement > start replica again.

```
mariadb-dump -u user -p --single-transaction --master-data=2 > fullbackup.sql
```

Please follow Setting up a Replica with mariadb-backup on restoring a replica with `mariadb-backup`:

```
$ mariadb-backup \
   --slave-info --safe-slave-backup \
   --target-dir=/var/mariadb/backup/ \
   --user=mariadb-backup --password=mypassword
```

Galera Cluster

For this process to work with Galera cluster, we first need to understand that some statements are not replicated across Galera nodes. One of which is the `DISCARD` and `IMPORT` for `ALTER TABLES` statements, and these statements will need to be ran on all nodes. We also need to run the OS level steps on each server as seen below.

Run the `ALTER DROP KEYS` statements on ONE NODE as these are replicated.

```
ALTER TABLE schemaname.tablename DROP FOREIGN KEY key_name;
...
```

Once completed, run the `DROP TABLE SPACE` statements on EVERY NODE, as these are not replicated.

```
ALTER TABLE test DISCARD TABLESPACE;
...
```

Exit out the database and change into the directory of the full backup location. Run the following commands to copy all the `.cfg` and `.ibd` files to the datadir such as `/var/lib/mysql/testdatabase` (change the `datadir` location if needed). Learn more about files that `mariadb-backup` generates with files-created-by-mariadb-backup. This step needs to be done on all nodes. You will need to copy the backup files to each node, we can use the same backup on all nodes.

```
cp *.cfg /var/lib/mysql
cp *.ibd /var/lib/mysql
```

After moving the files, it is very important that MySQL is the owner of the files, otherwise it won't have access to them and will error when we import the tablespaces.

```
sudo chown -R mysql:mysql /var/lib/mysql
```

Run the import table spaces statements on EVERY NODE.

```
ALTER TABLE test IMPORT TABLESPACE;
...
```

Run the add key statements on ONE NODE.

```
ALTER TABLE schmeaname.tablename ADD CONSTRAINT key_name FOREIGN KEY (`column_name`) REFERENCES
`foreign_table` (`colum_name`) ON UPDATE NO ACTION ON DELETE NO ACTION;
...
```

This page is licensed: CC BY-SA / Gnu FDL

Restoring Individual Tables and Partitions (mariadb-backup)

Restoring Individual Tables and Partitions with mariadb-backup

> **Note:**
>
> mariadb-backup was previously called mariabackup.

When using `mariadb-backup`, you don't necessarily need to restore every table and/or partition that was backed up. Even if you're starting from a full backup, it is certainly possible to restore only certain tables and/or partitions from the backup, as long as the table or partition involved is in an InnoDB file-per-table tablespace. This page documents how to restore individual tables and partitions.

> **Note:**
>
> For a complete list of `mariadb-backup` options, see this page.
>
> For a detailed description of `mariadb-backup` functionality, see this page.

Preparing the Backup

Before you can restore from a backup, you first need to **prepare** it to make the data files consistent. You can do so with the `--prepare` option.

The ability to restore individual tables and partitions relies on InnoDB's transportable tablespaces. For MariaDB to import tablespaces like these, InnoDB looks for a file with a `.cfg` extension. For mariadb `-backup` to create these files, you also need to add the `--export` option during the prepare step.

For example, you might execute the following command:

```
$ mariadb-backup --prepare --export \
   --target-dir=/var/mariadb/backup/ \
   --user=mariadb-backup --password=mypassword
```

If this operation completes without error, then the backup is ready to be restored.

Note

`mariadb-backup` did not support the –export option to begin with. See MDEV-13466 about that. In earlier versions of MariaDB, this means that mariadb-backup could not create `.cfg` files for InnoDB file-per-table tablespaces during the `--prepare` stage. You can still import file-per-table tablespaces without the `.cfg` files in many cases, so it may still be possible in those versions to restore partial backups or to restore individual tables and partitions with just the `.ibd` files. If you have a full backup and you need to create `.cfg` files for InnoDB file-per-table tablespaces, then you can do so by preparing the backup as usual without the `--export` option, and then restoring the backup, and then starting the server. At that point, you can use the server's built-in features to copy the transportable tablespaces.

Restoring the Backup

The restore process for restoring individual tables and/or partitions is quite different than the process for full backups.

Rather than using the `--copy-back` or the `--move-back`, each individual InnoDB file-per-table tablespace file will have to be manually imported into the target server. The process that is used to restore the backup will depend on whether partitioning is involved.

Restoring Individual Non-Partitioned Tables

To restore individual non-partitioned tables from a backup, find the `.ibd` and `.cfg` files for the table in the backup, and then import them using the Importing Transportable Tablespaces for Non-partitioned Tables process.

Restoring Individual Partitions and Partitioned Tables

To restore individual partitions or partitioned tables from a backup, find the `.ibd` and `.cfg` files for the partitions in the backup, and then import them using the Importing Transportable Tablespaces for Partitioned Tables process.

This page is licensed: CC BY-SA / Gnu FDL

Setting up a Replica (mariadb-backup)

Setting up a Replica with mariadb-backup

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

> **Note:**
>
> mariadb-backup was previously called mariabackup.

This page documents how to set up a replica from a backup.

If you are using MariaDB Galera Cluster, then you may want to try one of the following pages instead:

- Configuring MariaDB Replication between MariaDB Galera Cluster and MariaDB Server
- Configuring MariaDB Replication between Two MariaDB Galera Clusters

> **Note:**
>
> For a complete list of `mariadb-backup` options, see this page.
>
> For a detailed description of `mariadb-backup` functionality, see this page.

Back up the Database and Prepare it

The first step is to simply take and prepare a fresh full backup of a database server in the replication topology. If the source database server is the desired replication primary, then we do not need to add any additional options when taking the full backup. For example:

```
$ mariadb-backup --backup \
   --target-dir=/var/mariadb/backup/ \
   --user=mariadb-backup --password=mypassword
```

If the source database server is a replica of the desired primary, then we should add the `--slave-info` option, and possibly the `--safe-slave-backup` option. For example:

```
$ mariadb-backup --backup \
   --slave-info --safe-slave-backup \
   --target-dir=/var/mariadb/backup/ \
   --user=mariadb-backup --password=mypassword
```

And then we would prepare the backup as you normally would. For example:

```
$ mariadb-backup --prepare \
   --target-dir=/var/mariadb/backup/
```

Copy the Backup to the New Replica

Once the backup is done and prepared, we can copy it to the new replica. For example:

```
$ rsync -avP /var/mariadb/backup dbserver2:/var/mariadb/backup
```

Restore the Backup on the New Replica

At this point, we can restore the backup to the `datadir`, as you normally would. For example:

```
$ mariadb-backup --copy-back \
   --target-dir=/var/mariadb/backup/
```

And adjusting file permissions, if necessary:

```
$ chown -R mysql:mysql /var/lib/mysql/
```

Create a Replication User on the Primary

Before the new replica can begin replicating from the primary, we need to create a user account on the primary that the replica can use to connect, and we need to grant the user account the `REPLICATION SLAVE` privilege. For example:

```
CREATE USER 'repl'@'dbserver2' IDENTIFIED BY 'password';
GRANT REPLICATION SLAVE ON *.*  TO 'repl'@'dbserver2';
```

Configure the New Replica

Before we start the server on the new replica, we need to configure it. At the very least, we need to ensure that it has a unique server_id value. We also need to make sure other replication settings are what we want them to be, such as the various GTID system variables, if those apply in the specific environment.

Once configuration is done, we can start the MariaDB Server process on the new replica.

Start Replication on the New Replica

At this point, we need to get the replication coordinates of the primary from the original backup directory.

If we took the backup on the primary, then the coordinates are in the `xtrabackup_binlog_info` file. If we took the backup on another replica and if we provided the `--slave-info` option, then the coordinates are in the file `xtrabackup_slave_info` file.

mariadb-backup dumps replication coordinates in two forms: GTID coordinates and binary log file and position coordinates, like the ones you would normally see from `SHOW MASTER STATUS` output. We can choose which set of coordinates we would like to use to set up replication.

For example:

```
mariadb-bin.000096 568 0-1-2
```

Regardless of the coordinates we use, we will have to set up the primary connection using `CHANGE MASTER TO` and then start the replication threads with `START SLAVE`.

GTIDs

If we want to use GTIDs, then we will have to first set `gtid_slave_pos` to the GTID coordinates that we pulled from either the `xtrabackup_binlog_info` file or the `xtrabackup_slave_info` file in the backup directory. For example:

```
$ cat xtrabackup_binlog_info
mariadb-bin.000096 568 0-1-2
```

And then we would set `MASTER_USE_GTID=slave_pos` in the `CHANGE MASTER TO` statement. For example:

```
SET GLOBAL gtid_slave_pos = "0-1-2";
CHANGE MASTER TO
   MASTER_HOST="dbserver1",
   MASTER_PORT=3306,
   MASTER_USER="repl",
   MASTER_PASSWORD="password",
   MASTER_USE_GTID=slave_pos;
START SLAVE;
```

File and Position

If we want to use the binary log file and position coordinates, then we would set `MASTER_LOG_FILE` and `MASTER_LOG_POS` in the `CHANGE MASTER TO` statement to the file and position coordinates that we pulled; either the `xtrabackup_binlog_info` file or the `xtrabackup_slave_info` file in the backup directory, depending on whether the backup was taken from the primary or from a replica of the primary. For example:

```
CHANGE MASTER TO
   MASTER_HOST="dbserver1",
   MASTER_PORT=3306,
   MASTER_USER="repl",
   MASTER_PASSWORD="password",
   MASTER_LOG_FILE='mariadb-bin.000096',
   MASTER_LOG_POS=568;
START SLAVE;
```

Check the Status of the New Replica

We should be done setting up the replica now, so we should check its status with `SHOW SLAVE STATUS`. For example:

```
SHOW SLAVE STATUS\G
```

This page is licensed: CC BY-SA / Gnu FDL

Files Backed Up by mariadb-backup

> **Note:**
>
> mariadb-backup was previously called mariabackup.

Files Included in Backup

`mariadb-backup` backs up the files listed below.

InnoDB Data Files

mariadb-backup backs up the following InnoDB data files:

- InnoDB system tablespace
- InnoDB file-per-table tablespaces

MyRocks Data Files

> **CURRENT**
>
> `mariadb-backup` will back up tables that use the MyRocks storage engine. This data is located in the directory defined by the rocksdb_datadir system variable. `mariadb-backup` backs this data up by performing a checkpoint using the rocksdb_create_checkpoint system variable.
>
> **< 10.3.8 TO 10.2.16**
>
> `mariadb-backup` will back up tables that use the MyRocks storage engine.

Other Data Files

`mariadb-backup` also backs up files with the following extensions:

- `frm`
- `isl`
- `MYD`
- `MYI`
- `MAD`
- `MAI`

- `MRG`
- `TRG`
- `TRN`
- `ARM`
- `ARZ`
- `CSM`
- `CSV`
- `opt`
- `par`

Files Excluded From Backup

`mariadb-backup` does **not** back up the files listed below.

- InnoDB Temporary Tablespaces
- Binary logs
- Relay logs

This page is licensed: CC BY-SA / Gnu FDL

Files Created by mariadb-backup

> **Note:**
>
> mariadb-backup was previously called mariabackup.

`mariadb-backup` creates the following files:

`backup-my.cnf`

During the backup, any server options relevant to `mariadb-backup` are written to the `backup-my.cnf` option file, so that they can be re-read later during the `--prepare` stage.

`ib_logfile0`

`mariadb-backup` creates an empty InnoDB redo log file called `ib_logfile0` as part of the `--prepare` stage. This file has 3 roles:

1. In the source server, `ib_logfile0` is the first (and possibly the only) InnoDB redo log file.
2. In the non-prepared backup, `ib_logfile0` contains all of the InnoDB redo log copied during the backup.
3. During the `--prepare` stage, `ib_logfile0` is initialized as an empty InnoDB redo log file. That way, if the backup is manually restored, any pre-existing InnoDB redo log files get overwritten by the empty one. This helps to prevent certain kinds of known issues.

> **CURRENT**
>
> #### `mariadb_backup_binlog_info`
>
> This file stores the binary log file name and position that corresponds to the backup.
>
> This file also stores the value of the gtid_current_pos system variable that correspond to the backup, like this:
>
> ```
> mariadb-bin.000096 568 0-1-2
> ```

The values in this file are only guaranteed to be consistent with the backup if the –no-lock option was **not** provided when the backup was taken.

< 11.1

`xtrabackup_binlog_info`

This file stores the binary log file name and position that corresponds to the backup.

This file also stores the value of the gtid_current_pos system variable that correspond to the backup, like this:

```
mariadb-bin.000096 568 0-1-2
```

The values in this file are only guaranteed to be consistent with the backup if the –no-lock option was **not** provided when the backup was taken.

CURRENT

`mariadb_backup_binlog_pos_innodb`

This file is created by `mariadb-backup` to provide the binary log file name and position when the `--no-lock` option is used. It can be used instead of the `xtrabackup_binlog_info` file to obtain transactionally consistent binlog coordinates from the backup of a master server with the `--no-lock` option to minimize the impact on a running server.

Whenever a transaction is committed inside InnoDB when the binary log is enabled, the corresponding binlog coordinates are written to the InnoDB redo log along with the transaction commit. This allows one to restore the binlog coordinates corresponding to the last commit done by InnoDB along with a backup.

The limitation of using `xtrabackup_binlog_pos_innodb` with the `--no-lock` option is that no DDL or modification of non-transactional tables should be done during the backup. If the last event in the binlog is a DDL/non-transactional update, the coordinates in the file `xtrabackup_binlog_pos_innodb` are too old. But as long as only InnoDB updates are done during the backup, the coordinates are correct.

< 11.1

`xtrabackup_binlog_pos_innodb`

This file is created by `mariadb-backup` to provide the binary log file name and position when the `--no-lock` option is used. It can be used instead of the `xtrabackup_binlog_info` file to obtain transactionally consistent binlog coordinates from the backup of a master server with the `--no-lock` option to minimize the impact on a running server.

Whenever a transaction is committed inside InnoDB when the binary log is enabled, the corresponding binlog coordinates are written to the InnoDB redo log along with the transaction commit. This allows one to restore the binlog coordinates corresponding to the last commit done by InnoDB along with a backup.

The limitation of using `xtrabackup_binlog_pos_innodb` with the `--no-lock` option is that no DDL or modification of non-transactional tables should be done during the backup. If the last event in the binlog is a DDL/non-transactional update, the coordinates in the file `xtrabackup_binlog_pos_innodb` are too old. But as long as only InnoDB updates are done during the backup, the coordinates are correct.

CURRENT

`mariadb_backup_checkpoints`

The `xtrabackup_checkpoints` file contains metadata about the backup.

For example:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 1635102
last_lsn = 1635102
recover_binlog_info = 0
```

See below for a description of the fields.

If the `--extra-lsndir` option is provided, then an extra copy of this file are saved in that directory.

< 11.1

`xtrabackup_checkpoints`

The `xtrabackup_checkpoints` file contains metadata about the backup.

For example:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 1635102
last_lsn = 1635102
recover_binlog_info = 0
```

See below for a description of the fields.

If the `--extra-lsndir` option is provided, then an extra copy of this file are saved in that directory.

`backup_type`

If the backup is a non-prepared full backup or a non-prepared partial backup, then `backup_type` is set to `full-backuped`.

If the backup is a non-prepared incremental backup, then `backup_type` is set to `incremental`.

If the backup has already been prepared, then `backup_type` is set to `log-applied`.

`from_lsn`

If `backup_type` is `full-backuped`, then `from_lsn` has the value of `0`.

If `backup_type` is `incremental`, then `from_lsn` has the value of the log sequence number (LSN) at which the backup started reading from the InnoDB redo log. This is internally used by mariadb-backup when preparing incremental backups.

This value can be manually set during an incremental backup with the –incremental-lsn option. However, it is generally better to let mariadb-backup figure out the `from_lsn` automatically by specifying a parent backup with the `--incremental-basedir` option.

`to_lsn`

`to_lsn` has the value of the log sequence number (LSN) of the last checkpoint in the InnoDB redo log. This is internally used by mariadb-backup when preparing incremental backups.

`last_lsn`

`last_lsn` has the value of the last log sequence number (LSN) read from the InnoDB redo log. This is internally used by mariadb-backup when preparing incremental backups.

CURRENT

`mariadb_backup_info`

Contains information about the backup. The fields in this file are listed below.

If the `--extra-lsndir` option is provided, an extra copy of this file is saved in that directory.

< 11.1

`xtrabackup_info`

Contains information about the backup. The fields in this file are listed below.

If the `--extra-lsndir` option is provided, an extra copy of this file is saved in that directory.

`uuid`

If a UUID was provided by the –incremental-history-uuid option, then it are saved here. Otherwise, this is the empty string.

`name`

If a name was provided by the `--history` or the `---incremental-history-name` options, then it are saved here. Otherwise, this is the empty string.

`tool_name`

The name of the mariadb-backup executable that performed the backup. This is generally `mariadb-backup`.

`tool_command`

The arguments that were provided to mariadb-backup when it performed the backup.

`tool_version`

The version of mariadb-backup that performed the backup.

`ibbackup_version`

The version of mariadb-backup that performed the backup.

`server_version`

The version of MariaDB Server that was backed up.

`start_time`

The time that the backup started.

`end_time`

The time that the backup ended.

`lock_time`

The amount of time that mariadb-backup held its locks.

`binlog_pos`

This field stores the binary log file name and position that corresponds to the backup.

This field also stores the value of the `gtid_current_pos` system variable that correspond to the backup.

The values in this field are only guaranteed to be consistent with the backup if the `--no-lock` option was **not** provided when the backup was taken.

`innodb_from_lsn`

This is identical to `from_lsn` in `xtrabackup_checkpoints`.

If the backup is a full backup, then `innodb_from_lsn` has the value of `0`.

If the backup is an incremental backup, then `innodb_from_lsn` has the value of the log sequence number (LSN) at which the backup started reading from the InnoDB redo log.

`innodb_to_lsn`

This is identical to `to_lsn` in `xtrabackup_checkpoints`.

`innodb_to_lsn` has the value of the log sequence number (LSN) of the last checkpoint in the InnoDB redo log.

`partial`

If the backup is a partial backup, then this value are `Y`.

Otherwise, this value are `N`.

`incremental`

If the backup is an incremental backup, then this value are `Y`.

Otherwise, this value are `N`.

`format`

This field's value is the format of the backup.

If the `--stream` option was set to `xbstream`, then this value are `xbstream`.

If the `--stream` option was **not** provided, then this value are `file`.

`compressed`

If the `--compress` option was provided, then this value are `compressed`.

Otherwise, this value are `N`.

> **CURRENT**
>
> **`mariadb_backup_slave_info`**
>
> If the `--slave-info` option is provided, this file contains the `CHANGE MASTER` command that can be used to set up a new server as a slave of the original server's master after the backup has been restored.
>
> `mariadb-backup` does **not** check if GTIDs are being used in replication. It takes a shortcut and assumes that if the gtid_slave_pos system variable is non-empty, then it writes the `CHANGE MASTER` command with the `MASTER_USE_GTID` option set to `slave_pos`. Otherwise, it writes the `CHANGE MASTER` command with the `MASTER_LOG_FILE` and `MASTER_LOG_POS` options using the master's binary log file and position. See MDEV-19264 for more information.
>
> **< 11.1**
>
> **`xtrabackup_slave_info`**

If the `--slave-info` option is provided, this file contains the `CHANGE MASTER` command that can be used to set up a new server as a slave of the original server's master after the backup has been restored.

`mariadb-backup` does **not** check if GTIDs are being used in replication. It takes a shortcut and assumes that if the gtid_slave_pos system variable is non-empty, then it writes the `CHANGE MASTER` command with the `MASTER_USE_GTID` option set to `slave_pos`. Otherwise, it writes the `CHANGE MASTER` command with the `MASTER_LOG_FILE` and `MASTER_LOG_POS` options using the master's binary log file and position. See MDEV-19264 for more information.

CURRENT

`mariadb_backup_galera_info`

If the `--galera-info` option is provided, this file contains information about a Galera Cluster node's state.

The file contains the values of the wsrep_local_state_uuid and wsrep_last_committed status variables.

The values are written in the following format:

```
wsrep_local_state_uuid:wsrep_last_committed
```

For example:

```
d38587ce-246c-11e5-bcce-6bbd0831cc0f:1352215
```

< 11.1

`xtrabackup_galera_info`

If the `--galera-info` option is provided, this file contains information about a Galera Cluster node's state.

The file contains the values of the wsrep_local_state_uuid and wsrep_last_committed status variables.

The values are written in the following format:

```
wsrep_local_state_uuid:wsrep_last_committed
```

For example:

```
d38587ce-246c-11e5-bcce-6bbd0831cc0f:1352215
```

`<table>.delta`

If the backup is an incremental backup, this file contains changed pages for the table.

`<table>.delta.meta`

If the backup is an incremental backup, this file contains metadata about `<table>.delta` files. The fields in this file are listed below.

`page_size`

This field contains either the value of `innodb_page_size` or the value of the `KEY_BLOCK_SIZE` table option for the table if the `ROW_FORMAT` table option for the table is set to `COMPRESSED`.

`zip_size`

If the `ROW_FORMAT` table option for this table is set to `COMPRESSED`, this field contains the value of the compressed page size.

`space_id`

This field contains the value of the table's `space_id`.

This page is licensed: CC BY-SA / Gnu FDL

Using Encryption and Compression Tools With mariadb-backup

> **Note:**
>
> mariadb-backup was previously called mariabackup.

`mariadb-backup` supports streaming to stdout with the `--stream=xbstream` option. This option allows easy integration with popular encryption and compression tools. Below are several examples.

> **Note:**
>
> For a complete list of `mariadb-backup` options, see this page.
>
> For a detailed description of `mariadb-backup` functionality, see this page.

Encrypting and Decrypting Backup With openssl

The following example creates an AES-encrypted backup, protected with the password "mypass" and stores it in a file "backup.xb.enc":

```
mariadb-backup --user=root --backup --stream=xbstream  | openssl  enc -aes-256-cbc -k mypass >
backup.xb.enc
```

To decrypt and unpack this backup into the current directory, the following command can be used:

```
openssl  enc -d -aes-256-cbc -k mypass -in backup.xb.enc | mbstream -x
```

Compressing and Decompressing Backup With gzip

This example compresses the backup without encrypting:

```
mariadb-backup --user=root --backup --stream=xbstream | gzip > backupstream.gz
```

We can decompress and unpack the backup as follows:

```
gunzip -c backupstream.gz | mbstream -x
```

Compressing and Encrypting Backup, Using gzip and openssl

This example adds a compression step before the encryption, otherwise looks almost identical to the previous example:

```
mariadb-backup --user=root --backup --stream=xbstream | gzip | openssl  enc -aes-256-cbc -k mypass >
backup.xb.gz.enc
```

We can decrypt, decompress and unpack the backup as follow (note `gzip -d` in the pipeline):

```
openssl  enc -d -aes-256-cbc -k mypass -in backup.xb.gz.enc |gzip -d| mbstream -x
```

Compressing and Encrypting with 7Zip

7zip archiver is a popular utility (especially on Windows) that supports reading from standard output, with the `-si` option, and writing to stdout with the `-so` option, and can thus be used together with mariadb-backup.

Compressing backup with the 7z command line utility works as follows:

```
mariadb-backup --user=root --backup --stream=xbstream | 7z a -si backup.xb.7z
```

Uncompress and unpack the archive with

```
7z e backup.xb.7z -so |mbstream -x
```

7z also has builtin AES-256 encryption. To encrypt the backup from the previous example using password SECRET, add `-pSECRET` to the 7z command line.

Compressing with zstd

Compress

```
mariadb-backup --user=root --backup --stream=xbstream  | zstd - -o backup.xb.zst -f -1
```

Decompress, unpack

```
zstd -d backup.xbstream.zst -c | mbstream -x
```

Encrypting With GPG

Encryption

```
mariadb-backup --user=root --backup --stream=xbstream | gpg -c --passphrase SECRET --batch --yes -o
backup.xb.gpg
```

Decrypt, unpack

```
gpg --decrypt --passphrase SECRET --batch --yes  backup.xb.gpg | mbstream -x
```

Interactive Input for Passphrases

Most of the described tools also provide a way to enter a passphrase interactively (although 7zip does not seem to work well when reading input from stdin). Please consult documentation of the tools for more info.

Writing extra status files

By default files like xtrabackup_checkpoints are also written to the output stream only, and so would not be available for taking further incremental backups without prior extraction from the compressed or encrypted stream output file.

To avoid this these files can additionally be written to a directory that can then be used as input for further incremental backups using the –extra-lsndir=… option.

See also e.g: Combining incremental backups with streaming output

This page is licensed: CC BY-SA / Gnu FDL

Encrypted Files Backup (mariadb-backup)

This page addresses a major misconception: Many users assume that, because `mariadb-backup` reads encrypted files, it “unlocks” them. However, the encryption remains intact throughout the entire lifecycle, as explained in the following.

Compatibility with Data-at-Rest Encryption (TDE)

While this page primarily discusses using external tools like `openssl` to encrypt a backup stream, `mariadb-backup` is also fully compatible with MariaDB’s internal data-at-rest encryption (TDE) for InnoDB and Aria tables.

When backing up a server where encryption is enabled, there are specific behaviors and requirements during the backup, prepare, and restore phases.

1. The Backup Phase

When `mariadb-backup` performs a backup, it copies the physical files exactly as they exist on disk. If the files are encrypted at rest by the MariaDB server, the backup files remain encrypted.

- **No “double encryption” required:** You do not *need* to pipe the backup through `openssl` to secure it, as the data is already encrypted. However, you may still choose to do so for additional security or to encrypt non-InnoDB files (like configuration files or metadata).
- **Key Access:** `mariadb-backup` must be able to read the encrypted files. In order to do that, `mariadb-backup` queries the server as to what encryption plugin is used:

```
SELECT plugin_name, plugin_library, @@plugin_dir FROM information_schema.plugins WHERE
plugin_type='ENCRYPTION';
+---------------------+-----------------------+--------------------------------------+
| plugin_name         | plugin_library        | @@plugin_dir                         |
+---------------------+-----------------------+--------------------------------------+
| file_key_management | file_key_management.so | /opt/homebrew/opt/mariadb/lib/plugin/ |
+---------------------+-----------------------+--------------------------------------+
```

Once `mariadb-backup` knows that `file_key_management` is used, it queries the server for details, like the location of the key file, and which encryption algorithm is used:

```
SHOW VARIABLES LIKE 'file_key_management%';
+----------------------------------------+-------------------------------------------------+
| Variable_name                          | Value                                           |
+----------------------------------------+-------------------------------------------------+
| file_key_management_digest             | sha1                                            |
| file_key_management_encryption_algorithm | aes_ctr                                       |
| file_key_management_filekey            |                                                 |
| file_key_management_filename           | /opt/homebrew/etc/mysql/encryption/keyfile.txt |
| file_key_management_use_pbkdf2         | 0                                               |
+----------------------------------------+-------------------------------------------------+
```

`mariadb-backup` uses that information to calculate the checksums of `.ibd` files/pages, and to decrypt the redo log for further information.

2. The Prepare Phase

The “Prepare” phase is when `mariadb-backup` makes the data consistent by applying the redo logs.

- **Requirement:** To perform the prepare phase, the tool *must* have access to the same encryption keys used by the server at the time of the backup.
- **Persistence:** After the prepare phase is complete, the data files *remain encrypted*. The tool does not “decrypt” the database into plain text; it only ensures the encrypted pages are consistent.

3. The Restore Phase

The restore phase simply moves the encrypted files back to the target server's data directory.

> **Warning:**
>
> **CRITICAL LIMITATION:** A restore is **only** successful if the target server is configured with the **exact same keys** and KMS method used by the original server. If the target server has a different key or cannot load the key management plugin, the MariaDB service fails to start, or tables are marked as "corrupted" because the engine cannot decrypt the pages.

TDE Backup Requirements Checklist

To ensure a restorable backup of an encrypted MariaDB instance, you must:

1. **Access to encryption information:** Ensure that `mariadb-backup` has the same access to encryption information as the server does.
2. **Back up the keys:** The encryption keys themselves are **not** stored inside the backup. You must manually back up your `keyfile.txt`, AWS KMS credentials, or HashiCorp Vault tokens separately.
3. **Synchronize target keys:** Before running `mariadb-backup --copy-back`, verify that the destination server's configuration points to the identical keys used during the backup.

Key Takeaways

- **Data State:** Encrypted at source → Encrypted in backup → Encrypted at destination.
- **Prepare Phase:** Needs the key to read/apply logs, but writes encrypted data back out.
- **Restoration:** If you lose your keys, your backup is permanently unrecoverable.

Mixed-Encryption Backups

Mixed-encryption backups are a common scenario, especially in large production environments where a full "rotation" (encrypting all old data) can take days or weeks.

`mariadb-backup` is "smart" enough to handle a mixed-state tablespace.

When you run a backup on a mixed-encryption database, here is exactly what happens under the hood:

1. File-by-File Detection

`mariadb-backup` reads the *header* of every tablespace file (`.ibd` files). Each header contains metadata that tells the tool whether that specific file is encrypted or plain text.

- If the header says "Encrypted," the tool uses the configured KMS keys to read the pages.
- If the header says "Plain Text," it reads them normally.

2. Preservation of State

The backup is an exact physical replica. This means:

- **Table A (Encrypted)** stays encrypted in the backup.
- **Table B (Unencrypted)** stays unencrypted in the backup. The backup does not "standardize" the encryption; it respects the individual state of each table at the exact moment the backup started.

3. The Prepare Phase in a Mixed Environment

This is where the KMS[6] configuration is vital. During the "Prepare" phase (`--prepare`), `mariadb-backup` applies the Redo Log to the tables to ensure consistency.

- If a transaction in the Redo Log involves an *encrypted* table, the tool uses the keys to apply that change.

- If the transaction involves an *unencrypted* table, it applies it without keys.
- **The Result:** After the prepare phase, your backup remains in a mixed state, exactly matching the original source.

Key Limitations & Risks for Mixed Backups

Scenario	Behavior	Risk
Missing Keys	The backup fails or the `prepare` phase crashes when it hits the first encrypted table.	Even if 99% of your tables are unencrypted, 1 encrypted table requires the KMS to be active for the backup to be valid.
New Tables	If you enable encryption *during* a long-running backup, `mariadb-backup` handles it via the Redo Log.	Ensure the key used for the new table is the same one available to the backup tool.
Restore Target	You restore to a server without a KMS plugin.	The unencrypted tables work fine, but the encrypted tables are unreadable/corrupted.

> **Note:**
>
> **Note on Mixed Environments:** `mariadb-backup` supports databases containing a mix of encrypted and unencrypted tables. The tool detects the encryption status of each tablespace individually. However, if even a single table in the backup is encrypted, the **entire** `--prepare` process requires access to the Key Management System (KMS). Without valid keys, the tool cannot verify the consistency of encrypted tables, and the backup preparation fails.

Handling Mixed-Encryption Environments

In many production environments, a database may contain a mix of encrypted and unencrypted tables. This occurs frequently during a rolling migration to TDE or when only specific sensitive tables are targeted for encryption.

1. Tablespace Detection

`mariadb-backup` detects the encryption status of each tablespace individually by reading the file headers. It does not “force” encryption on unencrypted tables, nor does it decrypt encrypted ones. The backup is an exact physical replica of the source’s mixed state.

2. The Prepare Phase Requirement

Even if only one table in your database is encrypted, the **entire** `--prepare` phase requires the Key Management System (KMS) to be active and configured. The tool must be able to read the encryption keys to apply redo logs and ensure consistency across the entire dataset. If the KMS is unavailable, the prepare phase fails.

3. Best Practice: Layered Encryption

Because a mixed-state backup leaves unencrypted tables (and database metadata) vulnerable in the backup file, it is a recommended best practice to use **Layered Encryption**.

By piping the `mariadb-backup` stream through an external encryption tool (like OpenSSL), you ensure that:

- **Legacy/Unencrypted tables** are protected within the backup archive.
- **Metadata** (file names, schemas, and logs) is hidden.
- **Transport Security** is maintained regardless of the destination server’s TDE configuration.

Example: Encrypting a Mixed-State Backup Stream

```
mariadb-backup --backup --stream=xbstream | \ openssl enc -aes-256-cbc -k $BACKUP_PASSWORD >
full_secure_backup.xb.enc
```

> **Warning:**
>
> **Warning:** When using Layered Encryption, the restoration process requires **two** sets of credentials: the external password/key to decrypt the stream, and the MariaDB KMS keys to prepare and run the database. Loss of either set renders the backup unrecoverable.

Summary of TDE Backup Behavior

Component	State in Backup	Requires KMS for Prepare?
Encrypted InnoDB Table	Encrypted	**Yes**
Plaintext InnoDB Table	Plaintext	**Yes** (to initialize engine)
Aria System Tables	Encrypted (if configured)	**Yes**
Binary / Redo Logs	Encrypted (if configured)	**Yes**
Backup Metadata	Plaintext	No

This page is licensed: CC BY-SA / Gnu FDL

How mariadb-backup Works

> **Note:**
>
> mariadb-backup was previously called mariabackup.

This is a description of the different stages in `mariadb-backup`, what they do and why they are needed.

Execution Stages

Initialization Phase

- Connect to mysqld instance, find out important variables (`datadir`, InnoDB pagesize, encryption keys, encryption plugin etc)
- Scan the database directory, `datadir`, looking for InnoDB tablespaces, load the tablespaces (basically, it is an “open” in InnoDB sense)
- If `--lock-ddl-per-table` is used:
  - Do MDL locks, for InnoDB tablespaces that we want to copy. This is to ensure that there are no `ALTER`, `RENAME` , `TRUNCATE` or `DROP TABLE` on any of the tables that we want to copy.
  - This is implemented with:

```
BEGIN
FOR EACH affected TABLE
SELECT 1 FROM <TABLE> LIMIT 0
```

- If lock-ddl-per-table is not done, then mariadb-backup would have to know all tables that were created or altered during the backup. See MDEV-16791.

Redo Log Handling

Start a dedicated thread in mariadb-backup to copy InnoDB redo log (`ib_logfile*`).

- This is needed to record all changes done while the backup is running. (The redo log logically is a single circular file, split into `innodb_log_files_in_group` files.)
- The log is also used to see detect if any truncate or online alter tables are used.
- The assumption is that the copy thread are able to keep up with server. It should always be able keep up, if the redo log is big enough.

Copy-phase for InnoDB Tablespaces

- Copy all selected tablespaces, file by file, in dedicated threads in mariadb-backup without involving the mysqld server.
- This is special “careful” copy, it looks for page-level consistency by checking the checksum.
- The files are not point-in-time consistent as data may change during copy.
- The idea is that InnoDB recovery would make it point-in-time consistent.

Create a Consistent Backup Point

- Execute `FLUSH TABLE WITH READ LOCK`. This is default, but may be omitted with the `--no-lock` parameter. The reason why `FLUSH` is needed is to ensure that all tables are in a consistent state at the exact same point in time, independent of storage engine.
- If `--lock-ddl-per-table` is used and there is a user query waiting for MDL, the user query are killed to resolve a deadlock. Note that these are only queries of type `ALTER`, `DROP`, `TRUNCATE` or RENAME `TABLE`. (MDEV-15636)

Last Copy Phase

- Copy `.frm`, `MyISAM`, `Aria` and other storage engine files.
- If `MyRocks` is used, create rocksdb checkpoint via the `set rocksdb_create_checkpoint=$rocksdb_data_dir/mariadb-backup_rocksdb_checkpoint` command. The result of it is a directory with hardlinks to MyRocks files. Copy the checkpoint directory to the backup (or create hardlinks in backup directory is on the same partition as data directory). Remove the checkpoint directory.
- Copy tables that were created while the backup was running and do rename files that were changed during backup (since MDEV-16791).
- Copy the rest of InnoDB redo log, stop redo-log-copy thread.
- Write some metadata info (binlog position).

Release Locks

- If `FLUSH TABLE WITH READ LOCK` was done:
  - execute: `UNLOCK TABLES`
- If `--lock-ddl-per-table` was done:
  - execute `COMMIT`

Handle Log Tables (TODO)

- If log tables exists:
  - Take MDL lock for log tables
  - Copy part of log tables that wasn’t copied before
  - Unlock log tables

Notes

- If `FLUSH TABLE WITH READ LOCK` is not used, only InnoDB tables are consistent (not the privilege tables in the mysql database or the binary log). The backup point depends on the content of the redo log within the backup itself.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-backup and BACKUP STAGE

> **Note:**
>
> mariadb-backup was previously called mariabackup.

The BACKUP STAGE statements make it possible to make an efficient external backup tool. How `mariadb-backup` uses these statements depends on whether you are using the version that is bundled with MariaDB Community Server or the version that is bundled with MariaDB Enterprise Server.

> **Note:**
>
> For a complete list of `mariadb-backup` options, see this page.
>
> For a detailed description of `mariadb-backup` functionality, see this page.

`BACKUP STAGE` in MariaDB Community Server

The BACKUP STAGE statements are supported. However, the version of `mariadb-backup` that is bundled with MariaDB Community Server does not yet use the `BACKUP STAGE` statement in the most efficient way. `mariadb-backup` simply executes the following `BACKUP STAGE` statement to lock the database:

```
BACKUP STAGE START;
BACKUP STAGE BLOCK_COMMIT;
```

When the backup is complete, it executes the following `BACKUP STAGE` statement to unlock the database:

```
BACKUP STAGE END;
```

> **Note:**
>
> To use a version of `mariadb-backup` that uses the BACKUP STAGE statements in the most efficient way, use MariaDB Backup bundled with MariaDB Enterprise Server.

Tasks Performed Prior to `BACKUP STAGE` in MariaDB Community Server

- Copy some transactional tables.
  - InnoDB (i.e. `ibdataN` and file extensions `.ibd` and `.isl`)
- Copy the tail of some transaction logs.
  - The tail of the InnoDB redo log (i.e. `ib_logfileN` files) are copied for InnoDB tables.

`BACKUP STAGE START` in MariaDB Community Server

`mariadb-backup` from MariaDB Community Server does not perform any tasks in the `START` stage.

`BACKUP STAGE FLUSH` in MariaDB Community Server

`mariadb-backup` from MariaDB Community Server does not currently perform any tasks in the `FLUSH` stage.

`BACKUP STAGE BLOCK_DDL` in MariaDB Community Server

`mariadb-backup` from MariaDB Community Server does not currently perform any tasks in the `BLOCK_DDL` stage.

`BACKUP STAGE BLOCK_COMMIT` in MariaDB Community Server

`mariadb-backup` from MariaDB Community Server performs the following tasks in the `BLOCK_COMMIT` stage:

- Copy other files.

    - i.e. file extensions `.frm`, `.isl`, `.TRG`, `.TRN`, `.opt`, `.par`
- Copy some transactional tables.
    - Aria (i.e. `aria_log_control` and file extensions `.MAD` and `.MAI`)
- Copy the non-transactional tables.
    - `MyISAM` (i.e. file extensions `.MYD` and `.MYI`)
    - `MERGE` (i.e. file extensions `.MRG`)
    - `ARCHIVE` (i.e. file extensions `.ARM` and `.ARZ`)
    - `CSV` (i.e. file extensions `.CSM` and `.CSV`)
- Create a MyRocks checkpoint using the `rocksdb_create_checkpoint` system variable.
- Copy the tail of some transaction logs.
    - The tail of the InnoDB redo log (i.e. `ib_logfileN` files) are copied for InnoDB tables.
- Save the binary log position to `xtrabackup_binlog_info`.
- Save the Galera Cluster state information to `xtrabackup_galera_info`.

`BACKUP STAGE END` in MariaDB Community Server

`mariadb-backup` from MariaDB Community Server performs the following tasks in the `END` stage:

- Copy the MyRocks checkpoint into the backup.

`BACKUP STAGE` in MariaDB Enterprise Server

The following sections describe how the MariaDB Backup version of mariadb-backup that is bundled with MariaDB Enterprise Server uses each BACKUP STAGE statement in an efficient way.

`BACKUP STAGE START` in MariaDB Enterprise Server

`mariadb-backup` from MariaDB Enterprise Server performs the following tasks in the `START` stage:

- Copy all transactional tables.
    - InnoDB (i.e. `ibdataN` and file extensions `.ibd` and `.isl`)
    - Aria (i.e. `aria_log_control` and file extensions `.MAD` and `.MAI`)
- Copy the tail of all transaction logs.
    - The tail of the InnoDB redo log (i.e. `ib_logfileN` files) are copied for InnoDB tables.
    - The tail of the Aria redo log (i.e. `aria_log.N` files) are copied for Aria tables.

`BACKUP STAGE FLUSH` in MariaDB Enterprise Server

mariadb-backup from MariaDB Enterprise Server performs the following tasks in the `FLUSH` stage:

- Copy all non-transactional tables that are not in use. This list of used tables is found with `SHOW OPEN TABLES`.
    - `MyISAM` (i.e. file extensions `.MYD` and `.MYI`)
    - `MERGE` (i.e. file extensions `.MRG`)
    - `ARCHIVE` (i.e. file extensions `.ARM` and `.ARZ`)
    - `CSV` (i.e. file extensions `.CSM` and `.CSV`)
- Copy the tail of all transaction logs.
    - The tail of the InnoDB redo log (i.e. `ib_logfileN` files) are copied for InnoDB tables.
    - The tail of the Aria redo log (i.e. `aria_log.N` files) are copied for Aria tables.

`BACKUP STAGE BLOCK_DDL` in MariaDB Enterprise Server

`mariadb-backup` from MariaDB Enterprise Server performs the following tasks in the `BLOCK_DDL` stage:

- Copy other files.
  - i.e. file extensions `.frm`, `.isl`, `.TRG`, `.TRN`, `.opt`, `.par`
- Copy the non-transactional tables that were in use during `BACKUP STAGE FLUSH`.
  - `MyISAM` (i.e. file extensions `.MYD` and `.MYI`)
  - `MERGE` (i.e. file extensions `.MRG`)
  - `ARCHIVE` (i.e. file extensions `.ARM` and `.ARZ`)
  - `CSV` (i.e. file extensions `.CSM` and `.CSV`)
- Check `ddl.log` for DDL executed before the `BLOCK DDL` stage.
  - The file names of newly created tables can be read from `ddl.log`.
  - The file names of dropped tables can also be read from `ddl.log`.
  - The file names of renamed tables can also be read from `ddl.log`, so the files can be renamed instead of re-copying them.
- Copy changes to system log tables.
  - mysql.general_log
  - mysql.slow_log
  - This is easy as these are append only.
- Copy the tail of all transaction logs.
  - The tail of the InnoDB redo log (i.e. `ib_logfileN` files) are copied for InnoDB tables.
  - The tail of the Aria redo log (i.e. `aria_log.N` files) are copied for Aria tables.

`BACKUP STAGE BLOCK_COMMIT` in MariaDB Enterprise Server

`mariadb-backup` from MariaDB Enterprise Server performs the following tasks in the `BLOCK_COMMIT` stage:

- Create a MyRocks checkpoint using the rocksdb_create_checkpoint system variable.
- Copy changes to system log tables.
  - mysql.general_log
  - mysql.slow_log
  - This is easy as these are append only.
- Copy changes to statistics tables.
  - mysql.table_stats
  - mysql.column_stats
  - mysql.index_stats
- Copy the tail of all transaction logs.
  - The tail of the InnoDB redo log (i.e. `ib_logfileN` files) are copied for InnoDB tables.
  - The tail of the Aria redo log (i.e. `aria_log.N` files) are copied for Aria tables.
- Save the binary log position to `xtrabackup_binlog_info`.
- Save the Galera Cluster state information to `xtrabackup_galera_info`.

`BACKUP STAGE END` in MariaDB Enterprise Server

`mariadb-backup` from MariaDB Enterprise Server performs the following tasks in the `END` stage:

- Copy the MyRocks checkpoint into the backup.

This page is licensed: CC BY-SA / Gnu FDL

Point-In-Time Recovery (InnoDB Log Archiving)

> **Note:**
>
> This functionality is available from MariaDB 13.0.

When InnoDB log archiving is enabled, the server retains a continuous history of write-ahead log records across multiple `ib_lsn.log` files. You can use this history to perform point-in-time recovery (PITR) to a specific Log Sequence Number (LSN), by replaying the archived InnoDB write-ahead logs at server startup.

This is an alternative to PITR via mariadb-backup and binary logs, useful in InnoDB-only deployments or in recovery scenarios where binary logs are not available.

> **Warning:**
>
> No shipped backup tool yet generates or restores backups in the `innodb_log_archive=ON` format — `mariadb-backup` does not support it and fails when the server is running with `innodb_log_archive=ON`. A backup tool that uses this format is being worked on. Until it ships, this procedure assumes you already have an externally-prepared restore that contains a consistent set of `ib_lsn.log` files alongside the InnoDB data files.

Recovery Parameters

Two startup parameters define the range of the log replay:

- `innodb_log_recovery_start` — the LSN at which recovery begins. Set this to the end LSN of the previous restore when applying an incremental restore. The default `0` starts from the latest completed checkpoint, which is guaranteed to live in one of the last two `ib_lsn.log` files.
- `innodb_log_recovery_target` — the LSN at which recovery ends (the recovery point objective). When this is non-zero, persistent InnoDB tables become read-only and no log writes are allowed during the recovery session. The default `0` replays the log to its end.

You can identify candidate LSNs from the `Innodb_lsn_archived` status variable on the source server, and from the file names in the data directory (each `ib_lsn.log` file name encodes the LSN at file offset `0x3000`).

The data files in the restore must correspond to an LSN that lies between `Innodb_lsn_archived` (the first checkpoint in the first archived log file) and the end LSN of the last archived log file. To extend the available recovery range, copy additional archived log files in from the source server — no further copying of data files is required.

Procedure

> **Step 1**
>
> **Prepare the restore.**
>
> Place the restored InnoDB data files and the corresponding `ib_lsn.log` archive files in the data directory. With `innodb_log_archive=ON`, the server refuses to start if a legacy `ib_logfile0` exists, so make sure it is not present.

> **Step 2**
>
> **Stop the MariaDB Server (if it is running).**

```
$ sudo systemctl stop mariadb
```

Step 3

Start the server with recovery parameters.

Pass `innodb_log_recovery_start` and `innodb_log_recovery_target` either on the command line or in the configuration file, alongside `innodb_log_archive=ON`. While `innodb_log_recovery_target` is non-zero, the server replays the archived log up to the target LSN and then keeps the InnoDB tables read-only.

Step 4

Verify the state.

Once the server reaches the target LSN, the recovery process stops. Inspect `Innodb_lsn_archived`, `Innodb_lsn_current`, and the contents of the recovered tables to confirm the data is at the expected logical point.

Step 5

Resume normal operation.

To return to normal read-write operation, restart the server without `innodb_log_recovery_target` set.

Warning:

No data file may carry an LSN newer than `innodb_log_recovery_target`. Crash recovery's role is to bring every database page to the same LSN. If any data file is newer than the target, recovery completes in an inconsistent state where some pages carry an LSN past the target — the database is corrupted. The server cannot validate every such impossible target, and the resulting corruption may not surface until the affected pages are accessed (see MDEV-34830). See `innodb_log_recovery_target` for details.

Note:

Point-in-time recovery of DDL operations is limited even for an InnoDB-only deployment, because part of the data dictionary is stored in `.frm` files whose creation is not covered by the InnoDB write-ahead log.

See Also

- InnoDB Log Archiving — feature overview, file format, monitoring, and managing archived log files.
- `innodb_log_archive`, `innodb_log_recovery_start`, `innodb_log_recovery_target`.
- Point-In-Time Recovery (PITR, mariadb-backup) — the binary-log-based PITR procedure.

This page is licensed: CC BY-SA / Gnu FDL

Replication as a Backup Solution

Replication can be used to support the backup strategy.

Important:

Replication alone is *not* sufficient for backup. It assists in protecting against hardware failure on the primary server, but does not protect against data loss. An accidental or malicious `DROP DATABASE` or `TRUNCATE TABLE` statement are replicated onto the replica as well. Care needs to be taken to prevent data getting out of sync between the primary and the replica.

Note:

The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Replication is most commonly used to support backups as follows:

- A primary server replicates to a replica
- Backups are then run off the replica without any impact on the primary.

Backups can have a significant effect on a server, and a high-availability primary may not be able to be stopped, locked or simply handle the extra load of a backup. Running the backup from a replica has the advantage of being able to shutdown or lock the replica and perform a backup without any impact on the primary server.

Note that when backing up off a replica server, it is important to ensure that the servers keep the data in sync. See for example Replication and Foreign Keys for a situation when identical statements can result in different data on a replica and a primary.

Setting Up a Dedicated Backup Replica

To set up a replica specifically for backup purposes, you will need to configure outbound replication from your primary server to the replica. Here is a step-by-step guide to establishing this connection securely:

1. Create a User for Replication

On the **primary server**, create a dedicated user for replication and grant it the necessary privileges:

```
CREATE USER 'backup_replica'@'replica_ip_or_hostname' IDENTIFIED BY 'strong_password';
GRANT REPLICATION SLAVE ON *.* TO 'backup_replica'@'replica_ip_or_hostname';
```

(Optional: Confirm the grants by executing `SHOW GRANTS FOR 'backup_replica'@'replica_ip_or_hostname';`)

2. Obtain the GTID Position

On the primary server, obtain the current GTID position from which the replica should start replicating. If you want to start from the most recent transaction, query the `gtid_current_pos`:

```
SHOW GLOBAL VARIABLES LIKE 'gtid_current_pos';
```

3. Configure the GTID Position on the Replica

On the replica server, configure the starting GTID position using the value obtained in the previous step:

SQL

```
SET GLOBAL gtid_slave_pos='<gtid_value_from_primary>';
```

4. Configure Replication

On the replica server, configure the connection to the primary server using the `CHANGE MASTER TO` statement:

```
CHANGE MASTER TO
   MASTER_HOST='primary_domain_or_ip',
   MASTER_PORT=3306,
   MASTER_USER='backup_replica',
   MASTER_PASSWORD='strong_password',
   MASTER_SSL=1,
   MASTER_USE_GTID=slave_pos;
```

(Note: Adjust the `MASTER_PORT`, `MASTER_SSL`, and add `MASTER_SSL_CA` parameters as necessary depending on your network and security configuration).

5. Start Replication

On the replica server, start the replication process:

```
START REPLICA;
```

6. Check Replication Status

On the replica server, verify that replication is running smoothly:

```
SHOW REPLICA STATUS \G
```

Ensure that both `Slave_IO_Running` and `Slave_SQL_Running` are `Yes`. Once the replica is fully synced, you can safely pause it or run your backup tools (`mariadb-backup` or `mariadb-dump`) directly against this replica without affecting the primary server's performance.

See Also

- Replication
- Replication Compatibility
- Backup & Restore

This page is licensed: CC BY-SA / Gnu FDL

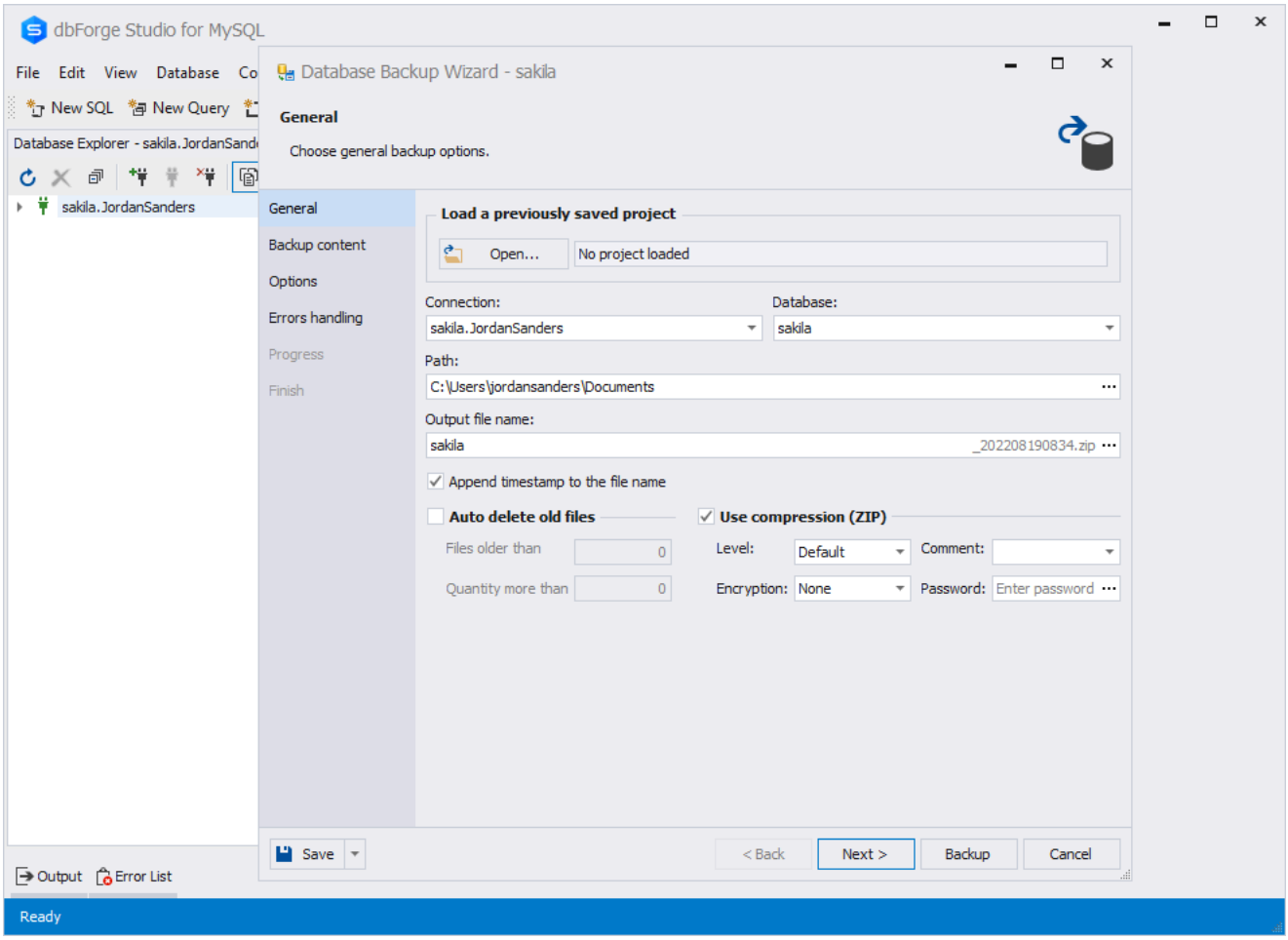
Backup and Restore via dbForge Studio

> **Note:**
>
> dbForge Studio is proprietary third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Content contributed by Devart; refer to its own documentation and license terms.

In the modern world, data importance is non-negotiable, and keeping data integrity and consistency is the top priority. Data stored in databases is vulnerable to system crashes, hardware problems, security breaches, and other failures causing data loss or corruption. To prevent database damage, it is important to back the data up regularly and implement the data restore policies.
MariaDB, one of the most popular database management systems, provides several methods to configure routines for backing up and recovering data. The current guideline illustrates both processes performed with the help of dbForge Studio for MySQL which is also a fully-functional GUI client for MariaDB that has everything you need to accomplish the database-related tasks on MariaDB.

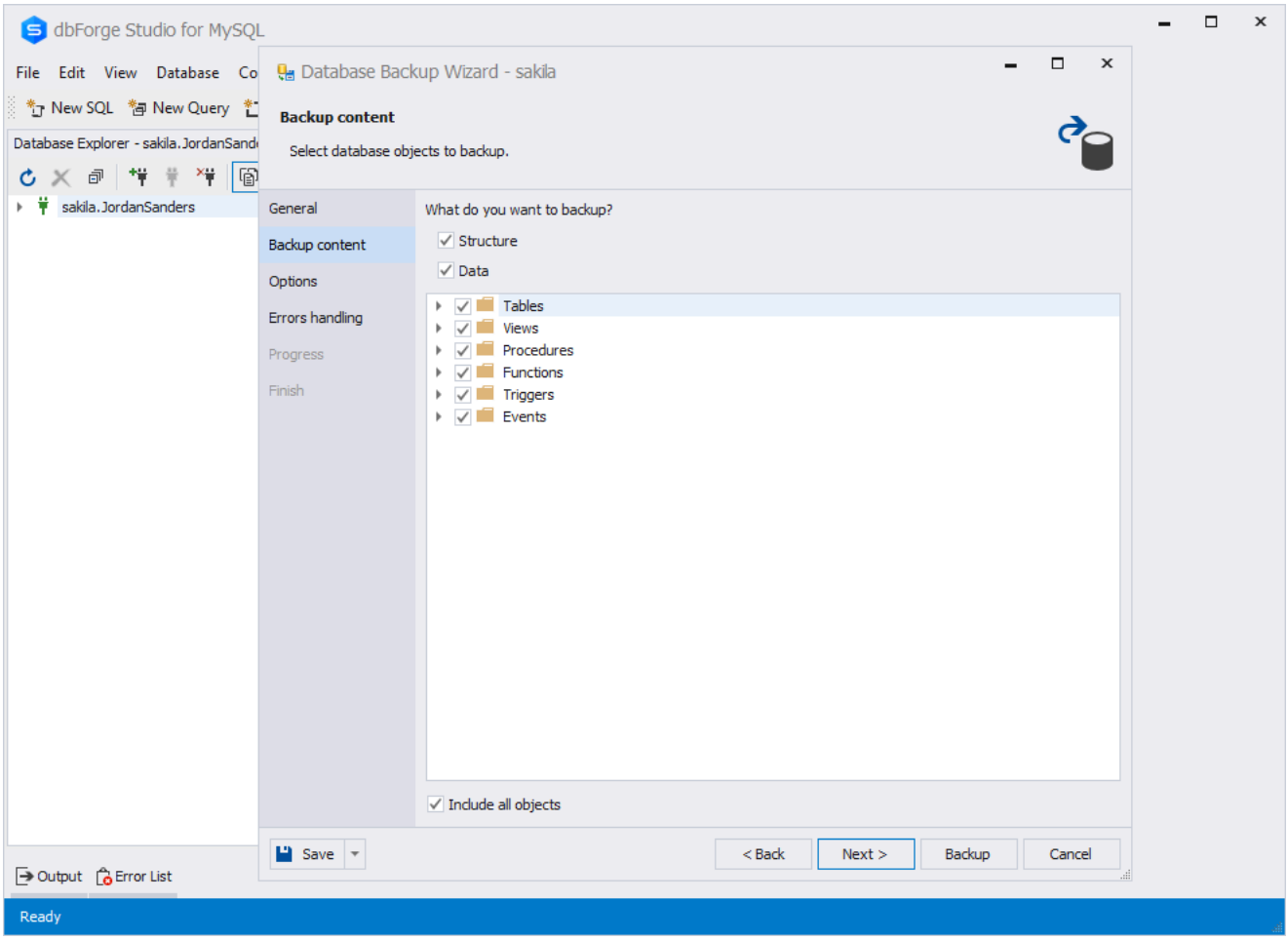
Create the backup on MariaDB

dbForge Studio for MySQL and MariaDB has a separate module dedicated to the data backing up and recovering jobs. Let us first look at how set the tool to create a MariaDB backup.
Launch the Studio and go to **Database > Backup and Restore > Backup Database**. The **Database Backup Wizard** with several pages will appear.
On the General page, specify the database in question and how to connect to it, then choose where to save the created backup file, and specify its name. There are additional optional settings – you can select to delete old files automatically, zip the output backup file, etc. When done, click **Next**.



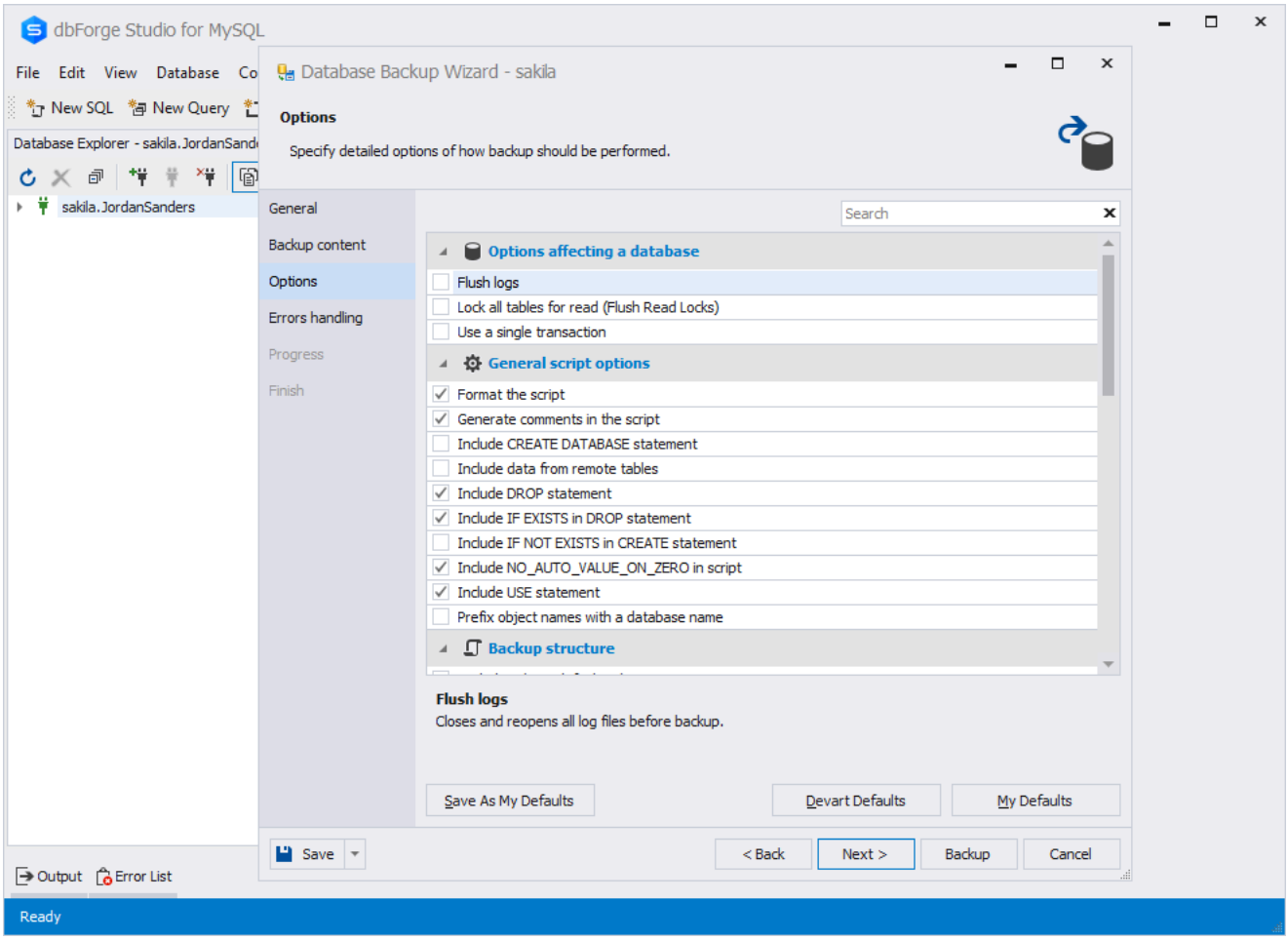
b1

On the **Backup content** page, select the objects to back up. Click **Next**.



b2

The **Options** page. Here you can specify the details of the data backing up process. Plenty of available options allow you to configure this task precisely to meet the specific requirements. When done, click **Next**.



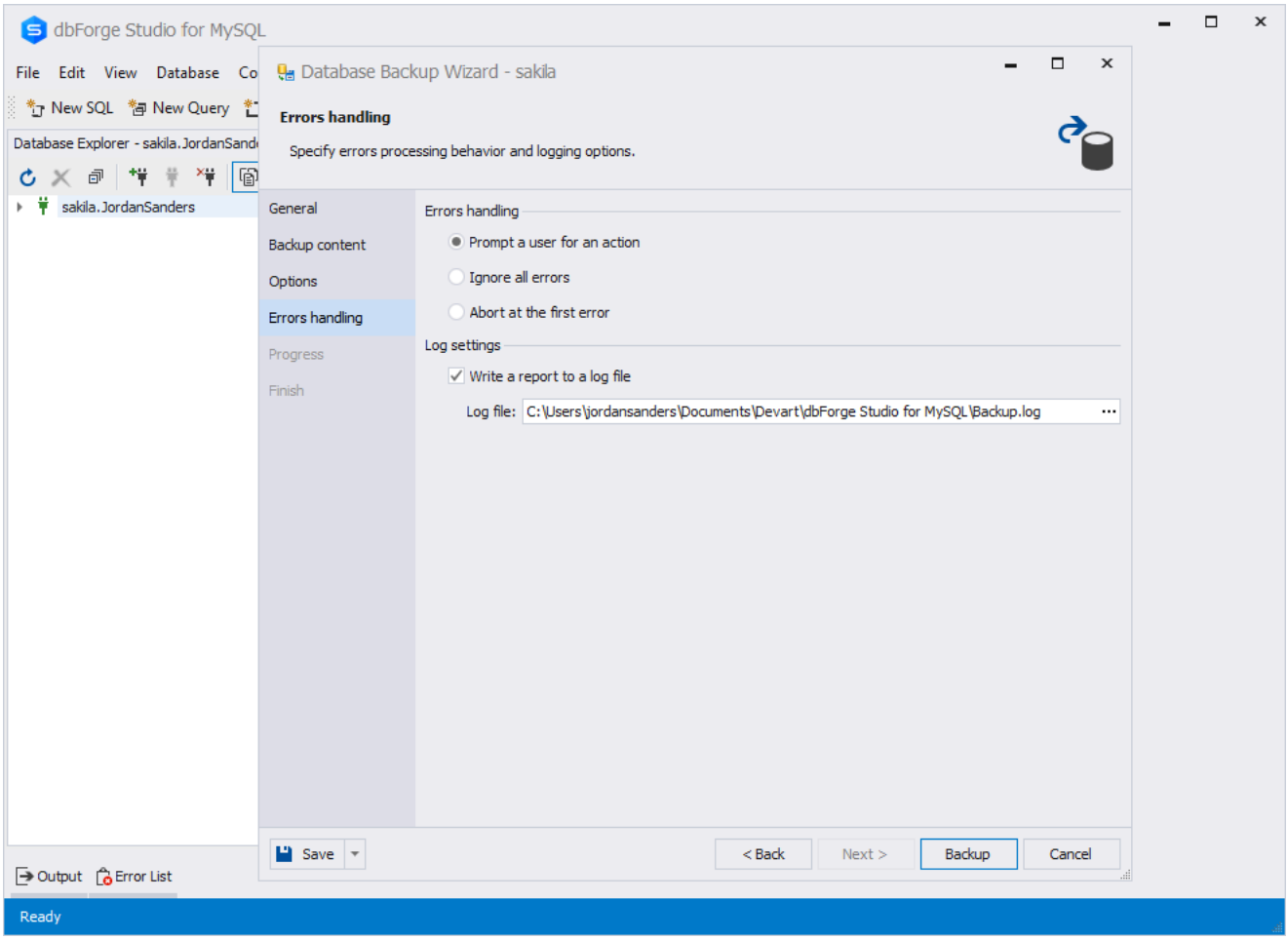
b3

The **Errors handling** page. Here you configure how the Studio should handle the errors that might occur during the backing up process. Also, you can set the Studio to write the information about the errors it encountered into the log file.

You can save the project settings to apply them in the future. For this, in the left bottom corner of the Wizard, select one of the saving options: **Save Project** or **Save Command Line**. The latter allows saving settings as a backup script which you can execute from the command line at any time later.

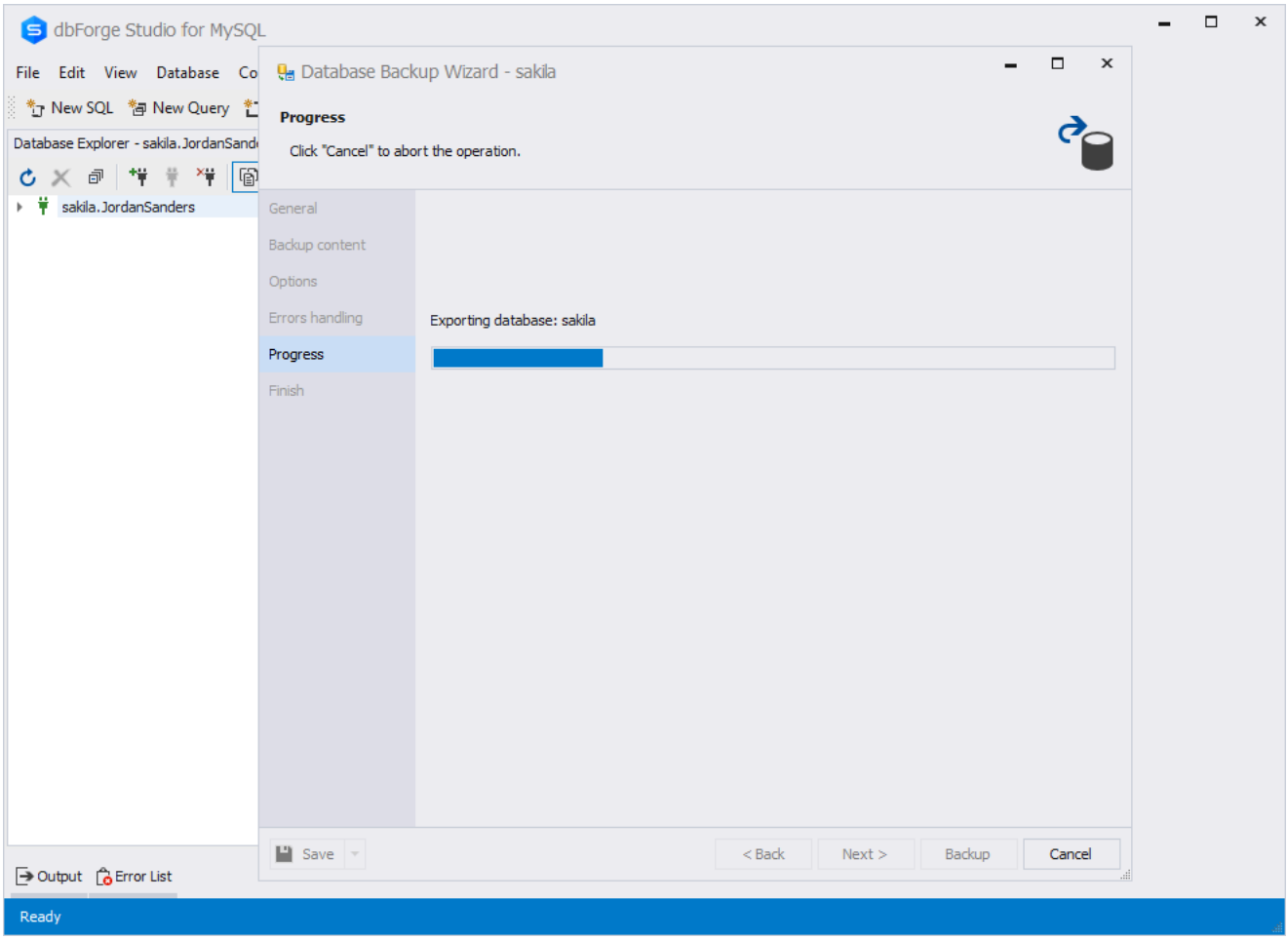
The configuration process is complete. Click Backup to launch the data backing up.

Note: It is not obligatory to go through all the pages of the Wizard. The Backup button is available no matter on which page you are. Thus, you can launch the process of backing the data up whenever you have set everything you needed.



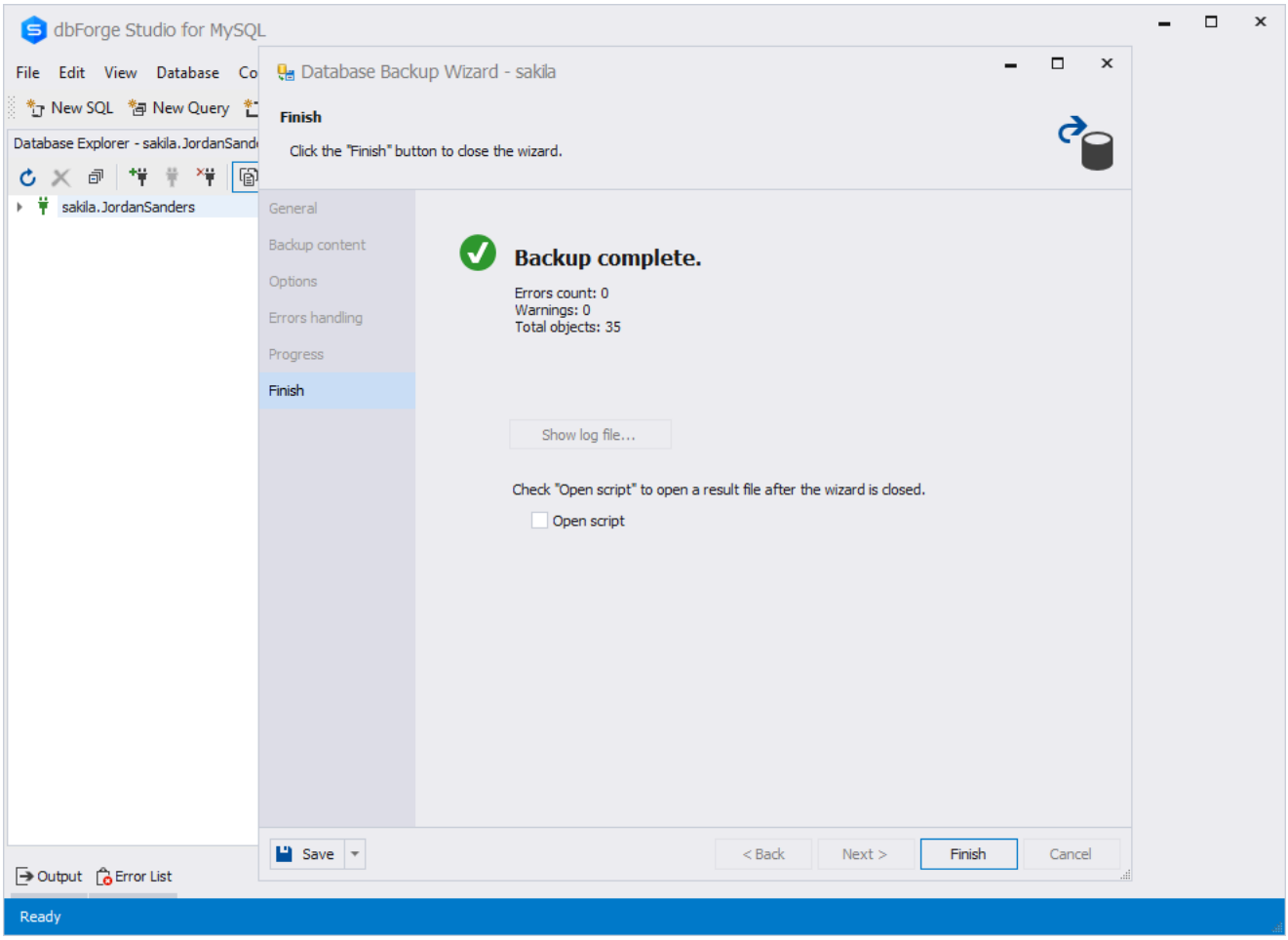
b4

After you have clicked **Backup**, dbForge Studio for MySQL starts to create a MariaDB backup.



b5

When this is done, you will see the confirmation message. Click **Finish**.



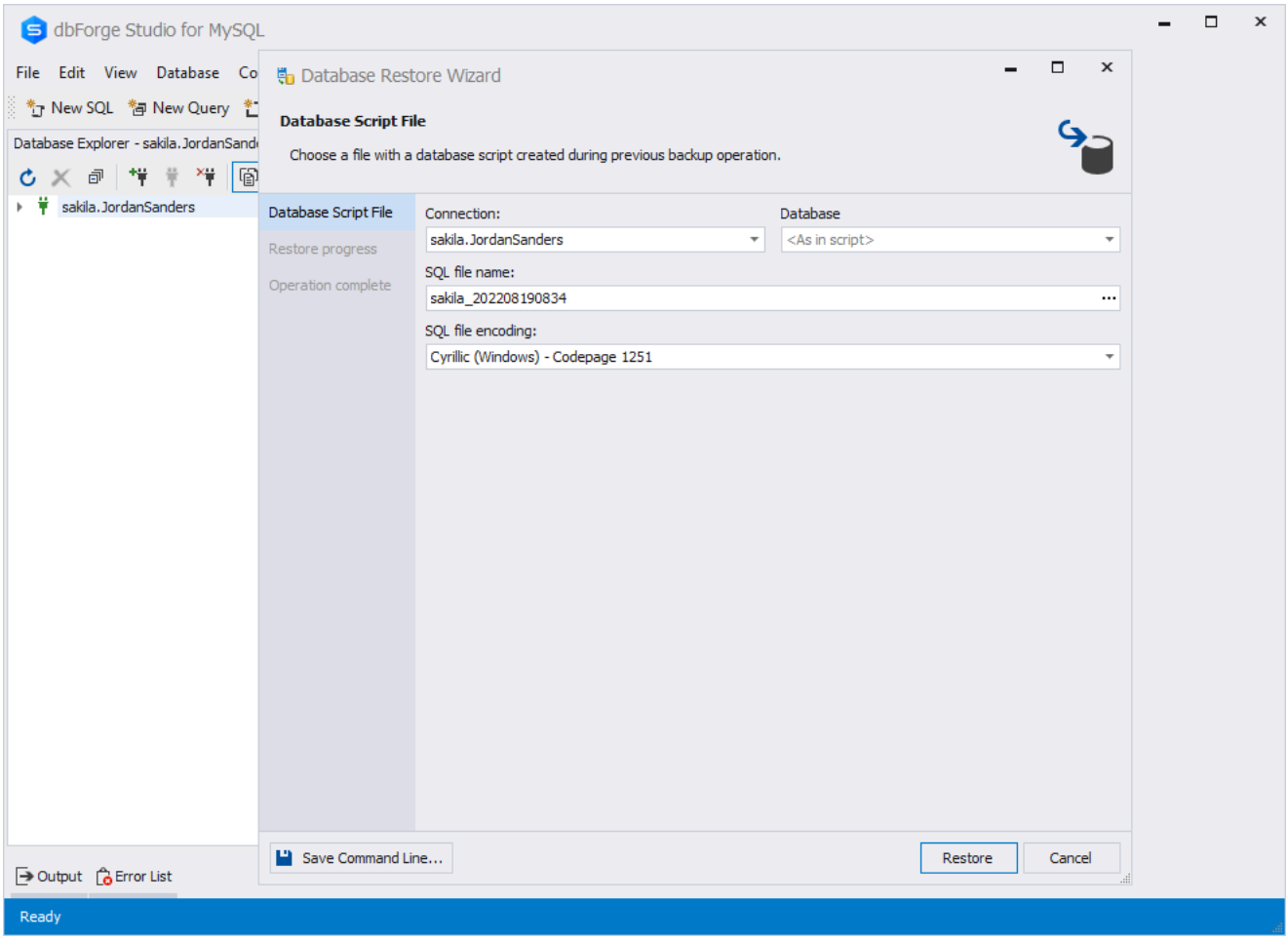
b6

Backup and restore policies suggest creating regular backups on a daily, weekly, monthly, quarterly, and yearly basis. Besides, to minimize the consequences of possible data loss, it is highly recommended make a backup before making any changes to a database, such as upgrading, modifying data, redesigning the structure, etc. Simply speaking, you always need a fresh backup to restore the most up-to-date database version.
To ensure regular backups on schedule, you can use a batch file created with the help of the Studio and **Windows Task Scheduler**, where you need to create and schedule the backup task.

Restore the backup file on MariaDB

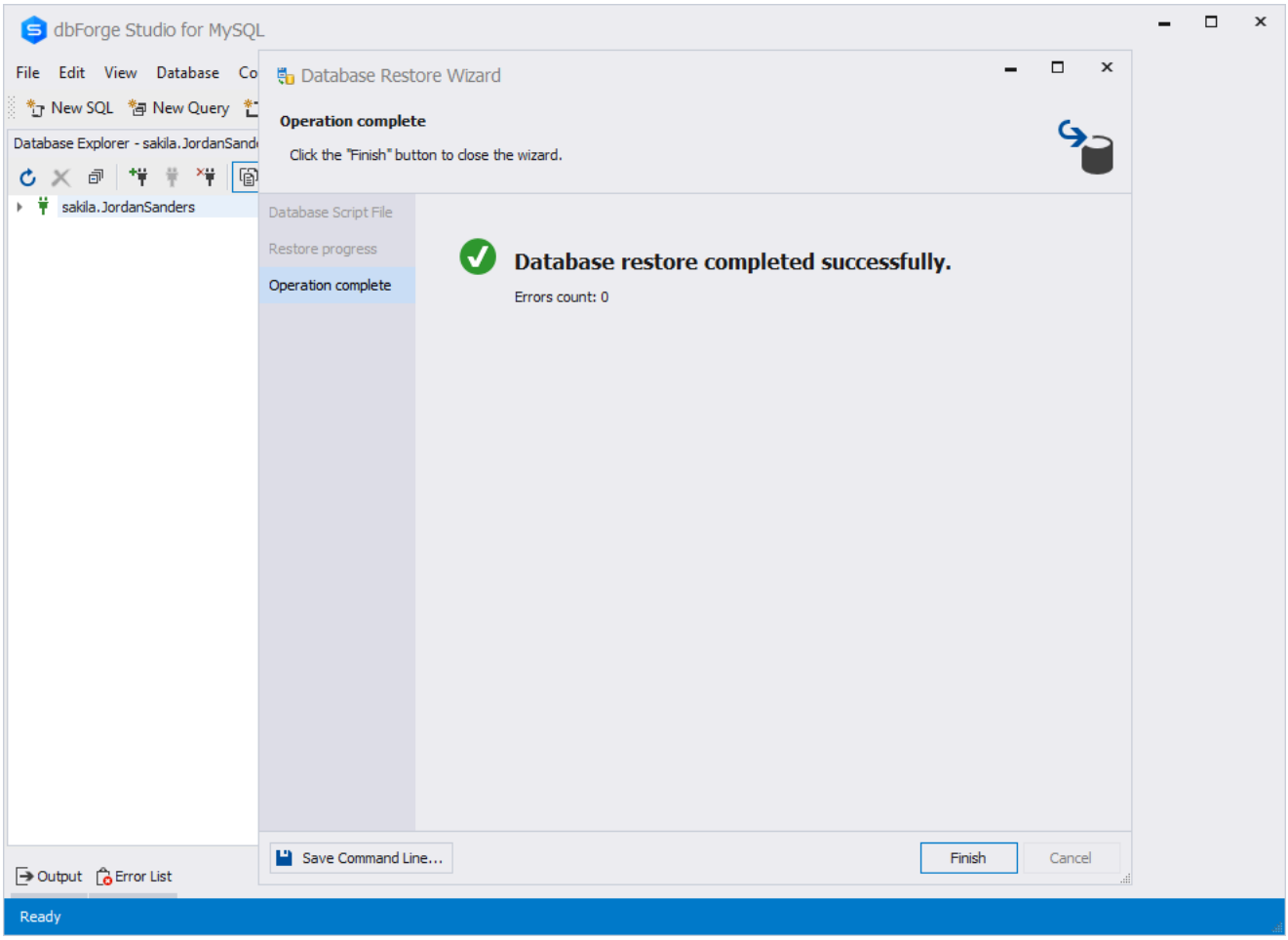
This is an even faster task, done in half as many steps.

The process of data recovery from the backup file is simple. It only takes several clicks:
Launch dbForge Studio for MySQL and go to **Database > Backup and Restore > Restore Database**. The **Database Restore Wizard** will appear.
Specify the database name, its connection parameters, and the path to the backup file you want to restore. Then click **Restore**, and the process will start immediately.



r1

When the process is complete, click **Finish**.



r2

More information about this essential feature is available on the dedicated backup and restore page – it explores the routines performed on MySQL, but they fully apply to MariaDB backups. You can use the same IDE and the same workflow.

To test-drive this and other features of the Studio (the IDE includes all the tools necessary for the development, management, and administration of databases on MariaDB), download dbForge Studio for a free 30-day trial. dbForge Studio for MySQL and MariaDB boasts truly advanced functionality that will help your teams deliver more value.

This page is licensed: CC BY-SA / Gnu FDL

Tables

See Copying Tables Between Databases and Servers.

This guide explains various methods for copying tables between MariaDB databases and servers, including using `FLUSH TABLES FOR EXPORT` and `mysqldump`.

See ALTER TABLE.

Complete `ALTER TABLE` guide for MariaDB. Complete syntax for modifying columns, indexes, constraints, and table properties with comprehensive examples.

See CREATE TABLE.

Complete guide to creating tables in MariaDB. Complete `CREATE TABLE` syntax for data types, constraints, indexes, and storage engines for production use.

See DROP TABLE.

Complete `DROP TABLE` syntax: `TEMPORARY`, `IF EXISTS`, `WAIT/NOWAIT`, `RESTRICT`/`CASCADE` options, metadata locks, atomic `DROP`, and replication behavior.

See Data Types.

Comprehensive MariaDB data types reference. Complete guide for numeric, string, date/time, spatial, and `JSON` types with storage specifications.

See Storage Engines.

Understand MariaDB Server's storage engines. Explore the features and use cases of InnoDB, Aria, MyISAM, and other engines to choose the best option for your specific data needs.

> **Note:**
>
> The task-oriented table tutorials now live in the Quickstart Guides:
>
> - Altering Tables
> - Getting Started with Indexes
> - Creating & Using Views

This page is licensed: CC BY-SA / Gnu FDL

Copying Tables Between Databases and Servers

With MariaDB it's very easy to copy tables between different MariaDB databases and different MariaDB servers. This works for tables created with the Archive, Aria, CSV, InnoDB, MyISAM, MERGE, and XtraDB engines.

The normal procedures to copy a table is:

```
FLUSH TABLES db_name.table_name FOR EXPORT

# Copy the relevant files associated with the table

UNLOCK TABLES;
```

The table files can be found in datadir/databasename (you can execute `SELECT @@datadir` to find the correct directory).
When copying the files, you should copy all files with the same table_name + various extensions. For example, for an Aria table of name foo, you will have files foo.frm, foo.MAI, foo.MAD and possibly foo.TRG if you have triggers.

If one wants to distribute a table to a user that doesn't need write access to the table and one wants to minimize the storage size of the table, the recommended engine to use is Aria or MyISAM as one can pack the table with aria_pack or myisampack respectively to make it notablly smaller. MyISAM is the most portable format as it's not dependent on whether the server settings are different. Aria and InnoDB require the same block size on both servers.

Copying Tables When the MariaDB Server is Down

The following storage engines support export without `FLUSH TABLES ... FOR EXPORT`, assuming the source server is down and the receiving server is not accessing the files during the copy.

Engine	Comment
Archive	

Engine	Comment
Aria	Requires clean shutdown. Table will automatically be fixed on the receiving server if aria_chk –zerofill was not run. If aria_chk –zerofill is run, then the table is immediately usable without any delays
CSV	
MyISAM	
MERGE	.MRG files can be copied even while server is running as the file only contains a list of tables that are part of merge.

Copying Tables Live From a Running MariaDB Server

For all of the above storage engines (Archive, Aria, CSV, MyISAM and MERGE), one can copy tables even from a live server under the following circumstances:

- You have done a `FLUSH TABLES` or `FLUSH TABLE table_name` for the specific table.
- The server is not accessing the tables during the copy process.

The advantage of FLUSH TABLES table_name FOR EXPORT is that the table is read locked until UNLOCK TABLES is executed.

Warning: If you do the above live copy, you are doing this on **your own risk** as if you do something wrong, the copied table is very likely to be corrupted. The original table will of course be fine.

An Efficient Way to Give Someone Else Access to a Read Only Table

If you want to give a user access to some data in a table for the user to use in their MariaDB server, you can do the following:

First let's create the table we want to export. To speed up things, we create this without any indexes. We use `TRANSACTIONAL=0 ROW_FORMAT=DYNAMIC` for Aria to use the smallest possible row format.

```
CREATE TABLE new_table ... ENGINE=ARIA TRANSACTIONAL=0;
ALTER TABLE new_table DISABLE_KEYS;
# Fill the table with data:
INSERT INTO new_table SELECT * ...
FLUSH TABLE new_table WITH READ LOCK;

# Copy table data to some external location, like /tmp with something
# like cp /my/data/test/new_table.* /tmp/

UNLOCK TABLES;
```

Then we pack it and generate the indexes. We use a big sort buffer to speed up generating the index.

```
> ls -l /tmp/new_table.*
-rw-rw---- 1 mysql my 42396148 Sep 21 17:58 /tmp/new_table.MAD
-rw-rw---- 1 mysql my     8192 Sep 21 17:58 /tmp/new_table.MAI
-rw-rw---- 1 mysql my     1039 Sep 21 17:58 /tmp/new_table.frm
> aria_pack /tmp/new_table
Compressing /tmp/new_table.MAD: (922666 records)
- Calculating statistics
- Compressing file
46.07%
> aria_chk -rq --ignore-control-file --sort_buffer_size=1G /tmp/new_table
Recreating table '/tmp/new_table'
- check record delete-chain
- recovering (with sort) Aria-table '/tmp/new_table'
Data records: 922666
- Fixing index 1
State updated
> ls -l /tmp/new_table.*
-rw-rw---- 1 mysql my 26271608 Sep 21 17:58 /tmp/new_table.MAD
-rw-rw---- 1 mysql my 10207232 Sep 21 17:58 /tmp/new_table.MAI
-rw-rw---- 1 mysql my     1039 Sep 21 17:58 /tmp/new_table.frm
```

The procedure for MyISAM tables is identical, except that myisamchk doesn't have the `--ignore-control-file` option.

Copying InnoDB's Transportable Tablespaces

InnoDB's file-per-table tablespaces are transportable, which means that you can copy a file-per-table tablespace from one MariaDB Server to another server. See Copying Transportable Tablespaces for more information.

Importing Tables

Tables that use most storage engines are immediately usable when their files are copied to the new datadir.

However, this is not true for tables that use InnoDB. InnoDB tables have to be imported with ALTER TABLE … IMPORT TABLESPACE. See Copying Transportable Tablespaces for more information.

See Also

- FLUSH TABLES FOR EXPORT
- FLUSH TABLES
- myisampack - Compressing the MyISAM data file for easier distribution.
- aria_pack - Compressing the Aria data file for easier distribution
- mariadb-dump - Copying tables to other SQL servers. You can use the `--tab` to create a CSV file of your table content.

This page is licensed: CC BY-SA / Gnu FDL

ALTER TABLE

Syntax

> **Note:**
>
> In the syntax, these options are the same as for the CREATE TABLE statement:
>
> - table_option
> - column_definition
> - partition_options
> - partition_definition

```
ALTER [ONLINE] [IGNORE] TABLE [IF EXISTS] tbl_name
    [WAIT n | NOWAIT]
    alter_specification [, alter_specification] ...

alter_specification:
    table_option ...
  | ADD [COLUMN] [IF NOT EXISTS] col_name column_definition
        [FIRST | AFTER col_name ]
  | ADD [COLUMN] [IF NOT EXISTS] (col_name column_definition, ...)
  | ADD {INDEX|KEY} [IF NOT EXISTS] [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY [IF NOT EXISTS]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [INDEX|KEY] [IF NOT EXISTS] [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD FULLTEXT [INDEX|KEY] [IF NOT EXISTS [index_name]
        (index_col_name,...) [index_option] ...
  | ADD SPATIAL [INDEX|KEY] [IF NOT EXISTS [index_name]
        (index_col_name,...) [index_option] ...
  | ADD VECTOR [INDEX|KEY] [IF NOT EXISTS [index_name]
        (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [IF NOT EXISTS] [index_name] (index_col_name,...)
        reference_definition
  | ADD PERIOD FOR [time_period_name|SYSTEM_TIME] (start_column_name, end_column_name)
  | ALTER [COLUMN] col_name SET DEFAULT literal | (expression)
  | ALTER [COLUMN] col_name DROP DEFAULT
  | ALTER {INDEX|KEY} [IF EXISTS] index_name [NOT] IGNORED
  | CHANGE [COLUMN] [IF EXISTS] old_col_name new_col_name column_definition
        [FIRST|AFTER col_name]
  | MODIFY [COLUMN] [IF EXISTS] col_name column_definition
        [FIRST | AFTER col_name]
  | DROP [COLUMN] [IF EXISTS] col_name [RESTRICT|CASCADE]
  | DROP [CONSTRAINT] PRIMARY KEY
  | DROP {INDEX|KEY} [IF EXISTS] index_name
  | DROP FOREIGN KEY [IF EXISTS] fk_symbol
  | DROP CONSTRAINT [IF EXISTS] constraint_name
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col_name [, col_name] ...
  | RENAME COLUMN [IF EXISTS] old_col_name TO new_col_name
  | RENAME {INDEX|KEY} old_index_name TO new_index_name
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | ALGORITHM [=] {DEFAULT|INPLACE|COPY|NOCOPY|INSTANT}
  | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
  | FORCE
  | partition_options
  | CONVERT TABLE normal_table TO partition_definition [{WITH | WITHOUT} VALIDATION]
  | CONVERT PARTITION partition_name TO TABLE tbl_name
  | ADD PARTITION [IF NOT EXISTS] (partition_definition)
  | DROP PARTITION [IF EXISTS] partition_names
  | TRUNCATE PARTITION partition_names
  | COALESCE PARTITION number
  | REORGANIZE PARTITION [partition_names INTO (partition_definitions)]
  | ANALYZE PARTITION partition_names
  | CHECK PARTITION partition_names
  | OPTIMIZE PARTITION partition_names
  | REBUILD PARTITION partition_names
  | REPAIR PARTITION partition_names
  | EXCHANGE PARTITION partition_name WITH TABLE tbl_name [{WITH | WITHOUT} VALIDATION]
  | REMOVE PARTITIONING
  | ADD SYSTEM VERSIONING
```

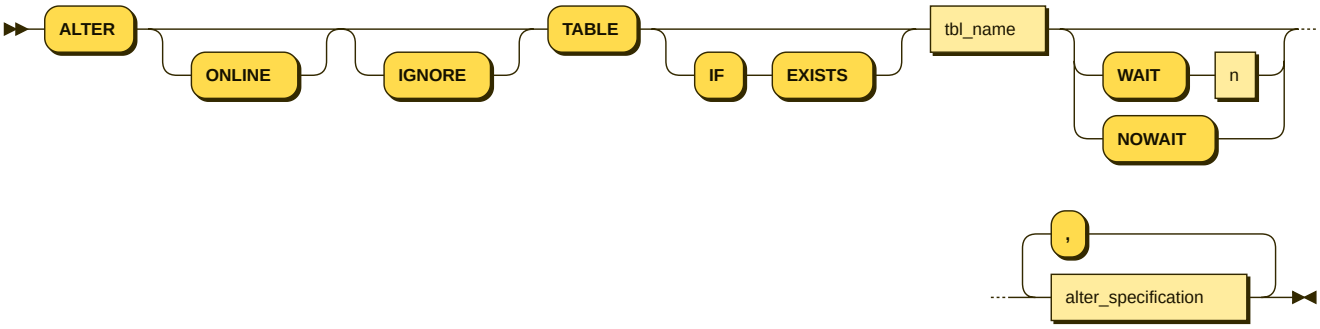
```
  | DROP SYSTEM VERSIONING

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH | RTREE}

index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | VISIBLE
  | COMMENT 'string'
  | CLUSTERING={YES | NO}
  | { IGNORED | NOT IGNORED }
  | DISTANCE={EUCLIDEAN | COSINE}
  | M=number

  table_option [[,] table_option] ...
```



This Railroad diagram is described in the preceding BNF diagram

Description

`ALTER TABLE` enables you to change the structure of an existing table. For example, you can add or delete columns, create or destroy indexes, change the type of existing columns, or rename columns or the table itself. You can also change the comment for the table and the storage engine of the table.

If another connection is using the table, a metadata lock is active, and this statement will wait until the lock is released. This is also true for non-transactional tables.

When adding a `UNIQUE` index on a column (or a set of columns) which have duplicated values, an error will be produced and the statement will be stopped. To suppress the error and force the creation of `UNIQUE` indexes, discarding duplicates, the IGNORE option can be specified. This can be useful if a column (or a set of columns) should be UNIQUE but it contains duplicate values; however, this technique provides no control on which rows are preserved and which are deleted. Also, note that `IGNORE` is accepted but ignored in `ALTER TABLE ... EXCHANGE PARTITION` statements.

This statement can also be used to rename a table. For details see RENAME TABLE.

When an index is created, the storage engine may use a configurable buffer in the process. Incrementing the buffer speeds up the index creation. Aria and MyISAM allocate a buffer whose size is defined by aria_sort_buffer_size or myisam_sort_buffer_size, also used for REPAIR TABLE. InnoDB allocates three buffers whose size is defined by innodb_sort_buffer_size.

Privileges

Executing the `ALTER TABLE` statement generally requires at least the ALTER privilege for the table or the database. If you are renaming a table, this also requires the DROP, CREATE and INSERT privileges for the table or the database.

Online DDL

Online DDL is supported with the ALGORITHM and LOCK clauses.

See InnoDB Online DDL Overview for more information on online DDL with InnoDB.

ALTER ONLINE TABLE

`ALTER ONLINE TABLE` also works for partitioned tables.

`ALTER ONLINE TABLE ...` is equivalent to the `ALTER TABLE ... LOCK = NONE`.

See the LOCK alter specification for more information.

WAIT/NOWAIT

Set the lock wait timeout. See WAIT and NOWAIT.

IF EXISTS

The `IF EXISTS` and `IF NOT EXISTS` clauses are available for the following clauses:

```
ADD COLUMN       [IF NOT EXISTS]
ADD INDEX        [IF NOT EXISTS]
ADD FOREIGN KEY  [IF NOT EXISTS]
ADD PARTITION    [IF NOT EXISTS]
CREATE INDEX     [IF NOT EXISTS]
DROP COLUMN      [IF EXISTS]
DROP INDEX       [IF EXISTS]
DROP FOREIGN KEY [IF EXISTS]
DROP PARTITION   [IF EXISTS]
CHANGE COLUMN    [IF EXISTS]
MODIFY COLUMN    [IF EXISTS]
DROP INDEX       [IF EXISTS]
```

When `IF EXISTS` and `IF NOT EXISTS` are used in clauses, queries will not report errors when the condition is triggered for that clause. A warning with the same message text will be issued and the `ALTER` will move on to the next clause in the statement (or end if finished).

If this is directive is used after `ALTER ... TABLE`, you don't get an error if the table doesn't exist. Example:

```
CREATE TABLE t (i INT);
ALTER TABLE IF EXISTS t RENAME COLUMN IF EXISTS i TO k;
```

Column Definitions

See CREATE TABLE: Column Definitions for information about column definitions.

Index Definitions

See CREATE TABLE: Index Definitions for information about index definitions.

The CREATE INDEX and DROP INDEX statements can also be used to add or remove an index.

Character Sets and Collations

```
CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
[DEFAULT] CHARACTER SET [=] charset_name
[DEFAULT] COLLATE [=] collation_name
```

See Setting Character Sets and Collations for details on setting the character sets and collations.

Alter Specifications

Table Options

See CREATE TABLE: Table Options for information about table options.

ADD COLUMN

```
... ADD COLUMN [IF NOT EXISTS]  (col_name [column_definition](../create/create-table.md#column-definitions),...)
```

Adds a column to the table. The syntax is the same as in CREATE TABLE. If you are using `IF NOT_EXISTS` the column will not be added if it was not there already. This is very useful when doing scripts to modify tables.

The `FIRST` and `AFTER` clauses affect the physical order of columns in the datafile. Use `FIRST` to add a column in the first (leftmost) position, or `AFTER` followed by a column name to add the new column in any other position. Note that, nowadays, the physical position of a column is usually irrelevant.

See also Instant ADD COLUMN for InnoDB.

DROP COLUMN

```
... DROP COLUMN [IF EXISTS] col_name [CASCADE|RESTRICT]
```

Drops the column from the table. If you are using `IF EXISTS` you will not get an error if the column didn't exist. If the column is part of any index, the column will be dropped from them, except if you add a new column with identical name at the same time. The index will be dropped if all columns from the index were dropped. If the column was used in a view or trigger, you will get an error next time the view or trigger is accessed. Dropping a column that is part of a multi-column `UNIQUE` constraint is not permitted. For example:

```
CREATE TABLE a (
  a int,
  b int,
  primary key (a,b)
);

ALTER TABLE x DROP COLUMN a;
[42000][1072] Key column 'A' doesn't exist in table
```

The reason is that dropping column `a` would result in the new constraint that all values in column `b` be unique. In order to drop the column, an explicit `DROP PRIMARY KEY` and `ADD PRIMARY KEY` would be required.

MariaDB supports instant DROP COLUMN. DROP COLUMN of an indexed column would imply DROP INDEX (and in the case of a non-UNIQUE multi-column index, possibly ADD INDEX). These will not be allowed with ALGORITHM=INSTANT, but unlike prior versions, they can be allowed with ALGORITHM=NOCOPY

`RESTRICT` and `CASCADE` are allowed to make porting from other database systems easier. In MariaDB, they do nothing.

MODIFY COLUMN

Allows you to modify the type of a column. The column will be at the same place as the original column and all indexes on the column will be kept. Note that when modifying column, you should specify all attributes for the new column.

```
CREATE TABLE t1 (a INT UNSIGNED AUTO_INCREMENT, PRIMARY KEY((a));
ALTER TABLE t1 MODIFY a BIGINT UNSIGNED AUTO_INCREMENT;
```

CHANGE COLUMN

Works like `MODIFY COLUMN` except that you can also change the name of the column. The column will be at the same place as the original column and all index on the column will be kept.

```
CREATE TABLE t1 (a INT UNSIGNED AUTO_INCREMENT, PRIMARY KEY(a));
ALTER TABLE t1 CHANGE a b BIGINT UNSIGNED AUTO_INCREMENT;
```

ALTER COLUMN

This lets you change column options.

```
CREATE TABLE t1 (a INT UNSIGNED AUTO_INCREMENT, b varchar(50), PRIMARY KEY(a));
ALTER TABLE t1 ALTER b SET DEFAULT 'hello';
```

RENAME INDEX/KEY

CURRENT

You can rename an index using the `RENAME INDEX` (or `RENAME KEY`) syntax:

```
ALTER TABLE t1 RENAME INDEX i_old TO i_new;
```

< 10.5.3

`RENAME INDEX/KEY` is not available.

RENAME COLUMN

CURRENT

You can rename a column using the `RENAME COLUMN` syntax:

```
ALTER TABLE t1 RENAME COLUMN c_old TO c_new;
```

< 10.5.3

`RENAME COLUMN` is not available.

ADD PRIMARY KEY

Adds a primary key. For `PRIMARY KEY` indexes, you can specify a name for the index, but it is silently ignored, and the name of the index is always `PRIMARY`. See Getting Started with Indexes: Primary Key for more information.

DROP PRIMARY KEY

Drops a primary key. For `PRIMARY KEY` indexes, you can specify a name for the index, but it is silently ignored, and the name of the index is always `PRIMARY`. See Getting Started with Indexes: Primary Key for more information.

ADD FOREIGN KEY

Adds a foreign key. For `FOREIGN KEY` indexes, a reference definition must be provided. For `FOREIGN KEY` indexes, you can specify a name for the constraint, using the `CONSTRAINT` keyword. That name will be used in error messages.

First, you have to specify the name of the target (parent) table and a column or a column list which must be indexed and whose values must match to the foreign key's values. The `MATCH` clause is accepted to improve the

compatibility with other DBMS's, but has no meaning in MariaDB. The `ON DELETE` and `ON UPDATE` clauses specify what must be done when a `DELETE` (or a `REPLACE`) statements attempts to delete a referenced row from the parent table, and when an `UPDATE` statement attempts to modify the referenced foreign key columns in a parent table row, respectively. The following options are allowed:

- `RESTRICT`: The delete/update operation is not performed. The statement terminates with a 1451 error (SQLSTATE '2300').
- `NO ACTION`: Synonym for `RESTRICT`.
- `CASCADE`: The delete/update operation is performed in both tables.
- `SET NULL`: The update or delete goes ahead in the parent table, and the corresponding foreign key fields in the child table are set to `NULL`. (They must not be defined as `NOT NULL` for this to succeed).
- `SET DEFAULT`: This option is implemented only for the legacy PBXT storage engine, which is disabled by default and no longer maintained. It sets the child table's foreign key fields to their `DEFAULT` values when the referenced parent table key entries are updated or deleted.

If either clause is omitted, the default behavior for the omitted clause is `RESTRICT`.

See Foreign Keys for more information.

DROP FOREIGN KEY

Drops a foreign key. See Foreign Keys for more information.

ADD INDEX

Adds a plain index. Plain indexes are regular indexes that are not unique, and are not acting as a primary key or a foreign key. They are also not the "specialized" `FULLTEXT` or `SPATIAL` indexes. For limits on InnoDB indexes, see InnoDB Limitations. See Getting Started with Indexes: Plain Indexes for more information.

DROP INDEX

Drops a plain index. Plain indexes are regular indexes that are not unique, and are not acting as a primary key or a foreign key. They are also not the "specialized" `FULLTEXT` or `SPATIAL` indexes. See Getting Started with Indexes: Plain Indexes for more information.

ADD UNIQUE INDEX

Adds a unique index. The `UNIQUE` keyword means that the index will not accept duplicated values, except for NULLs. An error will raise if you try to insert duplicate values in a UNIQUE index. For `UNIQUE` indexes, you can specify a name for the constraint, using the `CONSTRAINT` keyword. That name will be used in error messages. See Getting Started with Indexes: Unique Index for more information.

DROP UNIQUE INDEX

Drops a unique index. The `UNIQUE` keyword means that the index will not accept duplicated values, except for NULLs. An error will raise if you try to insert duplicate values in a UNIQUE index. For `UNIQUE` indexes, you can specify a name for the constraint, using the `CONSTRAINT` keyword. That name will be used in error messages. See Getting Started with Indexes: Unique Index for more information.

ADD FULLTEXT INDEX

Adds a `FULLTEXT` index. See Full-Text Indexes for more information.

DROP FULLTEXT INDEX

Drops a `FULLTEXT` index. See Full-Text Indexes for more information.

ADD SPATIAL INDEX

Adds a SPATIAL index. See SPATIAL INDEX for more information.

DROP SPATIAL INDEX

Drops a SPATIAL index. See SPATIAL INDEX for more information.

ENABLE/ DISABLE KEYS

`DISABLE KEYS` disables all non unique keys for the table for storage engines that support this (at least MyISAM and Aria). This can be used to speed up inserts into empty tables. `ENABLE KEYS` enables all disabled keys.

RENAME TO

Renames a table. See also RENAME TABLE.

ADD CONSTRAINT

Modifies the table adding a constraint on a particular column or columns.

```
ALTER TABLE table_name
ADD CONSTRAINT [constraint_name] CHECK(expression);
```

Before a row is inserted or updated, all constraints are evaluated in the order they are defined. If any constraint fails, then the row will not be updated. One can use most deterministic functions in a constraint, including UDF's.

```
CREATE TABLE account_ledger (
    id INT PRIMARY KEY AUTO_INCREMENT,
    transaction_name VARCHAR(100),
    credit_account VARCHAR(100),
    credit_amount INT,
    debit_account VARCHAR(100),
    debit_amount INT);

ALTER TABLE account_ledger
ADD CONSTRAINT is_balanced
    CHECK((debit_amount + credit_amount) = 0);
```

The `constraint_name` is optional. If you don't provide one in the `ALTER TABLE` statement, MariaDB auto-generates a name for you. This is done so that you can remove it later using DROP CONSTRAINT clause.

You can disable all constraint expression checks by setting the variable check_constraint_checks to `OFF`. You may find this useful when loading a table that violates some constraints that you want to later find and fix in SQL.

To view constraints on a table, query information_schema.TABLE_CONSTRAINTS:

```
SELECT CONSTRAINT_NAME, TABLE_NAME, CONSTRAINT_TYPE
FROM information_schema.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 'account_ledger';

+-----------------+----------------+-----------------+
| CONSTRAINT_NAME | TABLE_NAME     | CONSTRAINT_TYPE |
+-----------------+----------------+-----------------+
| is_balanced     | account_ledger | CHECK           |
+-----------------+----------------+-----------------+
```

DROP CONSTRAINT

`DROP CONSTRAINT` for `UNIQUE` and `FOREIGN KEY` constraints and `DROP CONSTRAINT` for `CHECK` constraints were introduced in an earlier version of MariaDB.

Modifies the table, removing the given constraint.

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

When you add a constraint to a table, whether through a CREATE TABLE or ALTER TABLE…ADD CONSTRAINT statement, you can either set a `constraint_name` yourself, or allow MariaDB to auto-generate one for you. To view constraints on a table, query information_schema.TABLE_CONSTRAINTS. For instance,

```
CREATE TABLE t (
   a INT,
   b INT,
   c INT,
   CONSTRAINT CHECK(a > b),
   CONSTRAINT check_equals CHECK(a = c));

SELECT CONSTRAINT_NAME, TABLE_NAME, CONSTRAINT_TYPE
FROM information_schema.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 't';

+-----------------+----------------+-----------------+
| CONSTRAINT_NAME | TABLE_NAME     | CONSTRAINT_TYPE |
+-----------------+----------------+-----------------+
| check_equals    | t              | CHECK           |
| CONSTRAINT_1    | t              | CHECK           |
+-----------------+----------------+-----------------+
```

To remove a constraint from the table, issue an `ALTER TABLE...DROP CONSTRAINT` statement:

```
ALTER TABLE t DROP CONSTRAINT is_unique;
```

ADD SYSTEM VERSIONING

Adds system versioning. See System-versioned tables.

DROP SYSTEM VERSIONING

Drops system versioning. See System-versioned tables.

ADD PERIOD FOR

See System-versioned tables, Application-time-period tables or Bitemporal Tables.

FORCE

`ALTER TABLE ... FORCE` forces MariaDB to rebuild the table.

```
ALTER TABLE tab_name FORCE;
```

With InnoDB, the table rebuild only reclaims unused space (i.e. the space previously used for deleted rows) if the innodb_file_per_table system variable is set to `ON` (the default). If the system variable is `OFF`, the space will not be reclaimed, but it will be reused for new data that's later added.

The rebuild may fail if conditions are violated due to a change in the sql_mode. For example:

```
CREATE OR REPLACE TABLE x (d DATE DEFAULT '0000-00-00');

SET SQL_MODE='NO_ZERO_DATE';

ALTER TABLE x FORCE;
ERROR 1067 (42000): Invalid default value for 'd'
```

Partitions

ADD PARTITION

See Partitioning Overview: Adding Partitions for details.

ANALYZE PARTITION

See Partitioning Overview: Analyzing Partitions for details.

CHECK PARTITION

See Partitioning Overview: Checking Partitions for details.

COALESCE PARTITION

Reduces the number of HASH or KEY partitions in a table. See Partitioning Overview: Coalescing Partitions.

CONVERT PARTITION / TABLE

CURRENT

`CONVERT PARTITION` can be used to remove a partition from a table and make this an ordinary table. For example:

```
ALTER TABLE partitioned_table CONVERT PARTITION part1 TO TABLE normal_table;
```

`CONVERT TABLE` will take an existing table and move this to another table as its own partition with a specified partition definition. For example the following moves `normal_table` to a partition of `partitioned_table` with a definition that its values, based on the `PARTITION BY` of the `partitioned_table`, are less than 12345.

```
ALTER TABLE partitioned_table CONVERT TABLE normal_table
  TO PARTITION part1 VALUES LESS THAN (12345);
```

The optional `[{WITH | WITHOUT} VALIDATION]` is permitted.

See Partitioning Overview: Converting Partitions to/from Tables for more details.

< 11.4

`CONVERT PARTITION` can be used to remove a partition from a table and make this an ordinary table. For example:

```
ALTER TABLE partitioned_table CONVERT PARTITION part1 TO TABLE normal_table;
```

`CONVERT TABLE` will take an existing table and move this to another table as its own partition with a specified partition definition. For example the following moves `normal_table` to a partition of `partitioned_table` with a definition that its values, based on the `PARTITION BY` of the `partitioned_table`, are less than 12345.

```
ALTER TABLE partitioned_table CONVERT TABLE normal_table
  TO PARTITION part1 VALUES LESS THAN (12345);
```

The optional clause `[{WITH | WITHOUT} VALIDATION]` is not available.

See Partitioning Overview: Converting Partitions to/from Tables for more details.

< 10.7

`CONVERT PARTITION` and `CONVERT TABLE` are not available.

DROP PARTITION

Used to drop specific partitions (and discard all data within the specified partitions) for RANGE and LIST partitions. See Partitioning Overview: Dropping Partitions.

EXCHANGE PARTITION

> #### CURRENT
>
> This clause is used to exchange the contents of a partition with another table. This is performed by swapping the tablespaces of the partition with the other table.
>
> The optional `[{WITH | WITHOUT} VALIDATION]` is permitted.
>
> See Partitioning Overview: Exchanging Partitions for more details.
>
> See also copying InnoDB's transportable tablespaces.
>
> #### < 11.4
>
> This clause is used to exchange the contents of a partition with another table. This is performed by swapping the tablespaces of the partition with the other table.
>
> The optional `[{WITH | WITHOUT} VALIDATION]` is not permitted.
>
> See Partitioning Overview: Exchanging Partitions for more details.
>
> See also copying InnoDB's transportable tablespaces.

OPTIMIZE PARTITION

See Partitioning Overview: Optimizing Partitions for details.

REMOVE PARTITIONING

See Partitioning Overview: Removing Partitioning.

REORGANIZE PARTITION

See Partitioning Overview: Reorganizing Partitions.

TRUNCATE PARTITION

See Partitioning Overview: Truncating Partitions.

DISCARD TABLESPACE

This is used to discard an InnoDB table's tablespace.

See copying InnoDB's transportable tablespaces for more information.

IMPORT TABLESPACE

This is used to import an InnoDB table's tablespace. The tablespace should have been copied from its original server after executing FLUSH TABLES FOR EXPORT.

See copying InnoDB's transportable tablespaces for more information.

`ALTER TABLE ... IMPORT` only applies to InnoDB tables. Most other popular storage engines, such as Aria and MyISAM, will recognize their data files as soon as they've been placed in the proper directory under the datadir, and no special DDL is required to import them.

ALGORITHM

The `ALTER TABLE` statement supports the `ALGORITHM` clause. This clause is one of the clauses that is used to implement online DDL. `ALTER TABLE` supports several different algorithms. An algorithm can be explicitly chosen for an `ALTER TABLE` operation by setting the `ALGORITHM` clause. The supported values are:

- `ALGORITHM=DEFAULT` - This implies the default behavior for the specific statement, such as if no `ALGORITHM` clause is specified.
- `ALGORITHM=COPY`
- `ALGORITHM=INPLACE`
- `ALGORITHM=NOCOPY`
- `ALGORITHM=INSTANT`

See InnoDB Online DDL Overview: ALGORITHM for information on how the `ALGORITHM` clause affects InnoDB.

ALGORITHM=DEFAULT

The default behavior, which occurs if `ALGORITHM=DEFAULT` is specified, or if `ALGORITHM` is not specified at all, usually only makes a copy if the operation doesn't support being done in-place at all. In this case, the most efficient available algorithm will usually be used.

The old_alter_table system variable is deprecated. Instead, the alter_algorithm system variable defines the default algorithm for `ALTER TABLE` operations. This was removed in MariaDB 11.5 for the following reasons:

- alter_algorithm was introduced as a replacement for the old_alter_table that was used to force the usage of the original alter table algorithm (copy) in cases where the new alter algorithm did not work. The new option was added as a way to force the usage of a specific algorithm when it should instead have made it possible to disable algorithms that would not work for some reason.
- alter_algorithm introduced some cases where ALTER TABLE would not work without specifying the ALGORITHM=XXX option together with ALTER TABLE.
- Having different values of alter_algorithm on the primary and replica could cause replicas to stop unexpectedly.
- ALTER TABLE FORCE, as used by mariadb-upgrade, would not always work if alter_algorithm was set for the server.
- As part of MDEV-33449 "improving repair of tables" it became clear that alter-algorithm made it harder to provide a better and more consistent ALTER TABLE FORCE and REPAIR TABLE, and it would be better to remove it.

ALGORITHM=COPY

`ALGORITHM=COPY` is the name for the original `ALTER TABLE` algorithm from early MariaDB versions.

When `ALGORITHM=COPY` is set, MariaDB essentially does the following operations:

```
-- Create a temporary table with the new definition
CREATE TEMPORARY TABLE tmp_tab (
...
);

-- Copy the data from the original table
INSERT INTO tmp_tab
   SELECT * FROM original_tab;

-- Drop the original table
DROP TABLE original_tab;

-- Rename the temporary table, so that it replaces the original one
RENAME TABLE tmp_tab TO original_tab;
```

This algorithm is very inefficient, but it is generic, so it works for all storage engines.

If `ALGORITHM=COPY` is specified, then the copy algorithm will be used even if it is not necessary. This can result in a lengthy table copy. If multiple `ALTER TABLE` operations are required that each require the table to be rebuilt, then it is best to specify all operations in a single `ALTER TABLE` statement, so that the table is only rebuilt once.

> **CURRENT**
>
> `ALTER TABLE` can perform most operations with `ALGORITHM=COPY`, `LOCK=NONE`. See LOCK=NONE.
>
> **< 11.2**
>
> `ALTER TABLE` cannot perform operations with `ALGORITHM=COPY`, `LOCK=NONE`.

ALGORITHM=INPLACE

`ALGORITHM=COPY` can be incredibly slow, because the whole table has to be copied and rebuilt. `ALGORITHM=INPLACE` was introduced as a way to avoid this by performing operations in-place and avoiding the table copy and rebuild, when possible.

When `ALGORITHM=INPLACE` is set, the underlying storage engine uses optimizations to perform the operation while avoiding the table copy and rebuild. However, `INPLACE` is a bit of a misnomer, since some operations may still require the table to be rebuilt for some storage engines. Regardless, several operations can be performed without a full copy of the table for some storage engines.

A more accurate name would have been `ALGORITHM=ENGINE`, where `ENGINE` refers to an "engine-specific" algorithm.

If an `ALTER TABLE` operation supports `ALGORITHM=INPLACE`, it can be performed using optimizations by the underlying storage engine, but it may rebuilt.

See InnoDB Online DDL Operations with ALGORITHM=INPLACE for more.

ALGORITHM=NOCOPY

`ALGORITHM=INPLACE` can sometimes be surprisingly slow in instances where it has to rebuild the clustered index, because when the clustered index has to be rebuilt, the whole table has to be rebuilt. `ALGORITHM=NOCOPY` was introduced as a way to avoid this.

If an `ALTER TABLE` operation supports `ALGORITHM=NOCOPY`, then it can be performed without rebuilding the clustered index.

If `ALGORITHM=NOCOPY` is specified for an `ALTER TABLE` operation that does not support `ALGORITHM=NOCOPY`, then an error will be raised. In this case, raising an error is preferable, if the alternative is for the operation to rebuild the clustered index, and perform unexpectedly slowly.

See InnoDB Online DDL Operations with ALGORITHM=NOCOPY for more.

ALGORITHM=INSTANT

`ALGORITHM=INPLACE` can sometimes be surprisingly slow in instances where it has to modify data files. `ALGORITHM=INSTANT` was introduced as a way to avoid this.

If an `ALTER TABLE` operation supports `ALGORITHM=INSTANT`, then it can be performed without modifying any data files.

If `ALGORITHM=INSTANT` is specified for an `ALTER TABLE` operation that does not support `ALGORITHM=INSTANT`, then an error will be raised. In this case, raising an error is preferable, if the alternative is for the operation to modify data files, and perform unexpectedly slowly.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT for more.

LOCK

The `ALTER TABLE` statement supports the `LOCK` clause. This clause is one of the clauses that is used to implement online DDL. `ALTER TABLE` supports several different locking strategies. A locking strategy can be explicitly chosen for an `ALTER TABLE` operation by setting the `LOCK` clause. The supported values are:

DEFAULT

Acquire the least restrictive lock on the table that is supported for the specific operation. Permit the maximum amount of concurrency that is supported for the specific operation.

NONE

Acquire no lock on the table. Permit **all** concurrent DML. If this locking strategy is not permitted for an operation, then an error is raised. From MariaDB 11.2, `ALTER TABLE` can do most operations with `ALGORITHM=COPY, LOCK=NONE`, that is, in most cases, unless the algorithm and lock level are explicitly specified, `ALTER TABLE` will be performed using the `COPY` algorithm while simultaneously allowing concurrent DML statements on the altered table. If this is not desired, one can explicitly specify a different lock level or set old_mode to LOCK_ALTER_TABLE_COPY that will make `ALGORITHM=COPY` use `LOCK=SHARED` by default (but still allowing `LOCK=NONE` to be specified explicitly).

SHARED

Acquire a read lock on the table. Permit **read-only** concurrent DML. If this locking strategy is not permitted for an operation, then an error is raised.

EXCLUSIVE

Acquire a write lock on the table. Do **not** permit concurrent DML.

Different storage engines support different locking strategies for different operations. If a specific locking strategy is chosen for an `ALTER TABLE` operation, and that table's storage engine does not support that locking strategy for that specific operation, then an error will be raised.

If the `LOCK` clause is not explicitly set, then the operation uses `LOCK=DEFAULT`.

ALTER ONLINE TABLE is equivalent to `LOCK=NONE`. Therefore, the ALTER ONLINE TABLE statement can be used to ensure that your `ALTER TABLE` operation allows all concurrent DML.

See InnoDB Online DDL Overview: LOCK for information on how the `LOCK` clause affects InnoDB.

Index Options

See CREATE TABLE page for meaning of the index options.

Progress Reporting

MariaDB provides progress reporting for `ALTER TABLE` statement for clients that support the new progress reporting protocol. For example, if you were using the mariadb client, then the progress report might look like this::

```
ALTER TABLE test ENGINE=Aria;
Stage: 1 of 2 'copy to tmp table'    46% of stage
```

The progress report is also shown in the output of the SHOW PROCESSLIST statement and in the contents of the information_schema.PROCESSLIST table.

See Progress Reporting for more information.

Aborting ALTER TABLE Operations

If an `ALTER TABLE` operation is being performed and the connection is killed, the changes will be rolled back in a controlled manner. The rollback can be a slow operation as the time it takes is relative to how far the operation has progressed.

Aborting `ALTER TABLE ... ALGORITHM=COPY` was made faster by removing excessive undo logging (MDEV-11415). This significantly shortened the time it takes to abort a running ALTER TABLE operation, compared with earlier releases.

Atomic ALTER TABLE

> **CURRENT**
>
> `ALTER TABLE` is atomic for most engines, including InnoDB, MyRocks, MyISAM and Aria (MDEV-25180). This means that if there is a crash (server down or power outage) during an `ALTER TABLE` operation, after recovery, either the old table and associated triggers and status will be intact, or the new table will be active. In older MariaDB versions one could get leftover #sql-alter..', '#sql-backup..' or 'table_name.frm˝' files if the system crashed during the `ALTER TABLE` operation.
>
> See Atomic DDL for more information.
>
> **< 10.6**
>
> Atomic `ALTER TABLE` is not available.

Replication

> **CURRENT**
>
> `ALTER TABLE` got fully executed on the primary first, and only then was it replicated and started executing on replicas. An option was added to replicate sooner and begin executing on replicas, directly when it *starts* executing on the primary, not when it *finishes*. This way the replication lag caused by a heavy `ALTER TABLE` can be completely eliminated (MDEV-11675).
>
> **< 10.8**
>
> The binlog_alter_two_phase option is not available.

Examples

Adding a new column:

```
ALTER TABLE t1 ADD x INT;
```

Dropping a column:

```
ALTER TABLE t1 DROP x;
```

Modifying the type of a column:

```
ALTER TABLE t1 MODIFY x bigint unsigned;
```

Changing the name and type of a column:

```
ALTER TABLE t1 CHANGE a b bigint unsigned auto_increment;
```

Combining multiple clauses in a single ALTER TABLE statement, separated by commas:

```
ALTER TABLE t1 DROP x, ADD x2 INT,  CHANGE y y2 INT;
```

Changing the storage engine and adding a comment:

```
ALTER TABLE t1
  ENGINE = InnoDB
  COMMENT = 'First of three tables containing usage info';
```

Rebuilding the table (the previous example will also rebuild the table if it was already InnoDB):

```
ALTER TABLE t1 FORCE;
```

Dropping an index:

```
ALTER TABLE rooms DROP INDEX u;
```

Adding a unique index:

```
ALTER TABLE rooms ADD UNIQUE INDEX u(room_number);
```

Adding a primary key for an application-time period table with a WITHOUT OVERLAPS constraint:

```
ALTER TABLE rooms ADD PRIMARY KEY(room_number, p WITHOUT OVERLAPS);
```

CURRENT

An `ALTER` query can be replicated faster with this statement, which must be run before the `ALTER` statement:

```
SET @@SESSION.binlog_alter_two_phase = TRUE;
```

Binlog would contain two event groups, of which the first one gets delivered to replicas before ALTER is taken to actual execution on the primary:

```
| master-bin.000001 | 495 | Gtid              |         1 |         537 | GTID 0-1-2 START ALTER
|
| master-bin.000001 | 537 | Query             |         1 |         655 | use `test`; alter table t add
column b int, algorithm=inplace |
| master-bin.000001 | 655 | Gtid              |         1 |         700 | GTID 0-1-3 COMMIT ALTER id=2
|
| master-bin.000001 | 700 | Query             |         1 |
```

< 10.8.1

This statement is not available:

```
SET @@SESSION.binlog_alter_two_phase = true;
```

See Also

- CREATE TABLE
- DROP TABLE
- Character Sets and Collations
- SHOW CREATE TABLE
- Instant ADD COLUMN for InnoDB

This page is licensed: GPLv2, originally from fill_help_tables.sql

CREATE TABLE

Syntax

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options    ]... [partition_options]
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)] [table_options   ]... [partition_options]
    select_statement
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   { LIKE old_table_name | (LIKE old_table_name) }

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

Description

Use the `CREATE TABLE` statement to create a table with the given name.

In its most basic form, the `CREATE TABLE` statement provides a table name followed by a list of columns, indexes, and constraints. By default, the table is created in the default database. Specify a database with `db_name.tbl_name`. If you quote the table name, you must quote the database name and table name separately as `` `db_name`.`tbl_name` ``. This is particularly useful for CREATE TABLE ... SELECT, because it allows creating a table in a database that contains data from other databases. See Identifier Qualifiers.

If a table with the same name exists, error 1050 results. Use IF NOT EXISTS to suppress this error and issue a note instead. Use SHOW WARNINGS to see notes.

The `CREATE TABLE` statement automatically commits the current transaction, except when using the TEMPORARY keyword.

For valid identifiers to use as table names, see Identifier Names.

> **Note:**
>
> If the `default_storage_engine` is set to `ColumnStore`, it needs to be set on all UMs. Otherwise, when the tables using the default engine are replicated across UMs, they will use the wrong engine. You should therefore not use this option as a session variable with ColumnStore.

Microsecond precision can be between 0-6. If no precision is specified, it is assumed to be 0, for backward compatibility reasons.

Privileges

Executing the `CREATE TABLE` statement requires the CREATE privilege for the table or the database.

CREATE OR REPLACE

If the `OR REPLACE` clause is used and the table already exists, then instead of returning an error, the server will drop the existing table and replace it with the newly defined table.

This syntax was originally added to make replication more robust if it has to rollback and repeat statements such as `CREATE ... SELECT` on replicas.

```
CREATE OR REPLACE TABLE table_name (a INT);
```

is basically the same as:

```
DROP TABLE IF EXISTS TABLE_NAME;
CREATE TABLE TABLE_NAME (a INT);
```

with the following exceptions:

- If `table_name` was locked with LOCK TABLES, it will continue to be locked after the statement.
- Temporary tables are only dropped if the `TEMPORARY` keyword was used. (With DROP TABLE, temporary tables are preferred to be dropped before normal tables).

Things to be Aware of With CREATE OR REPLACE

- The table is dropped first (if it existed), and after that, the `CREATE` is done. Because of this, if the `CREATE` fails, then the table will not exist anymore after the statement. If the table was used with `LOCK TABLES`, it will be unlocked.
- One can't use `OR REPLACE` together with `IF EXISTS`.
- Replicas will, by default, use `CREATE OR REPLACE` when replicating `CREATE` statements that don't use `IF EXISTS`. This can be changed by setting the variable slave-ddl-exec-mode to `STRICT`.

CREATE TABLE IF NOT EXISTS

If the `IF NOT EXISTS` clause is used, then the table will only be created if a table with the same name does not already exist. If the table already exists, then a warning will be triggered by default.

CREATE TEMPORARY TABLE

Use the `TEMPORARY` keyword to create a temporary table that is only available to the current session. Temporary tables are dropped when the session ends. Temporary table names are specific to the session. They will not conflict with other temporary tables from other sessions, even if they share the same name. They will shadow names of non-temporary tables or views, if they are identical. A temporary table can have the same name as a non-temporary table, which is located in the same database. In that case, their name will reference the temporary table when used in SQL statements. You must have the CREATE TEMPORARY TABLES privilege on the database to create temporary tables. If no storage engine is specified, the default_tmp_storage_engine setting will determine the engine.

> **CURRENT**
>
> ROCKSDB temporary tables cannot be created by setting the default_tmp_storage_engine system variable or using `CREATE TEMPORARY TABLE LIKE`. If you try, an error is returned. Explicitly creating a temporary table with `ENGINE=ROCKSDB` has never been permitted.
>
> **< 10.7**
>
> ROCKSDB temporary tables cannot be created by setting the default_tmp_storage_engine system variable, or using `CREATE TEMPORARY TABLE LIKE`. They can be specified, but fail silently, and a `MyISAM` table is created instead. Explicitly creating a temporary table with `ENGINE=ROCKSDB` has never been permitted.

Replicating temporary tables

> **CURRENT**
>
> By default, temporary tables are only created on the replica if the primary is using the STATEMENT binary log format.
>
> The new deterministic rules for logging of temporary tables are:

- The `STATEMENT` binlog format is used. The user can change this behavior by setting create_tmp_table_binlog_formats to `MIXED`, `STATEMENT` in which case the create is logged in statement format also in `MIXED` mode (as before).
- Changes to temporary tables are only binlogged if and only if the CREATE was logged. The logging happens under `STATEMENT` or `MIXED`. If binlog_format=ROW, temporary table changes are not binlogged. A temporary table that is changed under ROW is marked as 'not up to date in binlog' and no future row changes are logged. Any usage of this temporary table will force row logging of other tables in any future statements using the temporary table to be row logged.
- `DROP TEMPORARY` is binlogged only if the `CREATE` was binlogged.

< 12.0.1

In some contexts, temporary tables on the primary and replica can become inconsistent.
One example is if a temporary table is updated with the value of a non deterministic function like UUID(), in which the change is never sent to the replica.

In some other contexts, while using `MIXED` mode, all changes will be logged in `ROW` mode while the user has any active temporary tables, even if the temporary tables are not used in the query. This depends on in which format some previous independent commands were logged.

There are many other pitfalls with logging temporary table to the replica.

CREATE TABLE ... LIKE

Use the `LIKE` clause instead of a full table definition to create an empty table with the same definition as another table, including columns, indexes, and table options. Foreign key definitions, as well as any DATA DIRECTORY or INDEX DIRECTORY table options specified on the original table, will not be created.

`LIKE` does not preserve the `TEMPORARY` status of the original table. To make the new table `TEMPORARY` as well, use `CREATE TEMPORARY TABLE ... LIKE`.

`LIKE` does not work with views, only base tables. Attempting to use it on a view will result in an error:

```
CREATE VIEW v (mycol) AS SELECT 'abc';

CREATE TABLE v2 LIKE v;
ERROR 1347 (HY000): 'test.v' is not of type 'BASE TABLE'
```

The same version of the table storage format as found in the original table is used for the new table.

`CREATE TABLE ... LIKE` performs the same checks as `CREATE TABLE`. So a statement may fail if a change in the SQL_MODE renders it invalid. For example:

```
CREATE OR REPLACE TABLE x (d DATE DEFAULT '0000-00-00');

SET SQL_MODE='NO_ZERO_DATE';

CREATE OR REPLACE TABLE y LIKE x;
ERROR 1067 (42000): Invalid default value for 'd'
```

CREATE TABLE ... SELECT

You can create a table containing data from other tables using the `CREATE ... SELECT` statement. Columns will be created in the table for each field returned by the `SELECT` query.

You can also define some columns normally and add other columns from a `SELECT`. You can also create columns in the normal way and assign them some values using the query, this is done to force a certain type or other field characteristics. The columns that are not named in the query will be placed before the others. For example:

```
CREATE TABLE test (a INT NOT NULL, b CHAR(10)) ENGINE=MyISAM
    SELECT 5 AS b, c, d FROM another_table;
```

Remember that the query just returns data. If you want to use the same indexes or the same column attributes (`[NOT] NULL`, `DEFAULT`, `AUTO_INCREMENT`, `CHECK` constraints) in the new table, you need to specify them manually. Types and sizes are not automatically preserved if no data is returned by the `SELECT` that requires the full size, and `VARCHAR` could be converted into `CHAR`. The CAST() function can be used to force the new table to use certain types.

Aliases (`AS`) are taken into account, and they should always be used when you `SELECT` an expression (function, arithmetical operation, etc.).

If an error occurs during the query, the table will not be created at all.

If the new table has a primary key or `UNIQUE` indexes, you can use the IGNORE or `REPLACE` keywords to handle duplicate key errors during the query. `IGNORE` means that the newer values must not be inserted if an identical value exists in the index. `REPLACE` means that older values must be overwritten.

If the columns in the new table are more than the rows returned by the query, the columns populated by the query will be placed after the other columns. Note that if the strict `SQL_MODE` is on, and the columns that are not named in the query do not have a `DEFAULT` value, an error will be raised and no rows will be copied.

Concurrent inserts are not used during the execution of a `CREATE ... SELECT`.

If the table already exists, an error similar to the following will be returned:

```
ERROR 1050 (42S01): Table 't' already exists
```

If the `IF NOT EXISTS` clause is used, and the table exists, a note will be produced instead of an error.

To insert rows from a query into an existing table, INSERT … SELECT can be used.

Column Definitions

```
create_definition:
  { col_name column_definition | index_definition | period_definition | CHECK (expr) }

column_definition:
  data_type
    [NOT NULL | NULL] [DEFAULT default_value | (expression)]
    [ON UPDATE [NOW | CURRENT_TIMESTAMP] [(precision)]]
    [AUTO_INCREMENT] [ZEROFILL] [UNIQUE [KEY] | [PRIMARY] KEY]
    [INVISIBLE] [{WITH|WITHOUT} SYSTEM VERSIONING]
    [COMMENT 'string'] [REF_SYSTEM_ID = value]
    [reference_definition]
  | data_type [GENERATED ALWAYS]
  AS [ ROW {START|END} [NOT NULL ENABLE] [[PRIMARY] KEY]
        | (expression) [VIRTUAL | PERSISTENT | STORED] ]
      [INVISIBLE] [UNIQUE [KEY]] [COMMENT 'string']

constraint_definition:
   CONSTRAINT [constraint_name] CHECK (expression)
```

Note:

Note:

CURRENT

MariaDB accepts the shortcut format with a `REFERENCES` clause only in `ALTER TABLE` and `CREATE TABLE` statements, but that syntax does nothing. For example:

```
CREATE TABLE b(for_key INT REFERENCES a(not_key));
```

MariaDB will attempt to apply the constraint. See Foreign Keys examples.

< 10.5

MariaDB accepts the shortcut format with a `REFERENCES` clause only in `ALTER TABLE` and `CREATE TABLE` statements, but that syntax does nothing. For example:

```
CREATE TABLE b(for_key INT REFERENCES a(not_key));
```

Each definition either creates a column in the table or specifies an index or constraint on one or more columns. See Indexes below for details on creating indexes.

Create a column by specifying a column name and a data type, optionally followed by column options. See Data Types for a full list of data types allowed in MariaDB.

NULL and NOT NULL

Use the `NULL` or `NOT NULL` options to specify that values in the column may or may not be `NULL`, respectively. By default, values may be `NULL`. See also NULL Values in MariaDB.

DEFAULT Column Option

Specify a default value using the `DEFAULT` clause. If you don't specify `DEFAULT` then the following rules apply:

- If the column is not defined with `NOT NULL`, `AUTO_INCREMENT` or `TIMESTAMP`, an explicit `DEFAULT NULL` will be added.
  Note that in MySQL, you may get an explicit `DEFAULT` for primary key parts, if not specified with NOT NULL.

The default value will be used if you INSERT a row without specifying a value for that column, or if you specify DEFAULT for that column.

CURRENT_TIMESTAMP may also be used as the default value for a DATETIME

You can use most functions in `DEFAULT`. Expressions should have parentheses around them. If you use a non deterministic function in `DEFAULT` then all inserts to the table will be replicated in row mode. You can even refer to earlier columns in the `DEFAULT` expression (excluding `AUTO_INCREMENT` columns):

```
CREATE TABLE t1 (a INT DEFAULT (1+1), b INT DEFAULT (a+1));
CREATE TABLE t2 (a BIGINT PRIMARY KEY DEFAULT UUID_SHORT());
```

The `DEFAULT` clause cannot contain any stored functions or subqueries, and a column used in the clause must already have been defined earlier in the statement.

It is possible to assign BLOB or TEXT columns a `DEFAULT` value.

You can also use DEFAULT (NEXT VALUE FOR sequence).

AUTO_INCREMENT Column Option

Use AUTO_INCREMENT to create a column whose value can be set automatically from a simple counter. You can only use `AUTO_INCREMENT` on a column with an integer type. The column must be a key, and there can only be one `AUTO_INCREMENT` column in a table. If you insert a row without specifying a value for that column (or if you specify `0`, `NULL`, or DEFAULT as the value), the actual value will be taken from the counter, with each insertion

incrementing the counter by one. You can still insert a value explicitly. If you insert a value that is greater than the current counter value, the counter is set based on the new value. An `AUTO_INCREMENT` column is implicitly `NOT NULL`. Use LAST_INSERT_ID to get the AUTO_INCREMENT value most recently used by an INSERT statement.

ZEROFILL Column Option

If the `ZEROFILL` column option is specified for a column using a numeric data type, then the column will be set to `UNSIGNED` and the spaces used by default to pad the field are replaced with zeros. `ZEROFILL` is ignored in expressions or as part of a UNION, INTERSECT, or EXCEPT. `ZEROFILL` is a non-standard MariaDB and MySQL extension.

PRIMARY KEY Column Option

Use `PRIMARY KEY` to make a column a primary key. A primary key is a special type of a unique key. There can be at most one primary key per table, and it is implicitly `NOT NULL`.

Specifying a column as a unique key creates a unique index on that column. See the Index Definitions section below for more information.

UNIQUE KEY Column Option

Use `UNIQUE KEY` (or just `UNIQUE`) to specify that all values in the column must be distinct from each other. Unless the column is `NOT NULL`, there may be multiple rows with `NULL` in the column.

```
```

When any inserts or updates occur in the table, reading the binlog shows the hidden column (@3). it causes confusion for the user; we can document these behaviours.

```
### INSERT INTO `securedb`.`t_long_keys`
### SET
###   @1=1 /* INT meta=0 nullable=0 is_null=0 */
###   @2='a' /* VARSTRING(4073) meta=4073 nullable=1 is_null=0 */
###   @3=580 /* LONGINT meta=0 nullable=1 is_null=0 */
```

```
CREATE TABLE t_long_keys (   a INT PRIMARY KEY,   b  VARCHAR(4073),   UNIQUE KEY `uk_b` (b) )
ENGINE=InnoDB;
Query OK, 0 rows affected (0.022 sec)

show create table t_long_keys\G
*************************** 1. row ***************************
       Table: t_long_keys
Create Table: CREATE TABLE `t_long_keys` (
  `a` int(11) NOT NULL,
  `b` varchar(4073) DEFAULT NULL,
  PRIMARY KEY (`a`),
  UNIQUE KEY `uk_b` (`b`) USING HASH
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
1 row in set (0.001 sec)

select * from information_schema.INNODB_SYS_TABLES where name like '%t_long_keys%';;
+----------+----------------------+------+--------+-------+------------+---------------+------------+
| TABLE_ID | NAME                 | FLAG | N_COLS | SPACE | ROW_FORMAT | ZIP_PAGE_SIZE | SPACE_TYPE |
+----------+----------------------+------+--------+-------+------------+---------------+------------+
|       64 | securedb/t_long_keys |   33 |      5 |    43 | Dynamic    |             0 | Single     |
+----------+----------------------+------+--------+-------+------------+---------------+------------+
1 row in set (0.003 sec)

select * from information_schema.INNODB_SYS_COLUMNS where TABLE_ID=64;
+----------+---------------+-------+-------+--------+------+
| TABLE_ID | NAME          | POS   | MTYPE | PRTYPE | LEN  |
+----------+---------------+-------+-------+--------+------+
|       64 | a             |     0 |     6 |   1283 |    4 |
|       64 | b             |     1 |     1 | 528399 | 4073 |
|       64 | DB_ROW_HASH_1 | 65538 |     6 |   9736 |    8 |
+----------+---------------+-------+-------+--------+------+
```

See the Index Definitions section below for more information.

COMMENT Column Option

You can provide a comment for each column using the `COMMENT` clause. The maximum length is 1024 characters. Use the SHOW FULL COLUMNS statement to see column comments.

REF_SYSTEM_ID

`REF_SYSTEM_ID` can be used to specify Spatial Reference System IDs for spatial data type columns. For example:

```
CREATE TABLE t1(g GEOMETRY(9,4) REF_SYSTEM_ID=101);
```

Generated Columns

A generated column is a column in a table that cannot explicitly be set to a specific value in a DML query. Instead, its value is automatically generated based on an expression. This expression might generate the value based on the values of other columns in the table, or it might generate the value by calling built-in functions or user-defined functions (UDFs).

There are two types of generated columns:

- `PERSISTENT` or `STORED`: This type's value is actually stored in the table.
- `VIRTUAL`: This type's value is not stored at all. Instead, the value is generated dynamically when the table is queried. This type is the default.

Generated columns are also sometimes called computed columns or virtual columns.

For a complete description about generated columns and their limitations, see Generated (Virtual and Persistent/Stored) Columns.

COMPRESSED

Certain columns may be compressed. See Storage-Engine Independent Column Compression.

INVISIBLE

Columns may be made invisible, and hidden in certain contexts. See Invisible Columns.

WITH SYSTEM VERSIONING Column Option

Columns may be explicitly marked as included from system versioning. See System-versioned tables for details.

WITHOUT SYSTEM VERSIONING Column Option

Columns may be explicitly marked as excluded from system versioning. See System-versioned tables for details.

Index Definitions

```
index_definition:
    {INDEX|KEY} [index_name] [index_type] (index_col_name,...) [index_option] ...
  {{{|}}} {FULLTEXT|SPATIAL} [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
  {{{|}}} [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...) [index_option] ...
  {{{|}}} [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [index_name] [index_type] (index_col_name,...)
[index_option] ...
  {{{|}}} [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (index_col_name,...) reference_definition

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH | RTREE}

index_option:
    [ KEY_BLOCK_SIZE [=] value
  {{{|}}} index_type
  {{{|}}} WITH PARSER parser_name
  {{{|}}} VISIBLE
  {{{|}}} COMMENT 'string'
  {{{|}}} ADAPTIVE_HASH_INDEX [=] {DEFAULT | YES | NO}
  {{{|}}} COMPLETE_FIELDS [=] number
  {{{|}}} BYTES_FROM_INCOMPLETE_FIELD [=] number
  {{{|}}} FOR_EQUAL_HASH_POINT_TO_LAST_RECORD [=] {DEFAULT | YES | NO} ]
  [ IGNORED | NOT IGNORED ]

reference_definition:
    REFERENCES tbl_name (index_col_name,...)
      [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
      [ON DELETE reference_option]
      [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION
```

`INDEX` and `KEY` are synonyms.

Index names are optional, if not specified an automatic name will be assigned. Index name are needed to drop indexes and appear in error messages when a constraint is violated.

For limits on InnoDB indexes, see InnoDB Limitations.

Index Categories

Plain Indexes

Plain indexes are regular indexes that are not unique, and are not acting as a primary key or a foreign key. They are also not the “specialized” `FULLTEXT` or `SPATIAL` indexes.

See Getting Started with Indexes: Plain Indexes for more information.

PRIMARY KEY

For `PRIMARY KEY` indexes, you can specify a name for the index, but it is ignored, and the name of the index is always `PRIMARY`. A warning is explicitly issued if a name is specified. Before then, the name was silently ignored.

See Getting Started with Indexes: Primary Key for more information.

UNIQUE

The `UNIQUE` keyword means that the index will not accept duplicated values, except for NULLs. An error will raise if you try to insert duplicate values in a UNIQUE index.

For `UNIQUE` indexes, you can specify a name for the constraint, using the `CONSTRAINT` keyword. That name will be used in error messages.

> **CURRENT**
>
> > **Note:**
> >
> > Unique, if index type is not specified, is normally a BTREE index that can also be used by the optimizer to find rows. If the key is longer than the max key length for the used storage engine, a HASH key will be created. This enables MariaDB to enforce uniqueness for any type or number of columns.
>
> **< 10.5**
>
> -

See Getting Started with Indexes: Unique Index for more information.

FOREIGN KEY

For `FOREIGN KEY` indexes, a reference definition must be provided.

For `FOREIGN KEY` indexes, you can specify a name for the constraint, using the `CONSTRAINT` keyword. That name will be used in error messages.

First, you have to specify the name of the target (parent) table and a column or a column list which must be indexed and whose values must match to the foreign key’s values. The `MATCH` clause is accepted to improve the compatibility with other DBMS’s, but has no meaning in MariaDB. The `ON DELETE` and `ON UPDATE` clauses specify what must be done when a `DELETE` (or a `REPLACE`) statements attempts to delete a referenced row from the parent table, and when an `UPDATE` statement attempts to modify the referenced foreign key columns in a parent table row, respectively. The following options are allowed:

- `RESTRICT`: The delete/update operation is not performed. The statement terminates with a 1451 error (SQLSTATE ‘2300’).
- `NO ACTION`: Synonym for `RESTRICT`.
- `CASCADE`: The delete/update operation is performed in both tables.
- `SET NULL`: The update or delete goes ahead in the parent table, and the corresponding foreign key fields in the child table are set to `NULL`. (They must not be defined as `NOT NULL` for this to succeed).

- `SET DEFAULT`: This option is currently implemented only for the PBXT storage engine, which is disabled by default and no longer maintained. It sets the child table's foreign key fields to their `DEFAULT` values when the referenced parent table key entries are updated or deleted.

If either clause is omitted, the default behavior for the omitted clause is `RESTRICT`.

See Foreign Keys for more information.

FULLTEXT

Use the `FULLTEXT` keyword to create full-text indexes.

See Full-Text Indexes for more information.

SPATIAL

Use the `SPATIAL` keyword to create geometric indexes.

See SPATIAL INDEX for more information.

Index Options

KEY_BLOCK_SIZE Index Option

The `KEY_BLOCK_SIZE` index option is similar to the KEY_BLOCK_SIZE table option.

With the InnoDB storage engine, if you specify a non-zero value for the `KEY_BLOCK_SIZE` table option for the whole table, then the table will implicitly be created with the ROW_FORMAT table option set to `COMPRESSED`. However, this does not happen if you just set the `KEY_BLOCK_SIZE` index option for one or more indexes in the table. The InnoDB storage engine ignores the `KEY_BLOCK_SIZE` index option. However, the SHOW CREATE TABLE statement may still report it for the index.

For information about the `KEY_BLOCK_SIZE` index option, see the KEY_BLOCK_SIZE table option below.

ADAPTIVE_HASH_INDEX Index Option

> **Note:**
>
> Added in **MariaDB 13.1.1** (MDEV-37070).

The `ADAPTIVE_HASH_INDEX` index option takes the same `DEFAULT`, `YES`, and `NO` values as the ADAPTIVE_HASH_INDEX table option, but applies to a single index. When set to `YES` or `NO`, it overrides the table-level setting for that index. This option applies only to InnoDB.

Advanced Adaptive Hash Index Tuning Options

> **Note:**
>
> Added in **MariaDB 13.1.1** (MDEV-37070).

Three additional InnoDB index options give fine-grained control over how the adaptive hash index is built for an index:

Option	Values	Effect
`COMPLETE_FIELDS`	`0` to the number of columns the index is defined on (maximum `64`)	Number of complete index columns to include in the hash.

Option	Values	Effect
`BYTES_FROM_INCOMPLETE_FIELD`	`0` to `16383`	Number of leading bytes to take from the next column, beyond those covered by `COMPLETE_FIELDS`. Only meaningful for `memcmp()`-comparable index fields such as `VARBINARY` or integer types. For example, a 3-byte prefix on an `INT` returns one hash value for 0..255, another for 256..511, and so on.
`FOR_EQUAL_HASH_POINT_TO_LAST_RECORD`	`DEFAULT`, `YES`, `NO`	For a set of records that share the same hash value, controls which record the hash entry points to: `NO` points to the first record, `YES` points to the last.

The default for all three is unset (automatic), in which case InnoDB chooses the values from its internal heuristic.

> **Warning:**
>
> These are power-user options and are not needed for typical workloads. They override InnoDB's internal heuristic, which is computed and *then* replaced by the values you supply. Because the three options work together, set **all three** to known-good values whenever you set even one of them — setting only one or two leaves the others to interact with the overridden heuristic and can produce unintended results.
>
> InnoDB does not expose the values its heuristic would otherwise compute, so suitable settings must be derived from prior knowledge of the data and confirmed through performance testing.

The main reason to fix these values is to avoid *adaptive-hash-index churn*: when an index's lookup pattern is not constant, the internal heuristic keeps changing its parameters, repeatedly destroying and rebuilding the hash index. Pinning the parameters keeps a single hash index in place — less optimal for some queries, but stable and beneficial for the rest, instead of being continually rebuilt.

These options apply only to InnoDB, and only on servers built with adaptive hash index support.

Index Types

Each storage engine supports some or all index types. See Storage Engine Index Types for details on permitted index types for each storage engine.

Different index types are optimized for different kind of operations:

- `BTREE` is the default type, and normally is the best choice. It is supported by all storage engines. It can be used to compare a column's value with a value using the =, >, >=, <, <=, `BETWEEN`, and `LIKE` operators. `BTREE` can also be used to find `NULL` values. Searches against an index prefix are possible.
- `HASH` is only supported by the MEMORY storage engine. `HASH` indexes can only be used for =, <=, and >= comparisons. It can not be used for the `ORDER BY` clause. Searches against an index prefix are not possible.
- `RTREE` is the default for SPATIAL indexes, but if the storage engine does not support it `BTREE` can be used.

> **CURRENT**
>
> Index columns names are listed between parenthesis. After each column, a prefix length can be specified. If no length is specified, the whole column will be indexed. `ASC` and `DESC` can be specified. Individual columns in the index can be explicitly sorted in ascending or descending order. This can be useful for optimizing certain ORDER BY cases (MDEV-13756, MDEV-26938, MDEV-26939, MDEV-26996). Not only ascending, but also descending, indexes can be used to optimize MIN() and MAX() (MDEV-27576).
>
> **< 11.4 / 10.8**
>
> Index columns names are listed between parenthesis. After each column, a prefix length can be specified. If no length is specified, the whole column will be indexed. `ASC` and `DESC` can be specified. Prior to MariaDB 10.8, this was only for compatibility with other DBMSs, but had no meaning in MariaDB. From MariaDB 10.8, individual columns in the index can now be explicitly sorted in ascending or descending order. This can be useful for

> optimizing certain ORDER BY cases (MDEV-13756, MDEV-26938, MDEV-26939, MDEV-26996). From MariaDB 11.4.0, not only ascending, but also descending, indexes can now be used to optimize MIN() and MAX() (MDEV-27576).

The maximum number of parts in an index is 32.

WITH PARSER Index Option

The `WITH PARSER` index option only applies to FULLTEXT indexes and contains the fulltext parser name. The fulltext parser must be an installed plugin.

VISIBLE Index Option

> **CURRENT**
>
> Indexes can be declared visible. This is the default and it shows up in SHOW CREATE TABLE.
>
> **< 10.5.3**
>
> Indexes cannot be declared visible.

COMMENT Index Option

A comment of up to 1024 characters is permitted with the `COMMENT` index option.

The `COMMENT` index option allows you to specify a comment with user-readable text describing what the index is for. This information is not used by the server itself.

IGNORED / NOT IGNORED

> **CURRENT**
>
> Indexes can be specified to be ignored by the optimizer. See Ignored Indexes.
>
> **< 10.6.0**
>
> Indexes can be specified to be ignored by the optimizer. See Ignored Indexes.

Periods

```
period_definition:
    PERIOD FOR [time_period_name | SYSTEM_TIME] (start_column_name, end_column_name)
```

MariaDB supports System-versioned tables, Application-time-period tables or Bitemporal Tables.

Constraint Expressions

MariaDB introduced two ways to define a constraint:

- `CHECK(expression)` given as part of a column definition.
- `CONSTRAINT [constraint_name] CHECK (expression)`

Before a row is inserted or updated, all constraints are evaluated in the order they are defined. If any constraints fails, then the row will not be updated.
One can use most deterministic functions in a constraint, including UDFs.

```
CREATE TABLE t1 (a INT CHECK(a>0) ,b INT CHECK (b> 0), CONSTRAINT abc CHECK (a>b));
```

If you use the second format and you don't give a name to the constraint, then the constraint will get a auto generated name. This is done so that you can later delete the constraint with ALTER TABLE DROP constraint_name.

One can disable all constraint expression checks by setting the variable `check_constraint_checks` to `OFF`. This is useful for example when loading a table that violates some constraints that you want to later find and fix in SQL.

See CONSTRAINT for more information.

Table Options

For each individual table you create (or alter), you can set some table options. The general syntax for setting options is:

```
<OPTION_NAME> = <option_value>, [<OPTION_NAME> = <option_value> ...]
```

The equal sign is optional.

Some options are supported by the server and can be used for all tables, no matter what storage engine they use; other options can be specified for all storage engines, but have a meaning only for some engines. Also, engines can extend CREATE TABLE with new options.

If the `IGNORE_BAD_TABLE_OPTIONS` SQL_MODE is enabled, wrong table options generate a warning; otherwise, they generate an error.

```
table_option:
    [STORAGE] ENGINE [=] engine_name
  | ADAPTIVE_HASH_INDEX [=] {DEFAULT | YES | NO}
  | AUTO_INCREMENT [=] number
  | AVG_ROW_LENGTH [=] number
  | [DEFAULT] CHARACTER SET [=] <a data-footnote-ref href="#user-content-fn-7">charset_name</a>
  | CHECKSUM [=] {0 | 1}
  | [DEFAULT] COLLATE [=] <a data-footnote-ref href="#user-content-fn-7">collation_name</a>
  | COMMENT [=] 'string'
  | CONNECTION [=] 'connect_string'
  | DATA DIRECTORY [=] 'absolute path to directory'
  | DELAY_KEY_WRITE [=] {0 | 1}
  | ENCRYPTED [=] {YES | NO}
  | ENCRYPTION_KEY_ID [=] number
  | IETF_QUOTES [=] {YES | NO}
  | INDEX DIRECTORY [=] 'absolute path to directory'
  | INSERT_METHOD [=] { NO | FIRST | LAST }
  | KEY_BLOCK_SIZE [=] number
  | MAX_ROWS [=] number
  | MIN_ROWS [=] number
  | PACK_KEYS [=] {0 | 1 | DEFAULT}
  | PAGE_CHECKSUM [=] {0 | 1}
  | PAGE_COMPRESSED [=] {0 | 1}
  | PAGE_COMPRESSION_LEVEL [=] {0 .. 9}
  | PASSWORD [=] 'string'
  | ROW_FORMAT [=] {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT|PAGE}
  | SEQUENCE [=] {0|1}
  | STATS_AUTO_RECALC [=] {DEFAULT|0|1}
  | STATS_PERSISTENT [=] {DEFAULT|0|1}
  | STATS_SAMPLE_PAGES [=] {DEFAULT|number}
  | TABLESPACE tablespace_name
  | TRANSACTIONAL [=]  {0 | 1}
  | UNION [=] (tbl_name[,tbl_name]...)
  | WITH SYSTEM VERSIONING
```

[STORAGE] ENGINE

`[STORAGE] ENGINE` specifies a storage engine for the table. If this option is not used, the default storage engine is used instead. That is, the default_storage_engine session option value if it is set, or the value specified for the `--default-storage-engine` mariadbd startup option, or the default storage engine, InnoDB. If the specified storage engine is not installed and active, the default value will be used, unless the `NO_ENGINE_SUBSTITUTION` SQL MODE is set (default). This is only true for `CREATE TABLE`, not for `ALTER TABLE`. For a list of storage engines that are present in your server, issue a SHOW ENGINES.

ADAPTIVE_HASH_INDEX

> **Note:**
>
> The `ADAPTIVE_HASH_INDEX` table and index options were added in **MariaDB 13.1.1** (MDEV-37070).

`ADAPTIVE_HASH_INDEX` controls whether the InnoDB adaptive hash index (AHI) is used for an individual InnoDB table. It takes one of three values:

- `DEFAULT` — no per-table preference: the table follows the global innodb_adaptive_hash_index setting. This is the default, and removes the option from the stored table definition.
- `YES` — request the adaptive hash index for this table. AHI is built for the table only when it is also enabled at the server level (`innodb_adaptive_hash_index` set to `ON` or `IF_SPECIFIED`).
- `NO` — never use the adaptive hash index for this table, even when it is enabled at the server level.

The option only affects InnoDB tables, and only on servers built with adaptive hash index support. The same option can be set per index (see ADAPTIVE_HASH_INDEX Index Option); an index-level `YES` or `NO` overrides the table-level setting for that index.

AUTO_INCREMENT

`AUTO_INCREMENT` specifies the initial value for the AUTO_INCREMENT primary key. This works for MyISAM, Aria, InnoDB, MEMORY, and ARCHIVE tables. You can change this option with `ALTER TABLE`, but in that case the new value must be higher than the highest value which is present in the `AUTO_INCREMENT` column. If the storage engine does not support this option, you can insert (and then delete) a row having the wanted value - 1 in the `AUTO_INCREMENT` column.

AVG_ROW_LENGTH

`AVG_ROW_LENGTH` is the average rows size. It only applies to tables using MyISAM and Aria storage engines that have the ROW_FORMAT table option set to `FIXED` format.

MyISAM uses `MAX_ROWS` and `AVG_ROW_LENGTH` to decide the maximum size of a table (default: 256TB, or the maximum file size allowed by the system).

[DEFAULT] CHARACTER SET/CHARSET

`[DEFAULT] CHARACTER SET` (or `[DEFAULT] CHARSET`) is used to set a default character set for the table. This is the character set used for all columns where an explicit character set is not specified. If this option is omitted or `DEFAULT` is specified, the database's default character set will be used (except for the JSON data type, which is utf8mb4 by default). See Setting Character Sets and Collations for details on setting the character sets.

CHECKSUM/TABLE_CHECKSUM

`CHECKSUM` (or `TABLE_CHECKSUM`) can be set to 1 to maintain a live checksum for all table's rows. This makes write operations slower, but CHECKSUM TABLE will be very fast. This option is only supported for MyISAM and Aria tables.

[DEFAULT] COLLATE

`[DEFAULT] COLLATE` is used to set a default collation for the table. This is the collation used for all columns where an explicit character set is not specified. If this option is omitted or `DEFAULT` is specified, the database's default option will be used (except for the JSON data type, which uses utf8mb4_bin by default). See Setting Character Sets and Collations for details on setting the collations

COMMENT

`COMMENT` is a comment for the table. The maximum length is 2048 characters. Also used to define table parameters when creating a Spider table.

CONNECTION

`CONNECTION` is used to specify a server name or a connection string for a Spider, CONNECT, Federated or FederatedX table.

DATA DIRECTORY/INDEX DIRECTORY

`DATA DIRECTORY` and `INDEX DIRECTORY` are supported for MyISAM and Aria, and DATA DIRECTORY is also supported by InnoDB if the innodb_file_per_table server system variable is enabled, but only in CREATE TABLE, not in ALTER TABLE. So, carefully choose a path for InnoDB tables at creation time, because it cannot be changed without dropping and re-creating the table. These options specify the paths for data files and index files, respectively. If these options are omitted, the database's directory will be used to store data files and index files. Note that these table options do not work for partitioned tables (use the partition options instead), or if the server has been invoked with the –skip-symbolic-links startup option. To avoid the overwriting of old files with the same name that could be present in the directories, you can use the –keep_files_on_create option (an error will be issued if files already exist). These options are ignored if the `NO_DIR_IN_CREATE` SQL_MODE is enabled (useful for replicas). Also note that symbolic links cannot be used for InnoDB tables.

`DATA DIRECTORY` works by creating symlinks from where the table would normally have been (inside the datadir) to where the option specifies. For security reasons, to avoid bypassing the privilege system, the server does not permit symlinks inside the datadir. Therefore, `DATA DIRECTORY` cannot be used to specify a location inside the datadir. An attempt to do so will result in an error `1210 (HY000) Incorrect arguments to DATA DIRECTORY`.

DELAY_KEY_WRITE

`DELAY_KEY_WRITE` is supported by MyISAM and Aria, and can be set to 1 to speed up write operations. In that case, when data are modified, the indexes are not updated until the table is closed. Writing the changes to the index file altogether can be much faster. However, note that this option is applied only if the `delay_key_write` server variable is set to 'ON'. If it is 'OFF' the delayed index writes are always disabled, and if it is 'ALL' the delayed index writes are always used, disregarding the value of `DELAY_KEY_WRITE`.

ENCRYPTED

The `ENCRYPTED` table option can be used to manually set the encryption status of an InnoDB table. See InnoDB Encryption for more information.

Aria does not support the `ENCRYPTED` table option. See MDEV-18049.

See Data-at-Rest Encryption for more information.

ENCRYPTION_KEY_ID

The `ENCRYPTION_KEY_ID` table option can be used to manually set the encryption key of an InnoDB table. See InnoDB Encryption for more information.

Aria does not support the `ENCRYPTION_KEY_ID` table option. See MDEV-18049.

See Data-at-Rest Encryption for more information.

IETF_QUOTES

For the CSV storage engine, the `IETF_QUOTES` option, when set to `YES`, enables IETF-compatible parsing of embedded quote and comma characters. Enabling this option for a table improves compatibility with other tools that use CSV, but is not compatible with MySQL CSV tables, or MariaDB CSV tables created without this option. Disabled by default.

INSERT_METHOD

`INSERT_METHOD` is only used with MERGE tables. This option determines in which underlying table the new rows should be inserted. If you set it to 'NO' (which is the default) no new rows can be added to the table (but you will still be able to perform `INSERT`s directly against the underlying tables). `FIRST` means that the rows are inserted into the first table, and `LAST` means that they are inserted into the last table.

KEY_BLOCK_SIZE

`KEY_BLOCK_SIZE` is used to determine the size of key blocks, in bytes or kilobytes. However, this value is just a hint, and the storage engine could modify or ignore it. If `KEY_BLOCK_SIZE` is set to 0, the storage engine's default value will be used.

With the InnoDB storage engine, if you specify a non-zero value for the `KEY_BLOCK_SIZE` table option for the whole table, then the table will implicitly be created with the ROW_FORMAT table option set to `COMPRESSED`.

MIN_ROWS/MAX_ROWS

`MIN_ROWS` and `MAX_ROWS` let the storage engine know how many rows you are planning to store as a minimum and as a maximum. These values will not be used as real limits, but they help the storage engine to optimize the table. `MIN_ROWS` is only used by MEMORY storage engine to decide the minimum memory that is always allocated. `MAX_ROWS` is used to decide the minimum size for indexes.

PACK_KEYS

`PACK_KEYS` can be used to determine whether the indexes will be compressed. Set it to 1 to compress all keys. With a value of 0, compression will not be used. With the `DEFAULT` value, only long strings will be compressed. Uncompressed keys are faster.

PAGE_CHECKSUM

`PAGE_CHECKSUM` is only applicable to Aria tables, and determines whether indexes and data should use page checksums for extra safety.

PAGE_COMPRESSED

`PAGE_COMPRESSED` is used to enable InnoDB page compression for InnoDB tables.

PAGE_COMPRESSION_LEVEL

`PAGE_COMPRESSION_LEVEL` is used to set the compression level for InnoDB page compression for InnoDB tables. The table must also have the PAGE_COMPRESSED table option set to `1`.

Valid values for `PAGE_COMPRESSION_LEVEL` are 1 (the best speed) through 9 (the best compression), .

PASSWORD

`PASSWORD` is unused.

RAID_TYPE

`RAID_TYPE` is an obsolete option, as the raid support has been disabled since MySQL 5.0.

ROW_FORMAT

The `ROW_FORMAT` table option specifies the row format for the data file. Possible values are engine-dependent.

Supported MyISAM Row Formats

For MyISAM, the supported row formats are:

- `FIXED`
- `DYNAMIC`
- `COMPRESSED`

The `COMPRESSED` row format can only be set by the myisampack command line tool.

See MyISAM Storage Formats for more information.

Supported Aria Row Formats

For Aria, the supported row formats are:

- `PAGE`
- `FIXED`
- `DYNAMIC`.

See Aria Storage Formats for more information.

Supported InnoDB Row Formats

For InnoDB, the supported row formats are:

- `COMPACT`
- `REDUNDANT`
- `COMPRESSED`
- `DYNAMIC`.

If the `ROW_FORMAT` table option is set to `FIXED` for an InnoDB table, then the server will either return an error or a warning depending on the value of the innodb_strict_mode system variable. If the innodb_strict_mode system variable is set to `OFF`, then a warning is issued, and MariaDB will create the table using the default row format for the specific MariaDB server version. If the innodb_strict_mode system variable is set to `ON`, then an error will be raised.

See InnoDB Storage Formats for more information.

Other Storage Engines and ROW_FORMAT

Other storage engines do not support the `ROW_FORMAT` table option.

SEQUENCE

If the table is a sequence, then it will have the `SEQUENCE` set to `1`.

STATS_AUTO_RECALC

`STATS_AUTO_RECALC` indicates whether to automatically recalculate persistent statistics (see `STATS_PERSISTENT`, below) for an InnoDB table.
If set to `1`, statistics will be recalculated when more than 10% of the data has changed. When set to `0`, stats will be

recalculated only when an ANALYZE TABLE is run. If set to `DEFAULT`, or left out, the value set by the innodb_stats_auto_recalc system variable applies. See InnoDB Persistent Statistics.

STATS_PERSISTENT

`STATS_PERSISTENT` indicates whether the InnoDB statistics created by ANALYZE TABLE will remain on disk or not. It can be set to `1` (on disk), `0` (not on disk, the pre-MariaDB 10 behavior), or `DEFAULT` (the same as leaving out the option), in which case the value set by the innodb_stats_persistent system variable will apply. Persistent statistics stored on disk allow the statistics to survive server restarts, and provide better query plan stability. See InnoDB Persistent Statistics.

STATS_SAMPLE_PAGES

`STATS_SAMPLE_PAGES` indicates how many pages are used to sample index statistics. If 0 or DEFAULT, the default value, the innodb_stats_sample_pages value is used. See InnoDB Persistent Statistics.

TRANSACTIONAL

`TRANSACTIONAL` is only applicable for Aria tables. In future Aria tables created with this option will be fully transactional, but currently this provides a form of crash protection. See Aria Storage Engine for more details.

UNION

`UNION` must be specified when you create a MERGE table. This option contains a comma-separated list of MyISAM tables which are accessed by the new table. The list is enclosed between parenthesis. Example: `UNION = (t1,t2)`

WITH SYSTEM VERSIONING

`WITH SYSTEM VERSIONING` is used for creating System-versioned tables.

Partitions

```
partition_options:
    PARTITION BY
        { [LINEAR] HASH(expr)
        | [LINEAR] KEY(column_list)
        | RANGE(expr)
        | LIST(expr)
        | SYSTEM_TIME [INTERVAL time_quantity <a data-footnote-ref href="#user-content-fn-
8">time_unit</a>] [LIMIT num] }
    [PARTITIONS num]
    [SUBPARTITION BY
        { [LINEAR] HASH(expr)
        | [LINEAR] KEY(column_list) }
      [SUBPARTITIONS num]
    ]
    [(partition_definition [, partition_definition] ...)]

partition_definition:
    [PARTITION] partition_name
        [VALUES {LESS THAN {(expr) | MAXVALUE} | IN (value_list)}]
        [[STORAGE] ENGINE [=] engine_name]
        [COMMENT [=] 'comment_text' ]
        [DATA DIRECTORY [=] 'data_dir']
        [INDEX DIRECTORY [=] 'index_dir']
        [MAX_ROWS [=] max_number_of_rows]
        [MIN_ROWS [=] min_number_of_rows]
        [TABLESPACE [=] tablespace_name]
        [NODEGROUP [=] node_group_id]
        [(subpartition_definition [, subpartition_definition] ...)]

subpartition_definition:
    SUBPARTITION logical_name
        [[STORAGE] ENGINE [=] engine_name]
        [COMMENT [=] 'comment_text' ]
        [DATA DIRECTORY [=] 'data_dir']
        [INDEX DIRECTORY [=] 'index_dir']
        [MAX_ROWS [=] max_number_of_rows]
        [MIN_ROWS [=] min_number_of_rows]
        [TABLESPACE [=] tablespace_name]
        [NODEGROUP [=] node_group_id]
```

If the `PARTITION BY` clause is used, the table will be partitioned. A partition method must be explicitly indicated for partitions and subpartitions. Partition methods are:

- `[LINEAR] HASH` creates a hash key which will be used to read and write rows. The partition function can be any valid SQL expression which returns an `INTEGER` number. Thus, it is possible to use the `HASH` method on an integer column, or on functions which accept integer columns as an argument. However, `VALUES LESS THAN` and `VALUES IN` clauses can not be used with `HASH`. An example:

```
CREATE TABLE t1 (a INT, b CHAR(5), c DATETIME)
    PARTITION BY HASH ( YEAR(c) );
```

`[LINEAR] HASH` can be used for subpartitions, too.

- `[LINEAR] KEY` is similar to `HASH`, but the index has an even distribution of data. Also, the expression can only be a column or a list of columns. `VALUES LESS THAN` and `VALUES IN` clauses can not be used with `KEY`.
- RANGE partitions the rows using on a range of values, using the `VALUES LESS THAN` operator. `VALUES IN` is not allowed with `RANGE`. The partition function can be any valid SQL expression which returns a single value.
- LIST assigns partitions based on a table's column with a restricted set of possible values. It is similar to `RANGE`, but `VALUES IN` must be used for at least 1 columns, and `VALUES LESS THAN` is disallowed.

- `SYSTEM_TIME` partitioning is used for System-versioned tables to store historical data separately from current data.

Only HASH and KEY can be used for subpartitions, and they can be `[LINEAR]`.

It is possible to define up to 8092 partitions and subpartitions.

The number of defined partitions can be optionally specified as `PARTITION count`. This can be done to avoid specifying all partitions individually. But you can also declare each individual partition and, additionally, specify a `PARTITIONS count` clause; in the case, the number of `PARTITION`s must equal count.

Also see Partitioning Types Overview.

> **CURRENT**
>
> The `PARTITION` keyword is optional as part of the partition definition. Instead of this:
>
> ```
> CREATE OR REPLACE TABLE t1 (x INT)
>   PARTITION BY RANGE(x) (
>     PARTITION p1 VALUES LESS THAN (10),
>     PARTITION p2 VALUES LESS THAN (20),
>     PARTITION p3 VALUES LESS THAN (30),
>     PARTITION p4 VALUES LESS THAN (40),
>     PARTITION p5 VALUES LESS THAN (50),
>     PARTITION pn VALUES LESS THAN MAXVALUE);
> ```
>
> The following can be used:
>
> ```
> CREATE OR REPLACE TABLE t1 (x INT)
>   PARTITION BY RANGE(x) (
>     p1 VALUES LESS THAN (10),
>     p2 VALUES LESS THAN (20),
>     p3 VALUES LESS THAN (30),
>     p4 VALUES LESS THAN (40),
>     p5 VALUES LESS THAN (50),
>     pn VALUES LESS THAN MAXVALUE);
> ```
>
> **< 10.7.1**
>
> The `PARTITION` keyword is not optional as part of the partition definition. You must use this syntax:
>
> ```
> CREATE OR REPLACE TABLE t1 (x INT)
>   PARTITION BY RANGE(x) (
>     PARTITION p1 VALUES LESS THAN (10),
>     PARTITION p2 VALUES LESS THAN (20),
>     PARTITION p3 VALUES LESS THAN (30),
>     PARTITION p4 VALUES LESS THAN (40),
>     PARTITION p5 VALUES LESS THAN (50),
>     PARTITION pn VALUES LESS THAN MAXVALUE);
> ```

Sequences

`CREATE TABLE` can also be used to create a SEQUENCE. See CREATE SEQUENCE and Sequence Overview.

Atomic DDL

> **CURRENT**
>
> MariaDB supports Atomic DDL. `CREATE TABLE` is atomic, except for `CREATE OR REPLACE`, which are only crash-safe.
>
> **< 10.6.1**

-

Examples

```
CREATE TABLE IF NOT EXISTS test (
a BIGINT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(128) CHARSET utf8,
KEY name (name(32))
) ENGINE=InnoDB DEFAULT CHARSET latin1;
```

This example shows a couple of things:

- Usage of `IF NOT EXISTS`; If the table already existed, it will not be created. There will not be any error for the client, just a warning.
- How to create a `PRIMARY KEY` that is automatically generated.
- How to specify a table-specific character set and another for a column.
- How to create an index (`name`) that is only partly indexed (to save space).

The following clauses will work:

```
CREATE TABLE t1(
  a INT DEFAULT (1+1),
  b INT DEFAULT (a+1),
  expires DATETIME DEFAULT(NOW() + INTERVAL 1 YEAR),
  x BLOB DEFAULT USER()
);
```

See Also

- Identifier Names
- ALTER TABLE
- DROP TABLE
- Character Sets and Collations
- SHOW CREATE TABLE
- CREATE TABLE with Vectors
- Storage engines can add their own attributes for columns, indexes and tables
- Variable slave-ddl-exec-mode
- InnoDB Limitations

This page is licensed: GPLv2, originally from fill_help_tables.sql

DROP TABLE

Syntax

```
DROP [TEMPORARY] TABLE [IF EXISTS] [/*COMMENT TO SAVE*/]
    tbl_name [, tbl_name] ...
    [WAIT n|NOWAIT]
    [RESTRICT | CASCADE]
```

Description

`DROP TABLE` removes one or more tables. You must have the `DROP` privilege for each table. All table data and the table definition are removed, as well as triggers associated to the table, so be careful with this statement! If any of

the tables named in the argument list do not exist, MariaDB returns an error indicating by name which non-existing tables it was unable to drop, but it also drops all of the tables in the list that do exist.

Important: When a table is dropped, user privileges on the table are not automatically dropped. See GRANT.

If another thread is using the table in an explicit transaction or an autocommit transaction, then the thread acquires a metadata lock (MDL) on the table. The `DROP TABLE` statement will wait in the "Waiting for table metadata lock" thread state until the MDL is released. MDLs are released in the following cases:

- If an MDL is acquired in an explicit transaction, then the MDL will be released when the transaction ends.
- If an MDL is acquired in an autocommit transaction, then the MDL will be released when the statement ends.
- Transactional and non-transactional tables are handled the same.

Note that for a partitioned table, `DROP TABLE` permanently removes the table definition, all of its partitions, and all of the data which was stored in those partitions. It also removes the partitioning definition (.par) file associated with the dropped table.

For each referenced table, `DROP TABLE` drops a temporary table with that name, if it exists. If it does not exist, and the `TEMPORARY` keyword is not used, it drops a non-temporary table with the same name, if it exists. The `TEMPORARY` keyword ensures that a non-temporary table will not accidentally be dropped.

Use `IF EXISTS` to prevent an error from occurring for tables that do not exist. A `NOTE` is generated for each non-existent table when using `IF EXISTS`. See SHOW WARNINGS.

If a foreign key references this table, the table cannot be dropped. In this case, it is necessary to drop the foreign key first.

`RESTRICT` and `CASCADE` are allowed to make porting from other database systems easier. In MariaDB, they do nothing.

The comment before the table names (`/*COMMENT TO SAVE*/`) is stored in the binary log. That feature can be used by replication tools to send their internal messages.

It is possible to specify table names as `db_name.tab_name`. This is useful to delete tables from multiple databases with one statement. See Identifier Qualifiers for details.

The DROP privilege is required to use `DROP TABLE` on non-temporary tables. For temporary tables, no privilege is required, because such tables are only visible for the current session.

Note: `DROP TABLE` automatically commits the current active transaction, unless you use the `TEMPORARY` keyword.

> **CURRENT**
>
> `DROP TABLE` reliably deletes table remnants inside a storage engine even if the `.frm` file is missing.
>
> **< 10.5.4**
>
> `DROP TABLE` does **not** reliably delete table remnants inside a storage engine even if the `.frm` file is missing. A missing `.frm` file will result in the statement failing.

WAIT/NOWAIT

Set the lock wait timeout. See WAIT and NOWAIT.

DROP TABLE in replication

`DROP TABLE` has the following characteristics in replication:

- `DROP TABLE IF EXISTS` are always logged.

- `DROP TABLE` without `IF EXISTS` for tables that don't exist are not written to the binary log.
- Dropping of `TEMPORARY` tables are prefixed in the log with `TEMPORARY`. These drops are only logged when running statement or mixed mode replication.
- One `DROP TABLE` statement can be logged with up to 3 different `DROP` statements:
  - `DROP TEMPORARY TABLE list_of_non_transactional_temporary_tables`
  - `DROP TEMPORARY TABLE list_of_transactional_temporary_tables`
  - `DROP TABLE list_of_normal_tables`

`DROP TABLE` on the primary is treated on the replica as `DROP TABLE IF EXISTS`. You can change that by setting slave-ddl-exec-mode to `STRICT`.

Dropping an Internal #sql-… Table

> **CURRENT**
>
> DROP TABLE is atomic.
>
> **< 10.6**
>
> if the mariadbd process is killed during an ALTER TABLE, you may find a table named #sql-… in your data directory. These temporary tables will always be deleted automatically.
>
> If you want to delete one of these tables explicitly you can do so by using the following syntax:
>
> ```
> DROP TABLE `#mysql50##sql-...`;
> ```
>
> When running an `ALTER TABLE…ALGORITHM=INPLACE` that rebuilds the table, InnoDB will create an internal `#sql-ib` table.
>
> The same name as the .frm file is used for the intermediate copy of the table. The #sql-ib names are used by TRUNCATE and delayed DROP.
>
> The #sql-ib tables will be deleted automatically.

Dropping All Tables in a Database

The best way to drop all tables in a database is by executing DROP DATABASE, which will drop the database itself, and all tables in it.

However, if you want to drop all tables in the database, but you also want to keep the database itself and any other non-table objects in it, then you would need to execute `DROP TABLE` to drop each individual table. You can construct these `DROP TABLE` commands by querying the TABLES table in the information_schema database. For example:

```
SELECT CONCAT('DROP TABLE IF EXISTS `', TABLE_SCHEMA, '`.`', TABLE_NAME, '`;')
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'mydb';
```

Atomic DROP TABLE

MariaDB starting with 10.6.1

> **CURRENT**
>
> `DROP TABLE` for a single table is atomic (MDEV-25180) for most engines, including InnoDB, MyRocks, MyISAM and Aria. This means that if there is a crash (server down or power outage) during `DROP TABLE`, all tables that have been processed so far will be completely dropped, including related trigger files and status entries, and the binary log will include a `DROP TABLE` statement for the dropped tables. Tables for which the drop had not started

will be left intact. `DROP TABLE` was extended to be able to delete a table that was only partly dropped (MDEV-11412), as explained above. Atomic `DROP TABLE` is the final piece to make `DROP TABLE` fully reliable. Dropping multiple tables is crash-safe. See Atomic DDL for more information.

< 10.6.1

There is a small chance that, during a server crash happening in the middle of `DROP TABLE`, some storage engines that were using multiple storage files, like MyISAM, could have only a part of its internal files dropped. In MariaDB 10.5, `DROP TABLE` was extended to be able to delete a table that was only partly dropped (MDEV-11412) as explained above. Atomic `DROP TABLE` is the final piece to make `DROP TABLE` fully reliable. Dropping multiple tables is crash-safe. See Atomic DDL for more information.

Examples

```
DROP TABLE Employees, Customers;
```

Notes

Beware that `DROP TABLE` can drop both tables and sequences. This is mainly done to allow old tools like mariadb-dump (previously mysqldump) to work with sequences.

See Also

- CREATE TABLE
- ALTER TABLE
- SHOW CREATE TABLE
- DROP SEQUENCE
- Variable slave-ddl-exec-mode.

This page is licensed: GPLv2, originally from fill_help_tables.sql

Data Types

See Data Type Storage Requirements.

Reference for storage space usage. This page lists the disk space consumed by each data type, helping with capacity planning and schema optimization.

See AUTO_INCREMENT.

Complete AUTO_INCREMENT data type guide for MariaDB. Complete reference for syntax, valid values, storage requirements, and range limits for production use.

See AUTO_INCREMENT FAQ.

Answers to common questions about AUTO_INCREMENT. Learn about gaps in sequences, resetting values, and behavior in different storage engines.

See NULL Values.

Understand the concept of NULL. This page explains how NULL represents missing or unknown data and how it interacts with comparison operators and functions.

See ROW TYPE OF.

Declare row-based variables. This PL/SQL compatibility feature allows declaring variables that match the structure of a table row or cursor.

See SERIAL.

Alias for BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE. This shorthand data type is often used to define primary keys.

See TYPE OF.

Declare variables based on existing types. This feature allows defining variables or parameters that inherit the data type of a table column or another variable.

See Numeric Data Types.

Complete numeric types reference: INT/BIGINT ranges, DECIMAL(M,D) precision/scale, FLOAT/DOUBLE storage sizes, and numeric type selection guidelines.

See Date and Time Data Types.

Store temporal values. This section covers data types for dates, times, and timestamps, including DATE, DATETIME, TIMESTAMP, TIME, and YEAR.

See String Data Types.

Store text and binary data. This section covers character types like CHAR, VARCHAR, and TEXT, as well as binary types like BLOB and BINARY.

This page is licensed: CC BY-SA / Gnu FDL

Storage Engines

See Storage Engines Overview.

An introduction to MariaDB's pluggable storage engine architecture, highlighting key engines like InnoDB, MyISAM, and Aria for different workloads.

See Choosing the Right Storage Engine.

A guide to selecting the appropriate storage engine based on data needs, comparing features of general-purpose, columnar, and specialized engines.

See ARCHIVE.

The Archive storage engine is optimized for high-speed insertion and compression of large amounts of data, suitable for logging and auditing.

See ARIA.

Learn about the Aria storage engine in MariaDB Server. Understand its features, advantages, and use cases, particularly for crash-safe operations and transactional workloads.

See BLACKHOLE.

The BLACKHOLE storage engine discards all data written to it but records operations in the binary log, useful for replication filtering and testing.

See CONNECT.

The main characteristic of CONNECT is to enable accessing data scattered on a machine as if it was a centralized database.

See CSV.

The CSV storage engine stores data in text files using comma-separated values format, allowing easy data exchange with other applications.

See DuckDB.

DuckDB embedded as a MariaDB storage engine for analytical workloads. Experimental, at alpha maturity, and not built into MariaDB releases.

See FederatedX.

FederatedX is a storage engine that allows access to tables on remote MariaDB or MySQL servers as if they were local tables.

See InnoDB.

Discover InnoDB, the default storage engine for MariaDB Server. Learn about its transaction-safe capabilities, foreign key support, and high performance for demanding workloads.

See MEMORY.

The MEMORY storage engine stores tables in RAM for fast access, but data is lost upon server restart.

See MERGE.

The MERGE storage engine allows a collection of identical MyISAM tables to be treated as a single logical table, useful for managing large datasets.

See Mroonga.

Mroonga (formerly named Groonga Storage Engine) is a storage engine that provides fast CJK-ready full text searching using column store.

See MyISAM.

Explore the MyISAM storage engine in MariaDB Server. Understand its characteristics, including suitability for read-heavy workloads, and its role in specific use cases.

See MyRocks.

Learn about the MyRocks storage engine in MariaDB Server. Discover its advantages for flash storage, high write throughput, and compression efficiency in modern database deployments.

See OQGRAPH.

Explore the OQGRAPH storage engine in MariaDB Server. Learn how to efficiently manage hierarchical and complex graph data structures, perfect for social networks and bill of materials.

See PERFORMANCE_SCHEMA.

While technically a storage engine, PERFORMANCE_SCHEMA provides a way to inspect internal server execution details at a low level.

See SEQUENCE Storage Engine.

The Sequence engine generates virtual tables of number sequences on the fly, useful for generating series of integers without storing data.

See S3 Storage Engine.

Integrate MariaDB Server with Amazon S3 using the S3 Storage Engine. Learn how to store and retrieve data directly from cloud object storage for scalability and cost efficiency.

See SphinxSE.

Integrate MariaDB Server with Sphinx for advanced full-text search. The Sphinx storage engine allows you to query external Sphinx indexes directly from your database.

See Spider.

Explore the Spider storage engine in MariaDB Server. Learn how to shard data across multiple MariaDB and MySQL servers, enabling horizontal scaling and distributed database solutions.

See VIDEX Storage Engine.

The VIDEX storage engine is an aggregated, extensible engine suitable for what-if analyses in MariaDB. The name is derived from [VI]rtual in[DEX].

See Converting Tables from MyISAM to InnoDB.

This guide outlines the benefits and process of migrating tables from MyISAM to InnoDB, highlighting key differences like transaction support and foreign keys.

See Machine Learning with MindsDB.

Learn how to integrate MindsDB with MariaDB to train and query machine learning models directly using standard SQL commands.

See FEDERATED Storage Engine.

The legacy FEDERATED storage engine, superseded by the actively maintained FederatedX fork.

This page is licensed: CC BY-SA / Gnu FDL

Partitioning Tables

See Partitioning Overview.

Complete Partitioning Overview guide for MariaDB. Complete reference documentation for implementation, configuration, and usage for production use.

See Partition Pruning and Selection.

Understand how the optimizer automatically prunes irrelevant partitions and how to explicitly select partitions in your queries for efficiency.

See Partition Maintenance.

Discover administrative tasks for managing partitions, such as adding, dropping, reorganizing, and coalescing them to keep your data optimized.

See Partitioning Types Overview.

An introduction to the various partitioning strategies available in MariaDB, helping you choose the right method for your data distribution needs. For a complete list of partitioning types, see this page.

See Partitioning Limitations.

This page outlines constraints when using partitioning, such as the maximum number of partitions and restrictions on foreign keys and query cache usage.

See Partitions Files.

Learn how MariaDB stores partitioned tables on the filesystem, typically creating separate .ibd files for each partition when using InnoDB.

See Partitions Metadata.

Understand how to retrieve metadata about partitions using the `INFORMATION_SCHEMA.PARTITIONS` table to monitor row counts and storage usage.

This page is licensed: CC BY-SA / Gnu FDL

Partitioning Overview

In MariaDB, a table can be split in smaller subsets. Both data and indexes are partitioned.

Uses for Partitioning

There can be several reasons to use this feature:

- If you often need to delete a large set of rows, such as all rows for a given year, using partitions can help, as dropping a partition with many rows is very fast, while deleting a lot of rows can be very slow.
- Very large tables and indexes can be slow even with optimized queries. But if the target table is partitioned, queries that read a small number of partitions can be much faster. However, this means that the queries have to be written carefully in order to only access a given set of partitions.
- Partitioning allows one to distribute files over multiple storage devices. For example, we can have historical data on slower, larger disks (historical data are not supposed to be frequently read); and current data can be on faster disks, or SSD devices.
- In case we separate historical data from recent data, we will probably need to take regular backups of one partition, not the whole table.

Partitioning Types

When partitioning a table, the use should decide:

- a partitioning type;
- a partitioning expression.

A partitioning type is the method used by MariaDB to decide how rows are distributed over existing partitions. Choosing the proper partitioning type is important to distribute rows over partitions in an efficient way.

With some partitioning types, a partitioning expression is also required. A partitioning function is an SQL expression returning an integer or temporal value, used to determine which partition will contain a given row. The partitioning expression is used for all reads and writes on involving the partitioned table, thus it should be fast.

MariaDB supports the following partitioning types:

- RANGE
- LIST
- RANGE COLUMNS and LIST COLUMNS
- HASH
- LINEAR HASH
- KEY
- LINEAR KEY
- SYSTEM_TIME

Enabling Partitioning

By default, MariaDB permits partitioning. You can determine this by using the SHOW PLUGINS statement, for example:

```
SHOW PLUGINS;
...
| Aria                          | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| FEEDBACK                      | DISABLED | INFORMATION SCHEMA | NULL    | GPL     |
| partition                     | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
+-------------------------------+----------+--------------------+---------+---------+
```

If partition is listed as DISABLED:

```
| partition                     | DISABLED | STORAGE ENGINE     | NULL    | GPL     |
+-------------------------------+----------+--------------------+---------+---------+
```

MariaDB has either been built without partitioning support, or has been started with the –skip-partition option, or one of its variants:

```
--skip-partition
--disable-partition
--partition=OFF
```

and you will not be able to create partitions.

Using Partitions

It is possible to create a new partitioned table using CREATE TABLE.

ALTER TABLE allows one to:

- Partition an existing table;
- Remove partitions from a partitioned table (with all data in the partition);
- Add/remove partitions, or reorganize them, as long as the partitioning function allows these operations (see below);
- Exchange a partition with a table;
- Perform administrative operations on some or all partitions (analyze, optimize, check, repair).

Adding Partitions

```
ADD PARTITION [IF NOT EXISTS] (partition_definition)
```

`ADD PARTITION` can be used to add partitions to an existing table:

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014),
  PARTITION p2 VALUES LESS THAN (2015),
  PARTITION p3 VALUES LESS THAN (2016)
);

ALTER TABLE t1 ADD PARTITION (
  PARTITION p4 VALUES LESS THAN (2017),
  PARTITION p5 VALUES LESS THAN (2018)
);
```

With RANGE partitions, it is only possible to add a partition to the high end of the range, not the low end. For example, the following results in an error:

```
ALTER TABLE t1 ADD PARTITION (
  PARTITION p0a VALUES LESS THAN (2012)
);
ERROR 1493 (HY000): VALUES LESS THAN value must be strictly increasing for each partition
```

You can work around this by using REORGANIZE PARTITION to split the partition instead. See Splitting Partitions.

Coalescing Partitions

```
COALESCE PARTITION number
```

ALTER TABLE is used to reduce the number of `HASH` or `KEY` partitions by the specified number. For example, given the following table with 5 partitions:

```
CREATE OR REPLACE TABLE t1 (v1 INT)
  PARTITION BY KEY (v1)
  PARTITIONS 5;
```

The following statement reduces the number of partitions by 2, leaving the table with 3 partitions:

```
ALTER TABLE t1 COALESCE PARTITION 2;
```

Converting Partitions to/from Tables

> **Note:**
>
> This feature is available from MariaDB 10.7.

```
CONVERT PARTITION partition_name TO TABLE tbl_name
CONVERT TABLE normal_table TO partition_definition
```

ALTER TABLE can be used to convert partitions in an existing table to a standalone table:

```
CREATE OR REPLACE TABLE t1 (
   dt DATETIME NOT NULL
 )
   ENGINE = InnoDB
   PARTITION BY RANGE (YEAR(dt))
   (
   PARTITION p0 VALUES LESS THAN (2013),
   PARTITION p1 VALUES LESS THAN (2014),
   PARTITION p2 VALUES LESS THAN (2015),
   PARTITION p3 VALUES LESS THAN (2016)
 );

INSERT INTO t1 VALUES ('2013-11-11'),('2014-11-11'),('2015-11-11');

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2013-11-11 00:00:00 |
| 2014-11-11 00:00:00 |
| 2015-11-11 00:00:00 |
+---------------------+

ALTER TABLE t1 CONVERT PARTITION p3 TO TABLE t2;

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2013-11-11 00:00:00 |
| 2014-11-11 00:00:00 |
+---------------------+

SELECT * FROM t2;
+--------------+
| dt           |
+--------------+
| 2015-11-11 00:00:00 |
+---------------------+

SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       TABLE: t1
CREATE TABLE: CREATE TABLE `t1` (
  `dt` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
 PARTITION BY RANGE (year(`dt`))
(PARTITION `p0` VALUES LESS THAN (2013) ENGINE = InnoDB,
 PARTITION `p1` VALUES LESS THAN (2014) ENGINE = InnoDB,
 PARTITION `p2` VALUES LESS THAN (2015) ENGINE = InnoDB)

SHOW CREATE TABLE t2\G
*************************** 1. row ***************************
       TABLE: t2
CREATE TABLE: CREATE TABLE `t2` (
  `dt` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
```

`CONVERT TABLE` does the reverse, converting a table into a partition:

```
ALTER TABLE t1 CONVERT TABLE t2 TO PARTITION p3 VALUES LESS THAN (2016);

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2013-11-11 00:00:00 |
| 2014-11-11 00:00:00 |
| 2015-11-11 00:00:00 |
+---------------------+
3 rows in set (0.001 sec)

SELECT * FROM t2;
ERROR 1146 (42S02): Table 'test.t2' doesn't exist

SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       TABLE: t1
CREATE TABLE: CREATE TABLE `t1` (
  `dt` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
 PARTITION BY RANGE (year(`dt`))
(PARTITION `p0` VALUES LESS THAN (2013) ENGINE = InnoDB,
 PARTITION `p1` VALUES LESS THAN (2014) ENGINE = InnoDB,
 PARTITION `p2` VALUES LESS THAN (2015) ENGINE = InnoDB,
 PARTITION `p3` VALUES LESS THAN (2016) ENGINE = InnoDB)
```

CONVERT TABLE ... WITH / WITHOUT VALIDATION

> **Note:**
>
> This feature is available from MariaDB 11.4.

When converting tables to a partition, validation is performed on each row to ensure it meets the partition requirements. This can be very slow in the case of larger tables. It is possible to disable this validation by specifying the `WITHOUT VALIDATION` option.

```
CONVERT TABLE normal_table TO partition_definition [{WITH | WITHOUT} VALIDATION]
```

`WITH VALIDATION` will result in the validation being performed, and is the default behaviour.

An alternative to convert partitions to tables is to use ALTER TABLE EXCHANGE PARTITION. This requires having to manually do the following steps:

1. Create an empty table with the same structure as the partition.
2. Exchange the table with the partition.
3. Drop the empty partition.

For example:

```
CREATE OR REPLACE TABLE t1 (
   dt DATETIME NOT NULL
 )
   ENGINE = InnoDB
   PARTITION BY RANGE (YEAR(dt))
   (
   PARTITION p0 VALUES LESS THAN (2013),
   PARTITION p1 VALUES LESS THAN (2014),
   PARTITION p2 VALUES LESS THAN (2015),
   PARTITION p3 VALUES LESS THAN (2016)
 );

INSERT INTO t1 VALUES ('2013-11-11'),('2014-11-11'),('2015-11-11');

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2013-11-11 00:00:00 |
| 2014-11-11 00:00:00 |
| 2015-11-11 00:00:00 |
+---------------------+

CREATE OR REPLACE TABLE t2 LIKE t1;

ALTER TABLE t2 REMOVE PARTITIONING;

ALTER TABLE t1 EXCHANGE PARTITION p3 WITH TABLE t2;

ALTER TABLE t1 DROP PARTITION p3;

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2013-11-11 00:00:00 |
| 2014-11-11 00:00:00 |
+---------------------+

SELECT * FROM t2;
+--------------+
| dt           |
+--------------+
| 2015-11-11 00:00:00 |
+---------------------+

SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       TABLE: t1
CREATE TABLE: CREATE TABLE `t1` (
  `dt` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
 PARTITION BY RANGE (year(`dt`))
(PARTITION `p0` VALUES LESS THAN (2013) ENGINE = InnoDB,
 PARTITION `p1` VALUES LESS THAN (2014) ENGINE = InnoDB,
 PARTITION `p2` VALUES LESS THAN (2015) ENGINE = InnoDB)

SHOW CREATE TABLE t2\G
*************************** 1. row ***************************
       TABLE: t2
CREATE TABLE: CREATE TABLE `t2` (
  `dt` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
```

Similarly, to do the reverse and convert a table into a partition `[ALTER TABLE](#pg-ence-sql-statements-data-definition-alter-alter-table-readme-a2aa59e4) ... EXCHANGE PARTITION` can also be used, with the following manual steps required:

- create the partition
- exchange the partition with the table
- drop the old table:

For example:

```
ALTER TABLE t1 ADD PARTITION (PARTITION p3 VALUES LESS THAN (2016));

ALTER TABLE t1 EXCHANGE PARTITION p3 WITH TABLE t2;

DROP TABLE t2;

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2013-11-11 00:00:00 |
| 2014-11-11 00:00:00 |
| 2015-11-11 00:00:00 |
+---------------------+

SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       TABLE: t1
CREATE TABLE: CREATE TABLE `t1` (
  `dt` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
 PARTITION BY RANGE (year(`dt`))
(PARTITION `p0` VALUES LESS THAN (2013) ENGINE = InnoDB,
 PARTITION `p1` VALUES LESS THAN (2014) ENGINE = InnoDB,
 PARTITION `p2` VALUES LESS THAN (2015) ENGINE = InnoDB,
 PARTITION `p3` VALUES LESS THAN (2016) ENGINE = InnoDB)
```

Dropping Partitions

```
DROP PARTITION [IF EXISTS] partition_names
```

ALTER TABLE DROP PARTITIONcan be used to drop specific partitions (and discard all data within the specified partitions) for RANGE and LIST partitions. It cannot be used on HASH or KEY partitions. To rather remove all partitioning, while leaving the data unaffected, see Removing Partitioning.

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014),
  PARTITION p2 VALUES LESS THAN (2015),
  PARTITION p3 VALUES LESS THAN (2016)
);

INSERT INTO t1 VALUES ('2012-11-15');
SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2012-11-15 00:00:00 |
+---------------------+

ALTER TABLE t1 DROP PARTITION p0;

SELECT * FROM t1;
Empty set (0.002 sec)
```

Exchanging Partitions

CURRENT

```
EXCHANGE PARTITION partition_name WITH TABLE tbl_name [{WITH | WITHOUT} VALIDATION]
```

< 11.4

```
EXCHANGE PARTITION partition_name WITH TABLE tbl_name
```

`ALTER TABLE t1 EXCHANGE PARTITION p1 WITH TABLE t2` allows to exchange a partition or subpartition with another table.

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014)
);

CREATE OR REPLACE TABLE t2 (
  dt DATETIME NOT NULL
) ENGINE = InnoDB;

INSERT INTO t1 VALUES ('2012-01-01'),('2013-01-01');

INSERT INTO t2 VALUES ('2013-02-02');

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2012-01-01 00:00:00 |
| 2013-01-01 00:00:00 |
+---------------------+

SELECT * FROM t2;
+--------------+
| dt           |
+--------------+
| 2013-02-02 00:00:00 |
+---------------------+

ALTER TABLE t1 EXCHANGE PARTITION p1 WITH TABLE t2;

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2012-01-01 00:00:00 |
| 2013-02-02 00:00:00 |
+---------------------+

SELECT * FROM t2;
+--------------+
| dt           |
+--------------+
| 2013-01-01 00:00:00 |
+---------------------+
```

The following requirements must be met:

- Table t1 must be partitioned, and table t2 cannot be partitioned.
- Table t2 cannot be a temporary table.
- Table t1 and t2 must otherwise be identical.
- Any existing row in t2 must match the conditions for storage in the exchanged partition p1 unless, from MariaDB 11.4, the `WITHOUT VALIDATION` option is specified.

By default, MariaDB performs the validation to see that each row meets the partition requirements, and the statement fails if a row does not fit.

This attempted exchange fails, as the value is already in t2, and `2015-05-05` is outside of the partition conditions:

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014)
);

CREATE OR REPLACE TABLE t2 (
  dt DATETIME NOT NULL
) ENGINE = InnoDB;

INSERT INTO t1 VALUES ('2012-02-02'),('2013-03-03');

INSERT INTO t2 VALUES ('2015-05-05');

ALTER TABLE t1 EXCHANGE PARTITION p1 WITH TABLE t2;
```

```
ERROR 1526 (HY000): Table has no partition for value 0
```

WITH / WITHOUT VALIDATION

> **Note:**
>
> This feature is available from MariaDB 11.4.

This validation is performed for each row, and can be very slow in the case of larger tables. It is possible to disable this validation by specifying the `WITHOUT VALIDATION` option:

```
ALTER TABLE t1 EXCHANGE PARTITION p1 WITH TABLE t2 WITHOUT VALIDATION;
```

```
Query OK, 0 rows affected (0.048 sec)
```

`WITH VALIDATION` results in the validation being performed, and is the default behavior.

Removing Partitioning

```
REMOVE PARTITIONING
```

ALTER TABLE REMOVE PARTITIONING removes all partitioning from the table, while leaving the data unaffected. To rather drop a particular partition (and discard all of its data), see Dropping Partitions.

```
ALTER TABLE t1 REMOVE PARTITIONING;
```

Reorganizing Partitions

```
REORGANIZE PARTITION [partition_names INTO (partition_definitions)]
```

Reorganizing partitions allows one to adjust existing partitions, without losing data. Specifically, the statement can be used for:

- Splitting an existing partition into multiple partitions.
- Merging a number of existing partitions into a new, single, partition.
- Changing the ranges for a subset of existing partitions defined using `VALUES LESS THAN`.
- Changing the value lists for a subset of partitions defined using `VALUES I`.

- Renaming partitions.

Splitting Partitions

An existing partition can be split into multiple partitions. This can also be used to add a new partition at the low end of a RANGE partition (which is not possible by Adding Partitions).

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014),
  PARTITION p2 VALUES LESS THAN (2015),
  PARTITION p3 VALUES LESS THAN (2016)
);

ALTER TABLE t1 REORGANIZE PARTITION p0 INTO (
    PARTITION p0a VALUES LESS THAN (2012),
    PARTITION p0b VALUES LESS THAN (2013)
);
```

Similarly, if `MAXVALUE` binds the high end:

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014),
  PARTITION p2 VALUES LESS THAN (2015),
  PARTITION p3 VALUES LESS THAN (2016),
  PARTITION p4 VALUES LESS THAN MAXVALUE
);

ALTER TABLE t1 REORGANIZE PARTITION p4 INTO (
    PARTITION p4 VALUES LESS THAN (2017),
    PARTITION p5 VALUES LESS THAN MAXVALUE
);
```

Merging Partitions

A number of existing partitions can be merged into a new partition, for example:

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014),
  PARTITION p2 VALUES LESS THAN (2015),
  PARTITION p3 VALUES LESS THAN (2016)
);

ALTER TABLE t1 REORGANIZE PARTITION p2,p3 INTO (
    PARTITION p2 VALUES LESS THAN (2016)
);
```

Changing Ranges

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014),
  PARTITION p2 VALUES LESS THAN (2015),
  PARTITION p3 VALUES LESS THAN (2016)
);

ALTER TABLE t1 REORGANIZE PARTITION p3 INTO (
  PARTITION p3 VALUES LESS THAN (2017)
);
```

Renaming Partitions

The ALTER TABLE REORGANIZE PARTITION statement can also be used for renaming partitions. Note that this creates a copy of the partition:

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014),
  PARTITION p2 VALUES LESS THAN (2015),
  PARTITION p3 VALUES LESS THAN (2016)
);

ALTER TABLE t1 REORGANIZE PARTITION p3 INTO (
  PARTITION p3_new VALUES LESS THAN (2016)
);
```

Truncating Partitions

```
TRUNCATE PARTITION partition_names
```

`[ALTER TABLE](#pg-ence-sql-statements-data-definition-alter-alter-table-readme-a2aa59e4) ... TRUNCATE PARTITION`

ALTER TABLE TRUNCATE PARTITION removes all data from the specified partition/s, leaving the table and partition structure unchanged. Partitions don’t need to be contiguous:

```
CREATE OR REPLACE TABLE t1 (
  dt DATETIME NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE (YEAR(dt))
  (
  PARTITION p0 VALUES LESS THAN (2013),
  PARTITION p1 VALUES LESS THAN (2014),
  PARTITION p2 VALUES LESS THAN (2015),
  PARTITION p3 VALUES LESS THAN (2016)
);

INSERT INTO t1 VALUES ('2012-11-01'),('2013-11-02'),('2014-11-03'),('2015-11-04');

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2012-11-01 00:00:00 |
| 2013-11-02 00:00:00 |
| 2014-11-03 00:00:00 |
| 2015-11-04 00:00:00 |
+---------------------+

ALTER TABLE t1 TRUNCATE PARTITION p0,p2;

SELECT * FROM t1;
+--------------+
| dt           |
+--------------+
| 2013-11-02 00:00:00 |
| 2015-11-04 00:00:00 |
+---------------------+
```

Analyzing Partitions

Similar to ANALYZE TABLE, key distributions for specific partitions can also be analyzed and stored, for example:

```
ALTER TABLE t1 ANALYZE PARTITION p0,p1,p3;
+---------+---------+----------+----------+
| Table   | Op      | Msg_type | Msg_text |
+---------+---------+----------+----------+
| test.t1 | analyze | status   | OK       |
+---------+---------+----------+----------+
```

Checking Partitions

```
CHECK PARTITION {ALL | PARTITION [,partition2 ...]}
```

Similar to CHECK TABLE, specific partitions can be checked for errors, for example:

```
ALTER TABLE t1 CHECK PARTITION p1,p3;
+---------+-------+----------+----------+
| Table   | Op    | Msg_type | Msg_text |
+---------+-------+----------+----------+
| test.t1 | check | status   | OK       |
+---------+-------+----------+----------+
```

The `ALL` keyword can be used in place of the list of partition names, and the check operation are performed on all partitions.

Repairing Partitions

```
REPAIR PARTITION {ALL | partition [,partition2 ...]} [QUICK] [EXTENDED]
```

Similar to REPAIR TABLE, specific partitions can be repaired:

```
ALTER TABLE t1 REPAIR PARTITION p0,p3;
+---------+--------+----------+----------+
| Table   | Op     | Msg_type | Msg_text |
+---------+--------+----------+----------+
| test.t1 | repair | status   | OK       |
+---------+--------+----------+----------+
```

As with REPAIR TABLE, the `QUICK` and `EXTENDED` options are available. However, the `USE_FRM` option cannot be used with this statement on a partitioned table.

`REPAIR PARTITION` fails if there are duplicate key errors. `ALTER IGNORE TABLE ... REPAIR PARTITION` can be used in this case.

The `ALL` keyword can be used in place of the list of partition names, and the repair operation are performed on all partitions.

Optimizing Partitions

```
OPTIMIZE PARTITION {ALL | PARTITION [,partition2 ...]}
```

Similar to OPTIMIZE TABLE, specific partitions can be checked for errors:

```
ALTER TABLE t1 OPTIMIZE PARTITION p0,p3;
+---------+----------+----------+----------+
| Table   | Op       | Msg_type | Msg_text |
+---------+----------+----------+----------+
| test.t1 | optimize | status   | OK       |
+---------+----------+----------+----------+
```

`OPTIMIZE PARTITION` does not support per-partition optimization on InnoDB tables, and will issue a warning and cause the entire table to rebuilt and analyzed. `ALTER TABLE ... REBUILD PARTITION` and `ALTER TABLE ... ANALYZE PARTITION` can be used instead.

The `ALL` keyword can be used in place of the list of partition names, and the optimize operation are performed on all partitions.

Partitioning for Specific Storage Engines

Some MariaDB storage engines allow more interesting uses for partitioning.

The MERGE storage engine allows one to:

- Treat a set of identical defined MyISAM tables as one.
- A MyISAM table can be in many different `MERGE` sets and also used separately.

SPIDER allows one to:

- Move partitions of the same table on different servers. In this way, the workload can be distributed on more physical or virtual machines (data sharding).
- All partitions of a `SPIDER` table can also live on the same machine. In this case there are a small overhead (`SPIDER` uses connections to localhost), but queries that read multiple partitions will use parallel threads.

CONNECT allows one to:

- Build a table whose partitions are tables using different storage engines (like InnoDB, MyISAM, or even engines that do not support partitioning).
- Build an indexable, writeable table on several data files. These files can be in different formats.

See also: Using CONNECT - Partitioning and Sharding

See Also

- ALTER TABLE
- INFORMATION_SCHEMA.PARTITIONS contains information about existing partitions.
- Partition Maintenance for suggestions on using partitions

This page is licensed: CC BY-SA / Gnu FDL

Partition Pruning and Selection

When a `WHERE` clause is related to the partitioning expression, the optimizer knows which partitions are relevant for the query. Other partitions will not be read. This optimization is called *partition pruning*.

EXPLAIN PARTITIONS can be used to know which partitions are read for a given query. A column called `partitions` will contain a comma-separated list of the accessed partitions. For example:

```
EXPLAIN PARTITIONS SELECT * FROM orders WHERE id < 15000000;
+------+-------------+--------+------------+-------+---------------+---------+---------+------+------+---
----------+
| id   | select_type | table  | partitions | type  | possible_keys | key     | key_len | ref  | rows |
Extra       |
+------+-------------+--------+------------+-------+---------------+---------+---------+------+------+---
----------+
|    1 | SIMPLE      | orders | p0,p1      | range | PRIMARY       | PRIMARY | 4       | NULL |    2 |
Using where |
+------+-------------+--------+------------+-------+---------------+---------+---------+------+------+---
----------+
```

Sometimes the WHERE clause does not contain the necessary information to use partition pruning, or the optimizer cannot infer this information. However, we may know which partitions are relevant for the query. We can force MariaDB to only access the specified partitions by adding a `PARTITION` clause. This feature is called *partition selection*. For example:

```
SELECT * FROM orders PARTITION (p3) WHERE user_id = 50;
SELECT * FROM orders PARTITION (p2,p3) WHERE user_id >= 40;
```

The `PARTITION` clause is supported for all DML statements:

- SELECT
- INSERT
- UPDATE
- DELETE
- REPLACE

Partition Pruning and Triggers

In general, partition pruning is applied to statements contained in triggers.

However, note that if a `BEFORE INSERT` or `BEFORE UPDATE` trigger is defined on a table, MariaDB doesn't know in advance if the columns used in the partitioning expression are changed. For this reason, it is forced to lock all partitions.

This page is licensed: CC BY-SA / Gnu FDL

Partition Maintenance

Maintenance Instructions

You can perform several maintenance tasks on partitioned tables using standard SQL statements or specific `ALTER TABLE` extensions.

Table-Level Maintenance

For general maintenance, MariaDB supports the following statements on partitioned tables just as it does for non-partitioned tables:

- `CHECK TABLE`
- `OPTIMIZE TABLE`
- `ANALYZE TABLE`
- `REPAIR TABLE`

Partition-Specific Operations

To target one or more specific partitions rather than the entire table, use the `ALTER TABLE` extensions listed below. In the SQL syntaxes below, *`partition_names`* is a comma-separated list of partitions, like `p0, p1, p2`.

> **Note:**
>
> For an operation to be performed on all partitions, you can use the `ALL` keyword instead of specifying all the partitions in a comma-separated list. Example:
>
> ```
> ALTER TABLE table_name REBUILD PARTITION ALL
> ```

Rebuilding Partitions

Use this to defragment a partition. This operation drops all records in the partition and re-inserts them:

```
ALTER TABLE table_name REBUILD PARTITION partition_names
```

Optimizing Partitions

If you have deleted many rows or modified variable-length columns (such as `VARCHAR`, `BLOB`, or `TEXT`), use this statement to reclaim unused space and defragment the data file:

```
ALTER TABLE table_name OPTIMIZE PARTITION partition_names
```

> **Note:**
>
> Some storage engines, including `InnoDB`, do not support per-partition optimization. When you run `OPTIMIZE PARTITION` on an `InnoDB` table, MariaDB rebuilds and analyzes the entire table instead.

Analyzing Partitions

Use this to read and store the key distributions for specific partitions:

```
ALTER TABLE table_name ANALYZE PARTITION partition_names
```

Repairing Partitions

Use this to fix corrupted partitions:

```
ALTER TABLE table_name REPAIR PARTITION partition_names
```

Checking Partitions

You can verify the integrity of data and indexes within a partition:

```
ALTER TABLE table_name CHECK PARTITION partition_names
```

Truncating Partitions

To remove all rows from specific partitions while keeping the table structure, use the `TRUNCATE PARTITION` clause:

```
ALTER TABLE table_name TRUNCATE PARTITION partition_names
```

Reorganizing Partitions

Reorganizing partitions isn't just maintenance, but it can help with making future maintenance easier. Use the following statement to change the structure of existing partitions without losing data. This is particularly useful for splitting a partition that contains a `MAXVALUE` range into a new specific range and a new `MAXVALUE` partition. Use the following syntax:

```
ALTER TABLE table_name REORGANIZE PARTITION partition_names INTO (PARTITION partition_definition, ...)
```

Example: If you have a partition `p_future` defined as `VALUES LESS THAN MAXVALUE`, you can split it to add a specific range for the year 2026:

```
ALTER TABLE tbl REORGANIZE PARTITION p_future INTO (PARTITION p_2026 VALUES LESS THAN (2027), PARTITION
p_future VALUES LESS THAN MAXVALUE)
```

Best Practices and Considerations

When managing partitioned tables, follow these guidelines to ensure optimal performance and maintainability.

Managing Time-Series Data

Partitioning is most effective for tables containing time-series data where you periodically remove old records.

- Efficient Deletion: Use `DROP PARTITION` instead of `DELETE` to remove expired data. This is a metadata operation and is significantly faster than row-by-row deletion.
- The Future Partition: When using `RANGE` partitioning, define a "future" partition using `VALUES LESS THAN MAXVALUE`. To add a new specific range, use `REORGANIZE PARTITION` to split the "future" partition into a new range and a new `MAXVALUE` partition. See this section for the syntax.
- The Start Partition: Consider creating a small, empty "start" partition (for example, `VALUES LESS THAN (0)`) to catch `NULL` values or invalid data. Because the partition pruner often scans the first partition by default, keeping it empty improves query efficiency.

Performance and Scale

- Table Size: Partitioning generally provides noticeable benefits only for tables with more than one million rows.
- Partition Limits: Aim to keep the number of partitions below 50. While MariaDB supports up to 8192 partitions, high partition counts can increase the time required for the server to open the table or perform status checks.

- Index Efficiency: Partitioning is not a substitute for proper indexing. Point queries (finding a single row) are often just as fast with a proper index on a non-partitioned table.

This page is licensed: CC BY-SA / Gnu FDL

Partitioning Types

See Partitioning Types Overview.

An introduction to the various partitioning strategies available in MariaDB, helping you choose the right method for your data distribution needs.

See HASH Partitioning Type.

Learn about HASH partitioning, which distributes data based on a user-defined expression to ensure an even spread of rows across partitions.

See KEY Partitioning Type.

Understand KEY partitioning, similar to HASH but using MariaDB's internal hashing function on one or more columns to distribute data.

See LINEAR HASH Partitioning Type.

Explore LINEAR HASH partitioning, a variation of HASH that uses a powers-of-two algorithm for faster partition management at the cost of distribution.

See LINEAR KEY Partitioning Type.

Learn about LINEAR KEY partitioning, which combines the internal key hashing with a linear algorithm for efficient partition handling.

See LIST Partitioning Type.

Understand LIST partitioning, where rows are assigned to partitions based on whether a column value matches one in a defined list of values.

See RANGE COLUMNS and LIST COLUMNS Partitioning Types.

Discover these variants that allow partitioning based on multiple columns and non-integer types, offering greater flexibility than standard RANGE/LIST.

See RANGE Partitioning Type.

The RANGE partitioning type assigns rows to partitions based on whether column values fall within contiguous, non-overlapping ranges.

This page is licensed: CC BY-SA / Gnu FDL

Partitioning Types Overview

A partitioning type determines how a partitioned table's rows are distributed across partitions. Some partition types require the user to specify a partitioning expression that determines in which partition a row are stored.

The size of individual partitions depends on the partitioning type. Read and write performance are affected by the partitioning expression. Therefore, these choices should be made carefully.

Partitioning Types

MariaDB supports the following partitioning types:

- RANGE

- LIST
- RANGE COLUMNS and LIST COLUMNS
- HASH
- LINEAR HASH
- KEY
- LINEAR KEY
- SYSTEM_TIME

See Also

- Partitioning Overview

This page is licensed: CC BY-SA / Gnu FDL

HASH Partitioning Type

Syntax

```
PARTITION BY HASH (partitioning_expression)
[PARTITIONS(number_of_partitions)]
```

Description

`HASH` partitioning is a form of partitioning in which the server takes care of the partition in which to place the data, ensuring an even distribution among the partitions.

It requires a column value, or an expression based on a column value, which is hashed, as well as the number of partitions into which to divide the table.

- *`partitioning_expression`* needs to return a non-constant, deterministic integer. It is evaluated for each insert and update, so overly complex expressions can lead to performance issues. A hashing function operating on a single column, and where the value changes consistently with the column value, allows for easy pruning on ranges of partitions, and is usually a better choice. For this reason, using multiple columns in a hashing expression is not usually recommended.
- *`number_of_partitions`* is a positive integer specifying the number of partitions into which to divide the table. If the `PARTITIONS` clause is omitted, the default number of partitions is one.

Determining the Partition

To determine which partition to use, perform the following calculation:

```
MOD(partitioning_expression, number_of_partitions)
```

For example, if the expression is `TO_DAYS( datetime_column )` and the number of partitions is 5, inserting a datetime value of `'2023-11-15'` would determine the partition as follows:

- `TO_DAYS('2023-11-15')` gives a value of 739204.
- `MOD(739204,5)` returns 4, so the 4th partition is used.

`HASH` partitioning makes use of the modulus of the hashing function's value. The LINEAR HASH partitioning type is similar, using a powers-of-two algorithm. Data is more likely to be evenly distributed over the partitions than with the `LINEAR HASH` partitioning type; however, adding, dropping, merging and splitting partitions is much slower.

Examples

```
CREATE OR REPLACE TABLE t1 (c1 INT, c2 DATETIME)
  PARTITION BY HASH(TO_DAYS(c2))
  PARTITIONS 5;
```

Using the Information Schema PARTITIONS Table for more information:

```
INSERT INTO t1 VALUES (1,'2023-11-15');

SELECT PARTITION_NAME,TABLE_ROWS FROM INFORMATION_SCHEMA.PARTITIONS
  WHERE TABLE_SCHEMA='test' AND TABLE_NAME='t1';
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          0 |
| p1             |          0 |
| p2             |          0 |
| p3             |          0 |
| p4             |          1 |
+----------------+------------+
```

See Also

- Partition Maintenance for suggestions on using partitions

This page is licensed: CC BY-SA / Gnu FDL

KEY Partitioning Type

Syntax

CURRENT

```
PARTITION BY KEY
[ALGORITHM={MYSQL51|MYSQL55|BASE31|CRC32C|XXH32|XXH3}]
([column_names])
[PARTITIONS (number_of_partitions)]
```

- `MYSQL51` and `MYSQL55` are existing algorithms, with `MYSQL55` being the default, also used by default before 12.3
- `CRC32C`, `XXH32`, and `XXH3` use the established hash algorithms of the same names. These are recommended algorithms to use.
- `BASE31` uses a base-31 representation of the bytes and serves as a simple baseline that is more evenly distributed than `MYSQL51` or `MYSQL55` for simple sequential data.

< 12.3

```
PARTITION BY KEY ([column_names])
[PARTITIONS (number_of_partitions)]
```

Description

Partitioning by key is a type of partitioning that is similar to and can be used in a similar way as partitioning by hash.

`KEY` takes an optional list of *`column_names`*, and the hashing function is given by the server.

Compared to `HASH` partitioning, `KEY` partitioning distributes data using a preset hash algorithms on the specified columns, rather than an expression specified by the user.

If no *column_names* are specified, the table's primary key is used if present, or not null unique key if no primary key is present. If neither of these keys are present, not specifying any *column_names* will result in an error:

```
 ERROR 1488 (HY000): Field in list of fields for partition function not found in table
```

Unlike other partitioning types, columns used for partitioning by `KEY` are not limited to integer or `NULL` values.

`KEY` partitions do not support column index prefixes. Any columns in the partitioning key that make use of column prefixes are not used.

Examples

```
CREATE OR REPLACE TABLE t1 (v1 INT)
  PARTITION BY KEY (v1)
  PARTITIONS 2;
```

```
CREATE OR REPLACE TABLE t1 (v1 INT, v2 INT)
  PARTITION BY KEY (v1,v2)
  PARTITIONS 2;
```

```
CREATE OR REPLACE TABLE t1 (
    id INT NOT NULL PRIMARY KEY,
    name VARCHAR(5)
)
PARTITION BY KEY()
PARTITIONS 2;
```

```
CREATE OR REPLACE TABLE t1 (
    id INT NOT NULL UNIQUE KEY,
    name VARCHAR(5)
)
PARTITION BY KEY()
PARTITIONS 2;
```

The unique key must be `NOT NULL`:

```
CREATE OR REPLACE TABLE t1 (
    id INT NULL UNIQUE KEY,
    name VARCHAR(5)
)
PARTITION BY KEY()
PARTITIONS 2;
ERROR 1488 (HY000): Field in list of fields for partition function not found in table
```

`KEY` requires *column_values* if no primary key or not null unique key is present:

```
CREATE OR REPLACE TABLE t1 (
    id INT NULL UNIQUE KEY,
    name VARCHAR(5)
)
PARTITION BY KEY()
PARTITIONS 2;
ERROR 1488 (HY000): Field in list of fields for partition function not found in table
```

```
CREATE OR REPLACE TABLE t1 (
    id INT NULL UNIQUE KEY,
    name VARCHAR(5)
)
PARTITION BY KEY(name)
PARTITIONS 2;
```

Primary key columns with index prefixes are silently ignored, so the following two queries are equivalent:

```
CREATE OR REPLACE TABLE t1 (
    a VARCHAR(10),
    b VARCHAR(10),
    c VARCHAR(10),
    PRIMARY KEY (a(5), b, c(5))
) PARTITION BY KEY() PARTITIONS 2;

CREATE OR REPLACE TABLE t1 (
    a VARCHAR(10),
    b VARCHAR(10),
    c VARCHAR(10),
    PRIMARY KEY (b)
) PARTITION BY KEY() PARTITIONS 2;
```

`a(5)` and `c(5)` are silently ignored in the former.

If all columns use index prefixes, the statement fails with a slightly misleading error:

```
CREATE OR REPLACE TABLE t1 (
    a VARCHAR(10),
    b VARCHAR(10),
    c VARCHAR(10),
    PRIMARY KEY (a(5), b(5), c(5))
) PARTITION BY KEY() PARTITIONS 2;
ERROR 1503 (HY000): A PRIMARY KEY must include all columns in the table's partitioning function
```

This page is licensed: CC BY-SA / Gnu FDL

LINEAR HASH Partitioning Type

Syntax

```
PARTITION BY LINEAR HASH (partitioning_expression)
[PARTITIONS(number_of_partitions)]
```

Description

`LINEAR HASH` partitioning is a form of partitioning, similar to HASH partitioning, in which the server takes care of the partition in which to place the data, ensuring a relatively even distribution among the partitions.

`LINEAR HASH` partitioning makes use of a powers-of-two algorithm, while `HASH` partitioning uses the modulus of the hashing function's value. Adding, dropping, merging and splitting partitions is much faster than with the HASH partitioning type, however, data is less likely to be evenly distributed over the partitions.

Example

```
CREATE OR REPLACE TABLE t1 (c1 INT, c2 DATETIME)
  PARTITION BY LINEAR HASH(TO_DAYS(c2))
  PARTITIONS 5;
```

This page is licensed: CC BY-SA / Gnu FDL

LINEAR KEY Partitioning Type

Syntax

> **CURRENT**

```
PARTITION BY LINEAR KEY [ALGORITHM={MYSQL51|MYSQL55|BASE31|CRC32C|XXH32|XXH3}]
([column_names])
[PARTITIONS (number_of_partitions)]
```

For a description of the different `ALGORITHM` types, see KEY Partitioning.

< 12.3

```
PARTITION BY LINEAR KEY ([column_names])
[PARTITIONS (number_of_partitions)]
```

Description

`LINEAR KEY` partitioning is a form of partitioning, similar to KEY partitioning.

`LINEAR KEY` partitioning makes use of a powers-of-two algorithm, while `KEY` partitioning uses modulo arithmetic to determine the partition number.

Adding, dropping, merging and splitting partitions is much faster than with the KEY partitioning type; however, data is less likely to be evenly distributed over the partitions.

Example

```
CREATE OR REPLACE TABLE t1 (v1 INT)
  PARTITION BY LINEAR KEY (v1)
  PARTITIONS 2;
```

This page is licensed: CC BY-SA / Gnu FDL

LIST Partitioning Type

`LIST` partitioning is conceptually similar to RANGE partitioning. In both cases you decide a partitioning expression (a column, or a slightly more complex calculation) and use it to determine which partitions will contain each row. However, with the `RANGE` type, partitioning is done by assigning a range of values to each partition. With the `LIST` type, we assign a set of values to each partition. This is usually preferred if the partitioning expression can return a limited set of values.

A variant of this partitioning method, LIST COLUMNS, allows us to use multiple columns and more datatypes.

Syntax

The last part of a CREATE TABLE statement can be the definition of the new table's partitions. In the case of `LIST` partitioning, the syntax is as follows:

```
PARTITION BY LIST (partitioning_expression)
(
    PARTITION partition_name VALUES IN (value_list),
    [ PARTITION partition_name VALUES IN (value_list), ... ]
        [ PARTITION partition_name DEFAULT ]
)
```

`PARTITION BY LIST` indicates that the partitioning type is `LIST`.

The *`partitioning_expression`* is an SQL expression that returns a value from each row. In the simplest cases, it is a column name. This value is used to determine which partition should contain a row.

`partition_name` is the name of a partition.

value_list is a list of values. If *partitioning_expression* returns one of these values, the row are stored in this partition. If we try to insert something that does not belong to any of these value lists, the row are rejected with an error.

The DEFAULT partition catches all records which do not fit into other partitions.

Use Cases

LIST partitioning can be useful when we have a column that can only contain a limited set of values. Even in that case, RANGE partitioning could be used instead; but LIST partitioning allows us to equally distribute the rows by assigning a proper set of values to each partition.

Example

```
CREATE OR REPLACE TABLE t1 (
  num TINYINT(1) NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY LIST (num) (
    PARTITION p0 VALUES IN (0,1),
    PARTITION p1 VALUES IN (2,3),
    PARTITION p2 DEFAULT
  );
```

This page is licensed: CC BY-SA / Gnu FDL

RANGE COLUMNS and LIST COLUMNS Partitioning Types

RANGE COLUMNS and LIST COLUMNS are variants of, respectively, RANGE and LIST. With these partitioning types, there is not a single partitioning expression; instead, a list of one or more columns is accepted. The following rules apply:

- The list can contain one or more columns.
- Columns can be of any integer, string, DATE, and DATETIME types.
- Only bare columns are permitted; no expressions.

All the specified columns are compared to the specified values to determine which partition should contain a specific row. See below for details.

Syntax

The last part of a CREATE TABLE statement can be definition of the new table's partitions. In the case of RANGE COLUMNS partitioning, the syntax is as follows:

```
PARTITION BY RANGE COLUMNS (col1, col2, ...)
(
    PARTITION partition_name VALUES LESS THAN (value1, value2, ...),
    [ PARTITION partition_name VALUES LESS THAN (value1, value2, ...), ... ]
)
```

The syntax for LIST COLUMNS is as follows:

```
PARTITION BY LIST COLUMNS (partitioning_expression)
(
    PARTITION partition_name VALUES IN (value1, value2, ...),
    [ PARTITION partition_name VALUES IN (value1, value2, ...), ... ]
        [ PARTITION partititon_name DEFAULT ]
)
```

partition_name is the name of a partition.

Comparisons

To determine which partition should contain a row, all specified columns are compared to each partition definition.

With `LIST COLUMNS`, a row matches a partition if all row values are identical to the specified values. At most one partition can match the row.

With `RANGE COLUMNS`, a row matches a partition if it is less than the specified value tuple in lexicographic order. The first partition that matches the row values are used.

The `DEFAULT` partition catches all records which do not fit in other partitions. Only one `DEFAULT` partition is allowed.

Examples

`RANGE COLUMNS` partition:

```
CREATE OR REPLACE TABLE t1 (
  date1 DATE NOT NULL,
  date2 DATE NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY RANGE COLUMNS (date1,date2) (
    PARTITION p0 VALUES LESS THAN ('2013-01-01', '1994-12-01'),
    PARTITION p1 VALUES LESS THAN ('2014-01-01', '1995-12-01'),
    PARTITION p2 VALUES LESS THAN ('2015-01-01', '1996-12-01')
);
```

`LIST COLUMNS` partition:

```
CREATE OR REPLACE TABLE t1 (
  num TINYINT(1) NOT NULL
)
  ENGINE = InnoDB
  PARTITION BY LIST COLUMNS (num) (
    PARTITION p0 VALUES IN (0,1),
    PARTITION p1 VALUES IN (2,3),
    PARTITION p2 DEFAULT
  );
```

This page is licensed: CC BY-SA / Gnu FDL

RANGE Partitioning Type

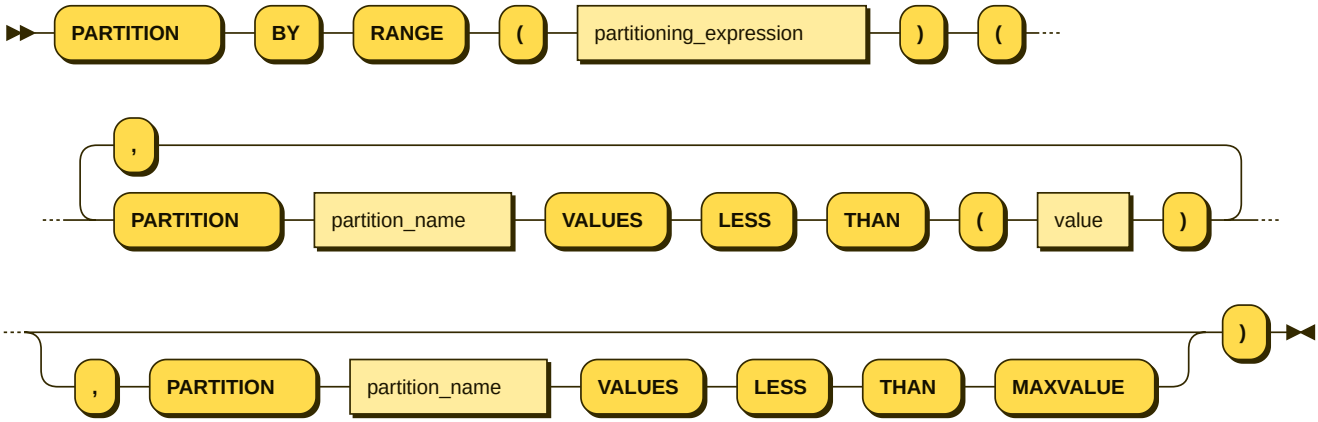
The `RANGE` partitioning type is used to assign each partition a range of values generated by the partitioning expression. Ranges must be ordered, contiguous and non-overlapping. The minimum value is always included in the first range. The highest value may or may not be included in the last range.

A variant of this partitioning method, RANGE COLUMNS, allows us to use multiple columns and more datatypes.

Syntax

The last part of a CREATE TABLE statement can be definition of the new table's partitions. In the case of `RANGE` partitioning, the syntax is the following:

```
PARTITION BY RANGE (partitioning_expression)
(
    PARTITION partition_name VALUES LESS THAN (value),
    [ PARTITION partition_name VALUES LESS THAN (value), ... ]
    [ PARTITION partition_name VALUES LESS THAN MAXVALUE ]
)
```



Railroad diagram of RANGE partitioning — equivalent to the BNF above

`PARTITION BY RANGE` indicates that the partitioning type is `RANGE`.

- *`partitioning_expression`* is an SQL expression that returns a value from each row. In the simplest cases, it is a column name. This value is used to determine which partition should contain a row.
- *`partition_name`* is the name of a partition.
- *`value`* indicates the upper bound for that partition. The values must be ascending. For the first partition, the lower limit is `NULL`. When trying to insert a row, if its value is higher than the upper limit of the last partition, the row are rejected (with an error, if the IGNORE keyword is not used).

As a catchall, `MAXVALUE` can be specified as a value for the last partition. Note, however, that in order to append a new partition, it is not possible to use ADD PARTITION; instead, REORGANIZE PARTITION must be used.

Use Cases

A typical use case is when we want to partition a table whose rows refer to a moment or period in time; for example commercial transactions, blog posts, or events of some kind. We can partition the table by year, to keep all recent data in one partition and distribute historical data in big partitions that are stored on slower disks. Or, if our queries always read rows which refer to the same month or week, we can partition the table by month or year week (in this case, historical data and recent data are stored together).

AUTO_INCREMENT values also represent a chronological order. So, these values can be used to store old data in separate partitions. However, partitioning by id is not the best choice if we usually query a table by date.

Examples

Partitioning a log table by year:

```
CREATE TABLE log
(
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    dt DATETIME NOT NULL,
    user INT UNSIGNED,
    PRIMARY KEY (id, dt)
)
    ENGINE = InnoDB
PARTITION BY RANGE (YEAR(dt))
(
    PARTITION p0 VALUES LESS THAN (2013),
    PARTITION p1 VALUES LESS THAN (2014),
    PARTITION p2 VALUES LESS THAN (2015),
    PARTITION p3 VALUES LESS THAN (2016)
);
```

Partitioning the table by both year and month:

```
CREATE TABLE log2
(
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    ts TIMESTAMP NOT NULL,
    user INT UNSIGNED,
    PRIMARY KEY (id, ts)
)
    ENGINE = InnoDB
PARTITION BY RANGE (UNIX_TIMESTAMP(ts))
(
    PARTITION p0 VALUES LESS THAN (UNIX_TIMESTAMP('2014-08-01 00:00:00')),
    PARTITION p1 VALUES LESS THAN (UNIX_TIMESTAMP('2014-11-01 00:00:00')),
    PARTITION p2 VALUES LESS THAN (UNIX_TIMESTAMP('2015-01-01 00:00:00')),
    PARTITION p3 VALUES LESS THAN (UNIX_TIMESTAMP('2015-02-01 00:00:00'))
);
```

In the last example, the UNIX_TIMESTAMP function is used to accomplish the purpose. Also, the first two partitions cover longer periods of time (probably because the logged activities were less intensive).

In both cases, when our tables become huge and we don't need to store all historical data any more, we can drop the oldest partitions in this way:

```
ALTER TABLE log DROP PARTITION p0;
```

We will still be able to drop a partition that does not contain the oldest data, but all rows stored in it will disappear.

Example of an error when inserting outside a defined partition range:

```
INSERT INTO log(id,dt) VALUES
  (1, '2016-01-01 01:01:01'),
  (2, '2015-01-01 01:01:01');
ERROR 1526 (HY000): Table has no partition for value 2016
```

To avoid the error, use the `IGNORE` keyword:

```
INSERT IGNORE INTO log(id,dt) VALUES
  (1, '2016-01-01 01:01:01'),
  (2, '2015-01-01 01:01:01');

SELECT * FROM log;
+----+---------------------+------+
| id | timestamp           | user |
+----+---------------------+------+
|  2 | 2015-01-01 01:01:01 | NULL |
+----+---------------------+------+
```

An alternative definition with `MAXVALUE` as a catchall:

```
CREATE TABLE log
(
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    dt DATETIME NOT NULL,
    user INT UNSIGNED,
    PRIMARY KEY (id, dt)
)
    ENGINE = InnoDB
PARTITION BY RANGE (YEAR(dt))
(
    PARTITION p0 VALUES LESS THAN (2013),
    PARTITION p1 VALUES LESS THAN (2014),
    PARTITION p2 VALUES LESS THAN (2015),
    PARTITION p3 VALUES LESS THAN (2016),
    PARTITION p4 VALUES LESS THAN MAXVALUE
);
```

This page is licensed: CC BY-SA / Gnu FDL

Partitioning Limitations

The following limitations apply to partitioning in MariaDB:

- Each table can contain a maximum of 8192 partitions.
- Queries are never parallelized, even when they involve multiple partitions.
- A table can only be partitioned if the storage engine supports partitioning.
- All partitions must use the same storage engine. For a workaround, see Using CONNECT - Partitioning and Sharding.
- A partitioned table cannot contain, or be referenced by, foreign keys.
- The query cache is not aware of partitioning and partition pruning. Modifying a partition will invalidate the entries related to the whole table.
- Updates can run more slowly when binlog_format=ROW and a partitioned table is updated than an equivalent update of a non-partitioned table.
- All columns used in the partitioning expression for a partitioned table must be part of every unique key that the table may have.
- In versions prior to MariaDB 11.3.2, it is not possible to create partitions on tables that contain GEOMETRY types.

See Also

- INFORMATION_SCHEMA.PARTITIONS contains information about existing partitions.
- Partition Maintenance for suggestions on using partitions

This page is licensed: CC BY-SA / Gnu FDL

Partitions Files

A partitioned table is stored in multiple files. By default, these files are stored in the MariaDB (or InnoDB) data directory. It is possible to keep them in different paths by specifying DATA_DIRECTORY and INDEX_DIRECTORY table options. This is useful to store different partitions on different devices.

Note that, if the innodb_file_per_table server system variable is set to 0 at the time of the table creation, all partitions are stored in the system tablespace.

The following files exist for each partitioned tables:

File name	Notes
table_name.frm	Contains the table definition. Non-partitioned tables have this file, too.
table_name.par	Contains the partitions definitions.
table_name#P#partition_name .ext	Normal files created by the storage engine use this pattern for names. The extension depends on the storage engine.

For example, an InnoDB table with 4 partitions will have the following files:

```
orders.frm
orders.par
orders#P#p0.ibd
orders#P#p1.ibd
orders#P#p2.ibd
orders#P#p3.ibd
```

If we convert the table to MyISAM, we will have these files:

```
orders.frm
orders.par
orders#P#p0.MYD
orders#P#p0.MYI
orders#P#p1.MYD
orders#P#p1.MYI
orders#P#p2.MYD
orders#P#p2.MYI
orders#P#p3.MYD
orders#P#p3.MYI
```

This page is licensed: CC BY-SA / Gnu FDL

Partitions Metadata

The PARTITIONS table in the INFORMATION_SCHEMA database contains information about partitions.

The SHOW TABLE STATUS statement contains a `Create_options` column, that contains the string 'partitioned' for partitioned tables.

The SHOW CREATE TABLE statement returns the CREATE TABLE statement that can be used to re-create a table, including the partitions definition.

This page is licensed: CC BY-SA / Gnu FDL

Stored Routines

See Stored Procedures.

Master stored procedures in MariaDB Server. This section covers creating, executing, and managing these powerful routines to encapsulate complex logic and improve application performance.

See Stored Functions.

Utilize stored functions in MariaDB Server. This section details creating, using, and managing user-defined functions to extend SQL capabilities and streamline data manipulation.

See Binary Logging of Stored Routines.

When binary logging is enabled, stored routines may require special handling (like SUPER privileges) if they are non-deterministic, to ensure consistent replication.

See Stored Routine Limitations.

Stored routines have specific restrictions, such as prohibiting certain SQL statements (e.g., LOAD DATA) and disallowing result sets in functions.

See DBMS_OUTPUT.

The DBMS_OUTPUT plugin provides Oracle-compatible output buffering functions (like PUT_LINE), allowing stored procedures to send messages to the client.

This page is licensed: CC BY-SA / Gnu FDL

Stored Procedures

See Stored Procedure Overview.

Stored procedures are precompiled collections of SQL statements stored on the server, allowing for encapsulated logic, parameterized execution, and improved application performance.

See CREATE PROCEDURE.

Complete CREATE PROCEDURE guide for MariaDB. Complete reference documentation for implementation, configuration, and usage with comprehensive examples and.

See ALTER PROCEDURE.

The ALTER PROCEDURE statement modifies the characteristics of an existing stored procedure, such as its security context or comment, without changing its logic.

See DROP PROCEDURE.

The DROP PROCEDURE statement permanently removes a stored procedure and its associated privileges from the database.

This page is licensed: CC BY-SA / Gnu FDL

Stored Procedure Overview

A Stored Procedure is a routine invoked with a CALL statement. It may have input parameters, output parameters and parameters that are both input parameters and output parameters.

Creating a Stored Procedure

Here's a skeleton example to see a stored procedure in action:

```
DELIMITER //

CREATE PROCEDURE Reset_animal_count()
 MODIFIES SQL DATA
 UPDATE animal_count SET animals = 0;
//

DELIMITER ;
```

First, the delimiter is changed, since the procedure definition will contain the regular semicolon delimiter. The procedure is named `Reset_animal_count`. `MODIFIES SQL DATA` indicates that the procedure will perform a write action of sorts, and modify data. It's for advisory purposes only. Finally, there's the actual SQL statement - an `UPDATE`.

```
SELECT * FROM animal_count;
+---------+
| animals |
+---------+
|     101 |
+---------+

CALL Reset_animal_count();

SELECT * FROM animal_count;
+---------+
| animals |
+---------+
|       0 |
+---------+
```

A more complex example, with input parameters, from an actual procedure used by banks:

```
CREATE PROCEDURE
  Withdraw                             /* Routine name */
  (parameter_amount DECIMAL(6,2),     /* Parameter list */
  parameter_teller_id INTEGER,
  parameter_customer_id INTEGER)
  MODIFIES SQL DATA                   /* Data access clause */
  BEGIN                         /* Routine body */
    UPDATE Customers
        SET balance = balance - parameter_amount
        WHERE customer_id = parameter_customer_id;
    UPDATE Tellers
        SET cash_on_hand = cash_on_hand + parameter_amount
        WHERE teller_id = parameter_teller_id;
    INSERT INTO Transactions VALUES (
        parameter_customer_id,
        parameter_teller_id,
        parameter_amount);
  END;
```

See CREATE PROCEDURE for full syntax details.

Why Use Stored Procedures?

Security is a key reason. Banks commonly use stored procedures so that applications and users don't have direct access to the tables. Stored procedures are also useful in an environment where multiple languages and clients are all used to perform the same operations.

Stored Procedure Listings and Definitions

To find which stored procedures are defined on the server, use SHOW PROCEDURE STATUS.

```
SHOW PROCEDURE STATUS\G
*************************** 1. row ***************************
                  Db: test
                Name: Reset_animal_count
                Type: PROCEDURE
             Definer: root@localhost
            Modified: 2013-06-03 08:55:03
             Created: 2013-06-03 08:55:03
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

or query the routines table in the INFORMATION_SCHEMA database directly:

```
SELECT ROUTINE_NAME FROM INFORMATION_SCHEMA.ROUTINES
  WHERE ROUTINE_TYPE='PROCEDURE';
+--------------------+
| ROUTINE_NAME       |
+--------------------+
| Reset_animal_count |
+--------------------+
```

To find out what the stored procedure does, use SHOW CREATE PROCEDURE.

```
SHOW CREATE PROCEDURE Reset_animal_count\G
*************************** 1. row ***************************
           Procedure: Reset_animal_count
            sql_mode:
    Create Procedure: CREATE DEFINER=`root`@`localhost` PROCEDURE `Reset_animal_count`()
    MODIFIES SQL DATA
UPDATE animal_count SET animals = 0
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

Dropping and Updating a Stored Procedure

To drop a stored procedure, use the DROP PROCEDURE statement.

```
DROP PROCEDURE Reset_animal_count;
```

To change the characteristics of a stored procedure, use ALTER PROCEDURE. However, you cannot change the parameters or body of a stored procedure using this statement; to make such changes, you must drop and re-create the procedure using CREATE OR REPLACE PROCEDURE (which retains existing privileges), or DROP PROCEDURE followed by CREATE PROCEDURE.

Permissions in Stored Procedures

See the article Stored Routine Privileges.

This page is licensed: CC BY-SA / Gnu FDL

CREATE PROCEDURE

Syntax

CURRENT

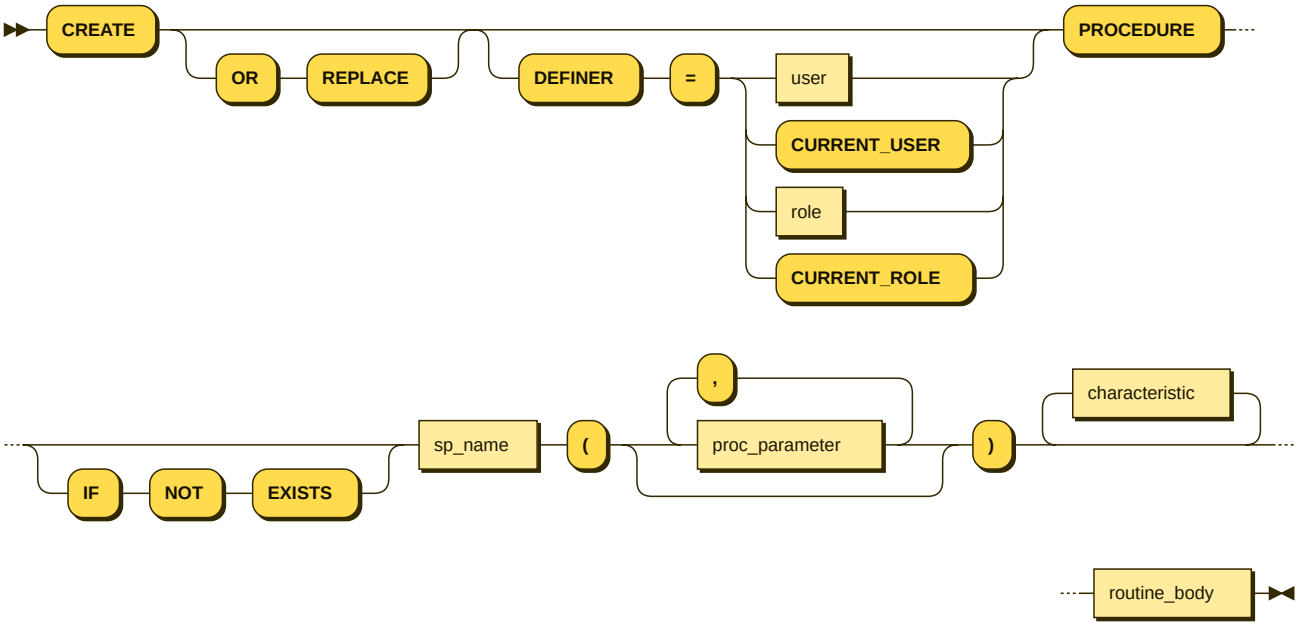
```
CREATE
    [OR REPLACE]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    PROCEDURE [IF NOT EXISTS] sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

proc_parameter:
    [ OUT | INOUT | IN OUT] param_name type |
    [ IN ] param_name type [DEFAULT value or expression]

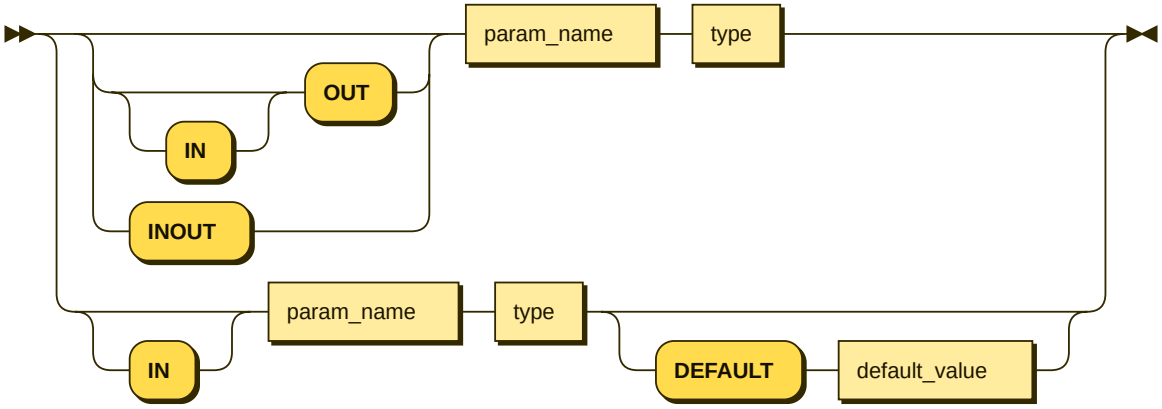
type:
    Any valid MariaDB data type

characteristic:
    LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'

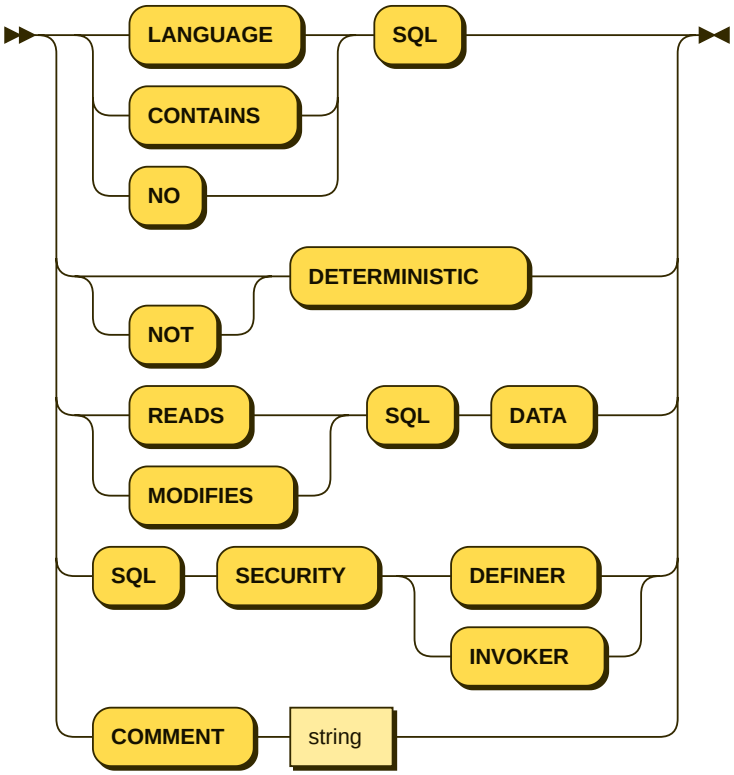
routine_body:
    Valid SQL procedure statement
```



Railroad diagram of CREATE PROCEDURE — equivalent to the BNF above



Railroad diagram of proc_parameter



Railroad diagram of characteristic

The `IN OUT` parameter works only in Oracle mode.

< 11.8

```
CREATE
    [OR REPLACE]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    PROCEDURE [IF NOT EXISTS] sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

proc_parameter:
    [ IN | OUT | INOUT ] param_name type

type:
    Any valid MariaDB data type

characteristic:
    LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'

routine_body:
    Valid SQL procedure statement
```

Description

Creates a stored procedure. By default, a routine is associated with the default database. To associate the routine explicitly with a given database, specify the name as `db_name.sp_name` when you create it.

When the routine is invoked, an implicit `USE``db_name` is performed (and undone when the routine terminates). The causes the routine to have the given default database while it executes. `USE` statements within stored routines are disallowed.

When a stored procedure has been created, you invoke it by using the `CALL` statement (see CALL).

To execute the `CREATE PROCEDURE` statement, it is necessary to have the `CREATE ROUTINE` privilege. By default, MariaDB automatically grants the `ALTER ROUTINE` and `EXECUTE` privileges to the routine creator. See also Stored Routine Privileges.

The `DEFINER` and `SQL SECURITY` clauses specify the security context to be used when checking access privileges at routine execution time, as described here. Requires the SET USER privilege.

If the routine name is the same as the name of a built-in SQL function, you must use a space between the name and the following parenthesis when defining the routine, or a syntax error occurs. This is also true when you invoke the routine later. For this reason, we suggest that it is better to avoid reusing the names of existing SQL functions for your own stored routines.

The `IGNORE_SPACE` SQL mode applies to built-in functions, not to stored routines. It is always allowable to have spaces after a routine name, regardless of whether `IGNORE_SPACE` is enabled.

The parameter list enclosed within parentheses must always be present. If there are no parameters, an empty parameter list of `()` should be used. Parameter names are not case sensitive.

For valid identifiers to use as procedure names, see Identifier Names.

Things to be Aware of With CREATE OR REPLACE

- One can't use `OR REPLACE` together with `IF EXISTS`.

CREATE PROCEDURE IF NOT EXISTS

If the `IF NOT EXISTS` clause is used, then the procedure will only be created if a procedure with the same name does not already exist. If the procedure already exists, then a warning are triggered by default.

IN/OUT/INOUT/IN OUT

Each parameter is an `IN` parameter by default. To specify otherwise for a parameter, use the keyword `OUT` or `INOUT` before the parameter name.

An `IN` parameter passes a value into a procedure. The procedure might modify the value, but the modification is not visible to the caller when the procedure returns. An `OUT` parameter passes a value from the procedure back to the caller. Its initial value is `NULL` within the procedure, and its value is visible to the caller when the procedure returns. An `INOUT` parameter is initialized by the caller, can be modified by the procedure, and any change made by the procedure is visible to the caller when the procedure returns.

For each `OUT` or `INOUT` parameter, pass a user-defined variable in the `CALL` statement that invokes the procedure so that you can obtain its value when the procedure returns. If you are calling the procedure from within another stored procedure or function, you can also pass a routine parameter or local routine variable as an `IN` or `INOUT` parameter.

DEFAULT value or expression

As of MariaDB 11.8, each parameter can be defined as having a default value or expression. This can be useful if needing to add extra parameters to a procedure which is already in use.

DETERMINISTIC/NOT DETERMINISTIC

`DETERMINISTIC` and `NOT DETERMINISTIC` apply only to functions. Specifying `DETERMINISTC` or `NON-DETERMINISTIC` in procedures has no effect. The default value is `NOT DETERMINISTIC`. Functions are `DETERMINISTIC` when they always return the same value for the same input. For example, a truncate or substring function. Any function involving data, therefore, is always `NOT DETERMINISTIC`.

CONTAINS SQL/NO SQL/READS SQL DATA/MODIFIES SQL DATA

`CONTAINS SQL`, `NO SQL`, `READS SQL DATA`, and `MODIFIES SQL DATA` are informative clauses that tell the server what the function does. MariaDB does not check in any way whether the specified clause is correct. If none of these clauses are specified, `CONTAINS SQL` is used by default.

`MODIFIES SQL DATA` means that the function contains statements that may modify data stored in databases. This happens if the function contains statements like DELETE, UPDATE, INSERT, REPLACE or DDL.

`READS SQL DATA` means that the function reads data stored in databases but does not modify any data. This happens if SELECT statements are used, but there no write operations are executed.

`CONTAINS SQL` means that the function contains at least one SQL statement, but it does not read or write any data stored in a database. Examples include SET or DO.

`NO SQL` means nothing, because MariaDB does not currently support any language other than SQL.

The routine_body consists of a valid SQL procedure statement. This can be a simple statement such as SELECT or INSERT, or it can be a compound statement written using BEGIN and END. Compound statements can contain declarations, loops, and other control structure statements. See Programmatic and Compound Statements for syntax details.

MariaDB allows routines to contain DDL statements, such as `CREATE` and `DROP`. MariaDB also allows stored procedures (but not stored functions) to contain SQL transaction statements such as `COMMIT`.

For additional information about statements that are not allowed in stored routines, see Stored Routine Limitations.

Invoking stored procedure from within programs

For information about invoking stored procedures from within programs written in a language that has a MariaDB/MySQL interface, see CALL.

OR REPLACE

If the optional `OR REPLACE` clause is used, it acts as a shortcut for the following statements, with the exception that any existing privileges for the procedure are not dropped:

```
DROP PROCEDURE IF EXISTS name;
CREATE PROCEDURE name ...;
```

sql_mode

MariaDB stores the sql_mode system variable setting that is in effect at the time a routine is created and always executes the routine with this setting in force, regardless of the server SQL mode in effect when the routine is invoked.

Character Sets and Collations

Procedure parameters can be declared with any character set/collation. If the character set and collation are not specifically set, the database defaults at the time of creation are used. If the database defaults change at a later stage, the stored procedure character set/collation will not be changed at the same time; the stored procedure needs to be dropped and recreated to ensure the same character set/collation as the database is used.

Oracle Mode

A subset of Oracle's PL/SQL language is supported in addition to the traditional SQL/PSM-based MariaDB syntax. See Oracle mode for details on changes when running Oracle mode.

Examples

The following example shows a simple stored procedure that uses an `OUT` parameter. It uses the `DELIMITER` command to set a new delimiter for the duration of the process — see Delimiters in the mariadb client.

```
DELIMITER //

CREATE PROCEDURE simpleproc (OUT param1 INT)
 BEGIN
  SELECT COUNT(*) INTO param1 FROM t;
 END;
//

DELIMITER ;

CALL simpleproc(@a);

SELECT @a;
+------+
| @a   |
+------+
|    1 |
+------+
```

Character set and collation:

```
DELIMITER //

CREATE PROCEDURE simpleproc2 (
  OUT param1 CHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8_bin'
)
 BEGIN
  SELECT CONCAT('a'),f1 INTO param1 FROM t;
 END;
//

DELIMITER ;
```

`CREATE OR REPLACE`:

Re-creating an existing procedure without `OR REPLACE` fails, while `CREATE OR REPLACE` replaces the existing definition:

```
DELIMITER //

CREATE PROCEDURE simpleproc2 (
  OUT param1 CHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8_bin'
)
 BEGIN
  SELECT CONCAT('a'),f1 INTO param1 FROM t;
 END;
//
ERROR 1304 (42000): PROCEDURE simpleproc2 already exists

CREATE OR REPLACE PROCEDURE simpleproc2 (
  OUT param1 CHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8_bin'
)
 BEGIN
  SELECT CONCAT('a'),f1 INTO param1 FROM t;
 END;
//
Query OK, 0 rows affected (0.03 sec)

DELIMITER ;
```

See Also

- Identifier Names
- Stored Procedure Overview
- ALTER PROCEDURE
- DROP PROCEDURE
- SHOW CREATE PROCEDURE
- SHOW PROCEDURE STATUS
- Stored Routine Privileges
- Information Schema ROUTINES Table

This page is licensed: CC BY-SA / Gnu FDL

ALTER PROCEDURE

Syntax

```
ALTER PROCEDURE proc_name [characteristic ...]

characteristic:
    { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'
```

Description

This statement can be used to change the characteristics of a stored procedure. More than one change may be specified in an `ALTER PROCEDURE` statement. However, you cannot change the parameters or body of a stored procedure using this statement. To make such changes, you must drop and re-create the procedure using CREATE OR REPLACE PROCEDURE.

You must have the `ALTER ROUTINE` privilege for the procedure. By default, that privilege is granted automatically to the procedure creator. See Stored Routine Privileges.

Example

```
ALTER PROCEDURE simpleproc SQL SECURITY INVOKER;
```

See Also

- Stored Procedure Overview
- CREATE PROCEDURE
- SHOW CREATE PROCEDURE
- DROP PROCEDURE
- SHOW CREATE PROCEDURE
- SHOW PROCEDURE STATUS
- Stored Routine Privileges
- Information Schema ROUTINES Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

DROP PROCEDURE

Syntax

```
DROP PROCEDURE [IF EXISTS] sp_name
```

Description

This statement is used to drop a stored procedure. That is, the specified routine is removed from the server along with all privileges specific to the procedure. You must have the `ALTER ROUTINE` privilege for the routine. If the automatic_sp_privileges server system variable is set, that privilege and `EXECUTE` are granted automatically to the routine creator - see Stored Routine Privileges.

The `IF EXISTS` clause is a MySQL/MariaDB extension. It prevents an error from occurring if the procedure or function does not exist. A `NOTE` is produced that can be viewed with SHOW WARNINGS.

While this statement takes effect immediately, threads which are executing a procedure can continue execution.

Examples

```
DROP PROCEDURE simpleproc;
```

IF EXISTS:

```
DROP PROCEDURE simpleproc;
ERROR 1305 (42000): PROCEDURE test.simpleproc does not exist

DROP PROCEDURE IF EXISTS simpleproc;
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------+
| Level | Code | Message                                 |
+-------+------+-----------------------------------------+
| Note  | 1305 | PROCEDURE test.simpleproc does not exist |
+-------+------+-----------------------------------------+
```

See Also

- DROP FUNCTION
- Stored Procedure Overview
- CREATE PROCEDURE
- ALTER PROCEDURE
- SHOW CREATE PROCEDURE
- SHOW PROCEDURE STATUS
- Information Schema ROUTINES Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

Stored Functions

See Stored Function Overview.

A Stored Function is a set of SQL statements that can be called by name, accepts parameters, and returns a single value, enhancing SQL with custom logic.

See Stored Aggregate Functions.

Stored Aggregate Functions allow users to create custom aggregate functions that process a sequence of rows and return a single summary result.

See Stored Routine Privileges.

This page explains the privileges required to create, alter, execute, and drop stored routines, including the automatic grants for creators.

See DROP FUNCTION.

The DROP FUNCTION statement removes a stored function from the database, deleting its definition and associated privileges.

See Stored Function Limitations.

This page details the restrictions on stored functions, such as the inability to return result sets or use transaction control statements.

This page is licensed: CC BY-SA / Gnu FDL

Stored Function Overview

A Stored Function is a defined function that is called from within an SQL statement like a regular function and returns a single value.

Creating Stored Functions

Here's a skeleton example to see a stored function in action:

```
DELIMITER //

CREATE FUNCTION FortyTwo() RETURNS TINYINT DETERMINISTIC
BEGIN
 DECLARE x TINYINT;
 SET x = 42;
 RETURN x;
END

//

DELIMITER ;
```

First, the delimiter is changed, since the function definition will contain the regular semicolon delimiter. See Delimiters in the mariadb client for more. Then the function is named `FortyTwo` and defined to return a `tinyin`. The `DETERMINISTIC` keyword is not necessary in all cases (although if binary logging is on, leaving it out will throw an error), and is to help the query optimizer choose a query plan. A deterministic function is one that, given the same arguments, will always return the same result.

Next, the function body is placed between BEGIN and END statements. It declares a tinyint, `X`, which is simply set to 42, and this is the result returned.

```
SELECT FortyTwo();
+------------+
| FortyTwo() |
+------------+
|         42 |
+------------+
```

Of course, a function that doesn't take any arguments is of little use. Here's a more complex example:

```
DELIMITER //
CREATE FUNCTION VatCents(price DECIMAL(10,2)) RETURNS INT DETERMINISTIC
BEGIN
 DECLARE x INT;
 SET x = price * 114;
 RETURN x;
END //
Query OK, 0 rows affected (0.04 sec)
DELIMITER ;
```

This function takes an argument, `price` which is defined as a DECIMAL, and returns an INT.

Take a look at the CREATE FUNCTION page for more details.

It is also possible to create stored aggregate functions.

Stored Function Listings and Definitions

To find which stored functions are running on the server, use SHOW FUNCTION STATUS:

```
SHOW FUNCTION STATUS\G
*************************** 1. row ***************************
                  Db: test
                Name: VatCents
                Type: FUNCTION
             Definer: root@localhost
            Modified: 2013-06-01 12:40:31
             Created: 2013-06-01 12:40:31
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Alternatively, query the routines table in the `INFORMATION_SCHEMA` database directly:

```
SELECT ROUTINE_NAME FROM INFORMATION_SCHEMA.ROUTINES WHERE
  ROUTINE_TYPE='FUNCTION';
+--------------+
| ROUTINE_NAME |
+--------------+
| VatCents     |
+--------------+
```

To find out what the stored function does, use SHOW CREATE FUNCTION:

```
SHOW CREATE FUNCTION VatCents\G
*************************** 1. row ***************************
            Function: VatCents
            sql_mode:
     Create Function: CREATE DEFINER=`root`@`localhost` FUNCTION `VatCents`(price DECIMAL(10,2)) RETURNS
int(11)
    DETERMINISTIC
BEGIN
 DECLARE x INT;
 SET x = price * 114;
 RETURN x;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

Dropping and Updating Stored Functions

To drop a stored function, use the DROP FUNCTION statement.

```
DROP FUNCTION FortyTwo;
```

To change the characteristics of a stored function, use ALTER FUNCTION. Note that you cannot change the parameters or body of a stored function using this statement; to make such changes, you must drop and re-create the function using `DROP FUNCTION` and `CREATE FUNCTION`.

Permissions in Stored Functions

See the article Stored Routine Privileges.

See Also

- CREATE FUNCTION
- SHOW CREATE FUNCTION
- DROP FUNCTION
- Stored Routine Privileges
- SHOW FUNCTION STATUS
- Information Schema ROUTINES Table
- Stored Aggregate Functions.

This page is licensed: CC BY-SA / Gnu FDL

Stored Aggregate Functions

Aggregate functions are functions that are computed over a sequence of rows and return one result for the sequence of rows.

Creating a custom aggregate function is done using the CREATE FUNCTION statement with two main differences:

- The addition of the AGGREGATE keyword, so `CREATE AGGREGATE FUNCTION`
- The `FETCH GROUP NEXT ROW` instruction inside the loop
- Oracle PL/SQL compatibility using SQL/PL is provided

Standard Syntax

```
CREATE AGGREGATE FUNCTION function_name (parameters) RETURNS return_type
BEGIN
      ALL types of declarations
      DECLARE CONTINUE HANDLER FOR NOT FOUND RETURN return_val;
      LOOP
           FETCH GROUP NEXT ROW; // fetches next row FROM TABLE
           other instructions
      END LOOP;
END
```

Stored aggregate functions were a 2016 Google Summer of Code project by Varun Gupta.

Using SQL/PL

```
SET sql_mode=Oracle;
DELIMITER //

CREATE AGGREGATE FUNCTION function_name (parameters) RETURN return_type
   declarations
BEGIN
   LOOP
      FETCH GROUP NEXT ROW; -- fetches next row from table
      -- other instructions

   END LOOP;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN return_val;
END //

DELIMITER ;
```

Examples

First a simplified example:

```
CREATE TABLE marks(stud_id INT, grade_count INT);

INSERT INTO marks VALUES (1,6), (2,4), (3,7), (4,5), (5,8);

SELECT * FROM marks;
+---------+-------------+
| stud_id | grade_count |
+---------+-------------+
|       1 |           6 |
|       2 |           4 |
|       3 |           7 |
|       4 |           5 |
|       5 |           8 |
+---------+-------------+

DELIMITER //
CREATE AGGREGATE FUNCTION IF NOT EXISTS aggregate_count(x INT) RETURNS INT
BEGIN
 DECLARE count_students INT DEFAULT 0;
 DECLARE CONTINUE HANDLER FOR NOT FOUND
 RETURN count_students;
      LOOP
          FETCH GROUP NEXT ROW;
          IF x  THEN
            SET count_students = count_students+1;
          END IF;
      END LOOP;
END //
DELIMITER ;
```

A non-trivial example that cannot easily be rewritten using existing functions:

```
DELIMITER //
CREATE AGGREGATE FUNCTION medi_int(x INT) RETURNS DOUBLE
BEGIN
  DECLARE CONTINUE HANDLER FOR NOT FOUND
    BEGIN
      DECLARE res DOUBLE;
      DECLARE cnt INT DEFAULT (SELECT COUNT(*) FROM tt);
      DECLARE lim INT DEFAULT (cnt-1) DIV 2;
      IF cnt % 2 = 0 THEN
        SET res = (SELECT AVG(a) FROM (SELECT a FROM tt ORDER BY a LIMIT lim,2) ttt);
      ELSE
        SET res = (SELECT a FROM tt ORDER BY a LIMIT lim,1);
      END IF;
      DROP TEMPORARY TABLE tt;
      RETURN res;
    END;
  CREATE TEMPORARY TABLE tt (a INT);
  LOOP
    FETCH GROUP NEXT ROW;
    INSERT INTO tt VALUES (x);
  END LOOP;
END //
DELIMITER ;
```

SQL/PL Example

This uses the same marks table as created above.

```
SET sql_mode=Oracle;
DELIMITER //

CREATE AGGREGATE FUNCTION aggregate_count(x INT) RETURN INT AS count_students INT DEFAULT 0;
BEGIN
   LOOP
      FETCH GROUP NEXT ROW;
      IF x  THEN
        SET count_students := count_students+1;
      END IF;
   END LOOP;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN count_students;
END aggregate_count //
DELIMITER ;

SELECT aggregate_count(stud_id) FROM marks;
```

See Also

- Stored Function Overview
- CREATE FUNCTION
- SHOW CREATE FUNCTION
- DROP FUNCTION
- Stored Routine Privileges
- SHOW FUNCTION STATUS
- Information Schema ROUTINES Table

This page is licensed: CC BY-SA / Gnu FDL

Stored Routine Privileges

It's important to give careful thought to the privileges associated with stored functions and stored procedures. The following is an explanation of how they work.

Creating Stored Routines

- To create a stored routine, the CREATE ROUTINE privilege is needed. The SUPER privilege is required if a `DEFINER` is declared that's not the creator's account (see DEFINER clause below). The `SUPER` privilege is also required if statement-based binary logging is used. See Binary Logging of Stored Routines for more details.

Altering Stored Routines

- To make changes to, or drop, a stored routine, the ALTER ROUTINE privilege is needed. The creator of a routine is temporarily granted this privilege if they attempt to change or drop a routine they created, unless the automatic_sp_privileges variable is set to `0` (it defaults to 1).
- The `SUPER` privilege is also required if statement-based binary logging is used. See Binary Logging of Stored Routines for more details.

Running Stored Routines

- To run a stored routine, the EXECUTE privilege is needed. This is also temporarily granted to the creator if they attempt to run their routine unless the automatic_sp_privileges variable is set to `0`.
- The SQL SECURITY clause (by default `DEFINER`) specifies what privileges are used when a routine is called. If `SQL SECURITY` is `INVOKER`, the function body are evaluated using the privileges of the user calling the function. If `SQL SECURITY` is `DEFINER`, the function body is always evaluated using the privileges of the definer account. `DEFINER` is the default. Thus, by default, users who can access the database associated with the stored routine can also run the routine, and potentially perform operations they wouldn't normally have permissions for.
- The creator of a routine is the account that ran the CREATE FUNCTION or CREATE PROCEDURE statement, regardless of whether a `DEFINER` is provided. The definer is by default the creator unless otherwise specified.
- The server automatically changes the privileges in the mysql.proc table as required, but will not look out for manual changes.

DEFINER Clause

If left out, the `DEFINER` is treated as the account that created the stored routine or view. If the account creating the routine has the `SUPER` privilege, another account can be specified as the `DEFINER`.

SQL SECURITY Clause

This clause specifies the context the stored routine or view will run as. It can take two values - `DEFINER` or `INVOKER`. `DEFINER` is the account specified as the `DEFINER` when the stored routine or view was created (see the section above). `INVOKER` is the account invoking the routine or view.

As an example, let's assume a routine, created by a superuser who's specified as the `DEFINER`, deletes all records from a table. If `SQL SECURITY=DEFINER`, anyone running the routine, regardless of whether they have delete privileges, are able to delete the records. If `SQL SECURITY = INVOKER`, the routine will only delete the records if the account invoking the routine has permission to do so.

`INVOKER` is usually less risky, as a user cannot perform any operations they're normally unable to. However, it's not uncommon for accounts to have relatively limited permissions, but be specifically granted access to routines, which are then invoked in the `DEFINER` context.

Dropping Stored Routines

All privileges that are specific to a stored routine are dropped when a DROP FUNCTION or DROP ROUTINE is run. However, if a CREATE OR REPLACE FUNCTION or CREATE OR REPLACE PROCEDURE is used to drop and replace and the routine, any privileges specific to that routine will not be dropped.

See Also

- Changing the DEFINER of MySQL stored routines etc. - maria.com post on what to do after you've dropped a user, and now want to change the DEFINER on all database objects that currently have it set to this dropped user.

This page is licensed: CC BY-SA / Gnu FDL

DROP FUNCTION

Syntax

```
DROP FUNCTION [IF EXISTS] f_name
```

Description

The `DROP FUNCTION` statement is used to drop a stored function or a user-defined function (UDF). That is, the specified routine is removed from the server, along with all privileges specific to the function. You must have the `ALTER ROUTINE` privilege for the routine in order to drop it. If the automatic_sp_privileges server system variable is set, both the `ALTER ROUTINE` and `EXECUTE` privileges are granted automatically to the routine creator - see Stored Routine Privileges.

IF EXISTS

The `IF EXISTS` clause is a MySQL/MariaDB extension. It prevents an error from occurring if the function does not exist. A `NOTE` is produced that can be viewed with SHOW WARNINGS.

For dropping a user-defined functions (UDF), see DROP FUNCTION UDF.

Examples

```
DROP FUNCTION hello;
Query OK, 0 rows affected (0.042 sec)

DROP FUNCTION hello;
ERROR 1305 (42000): FUNCTION test.hello does not exist

DROP FUNCTION IF EXISTS hello;
Query OK, 0 rows affected, 1 warning (0.000 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------+
| Level | Code | Message                           |
+-------+------+-----------------------------------+
| Note  | 1305 | FUNCTION test.hello does not exist |
+-------+------+-----------------------------------+
```

See Also

- DROP PROCEDURE
- Stored Function Overview
- CREATE FUNCTION
- CREATE FUNCTION UDF

- ALTER FUNCTION
- SHOW CREATE FUNCTION
- SHOW FUNCTION STATUS
- Stored Routine Privileges
- INFORMATION_SCHEMA ROUTINES Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

Stored Function Limitations

The following restrictions apply to stored functions.

- All of the restrictions listed in Stored Routine Limitations.
- Any statements that return a result set are not permitted. For example, a regular SELECTs is not permitted, but a SELECT INTO is. A cursor and FETCH statement is permitted.
- FLUSH statements are not permitted.
- Statements that perform explicit or implicit commits or rollbacks are not permitted.
- Cannot be used recursively.
- Cannot make changes to a table that is already in use (reading or writing) by the statement invoking the stored function.
- Cannot refer to a temporary table multiple times under different aliases, even in different statements.
- ROLLBACK TO SAVEPOINT and RELEASE SAVEPOINT statement which are in a stored function cannot refer to a savepoint which has been defined out of the current function.
- Prepared statements (PREPARE, EXECUTE, DEALLOCATE PREPARE) cannot be used, and therefore nor can statements be constructed as strings and then executed.

This page is licensed: CC BY-SA / Gnu FDL

Binary Logging of Stored Routines

Binary logging can be row-based, statement-based, or a mix of the two. See Binary Log Formats for more details on the formats. If logging is statement-based, it is possible that a statement will have different effects on the master and on the slave.

Stored routines are particularly prone to this, for two main reasons:

- stored routines can be non-deterministic, in other words non-repeatable, and therefore have different results each time they are run.
- the slave thread executing the stored routine on the slave holds full privileges, while this may not be the case when the routine was run on the master.

The problems with replication will only occur with statement-based logging. If row-based logging is used, since changes are made to rows based on the master's rows, there is no possibility of the slave and master getting out of sync.

By default, with row-based replication, triggers run on the master, and the effects of their executions are replicated to the slaves. However, it is possible to run triggers on the slaves. See Running triggers on the slave for Row-based events.

How MariaDB Handles Statement-Based Binary Logging of Routines

If the following criteria are met, then there are some limitations on whether stored routines can be created:

- The binary log is enabled, and the binlog_format system variable is set to `STATEMENT`. See Binary Log Formats for more information.

- The log_bin_trust_function_creators is set to `OFF`, which is the default value.

If the above criteria are met, then the following limitations apply:

- When a stored function is created, it must be declared as either `DETERMINISTIC`, `NO SQL` or `READS SQL DATA`, or else an error will occur. MariaDB cannot check whether a function is deterministic, and relies on the correct definition being used.
- To create or modify a stored function, a user requires the `SUPER` privilege as well as the regular privileges. See Stored Routine Privileges for these details.
- Triggers work in the same way, except that they are always assumed to be deterministic for logging purposes, even if this is obviously not the case, such as when they use the UUID function.
- Triggers can also update data. The slave uses the DEFINER attribute to determine which user is taken to have created the trigger.
- Note that the above limitations do no apply to stored procedures or to events.

Examples

A deterministic function:

```
DELIMITER //

CREATE FUNCTION trust_me(x INT)
RETURNS INT
DETERMINISTIC
READS SQL DATA
BEGIN
   RETURN (x);
END //

DELIMITER ;
```

A non-deterministic function, since it uses the UUID_SHORT function:

```
DELIMITER //

CREATE FUNCTION dont_trust_me()
RETURNS INT
BEGIN
   RETURN UUID_SHORT();
END //

DELIMITER ;
```

This page is licensed: CC BY-SA / Gnu FDL

Stored Routine Limitations

The following SQL statements are not permitted inside any stored routines (stored functions, stored procedures, events or triggers).

- ALTER VIEW; you can use CREATE OR REPLACE VIEW instead.
- LOAD DATA and LOAD TABLE.
- CHANGE MASTER TO
- INSERT DELAYED is permitted, but the statement is handled as a regular INSERT.
- LOCK TABLES and UNLOCK TABLES.
- References to local variables within prepared statements inside a stored routine (use user-defined variables instead).

- BEGIN (WORK) is treated as the beginning of a BEGIN END block, not a transaction, so START TRANSACTION needs to be used instead.
- The number of permitted recursive calls is limited to max_sp_recursion_depth. If this variable is 0 (default), recursivity is disabled. The limit does not apply to stored functions.
- Most statements that are not permitted in prepared statements are not permitted in stored programs. See Prepare Statement:Permitted statements for a list of statements that can be used. SIGNAL, RESIGNAL and GET DIAGNOSTICS are exceptions, and may be used in stored routines.

There are also further limitations specific to the kind of stored routine.

Note that, if a stored program calls another stored program, the latter will inherit the caller's limitations. So, for example, if a stored procedure is called by a stored function, that stored procedure will not be able to produce a result set, because stored functions can't do this.

See Also

- Stored Function Limitations
- Trigger Limitations
- Event Limitations

This page is licensed: CC BY-SA / Gnu FDL

DBMS_OUTPUT

> **Note:**
>
> This feature is available from MariaDB Enterprise Server 11.8.

Overview

Oracle documentation describing `DBMS_OUTPUT` can be found here:
https://docs.oracle.com/en/database/oracle/oracle-database/21/arpls/DBMS_OUTPUT.html

The main idea of `DBMS_OUTPUT` is:

- Messages submitted by `DBMS_OUTPUT.PUT_LINE()` are not sent to the client until the sending subprogram (or trigger) completes. There is no a way to flush output during the execution of a procedure.
- Therefore, lines are collected into a server side buffer, which, at the end of the current user statement, can be fetched to the client side using another SQL statement. Then, they can be read using a regular MariaDB Connector-C API. No changes in the client-protocol are needed.
- Oracle's SQLPlus uses the procedure `DBMS_PACKAGE.GET_LINES()` to fetch the output to the client side as an array of strings.
- For JDBC, using `GET_LINES()` is preferable, because it's more efficient than individual `GET_LINE()` calls.

Package Routines

Package Overview

MariaDB implements all routines supported by Oracle, except `GET_LINES()`:

- Procedure `ENABLE()` - enable the routines.
- Procedure `DISABLE()` - disable the routines. If the package is disabled, all calls to subprograms, such as `PUT()` and `PUT_LINE()`, are ignored (or exit immediately without doing anything).
- Procedure `PUT_LINE()` - submit a line into the internal buffer.
- Procedure `PUT()` - submit a partial line into the buffer.

- Procedure `NEW_LINE()` - terminate a line submitted by `PUT()`.
- Procedure `GET_LINE()` - read one line (the earliest) from the buffer. When a line is read by `GET_LINE()`, it's automatically removed from the buffer.
- Procedure `GET_LINES()` - read all lines (as an array of strings) from the buffer - this procedure isn't implemented.

The package starts in disabled mode, so an explicit enabling is needed:

```
CALL DBMS_OUTPUT.ENABLE;
```

Details

If a call for `GET_LINE` or `GET_LINES` did not retrieve all lines, then a subsequent call for `PUT`, `PUT_LINE`, or `NEW_LINE` discards the remaining lines (to avoid confusing with the next message). This script demonstrates the principle:

```
DROP TABLE t1;
CREATE TABLE t1 (line VARCHAR2(400), status INTEGER);
DECLARE
  line   VARCHAR2(400);
  status INTEGER;
BEGIN
  DBMS_OUTPUT.PUT_LINE('line1');
  DBMS_OUTPUT.PUT_LINE('line2');
  DBMS_OUTPUT.GET_LINE(line, status);
  INSERT INTO t1 VALUES (line, status);
  DBMS_OUTPUT.PUT_LINE('line3'); -- This clears the buffer (removes line2) before putting line3
  LOOP
    DBMS_OUTPUT.GET_LINE(line, status);
    INSERT INTO t1 VALUES (line, status);
    EXIT WHEN status <> 0;
  END LOOP;
END;
/
SELECT * FROM t1;
```

LINE	STATUS
line1	0
line3	0
-	1

Data Type for the Buffer

Oracle uses this data type as a storage for the buffer:

```
TYPE CHARARR IS TABLE OF VARCHAR2(32767) INDEX BY BINARY_INTEGER;
```

Like Oracle, MariaDB uses an associative array as a storage for the buffer.

Data Type Used for GET_LINES()

> **Warning:**
>
> This functionality is not implemented.

In Oracle, the function `GET_LINES()` returns an array of strings of this data type:

```
TYPE DBMSOUTPUT_LINESARRAY IS VARRAY(2147483647) OF VARCHAR2(32767);
```

MariaDB does not have array data types in the C and C++ connectors, so they can't take advantage of `GET_LINES()` in a client program.

Fetching all Lines in a PL/SQL Program

Fetching all lines in a PL/SQL program is implemented using a loop of `sys.DBMS_OUTPUT.GET_LINE()` calls:

```
SET sql_mode=ORACLE;
DELIMITER /
DECLARE
  all_lines MEDIUMTEXT CHARACTER SET utf8mb4 :='';
  line MEDIUMTEXT CHARACTER SET utf8mb4;
  status INT;
BEGIN
  sys.DBMS_OUTPUT.PUT_LINE('line1');
  sys.DBMS_OUTPUT.PUT_LINE('line2');
  sys.DBMS_OUTPUT.PUT_LINE('line3');
  LOOP
    sys.DBMS_OUTPUT.GET_LINE(line, status);
    EXIT WHEN status > 0;
    all_lines:= all_lines || line || '\n';
  END LOOP;
  SELECT all_lines;
END;
/
DELIMITER ;
```

Fetching all Lines on the Client Side

Fetching all lines on the client side (for instance, in a C program using Connector/C) is done by using a loop of `sys.DBMS_OUTPUT.GET_LINE()` queries.

Limits

Oracle has the following limits:

- The maximum individual line length (sent to `DBMS_OUTPUT`) is 32767 bytes.
- The default buffer size is 20000 bytes. The minimum size is 2000 bytes. The maximum is unlimited.

MariaDB also implements some limits, either using the total size of all rows or using the row count.

Installation

Like other bootstrap scripts, the script creating `DBMS_OUTPUT`:

- Is put into a new separate `/scripts/dbms_ouput.sql` file in the source directory;
- Is installed into `/share/dbms_ouput.sql` of the installation directory.

\

This page is licensed: CC BY-SA / Gnu FDL

Storage Engines Overview

Overview

MariaDB features pluggable storage engines to allow per-table workload optimization.

A storage engine is a type of plugin for MariaDB:

- Different storage engines may be optimized for different workloads, such as transactional workloads, analytical workloads, or high throughput workloads.

- Different storage engines may be designed for different use cases, such as federated table access, table sharding, and table archiving in the cloud.
- Different tables on the same server may use different storage engines.

Engine	Target	Optimization	Availability
Aria	Read-Heavy	Reads	ES 10.5+
ColumnStore	Analytics, HTAP	Big Data, Analytical	ES 10.5+
InnoDB	General Purpose	Mixed Read/Write	ES 10.5+
Memory	Cache, Temp	Temporary Data	ES 10.5+
MyISAM	Reads	Reads	ES 10.5+
MyRocks	Write-Heavy	I/O Reduction, SSD	ES 10.5+
S3	Cloud	Read-Only	ES 10.5+
Spider	Federation	Sharding, Interlink	ES 10.5+

Examples

Identify the Default Storage Engine

Identify the server's global default storage engine by using SHOW GLOBAL VARIABLES to query the default_storage_engine system variable:

```
SHOW GLOBAL VARIABLES LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

Identify the session's default storage engine by using SHOW SESSION VARIABLES:

```
SHOW SESSION VARIABLES LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

Set the Default Storage Engine

Global default storage engine:

```
SET GLOBAL default_storage_engine='MyRocks';
```

Session default storage engine supersedes global default during this session:

```
SET SESSION default_storage_engine='MyRocks';
```

Configure the Default Storage Engine

```
[mariadb]
...
default_storage_engine=MyRocks
```

Identify Available Storage Engines

```
SHOW ENGINES;
```

Choose Storage Engine for a New Table

Storage engine is specified at time of table creation using a ENGINE = parameter.

```
CREATE TABLE accounts.messages (
  id INT PRIMARY KEY AUTO_INCREMENT,
  sender_id INT,
  receiver_id INT,
  message TEXT
) ENGINE = MyRocks;
```

Resources

Engines for System Tables

Standard MariaDB storage engines are used for System Table storage:

- Aria Storage Engine
- MyISAM Storage Engine

FAQ

Can I use more than one storage engine on a server?

- Yes, different tables can use different storage engines on the same server.
- To create a table with a specific storage engine, specify the ENGINE table option to the CREATE TABLE statement.

Can I use more than one storage engine in a single query?

- Yes, a single query can reference tables that use multiple storage engines.
- In some cases, special configuration may be required. For example, ColumnStore requires cross engine joins to be configured.

What storage engine should I use for transactional or OLTP workloads?

- InnoDB is the recommended storage engine for transactional or OLTP workloads.

What storage engine should I use for analytical or OLAP workloads?

- ColumnStore is the recommended storage engine for analytical or OLAP workloads.

What storage engine should I use if my application performs both transactional and analytical queries?

An application that performs both transactional and analytical queries is known as hybrid transactional-analytical processing (HTAP).

HTAP can be implemented with MariaDB by using InnoDB for transactional queries and ColumnStore for analytical queries.

Reference

MariaDB Server Reference

- Plugins.

- Information Schema ENGINES table, which shows available storage engines.
- Information Schema TABLES table, which shows storage engine by table.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Choosing the Right Storage Engine

A high-level overview of the main reasons for choosing a particular storage engine:

> **Note:**
>
> CONNECT, Mroonga, MyRocks, OQGRAPH, SphinxSE, and VIDEX are community contributions, developed and maintained by their respective contributors. Each of those engines' pages says who maintains it.

Topic List

General Purpose

- InnoDB is a good general transaction storage engine, and the best choice in most cases. It is the default storage engine.
- Aria, MariaDB's more modern improvement on MyISAM, has a small footprint and allows for easy copying between systems.
- MyISAM has a small footprint and allows for easy copying between systems. MyISAM is MySQL's oldest storage engine. There is usually little reason to use it except for legacy purposes. Aria is MariaDB's more modern improvement.
- XtraDB is no longer available. It was a performance-enhanced fork of InnoDB and was MariaDB's default engine until MariaDB 10.1.

Scaling, Partitioning

When you want to split your database load on several servers or optimize for scaling. We also suggest looking at Galera, a synchronous multi-master cluster.

- Spider uses partitioning to provide data sharding through multiple servers.
- ColumnStore utilizes a massively parallel distributed data architecture and is designed for big data scaling to process petabytes of data.
- The MERGE storage engine is a collection of identical MyISAM tables that can be used as one. "Identical" means that all tables have identical column and index information.

Compression / Archive

- MyRocks enables greater compression than InnoDB, as well as less write amplification giving better endurance of flash storage and improving overall throughput.
- The Archive storage engine is, unsurprisingly, best used for archiving.

Connecting to Other Data Sources

When you want to use data not stored in a MariaDB database.

- The CSV storage engine can read and append to files stored in CSV (comma-separated-values) format. However, since MariaDB 10.0, CONNECT is a better choice and is more flexibly able to read and write such files.

Search Optimized

Search engines optimized for search.

- SphinxSE is used as a proxy to run statements on a remote Sphinx database server (mainly useful for advanced fulltext searches).
- Mroonga provides fast CJK-ready full text searching using column store.

Cache, Read-only

- MEMORY does not write data on-disk (all rows are lost on crash) and is best-used for read-only caches of data from other tables, or for temporary work areas. With the default InnoDB and other storage engines having good caching, there is less need for this engine than in the past.

Other Specialized Storage Engines

- S3 Storage Engine is a read-only storage engine that stores its data in Amazon S3.
- Sequence allows the creation of ascending or descending sequences of numbers (positive integers) with a given starting value, ending value and increment, creating virtual, ephemeral tables automatically when you need them.
- The BLACKHOLE storage engine accepts data but does not store it and always returns an empty result. This can be useful in replication environments, for example, if you want to run complex filtering rules on a slave without incurring any overhead on a master.
- OQGRAPH allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions).
- The VIDEX storage engine is an aggregated, extensible engine suitable for what-if analyses in MariaDB. The name is derived from [VI]rtual in[DEX]. Using VIDEX, you can evaluate how potential indexes (and optimizer decisions such as join orders) would change query plans *without creating real indexes on production data*.

Alphabetical List

- The Archive storage engine is, unsurprisingly, best used for archiving.
- Aria, MariaDB's more modern improvement on MyISAM, has a small footprint and allows for easy copy between systems.
- The BLACKHOLE storage engine accepts data but does not store it and always returns an empty result. This can be useful in replication environments, for example, if you want to run complex filtering rules on a slave without incurring any overhead on a master.
- ColumnStore utilizes a massively parallel distributed data architecture and is designed for big data scaling to process petabytes of data.
- CONNECT allows access to different kinds of text files and remote resources as if they were regular MariaDB tables.
- The CSV storage engine can read and append to files stored in CSV (comma-separated-values) format. However, since MariaDB 10.0, CONNECT is a better choice and is more flexibly able to read and write such files.
- InnoDB is a good general transaction storage engine, and the best choice in most cases. It is the default storage engine.
- The MERGE storage engine is a collection of identical MyISAM tables that can be used as one. "Identical" means that all tables have identical column and index information.
- MEMORY does not write data on-disk (all rows are lost on crash) and is best-used for read-only caches of data from other tables, or for temporary work areas. With the default InnoDB and other storage engines having good caching, there is less need for this engine than in the past.
- Mroonga provides fast CJK-ready full text searching using column store.
- MyISAM has a small footprint and allows for easy copying between systems. MyISAM is MySQL's oldest storage engine. There is usually little reason to use it except for legacy purposes. Aria is MariaDB's more modern improvement.

- MyRocks enables greater compression than InnoDB, as well as less write amplification giving better endurance of flash storage and improving overall throughput.
- OQGRAPH allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions).
- S3 Storage Engine is a read-only storage engine that stores its data in Amazon S3.
- Sequence allows the creation of ascending or descending sequences of numbers (positive integers) with a given starting value, ending value and increment, creating virtual, ephemeral tables automatically when you need them.
- SphinxSE is used as a proxy to run statements on a remote Sphinx database server (mainly useful for advanced fulltext searches).
- Spider uses partitioning to provide data sharding through multiple servers.
- The VIDEX storage engine is an aggregated, extensible engine suitable for what-if analyses in MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

ARCHIVE

The `ARCHIVE` storage engine is a storage engine that uses gzip to compress rows. It is mainly used for storing large amounts of data, without indexes, with only a very small footprint.

A table using the `ARCHIVE` storage engine is stored in two files on disk. There's a table definition file with an extension of .frm, and a data file with the extension .ARZ. At times during optimization, a .ARN file will appear.

New rows are inserted into a compression buffer and are flushed to disk when needed. SELECTs cause a flush. Sometimes, rows created by multi-row inserts are not visible until the statement is complete.

`ARCHIVE` allows a maximum of one key. The key must be on an AUTO_INCREMENT column, and can be a `PRIMARY KEY` or a non-unique key. However, it has a limitation: it is not possible to insert a value which is lower than the next `AUTO_INCREMENT` value.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'ha_archive';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = ha_archive
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'ha_archive';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time

the server is restarted.

Characteristics

- Supports INSERT and SELECT, but not DELETE, UPDATE or REPLACE.
- Data is compressed with zlib as it is inserted, making it very small.
- Data is slow the select, as it needs to be uncompressed, and, besides the query cache, there is no cache.
- Supports AUTO_INCREMENT (since MariaDB/MySQL 5.1.6), which can be a unique or a non-unique index.
- Since MariaDB/MySQL 5.1.6, selects scan past BLOB columns unless they are specifically requested, making these queries much more efficient.
- Does not support spatial data types.
- Does not support transactions.
- Does not support foreign keys.
- Does not support virtual columns.
- No storage limit.
- Supports row locking.
- Supports table discovery, and the server can access ARCHIVE tables even if the corresponding `.frm` file is missing.
- OPTIMIZE TABLE and REPAIR TABLE can be used to compress the table in its entirety, resulting in slightly better compression.
- With MariaDB, it is possible to upgrade from the MySQL 5.0 format without having to dump the tables.
- INSERT DELAYED is supported.
- Running many SELECTs during the insertions can deteriorate the compression, unless only multi-rows INSERTs and INSERT DELAYED are used.

This page is licensed: CC BY-SA / Gnu FDL

ARIA

See Aria Storage Engine.

An overview of Aria, a storage engine designed as a crash-safe alternative to MyISAM, featuring transactional capabilities and improved caching.

See Aria Group Commit.

Learn about Aria's group commit functionality, which improves performance by batching commit operations to the transaction log.

See Aria Status Variables.

A list of status variables specific to the Aria engine, providing metrics on page cache usage, transaction log syncs, and other internal operations.

See Aria Storage Formats.

Understand the different row formats supported by Aria, particularly the default PAGE format which enables crash safety and better concurrency.

See Aria System Variables.

A comprehensive list of system variables for configuring Aria, including buffer sizes, log settings, and recovery options.

See Aria Two-step Deadlock Detection.

Explains Aria's deadlock detection mechanism, which uses a two-step process with configurable search depths and timeouts to resolve conflicts.

See Aria FAQ.

Frequently asked questions about the Aria storage engine, covering its history, comparison with MyISAM, and key features like crash safety.

See The Aria Name.

A brief history of the naming of the Aria storage engine, explaining its origins as "Maria" and the reasons for the eventual name change.

This page is licensed: CC BY-SA / Gnu FDL

Aria Storage Engine

The Aria storage engine is compiled in by default from MariaDB 5.1 and it is required to be 'in use' when MariaDB is started.

All system tables are Aria.

Additionally, internal on-disk tables are in the Aria table format instead of the MyISAM table format. This should speed up some GROUP BY and DISTINCT queries because Aria has better caching than MyISAM.

Note: The ***Aria*** storage engine was previously called *Maria* (see The Aria Name for details on the rename) and in previous versions of MariaDB the engine was still called Maria.

The following table options to Aria tables in CREATE TABLE and ALTER TABLE:

- `TRANSACTIONAL= 0 | 1` : If the `TRANSACTIONAL` table option is set for an Aria table, then the table are crash-safe. This is implemented by logging any changes to the table to Aria's transaction log, and syncing those writes at the end of the statement. This will marginally slow down writes and updates. However, the benefit is that if the server dies before the statement ends, all non-durable changes will roll back to the state at the beginning of the statement. This also needs up to 6 bytes more for each row and key to store the transaction id (to allow concurrent insert's and selects).
  - `TRANSACTIONAL=1` is not supported for partitioned tables.
  - An Aria table's default value for the `TRANSACTIONAL` table option depends on the table's value for the `ROW_FORMAT` table option. See below for more details.
  - If the `TRANSACTIONAL` table option is set for an Aria table, the table does not actually support transactions. See MDEV-21364 for more information. In this context, transactional just means crash-safe.
- `PAGE_CHECKSUM= 0 | 1` : If index and data should use page checksums for extra safety.
- `TABLE_CHECKSUM= 0 | 1` :
  Same as `CHECKSUM` in MySQL 5.1
- `ROW_FORMAT=PAGE | FIXED | DYNAMIC` : The table's row format.
  - The default value is `PAGE`.
  - To emulate MyISAM, set `ROW_FORMAT=FIXED` or `ROW_FORMAT=DYNAMIC`

The `TRANSACTIONAL` and `ROW_FORMAT` table options interact as follows:

- If `TRANSACTIONAL=1` is set, then the only supported row format is `PAGE`. If `ROW_FORMAT` is set to some other value, then Aria issues a warning, but still forces the row format to be `PAGE`.
- If `TRANSACTIONAL=0` is set, then the table are not be crash-safe, and any row format is supported.
- If `TRANSACTIONAL` is not set to any value, then any row format is supported. If `ROW_FORMAT` is set, then the table will use that row format. Otherwise, the table will use the default `PAGE` row format. In this case, if the table uses the `PAGE` row format, then it are crash-safe. If it uses some other row format, then it will not be crash-safe.

Some other improvements are:

- CHECKSUM TABLE now ignores values in `NULL` fields. This makes `CHECKSUM TABLE` faster and fixes some cases where same table definition could give different checksum values depending on row format. The disadvantage is that the value is now different compared to other MySQL installations. The new checksum calculation is fixed for all table engines that uses the default way to calculate and MyISAM which does the calculation internally. Note: Old MyISAM tables with internal checksum returns the same checksum as before. To fix them to calculate according to new rules you have to do an ALTER TABLE. You can use the old ways to calculate checksums by using the option `--old` to mariadbdmysqld or set the system variable '`@@old`' to `1` when you do `CHECKSUM TABLE ... EXTENDED;`
- At startup Aria will check the Aria logs and automatically recover the tables from the last checkpoint if the server was not taken down correctly. See Aria Log Files

Startup Options for Aria

For a full list, see Aria System Variables.

In normal operations, the only variables you have to consider are:

- aria-pagecache-buffer-size
  - This is where all index and data pages are cached. The bigger this is, the faster Aria will work.
- aria-block-size
  - The default value 8192, should be ok for most cases. The only problem with a higher value is that it takes longer to find a packed key in the block as one has to search roughly 8192/2 to find each key. We plan to fix this by adding a dictionary at the end of the page to be able to do a binary search within the block before starting a scan. Until this is done and key lookups takes too long time even if you are not hitting disk, then you should consider making this smaller.
  - Possible values to try are `2048`, `4096` or `8192`
  - Note that you can't change this without dumping, deleting old tables and deleting all log files and then restoring your Aria tables. (This is the only option that requires a dump and load.)
- aria-log-purge-type
  - Set this to "`at_flush`" if you want to keep a copy of the transaction logs (good as an extra backup). The logs will stay around until you execute FLUSH ENGINE LOGS.

Aria Log Files

`aria_log_control` file is a very short log file (52 bytes) that contains the current state of all Aria tables related to logging and checkpoints. In particular, it contains the following information:

```
Aria file version: 1
Block size: 8192
maria_uuid: ee948482-6cb7-11ed-accb-3c7c3ff16468
last_checkpoint_lsn: (1,0x235a)
last_log_number: 1
trid: 28
recovery_failures: 0
```

- The `uuid` is a unique identifier per system. All Aria files created will have a copy of this in their .MAI headers. This is mainly used to check if someone has copied an Aria file between MariaDB servers.
- `last_checkpoint_lsn` and `last_log_number` are information about the current aria_log files.
- `trid` is the highest transaction number seen so far. Used by recovery.

`aria_log.*` files contain the log of all operations that change Aria files (including create table, drop table, insert etc..) This is a 'normal' WAL (Write Ahead Log), similar to the InnoDB log file, except that aria_logs contain both redo and undo. Old aria_log files are automatically deleted when they are not needed anymore (Neither the last checkpoint or any running transaction need to refer to the old data anymore).

Missing valid id

The error `Missing valid id at start of file. File is not a valid aria control file` means that something overwrote at least the first 4 bytes in the file. This can happen due to a problem with the file system (hardware or software), or a bug in which a thread inside MariaDB wrote on the wrong file descriptor (in which case you should report the bug, attaching a copy of the control file to assist).

In the case of a corrupted log file, with the server shut down, one should be able to fix that by deleting all aria_log files.
If the control_file is corrupted, then one has to delete the aria_control_file and all aria_log.* files.
The effect of this is that on table open of an Aria table, the server will think that it has been moved from another system and do an automatic check and repair of it. If there was no issues, the table are opened and can be used as normal. See also When is it safe to remove old log files.

See Also

- Aria FAQ

This page is licensed: CC BY-SA / Gnu FDL

Aria Group Commit

The Aria storage engine includes a feature to group commits to speed up concurrent threads doing many inserts into the same or different Aria tables.

By default, group commit for Aria is turned off. It is controlled by the aria_group_commit and aria_group_commit_interval system variables.

Information on setting server variables can be found on the Server System Variables page.

Terminology

- A `commit` is `flush of logs` followed by a sync.
- `sent to disk` means written to disk but not sync()ed,
- `flushed` mean sent to disk and synced().
- `LSN` means log serial number. It's refers to the position in the transaction log.

Non Group commit logic (aria_group_commit="none")

The thread which first started the `commit` is performing the actual flush of logs. Other threads set the new goal (LSN) of the next pass (if it is maximum) and wait for the pass end or just wait for the pass end.

The effect of this is that a flush (write of logs + sync) will save all data for all threads/transactions that have been waiting since the last flush.

If hard group commit is enabled (aria_group_commit="hard")

If hard commit and aria_group_commit_interval=0

The first thread sends all changed buffers to disk. This is repeated as long as there are new LSNs added. The process can not loop forever because we have a limited number of threads and they will wait for the data to be synced.

Pseudo code:

```
do
   send changed buffers to disk
 while new_goal
sync
```

If hard commit and aria_group_commit_interval > 0

If less than rate microseconds has passed since the last sync, then after buffers have been sent to disk, wait until rate microseconds has passed since last sync, do sync and return.
This ensures that if we call sync infrequently we don't do any waits.

If soft group commit is enabled (aria_group_commit="soft")

Note that soft group commit should only be used if you can afford to lose a few rows if your machine shuts down hard (as in the case of a power failure).

Works like in `non group commit'` but the thread doesn't do any real sync(). If `aria_group_commit_interval` is not zero, the `sync()` calls are performed by a service thread with the given rate when needed (new LSN appears). If aria_group_commit_interval is zero, there are no `sync()` calls.

Code

The code for this can be found in `storage/maria/ma_loghandler.c::translog_flush()`.

This page is licensed: CC BY-SA / Gnu FDL

Aria Status Variables

This page documents status variables related to the Aria storage engine. See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

See also the Full list of MariaDB options, system and status variables.

`Aria_pagecache_blocks_not_flushed`

- Description: The number of dirty blocks in the Aria page cache. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Aria_pagecache_blocks_unused`

- Description: Free blocks in the Aria page cache. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Aria_pagecache_blocks_used`

- Description: Blocks used in the Aria page cache. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Aria_pagecache_read_requests`

- Description: The number of requests to read something from the Aria page cache.
- Scope: Global

- Data Type: `numeric`

`Aria_pagecache_reads`

- Description: The number of Aria page cache read requests that caused a block to be read from the disk.
- Scope: Global
- Data Type: `numeric`

`Aria_pagecache_write_requests`

- Description: The number of requests to write a block to the Aria page cache.
- Scope: Global
- Data Type: `numeric`

`Aria_pagecache_writes`

- Description: The number of blocks written to disk from the Aria page cache.
- Scope: Global
- Data Type: `numeric`

`Aria_transaction_log_syncs`

- Description: The number of Aria log fsyncs.
- Scope: Global
- Data Type: `numeric`

This page is licensed: CC BY-SA / Gnu FDL

Aria Storage Formats

The Aria storage engine supports three different table storage formats.

These are FIXED, DYNAMIC and PAGE, and they can be set with the ROW FORMAT option in the CREATE TABLE statement. PAGE is the default format, while FIXED and DYNAMIC are essentially the same as the MyISAM formats.

The SHOW TABLE STATUS statement can be used to see the storage format used by a table.

Fixed-length

Fixed-length (or static) tables contain records of a fixed-length. Each column is the same length for all records, regardless of the actual contents. It is the default format if a table has no BLOB, TEXT, VARCHAR or VARBINARY fields, and no ROW FORMAT is provided. You can also specify a fixed table with ROW_FORMAT=FIXED in the table definition.

Tables containing BLOB or TEXT fields cannot be FIXED, as by design these are both dynamic fields.

Fixed-length tables have a number of characteristics:

- fast, since MariaDB will always know where a record begins
- easy to cache
- take up more space than dynamic tables, as the maximum amount of storage space are allocated to each record.
- reconstructing after a crash is uncomplicated due to the fixed positions
- no fragmentation or need to re-organize, unless records have been deleted and you want to free the space up.

Dynamic

Dynamic tables contain records of a variable length. It is the default format if a table has any `BLOB`, `TEXT`, `VARCHAR` or `VARBINARY` fields, and no `ROW FORMAT` is provided. You can also specify a `DYNAMIC` table with `ROW_FORMAT=DYNAMIC` in the table definition.

Dynamic tables have a number of characteristics

- Each row contains a header indicating the length of the row.
- Rows tend to become fragmented easily. UPDATING a record to be longer will likely ensure it is stored in different places on the disk.
- All string columns with a length of four or more are dynamic.
- They require much less space than fixed-length tables.
- Restoring after a crash is more complicated than with `FIXED` tables.

Page

Page format is the default format for Aria tables, and is the only format that can be used if `TRANSACTIONAL=1`.

Page tables have a number of characteristics:

- It's cached by the page cache, which gives better random performance as it uses fewer system calls.
- Does not fragment as easily as the `DYNAMIC` format during `UPDATES`. The maximum number of fragments are very low.
- Updates more quickly than dynamic tables.
- Has a slight storage overhead, mainly notable on very small rows
- Slower to perform a full table scan
- Slower if there are multiple duplicated keys, as Aria will first write a row, then keys, and only then check for duplicates

Transactional

See Aria Storage Engine for the impact of the `TRANSACTIONAL` option on the row format.

This page is licensed: CC BY-SA / Gnu FDL

Aria System Variables

This page documents system variables related to the Aria storage engine. For options that are not system variables, see Aria Options.

See Server System Variables for instructions on setting system variables.

> **Note:**
>
> **For a full list of server options, system variables and status variables, see this page.**

`aria_block_size`

- Description: Block size to be used for Aria index pages. Changing this requires dumping, deleting old tables and deleting all log files, and then restoring your Aria tables. If key lookups take too long (and one has to search roughly 8192/2 by default to find each key), can be made smaller, e.g. `4096`.
- Command line: `--aria-block-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`

- Default Value: `8192`
- Range:
  - = MariaDB 10.4.7: `4096` to `32768` in increments of `1024`
  - <= MariaDB 10.4.6: `1024` to `32768` in increments of `1024`

`aria_checkpoint_interval`

- Description: Interval in seconds between automatic checkpoints. 0 means 'no automatic checkpoints' which makes sense only for testing.
- Command line: `--aria-checkpoint-interval=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `30`
- Range: `0` to `4294967295`

`aria_checkpoint_log_activity`

- Description: Number of bytes that the transaction log has to grow between checkpoints before a new checkpoint is written to the log.
- Command line: `aria-checkpoint-log-activity=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1048576`
- Range `0` to `4294967295`

`aria_encrypt_tables`

- Description: Enables automatic encryption of all user-created Aria tables that have the ROW_FORMAT table option set to PAGE. See Data at Rest Encryption and Enabling Encryption for User-created Tables.
- Command line: `aria-encrypt-tables={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`aria_force_start_after_recovery_failures`

- Description: Number of consecutive log recovery failures after which logs are automatically deleted to cure the problem; 0 (the default) disables the feature.
- Command line: `--aria-force-start-after-recovery-failures=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`

`aria_group_commit`

- Description: Specifies Aria group commit mode.

- Command line: `--aria_group_commit="value"`
- Alias: `maria_group_commit`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Valid values:
  - `none` - Group commit is disabled.
  - `hard` - Wait the number of microseconds specified by aria_group_commit_interval before actually doing the commit. If the interval is 0 then just check if any other threads have requested a commit during the time this commit was preparing (just before sync() file) and send their data to disk also before sync().
  - `soft` - The service thread will wait the specified time and then sync() to the log. If the interval is 0 then it won't wait for any commits (this is dangerous and should generally not be used in production)
- Default Value: `none`

`aria_group_commit_interval`

- Description: Interval between Aria group commits in microseconds (1/1000000 second) for other threads to come and do a commit in "hard" mode and sync()/commit at all in "soft" mode. Option only has effect if aria_group_commit is used.
- Command line: `--aria_group_commit_interval=#`
- Alias: `maria_group_commit_interval`
- Scope: Global
- Dynamic: No
- Type: numeric
- Valid Values:
  - Default Value: `0` (no waiting)
  - Range: `0-4294967295`

`aria_log_dir_path`

- Description: Path to the directory where transactional log should be stored
- Command line: `--aria-log-dir-path=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: Same as DATADIR
- Introduced: MariaDB 10.5.20, MariaDB 10.6.13, MariaDB 10.11.3 (as a system variable, existed as an option only before that)

`aria_log_file_size`

- Description: Limit for Aria transaction log size
- Command line: `--aria-log-file-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1073741824`

`aria_log_purge_type`

- Description: Specifies how the Aria transactional log are purged. Set to `at_flush` to keep a copy of the transaction logs (good as an extra backup). The logs will stay until the next FLUSH LOGS;
- Command line: `--aria-log-purge-type=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `immediate`
- Valid Values: `immediate`, `external`, `at_flush`

`aria_max_sort_file_size`

- Description: Don't use the fast sort index method to created index if the temporary file would get bigger than this.
- Command line: `--aria-max-sort-file-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `9223372036853727232`
- Range: `0` to `9223372036854775807`

`aria_page_checksum`

- Description: Determines whether index and data should use page checksums for extra safety. Can be overridden per table with PAGE_CHECKSUM clause in CREATE TABLE.
- Command line: `--aria-page-checksum=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`aria_pagecache_age_threshold`

- Description: This characterizes the number of hits a hot block has to be untouched until it is considered aged enough to be downgraded to a warm block. This specifies the percentage ratio of that number of hits to the total number of blocks in the page cache.
- Command line: `--aria-pagecache-age-threshold=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `300`
- Range: `100` to `9999900`

`aria_pagecache_buffer_size`

- Description: The size of the buffer used for index and data blocks for Aria tables. This can include explicit Aria tables, system tables, and temporary tables. Increase this to get better handling and measure by looking at aria-status-variables/#aria_pagecache_reads (should be small) vs aria-status-variables/#aria_pagecache_read_requests.
- Command line: `--aria-pagecache-buffer-size=#`

- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `134217728` (128MB)
- Range: `131072` (128KB) upwards

`aria_pagecache_division_limit`

- Description: The minimum percentage of warm blocks in the key cache.
- Command line: `--aria-pagecache-division-limit=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `1` to `100`

`aria_pagecache_file_hash_size`

- Description: Number of hash buckets for open and changed files. If you have many Aria files open you should increase this for faster flushing of changes. A good value is probably 1/10th of the number of possible open Aria files.
- Command line: `--aria-pagecache-file-hash-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `512`
- Range: `128` to `16384`

`aria_pagecache_segments`

- Description: The number of segments in the page_cache. Each file is put in their own segments of size pagecache_buffer_size / segments. Having many segments improves parallel performance.
- Command line: `--aria-pagecache-segments=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `128`
- Introduced: MariaDB Community Server 12.1, MariaDB Enterprise Server 11.4.8-5, MariaDB Enterprise Server 11.8

`aria_recover`

- Description: `aria_recover` has been renamed to `aria_recover_options` in MariaDB 10.2.0. See aria_recover_options for the description.

`aria_recover_options`

- Description: Specifies how corrupted tables should be automatically repaired. More than one option can be specified, for example `FORCE,BACKUP`.
  - `NORMAL`: Normal automatic repair, the default until MariaDB 10.2.3

    - `OFF`: Autorecovery is disabled, the equivalent of not using the option
    - `QUICK`: Does not check rows in the table if there are no delete blocks.
    - `FORCE`: Runs the recovery even if it determines that more than one row from the data file are lost.
    - `BACKUP`: Keeps a backup of the data files.
- Command line: `--aria-recover-options[=#]`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value:
    - `BACKUP,QUICK` (>= MariaDB 10.2.4)
    - `NORMAL` (<= MariaDB 10.2.3)
- Valid Values: `NORMAL`, `BACKUP`, `FORCE`, `QUICK`, `OFF`
- Introduced: MariaDB 10.2.0

`aria_repair_threads`

- Description: Number of threads to use when repairing Aria tables. The value of 1 disables parallel repair. Increasing from the default will usually result in faster repair, but will use more CPU and memory.
- Command line: `--aria-repair-threads=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`

`aria_sort_buffer_size`

- Description: The buffer that is allocated when sorting the index when doing a REPAIR or when creating indexes with CREATE INDEX or ALTER TABLE.
- Command line: `--aria-sort-buffer-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `268434432`

`aria_stats_method`

- Description: Determines how NULLs are treated for Aria index statistics purposes. If set to `nulls_equal`, all NULL index values are treated as a single group. This is usually fine, but if you have large numbers of NULLs the average group size is slanted higher, and the optimizer may miss using the index for ref accesses when it would be useful. If set to `nulls_unequal`, the default, the opposite approach is taken, with each NULL forming its own group of one. Conversely, the average group size is slanted lower, and the optimizer may use the index for ref accesses when not suitable. Setting to `nulls_ignored` ignores NULLs altogether from index group calculations. Statistics need to be recalculated after this method is changed. See also Index Statistics, myisam_stats_method and innodb_stats_method.
- Command line: `--aria-stats-method=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`

- Default Value: `nulls_unequal`
- Valid Values: `nulls_equal`, `nulls_unequal`, `nulls_ignored`

`aria_sync_log_dir`

- Description: Controls syncing directory after log file growth and new file creation.
- Command line: `--aria-sync-log-dir=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `NEWFILE`
- Valid Values: `NEWFILE`, `NEVER`, `ALWAYS`

`aria_used_for_temp_tables`

- Description: Readonly variable indicating whether the Aria storage engine is used for temporary tables. If set to `ON`, the default, the Aria storage engine is used. If set to `OFF`, MariaDB reverts to using MyISAM for on-disk temporary tables. The MEMORY storage engine is used for temporary tables regardless of this variable's setting where appropriate. The default can be changed by not using the `--with-aria-tmp-tables` option when building MariaDB.
- Command line: No
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`deadlock_search_depth_long`

- Description: Long search depth for the two-step deadlock detection. Only used by the Aria storage engine.
- Command line: `--deadlock-search-depth-long=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `15`
- Range: `0` to `33`

`deadlock_search_depth_short`

- Description: Short search depth for the two-step deadlock detection. Only used by the Aria storage engine.
- Command line: `--deadlock-search-depth-short=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4`
- Range: `0` to `32`

`deadlock_timeout_long`

- Description: Long timeout in microseconds for the two-step deadlock detection. Only used by the Aria storage engine.

- Command line: `--deadlock-timeout-long=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `50000000`
- Range: `0` to `4294967295`

`deadlock_timeout_short`

- Description: Short timeout in microseconds for the two-step deadlock detection. Only used by the Aria storage engine.
- Command line: `--deadlock-timeout-short=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10000`
- Range: `0` to `4294967295`

This page is licensed: CC BY-SA / Gnu FDL

Aria Two-step Deadlock Detection

Description

The Aria storage engine can automatically detect and deal with deadlocks (see the Wikipedia deadlocks article).

This feature is controlled by four configuration variables, two that control the search depth and two that control the timeout.

- deadlock_search_depth_long
- deadlock_search_depth_short
- deadlock_timeout_long
- deadlock_timeout_short

How it Works

If Aria is ever unable to obtain a lock, we might have a deadlock. There are two primary ways for detecting if a deadlock has actually occurred. First is to search a wait-for graph (see the wait-for graph on Wikipedia) and the second is to just wait and let the deadlock exhibit itself. Aria Two-step Deadlock Detection does a combination of both.

First, if the lock request cannot be granted immediately, we do a short search of the wait-for graph with a small search depth as configured by the `deadlock_search_depth_short` variable. We have a depth limit because the graph can (theoretically) be arbitrarily big and we don't want to recursively search the graph arbitrarily deep. This initial, short search is very fast and most deadlocks are detected right away. If no deadlock cycles are found with the short search the system waits for the amount of time configured in `deadlock_timeout_short` to see if the lock conflicts are removed and the lock can be granted. Assuming this did not happen and the lock request still waits, the system then moves on to step two, which is a repeat of the process but this time searching deeper using the `deadlock_search_depth_long`. If no deadlock has been detected, it waits `deadlock_timeout_long` and times out.

When a deadlock is detected the system uses a weighting algorithm to determine which thread in the deadlock should be killed and then kills it.

This page is licensed: CC BY-SA / Gnu FDL

Aria FAQ

This FAQ provides information on the Aria storage engine.

The ***Aria*** storage engine was previously known as ***Maria***, (see, the Aria Name). In current releases of MariaDB, you can refer to the engine as Maria or Aria. As this will change in future releases, please update references in your scripts and automation to use the correct name.

What is Aria?

Aria is a storage engine for MySQL® and MariaDB. It was originally developed with the goal of becoming the default transactional **and** non-transactional storage engine for MariaDB and MySQL.

It has been in development since 2007 and was first announced on Monty's blog. The same core MySQL engineers who developed the MySQL server and the MyISAM, MERGE, and MEMORY storage engines are also working on Aria.

Why is the engine called Aria?

Originally, the storage engine was called **Maria**, after Monty's younger daughter. Monty named MySQL after his first child, **My** and his second child **Max** gave his name to MaxDB and the MySQL-Max distributions.

In practice, having both *MariaDB* the database server and *Maria* the storage engine with such similar names proved confusing. To mitigate this, the decision was made to change the name. A Rename Maria contest was held during the first half of 2010 and names were submitted from around the world. Monty picked the name ***Aria*** from a short list of finalist. Chris Tooley, who suggested it, received the prize of a Linux-powered System 76 Meerkat NetTop from Monty Program.

For more information, see the Aria Name.

What's the goal for the current version?

The current version of Aria is 1.5. The goal of this release is to develop a crash-safe alternative to MyISAM. That is, when MariaDB restarts after a crash, Aria recovers all tables to the state as of the start of a statement or at the start of the last `LOCK TABLES` statement.

The current goal is to keep the code stable and fix all bugs.

What's the goal for the next version?

The next version of Aria is 2.0. The goal for this release is to develop a fully transactional storage engine with at least all the major features of InnoDB.

Currently, Aria 2.0 is on hold as its developers are focusing on improving MariaDB. However, they are interested in working with interested customers and partners to add more features to Aria and eventually release 2.0.

These are some of the goals for Aria 2.0:

- ACID compliant
- Commit/Rollback
- Concurrent updates/deletes
- Row locking
- Group commit (Already in MariaDB 5.2)
- Faster lookup in index pages (Page directory)

Beginning in Aria 2.5, the plan is to focus on improving performance.

What is the ultimate goal of Aria?

Long term, we have the following goals for Aria:

- To create a new, ACID and Multi-Version Concurrency Control (MVCC), transactional storage engine that can function as both the default non-transactional and transactional storage engine for MariaDB and MySQL®.
- To be a MyISAM replacement. This is possible because Aria can also be run in non-transactional mode, supports the same row formats as MyISAM, and supports or will support all major features of MyISAM.
- To be the default non-transactional engine in MariaDB (instead of MyISAM).

What are the design goals in Aria?

- Multi-Version Concurrency Control (MVCC) and ACID storage engine.
- Optionally non-transactional tables that should be 'as fast and as compact' as MyISAM tables.
- Be able to use Aria for internal temporary tables in MariaDB (instead of MyISAM).
- All indexes should have equal speed (clustered index is not on our current road map for Aria. If you need clustered index, you should use XtraDB).
- Allow 'any' length transactions to work (Having long running transactions will cause more log space to be used).
- Allow log shipping; that is, you can do incremental backups of Aria tables just by copying the Aria logs.
- Allow copying of Aria tables between different Aria servers (under some well-defined constraints).
- Better blob handling (than is currently offered in MyISAM, at a minimum).
- No memory copying or extra memory used for blobs on insert/update.
- Blobs allocated in big sequential blocks - Less fragmentation over time.
- Blobs are stored so that Aria can easily be extended to have access to any part of a blob with a single fetch in the future.
- Efficient storage on disk (that is, low row data overhead, low page data overhead and little lost space on pages). Note: There is still some more work to succeed with this goal. The disk layout is fine, but we need more in-memory caches to ensure that we get a higher fill factor on the pages.
- Small footprint, to make MariaDB + Aria suitable for desktop and embedded applications.
- Flexible memory allocation and scalable algorithms to utilize large amounts of memory efficiently, when it is available.

Where can I find documentation and help about Aria?

Documentation is available at Aria and related topics. The project is maintained on GitHub.

If you want to know what happens or be part of developing Aria, you can subscribe to the developers, docs, or discuss mailing lists.

To report and check bugs in Aria, see Reporting Bugs.

You can usually find some of the Maria developers on our Zulip instance at mariadb.zulipchat.com or on the IRC channel #maria at.

Who develops Aria?

The Core Team who develop Aria are:

Technical lead

- Michael "Monty" Widenius - Creator of MySQL and MyISAM

Core Developers (in alphabetical order)

- Guilhem Bichot - Replication expert, on line backup for MyISAM, etc.
- Kristian Nielsen - MySQL build tools, NDB, MySQL server

- Oleksandr Byelkin - Query cache, sub-queries, views.
- Sergei Golubchik - Server Architect, Full text search, keys for MyISAM-Merge, Plugin architecture, etc.

All except Guilhem Bichot are working for MariaDB Corporation Ab.

What is the release policy/schedule of Aria?

Aria follows the same release criteria as for MariaDB. Some clarifications, unique for the Aria storage engine:

- Aria index and data file formats should be backwards and forwards compatible to ensure easy upgrades and downgrades.
- The log file format should also be compatible, but we don't make any guarantees yet. In some cases when upgrading, you must remove the old `aria_log.%` and `maria_log.%` files before restarting MariaDB. (So far, this has only occurred in the upgrade from MariaDB 5.1 and MariaDB 5.2).

Extended commitment for Beta 1.5

- Aria is now feature complete according to specification.

How does Aria 1.5 Compare to MyISAM?

Aria 1.0 was basically a crash-safe non-transactional version of MyISAM. Aria 1.5 added more concurrency (multiple inserter) and some optimizations.

Aria supports all aspects of MyISAM, except as noted below. This includes external and internal check/repair/compressing of rows, different row formats, different index compress formats, aria_chk etc. After a normal shutdown you can copy Aria files between servers.

Advantages of Aria compared to MyISAM

- Data and indexes are crash safe.
- On a crash, changes are rolled back to state of the start of a statement or a last `LOCK TABLES` statement.
- Aria can replay almost everything from the log. (Including `CREATE`, `DROP`, `RENAME`, `TRUNCATE` tables). Therefore, you make a backup of Aria by just copying the log. The things that can't be replayed (yet) are:
  - Batch `INSERT` into an empty table (This includes `LOAD DATA INFILE`, `SELECT... INSERT` and `INSERT` (many rows)).
  - `ALTER TABLE`. Note that `.frm` tables are NOT recreated!
- `LOAD INDEX` can skip index blocks for unwanted indexes.
- Supports all MyISAM `ROW` formats and new `PAGE` format where data is stored in pages. (default size is 8K).
- Multiple concurrent inserters into the same table.
- When using `PAGE` format (default) row data is cached by page cache.
- Aria has unit tests of most parts.
- Supports both crash-safe (soon to be transactional) and not transactional tables. (Non-transactional tables are not logged and rows uses less space): `CREATE TABLE foo (...) TRANSACTIONAL=0|1 ENGINE=Aria`.
- `PAGE` is the only crash-safe/transactional row format.
- `PAGE` format should give a notable speed improvement on systems which have bad data caching. (For example Windows).
- From MariaDB 10.5, max key length is 2000 bytes, compared to 1000 bytes in MyISAM.

Differences between Aria and MyISAM

- Aria uses BIG (1G by default) log files.

- Aria has a log control file (`aria_log_control`) and log files (`aria_log.%`). The log files can be automatically purged when not needed or purged on demand (after backup).
- Aria uses 8K pages by default (MyISAM uses 1K). This makes Aria a bit faster when using keys of fixed size, but slower when using variable-length packed keys (until we add a directory to index pages).

Disadvantages of Aria compared to MyISAM

- Aria doesn't support `INSERT DELAYED`.
- Aria does not support multiple key caches.
- Storage of very small rows (< 25 bytes) are not efficient for `PAGE` format.
- `MERGE` tables don't support Aria (should be very easy to add later).
- Aria data pages in block format have an overhead of 10 bytes/page and 5 bytes/row. Transaction and multiple concurrent-writer support will use an extra overhead of 7 bytes for new rows, 14 bytes for deleted rows and 0 bytes for old compacted rows.
- No external locking (MyISAM has external locking, but this is a rarely used feature).
- Aria has one page size for both index and data (defined when Aria is used the first time). MyISAM supports different page sizes per index.
- Small overhead (15 bytes) per index page.
- Aria doesn't support MySQL internal RAID (disabled in MyISAM too, it's a deprecated feature).
- Minimum data file size for PAGE format is 16K (with 8K pages).
- Aria doesn't support indexes on virtual fields.

Differences between MariaDB 5.1 release and the normal MySQL-5.1 release?

See:

- Aria storage engine
- MariaDB versus MySQL

Why do you use the `TRANSACTIONAL` keyword now when Aria is not yet transactional?

In the current development phase Aria tables created with `TRANSACTIONAL=1` are crash safe and atomic but not transactional because changes in Aria tables can't be rolled back with the `ROLLBACK` command. As we planned to make Aria tables fully transactional, we decided it was better to use the `TRANSACTIONAL` keyword from the start so that applications don't need to be changed later.

What are the known problems with the MySQL-5.1-Maria release?

- See `KNOWN_BUGS.txt` for open/design bugs.
- See jira.mariadb.org for newly reported bugs. Please report anything you can't find here!
- If there is a bug in the Aria recovery code or in the code that generates the logs, or if the logs become corrupted, then mysqld may fail to start because Aria can't execute the logs at start up.
- Query cache and concurrent insert using page row format have a bug, please disable query cache while using page row format and MDEV-6817 isn't complete

If Aria doesn't start or you have an unrecoverable table (shouldn't happen):

- Remove the `aria_log.%` files from the data directory.
- Restart `mysqld` and run CHECK TABLE, REPAIR TABLE or mariadb-check on your Aria tables.

Alternatively,

- Remove logs and run aria_chk on your `*.MAI` files.

What is going to change in later Aria main releases?

The `LOCK TABLES` statement will not start a crash-safe segment. You should use begin and COMMIT instead.

To make things future safe, you could do this:

```
BEGIN;
LOCK TABLES ....
UNLOCK TABLES;
COMMIT;
```

And later you can just remove the `LOCK TABLES` and `UNLOCK TABLES` statements.

How can I create a MyISAM-like (non-transactional) table in Aria?

Example:

```
CREATE TABLE t1 (a INT) ROW_FORMAT=FIXED TRANSACTIONAL=0 PAGE_CHECKSUM=0;
CREATE TABLE t2 (a INT) ROW_FORMAT=DYNAMIC TRANSACTIONAL=0 PAGE_CHECKSUM=0;
SHOW CREATE TABLE t1;
SHOW CREATE TABLE t2;
```

Note that the rows are not cached in the page cache for `FIXED` or `DYNAMIC` format. If you want to have the data cached (something MyISAM doesn't support) you should use `ROW_FORMAT=PAGE`:

```
CREATE TABLE t3 (a INT) ROW_FORMAT=PAGE TRANSACTIONAL=0 PAGE_CHECKSUM=0;
SHOW CREATE TABLE t3;
```

You can use `PAGE_CHECKSUM=1` also for non-transactional tables; This puts a page checksums on all index pages. It also puts a checksum on data pages if you use `ROW_FORMAT=PAGE`.

You may still have a speed difference (may be slightly positive or negative) between MyISAM and Aria because of different page sizes. You can change the page size for MariaDB with `--aria-block-size=\#`, where `\`

is 1024, 2048, 4096, 8192, 16384 or 32768.

Note that if you change the page size you have to dump all your old tables into text (with mariadb-dump) and remove the old Aria log and files:

```
# rm datadir/aria_log*
```

What are the advantages/disadvantages of the new `PAGE` format compared to the old MyISAM-like row formats (`DYNAMIC` and `FIXED`)

The MyISAM-like `DYNAMIC` and `FIXED` format are extremely simple and have very little space overhead, so it's hard to beat them for when it comes to simple scanning of unmodified data. The `DYNAMIC` format does however get notably worse over time if you update the row a lot in a manner that increases the size of the row.

The advantages of the `PAGE` format (compared to `DYNAMIC` or `FIXED`) for non-transactional tables are:

- It's cached by the Page Cache, which gives better random performance (as it uses less system calls).
- Does not fragment as easily as the `DYNAMIC` format during `UPDATE` statements. The maximum number of fragments are very low.
- Code can easily be extended to only read the accessed columns (for example to skip reading blobs).
- Faster updates (compared to `DYNAMIC`).

The disadvantages are:

- Slight storage overhead (should only be notable for very small row sizes)

- Slower full table scan time.
- When using `row_format=PAGE`, (the default), Aria first writes the row, then the keys, at which point the check for duplicate keys happens. This makes `PAGE` format slower than `DYNAMIC` (or MyISAM) if there is a lot of duplicated keys because of the overhead of writing and removing the row. If this is a problem, you can use `row_format=DYNAMIC` to get same behavior as MyISAM.

What's the proper way to copy a Aria table from one place to another?

An Aria table consists of 3 files:

```
XXX.frm : The definition for the table, used by MySQL.
XXX.MYI : Aria internal information about the structure of the data and index and data for all indexes.
XXX.MAD : The data.
```

It's safe to copy all the Aria files to another directory or MariaDB instance if any of the following holds:

- If you shutdown the MariaDB Server properly with mariadb-admin shutdown, so that there is nothing for Aria to recover when it starts.

or

- If you have run a FLUSH TABLES statement and not accessed the table using SQL from that time until the tables have been copied.

In addition, you must adhere the following rule for transactional tables:

You can't copy the table to a location within the same MariaDB server if the new table has existed before and the new table is still active in the Aria recovery log (that is, Aria may need to access the old data during recovery). If you are unsure whether the old name existed, run aria_chk –zerofill on the table before you use it.

After copying a transactional table and before you use the table, we recommend that you run the command:

```
$ aria_chk --zerofill table_name
```

This will overwrite all references to the logs (LSN), all transactional references (TRN) and all unused space with 0. It also marks the table as 'movable'. An additional benefit of zerofill is that the Aria files will compress better. No real data is ever removed as part of zerofill.

Aria will automatically notice if you have copied a table from another system and do 'zerofill' for the first access of the table if it was not marked as 'movable'. The reason for using aria_chk –zerofill is that you avoid a delay in the MariaDB server for the first access of the table.

Note that this automatic detection doesn't work if you copy tables within the same MariaDB server!

When is it safe to remove old log files?

If you want to remove the Aria log files (`aria_log.%`) with `rm` or delete, then you must first shut down MariaDB cleanly (for example, with mariadb-admin shutdown) before deleting the old files.

The same rules apply when upgrading MariaDB; When upgrading, first take down MariaDB in a clean way and then upgrade. This will allow you to remove the old log files if there are incompatible problems between releases.

Don't remove the `aria_log_control` file! This is not a log file, but a file that contains information about the Aria setup (current transaction id, unique id, next log file number etc.).

If you do, Aria will generate a new `aria_log_control` file at startup and will regard all old Aria files as files moved from another system. This means that they have to be 'zerofilled' before they can be used. This will happen automatically at next access of the Aria files, which can take some time if the files are big.

If this happens, you will see things like this in your mysqld.err file:

```
[Note] Zerofilling moved table: '.\database\xxxx'
```

As part of zerofilling no vital data is removed.

How does one solve the Missing valid id error?

See Aria Log Files for details.

This page is licensed: CC BY-SA / Gnu FDL

The Aria Name

The Aria storage engine used to be called Maria. This page gives the history and background of how and why this name was changed to Aria.

Backstory

When starting what became the MariaDB project, Monty and the initial developers only planned to work on a next generation MyISAM storage engine replacement.
This storage engine would be crash safe and eventually support transactions.
Monty named the storage engine, and the project, after his daughter, Maria.

Work began in earnest on the Maria storage engine but the plans quickly expanded and morphed and soon the developers were not just working on a storage engine, but on a complete branch of the MySQL database. Since the project was already called Maria, it made sense to call the whole database server MariaDB.

Renaming Maria (the engine)

So now there was the database, MariaDB, and the storage engine, Maria. To end the confusion this caused, the decision was made to rename the storage engine.

Monty's first suggestion was to name it *Lucy*, after his dog, but few who heard it liked that idea. So the decision was made that the next best thing was for the community to suggest and vote on names.

This was done by running a contest in 2009 through the end of May 2010.
After that the best names were voted on by the community and Monty picked and announced the winner (Aria) at OSCon 2010 in Portland.

The winning entry was submitted by Chris Tooley. He received a Linux-powered System 76 Meerkat NetTop as his prize for suggesting the winning name from Monty Program.

See Also

- Why is the Software Called MariaDB?

This page is licensed: CC BY-SA / Gnu FDL

BLACKHOLE

The `BLACKHOLE` storage engine accepts data but does not store it and always returns an empty result.

A table using the `BLACKHOLE` storage engine consists of a single .frm table format file, but no associated data or index files.

This storage engine can be useful, for example, if you want to run complex filtering rules on a slave without incurring any overhead on a master. The master can run a `BLACKHOLE` storage engine, with the data replicated to the slave for processing.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'ha_blackhole';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = ha_blackhole
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'ha_blackhole';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Using the BLACKHOLE Storage Engine

Using with DML

INSERT, UPDATE, and DELETE statements all work with the `BLACKHOLE` storage engine. However, no data changes are actually applied.

Using with Replication

If the binary log is enabled, all SQL statements are logged as usual, and replicated to any slave servers. However, since rows are not stored, it is important to use statement-based rather than the row or mixed format, as UPDATE and DELETE statements are neither logged nor replicated. See Binary Log Formats.

Using with Triggers

Some triggers work with the `BLACKHOLE` storage engine.

`BEFORE` triggers for INSERT statements are still activated.

Triggers for UPDATE and DELETE statements are **not** activated.

Triggers with the `FOR EACH ROW` clause do not apply, since the tables have no rows.

Using with Foreign Keys

Foreign keys are not supported. If you convert an InnoDB table to `BLACKHOLE`, then the foreign keys will disappear. If you convert the same table back to InnoDB, then you will have to recreate them.

Using with Virtual Columns

If you convert an InnoDB table which contains virtual columns to `BLACKHOLE`, then it produces an error.

Using with AUTO_INCREMENT

Because a BLACKHOLE table does not store data, it will not maintain the AUTO_INCREMENT value. If you are replicating to a table that can handle `AUTO_INCREMENT` columns, and are not explicitly setting the primary key auto-increment value in the INSERT query, or using the SET INSERT_ID statement, inserts will fail on the slave due to duplicate keys.

Limits

The maximum key size is:

- 3500 bytes (>= MariaDB 10.1.48, MariaDB 10.2.35, MariaDB 10.3.26, MariaDB 10.4.16 and MariaDB 10.5.7)
- 1000 bytes (<= MariaDB 10.1.47, MariaDB 10.2.34, MariaDB 10.3.25, MariaDB 10.4.15 and MariaDB 10.5.6).

Examples

```
CREATE TABLE table_name (
   id INT UNSIGNED PRIMARY KEY NOT NULL,
   v VARCHAR(30)
) ENGINE=BLACKHOLE;

INSERT INTO table_name VALUES (1, 'bob'),(2, 'jane');

SELECT * FROM table_name;
Empty set (0.001 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

CONNECT

> **Note:**
>
> CONNECT is a community contribution to MariaDB Server, developed and maintained by its contributor, Olivier Bertrand.

You can download a PDF version of the CONNECT documentation (1.7.0003):

File: `../../../.gitbook/assets/connect_1_7_03.pdf`

Connect Version	Introduced	Maturity
Connect 1.07.0002	MariaDB 10.5.9, MariaDB 10.4.18, MariaDB 10.3.28, MariaDB 10.2.36	Stable
Connect 1.07.0001	MariaDB 10.4.12, MariaDB 10.3.22, MariaDB 10.2.31, MariaDB 10.1.44	Stable
Connect 1.06.0010	MariaDB 10.4.8, MariaDB 10.3.18, MariaDB 10.2.27	Stable
Connect 1.06.0007	MariaDB 10.3.6, MariaDB 10.2.14, MariaDB 10.1.33	Stable
Connect 1.06.0005	MariaDB 10.3.3, MariaDB 10.2.10, MariaDB 10.1.29	Stable
Connect 1.06.0004	MariaDB 10.3.2, MariaDB 10.2.9, MariaDB 10.1.28	Stable
Connect 1.06.0001	MariaDB 10.3.1, MariaDB 10.2.8, MariaDB 10.1.24	Beta
Connect 1.05.0003	MariaDB 10.3.0, MariaDB 10.2.5, MariaDB 10.1.22	Stable
Connect 1.05.0001	MariaDB 10.2.4, MariaDB 10.1.21	Stable
Connect 1.04.0008	MariaDB 10.2.2, MariaDB 10.1.17	Stable
Connect 1.04.0006	MariaDB 10.2.0, MariaDB 10.1.13,	Stable
Connect 1.04.0005	MariaDB 10.1.10	Beta
Connect 1.04.0003	MariaDB 10.1.9	Beta

The CONNECT storage engine enables MariaDB to access external local or remote data (MED). This is done by defining tables based on different data types, in particular files in various formats, data extracted from other DBMS or products (such as Excel or MongoDB) via ODBC or JDBC, or data retrieved from the environment (for example DIR, WMI, and MAC tables)

This storage engine supports table partitioning, MariaDB virtual columns and permits defining *special* columns such as `ROWID`, `FILEID`, and `SERVID`.

No precise definition of maturity exists. Because CONNECT handles many table types, each type has a different maturity depending on whether it is old and well-tested, less well-tested or newly implemented. This is indicated for all data types.

See Introduction to the CONNECT Engine.

The CONNECT storage engine.

See Using CONNECT.

The CONNECT storage engine.

See Installing CONNECT.

The CONNECT storage engine.

See CONNECT Create Table Options.

The CONNECT storage engine.

See CONNECT Data Types.

The CONNECT storage engine.

See CONNECT Table Types.

The CONNECT storage engine.

See CONNECT Security.

The CONNECT storage engine.

See CONNECT System Variables.

The CONNECT storage engine has been deprecated.

See Adding the REST Feature as a Library Called by an OEM Table.

The CONNECT storage engine.

See Compiling JSON UDFs in a Separate Library.

The CONNECT storage engine.

See Making the GetRest Library.

The CONNECT storage engine.

See OEM Table Example.

The CONNECT storage engine.

See Current Status of the CONNECT Handler.

The CONNECT storage engine.

See JSON Sample Files.

The CONNECT storage engine.

This page is licensed: CC BY-SA / Gnu FDL

Introduction to the CONNECT Engine

CONNECT is not just a new “YASE” (Yet another Storage Engine) that provides another way to store data with additional features. It brings a new dimension to MariaDB, already one of the best products to deal with traditional database transactional applications, further into the world of business intelligence and data analysis, including NoSQL facilities. Indeed, BI is the set of techniques and tools for the transformation of raw data into meaningful and useful information. And where is this data?

> “It’s amazing in an age where relational databases reign supreme when it comes to managing data that so much information still exists outside RDBMS engines in the form of flat files and other such constructs. In most enterprises, data is passed back and forth between disparate systems in a fashion and speed that would rival the busiest expressways in the world, with much of this data existing in common, delimited files. Target systems intercept these source files and then typically proceed to load them via ETL (extract, transform, load) processes into databases that then utilize the information for business intelligence, transactional functions, or other standard operations. ETL tasks and data movement jobs can consume quite a bit of time and resources, especially if large volumes of data are present that require loading into a database.
> This being the case, many DBAs welcome alternative means of accessing and managing data that exists in file format.”

- Robin Schumacher[1]

What he describes is known as MED (Management of External Data) enabling the handling of data not stored in a DBMS database as if it were stored in tables. An ISO standard exists that describes one way to implement and use MED in SQL by defining foreign tables for which an external FDW (Foreign Data Wrapper) has been developed in C.

However, since this was written, a new source of data was developed as the “cloud”. Data are existing worldwide and, in particular, can be obtained in JSON or XML format in answer to REST queries. From Connect 1.06.0010, it is possible to create JSON, XML or CSV tables based on data retrieved from such REST queries.

MED as described above is a rather complex way to achieve this goal and MariaDB does not support the ISO SQL/MED standard. But, to cover the need, possibly in transactional but mostly in decision support applications, the CONNECT storage engine supports MED in a much simpler way.

The main features of CONNECT are:

1. No need for additional SQL language extensions.
2. Embedded wrappers for many external data types (files, data sources, virtual).
3. NoSQL query facilities for JSON, XML, HTML files and using JSON UDFs.
4. NoSQL data obtained from REST queries (requires cpprestsdk).
5. NoSQL new data type MONGO accessing MongoDB collections as relational tables.
6. Read/Write access to external files of most commonly used formats.
7. Direct access to most external data sources via ODBC, JDBC and MySQL or MongoDB API.
8. Only used columns are retrieved from external scan.
9. Push-down WHERE clauses when appropriate.
10. Support of special and virtual columns.
11. Parallel execution of multi-table tables (currently unavailable).
12. Supports partitioning by sub-files or by sub-tables (enabling table sharding).
13. Support of MRR for SELECT, UPDATE and DELETE.
14. Provides remote, block, dynamic and virtual indexing.
15. Can execute complex queries on remote servers.
16. Provides an API that allows writing additional FDW in C++.

With CONNECT, MariaDB has one of the most advanced implementations of MED and NoSQL, without the need for complex additions to the SQL syntax (foreign tables are “normal” tables using the CONNECT engine).

Giving MariaDB easy and natural access to external data enables the use of all of its powerful functions and SQL-handling abilities for developing business intelligence applications.

With version 1.07 of CONNECT, retrieving data from REST queries is available in all binary distributed version of MariaDB, and, from 1.07.002, CONNECT allows workspaces greater than 4GB.

1. ↑ Robin Schumacher is Vice President Products at DataStax and former Director of Product Management at MySQL. He has over 13 years of database experience in DB2, MySQL, Oracle, SQL Server and other database engines.

This page is licensed: GPLv2

Using CONNECT

See Using CONNECT - Condition Pushdown.

The CONNECT storage engine.

See Using CONNECT - Exporting Data From MariaDB.

The CONNECT storage engine.

See Using CONNECT - General Information.

The CONNECT storage engine.

See Using CONNECT - Importing File Data Into MariaDB Tables.

The CONNECT storage engine.

See Using CONNECT - Indexing.

The CONNECT storage engine.

See Using CONNECT - Partitioning and Sharding.

The CONNECT storage engined.

See Using CONNECT - Virtual and Special Columns.

The CONNECT storage engine.

This page is licensed: CC BY-SA / Gnu FDL

Using CONNECT - Condition Pushdown

The ODBC, JDBC, MYSQL, TBL and WMI table types use engine condition pushdown in order to restrict the number of rows returned by the RDBS source or the WMI component.

The CONDITION_PUSHDOWN argument used in old versions of CONNECT is no longer needed because CONNECT uses condition pushdown unconditionally.

This page is licensed: GPLv2

Using CONNECT - Exporting Data From MariaDB

Exporting data from MariaDB is obviously possible with CONNECT in particular for all formats not supported by the SELECT INTO OUTFILE statement. Let us consider the query:

```
SELECT
    plugin_name AS handler,
    plugin_version AS version,
    plugin_author AS author,
    plugin_description AS description,
    plugin_maturity AS maturity
FROM
    information_schema.plugins
WHERE
    plugin_type = 'STORAGE ENGINE';
```

Supposing you want to get the result of this query into a file handlers.htm in XML/HTML format, allowing displaying it on an Internet browser, this is how you can do it:

Just create the CONNECT table that are used to make the file:

```
CREATE TABLE handout
ENGINE=CONNECT
table_type=XML
file_name='handout.htm'
header=yes
option_list='name=TABLE,coltype=HTML,attribute=border=1;cellpadding=5,headattr=bgcolor=yellow'
AS
SELECT
    plugin_name AS handler,
    plugin_version AS version,
    plugin_author AS author,
    plugin_description AS description,
    plugin_maturity AS maturity
FROM
    information_schema.plugins
WHERE
    plugin_type = 'STORAGE ENGINE';
```

Here the column definition is not given and will come from the Select statement following the Create. The CONNECT options are the same we have seen previously. This will do both actions, creating the matching *handlers* CONNECT table and 'filling' it with the query result.

Note 1: This could not be done in only one statement if the table type had required using explicit CONNECT column options. In this case, firstly create the table, and then populate it with an Insert statement.

Note 2: The source "plugins" table column "description" is a long text column, data type not supported for CONNECT tables. It has been silently internally replaced by varchar(256).

This page is licensed: GPLv2

Using CONNECT - General Information

The main characteristic of CONNECT is to enable accessing data scattered on a machine as if it was a centralized database. This, and the fact that locking is not used by connect (data files are open and closed for each query) makes CONNECT very useful for importing or exporting data into or from a MariaDB database and also for all types of Business Intelligence applications. However, it is not suited for transactional applications.

For instance, the index type used by CONNECT is closer to bitmap indexing than to B-trees. It is very fast for retrieving result but not when updating is done. In fact, even if only one indexed value is modified in a big table, the index is entirely remade (yet this being four to five times faster than for a b-tree index). But normally in Business Intelligence applications, files are not modified so often.

If you are using CONNECT to analyze files that can be modified by an external process, the indexes are of course not modified by it and become outdated. Use the OPTIMIZE TABLE command to update them before using the tables based on them.

This means also that CONNECT is not designed to be used by centralized servers, which are mostly used for transactions and often must run a long time without human intervening.

Performance

Performances vary a great deal depending on the table type. For instance, ODBC tables are only retrieved as fast as the other DBMS can do. If you have a lot of queries to execute, the best way to optimize your work can be sometime to translate the data from one type to another. Fortunately this is very simple with CONNECT. Fixed formats like FIX, BIN or VEC tables can be created from slower ones by commands such as:

```
CREATE TABLE fastable table_specs SELECT * FROM slowtable;
```

`FIX` and `BIN` are often the better choice because the I/O functions are done on blocks of `BLOCK_SIZE` rows. `VEC` tables can be very efficient for tables having many columns only a few being used in each query. Furthermore, for tables of reasonable size, the `MAPPED` option can very often speed up many queries.

Create Table statement

Be aware of the two broad kinds of CONNECT tables:

Inward	They are table whose file name is not specified at create. An empty file are given a default name (tabname.tabtype) and are populated like for other engines. They do not require the FILE privilege and can be used for testing purpose.
Outward	They are all other CONNECT tables and access external data sources or files. They are the true useful tables but require the FILE privilege.

Drop Table statement

For outward tables, the DROP TABLE statement just removes the table definition but does not erase the table data. However, dropping an inward tables also erase the table data as well.

Alter Table statement

Be careful using the ALTER TABLE statement. Currently the data compatibility is not tested and the modified definition can become incompatible with the data. In particular, Alter modifies the table definition only but does not modify the table data. Consequently, the table type should not be modified this way, except to correct an incorrect definition. Also adding, dropping or modifying columns may be wrong because the default offset values (when not explicitly given by the FLAG option) may be wrong when recompiled with missing columns.

Safe use of ALTER is for indexing, as we have seen earlier, and to change options such as MAPPED or HUGE those do not impact the data format but just the way the data file is accessed. Modifying the BLOCK_SIZE option is all right with FIX, BIN, DBF, split VEC tables; however it is unsafe for VEC tables that are not split (only one data file) because at their creation the estimate size has been made a multiple of the block size. This can cause errors if this estimate is not a multiple of the new value of the block size.

In all cases, it is safer to drop and re-create the table (outward tables) or to make another one from the table that must be modified.

Update and Delete for File Tables

CONNECT can execute these commands using two different algorithms:

- It can do it in place, directly modifying rows (update) or moving rows (delete) within the table file. This is a fast way to do it in particular when indexing is used.

- It can do it using a temporary file to make the changes. This is required when updating variable record length tables and is more secure in all cases.

The choice between these algorithms depends on the session variable connect_use_tempfile.

This page is licensed: GPLv2

Using CONNECT - Importing File Data Into MariaDB Tables

Directly using external (file) data has many advantages, such as to work on “fresh” data produced for instance by cash registers, telephone switches, or scientific apparatus. However, you may want in some case to import external data into your MariaDB database. This is extremely simple and flexible using the CONNECT handler. For instance, let us suppose you want to import the data of the xsample.xml XML file previously given in example into a MyISAM table called *biblio* belonging to the connect database. All you have to do is to create it by:

```
CREATE TABLE biblio ENGINE=myisam SELECT * FROM xsampall2;
```

This last statement creates the MyISAM table and inserts the original XML data, translated to tabular format by the *xsampall2* CONNECT table, into the MariaDB *biblio* table. Note that further transformation on the data could have been achieved by using a more elaborate Select statement in the Create statement, for instance using filters, alias or applying functions to the data. However, because the Create Table process copies table data, later modifications of the *xsample.xml* file will not change the *biblio* table and changes to the *biblio* table will not modify the *xsample.xml* file.

All these can be combined or transformed by further SQL operations. This makes working with CONNECT much more flexible than just using the LOAD statement.

This page is licensed: GPLv2

Using CONNECT - Indexing

> **Warning:**
>
> This storage engine has been deprecated.

Indexing is one of the main ways to optimize queries. Key columns, in particular when they are used to join tables, should be indexed. But what should be done for columns that have only few distinct values? If they are randomly placed in the table they should not be indexed because reading many rows in random order can be slower than reading the entire table sequentially. However, if the values are sorted or clustered, indexing can be acceptable because CONNECT indexes store the values in the order they appear into the table and this will make retrieving them almost as fast as reading them sequentially.

CONNECT provides four indexing types:

1. Standard Indexing
2. Block Indexing
3. Remote Indexing
4. Dynamic Indexing

Standard Indexing

CONNECT standard indexes are created and used as the ones of other storage engines although they have a specific internal format. The CONNECT handler supports the use of standard indexes for most of the file based table types.

You can define them in the CREATE TABLE statement, or either using the `CREATE INDEX` statement or the ALTER TABLE statement. In all cases, the index files are automatically made. They can be dropped either using the DROP

INDEX statement or the ALTER TABLE statement, and this erases the index files.

Indexes are automatically reconstructed when the table is created, modified by `INSERT`, `UPDATE` or `DELETE` statements, or when the `SEPINDEX` option is changed. If you have a lot of changes to do on a table at one moment, you can use table locking to prevent indexes to be reconstructed after each statement. The indexes are reconstructed when unlocking the table. For instance:

```
LOCK TABLE t1 WRITE;
INSERT INTO t1 VALUES(...);
INSERT INTO t1 VALUES(...);
...
UNLOCK TABLES;
```

If a table was modified by an external application that does not handle indexing, the indexes must be reconstructed to prevent returning false or incomplete results. To do this, use the OPTIMIZE TABLE command.

For outward tables, index files are not erased when dropping the table. This is the same as for the data file and preserves the possibility of several users using the same data file via different tables.

Unlike other storage engines, CONNECT constructs the indexes as files that are named by default from the data file name, not from the table name, and located in the data file directory. Depending on the `SEPINDEX` table option, indexes are saved in a unique file or in separate files (if `SEPINDEX` is true). For instance, if indexes are in separate files, the primary index of the `table_dept.dat_` of type `DOS` is a file named *`dept_PRIMARY.dnx`*. This makes possible to define several tables on the same data file, with eventual different options such as mapped or not mapped, and to share the index files as well.

If the index file should have a different name, for instance because several tables are created on the same data file with different indexes, specify the base index file name with the `XFILE_NAME` option.

Note1: Indexed columns must be declared `NOT NULL`; CONNECT doesn't support indexes containing null values.

Note 2: MRR is used by standard indexing if it is enabled.

Note 3: Prefix indexing is not supported. If specified, the CONNECT engine ignores the prefix and builds a whole index.

Handling index errors

The way CONNECT handles indexing is very specific. All table modifications are done regardless of indexing. Only after a table has been modified, or when an `OPTIMIZE TABLE` command is sent are the indexes made. If an error occurs, the corresponding index is not made. However, CONNECT being a non-transactional engine, it is unable to roll back the changes made to the table. The main causes of indexing errors are:

- Trying to index a nullable column. In this case, you can alter the table to declare the column as not nullable or, if the column is nullable indeed, make it not indexed.
- Entering duplicate values in a column indexed by a unique index. In this case, if the index was wrongly declared as unique, alter is declaration to reflect this. If the column should really contain unique values, you must manually remove or update the duplicate values.

In both cases, after correcting the error, remake the indexes with the OPTIMIZE TABLE command.

Index file mapping

To accelerate the indexing process, CONNECT makes an index structure in memory from the index file. This can be done by reading the index file or using it as if it was in memory by "file mapping". On enabled versions, file mapping is used according to the boolean connect_indx_map system variable. Set it to 0 (file read) or 1 (file mapping).

Block Indexing

To accelerate input/output, CONNECT uses when possible a read/write mode by blocks of n rows, n being the value given in the BLOCK _ SIZE option of the Create Table, or a default value depending on the table type. This is automatic for fixed files (FIX, BIN, DBF or VEC), but must be specified for variable files (DOS , CSV or FMT).

For blocked tables, further optimization can be achieved if the data values for some columns are "clustered" meaning that they are not evenly scattered in the table but grouped in some consecutive rows. Block indexing permits to skip blocks in which no rows fulfill a conditional predicate without having even to read the block. This is true in particular for sorted columns.

You indicate this when creating the table by using the DISTRIB =d column option. The enum value d can be *scattered*, *clustered*, or *sorted*. In general only one column can be sorted. Block indexing is used only for clustered and sorted columns.

Difference between standard indexing and block indexing

- Block indexing is internally handled by CONNECT while reading sequentially a table data. This means in particular that when standard indexing is used on a table, block indexing is not used.
- In a query, only one standard index can be used. However, block indexing can combine the restrictions coming from a where clause implying several clustered/sorted columns.
- The block index files are faster to make and much smaller than standard index files.

Notes for this Release:

- On all operations that create or modify a table, CONNECT automatically calculates or recalculates and saves the mini/maxi or bitmap values for each block, enabling it to skip block containing no acceptable values. In the case where the optimize file does not correspond anymore to the table, because it has been accidentally destroyed, or because some column definitions have been altered, you can use the `OPTIMIZE TABLE` command to reconstruct the optimization file.
- Sorted column special processing is currently restricted to ascending sort. Column sorted in descending order must be flagged as clustered. Improper sorting is not checked in Update or Insert operations but is flagged when optimizing the table.
- Block indexing can be done in two ways. Keeping the min/max values existing for each block, or keeping a bitmap allowing knowing what column distinct values are met in each blocks. This second ways often gives a better optimization, except for sorted columns for which both are equivalent. The bitmap approach can be done only on columns having not too many distinct values. This is estimated by the `MAX_DIST` option value associated to the column when creating the table. Bitmap block indexing are used if this number is not greater than the `MAXBMP` setting for the database.
- CONNECT cannot perform block indexing on case insensitive character columns. To force block indexing on a character column, specify its charset as not case insensitive, for instance as binary.
  However this will also apply to all other clauses, this column being now case sensitive.

Remote Indexing

Remote indexing is specific to the MYSQL table type. It is equivalent to what the FEDERATED storage does. A MYSQL table does not support indexes per se. Because access to the table is handled remotely, it is the remote table that supports the indexes. What the MYSQL table does is just to add a WHERE clause to the SELECT command sent to the remote server allowing the remote server to use indexing when applicable.

Note, however, that because CONNECT adds when possible all or part of the where clause of the original query, this happens often even if the remote indexed column is not declared locally indexed.
The only, but very important, case a column should be locally declared indexed is when it is used to join tables. Otherwise, the required where clause would not be added to the sent `SELECT` query.

See Indexing of MYSQL tables for more.

Dynamic Indexing

An indexed created as “dynamic” is a standard index which, in some cases, can be reconstructed for a specific query. This happens in particular for some queries where two tables are joined by an indexed key column. If the “from” table is big and the “to” big table reduced in size because of a where clause, it can be worthwhile to reconstruct the index on this reduced table.

Because of the time added by reconstructing the index, this is valuable only if the time gained by reducing the index size if more than this reconstruction time. This is why this should not be done if the “from” table is small because there will not be enough row joining to compensate for the additional time.
Otherwise, the gain of using a dynamic index is:

- Indexing time is a little faster if the index is smaller.
- The join process will return only the rows fulfilling the where clause.
- Because the table is read sequentially when reconstructing the index there no need for MRR.
- Constructing the index can be faster if the table is reduced by block indexing.
- While constructing the index, CONNECT also stores in memory the values of other used columns.

This last point is particularly important. It means that after the index is reconstructed, the join is done on a temporary memory table.

Unfortunately, storage engines being called independently by MariaDB for each table, CONNECT has no global information to decide when it is good to use dynamic indexing. This is why you should use it only on cases where you see that some important join queries take a very long time and only on columns used for joining the table. How to declare an index to be dynamic is by using the Boolean
DYNAM index option. For instance, the query:

```
SELECT d.diag, COUNT(*) cnt FROM diag d, patients p WHERE d.pnb =
p.pnb AND ageyears < 17 AND county = 30 AND drg <> 11 AND d.diag
BETWEEN 4296 AND 9434 GROUP BY d.diag ORDER BY cnt DESC;
```

Such a query joining the diag table to the patients table may last a very long time if the tables are big.
To declare the primary key on the pnb column of the patients table to be dynamic:

```
ALTER TABLE patients DROP PRIMARY KEY;
ALTER TABLE patients ADD PRIMARY KEY (pnb) COMMENT 'DYNAMIC' dynam=1;
```

Note 1: The comment is not mandatory here but useful to see that the index is dynamic if you use the SHOW INDEX command.

Note 2: There is currently no way to just change the DYNAM option without dropping and adding the index. This is unfortunate because it takes time.

Virtual Indexing

It applies only to the virtual tables of type VIR and must be made on a column specifying `SPECIAL=ROWID` or `SPECIAL=ROWNUM`.

This page is licensed: GPLv2

Using CONNECT - Partitioning and Sharding

CONNECT supports the MySQL/MariaDB partition specification. It is done similar to the way MyISAM or InnoDB do by using the PARTITION engine that must be enabled for this to work. This type of partitioning is sometimes referred as “horizontal partitioning”.

Partitioning enables you to distribute portions of individual tables across a file system according to rules which you can set largely as needed. In effect, different portions of a table are stored as separate tables in different locations. The user-selected rule by which the division of data is accomplished is known as a partitioning function, which in MariaDB can be the modulus, simple matching against a set of ranges or value lists, an internal hashing function, or a linear hashing function.

CONNECT takes this notion a step further, by providing two types of partitioning:

1. File partitioning. Each partition is stored in a separate file like in multiple tables.
2. Table partitioning. Each partition is stored in a separate table like in TBL tables.

Partition engine issues

Using partitions sometimes requires creating the tables in an unnatural way to avoid some error due to several partition engine bugs:

1. Engine specific column and index options are not recognized and cause a syntax error when the table is created. The workaround is to create the table in two steps, a `CREATE TABLE` statement followed by an `ALTER TABLE` statement.
2. The connection string, when specified for the table, is lost by the partition engine. The workaround is to specify the connection string in the option_list.
3. MySQL upstream bug #71095. In case of list columns partitioning it sometimes causes a false “impossible where” clause to be raised. This makes a wrong void result returned when it should not be void. There is no workaround but this bug should be hopefully fixed.

The following examples are using the above workaround syntax to address these issues.

File Partitioning

File partitioning applies to file-based CONNECT table types. As with multiple tables, physical data is stored in several files instead of just one. The differences to multiple tables are:

1. Data is distributed amongst the different files following the partition rule.
2. Unlike multiple tables, partitioned tables are not read only.
3. Unlike multiple tables, partitioned tables can be indexable.
4. The file names are generated from the partition names.
5. Query pruning is automatically made by the partition engine.

The table file names are generated differently depending on whether the table is an inward or outward table. For inward tables, for which the file name is not specified, the partition file names are:

```
Data file name: table_name#P#partition_name.table_file_type
Index file name: table_name#P#partition_name.index_file_type
```

For instance for the table:

```
CREATE TABLE t1 (
id INT KEY NOT NULL,
msg VARCHAR(32))
ENGINE=CONNECT TABLE_TYPE=FIX
PARTITION BY RANGE(id) (
PARTITION first VALUES LESS THAN(10),
PARTITION middle VALUES LESS THAN(50),
PARTITION last VALUES LESS THAN(MAXVALUE));
```

CONNECT will generate in the current data directory the files:

```
| t1#P#first.fix
| t1#P#first.fnx
| t1#P#middle.fix
| t1#P#middle.fnx
| t1#P#last.fix
| t1#P#last.fnx
```

This is similar to what the partition engine does for other engines - CONNECT partitioned inward tables behave like other engines partition tables do. Just the data format is different.

Note: If sub-partitioning is used, inward table files and index files are named:

```
| table_name#P#partition_name#SP#subpartition_name.type
| table_name#P#partition_name#SP#subpartition_name.index_type
```

Outward Tables

The real problems occur with outward tables, in particular when they are created from already existing files. The first issue is to make the partition table use the correct existing file names. The second one, only for already existing not void tables, is to be sure the partitioning function match the distribution of the data already existing in the files.

The first issue is addressed by the way data file names are constructed. For instance let us suppose we want to make a table from the fixed formatted files:

```
E:\Data\part1.txt
E:\Data\part2.txt
E:\Data\part3.txt
```

This can be done by creating a table such as:

```
CREATE TABLE t2 (
id INT NOT NULL,
msg VARCHAR(32),
INDEX XID(id))
ENGINE=connect table_type=FIX file_name='E:/Data/part%s.txt'
PARTITION BY RANGE(id) (
PARTITION `1` VALUES LESS THAN(10),
PARTITION `2` VALUES LESS THAN(50),
PARTITION `3` VALUES LESS THAN(MAXVALUE));
```

The rule is that for each partition the matching file name is internally generated by replacing in the given FILE _ NAME option value the “%s” part by the partition name.

If the table was initially void, further inserts will populate it according to the partition function.
However, if the files did exist and contained data, this is your responsibility to determine what partition function actually matches the data distribution in them. This means in particular that partitioning by key or by hash cannot be used (except in exceptional cases) because you have almost no control over what the used algorithm does.

In the example above, there is no problem if the table is initially void, but if it is not, serious problems can be met if the initial distribution does not match the table distribution. Supposing a row in which “id” as the value 12 was initially contained in the part1.txt file, it are seen when selecting the whole table but if you ask:

```
SELECT * FROM t2 WHERE id = 12;
```

The result will have 0 rows. This is because according to the partition function query pruning will only look inside the second partition and will miss the row that is in the wrong partition.

One way to check for wrong distribution if for instance to compare the results from queries such as:

```
SELECT partition_name, table_rows FROM
information_schema.partitions WHERE table_name = 't2';
```

And

```
SELECT CASE WHEN id < 10 THEN 1 WHEN id < 50 THEN 2 ELSE 3 END
AS pn, COUNT(*) FROM part3 GROUP BY pn;
```

If they match, the distribution can be correct although this does not prove it. However, if they do not match, the distribution is surely wrong.

Partitioning on a Special Column

There are some cases where the files of a multiple table do not contain columns that can be used for range or list partitioning. For instance, let's suppose we have a multiple table based on the following files:

```
tmp/boston.txt
tmp/chicago.txt
tmp/atlanta.txt
```

Each of them containing the same kind of data:

```
ID: int
First_name: varchar(16)
Last_name: varchar(30)
Birth: date
Hired: date
Job: char(10)
Salary: double(8,2)
```

A multiple table can be created on them, for instance by:

```
CREATE TABLE mulemp (
id INT NOT NULL,
first_name VARCHAR(16) NOT NULL,
last_name VARCHAR(30) NOT NULL,
birth DATE NOT NULL date_format='DD/MM/YYYY',
hired DATE NOT NULL date_format='DD/MM/YYYY',
job CHAR(10) NOT NULL,
salary DOUBLE(8,2) NOT NULL
) ENGINE=CONNECT table_type=FIX file_name='tmp/*.txt' multiple=1;
```

The issue is that if we want to create a partitioned table on these files, there are no columns to use for defining a partition function. Each city file can have the same kind of column values and there is no way to distinguish them.

However, there is a solution. It is to add to the table a special column that are used by the partition function. For instance, the new table creation can be done by:

```
CREATE TABLE partemp (
id INT NOT NULL,
first_name VARCHAR(16) NOT NULL,
last_name VARCHAR(30) NOT NULL,
birth DATE NOT NULL date_format='DD/MM/YYYY',
hired DATE NOT NULL date_format='DD/MM/YYYY',
job CHAR(16) NOT NULL,
salary DOUBLE(10,2) NOT NULL,
city CHAR(12) DEFAULT 'boston' special=PARTID,
INDEX XID(id)
) ENGINE=CONNECT table_type=FIX file_name='E:/Data/Test/%s.txt';
ALTER TABLE partemp
PARTITION BY LIST COLUMNS(city) (
PARTITION `atlanta` VALUES IN('atlanta'),
PARTITION `boston` VALUES IN('boston'),
PARTITION `chicago` VALUES IN('chicago'));
```

Note 1: we had to do it in two steps because of the column CONNECT options.

Note 2: the special column PARTID returns the name of the partition in which the row is located.

Note 3: here we could have used the FNAME special column instead because the file name is specified as being the partition name.

This may seem rather stupid because it means for instance that a row are in partition boston if it belongs to the partition boston! However, it works because the partition engine doesn't know about special columns and behaves as if the city column was a real column.

What happens if we populate it by?

```
INSERT INTO partemp(id,first_name,last_name,birth,hired,job,salary) VALUES
(1205,'Harry','Cover','1982-10-07','2010-09-21','MANAGEMENT',125000.00);
INSERT INTO partemp VALUES
(1524,'Jim','Beams','1985-06-18','2012-07-25','SALES',52000.00,'chicago'),
(1431,'Johnny','Walker','1988-03-12','2012-08-09','RESEARCH',46521.87,'boston'),
(1864,'Jack','Daniels','1991-12-01','2013-02-16','DEVELOPMENT',63540.50,'atlanta');
```

The value given for the city column (explicitly or by default) are used by the partition engine to decide in which partition to insert the rows. It are ignored by CONNECT (a special column cannot be given a value) but later will return the matching value. For instance:

```
SELECT city, first_name, job FROM partemp WHERE id IN (1524,1431);
```

This query returns:

city	first_name	job
boston	Johnny	RESEARCH
chicago	Jim	SALES

Everything works as if the city column was a real column contained in the table data files.

Partitioning of Zipped Tables

Two cases are currently supported:
If a table is based on several zipped files, portioning is done the standard way as above. This is the *file_name* option specifying the name of the zip files that shall contain the '%s' part used to generate the file names.
If a table is based on only one zip file containing several entries, this is indicated by placing the '%s' part in the entry option value.
Note: If a table is based on several zipped files each containing several entries, only the first case is possible. Using sub-partitioning to make partitions on each entries is not supported yet.

Table Partitioning

With table partitioning, each partition is physically represented by a sub-table. Compared to standard partitioning, this brings the following features:

1. The partitions can be tables driven by different engines. This relieves the current existing limitation of the partition engine.
2. The partitions can be tables driven by engines not currently supporting partitioning.
3. Partition tables can be located on remote servers, enabling table sharding.
4. Like for TBL tables, the columns of the partition table do not necessarily match the columns of the sub-tables.

The way it is done is to create the partition table with a table type referring to other tables, PROXY,MYSQL ODBC or JDBC. Let us see how this is done on a simple example. Supposing we have created the following tables:

```
CREATE TABLE xt1 (
id INT NOT NULL,
msg VARCHAR(32))
ENGINE=myisam;

CREATE TABLE xt2 (
id INT NOT NULL,
msg VARCHAR(32)); /* engine=innoDB */

CREATE TABLE xt3 (
id INT NOT NULL,
msg VARCHAR(32))
ENGINE=connect table_type=CSV;
```

We can for instance create a partition table using these tables as physical partitions by:

```
CREATE TABLE t3 (
id INT NOT NULL,
msg VARCHAR(32))
ENGINE=connect table_type=PROXY tabname='xt%s'
PARTITION BY RANGE COLUMNS(id) (
PARTITION `1` VALUES LESS THAN(10),
PARTITION `2` VALUES LESS THAN(50),
PARTITION `3` VALUES LESS THAN(MAXVALUE));
```

Here the name of each partition sub-table are made by replacing the '%s' part of the tabname option value by the partition name. Now if we do:

```
INSERT INTO t3 VALUES
(4, 'four'),(7,'seven'),(10,'ten'),(40,'forty'),
(60,'sixty'),(81,'eighty one'),(72,'seventy two'),
(11,'eleven'),(1,'one'),(35,'thirty five'),(8,'eight');
```

The rows are distributed in the different sub-tables according to the partition function. This can be seen by executing the query:

```
SELECT partition_name, table_rows FROM
information_schema.partitions WHERE table_name = 't3';
```

This query replies:

partition_name	table_rows
1	4
2	4
3	3

Query pruning is of course automatic, for instance:

```
EXPLAIN PARTITIONS SELECT * FROM t3 WHERE id = 81;
```

This query replies:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	part5	3	ALL					22	Using where

When executing this select query, only sub-table xt3 are used.

Indexing with Table Partitioning

Using the PROXY table type seems natural. However, in this current version, the issue is that PROXY (and ODBC) tables are not indexable. This is why, if you want the table to be indexed, you must use the MYSQL table type. The CREATE TABLE statement are almost the same:

```
CREATE TABLE t4 (
id INT KEY NOT NULL,
msg VARCHAR(32))
ENGINE=connect table_type=MYSQL tabname='xt%s'
PARTITION BY RANGE COLUMNS(id) (
PARTITION `1` VALUES LESS THAN(10),
PARTITION `2` VALUES LESS THAN(50),
PARTITION `3` VALUES LESS THAN(MAXVALUE));
```

The column *id* is declared as a key, and the table type is now MYSQL. This makes Sub-tables accessed by calling a MariaDB server as MYSQL tables do. Note that this modifies only the way
CONNECT sub-tables are accessed.

However, indexing just make the partitioned table use “remote indexing” the way FEDERATED tables do. This means that when sending the query to retrieve the table data, a where clause are added to the query. For instance, let’s suppose you ask:

```
SELECT * FROM t4 WHERE id = 7;
```

The query sent to the server are:

```
SELECT `id`, `msg` FROM `xt1` WHERE `id` = 7
```

On a query like this one, it does not change much because the where clause could have been added anyway by the cond_push function, but it does make a difference in case of joins. The main thing to understand is that real indexing is done by the called table and therefore that it should be indexed.

This also means that the `xt1`, `xt2`, and `xt3` table indexes should be made separately because creating the `t2` table as indexed does not make the indexes on the sub-tables.

Sharding with Table Partitioning

Using table partitioning can have one more advantage. Because the sub-tables can address a table located on another server, it is possible to shard a table on separate servers and hardware machines.
This may be required to access as one table data already located on several remote machines, such as servers of a company branches. Or it can be just used to split a huge table for performance reason. For instance, supposing we have created the following tables:

```
CREATE TABLE rt1 (id INT KEY NOT NULL, msg VARCHAR(32))
ENGINE=federated connection='mysql://root@host1/test/sales';

CREATE TABLE rt2 (id INT KEY NOT NULL, msg VARCHAR(32))
ENGINE=federated connection='mysql://root@host2/test/sales';

CREATE TABLE rt3 (id INT KEY NOT NULL, msg VARCHAR(32))
ENGINE=federated connection='mysql://root@host3/test/sales';
```

Creating the partition table accessing all these are almost like what we did with the t4 table:

```
CREATE TABLE t5 (
id INT KEY NOT NULL,
msg VARCHAR(32))
ENGINE=connect table_type=MYSQL tabname='rt%s'
PARTITION BY RANGE COLUMNS(id) (
PARTITION `1` VALUES LESS THAN(10),
PARTITION `2` VALUES LESS THAN(50),
PARTITION `3` VALUES LESS THAN(MAXVALUE));
```

The only difference is the tabname option now referring to the rt1, rt2, and rt3 tables. However, even if it works, this is not the best way to do it. This is because accessing a table via the MySQL API is done twice per table. Once by CONNECT to access the FEDERATED table on the local server, then a second time by FEDERATED engine to access the remote table.

The CONNECT MYSQL table type being used anyway, you'd rather use it to directly access the remote tables. Indeed, the partition names can also be used to modify the connection URL's. For instance, in the case shown above, the partition table can be created as:

```
CREATE TABLE t6 (
id INT KEY NOT NULL,
msg VARCHAR(32))
ENGINE=connect table_type=MYSQL
option_list='connect=mysql://root@host%s/test/sales'
PARTITION BY RANGE COLUMNS(id) (
PARTITION `1` VALUES LESS THAN(10),
PARTITION `2` VALUES LESS THAN(50),
PARTITION `3` VALUES LESS THAN(MAXVALUE));
```

Several things can be noted here:

1. As we have seen before, the partition engine currently loses the connection string. This is why it was specified as "connect" in the option list.
2. For each partition sub-tables, the "%s" part of the connection string has been replaced by the partition name.
3. It is not needed anymore to define the rt1, rt2, and rt3 tables (even it does not harm) and the FEDERATED engine is no more used to access the remote tables.

This is a simple case where the connection string is almost the same for all the sub-tables. But what if the sub-tables are accessed by very different connection strings? For instance:

```
For rt1: connection='mysql://root:tinono@127.0.0.1:3307/test/xt1'
For rt2: connection='mysql://foo:foopass@denver/dbemp/xt2'
For rt3: connection='mysql://root@huston :5505/test/tabx'
```

There are two solutions. The first one is to use the parts of the connection string to differentiate as partition names:

```
CREATE TABLE t7 (
id INT KEY NOT NULL,
msg VARCHAR(32))
ENGINE=connect table_type=MYSQL
option_list='connect=mysql://%s'
PARTITION BY RANGE COLUMNS(id) (
PARTITION `root:tinono@127.0.0.1:3307/test/xt1` VALUES LESS THAN(10),
PARTITION `foo:foopass@denver/dbemp/xt2` VALUES LESS THAN(50),
PARTITION `root@huston :5505/test/tabx` VALUES LESS THAN(MAXVALUE));
```

The second one, allowing avoiding too complicated partition names, is to create federated servers to access the remote tables (if they do not already exist, else just use them). For instance the first one could be:

```
CREATE SERVER `server_one` FOREIGN DATA WRAPPER 'mysql'
OPTIONS
(HOST '127.0.0.1',
DATABASE 'test',
USER 'root',
PASSWORD 'tinono',
PORT 3307);
```

Similarly, “server_two” and “server_three” would be created and the final partition table would be created as:

```
CREATE TABLE t8 (
id INT KEY NOT NULL,
msg VARCHAR(32))
ENGINE=connect table_type=MYSQL
option_list='connect=server_%s'
PARTITION BY RANGE COLUMNS(id) (
PARTITION `one/xt1` VALUES LESS THAN(10),
PARTITION `two/xt2` VALUES LESS THAN(50),
PARTITION `three/tabx` VALUES LESS THAN(MAXVALUE));
```

It would be even simpler if all remote tables had the same name on the remote databases, for instance if they all were named xt1, the connection string could be set as “server_%s/xt1” and the partition names would be just “one”, “two”, and “three”.

Sharding on a Special Column

The technique we have seen above with file partitioning is also available with table partitioning.
Companies willing to use as one table data sharded on the company branch servers can, as we have seen, add to the table create definition a special column. For instance:

```
CREATE TABLE t9 (
id INT NOT NULL,
msg VARCHAR(32),
branch CHAR(16) DEFAULT 'main' special=PARTID,
INDEX XID(id))
ENGINE=connect table_type=MYSQL
option_list='connect=server_%s/sales'
PARTITION BY RANGE COLUMNS(id) (
PARTITION `main` VALUES IN('main'),
PARTITION `east` VALUES IN('east'),
PARTITION `west` VALUES IN('west'));
```

This example assumes that federated servers had been created named “server_main”, “server_east” and “server_west” and that all remote tables are named “sales”. Note also that in this example, the column id is no more a key.

Current Partition Limitations

Because the partition engine was written before some other engines were added to MariaDB, the way it works is sometime incompatible with these engines, in particular with CONNECT.

Update Statement

With the sample tables above, you can do update statements such as:

```
UPDATE t2 SET msg = 'quatre' WHERE id = 4;
```

It works perfectly and is accepted by CONNECT. However, let us consider the statement:

```
UPDATE t2 SET id = 41 WHERE msg = 'four';
```

This statement is not accepted by CONNECT. The reason is that the column id being part of the partition function, changing its value may require the modified row to be moved to another partition.
The way it is done by the partition engine is to delete the old row and to re-insert the new modified one. However, this is done in a way that is not currently compatible with CONNECT (remember that
CONNECT supports UPDATE in a specific way, in particular for the table type MYSQL)
This limitation could be temporary. Meanwhile the workaround is to manually do what is done above,

Deleting the row to modify and inserting the modified row:

```
DELETE FROM t2 WHERE id = 4;
INSERT INTO t2 VALUES(41, 'four');
```

Alter Table Statement

For all CONNECT outward tables, the `ALTER TABLE` statement does not make any change in the table data. This is why `ALTER TABLE` should not be used; in particular to modify the partition definition, except of course to correct a wrong definition. Note that using `ALTER TABLE` to create a partition table in two steps because column options would be lost is valid as it applies to a table that is not yet partitioned.

As we have seen, it is also safe to use it to create or drop indexes. Otherwise, a simple rule of thumb is to avoid altering a table definition and better drop and re-create a table whose definition must be modified. Just remember that for outward CONNECT tables, dropping a table does not erase the data and that creating it does not modify existing data.

Rowid Special Column

Each partition being handled separately as one table, the `ROWID` special column returns the rank of the row in its partition, not in the whole table. This means that for partition tables `ROWID` and `ROWNUM` are equivalent.

This page is licensed: CC BY-SA / Gnu FDL

Using CONNECT - Virtual and Special Columns

CONNECT supports MariaDB virtual and persistent columns. It is also possible to declare a column as being a CONNECT special column. Let us see on an example how this can be done. The boys table we have seen previously can be recreated as:

```
CREATE TABLE boys (
  linenum INT(6) NOT NULL DEFAULT 0 special=ROWID,
  name CHAR(12) NOT NULL,
  city CHAR(12) NOT NULL,
  birth DATE NOT NULL date_format='DD/MM/YYYY',
  hired DATE NOT NULL date_format='DD/MM/YYYY' flag=36,
  agehired INT(3) AS (floor(datediff(hired,birth)/365.25))
  virtual,
  fn CHAR(100) NOT NULL DEFAULT '' special=FILEID)
ENGINE=CONNECT table_type=FIX file_name='boys.txt' mapped=YES lrecl=47;
```

We have defined two CONNECT special columns. You can give them any name; it is the field `SPECIAL` option that specifies the special column functional name.

Note: the default values specified for the special columns do not mean anything. They are specified just to prevent getting warning messages when inserting new rows.

For the definition of the *agehired* virtual column, no CONNECT options can be specified as it has no offset or length, not being stored in the file.

The command:

```
SELECT * FROM boys WHERE city = 'boston';
```

will return:

linenum	name	city	birth	hired	agehired	fn
1	John	Boston	1986-01-25	2010-06-02	24	d:\mariadb\sql\data\boys.txt
2	Henry	Boston	1987-06-07	2008-04-01	20	d:\mariadb\sql\data\boys.txt
6	Bill	Boston	1986-09-11	2008-02-10	21	d:\mariadb\sql\data\boys.txt

Existing special columns are listed in the following table:

Special Name	Type	Description of the column value
ROWID	Integer	The row ordinal number in the table. This is not quite equivalent to a virtual column with an auto increment of 1 because rows are renumbered when deleting rows.
ROWNUM	Integer	The row ordinal number in the file. This is different from ROWID for multiple tables, TBL/XCOL/OCCUR/PIVOT tables, XML tables with a multiple column, and for DBF tables where ROWNUM includes soft deleted rows.
FILEID FDISK FPATH FNAME FTYPE	String	FILEID returns the full name of the file this row belongs to. Useful in particular for multiple tables represented by several files. The other special columns can be used to retrieve only one part of the full name.
TABID	String	The name of the table this row belongs to. Useful for TBL tables.
PARTID	String	The name of the partition this row belongs to. Specific to partitioned tables.
SERVID	String	The name of the federated server or server host used by a MYSQL table. “ODBC” for an ODBC table, “JDBC” for a JDBC table and “Current” for all other tables.

Note: CONNECT does not currently support auto incremented columns. However, a `ROWID` special column will do the job of a column auto incremented by 1.

This page is licensed: GPLv2

Installing CONNECT

The `CONNECT` storage engine enables MariaDB to access external local or remote data (MED). This is done by defining tables based on different data types, in particular files in various formats, data extracted from other DBMS or products (such as Excel or MongoDB) via ODBC or JDBC, or data retrieved from the environment (for example DIR, WMI, and MAC tables).

This storage engine supports table partitioning, MariaDB virtual columns and permits defining special columns such as ROWID, FILEID, and SERVID.

The storage engine must be installed before it can be used.

Installing the Plugin's Package

The `CONNECT` storage engine's shared library is included in MariaDB packages as the `ha_connect.so` or `ha_connect.so` shared library on systems where it can be built.

Installing on Linux

The `CONNECT` storage engine is included in binary tarballs on Linux.

Installing with a Package Manager

The `CONNECT` storage engine can also be installed via a package manager on Linux. In order to do so, your system needs to be configured to install from one of the MariaDB repositories.

You can configure your package manager to install it from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script.

You can also configure your package manager to install it from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool.

Installing with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using yum or dnf. Starting with RHEL 8 and Fedora 22, `yum` has been replaced by `dnf`, which is the next major version of `yum`. However, `yum` commands still work on many systems that use `dnf`:

```
sudo yum install MariaDB-connect-engine
```

Installing with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant DEB package from MariaDB's repository using apt-get:

```
sudo apt-get install mariadb-plugin-connect
```

Installing with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using zypper:

```
sudo zypper install MariaDB-connect-engine
```

Installing the Plugin

Once the shared library is in place, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'ha_connect';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = ha_connect
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'ha_connect';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Installing Dependencies

The `CONNECT` storage engine has some external dependencies.

Installing unixODBC

The `CONNECT` storage engine requires an ODBC library. On Unix-like systems, that usually means installing unixODBC. On some systems, this is installed as the `unixODBC` package:

```
sudo yum install unixODBC
```

On other systems, this is installed as the `libodbc1` package:

```
sudo apt-get install libodbc1
```

If you do not have the ODBC library installed, then you may get an error about a missing library when you attempt to install the plugin:

```
INSTALL SONAME 'ha_connect';
ERROR 1126 (HY000): Can't open shared library
'/home/ian/MariaDB_Downloads/10.1.17/lib/plugin/ha_connect.so'
  (errno: 2, libodbc.so.1: cannot open shared object file: No such file or directory)
```

See Also

- PLUGIN OVERVIEW

This page is licensed: GPLv2

CONNECT Create Table Options

Create Table statements for “CONNECT” tables are standard MariaDB create statements specifying `engine=CONNECT` . There are a few additional table and column options specific to CONNECT.

Table Options

Table Option	Type	Description
AVG_ROW_LENGTH	Integer	Can be specified to help CONNECT estimate the size of a variable record table length.
BLOCK_SIZE	Integer	The number of rows each block of a FIX, BIN, DBF, or VEC table contains. For an ODBC table this is the RowSet size option. For a JDBC table this is the fetch size.
CATFUNC	String	The catalog function used by a catalog table.
COLIST	String	The column list of OCCUR tables or $project of MONGO tables.
COMPRESS	Number	1 or 2 if the data file is g-zip compressed. Defaults to 0. Before CONNECT 1.05.0001, this was boolean, and true if the data file is compressed.
CONNECTION	String	Specifies the connection of an ODBC, JDBC or MYSQL table.
DATA_CHARSET	String	The character set used in the external file or data source.
DBNAME	String	The target database for ODBC, JDBC, MYSQL, catalog, and PROXY based tables. The database concept is sometimes known as a schema.
ENGINE	String	Must be specified as CONNECT.
ENDING	Integer	End of line length. Defaults to 1 for Unix/Linux and 2 for Windows.
FILE_NAME	String	The file (path) name for all table types based on files. Can be absolute or relative to the current data directory. If not specified, this is an Inward table and a default value is used.
FILTER	String	To filter an external table. Currently MONGO tables only.
HEADER	Integer	Applies to CSV, VEC, and HTML files. Its meaning depends on the table type.
HTTP	String	The HTTP of the client of REST queries. From Connect 1.06.0010.
HUGE	Boolean	To specify that a table file can be larger than 2GB. For a MYSQL table, prevents the result set from being stored in memory.
LRECL	Integer	The file record size (often calculated by default).
MAPPED	Boolean	Specifies whether file mapping is used to handle the table file.
MODULE	String	The (path) name of the DLL or shared lib implementing the access of a non-standard (OEM) table type.
MULTIPLE	Integer	Used to specify multiple file tables.
OPTION_LIST	String	Used to specify all other options not yet directly defined.
QCHAR	String	Specifies the character used for quoting some fields of a CSV table or the identifiers of an ODBC/JDBC tables.
QUOTED	Integer	The level of quoting used in CSV table files.

Table Option	Type	Description
READONLY	Boolean	True if the data file must not be modified or erased.
SEP_CHAR	String	Specifies the field separator character of a CSV or XCOL table. Also, used to specify the Jpath separator for JSON tables.
SEPINDEX	Boolean	When true, indexes are saved in separate files.
SPLIT	Boolean	True for a VEC table when all columns are in separate files.
SRCDEF	String	The source definition of a table retrieved via ODBC, JDBC or the MySQL API or used by a PIVOT table.
SUBTYPE	String	The subtype of an OEM table type.
TABLE_LIST	String	The comma separated list of TBL table sub-tables.
TABLE_TYPE	String	The external table type: DOS, FIX, BIN, CSV, FMT, XML, JSON, INI, DBF, VEC, ODBC, JDBC, MYSQL, TBL, PROXY, XCOL, OCCUR, PIVOT, ZIP, VIR, DIR, WMI, MAC, and OEM. Defaults to DOS, MYSQL, or PROXY depending on what options are used.
TABNAME	String	The target table or node for ODBC, JDBC, MYSQL, PROXY, or catalog tables; or the top node name for XML tables.
URI	String	The URI of a REST request.. From Connect 1.06.0010.
XFILE_NAME	String	The file (path) base name for table index files. Can be absolute or relative to the data directory. Defaults to the file name.
ZIPPED	Boolean	True if the table file(s) is/are zipped in one or several zip files.

All integers in the above table are unsigned big integers.

Because CONNECT handles many table types; many table type specific options are not in the above list and must be entered using the `OPTION_LIST` option. The syntax to use is:

```
... option_list='opname1=opvalue1,opname2=opvalue2...'
```

Be aware that until Connect 1.5.5, no blanks should be inserted before or after the '`=`' and '`,`' characters. The option name is all that is between the start of the string or the last '`,`' character and the next '`=`' character, and the option value is all that is between this '`=`' character and the next '`,`' or end of string. For instance:

```
option_list='name=TABLE,coltype=HTML,attribute=border=1;cellpadding=5,headattr=bgcolor=yellow';
```

This defines four options, '`name`', '`coltype`', '`attribute`', and '`headattr`'; with values '`TABLE`', '`HTML`', '`border=1;cellpadding=5`', and '`bgcolor=yellow`', respectively. The only restriction is that values cannot contain commas, but they can contain equal signs.

Column Options

Column Option	Type	Description
DATE_FORMAT	String	The format indicating how a date is stored in the file.
DISTRIB	Enum	"scattered", "clustered", "sorted" (ascending).
FIELD_FORMAT	String	The column format for some table types.

Column Option	Type	Description
FIELD_LENGTH	Integer	Set the internal field length for DATE columns.
FLAG	Integer	An integer value whose meaning depends on the table type.
JPATH	String	The Json path of JSON table columns.
MAX_DIST	Integer	Maximum number of distinct values in this column.
SPECIAL	String	The name of the SPECIAL column that set this column value.
XPATH	String	The XML path of XML table columns.

- The `MAX_DIST` and `DISTRIB` column options are used for block indexing.
- All integers in the above table are unsigned big integers.
- JPATH and XPATH were added to make CREATE TABLE statements more readable, but they do the same thing as FIELD_FORMAT and any of them can be used with the same result.

Index Options

Index Option	Type	Description
DYNAM	Boolean	Set the index as “dynamic”.
MAPPED	Boolean	Use index file mapping.

Note 1: Creating a CONNECT table based on file does not erase or create the file if the file name is specified in the `CREATE TABLE` statement (“outward” table). If the file does not exist, it are populated by subsequent `INSERT` or `LOAD` statements or by the “AS select statement” of the CREATE TABLE statement. Unlike the CSV engine, CONNECT easily permits the creation of tables based on already existing files, for instance files made by other applications. However, if the file name is not specified, a file with a name defaulting to `tablename.tabletype` are created in the data directory (“inward” table).

Note 2: Dropping a CONNECT table is done with a standard `DROP` statement. For outward tables, this drops only the CONNECT table definition but does not erase the corresponding data file and index files. Use `DELETE` or `TRUNCATE` to do so. This is contrary to data and index files of inward tables are erased on `DROP` like for other MariaDB engines.

This page is licensed: GPLv2

CONNECT Data Types

Many data types make no or little sense when applied to plain files. This why CONNECT supports only a restricted set of data types. However, ODBC, JDBC or MYSQL source tables may contain data types not supported by CONNECT. In this case, CONNECT makes an automatic conversion to a similar supported type when it is possible.

The data types currently supported by CONNECT are:

Type name	Description	Used for
TYPE_STRING	Zero ended string	char, varchar, text
TYPE_INT	4 bytes integer	int, mediumint, integer
TYPE_SHORT	2 bytes integer	smallint
TYPE_TINY	1 byte integer	tinyint
TYPE_BIGINT	8 bytes integer	bigint, longlong
TYPE_DOUBLE	8 bytes floating point	double, float, real
TYPE_DECIM	Numeric value	decimal, numeric, number
TYPE_DATE	4 bytes integer	date, datetime, time, timestamp, year

TYPE_STRING

This type corresponds to what is generally known as CHAR or VARCHAR by database users, or as strings by programmers. Columns containing characters have a maximum length but the character string is of fixed or variable length depending on the file format.

The `DATA_CHARSET` option must be used to specify the character set used in the data source or file. Note that, unlike usually with MariaDB, when a multi-byte character set is used, the column size represents the number of bytes the column value can contain, not the number of characters.

TYPE_INT

The INTEGER type contains signed integer numeric 4-byte values (the *int* of the C language) ranging from `-2,147,483,648` to `2,147,483,647` for signed type and `0` to `4,294,967,295` for unsigned type.

TYPE_SHORT

The `SHORT` data type contains signed integer numeric 2-byte values (the *short integer* of the C language) ranging from `-32,768` to `32,767` for signed type and `0` to `65,535` for unsigned type.

TYPE_TINY

The `TINY` data type contains integer numeric 1-byte values (the *char* of the C language) ranging from `-128` to `127` for signed type and `0` to `255` for unsigned type. For some table types, TYPE_TINY is used to represent Boolean values (0 is false, anything else is true).

TYPE_BIGINT

The BIGINT data type contains signed integer 8-byte values (the *long long* of the C language) ranging from `-9,223,372,036,854,775,808` to `9,223,372,036,854,775,807` for signed type and from `0` to `18,446,744,073,709,551,615` for unsigned type.

Inside tables, the coding of all integer values depends on the table type. In tables represented by text files, the number is written in characters, while in tables represented by binary files (`BIN` or `VEC`) the number is directly stored in the binary representation corresponding to the platform.

The *length* (or *precision*) specification corresponds to the length of the table field in which the value is stored for text files only. It is used to set the output field length for all table types.

TYPE_DOUBLE

The `DOUBLE` data type corresponds to the C language double type, a floating-point double precision value coded with 8 bytes. Like for integers, the internal coding in tables depends on the table type, characters for text files, and platform binary representation for binary files.

The *length* specification corresponds to the length of the table field in which the value is stored for text files only. The *scale* (was_precision_) is the number of decimal digits written into text files. For binary table types (BIN and VEC) this does not apply. The *length* and_scale_ specifications are used to set the output field length and number of decimals for all types of tables.

TYPE_DECIM

The `DECIMAL` data type corresponds to what MariaDB or ODBC data sources call `NUMBER` , `NUMERIC` , or DECIMAL: a numeric value with a maximum number of digits (the precision) some of them eventually being decimal digits (the scale). The internal coding in `CONNECT` is a character representation of the number. For instance:

```
colname DECIMAL(14,6)
```

This defines a column *colname* as a number having a *precision* of 14 and a *scale* of 6. Supposing it is populated by:

```
INSERT INTO xxx VALUES (-2658.74);
```

The internal representation of it are the character string `-2658.740000`. The way it is stored in a file table depends on the table type. The *length* field specification corresponds to the length of the table field in which the value is stored and is calculated by CONNECT from the_precision_ and the *scale* values. This length is *precision* plus 1 if_scale_ is not 0 (for the decimal point) plus 1 if this column is not unsigned (for the eventual minus sign). In fix formatted tables the number is right justified in the field of width *length*, for variable formatted tables, such as CSV, the field is the representing character string.

Because this type is mainly used by CONNECT to handle numeric or decimal fields of ODBC, JDBC and MySQL table types, CONNECT does not provide decimal calculations or comparison by itself. This is why decimal columns of CONNECT tables cannot be indexed.

DATE Data Type

Internally, date/time values are stored by CONNECT as a signed 4-byte integer. The value 0 corresponds to 01 January 1970 12:00:00 am coordinated universal time (UTC). All other date/time values are represented by the number of seconds elapsed since or before midnight (00:00:00), 1 January 1970, to that date/time value. Date/time values before midnight 1 January 1970 are represented by a negative number of seconds.

`CONNECT` handles dates from **13 December 1901, 20:45:52** to**18 January 2038, 19:14:07**.

Although date and time information can be represented in both CHAR and INTEGER data types, the DATE data type has special associated properties. For each DATE value, CONNECT can store all or only some of the following information: century, year, month, day, hour, minute, and second.

Date Format in Text Tables

Internally, date/time values are handled as a signed 4-byte integer. But in text tables (type DOS, FIX, CSV, FMT, and DBF) dates are most of the time stored as a formatted character string (although they also can be stored as a numeric string representing their internal value). Because there are infinite ways to format a date, the format to use for decoding dates, as well as the field length in the file, must be associated to date columns (except when they are stored as the internal numeric value).

Note that this associated format is used only to describe the way the temporal value is stored internally. This format is used both for output to decode the date in a SELECT statement as well as for input to encode the date in INSERT or UPDATE statements. However, what is kept in this value depends on the data type used in the column definition (all the MariaDB temporal values can be specified). When creating a table, the format is associated to a date column using the DATE_FORMAT option in the column definition, for instance:

```
CREATE TABLE birthday (
  Name VARCHAR(17),
  Bday DATE field_length=10 date_format='MM/DD/YYYY',
  Btime TIME field_length=8 date_format='hh:mm tt')
engine=CONNECT table_type=CSV;

INSERT INTO birthday VALUES ('Charlie','2012-11-12','15:30:00');

SELECT * FROM birthday;
```

The `SELECT` query returns:

Name	Bday	Btime
Charlie	2012-11-12	15:30:00

The values of the INSERT statement must be specified using the standard MariaDB syntax and these values are displayed as MariaDB temporal values. Sure enough, the column formats apply only to the way these values are represented inside the CSV files. Here, the inserted record are:

```
Charlie,11/12/2012,03:30 PM
```

Note: The field_length option exists because the MariaDB syntax does not allow specifying the field length between parentheses for temporal column types. If not specified, the field length is calculated from the date format (sometimes as a max value) or made equal to the default length value if there is no date format. In the above example it could have been removed as the calculated values are the ones specified. However, if the table type would have been DOS or FIX, these values could be adjusted to fit the actual field length within the file.

A `CONNECT` format string consists of a series of elements that represent a particular piece of information and define its format. The elements are recognized in the order they appear in the format string. Date and time format elements are replaced by the actual date and time as they appear in the source string. They are defined by the following groups of characters:

Element	Description
YY	The last two digits of the year (that is, 1996 would be coded as “96”).
YYYY	The full year (that is, 1996 could be entered as “96” but displayed as “1996”).
MM	The one or two-digit month number.
MMM	The three-character month abbreviation.
MMMM	The full month name.
DD	The one or two-digit month day.
DDD	The three-character weekday abbreviation.
DDDD	The full weekday name.
hh	The one or two-digit hour in 12-hour or 24-hour format.
mm	The one or two-digit minute.
ss	The one or two-digit second.
t	The one-letter AM/PM abbreviation (that is, AM is entered as “A”).
tt	The two-letter AM/PM abbreviation (that is, AM is entered as “AM”).

Usage Notes

- To match the source string, you can add body text to the format string, enclosing it in single quotes or double quotes if it would be ambiguous. Punctuation marks do not need to be quoted.
- The hour information is regarded as 12-hour format if a “t” or “tt” element follows the “hh” element in the format or as 24-hour format otherwise.
- The “MM”, “DD”, “hh”, “mm”, “ss” elements can be specified with one or two letters (e.g. “MM” or “M”) making no difference on input, but placing a leading zero to one-digit values on output [1] for two-letter elements.
- If the format contains elements DDD or DDDD, the day of week name is skipped on input and ignored to calculate the internal date value. On output, the correct day of week name is generated and displayed.
- Temporal values are always stored as numeric in BIN and VEC tables.

Handling dates that are out of the range of supported CONNECT dates

If you want to make a table containing, for instance, historical dates not being convertible into CONNECT dates, make your column `CHAR` or `VARCHAR` and store the dates in the MariaDB format. All date functions applied to these

strings will convert them to MariaDB dates and will work as if they were real dates. Of course they must be inserted and are displayed using the MariaDB format.

NULL Handling

`CONNECT` handles null values for data sources able to produce nulls. Currently this concerns mainly the ODBC, JDBC, MONGO, MYSQL, XML, JSON and INI table types. For INI, JSON, MONGO or XML types, null values are returned when the key is missing in the section (INI) or when the corresponding node does not exist in a row (XML, JSON, MONGO).

For other file tables, the issue is to define what a null value is. In a numeric column, 0 can sometimes be a valid value but, in some other cases, it can make no sense. The same for character columns; is a blank field a valid value or not?

A special case is DATE columns with a DATE _FORMAT specified. Any value not matching the format can be regarded as NULL.

`CONNECT` leaves the decision to you. When declaring a column in the CREATE TABLE statement, if it is declared NOT NULL, blank or zero values are considered as valid values. Otherwise they are considered as NULL values. In all cases, nulls are replaced on insert or update by pseudo null values, a zero-length character string for text types or a zero value for numeric types. Once converted to pseudo null values, they are recognized as NULL only for columns declared as nullable.

For instance:

```
CREATE TABLE t1 (a INT, b CHAR(10)) ENGINE=CONNECT;
INSERT INTO t1 VALUES (0,'zero'),(1,'one'),(2,'two'),(NULL,'???');
SELECT * FROM t1 WHERE a IS NULL;
```

The select query replies:

a	b
NULL	zero
NULL	???

Sure enough, the value 0 entered on the first row is regarded as NULL for a nullable column. However, if we execute the query:

```
SELECT * FROM t1 WHERE a = 0;
```

This will return no line because a NULL is not equal to 0 in an SQL where clause.

Now let us see what happens with not null columns:

```
CREATE TABLE t1 (a INT NOT NULL, b CHAR(10) NOT NULL) ENGINE=CONNECT;
INSERT INTO t1 VALUES (0,'zero'),(1,'one'),(2,'two'),(NULL,'???');
```

The insert statement will produce a warning saying:

Level	Code	Message
Warning	1048	Column 'a' cannot be null

It is replaced by a pseudo null `0` on the fourth row. Let us see the result:

```
SELECT * FROM t1 WHERE a IS NULL;
SELECT * FROM t1 WHERE a = 0;
```

The first query returns no rows, 0 are valid values and not NULL. The second query replies:

a	b
0	zero
0	???

It shows that the NULL inserted value was replaced by a valid 0 value.

Unsigned Numeric Types

They are supported by CONNECT since version 1.01.0010 for fixed numeric types (TINY, SHORT, INTEGER, and BITINT).

Data Type Conversion

`CONNECT` is able to convert data from one type to another in most cases. These conversions are done without warning even when this leads to truncation or loss of precision. This is true, in particular, for tables of type ODBC, JDBC, MYSQL and PROXY (via MySQL) because the source table may contain some data types not supported by `CONNECT`. They are converted when possible to `CONNECT` types.

When converted, MariaDB types are converted as:

MariaDB Types	CONNECT Type	Remark
integer, medium integer	TYPE_INT	4 byte integer
small integer	TYPE_SHORT	2 byte integer
tiny integer	TYPE_TINY	1 byte integer
char, varchar	TYPE_STRING	Same length
double, float, real	TYPE_DOUBLE	8 byte floating point
decimal, numeric	TYPE_DECIM	Length depends on precision and scale
all date related types	TYPE_DATE	Date format can be set accordingly
bigint, longlong	TYPE_BIGINT	8 byte integer
enum, set	TYPE_STRING	Numeric value not accessible
All text types	TYPE_STRING TYPE_ERROR	Depending on the value of the connect_type_conv system variable value.
Other types	TYPE_ERROR	Not supported, no conversion provided.

For ENUM, the length of the column is the length of the longest value of the enumeration. For SET the length is enough to contain all the set values concatenated with comma separator.

In the case of TEXT columns, the handling depends on the values given to the connect_type_conv and connect_conv_size system variables.

Note: BLOB is currently not converted by default until a TYPE_BIN type is added to CONNECT. However, the FORCE option (from Connect 1.06.006) can be specified for blob columns containing text and the SKIP option also applies to ODBC BLOB columns.

ODBC SQL types are converted as:

SQL Types	Connect Type	Remark
SQL_CHAR, SQL_VARCHAR	TYPE_STRING	
SQL_LONGVARCHAR	TYPE_STRING	len = min(abs(len), connect_conv_size) If the column is generated by discovery (columns not specified) its length is connect_conv_size.

SQL Types	Connect Type	Remark
SQL_NUMERIC, SQL_DECIMAL	TYPE_DECIM	
SQL_INTEGER	TYPE_INT	
SQL_SMALLINT	TYPE_SHORT	
SQL_TINYINT, SQL_BIT	TYPE_TINY	
SQL_FLOAT, SQL_REAL, SQL_DOUBLE	TYPE_DOUBLE	
SQL_DATETIME	TYPE_DATE	len = 10
SQL_INTERVAL	TYPE_STRING	len = 8 + ((scale) ? (scale+1) : 0)
SQL_TIMESTAMP	TYPE_DATE	len = 19 + ((scale) ? (scale +1) : 0)
SQL_BIGINT	TYPE_BIGINT	
SQL_GUID	TYPE_STRING	llen=36
SQL_BINARY, SQL_VARBINARY, SQL_LONG-VARBINARY	TYPE_STRING	len = min(abs(len), connect_conv_size) Only if the value of connect_type_conv is force. The column should use the binary charset.
Other types	TYPE_ERROR	Not supported.

JDBC SQL types are converted as:

JDBC Types	Connect Type	Remark
(N)CHAR, (N)VARCHAR	TYPE_STRING	
LONG(N)VARCHAR	TYPE_STRING	len = min(abs(len), connect_conv_size) If the column is generated by discovery (columns not specified), its length is connect_conv_size
NUMERIC, DECIMAL, VARBINARY	TYPE_DECIM	
INTEGER	TYPE_INT	
SMALLINT	TYPE_SHORT	
TINYINT, BIT	TYPE_TINY	
FLOAT, REAL, DOUBLE	TYPE_DOUBLE	
DATE	TYPE_DATE	len = 10
TIME	TYPE_DATE	len = 8 + ((scale) ? (scale+1) : 0)
TIMESTAMP	TYPE_DATE	len = 19 + ((scale) ? (scale +1) : 0)
BIGINT	TYPE_BIGINT	
UUID (specific to PostgreSQL)	TYPE_STRINGTYPE_ERROR	len=36If connect_type_conv=NO
Other types	TYPE_ERROR	Not supported.

Note: The connect_type_conv SKIP option also applies to ODBC and JDBC tables.

1. ↑ Here input and output are used to specify respectively decoding the date to get its numeric value from the data file and encoding a date to write it in the table file. Input is performed within SELECT queries; output is performed in UPDATE or INSERT queries.

This page is licensed: GPLv2

CONNECT Table Types

- CONNECT Table Types Overview
- CONNECT BIN Table Type
- CONNECT CSV and FMT Table Types
- CONNECT DBF Table Type
- CONNECT DOS and FIX Table Types
- CONNECT - External Table Types
- CONNECT - Files Retrieved Using Rest Queries
- CONNECT INI Table Type
- CONNECT JDBC Table Type: Accessing Tables from Another DBMS
- CONNECT JSON Table Type
- CONNECT MONGO Table Type: Accessing Collections from MongoDB
- CONNECT MYSQL Table Type: Accessing MySQL/MariaDB Tables
- CONNECT - NoSQL Table Types
- CONNECT OCCUR Table Type
- CONNECT ODBC Table Type: Accessing Tables From Another DBMS
- CONNECT PIVOT Table Type
- CONNECT PROXY Table Type
- CONNECT Table Types - Catalog Tables
- CONNECT Table Types - Data Files
- CONNECT Table Types - OEM: Implemented in an External LIB
- CONNECT Table Types - Special “Virtual” Tables
- CONNECT Table Types - VIR
- CONNECT TBL Table Type: Table List
- CONNECT - Using the TBL and MYSQL Table Types Together
- CONNECT VEC Table Type
- CONNECT XCOL Table Type
- CONNECT XML Table Type
- CONNECT Zipped File Tables
- Inward and Outward Tables

This page is licensed: CC BY-SA / Gnu FDL

CONNECT Table Types Overview

CONNECT can handle very many table formats; it is indeed one of its main features. The Type option specifies the type and format of the table. The Type options available values and their descriptions are listed in the following table:

Type	Description
BIN	Binary file with numeric values in platform representation, also with columns at fixed offset within records and fixed record length.

Type	Description
BSON	(Temporary) JSON table handled by the new JSON handling.
CSV *$	“Comma Separated Values” file in which each variable length record contains column values separated by a specific character (defaulting to the comma)
DBF *	File having the dBASE format.
DOS	The table is contained in one or several files. The file format can be refined by some other options of the command or more often using a specific type as many of those described below. Otherwise, it is a flat text file where columns are placed at a fixed offset within each record, the last column being of variable length.
DIR	Virtual table that returns a file list like the Unix ls or DOS dir command.
FIX	Text file arranged like DOS but with fixed length records.
FMT	File in which each record contains the column values in a non-standard format (the same for each record) This format is specified in the column definition.
INI	File having the format of the initialization or configuration files used by many applications.
JDBC*	Table accessed via a JDBC driver.
JSON*$	File having the JSON format.
MAC	Virtual table returning information about the machine and network cards (Windows only).
MONGO *	Table accessed via the MongoDB C Driver API.
MYSQL*	Table accessed using the MySQL API like the FEDERATED engine.
OCCUR*	A table based on another table existing on the current server, several columns of the object table containing values that can be grouped in only one column.
ODBC*	Table extracted from an application accessible via ODBC or unixODBC. For example from another DBMS or from an Excel spreadsheet.
OEM *	Table of any other formats not directly handled by CONNECT but whose access is implemented by an external FDW (foreign data wrapper) written in C++ (as a DLL or Shared Library).
PIVOT*	Used to “pivot” the display of an existing table or view.
PROXY*	A table based on another table existing on the current server.
TBL*	Accessing a collection of tables as one table (like the MERGE engine does for MyIsam tables)
VEC	Binary file organized in vectors, in which column values are grouped consecutively, either split in separate files or in a unique file.
VIR*	Virtual table containing only special and virtual columns.
WMI *	Windows Management Instrumentation table displaying information coming from a WMI provider. This type enables to get in tabular format all sorts of information about the machine hardware and operating system (Windows only).
XCOL*	A table based on another table existing on the current server with one of its columns containing comma separated values.
XML *$	File having the XML or HTML format.
ZIP	Table giving information about the contents of a zip file.

Catalog Tables

For all table types marked with a '*' in the table above, CONNECT is able to analyze the data source to retrieve the column definition. This can be used to define a "catalog" table that display the column description of the source, or to create a table without specifying the column definition that are automatically constructed by CONNECT when creating the table.

When marked with a '$' the file can be the result returned by a REST query.

This page is licensed: GPLv2

CONNECT BIN Table Type

Overview

A table of type BIN is physically a binary file in which each row is a logical record of fixed length[1]. Within a record, column fields are of a fixed offset and length as with FIX tables. Specific to BIN tables is that numerical values are internally encoded using native platform representation, so no conversion is needed to handle numerical values in expressions.

It is not required that the lines of a BIN file be separated by characters such as CR and/or LF but this is possible. In such an event, the *lrecl* option must be specified accordingly.

Note: Unlike for the DOS and FIX types, the width of the fields is the length of their internal representation in the file. For instance for a column declared as:

```
number int(5) not null,
```

The field width in the file is 4 characters, the size of a binary integer. This is the value used to calculate the offset of the next field if it is not specified. Therefore, if the next field is placed 5 characters after this one, this declaration is not enough, and the flag option will have to be used on the next field.

Type Conversion in BIN Tables

Here are the correspondences between the column type and field format provided by default:

Column type	File default format
Char(n)	Text of n characters.
Date	Integer (4 bytes)
Int(n)	Integer (4 bytes)
Smallint(n)	Short integer (2 bytes)
TinyInt(n)	Char (1 Byte)
Bigint(n)	Large integer (8 bytes)
Double(n,d)	Double floating point (8 bytes)

However, the column type need not necessarily match the field format within the table file. In particular, this occurs for field formats that correspond to numeric types that are not handled by CONNECT[2]. Indeed, BIN table files may internally contain float numbers or binary numbers of any byte length in big-endian or little-endian representation[3]. Also, as in DOS or FIX types tables, you may want to handle some character fields as numeric or vice versa.

This is why it is possible to specify the field format when it does not correspond to the column type default using the *field_format* column option in the CREATE TABLE statement. Here are the available field formats for BIN tables:

Field_format	Internal representation
[n]{L or B or H}[n]	n bytes binary number in little endian, big endian or host endian representation.

Field_format	Internal representation
C	Characters string (n bytes)
I	integer (4 bytes)
D	Double float (8 bytes)
S	Short integer (2 bytes)
T	Tiny integer (1 byte)
G	Big integer (8 bytes)
F or R	Real or float (Floating point number on 4 bytes)
X	Use the default format field for the column type

All field formats (except the first one) are a one-character specification[4].
'X' is equivalent to not specifying the field format. For the 'C' character specification, *n* is the column width as specified with the column type. For one-column formats, the number of bytes of the numeric fields corresponds to what it is on most platforms. However, it could vary for some. The G, I, S and T formats are deprecated because they correspond to supported data types and may not be supported in future versions.

Example

Here is an example of a BIN table. The file record layout is supposed to be:

```
NNNNCCCCCCCCCCIIIISSFFFFSS
```

Here N represents numeric characters, C any characters, I integer bytes,
S short integer bytes, and F float number bytes. The `IIII` field contains a date in numeric format.

The table could be created by:

```
CREATE TABLE testbal (
fig INT(4) NOT NULL field_format='C',
name CHAR(10) NOT NULL,
birth DATE NOT NULL field_format='L',
id CHAR(5) NOT NULL field_format='L2',
salary DOUBLE(9,2) NOT NULL DEFAULT 0.00 field_format='F',
dept INT(4) NOT NULL field_format='L2')
ENGINE=CONNECT table_type=BIN block_size=5 file_name='Testbal.dat';
```

Specifying the little-endian representation for binary values is not useful on most machines, but makes the create table statement portable on a machine using big endian, as well as the table file.

The field offsets and the file record length are calculated according the column internal format and eventually modified by the field format. It is not necessary to specify them for a packed binary file without line endings. If a line ending is desired, specify the ending option or specify the `lrecl` option adding the ending width. The table can be filled by:

```
INSERT INTO testbal VALUES
  (5500,'ARCHIBALD','1980-01-25','3789',4380.50,318),
  (123,'OLIVER','1953-08-10','23456',3400.68,2158),
  (3123,'FOO','2002-07-23','888',default,318);
```

Note that the types of the inserted values must match the column type, not the field format type.

The query:

```
SELECT * FROM testbal;
```

returns:

fig	name	birth	id	salary	dept
5500	ARCHIBALD	1980-01-25	3789	4380.50	318
123	OLIVER	1953-08-10	23456	3400.68	2158
3123	FOO	2002-07-23	888	0.00	318

Numeric fields alignment

In binary files, numeric fields and record length can be aligned on 4-or-8-byte boundaries to optimize performance on certain processors. This can be modified in the OPTION_LIST with an “align” option (“packed” meaning `align=1` is the default).

1. ↑ Sometimes it can be a physical record if LF or CRLF have been written in the file.
2. ↑ Most of these are obsolete because CONNECT supports all column types except float
3. ↑ The default endian representation used in the table file can be specified by setting the ENDIAN option as ‘L’ or ‘B’ in the option list.
4. ↑ It can be specified with more than one character, but only the first one is significant.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT CSV and FMT Table Types

CSV Type

Many source data files are formatted with variable length fields and records.
The simplest format, known as `CSV` (Comma Separated Variables), has column fields separated by a separator character. By default, the separator is a comma but can be specified by the `SEP_CHAR` option as any character, for instance a semi-colon.

If the CSV file first record is the list of column names, specifying the `HEADER=1` option will skip the first record on reading. On writing, if the file is empty, the column names record is automatically written.

For instance, given the following *people.csv* file:

```
Name;birth;children
"Archibald";17/05/01;3
"Nabucho";12/08/03;2
```

You can create the corresponding table by:

```
CREATE TABLE people (
  name CHAR(12) NOT NULL,
  birth DATE NOT NULL date_format='DD/MM/YY',
  children SMALLINT(2) NOT NULL)
ENGINE=CONNECT table_type=CSV file_name='people.csv'
header=1 sep_char=';' quoted=1;
```

Alternatively the engine can attempt to automatically detect the column names, data types and widths using:

```
CREATE TABLE people
ENGINE=CONNECT table_type=CSV file_name='people.csv'
header=1 sep_char=';' quoted=1;
```

For CSV tables, the *flag* column option is the rank of the column into the file starting from 1 for the leftmost column. This is to enable having column displayed in a different order than in the file and/or to define the table specifying

only some columns of the CSV file. For instance:

```
CREATE TABLE people (
  name CHAR(12) NOT NULL,
  children SMALLINT(2) NOT NULL flag=3,
  birth DATE NOT NULL flag=2 date_format='DD/MM/YY')
ENGINE=CONNECT table_type=CSV file_name='people.csv'
header=1 sep_char=';' quoted=1;
```

In this case the command:

```
SELECT * FROM people;
```

will display the table as:

name	children	birth
Archibald	3	2001-05-17
Nabucho	2	2003-08-12

Many applications produce CSV files having some fields quoted, in particular because the field text contains the separator character. For such files, specify the 'QUOTED=*n*' option to indicate the level of quoting and/or the '`QCHAR=c`' to specify what is this eventual quoting character, which is `"` by default. Quoting with single quotes must be specified as `QCHAR=''''`. On writing, fields are quoted depending on the value of the quoting level, which is `-1` by default meaning no quoting:

0	The fields between quotes are read and the quotes discarded. On writing, fields are quoted only if they contain the separator character or begin with the quoting character. If they contain the quoting character, it are doubled.
1	Only text fields are written between quotes, except null fields. This includes also the column names of an eventual header.
2	All fields are written between quotes, except null fields.
3	All fields are written between quotes, including null fields.

Files written this way are successfully read by most applications including spreadsheets.

Note 1: If only the QCHAR option is specified, the QUOTED option will default to 1.

Note 2: For CSV tables whose separator is the tab character, specify `sep_char='\t'`.

Note 3: When creating a table on an existing CSV file, you can let CONNECT analyze the file and make the column description. However, this is a not an elaborate analysis of the file and, for instance, `DATE` fields will not be recognized as such but are regarded as string fields.

Note 4: The CSV parser only reads and buffers up to 4KB per row by default, rows longer than this is truncated when read from the file. If the rows are expected to be longer than this use `lrecl` to increase this. For example to set an 8KB maximum row read you would use `lrecl=8192`

Restrictions on CSV Tables

- If secure_file_priv is set to the path of some directory, then CSV tables can only be created with files in that directory.

FMT Type

FMT tables handle files of various formats that are an extension of the concept of CSV files. CONNECT supports these files providing all lines have the same format and that all fields present in all records are recognizable (optional fields must have recognizable delimiters). These files are made by specific application and CONNECT handles them in read only mode.

FMT tables must be created as CSV tables, specifying their type as FMT. In addition, each column description must be added to its format specification.

Column Format Specification of FMT tables

The input format for each column is specified as a FIELD_FORMAT option. A simple example is:

```
IP Char(15) not null field_format=' %n%s%n',
```

In the above example, the format for this (1st) field is `' %n%s%n'`. Note that the blank character at the beginning of this format **is** significant. No trailing blank should be specified in the column formats.

The syntax and meaning of the column input format is the one of the C **scanf** function.

However, CONNECT uses the input format in a specific way. Instead of using it to directly store the input value in the column buffer; it uses it to delimit the sub string of the input record that contains the corresponding column value. Retrieving this value is done later by the column functions as for standard CSV files.

This is why all column formats are made of five components:

1. An eventual description of what is met and ignored before the column value.
2. A marker of the beginning of the column value written as `%n`.
3. The format specification of the column value itself.
4. A marker of the end of the column value written as `%n` (or `%m` for optional fields).
5. An eventual description of what is met after the column value (not valid is `%m` was used).

For example, taking the file *funny.txt*:

```
12345,'BERTRAND',#200;5009.13
 56, 'POIROT-DELMOTTE' ,#4256 ;18009
345 ,'TRUCMUCHE' , #67; 19000.25
```

You can make a table *fmtsample* with 4 columns ID, NAME, DEPNO and SALARY, using the Create Table statement and column formats:

```
CREATE TABLE FMTSAMPLE (
  ID INTEGER(5) NOT NULL field_format=' %n%d%n',
  NAME CHAR(16) NOT NULL field_format=' , ''%n%[^'']%n''',
  DEPNO INTEGER(4) NOT NULL field_format=' , #%n%d%n',
  SALARY DOUBLE(12,2) NOT NULL field_format=' ; %n%f%n')
ENGINE=CONNECT table_type=FMT file_name='funny.txt';
```

Field 1 is an integer (`%d`) with eventual leading blanks.

Field 2 is separated from field 1 by optional blanks, a comma, and other optional blanks and is between single quotes. The leading quote is included in component 1 of the column format, followed by the `%n` marker. The column value is specified as `%[^pg-connect-connect-table-types-connect-csv-and-fmt-table-types-7399dbd1-']` meaning to keep any characters read until a quote is met. The ending marker (`%n`) is followed by the 5th component of the column format, the single quote that follows the column value.

Field 3, also separated by a comma, is a number preceded by a pound sign.

Field 4, separated by a semicolon eventually surrounded by blanks, is a number with an optional decimal point (`%f`).

This table are displayed as:

ID	NAME	DEPNO	SALARY
12345	BERTRAND	200	5009.13

ID	NAME	DEPNO	SALARY
56	POIROT-DELMOTTE	4256	18009.00
345	TRUCMUCHE	67	19000.25

Optional Fields

To be recognized, a field normally must be at least one character long. For instance, a numeric field must have at least one digit, or a character field cannot be void. However many existing files do not follow this format.

Let us suppose for instance that the preceding example file could be:

```
12345,'BERTRAND',#200;5009.13
 56, 'POIROT-DELMOTTE' ,# ;18009
345 ,'' , #67; 19000.25
```

This will display an error message such as *"Bad format line x field y of FMTSAMPLE".* To avoid this and accept these records, the corresponding fields must be specified as "optional". In the above example, fields 2 and 3 can have null values (in lines 3 and 2 respectively). To specify them as optional, their format must be terminated by `%m` (instead of the second `%n`). A statement such as this can do the table creation:

```
CREATE TABLE FMTAMPLE (
  ID INTEGER(5) NOT NULL field_format=' %n%d%n',
  NAME CHAR(16) NOT NULL field_format=' , ''%n%[^''']%m',
  DEPNO INTEGER(4) field_format=''' , #%n%d%m',
  SALARY DOUBLE(12,2) field_format=' ; %n%f%n')
ENGINE=CONNECT table_type=FMT file_name='funny.txt';
```

Note that, because the statement must be terminated by `%m` with no additional characters, skipping the ending quote of field 2 was moved from the end of the second column format to the beginning of the third column format.

The table result is:

ID	NAME	DEPNO	SALARY
12345	BERTRAND	200	5,009.13
56	POIROT-DELMOTTE	NULL	18,009.00
345	NULL	67	19,000.25

Missing fields are replaced by null values if the column is nullable, blanks for character strings and 0 for numeric fields if it is not.

Note 1: Because the formats are specified between quotes, quotes belonging to the formats must be doubled or escaped to avoid a `CREATE TABLE` statement syntax error.

Note 2: Characters separating columns can be included as well in component
5 of the preceding column format or in component 1 of the succeeding column format but for blanks, which should be always included in component 1 of the succeeding column format because line trailing blanks can be sometimes lost.
This is also mandatory for optional fields.

Note 3: Because the format is mainly used to find the sub-string corresponding to a column value, the field specification does not necessarily match the column type. For instance supposing a table contains two integer columns, NBONE and NBTWO, the two lines describing these columns could be:

```
NBONE integer(5) not null field_format=' %n%d%n',
NBTWO integer(5) field_format=' %n%s%n',
```

The first one specifies a required integer field (`%d`), the second line describes a field that can be an integer, but can be replaced by a “-” (or any other) character. Specifying the format specification for this column as a character field (`%s`) enables to recognize it with no error in all cases. Later on, this field are converted to integer by the column read function, and a null 0 value are generated for field specified in their format as non-numeric.

Bad Record Error Processing

When no match if found for a column field the process aborts with a message such as:

```
Bad format line 3 field 4 of funny.txt
```

This can mean as well that one line of the input line is ill formed or that the column format for this field has been wrongly specified. When you know that your file contains records that are ill formatted and should be eliminated from normal processing, set the “maxerr” option of the CREATE TABLE statement, for instance:

```
Option_list='maxerr=100'
```

This will indicate that no error message be raised for the 100 first wrong lines. You can set Maxerr to a number greater than the number of wrong lines in your files to ignore them and get no errors.

Additionally, the “accept” option permit to keep those ill formatted lines with the bad field, and all succeeding fields of the record, nullified. If “accept” is specified without “maxerr”, all ill formatted lines are accepted.

Note: This error processing also applies to CSV tables.

Fields Containing a Formatted Date

A special case is one of columns containing a formatted date. In this case, two formats must be specified:

1. The field recognition format used to delimit the date in the input record.
2. The date format used to interpret the date.
3. The field length option if the date representation is different than the standard type size.

For example, let us suppose we have a web log source file containing records such as:

```
165.91.215.31 - - [17/Jul/2001:00:01:13 -0400] - "GET /usnews/home.htm HTTP/1.1" 302
```

The create table statement shall be like this:

```
CREATE TABLE WEBSAMP (
  IP CHAR(15) NOT NULL field_format='%n%s%n',
  DATE DATETIME NOT NULL field_format=' - - [%n%s%n -0400]'
  date_format='DD/MMM/YYYY:hh:mm:ss' field_length=20,
  FILE CHAR(128) NOT NULL field_format=' - "GET %n%s%n',
  HTTP DOUBLE(4,2) NOT NULL field_format=' HTTP/%n%f%n"',
  NBONE INT(5) NOT NULL field_format=' %n%d%n')
ENGINE=CONNECT table_type=FMT lrecl=400
file_name='e:\\data\\token\\Websamp.dat';
```

Note 1: Here, `field_length=20` was necessary because the default size for datetime columns is only 19. The `lrecl=400` was also specified because the actual file contains more information in each records making the record size calculated by default too small.

Note 2: The file name could have been specified as `'e:/data/token/Websamp.dat'` .

Note 3: FMT tables are currently read only.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT DBF Table Type

Overview

A table of type `DBF` is physically a dBASE III or IV formatted file (used by many products like dBASE, Xbase, FoxPro etc.). This format is similar to the FIX type format with in addition a prefix giving the characteristics of the file, describing in particular all the fields (columns) of the table.

Because `DBF` files have a header that contains Meta data about the file, in particular the column description, it is possible to create a table based on an existing `DBF` file without giving the column description, for instance:

```
CREATE TABLE cust ENGINE=CONNECT table_type=DBF file_name='cust.dbf';
```

To see what CONNECT has done, you can use the `DESCRIBE` or `SHOW CREATE TABLE` statements, and eventually modify some options with the `ALTER TABLE` statement.

The case of deleted lines is handled in a specific way for DBF tables. Deleted lines are not removed from the file but are “soft deleted” meaning they are marked as deleted. In particular, the number of lines contained in the file header does not take care of soft deleted lines. This is why if you execute these two commands applied to a DBF table named *tabdbf*:

```
SELECT COUNT(*) FROM tabdbf;
SELECT COUNT(*) FROM tabdbf WHERE 1;
```

They can give a different result, the (fast) first one giving the number of physical lines in the file and the second one giving the number of line that are not (soft) deleted.

The commands `UPDATE`, `INSERT`, and `DELETE` can be used with DBF tables. The `DELETE` statement marks the deleted lines as suppressed but keeps them in the file. The `INSERT` statement, if it is used to populate a newly created table, constructs the file header before inserting new lines.

Note: For DBF tables, column name length is limited to 11 characters and field length to 256 bytes.

Conversion of dBASE Data Types

CONNECT handles only types that are stored as characters.

Symbol	DBF Type	CONNECT Type	Description
B	Binary (string)	TYPE_STRING	10 digits representing a .DBT block number.
C	Character	TYPE_STRING	All OEM code page characters - padded with blanks to the width of the field.
D	Date	TYPE_DATE	8 bytes - date stored as a string in the format YYYYMMDD.
N	Numeric	TYPE_INT TYPE_BIGINT TYPE_DOUBLE	Number stored as a string, right justified, and padded with blanks to the width of the field.
L	Logical	TYPE_STRING	1 byte - initialized to 0x20 otherwise T or F.
M	Memo (string)	TYPE_STRING	10 digits representing a .DBT block number.
@	Timestamp	Not supported	8 bytes - two longs, first for date, second for time. It is the number of days since 01/01/4713 BC.
I	Long	Not supported	4 bytes. Leftmost bit used to indicate sign, 0 negative.
+	Autoincrement	Not supported	Same as a Long

Symbol	DBF Type	CONNECT Type	Description
F	Float	TYPE_DOUBLE	Number stored as a string, right justified, and padded with blanks to the width of the field.
O	Double	Not supported	8 bytes - no conversions, stored as a double.
G	OLE	TYPE_STRING	10 digits representing a .DBT block number.

For the N numeric type, CONNECT converts it to `TYPE_DOUBLE` if the decimals value is not 0, to `TYPE_BIGINT` if the length value is greater than 10, else to TYPE_INT.

For M, B, and G types, CONNECT just returns the DBT number.

Reading soft deleted lines of a DBF table

It is possible to read these lines by changing the read mode of the table. This is specified by an option `READMODE` that can take the values:

0	Standard mode. This is the default option.
1	Read all lines including soft deleted ones.
2	Read only the soft deleted lines.

For example, to read all lines of the tabdbf table, you can do:

```
ALTER TABLE tabdbf option_list='Readmode=1';
```

To come back to normal mode, specify `READMODE=0`.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT DOS and FIX Table Types

Overview

Tables of type DOS and FIX are based on text files (see CONNECT Table Types - Data Files). Within a record, column fields are positioned at a fixed offset from the beginning of the record. Except sometimes for the last field, column fields are also of fixed length. If the last field has varying length, the type of the table is DOS. For instance, having the file *dept.dat* formatted like:

```
0318 KINGSTON       70012 SALES       Bank/Insurance
0021 ARMONK         87777 CHQ         Corporate headquarter
0319 HARRISON       40567 SALES       Federal Administration
2452 POUGHKEEPSIE   31416 DEVELOPMENT Research & development
```

You can define a table based on it with:

```
CREATE TABLE department (
  NUMBER CHAR(4) NOT NULL,
  LOCATION CHAR(15) NOT NULL flag=5,
  director CHAR(5) NOT NULL flag=20,
  FUNCTION CHAR(12) NOT NULL flag=26,
  name CHAR(22) NOT NULL flag=38)
ENGINE=CONNECT table_type=DOS file_name='dept.dat';
```

Here the flag column option represents the offset of this column inside the records. If the offset of a column is not specified, it defaults to the end of the previous column and defaults to 0 for the first one. The *`lrecl`* parameter that represents the maximum size of a record is calculated by default as the end of the rightmost column and can be unspecified except when some trailing information exists after the rightmost column.

Note: A special case is files having an encoding such as UTF-8 (for instance specifying `charset=UTF8`) in which some characters may be represented with several bytes. Unlike the type size that MariaDB interprets as a number of characters, the `lrecl` value is the record size in bytes and the flag value represents the offset of the field in the record in bytes. If the flag and/or the `lrecl` value are not specified, they are calculated by the number of characters in the fields multiplied by a value that is the maximum size in bytes of a character for the corresponding charset. For UTF-8 this value is 3 which is often far too much as there are very few characters requiring 3 bytes to be represented. When creating a new file, you are on the safe side by only doubling the maximum number of characters of a field to calculate the offset of the next field. Of course, for already existing files, the offset must be specified according to what it is in it.

Although the field representation is always text in the table file, you can freely choose the corresponding column type, characters, date, integer or floating point according to its contents.

Sometimes, as in the *number* column of the above *department* table, you have the choice of the type, numeric or characters. This will modify how the column is internally handled — in characters `0021` is different from `21` but not in numeric — as well as how it is displayed.

If the last field has fixed length, the table should be referred as having the type `FIX`. For instance, to create a table on the file *boys.txt*:

```
John      Boston      25/01/1986  02/06/2010
Henry     Boston      07/06/1987  01/04/2008
George    San Jose    10/08/1981  02/06/2010
Sam       Chicago     22/11/1979  10/10/2007
James     Dallas      13/05/1992  14/12/2009
Bill      Boston      11/09/1986  10/02/2008
```

You can for instance use the command:

```
CREATE TABLE boys (
  name CHAR(12) NOT NULL,
  city CHAR(12) NOT NULL,
  birth DATE NOT NULL date_format='DD/MM/YYYY',
  hired DATE NOT NULL date_format='DD/MM/YYYY' flag=36)
ENGINE=CONNECT table_type=FIX file_name='boys.txt' lrecl=48;
```

Here some *flag* options were not specified because the fields have no intermediate space between them except for the last column. The offsets are calculated by default adding the field length to the *offset* of the preceding field. However, for formatted date columns, the offset in the file depends on the format and cannot be calculated by default. For fixed files, the *lrecl* option is the physical length of the record including the line ending character(s). It is calculated by adding to the end of the last field 2 bytes under Windows (CRLF) or 1 byte under UNIX. If the file is imported from another operating system, the `ENDING` option will have to be specified with the proper value.

For this table, the last offset and the record length must be specified anyway because the date columns have field length coming from their format that is not known by CONNECT. Do not forget to add the line ending length to the total length of the fields.

This table is displayed as:

name	city	birth	hired
John	Boston	1986-01-25	2010-06-02
Henry	Boston	1987-06-07	2008-04-01
George	San Jose	1981-08-10	2010-06-02
Sam	Chicago	1979-11-22	2007-10-10
James	Dallas	1992-05-13	2009-12-14

name	city	birth	hired
Bill	Boston	1986-09-11	2008-02-10

Whenever possible, the fixed format should be preferred to the varying one because it is much faster to deal with fixed tables than with variable tables.
Sure enough, instead of being read or written record by record, FIX tables are processed by blocks of `BLOCK_SIZE` records, resulting in far less input/output operations to execute. The block size defaults to 100 if not specified in the Create Table statement.

Note 1: It is not mandatory to declare in the table all the fields existing in the source file. However, if some fields are ignored, the *flag* option of the following field and/or the *lrecl* option will have to be specified.

Note 2: Some files have an EOF marker (CTRL+Z 1A) that can prevent the table to be recognized as fixed because the file length is not a multiple of the fixed record size. To indicate this, use in the option list the create option EOF. For instance, if after creating the FIX table *xtab* on the file *foo.dat* that you know have fixed record size, you get, when you try to use it, a message such as:

```
File foo.dat is not fixed length, len=302587 lrecl=141
```

After checking that the LRECL default or specified specification is correct, you can indicate to ignore that extra EOF character by:

```
ALTER TABLE xtab option_list='eof=1';
```

Of course, you can specify this option directly in the Create statement. All this applies to some other table types, in particular to BIN tables.

Note 3: The width of the fields is the length specified in the column declaration. For instance for a column declared as:

```
number int(3) not null,
```

The field width in the file is 3 characters. This is the value used to calculate the offset of the next field if it is not specified. If this length is not specified, it defaults to the MySQL default type length.

Specifying the Field Format

Some files have specific format for their numeric fields. For instance, the decimal point is absent and/or the field should be filled with leading zeros.
To deal with such files, as well in reading as in writing, the format can be specified in the `CREATE TABLE` column definition. The syntax of the field format specification is:

```
Field_format='[Z][N][d]'
```

The optional parts of the format are:

Z	The field has leading zeros
N	No decimal point exist in the file
d	The number of decimals, defaults to the column precision

Example

Let us see how it works in the following example. We define a table based on the file xfmt.txt having eight fields of 12 characters:

```
CREATE TABLE xfmt (
  col1 DOUBLE(12,3) NOT NULL,
  col2 DOUBLE(12,3) NOT NULL field_format='4',
  col3 DOUBLE(12,2) NOT NULL field_format='N3',
  col4 DOUBLE(12,3) NOT NULL field_format='Z',
  col5 DOUBLE(12,3) NOT NULL field_format='Z3',
  col6 DOUBLE(12,5) NOT NULL field_format='ZN5',
  col7 INT(12) NOT NULL field_format='N3',
  col8 SMALLINT(12) NOT NULL field_format='N3')
ENGINE=CONNECT table_type=FIX file_name='xfmt.txt';

INSERT INTO xfmt VALUES(4567.056,4567.056,4567.056,4567.056,-23456.8,
    3.14159,4567,4567);
SELECT * FROM xfmt;
```

The first row is displayed as:

COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8
4567.056	4567.056	4567.06	4567.056	-23456.800	3.14159	4567	4567

The number of decimals displayed for all float columns is the column precision, the second argument of the column type option. Of course, integer columns have no decimals, although their formats specify some.

More interesting is the file layout. To see it let us define another table based on the same file but whose columns are all characters:

```
CREATE TABLE cfmt (
  col1 CHAR(12) NOT NULL,
  col2 CHAR(12) NOT NULL,
  col3 CHAR(12) NOT NULL,
  col4 CHAR(12) NOT NULL,
  col5 CHAR(12) NOT NULL,
  col6 CHAR(12) NOT NULL,
  col7 CHAR(12) NOT NULL,
  col8 CHAR(12) NOT NULL)
ENGINE=CONNECT table_type=FIX file_name='xfmt.txt';
SELECT * FROM cfmt;
```

The (transposed) display of the select command shows the file text layout for each field. Below a third column was added in this document to comment this result.

Column	Row 1	Comment (all numeric fields are written right justified)
COL1	4567.056	No format, the value was entered as is.
COL2	4567.0560	The format '4' forces to write 4 decimals.
COL3	4567060	N3 → No decimal point. The last 3 digits are decimals. However, the second decimal was rounded because of the column precision.
COL4	00004567.056	Z → Leading zeros, 3 decimals (the column precision)
COL5	-0023456.800	Z3 → (Minus sign) leading zeros, 3 decimals.
COL6	000000314159	ZN5 → Leading zeros, no decimal point, 5 decimals.
COL7	4567000	N3 → No decimal point. The last 3 digits are decimals.
COL8	4567000	Same. Any decimals would be ignored.

Note: For columns internally using double precision floating-point numbers,
MariaDB limits the decimal precision of any calculation to the column precision. The declared column precision

should be at least the number of decimals of the format to avoid a loss of decimals as it happened for `col3` of the above example.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT - External Table Types

Because so many ODBC and JDBC drivers exist and only the main ones have been heavily tested, these table types cannot be ranked as stable. Use them with care in production applications.

These types can be used to access tables belonging to the current or another database server. Six types are currently provided:

ODBC: To be used to access tables from a database management system providing an ODBC connector. ODBC is a standard of Microsoft and is currently available on Windows. On Linux, it can also be used provided a specific application emulating ODBC is installed. Currently only unixODBC is supported.

JDBC: To be used to access tables from a database management system providing a JDBC connector. JDBC is an Oracle standard implemented in Java and principally meant to be used by Java applications. Using it directly from C or C++ application seems to be almost impossible due to an Oracle bug still not fixed. However, this can be achieved using a Java wrapper class used as an interface between C++ and JDBC. On another hand, JDBC is available on all platforms and operating systems.

Mongo: To access MongoDB collections as tables via their MongoDB C Driver. Because this requires both MongoDB and the C Driver to be installed and operational, this table type is not currently available in binary distributions but only when compiling MariaDB from source.

MySQL: This type is the preferred way to access tables belonging to another MySQL or MariaDB server. It uses the MySQL API to access the external table. Even though this can be obtained using the FEDERATED(X) plugin, this specific type is used internally by CONNECT because it also makes it possible to access tables belonging to the current server.

PROXY: Internally used by some table types to access other tables from one table.

External Table Specification

The four main external table types – odbc, jdbc, mongo and mysql – are specified giving the following information:

1. The data source. This is specified in the connection option.
2. The remote table or view to access. This can be specified within the connection string or using specific CONNECT options.
3. The column definitions. This can be also left to CONNECT to find them using the discovery MariaDB feature.
4. The optional Quoted option. Can be set to 1 to quote the identifiers in the query sent to the remote server. This is required if columns or table names can contain blanks.

The way this works is by establishing a connection to the external data source and by sending it an SQL statement (or its equivalent using API functions for MONGO) enabling it to execute the original query. To enhance performance, it is necessary to have the remote data source do the maximum processing. This is needed in particular to reduce the amount of data returned by the data source.

This is why, for SELECT queries, CONNECT uses the cond_push MariaDB feature to retrieve the maximum of the where clause of the original query that can be added to the query sent to the data source. This is automatic and does not require anything to be done by the user.

However, more can be done. In addition to accessing a remote table, CONNECT offers the possibility to specify what the remote server must do. This is done by specifying it as a view in the srcdef option:

```
CREATE TABLE custnum ENGINE=CONNECT TABLE_TYPE=XXX
CONNECTION='connecton string'
SRCDEF='select pays as country, count(*) as customers from custnum group by pays';
```

Doing so, the group by clause are done by the remote server considerably reducing the amount of data sent back on the connection.

This may even be increased by adding to the srcdef part of the “compatible” part of the query where clauses like this are done for table-based tables. Note that for MariaDB, this table has two columns, country and customers. Supposing the original query is:

```
SELECT * FROM custnum WHERE (country = 'UK' OR country = 'USA') AND customers > 5;
```

How can we make the where clause be added to the sent srcdef? There are many problems:

1. Where to include the additional information.
2. What about the use of alias.
3. How to know what are a where clause or a having clause.

The first problem is solved by preparing the srcdef view to receive clauses. The above example srcdef becomes:

```
SRCDEF='select pays as country, count(*) as customers from custnum where %s group by pays having %s';
```

The *%s* in the srcdef are place holders for eventual compatible parts of the original query where clause. If the select query does not specify a where clause, or a gives an unacceptable where clause, place holders are filled by dummy clauses (1=1).

The other problems must be solved by adding to the create table a list of columns that must be translated because they are aliases or/and aliases on aggregate functions that must become a having clause. For example, in this case:

```
CREATE TABLE custnum ENGINE=CONNECT TABLE_TYPE=XXX
CONNECTION='connecton string'
SRCDEF='select pays as country, count(*) as customers from custnum where %s group by pays having %s'
OPTION_LIST='Alias=customers=*count(*);country=pays';
```

This is specified by the alias option, to be used in the option list. It is made of a semi-colon separated list of items containing:

1. The local column name (alias in the remote server)
2. An equal sign.
3. An eventual ‘*’ indicating this is column correspond to an aggregate function.
4. The remote column name.

With this information, CONNECT are able to make the query sent to the remote data source:

```
SELECT pays AS country, COUNT(*) AS customers FROM custnum WHERE (pays = 'UK' OR pays = 'USA') GROUP BY
country HAVING COUNT(*) > 5
```

Note: Some data sources, including MySQL and MariaDB, accept aliases in the having clause. In that case, the alias option could have been specified as:

```
OPTION_LIST='Alias=customers=*;country=pays';
```

Another option exists, phpos, enabling to specify what place holders are present and in what order. To be specified as “W”, “WH”, “H”, or “HW”. It is rarely used because by default CONNECT can set it from the srcdef content. The only cases it is needed is when the srcdef contains only a having place holder or when the having place holder

occurs before the where place holder, which can occur on queries containing joins. CONNECT cannot handle more than one place holder of each type.

SRCDEF is not available for MONGO tables, but other ways of achieving this exist and are described in the MONGO table type chapter.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT - Files Retrieved Using Rest Queries

> **Warning:**
>
> This storage engine has been deprecated.

Starting with CONNECT version 1.07.0001, JSON, XML and possibly CSV data files can be retrieved as results from REST queries when creating or querying such tables. This is done internally by CONNECT using the CURL program generally available on all systems (if not just install it).

This can also be done using the Microsoft Casablanca (cpprestsdk) package. To enable it, first, install the package as explained in cpprestsdk. Then make the GetRest library (dll or so) as explained in Making the GetRest Library.

Note: If both are available, cpprestsdk is used preferably because it is faster. This can be changed by specifying 'curl=1' in the option list.

Note: If you want to use this feature with an older distributed version of MariaDB not featuring REST, it is possible to add it as an OEM module as explained in Adding the REST Feature as a Library Called by an OEM Table.

Creating Tables using REST

To do so, specify the HTTP of the web client and eventually the URI of the request in the CREATE TABLE statement. For example, for a query returning JSON data:

```
CREATE TABLE webusers (
  id BIGINT(2) NOT NULL,
  name CHAR(24) NOT NULL,
  username CHAR(16) NOT NULL,
  email CHAR(25) NOT NULL,
  address VARCHAR(256) DEFAULT NULL,
  phone CHAR(21) NOT NULL,
  website CHAR(13) NOT NULL,
  company VARCHAR(256) DEFAULT NULL
) ENGINE=CONNECT DEFAULT CHARSET=utf8
TABLE_TYPE=JSON FILE_NAME='users.json' HTTP='http://jsonplaceholder.typicode.com' URI='/users';
```

As with standard JSON tables, discovery is possible, meaning that you can leave CONNECT to define the columns by analyzing the JSON file. Here you could just do:

```
CREATE TABLE webusers
ENGINE=CONNECT DEFAULT CHARSET=utf8
TABLE_TYPE=JSON FILE_NAME='users.json'
HTTP='http://jsonplaceholder.typicode.com' URI='/users';
```

For example, executing:

```
SELECT name, address FROM webusers2 LIMIT 1;
```

returns:

name	address
Leanne Graham	Kulas Light Apt. 556 Gwenborough 92998-3874 -37.3159 81.1496

Here we see that for some complex elements such as *address*, which is a Json object containing values and objects, CONNECT by default has just listed their texts separated by blanks. But it is possible to ask it to analyze in more depth the json result by adding the DEPTH option. For instance:

```
CREATE OR REPLACE TABLE webusers
ENGINE=CONNECT DEFAULT CHARSET=utf8
TABLE_TYPE=JSON FILE_NAME='users.json'
HTTP='http://jsonplaceholder.typicode.com' URI='/users'
OPTION_LIST='Depth=2';
```

Then the table are created as:

```
CREATE TABLE `webusers3` (
  `id` BIGINT(2) NOT NULL,
  `name` CHAR(24) NOT NULL,
  `username` CHAR(16) NOT NULL,
  `email` CHAR(25) NOT NULL,
  `address_street` CHAR(17) NOT NULL `JPATH`='$.address.street',
  `address_suite` CHAR(9) NOT NULL `JPATH`='$.address.suite',
  `address_city` CHAR(14) NOT NULL `JPATH`='$.address.city',
  `address_zipcode` CHAR(10) NOT NULL `JPATH`='$.address.zipcode',
  `address_geo_lat` CHAR(8) NOT NULL `JPATH`='$.address.geo.lat',
  `address_geo_lng` CHAR(9) NOT NULL `JPATH`='$.address.geo.lng',
  `phone` CHAR(21) NOT NULL,
  `website` CHAR(13) NOT NULL,
  `company_name` CHAR(18) NOT NULL `JPATH`='$.company.name',
  `company_catchPhrase` CHAR(40) NOT NULL `JPATH`='$.company.catchPhrase',
  `company_bs` VARCHAR(36) NOT NULL `JPATH`='$.company.bs'
) ENGINE=CONNECT DEFAULT CHARSET=utf8 `TABLE_TYPE`='JSON' `FILE_NAME`='users.json'
`OPTION_LIST`='Depth=2' `HTTP`='http://jsonplaceholder.typicode.com' `URI`='/users';
```

Allowing one to get all the values of the Json result, for example:

```
SELECT name, address_city city, company_name company FROM webusers3;
```

That results in:

name	city	company
Leanne Graham	Gwenborough	Romaguera-Crona
Ervin Howell	Wisokyburgh	Deckow-Crist
Clementine Bauch McKenziehaven	Romaguera-Jacobson	
Patricia Lebsack	South Elvis	Robel-Corkery
Chelsey Dietrich	Roscoeview	Keebler LLC
Mrs. Dennis Schulist	South Christy	Considine-Lockman
Kurtis Weissnat	Howemouth	Johns Group
Nicholas Runolfsdottir V	Aliyaview	Abernathy Group
Glenna Reichert	Bartholomebury	Yost and Sons
Clementina DuBuque	Lebsackbury	Hoeger LLC

Of course, the complete create table (obtained by SHOW CREATE TABLE) can later be edited to make your table return exactly what you want to get. See the JSON table type for details about what and how to specify these.

Note that such tables are read only. In addition, the data are retrieved from the web each time you query the table with a SELECT statement. This is fine if the result varies each time, such as when you query a weather forecasting site. But if you want to use the retrieved file many times without reloading it, just create another table on the same file without specifying the HTTP option.

Note: For JSON tables, specifying the file name is optional and defaults to tabname.type. However, you should specify it if you want to use the file later for other tables.

See the JSON table type for changes that will occur in the new CONNECT versions (distributed in early 2021).

This page is licensed: CC BY-SA / Gnu FDL

CONNECT INI Table Type

Overview

The INI type is one of the configuration or initialization files often found on
Windows machines. For instance, let us suppose you have the following contact file *contact.ini*:

```
[BER]
name=Bertrand
forename=Olivier
address=21 rue Ferdinand Buisson
city=Issy-les-Mlx
zipcode=92130
tel=09.54.36.29.60
cell=06.70.06.04.16

[WEL]
name=Schmitt
forename=Bernard
hired=19/02/1985
address=64 tiergarten strasse
city=Berlin
zipcode=95013
tel=03.43.377.360

[UK1]
name=Smith
forename=Henry
hired=08/11/2003
address=143 Blum Rd.
city=London
zipcode=NW1 2BP
```

CONNECT lets you view it as a table in two different ways.

Column Layout

The first way is to regard it as a table having one line per section, the columns being the keys you want to display. In this case, the `CREATE` statement could be:

```
CREATE TABLE contact (
  contact CHAR(16) flag=1,
  name CHAR(20),
  forename CHAR(32),
  hired DATE date_format='DD/MM/YYYY',
  address CHAR(64),
  city CHAR(20),
  zipcode CHAR(8),
  tel CHAR(16))
ENGINE=CONNECT table_type=INI file_name='contact.ini';
```

The column that will contain the section name can have any name but must specify `flag=1`. All other columns must have the names of the keys we want to display (case insensitive). The type can be character or numeric depending on the key value type, and the length is the maximum expected length for the key value. Once done, the statement:

```
SELECT contact, name, hired, city, tel FROM contact;
```

This statement will display the file in tabular format.

contact	name	hired	city	tel
BER	Bertrand	1970-01-01	Issy-les-Mlx	09.54.36.29.60
WEL	Schmitt	1985-02-19	Berlin	03.43.377.360
UK1	Smith	2003-11-08	London	NULL

Only the keys defined in the create statements are visible; keys that do not exist in a section are displayed as null or pseudo null (blank for character, 1/1/70 for dates, and 0 for numeric) for columns declared `NOT NULL`.

All relational operations can be applied to this table. The table (and the file) can be updated, inserted and conditionally deleted. The only constraint is that when inserting values, the section name must be the first in the list of values.

Note 1: When inserting, if a section already exists, no new section are created but the new values are added or replace those of the existing section. Thus, the following two commands are equivalent:

```
UPDATE contact SET forename = 'Harry' WHERE contact = 'UK1';
INSERT INTO contact (contact,forename) VALUES('UK1','Harry');
```

Note 2: Because sections represent one line, a DELETE statement on a section key will delete the whole section.

Row Layout

To be a good candidate for tabular representation, an INI file should have often the same keys in all sections. In practice, many files commonly found on computers, such as the *win.ini* file of the Windows directory or the_my.ini_ file cannot be viewed that way because each section has different keys. In this case, a second way is to regard the file as a table having one row per section key and whose columns can be the section name, the key name, and the key value.

For instance, let us define the table:

```
create table xcont (
  section char(16) flag=1,
  keyname char(16) flag=2,
  value char(32))
engine=CONNECT table_type=INI file_name='contact.ini'
option_list='Layout=Row';
```

In this statement, the “Layout” option sets the display format, Column by default or anything else not beginning by ‘C’ for row layout display. The names of the three columns can be freely chosen. The Flag option gives the meaning of the column. Specify `flag=1` for the section name and `flag=2` for the key name. Otherwise, the column will contain the key value.

Once done, the command:

```
SELECT * FROM xcont;
```

This displays the following result:

section	keyname	value
BER	name	Bertrand
BER	forename	Olivier
BER	address	21 rue Ferdinand Buisson
BER	city	Issy-les-Mlx

section	keyname	value
BER	zipcode	92130
BER	tel	09.54.36.29.60
BER	cell	06.70.06.04.16
WEL	name	Schmitt
WEL	forename	Bernard
WEL	hired	19/02/1985
WEL	address	64 tiergarten strasse
WEL	city	Berlin
WEL	zipcode	95013
WEL	tel	03.43.377.360
UK1	name	Smith
UK1	forename	Henry
UK1	hired	08/11/2003
UK1	address	143 Blum Rd.
UK1	city	London
UK1	zipcode	NW1 2BP

Note: When processing an INI table, all section names are retrieved in a buffer of 8K bytes (2048 bytes before 10.0.17). For a big file having many sections, this size can be increased using for example:

```
option_list='seclen=16K';
```

This page is licensed: CC BY-SA / Gnu FDL

CONNECT JDBC Table Type: Accessing Tables from Another DBMS

The JDBC table type should be distributed with all recent versions of MariaDB. However, if the automatic compilation of it is possible after the java JDK was installed, the complete distribution of it is not fully implemented in older versions. The distributed JdbcInterface.jar file contains the JdbcInterface wrapper only. New versions distribute a JavaWrappers.jar that contains all currently existing wrappers.

This will require that:

1. The Java SDK is installed on your system.
2. The java wrapper class files are available on your system.
3. And of course, some JDBC drivers exist to be used with the matching DBMS.

Point 2 was made automatic in the newest versions of MariaDB.

Compiling From Source Distribution

Even when the Java JDK has been installed, CMake sometimes cannot find the location where it stands. For instance on Linux the Oracle Java JDK package might be installed in a path not known by the CMake lookup functions causing error message such as:

```
CMake Error at /usr/share/cmake/Modules/FindPackageHandleStandardArgs.cmake:148 (message):
  Could NOT find Java (missing: Java_JAR_EXECUTABLE Java_JAVAC_EXECUTABLE
  Java_JAVAH_EXECUTABLE Java_JAVADOC_EXECUTABLE)
```

When this happen, provide a Java prefix as a hint on where the package was loaded. For instance on Ubuntu I was obliged to enter:

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

After that, the compilation of the CONNECT JDBC type was completed successfully.

Compiling the Java source files

They are the source of the java wrapper classes used to access JDBC drivers. In the source distribution, they are located in the CONNECT source directory.

The default wrapper, JdbcInterface, is the only one distributed with binary distribution. It uses the standard way to get a connection to the drivers via the DriverManager.getConnection method. Other wrappers, only available with source distribution, enable connection to a Data Source, eventually implementing pooling. However, they must be compiled and installed manually.

The available wrappers are:

Wrapper	Description
JdbcInterface	Used to make the connection with available drivers the standard way.
ApacheInterface	Based on the Apache common-dbcp2 package this interface enables making connections to DBCP data sources with any JDBC drivers.
MariadbInterface	Makes connection to a MariaDB data source.
MysqlInterface	Makes connection to a Mysql data source. Must be used with a MySQL driver that implements data sources.
OracleInterface	Makes connection to an Oracle data source.
PostgresqlInterface	Makes connection to a Postgresql data source.

The wrapper used by default is specified by the connect_java_wrapper session variable and is initially set to `wrappers/JdbcInterface`. The wrapper to use for a table can also be specified in the option list as a wrapper option of the “create table” statements.

Note: Conforming java naming usage, class names are preceded by the java package name with a slash separator. However, this is not mandatory for CONNECT which adds the package name if it is missing.

The JdbcInterface wrapper is always usable when Java is present on your machine. Binary distributions have this wrapper already compiled as a JdbcInterface.jar file installed in the plugin directory whose path is automatically included in the class path of the JVM. Recent versions also add a JavaWrappers.jar that contains all these wrappers. Therefore there is no need to worry about its path.

Compiling the ApacheInterface wrapper requires that the Apache common-DBCP2 package be installed. Other wrappers are to be used only with the matching JDBC drivers that must be available when compiling them.

Installing the jar file in the plugin directory is the best place because it is part of the class path. Depending on what is installed on your system, the source files can be reduced accordingly. To compile only the JdbcInterface.java file the CMAKE_JAVA_INCLUDE_PATH is not required. Here the paths are the ones existing on my Windows 7 machine and should be localized.

Setting the Required Information

Before any operation with a JDBC driver can be made, CONNECT must initialize the environment that will make working with Java possible. This will consist of:

1. Loading dynamically the JVM library module.
2. Creating the Java Virtual Machine.

3. Establishing contact with the java wrapper class.
4. Connecting to the used JDBC driver.

Indeed, the JVM library module is not statically linked to the CONNECT plugin. This is to make it possible to use a CONNECT plugin that has been compiled with the JDBC table type on a machine where the Java SDK is not installed. Otherwise, users not interested in the JDBC table type would be obliged to install the Java SDK on their machine to be able to load the CONNECT storage engine.

JVM Library Location

If the JVM library (jvm.dll on Windows, libjvm.so on Linux) was not placed in the standard library load path, CONNECT cannot find it and must be told where to search for it. This happens in particular on Linux when the Oracle Javapackage was installed in a private location.

If the JAVA_HOME variable was exported as explained above, CONNECT can sometimes find it using this information. Otherwise, its search path can be added to the LD_LIBRARY_PATH environment variable. But all this is complicated because making environment variables permanent on Linux is painful (many different methods must be used depending on the Linux version and the used shell).

This is why CONNECT introduced a new global variable connect_jvm_path to store this information. It can be set when starting the server as a command line option or even afterwards before the first use of the JDBC table type:

```
set global connect_jvm_path="/usr/lib/jvm/java-8-oracle/jre/lib/i386/client"
```

or

```
set global connect_jvm_path="/usr/lib/jvm/java-8-oracle/jre/lib/i386/server"
```

The client library is smaller and faster for connection. The server library is more optimized and can be used in case of heavy load usage.

Note that this may not be required on Windows because the path to the JVM library can sometimes be found in the registry.

Once this library is loaded, CONNECT can create the required Java Virtual Machine.

Java Class Path

This is the list of paths Java searches when loading classes. With CONNECT, the classes to load are the java wrapper classes used to communicate with the drivers, and the used JDBC driver classes that are grouped inside jar files. If the ApacheInterface wrapper must be used, the class path must also include all three jars used by the Apache package.

Caution: This class path is passed as a parameter to the Java Virtual Machine (JVM) when creating it and cannot be modified as it is a read only property. In addition, because MariaDB is a multi-threading application, this JVM cannot be destroyed and are used throughout the entire life of the MariaDB server. Therefore, be sure it is correctly set before you use the JDBC table type for the first time. Otherwise, there are practically no alternative than to shut down the server and restart it.

The path to the wrapper classes must point to the directory containing the wrappers sub-directory. If a JdbcInterface.jar file was made, its path is the directory where it is located followed by the jar file name. It is unclear where because this will depend on the installation process. If you start from a source distribution, it can be in the storage/connect directory where the CONNECT source files are or where you moved them or compiled the JdbcInterface.jar file.

For binary distributions, there is nothing to do because the jar file has been installed in the mysql share directory whose path is always automatically included in the class path available to the JVM.

Remaining are the paths of all the installed JDBC drivers that you intend to use. Remember that their path must include the jar file itself. Some applications use an environment variable CLASSPATH to contain them. Paths are separated by ':' on Linux and by ';' on Windows.

If the CLASSPATH variable actually exists and if it is available inside MariaDB, so far so good. You can check this using an UDF function provided by CONNECT that returns environment variable values:

```
CREATE FUNCTION envar RETURNS STRING soname 'ha_connect.so';
SELECT envar('CLASSPATH');
```

Most of the time, this will return null or some required files are missing. This is why CONNECT introduced a global variable to store this information. The paths specified in this variable are added and have precedence to the ones, if any, of the CLASSPATH environment variable. As for the jvm path, this variable connect_class_path should be specified when starting the server but can also be set before using the JDBC table type for the first time.

The current directory (sql/data) is also placed by CONNECT at the beginning of the class path.

As an example, here is how I start MariaDB when doing tests on Linux:

```
olivier@olivier-Aspire-8920:~$ sudo /usr/local/mysql/bin/mysqld -u root --console --default-storage-
engine=myisam --skip-innodb --connect_jvm_path="/usr/lib/jvm/java-8-oracle/jre/lib/i386/server" --
connect_class_path="/home/olivier/mariadb/10.1/storage/connect:/media/olivier/SOURCE/mysql-connector-
java-6.0.2/mysql-connector-java-6.0.2-bin.jar"
```

CONNECT JDBC Tables

These tables are given the type JDBC. For instance, supposing you want to access the boys table located on and external local or remote database management system providing a JDBC connector:

```
CREATE TABLE boys (
name CHAR(12),
city CHAR(12),
birth DATE,
hired DATE);
```

To access this table via JDBC you can create a table such as:

```
CREATE TABLE jboys ENGINE=CONNECT table_type=JDBC tabname=boys
CONNECTION='jdbc:mysql://localhost/dbname?user=root';
```

The CONNECTION option is the URL used to establish the connection with the remote server. Its syntax depends on the external DBMS and in this example is the one used to connect as root to a MySQL or MariaDB local database using the MySQL JDBC connector.

As for ODBC, the columns definition can be omitted and are retrieved by the discovery process. The restrictions concerning column definitions are the same as for ODBC.

Note: The dbname indicated in the URL corresponds for many DBMS to the catalog information. For MySQL and MariaDB it is the schema (often called database) of the connection.

Using a Federated Server

Alternatively, a JDBC table can specify its connection options via a Federated server. For instance, supposing you have a table accessing an external Postgresql table defined as:

```
CREATE TABLE juuid ENGINE=CONNECT table_type=JDBC tabname=testuuid
CONNECTION='jdbc:postgresql:test?user=postgres&password=pwd';
```

You can create a Federated server:

```
CREATE server 'post1' FOREIGN DATA wrapper 'postgresql' OPTIONS (
HOST 'localhost',
DATABASE 'test',
USER 'postgres',
PASSWORD 'pwd',
PORT 0,
SOCKET '',
OWNER 'postgres');
```

Now the JDBC table can be created by:

```
CREATE TABLE juuid ENGINE=CONNECT table_type=JDBC CONNECTION='post1' tabname=testuuid;
```

or by:

```
CREATE TABLE juuid ENGINE=CONNECT table_type=JDBC CONNECTION='post1/testuuid';
```

In any case, the location of the remote table can be changed in the Federated server without having to alter all the tables using this server.

JDBC needs a URL to establish a connection. CONNECT was able to construct that URL from the information contained in such Federated server definition when the URL syntax is similar to the one of MySQL, MariaDB or Postgresql. However, other DBMSs such as Oracle use a different URL syntax. In this case, simply replace the HOST information by the required URL in the Federated server definition. For instance:

```
CREATE server 'oracle' FOREIGN DATA wrapper 'oracle' OPTIONS (
HOST 'jdbc:oracle:thin:@localhost:1521:xe',
DATABASE 'SYSTEM',
USER 'system',
PASSWORD 'manager',
PORT 0,
SOCKET '',
OWNER 'SYSTEM');
```

Now you can create an Oracle table with something like this:

```
CREATE TABLE empor ENGINE=CONNECT table_type=JDBC CONNECTION='oracle/HR.EMPLOYEES';
```

Note: Oracle, as Postgresql, does not seem to understand the DATABASE setting as the table schema that must be specified in the Create Table statement.

Connecting to a JDBC driver

When the connection to the driver is established by the JdbcInterface wrapper class, it uses the options that are provided when creating the CONNECT JDBC tables. Inside the default Java wrapper, the driver's main class is loaded by the DriverManager.getConnection function that takes three arguments:

URL	User	Password
URL	That is the URL that you specified in the CONNECTION option.	
User	As specified in the OPTION_LIST or NULL if not specified.	
Password	As specified in the OPTION_LIST or NULL if not specified.	

The URL varies depending on the connected DBMS. Refer to the documentation of the specific JDBC driver for a description of the syntax to use. User and password can also be specified in the option list.

Beware that the database name in the URL can be interpreted differently depending on the DBMS. For MySQL this is the schema in which the tables are found. However, for Postgresql, this is the catalog and the schema must be specified using the CONNECT dbname option.

For instance a table accessing a Postgresql table via JDBC can be created with a create statement such as:

```
CREATE TABLE jt1 ENGINE=CONNECT table_type=JDBC
CONNECTION='jdbc:postgresql://localhost/mtr' dbname=PUBLIC tabname=t1
option_list='User=mtr,Password=mtr';
```

Note: In previous versions of JDBC, to obtain a connection, java first had to initialize the JDBC driver by calling the method Class.forName. In this case, see the documentation of your DBMS driver to obtain the name of the class that implements the interface java.sql.Driver. This name can be specified as an option DRIVER to be put in the option list. However, most modern JDBC drivers since version 4 are self-loading and do not require this option to be specified.

The wrapper class also creates some required items and, in particular, a statement class. Some characteristics of this statement will depend on the options specified when creating the table:

Scrollable	Block_size
Scrollable	To be specified in the option list. Determines the cursor type: no= forward_only or yes=scroll_insensitive.
Block_size	Will be used to set the statement fetch size.

Fetch Size

The fetch size determines the number of rows that are internally retrieved by the driver on each interaction with the DBMS. Its default value depends on the JDBC driver. It is equal to 10 for some drivers but not for the MySQL or MariaDB connectors.

The MySQL/MariaDB connectors retrieve all the rows returned by one query and keep them in a memory cache. This is generally fine in most cases, but not when retrieving a large result set that can make the query fail with a memory exhausted exception.

To avoid this, when accessing a big table and expecting large result sets, you should specify the BLOCK_SIZE option to 1 (the only acceptable value). However a problem remains:

Suppose you execute a query such as:

```
SELECT id, name, phone FROM jbig LIMIT 10;
```

Not knowing the limit clause, CONNECT sends to the remote DBMS the query:

```
SELECT id, name, phone FROM big;
```

In this query big can be a huge table having million rows. Having correctly specified the block size as 1 when creating the table, the wrapper just reads the 10 first rows and stops. However, when closing the statement, these MySQL/MariaDB drivers must still retrieve all the rows returned by the query. This is why, the wrapper class when closing the statement also cancels the query to stop that extra reading.

The bad news is that if it works all right for some previous versions of the MySQL driver, it does not work for new versions as well as for the MariaDB driver that apparently ignores the cancel command. The good news is that you can use an old MySQL driver to access MariaDB databases. It is also possible that this bug are fixed in future versions of the drivers.

Connection to a Data Source

This is the java preferred way to establish a connection because a data source can keep a pool of connections that can be reused when necessary. This makes establishing connections much faster once it was done for the first time.

CONNECT provide additional wrappers whose files are located in the CONNECT source directory. The wrapper to use can be specified in the global variable connect_java_wrapper, which defaults to “JdbcInterface”.

It can also be specified for a table in the option list by setting the option wrapper to its name. For instance:

```
CREATE TABLE jboys
ENGINE=CONNECT table_type=JDBC tabname='boys'
CONNECTION='jdbc:mariadb://localhost/connect?user=root&useSSL=false'
option_list='Wrapper=MariadbInterface,Scrollable=1';
```

They can be used instead of the standard JdbcInterface and are using created data sources.

The Apache one uses data sources implemented by the Apache-commons-dbcp2 package and can be used with all drivers including those not implementing data sources. However, the Apache package must be installed and its three required jar files accessible via the class path.

1. commons-dbcp2-2.1.1.jar
2. commons-pool2-2.4.2.jar
3. commons-logging-1.2.jar

Note: the versions numbers can be different on your installation.

The other ones use data sources provided by the matching JDBC driver. There are currently four wrappers to be used with mysql-6.0.2, mariadb, oracle and postgresql.

Unlike the class path, the used wrapper can be changed even after the JVM machine was created.

Random Access to JDBC Tables

The same methods described for ODBC tables can be used with JDBC tables.

Note that in the case of the MySQL or MariaDB connectors, because they internally read the whole result set in memory, using the MEMORY option would be a waste of memory. It is much better to specify the use of a scrollable cursor when needed.

Other Operations with JDBC Tables

Except for the way the connection string is specified and the table type set to JDBC, all operations with ODBC tables are done for JDBC tables the same way. Refer to the ODBC chapter to know about:

- Accessing specified views (SRCDEF)
- Data modifying operations.
- Sending commands to a data source.
- JDBC catalog information.

Note: Some JDBC drivers fail when the global time_zone variable is ambiguous, which sometimes happens when it is set to SYSTEM. If so, reset it to a not ambiguous value, for instance:

```
set global time_zone = '+2:00';
```

JDBC Specific Restrictions

Connecting via data sources created externally (for instance using Tomcat) is not supported yet.

Other restrictions are the same as for the ODBC table type.

Handling the UUID Data Type

PostgreSQL has a native UUID data type, internally stored as BIN(16). This is neither an SQL nor a MariaDB data type. The best we can do is to handle it by its character representation.

UUID are translated to CHAR(36) when column definitions are set using discovery. Locally a PostgreSQL UUID column are handled like a CHAR or VARCHAR column. Example:

Using the PostgreSQL table testuuid in the text database:

```
Table « public.testuuid »
 Column | Type | Default
--------+------+--------------------
 id     | uuid | uuid_generate_v4()
 msg    | text |
```

Its column definitions can be queried by:

```
CREATE OR REPLACE TABLE juuidcol ENGINE=CONNECT table_type=JDBC tabname=testuuid catfunc=columns
CONNECTION='jdbc:postgresql:test?user=postgres&password=pwd';
```

```
SELECT TABLE_NAME "Table", COLUMN_NAME "Column", data_type "Type",
  type_name "Name", column_size "Size"
 FROM juuidcol;
```

This query returns:

Table	Column	Type	Name	Size
testuuid	id	1111	uuid	2147483647
testuuid	msg	12	text	2147483647

Note: PostgreSQL, when a column size is undefined, returns 2147483647, which is not acceptable for MariaDB. CONNECT change it to the value of the connect_conv_size session variable. Also, for TEXT columns the data type returned is 12 (SQL_VARCHAR) instead of -1 the SQL_TEXT value.

Accessing this table via JDBC by:

```
CREATE TABLE juuid ENGINE=CONNECT TABLE_TYPE=JDBC TABNAME=testuuid
CONNECTION='jdbc:postgresql:test?user=postgres&password=pwd';
```

it are created by discovery as:

```
CREATE TABLE `juuid` (
  `id` CHAR(36) DEFAULT NULL,
  `msg` VARCHAR(8192) DEFAULT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 CONNECTION='jdbc:postgresql:test?user=postgres&password=pwd'
`TABLE_TYPE`='JDBC' `TABNAME`='testuuid';
```

Note: 8192 being here the *connect_conv_size* value.

Let's populate it:

```
INSERT INTO juuid(msg) VALUES('First');
INSERT INTO juuid(msg) VALUES('Second');
SELECT * FROM juuid;
```

Result:

id	msg
4b173ee1-1488-4355-a7ed-62ba59c2b3e7	First
6859f850-94a7-4903-8d3c-fc3c874fc274	Second

Here the id column values come from the DEFAULT of the PostgreSQL column that was specified as uuid_generate_v4().

It can be set from MariaDB. For instance:

```
INSERT INTO juuid
  VALUES('2f835fb8-73b0-42f3-a1d3-8a532b38feca','inserted');
INSERT INTO juuid VALUES(NULL,'null');
INSERT INTO juuid VALUES('','random');
SELECT * FROM juuid;
```

Result:

id	msg
4b173ee1-1488-4355-a7ed-62ba59c2b3e7	First
6859f850-94a7-4903-8d3c-fc3c874fc274	Second
2f835fb8-73b0-42f3-a1d3-8a532b38feca	inserted
	null
8fc0a30e-dc66-4b95-ba57-497a161f4180	random

The first insert specifies a valid UUID character representation. The second one set it to NULL. The third one (a void string) generates a Java random UUID. UPDATE commands obey the same specification.

These commands both work:

```
SELECT * FROM juuid WHERE id = '2f835fb8-73b0-42f3-a1d3-8a532b38feca';
DELETE FROM juuid WHERE id = '2f835fb8-73b0-42f3-a1d3-8a532b38feca';
```

However, this one fails:

```
SELECT * FROM juuid WHERE id like '%42f3%';
```

Returning:

1296: Got error 174 ’ExecuteQuery: org.postgresql.util.PSQLException:
ERROR: operator does not exist: uuid ~ unknown hint: no operator corresponds to the data name and to the argument types.

because CONNECT cond_push feature added the WHERE clause to the query sent to PostgreSQL:

```
SELECT id, msg FROM testuuid WHERE id LIKE '%42f3%'
```

and the LIKE operator does not apply to UUID in PostgreSQL.

To handle this, a new session variable was added to CONNECT: connect_cond_push. It permits to specify if cond_push is enabled or not for CONNECT and defaults to 1 (enabled). In this case, you can execute:

```
set connect_cond_push=0;
```

Doing so, the where clause are executed by MariaDB only and the query will not fail anymore.

Executing the JDBC tests

Four tests exist but they are disabled because requiring some work to localized them according to the operating system and available java package and JDBC drivers and DBMS.

Two of them, jdbc.test and jdbc_new.test, are accessing MariaDB via JDBC drivers that are contained in a fat jar file that is part of the test. They should be executable without anything to do on Windows; simply adding the option –enable-disabled when running the tests.

However, on Linux these tests can fail to locate the JVM library. Before executing them, you should export the JAVA_HOME environment variable set to the prefix of the java installation or export the LD_LIBRARY_PATH containing the path to the JVM lib.

Fixing Problem With mariadb-dump

In some case or some platform, when CONNECT is set up for use with JDBC table types, this causes mariadb-dump with the option –all-databases to fail.

This was reported by Robert Dyas who found the cause - see the discussion at MDEV-11238.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT JSON Table Type

Overview

JSON (JavaScript Object Notation) is a lightweight data-interchange format widely used on the Internet. Many applications, generally written in JavaScript or PHP use and produce JSON data, which are exchanged as files of different physical formats. JSON data is often returned from REST queries.

It is also possible to query, create or update such information in a database-like manner. MongoDB does it using a JavaScript-like language. PostgreSQL includes these facilities by using a specific data type and related functions like dynamic columns.

The CONNECT engine adds this facility to MariaDB by supporting tables based on JSON data files. This is done like for XML tables by creating tables describing what should be retrieved from the file and how it should be processed.

Starting with 1.07.0002, the internal way JSON was parsed and handled was changed. The main advantage of the new way is to reduce the memory required to parse JSON. It was from 6 to 10 times the size of the JSON source and is now only 2 to 4 times. However, this is in Beta mode and JSON tables are still handled using the old mode. To use the new mode, tables should be created with TABLE_TYPE=BSON. Another way is the set the connect_force_bson session variable to 1 or ON. Then all JSON tables are handled as BSON. Of course, this is temporary and when successfully tested, the new way will replace the old way and all tables be created as JSON.

Let us start from the file “biblio3.json” that is the JSON equivalent of the XML Xsample file described in the XML table chapter:

```
[
  {
    "ISBN": "9782212090819",
    "LANG": "fr",
    "SUBJECT": "applications",
    "AUTHOR": [
      {
        "FIRSTNAME": "Jean-Christophe",
        "LASTNAME": "Bernadac"
      },
      {
        "FIRSTNAME": "François",
        "LASTNAME": "Knab"
      }
    ],
    "TITLE": "Construire une application XML",
    "PUBLISHER": {
      "NAME": "Eyrolles",
      "PLACE": "Paris"
    },
    "DATEPUB": 1999
  },
  {
    "ISBN": "9782840825685",
    "LANG": "fr",
    "SUBJECT": "applications",
    "AUTHOR": [
      {
        "FIRSTNAME": "William J.",
        "LASTNAME": "Pardi"
      }
    ],
    "TITLE": "XML en Action",
    "TRANSLATED": {
       "PREFIX": "adapté de l'anglais par",
       "TRANSLATOR": {
          "FIRSTNAME": "James",
        "LASTNAME": "Guerin"
        }
    },
    "PUBLISHER": {
      "NAME": "Microsoft Press",
      "PLACE": "Paris"
    },
    "DATEPUB": 1999
  }
]
```

This file contains the different items existing in JSON.

- `Arrays` : They are enclosed in square brackets and contain a list of comma separated values.
- `Objects` : They are enclosed in curly brackets. They contain a comma separated list of pairs, each pair composed of a key name between double quotes, followed by a ':' character and followed by a value.
- `Values` : Values can be an array or an object. They also can be a string between double quotes, an integer or float number, a Boolean value or a null value.
  The simplest way for CONNECT to locate a table in such a file is by an array containing a list of objects (this is what MongoDB calls a collection of documents). Each array value are a table row and each pair of the row objects will represent a column, the key being the column name and the value the column value.

A first try to create a table on this file are to take the outer array as the table:

```
CREATE TABLE jsample (
ISBN CHAR(15),
LANG CHAR(2),
SUBJECT CHAR(32),
AUTHOR CHAR(128),
TITLE CHAR(32),
TRANSLATED CHAR(80),
PUBLISHER CHAR(20),
DATEPUB INT(4))
ENGINE=CONNECT table_type=JSON
File_name='biblio3.json';
```

If we execute the query:

```
SELECT isbn, author, title, publisher FROM jsample;
```

We get the result:

isbn	author	title	publisher
9782212090819	Jean-Christophe Bernadac	Construire une application XML	Eyrolles Paris
9782840825685	William J. Pardi	XML en Action	Microsoft Press Pari

Note that by default, column values that are objects have been set to the concatenation of all the string values of the object separated by a blank. When a column value is an array, only the first item of the array is retrieved (This will change in later versions of Connect).

However, things are generally more complicated. If JSON files do not contain attributes (although object pairs are similar to attributes) they contain a new item, arrays. We have seen that they can be used like XML multiple nodes, here to specify several authors, but they are more general because they can contain objects of different types, even it may not be advisable to do so.

This is why CONNECT enables the specification of a column field_format option “JPATH” (FIELD_FORMAT until Connect 1.6) that is used to describe exactly where the items to display are and how to handles arrays.

Here is an example of a new table that can be created on the same file, allowing choosing the column names, to get some sub-objects and to specify how to handle the author array.

Until Connect 1.5:

```
CREATE TABLE jsampall (
ISBN CHAR(15),
LANGUAGE CHAR(2) field_format='LANG',
Subject CHAR(32) field_format='SUBJECT',
Author CHAR(128) field_format='AUTHOR:[" and "]',
Title CHAR(32) field_format='TITLE',
TRANSLATION CHAR(32) field_format='TRANSLATOR:PREFIX',
Translator CHAR(80) field_format='TRANSLATOR',
Publisher CHAR(20) field_format='PUBLISHER:NAME',
LOCATION CHAR(16) field_format='PUBLISHER:PLACE',
YEAR INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=JSON File_name='biblio3.json';
```

From Connect 1.6:

```
CREATE TABLE jsampall (
ISBN CHAR(15),
LANGUAGE CHAR(2) field_format='LANG',
Subject CHAR(32) field_format='SUBJECT',
Author CHAR(128) field_format='AUTHOR.[" and "]',
Title CHAR(32) field_format='TITLE',
TRANSLATION CHAR(32) field_format='TRANSLATOR.PREFIX',
Translator CHAR(80) field_format='TRANSLATOR',
Publisher CHAR(20) field_format='PUBLISHER.NAME',
LOCATION CHAR(16) field_format='PUBLISHER.PLACE',
YEAR INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=JSON File_name='biblio3.json';
```

From Connect 1.07.0002

```
CREATE TABLE jsampall (
ISBN CHAR(15),
LANGUAGE CHAR(2) jpath='$.LANG',
Subject CHAR(32) jpath='$.SUBJECT',
Author CHAR(128) jpath='$.AUTHOR[" and "]',
Title CHAR(32) jpath='$.TITLE',
TRANSLATION CHAR(32) jpath='$.TRANSLATOR.PREFIX',
Translator CHAR(80) jpath='$.TRANSLATOR',
Publisher CHAR(20) jpath='$.PUBLISHER.NAME',
LOCATION CHAR(16) jpath='$.PUBLISHER.PLACE',
YEAR INT(4) jpath='$.DATEPUB')
ENGINE=CONNECT table_type=JSON File_name='biblio3.json';
```

Given the query:

```
SELECT title, author, publisher, LOCATION FROM jsampall;
```

The result is:

title	author	publisher	location
Construire une application XML	Jean-Christophe Bernadac and François Knab	Eyrolles	Paris
XML en Action	William J. Pardi	Microsoft Press	Paris

Note: The JPATH was not specified for column ISBN because it defaults to the column name.

Here is another example showing that one can choose what to extract from the file and how to “expand” an array, meaning to generate one row for each array value:

Until Connect 1.5:

```
CREATE TABLE jsampex (
ISBN CHAR(15),
Title CHAR(32) field_format='TITLE',
AuthorFN CHAR(128) field_format='AUTHOR:[X]:FIRSTNAME',
AuthorLN CHAR(128) field_format='AUTHOR:[X]:LASTNAME',
YEAR INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=JSON File_name='biblio3.json';
```

From Connect 1.6:

```
CREATE TABLE jsampex (
ISBN CHAR(15),
Title CHAR(32) field_format='TITLE',
AuthorFN CHAR(128) field_format='AUTHOR.[X].FIRSTNAME',
AuthorLN CHAR(128) field_format='AUTHOR.[X].LASTNAME',
YEAR INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=JSON File_name='biblio3.json';
```

From Connect 1.06.006:

```
CREATE TABLE jsampex (
ISBN CHAR(15),
Title CHAR(32) field_format='TITLE',
AuthorFN CHAR(128) field_format='AUTHOR[*].FIRSTNAME',
AuthorLN CHAR(128) field_format='AUTHOR[*].LASTNAME',
YEAR INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=JSON File_name='biblio3.json';
```

From Connect 1.07.0002

```
CREATE TABLE jsampex (
ISBN CHAR(15),
Title CHAR(32) jpath='TITLE',
AuthorFN CHAR(128) jpath='AUTHOR[*].FIRSTNAME',
AuthorLN CHAR(128) jpath='AUTHOR[*].LASTNAME',
YEAR INT(4) jpath='DATEPUB')
ENGINE=CONNECT table_type=JSON File_name='biblio3.json';
```

It is displayed as:

ISBN	Title	AuthorFN	AuthorLN	Year
9782212090819	Construire une application XML	Jean-Christophe	Bernadac	1999
9782212090819	Construire une application XML	François	Knab	1999
9782840825685	XML en Action	William J.	Pardi	1999

Note: The example above shows that the '$.', that means the beginning of the path, can be omitted.

The Jpath Specification

From Connect 1.6, the Jpath specification has changed to be the one of the native JSON functions and more compatible with what is generally used. It is close to the standard definition and compatible to what MongoDB and other products do. The ':' separator is replaced by '.'. Position in array is accepted MongoDB style with no square brackets. Array specification specific to CONNECT are still accepted but [*] is used for expanding and [x] for multiply. However, tables created with the previous syntax can still be used by adding SEP_CHAR=':' (can be done with alter table). Also, it can be now specified as JPATH (was FIELD_FORMAT) but FIELD_FORMAT is still accepted.

Until Connect 1.5, it is the description of the path to follow to reach the required item. Each step is the key name (case sensitive) of the pair when crossing an object, and the number of the value between square brackets when crossing an array. Each specification is separated by a ':' character.

From Connect 1.6, It is the description of the path to follow to reach the required item. Each step is the key name (case sensitive) of the pair when crossing an object, and the position number of the value when crossing an array. Key specifications are separated by a '.' character.

For instance, in the above file, the last name of the second author of a book is reached by:

$.AUTHOR[1].LASTNAME *standard style$AUTHOR.1.LASTNAME* MongoDB style
AUTHOR:[1]:LASTNAME *old style when SEP_CHAR=':' or until Connect 1.5*

The '$' or "$." prefix specifies the root of the path and can be omitted with CONNECT.

The array specification can also indicate how it must be processed:

For instance, in the above file, the last name of the second author of a book is reached by:

```
AUTHOR:[1]:LASTNAME
```

The array specification can also indicate how it must be processed:

Specification	Array Type	Limit	Description
n (Connect >= 1.6) or [n][1]	All	N.A	Take the nth value of the array.
[*] (Connect >= 1.6), [X] or [x] (Connect <= 1.5)	All		Expand. Generate one row for each array value.
[“string”]	String		Concatenate all values separated by the specified string.
[+]	Numeric		Make the sum of all the non-null array values.
[x] (Connect >= 1.6), [*] (Connect <= 1.5)	Numeric		Make the product of all non-null array values.
[!]	Numeric		Make the average of all the non-null array values.
[>] or [<]	All		Return the greatest or least non-null value of the array.
[#]	All	N.A	Return the number of values in the array.
[]	All		Expand if under an expanded object. Otherwise sum if numeric, else concatenation separated by “, “.
	All		Between two separators, if an array, expand it if under an expanded object or take the first value of it.

Note 1: When the LIMIT restriction is applicable, only the first *m* array items are used, *m* being the value of the LIMIT option (to be specified in option_list). The LIMIT default value is 10.

Note 2: An alternative way to indicate what is to be expanded is to use the expand option in the option list, for instance:

```
OPTION_LIST='Expand=AUTHOR'
```

`AUTHOR` is here the key of the pair that has the array as a value (case sensitive). Expand is limited to only one branch (expanded arrays must be under the same object).

Let us take as an example the file `expense.json` (found here).
The table jexpall expands all under and including the week array:

From Connect 1.07.0002

```
CREATE TABLE jexpall (
WHO CHAR(12),
WEEK INT(2) jpath='$.WEEK[*].NUMBER',
WHAT CHAR(32) jpath='$.WEEK[*].EXPENSE[*].WHAT',
AMOUNT DOUBLE(8,2) jpath='$.WEEK[*].EXPENSE[*].AMOUNT')
ENGINE=CONNECT table_type=JSON File_name='expense.json';
```

From Connect.1.6

```
CREATE TABLE jexpall (
WHO CHAR(12),
WEEK INT(2) field_format='$.WEEK[*].NUMBER',
WHAT CHAR(32) field_format='$.WEEK[*].EXPENSE[*].WHAT',
AMOUNT DOUBLE(8,2) field_format='$.WEEK[*].EXPENSE[*].AMOUNT')
ENGINE=CONNECT table_type=JSON File_name='expense.json';
```

Until Connect 1.5:

```
CREATE TABLE jexpall (
WHO CHAR(12),
WEEK INT(2) field_format='WEEK:[x]:NUMBER',
WHAT CHAR(32) field_format='WEEK:[x]:EXPENSE:[x]:WHAT',
AMOUNT DOUBLE(8,2) field_format='WEEK:[x]:EXPENSE:[x]:AMOUNT')
ENGINE=CONNECT table_type=JSON File_name='expense.json';
```

WHO	WEEK	WHAT	AMOUNT
Joe	3	Beer	18.00
Joe	3	Food	12.00
Joe	3	Food	19.00
Joe	3	Car	20.00
Joe	4	Beer	19.00
Joe	4	Beer	16.00
Joe	4	Food	17.00
Joe	4	Food	17.00
Joe	4	Beer	14.00
Joe	5	Beer	14.00
Joe	5	Food	12.00
Beth	3	Beer	16.00
Beth	4	Food	17.00
Beth	4	Beer	15.00
Beth	5	Food	12.00
Beth	5	Beer	20.00
Janet	3	Car	19.00
Janet	3	Food	18.00
Janet	3	Beer	18.00
Janet	4	Car	17.00
Janet	5	Beer	14.00
Janet	5	Car	12.00
Janet	5	Beer	19.00
Janet	5	Food	12.00

The table `jexpw` shows what was bought and the sum and average of amounts for each person and week:

From Connect 1.07.0002

```
CREATE TABLE jexpw (
WHO CHAR(12) NOT NULL,
WEEK INT(2) NOT NULL jpath='$.WEEK[*].NUMBER',
WHAT CHAR(32) NOT NULL jpath='$.WEEK[].EXPENSE[", "].WHAT',
SUM DOUBLE(8,2) NOT NULL jpath='$.WEEK[].EXPENSE[+].AMOUNT',
AVERAGE DOUBLE(8,2) NOT NULL jpath='$.WEEK[].EXPENSE[!].AMOUNT')
ENGINE=CONNECT table_type=JSON File_name='expense.json';
```

From Connect 1.6:

```
CREATE TABLE jexpw (
WHO CHAR(12) NOT NULL,
WEEK INT(2) NOT NULL field_format='$.WEEK[*].NUMBER',
WHAT CHAR(32) NOT NULL field_format='$.WEEK[].EXPENSE[", "].WHAT',
SUM DOUBLE(8,2) NOT NULL field_format='$.WEEK[].EXPENSE[+].AMOUNT',
AVERAGE DOUBLE(8,2) NOT NULL field_format='$.WEEK[].EXPENSE[!].AMOUNT')
ENGINE=CONNECT table_type=JSON File_name='expense.json';
```

Until Connect 1.5:

```
CREATE TABLE jexpw (
WHO CHAR(12) NOT NULL,
WEEK INT(2) NOT NULL field_format='WEEK:[x]:NUMBER',
WHAT CHAR(32) NOT NULL field_format='WEEK::EXPENSE:[", "]:WHAT',
SUM DOUBLE(8,2) NOT NULL field_format='WEEK::EXPENSE:[+]:AMOUNT',
AVERAGE DOUBLE(8,2) NOT NULL field_format='WEEK::EXPENSE:[!]:AMOUNT')
ENGINE=CONNECT table_type=JSON File_name='expense.json';
```

WHO	WEEK	WHAT	SUM	AVERAGE
Joe	3	Beer, Food, Food, Car	69.00	17.25
Joe	4	Beer, Beer, Food, Food, Beer	83.00	16.60
Joe	5	Beer, Food	26.00	13.00
Beth	3	Beer	16.00	16.00
Beth	4	Food, Beer	32.00	16.00
Beth	5	Food, Beer	32.00	16.00
Janet	3	Car, Food, Beer	55.00	18.33
Janet	4	Car	17.00	17.00
Janet	5	Beer, Car, Beer, Food	57.00	14.25

Let us see what the table `jexpz` does:

From Connect 1.6:

```
CREATE TABLE jexpz (
WHO CHAR(12) NOT NULL,
WEEKS CHAR(12) NOT NULL field_format='WEEK[", "].NUMBER',
SUMS CHAR(64) NOT NULL field_format='WEEK["+"].EXPENSE[+].AMOUNT',
SUM DOUBLE(8,2) NOT NULL field_format='WEEK[+].EXPENSE[+].AMOUNT',
AVGS CHAR(64) NOT NULL field_format='WEEK["+"].EXPENSE[!].AMOUNT',
SUMAVG DOUBLE(8,2) NOT NULL field_format='WEEK[+].EXPENSE[!].AMOUNT',
AVGSUM DOUBLE(8,2) NOT NULL field_format='WEEK[!].EXPENSE[+].AMOUNT',
AVERAGE DOUBLE(8,2) NOT NULL field_format='WEEK[!].EXPENSE[*].AMOUNT')
ENGINE=CONNECT table_type=JSON File_name='expense.json';
```

From Connect 1.07.0002

```
CREATE TABLE jexpz (
WHO CHAR(12) NOT NULL,
WEEKS CHAR(12) NOT NULL jpath='WEEK[", "].NUMBER',
SUMS CHAR(64) NOT NULL jpath='WEEK["+"].EXPENSE[+].AMOUNT',
SUM DOUBLE(8,2) NOT NULL jpath='WEEK[+].EXPENSE[+].AMOUNT',
AVGS CHAR(64) NOT NULL jpath='WEEK["+"].EXPENSE[!].AMOUNT',
SUMAVG DOUBLE(8,2) NOT NULL jpath='WEEK[+].EXPENSE[!].AMOUNT',
AVGSUM DOUBLE(8,2) NOT NULL jpath='WEEK[!].EXPENSE[+].AMOUNT',
AVERAGE DOUBLE(8,2) NOT NULL jpath='WEEK[!].EXPENSE[*].AMOUNT')
ENGINE=CONNECT table_type=JSON File_name='expense.json';
```

Until Connect 1.5:

```
CREATE TABLE jexpz (
WHO CHAR(12) NOT NULL,
WEEKS CHAR(12) NOT NULL field_format='WEEK:[", "]:NUMBER',
SUMS CHAR(64) NOT NULL field_format='WEEK:["+"]:EXPENSE:[+]:AMOUNT',
SUM DOUBLE(8,2) NOT NULL field_format='WEEK:[+]:EXPENSE:[+]:AMOUNT',
AVGS CHAR(64) NOT NULL field_format='WEEK:["+"]:EXPENSE:[!]:AMOUNT',
SUMAVG DOUBLE(8,2) NOT NULL field_format='WEEK:[+]:EXPENSE:[!]:AMOUNT',
AVGSUM DOUBLE(8,2) NOT NULL field_format='WEEK:[!]:EXPENSE:[+]:AMOUNT',
AVERAGE DOUBLE(8,2) NOT NULL field_format='WEEK:[!]:EXPENSE:[x]:AMOUNT')
ENGINE=CONNECT table_type=JSON
File_name='E:/Data/Json/expense2.json';
```

WHO	WEEKS	SUMS	SUM	AVGS	SUMAVG	AVGSUM	AVERAGE
Joe	3, 4, 5	69.00+83.00+26.00	178.00	17.25+16.60+13.00	46.85	59.33	16.18
Beth	3, 4, 5	16.00+32.00+32.00	80.00	16.00+16.00+16.00	48.00	26.67	16.00
Janet	3, 4, 5	55.00+17.00+57.00	129.00	18.33+17.00+14.25	49.58	43.00	16.12

For all persons:

- Column 1 show the person name.
- Column 2 shows the weeks for which values are calculated.
- Column 3 lists the sums of expenses for each week.
- Column 4 calculates the sum of all expenses by person.
- Column 5 shows the week's expense averages.
- Column 6 calculates the sum of these averages.
- Column 7 calculates the average of the week's sum of expenses.
- Column 8 calculates the average expense by person.

It would be very difficult, if even possible, to obtain this result from table `jexpall` using an SQL query.

Handling of NULL Values

Json has a null explicit value that can be met in arrays or object key values. When regarding json as a relational table, a column value can be null because the corresponding json item is explicitly null, or implicitly because the corresponding item is missing in an array or object. CONNECT does not make any distinction between explicit and implicit nulls.

However, it is possible to specify how nulls are handled and represented. This is done by setting the string session variable connect_json_null. The default value of connect_json_null is “”; it can be changed, for instance, by:

```
SET connect_json_null='NULL';
```

This changes its representation when a column displays the text of an object or the concatenation of the values of an array.

It is also possible to tell CONNECT to ignore nulls by:

```
SET connect_json_null=NULL;
```

When doing so, nulls do not appear in object text or array lists. However, this does not change the behavior of array calculation nor the result of array count.

Having Columns defined by Discovery

It is possible to let the MariaDB discovery process do the job of column specification. When columns are not defined in the create table statement, CONNECT endeavors to analyze the JSON file and to provide the column

specifications. This is possible only for tables represented by an array of objects because CONNECT retrieves the column names from the object pair keys and their definition from the object pair values. For instance, the jsample table could be created saying:

```
CREATE TABLE jsample ENGINE=CONNECT table_type=JSON file_name='biblio3.json';
```

Let's check how it was actually specified using the show create table statement:

```
CREATE TABLE `jsample` (
  `ISBN` CHAR(13) NOT NULL,
  `LANG` CHAR(2) NOT NULL,
  `SUBJECT` CHAR(12) NOT NULL,
  `AUTHOR` VARCHAR(256) DEFAULT NULL,
  `TITLE` CHAR(30) NOT NULL,
  `TRANSLATED` VARCHAR(256) DEFAULT NULL,
  `PUBLISHER` VARCHAR(256) DEFAULT NULL,
  `DATEPUB` INT(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON' `FILE_NAME`='biblio3.json';
```

It is equivalent except for the column sizes that have been calculated from the file as the maximum length of the corresponding column when it was a normal value. For columns that are json arrays or objects, the column is specified as a varchar string of length 256, supposedly big enough to contain the sub-object's concatenated values. Nullable is set to true if the column is null or missing in some rows or if its JPATH contains arrays.

If a more complex definition is desired, you can ask CONNECT to analyse the JPATH up to a given depth using the DEPTH or LEVEL option in the option list. Its default value is 0 but can be changed setting the connect_default_depth session variable (in future versions the default are 5). The depth value is the number of sub-objects that are taken in the JPATH2 (this is different from what is defined and returned by the native Json_Depth function).

For instance:

```
CREATE TABLE jsampall2 ENGINE=CONNECT table_type=JSON
  file_name='biblio3.json' option_list='level=1';
```

This will define the table as:

From Connect 1.07.0002

```
CREATE TABLE `jsampall2` (
  `ISBN` CHAR(13) NOT NULL,
  `LANG` CHAR(2) NOT NULL,
  `SUBJECT` CHAR(12) NOT NULL,
  `AUTHOR_FIRSTNAME` CHAR(15) NOT NULL `JPATH`='$.AUTHOR.[0].FIRSTNAME',
  `AUTHOR_LASTNAME` CHAR(8) NOT NULL `JPATH`='$.AUTHOR.[0].LASTNAME',
  `TITLE` CHAR(30) NOT NULL,
  `TRANSLATED_PREFIX` CHAR(23) DEFAULT NULL `JPATH`='$.TRANSLATED.PREFIX',
  `TRANSLATED_TRANSLATOR` VARCHAR(256) DEFAULT NULL `JPATH`='$.TRANSLATED.TRANSLATOR',
  `PUBLISHER_NAME` CHAR(15) NOT NULL `JPATH`='$.PUBLISHER.NAME',
  `PUBLISHER_PLACE` CHAR(5) NOT NULL `JPATH`='$.PUBLISHER.PLACE',
  `DATEPUB` INT(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON'
  `FILE_NAME`='biblio3.json' `OPTION_LIST`='depth=1';
```

From Connect 1.6:

```
CREATE TABLE `jsampall2` (
  `ISBN` CHAR(13) NOT NULL,
  `LANG` CHAR(2) NOT NULL,
  `SUBJECT` CHAR(12) NOT NULL,
  `AUTHOR_FIRSTNAME` CHAR(15) NOT NULL `FIELD_FORMAT`='AUTHOR..FIRSTNAME',
  `AUTHOR_LASTNAME` CHAR(8) NOT NULL `FIELD_FORMAT`='AUTHOR..LASTNAME',
  `TITLE` CHAR(30) NOT NULL,
  `TRANSLATED_PREFIX` CHAR(23) DEFAULT NULL `FIELD_FORMAT`='TRANSLATED.PREFIX',
  `TRANSLATED_TRANSLATOR` VARCHAR(256) DEFAULT NULL `FIELD_FORMAT`='TRANSLATED.TRANSLATOR',
  `PUBLISHER_NAME` CHAR(15) NOT NULL `FIELD_FORMAT`='PUBLISHER.NAME',
  `PUBLISHER_PLACE` CHAR(5) NOT NULL `FIELD_FORMAT`='PUBLISHER.PLACE',
  `DATEPUB` INT(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON'
  `FILE_NAME`='biblio3.json' `OPTION_LIST`='level=1';
```

Until Connect 1.5:

```
CREATE TABLE `jsampall2` (
  `ISBN` CHAR(13) NOT NULL,
  `LANG` CHAR(2) NOT NULL,
  `SUBJECT` CHAR(12) NOT NULL,
  `AUTHOR_FIRSTNAME` CHAR(15) NOT NULL `FIELD_FORMAT`='AUTHOR::FIRSTNAME',
  `AUTHOR_LASTNAME` CHAR(8) NOT NULL `FIELD_FORMAT`='AUTHOR::LASTNAME',
  `TITLE` CHAR(30) NOT NULL,
  `TRANSLATED_PREFIX` CHAR(23) DEFAULT NULL `FIELD_FORMAT`='TRANSLATED:PREFIX',
  `TRANSLATED_TRANSLATOR` VARCHAR(256) DEFAULT NULL `FIELD_FORMAT`='TRANSLATED:TRANSLATOR',
  `PUBLISHER_NAME` CHAR(15) NOT NULL `FIELD_FORMAT`='PUBLISHER:NAME',
  `PUBLISHER_PLACE` CHAR(5) NOT NULL `FIELD_FORMAT`='PUBLISHER:PLACE',
  `DATEPUB` INT(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON' `
  FILE_NAME`='biblio3.json' `OPTION_LIST`='level=1';
```

For columns that are a simple value, the Json path is the column name. This is the default when the Jpath option is not specified, so it was not specified for such columns. However, you can force discovery to specify it by setting the connect_all_path variable to 1 or ON. This can be useful if you plan to change the name of such columns and relieves you of manually specifying the path (otherwise it would default to the new name and cause the column to not or wrongly be found).

Another problem is that CONNECT cannot guess what you want to do with arrays. Here the AUTHOR array is set to 0, which means that only its first value are retrieved unless you also had specified “Expand=AUTHOR” in the option list. But of course, you can replace it with anything else.

This method can be used as a quick way to make a “template” table definition that can later be edited to make the desired definition. In particular, column names are constructed from all the object keys of their path in order to have distinct column names. This can be manually edited to have the desired names, provided their JPATH key names are not modified.

DEPTH can also be given the value -1 to create only columns that are simple values (no array or object). It normally defaults to 0 but this can be modified setting the connect_default_depth variable.

Note: Since version 1.6.4, CONNECT eliminates columns that are “void” or whose type cannot be determined. For instance given the file sresto.json:

```
{"_id":1,"name":"Corner Social","cuisine":"American","grades":[{"grade":"A","score":6}]}
{"_id":2,"name":"La Nueva Clasica Antillana","cuisine":"Spanish","grades":[]}
```

Previously, when using discovery, creating the table by:

```
CREATE TABLE sjr0
ENGINE=CONNECT table_type=JSON file_name='sresto.json'
option_list='Pretty=0,Depth=1' lrecl=128;
```

The table was previously created as:

```
CREATE TABLE `sjr0` (
  `_id` BIGINT(1) NOT NULL,
  `name` CHAR(26) NOT NULL,
  `cuisine` CHAR(8) NOT NULL,
  `grades` CHAR(1) DEFAULT NULL,
  `grades_grade` CHAR(1) DEFAULT NULL `JPATH`='$.grades[0].grade',
  `grades_score` BIGINT(1) DEFAULT NULL `JPATH`='$.grades[0].score'
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON'
  `FILE_NAME`='sresto.json'
  `OPTION_LIST`='Pretty=0,Depth=1,Accept=1' `LRECL`=128;
```

The column “grades” was added because of the void array in line 2. Now this column is skipped and does not appear anymore (unless the option `Accept=1` is added in the option list).

JSON Catalogue Tables

Another way to see JSON table column specifications is to use a catalogue table. For instance:

```
CREATE TABLE bibcol ENGINE=CONNECT table_type=JSON file_name='biblio3.json'
  option_list='level=2' catfunc=columns;
SELECT COLUMN_NAME, type_name TYPE, column_size SIZE, jpath FROM bibcol;
```

which returns:

From Connect 1.07.0002:

column_name	type	size	jpath
ISBN	CHAR	13	$.ISBN
LANG	CHAR	2	$.LANG
SUBJECT	CHAR	12	$.SUBJECT
AUTHOR_FIRSTNAME	CHAR	15	$.AUTHOR[0].FIRSTNAME
AUTHOR_LASTNAME	CHAR	8	$.AUTHOR[0].LASTNAME
TITLE	CHAR	30	$.TITLE
TRANSLATED_PREFIX	CHAR	23	$.TRANSLATED.PREFIX
TRANSLATED_TRANSLATOR_FIRSTNAME	CHAR	5	$TRANSLATED.TRANSLATOR.FIRSTNAME
TRANSLATED_TRANSLATOR_LASTNAME	CHAR	6	$.TRANSLATED.TRANSLATOR.LASTNAME
PUBLISHER_NAME	CHAR	15	$.PUBLISHER.NAME
PUBLISHER_PLACE	CHAR	5	$.PUBLISHER.PLACE
DATEPUB	INTEGER	4	$.DATEPUB

From Connect 1.6:

column_name	type	size	jpath
ISBN	CHAR	13	
LANG	CHAR	2	
SUBJECT	CHAR	12	
AUTHOR_FIRSTNAME	CHAR	15	AUTHOR..FIRSTNAME
AUTHOR_LASTNAME	CHAR	8	AUTHOR..LASTNAME
TITLE	CHAR	30	
TRANSLATED_PREFIX	CHAR	23	TRANSLATED.PREFIX

column_name	type	size	jpath
TRANSLATED_TRANSLATOR_FIRSTNAME	CHAR	5	TRANSLATED.TRANSLATOR.FIRSTNAME
TRANSLATED_TRANSLATOR_LASTNAME	CHAR	6	TRANSLATED.TRANSLATOR.LASTNAME
PUBLISHER_NAME	CHAR	15	PUBLISHER.NAME
PUBLISHER_PLACE	CHAR	5	PUBLISHER.PLACE
DATEPUB	INTEGER	4	

Until Connect 1.5:

column_name	type	size	jpath
ISBN	CHAR	13	
LANG	CHAR	2	
SUBJECT	CHAR	12	
AUTHOR_FIRSTNAME	CHAR	15	AUTHOR::FIRSTNAME
AUTHOR_LASTNAME	CHAR	8	AUTHOR::LASTNAME
TITLE	CHAR	30	
TRANSLATED_PREFIX	CHAR	23	TRANSLATED:PREFIX
TRANSLATED_TRANSLATOR_FIRSTNAME	CHAR	5	TRANSLATED:TRANSLATOR:FIRSTNAME
TRANSLATED_TRANSLATOR_LASTNAME	CHAR	6	TRANSLATED:TRANSLATOR:LASTNAME
PUBLISHER_NAME	CHAR	15	PUBLISHER:NAME
PUBLISHER_PLACE	CHAR	5	PUBLISHER:PLACE
DATEPUB	INTEGER	4	

All this is mostly useful when creating a table on a remote file that you cannot easily see.

Finding the table within a JSON file

Given the file “facebook.json”:

```
{
   "data": [
      {
         "id": "X999_Y999",
         "from": {
            "name": "Tom Brady", "id": "X12"
         },
         "message": "Looking forward to 2010!",
         "actions": [
            {
               "name": "Comment",
               "link": "http://www.facebook.com/X999/posts/Y999"
            },
            {
               "name": "Like",
               "link": "http://www.facebook.com/X999/posts/Y999"
            }
         ],
         "type": "status",
         "created_time": "2010-08-02T21:27:44+0000",
         "updated_time": "2010-08-02T21:27:44+0000"
      },
      {
         "id": "X998_Y998",
         "from": {
            "name": "Peyton Manning", "id": "X18"
         },
         "message": "Where's my contract?",
         "actions": [
            {
               "name": "Comment",
               "link": "http://www.facebook.com/X998/posts/Y998"
            },
            {
               "name": "Like",
               "link": "http://www.facebook.com/X998/posts/Y998"
            }
         ],
         "type": "status",
         "created_time": "2010-08-02T21:27:44+0000",
         "updated_time": "2010-08-02T21:27:44+0000"
      }
   ]
}
```

The table we want to analyze is represented by the array value of the “data” object. Here is how this is specified in the create table statement:

From Connect 1.07.0002:

```
CREATE TABLE jfacebook (
`ID` CHAR(10) jpath='id',
`Name` CHAR(32) jpath='from.name',
`MyID` CHAR(16) jpath='from.id',
`Message` VARCHAR(256) jpath='message',
`Action` CHAR(16) jpath='actions..name',
`Link` VARCHAR(256) jpath='actions..link',
`Type` CHAR(16) jpath='type',
`Created` DATETIME date_format='YYYY-MM-DD\'T\'hh:mm:ss' jpath='created_time',
`Updated` DATETIME date_format='YYYY-MM-DD\'T\'hh:mm:ss' jpath='updated_time')
ENGINE=CONNECT table_type=JSON file_name='facebook.json' option_list='Object=data,Expand=actions';
```

From Connect 1.6:

```
CREATE TABLE jfacebook (
`ID` CHAR(10) field_format='id',
`Name` CHAR(32) field_format='from.name',
`MyID` CHAR(16) field_format='from.id',
`Message` VARCHAR(256) field_format='message',
`Action` CHAR(16) field_format='actions..name',
`Link` VARCHAR(256) field_format='actions..link',
`Type` CHAR(16) field_format='type',
`Created` DATETIME date_format='YYYY-MM-DD\'T\'hh:mm:ss' field_format='created_time',
`Updated` DATETIME date_format='YYYY-MM-DD\'T\'hh:mm:ss' field_format='updated_time')
ENGINE=CONNECT table_type=JSON file_name='facebook.json' option_list='Object=data,Expand=actions';
```

Until Connect 1.5:

```
CREATE TABLE jfacebook (
`ID` CHAR(10) field_format='id',
`Name` CHAR(32) field_format='from:name',
`MyID` CHAR(16) field_format='from:id',
`Message` VARCHAR(256) field_format='message',
`Action` CHAR(16) field_format='actions::name',
`Link` VARCHAR(256) field_format='actions::link',
`Type` CHAR(16) field_format='type',
`Created` DATETIME date_format='YYYY-MM-DD\'T\'hh:mm:ss' field_format='created_time',
`Updated` DATETIME date_format='YYYY-MM-DD\'T\'hh:mm:ss' field_format='updated_time')
ENGINE=CONNECT table_type=JSON file_name='facebook.json' option_list='Object=data,Expand=actions';
```

This is the object option that gives the Jpath of the table. Note also an alternate way to declare the array to be expanded by the expand option of the option_list.

Because some string values contain a date representation, the corresponding columns are declared as datetime and the date format is specified for them.

The Jpath of the object option has the same syntax as the column Jpath but of course all array steps must be specified using the [n] (until Connect 1.5) or n (from Connect 1.6) format.

Note: This applies to the whole document for tables having `PRETTY = 2` (see below). Otherwise, it applies to the document objects of each file records.

JSON File Formats

The examples we have seen so far are files that, even they can be formatted in different ways (blanks, tabs, carriage return and line feed are ignored when parsing them), respect the JSON syntax and are made of only one item (Object or Array). Like for XML files, they are entirely parsed and a memory representation is made used to process them. This implies that they are of reasonable size to avoid an out of memory condition. Tables based on such files are recognized by the option Pretty=2 that we did not specify above because this is the default.

An alternate format, which is the format of exported MongoDB files, is a file where each row is physically stored in one file record. For instance:

```
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338, "state" : "MA" }
{ "_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51564999999999, 42.377017 ], "pop" : 36963, "state"
: "MA" }
{ "_id" : "01005", "city" : "BARRE", "loc" : [ -72.1083540000001, 42.409698 ], "pop" : 4546, "state" :
"MA" }
{ "_id" : "01007", "city" : "BELCHERTOWN", "loc" : [ -72.4109530000001, 42.275103 ], "pop" : 10579,
"state" : "MA" }
…
{ "_id" : "99929", "city" : "WRANGELL", "loc" : [ -132.352918, 56.433524 ], "pop" : 2573, "state" : "AK"
}
{ "_id" : "99950", "city" : "KETCHIKAN", "loc" : [ -133.18479, 55.942471 ], "pop" : 422, "state" : "AK" }
```

The original file, “cities.json”, has 29352 records. To base a table on this file we must specify the option Pretty=0 in the option list. For instance:

From Connect 1.07.0002:

```
CREATE TABLE cities (
`_id` CHAR(5) KEY,
`city` CHAR(32),
`lat` DOUBLE(12,6) jpath='loc.0',
`long` DOUBLE(12,6) jpath='loc.1',
`pop` INT(8),
`state` CHAR(2) distrib='clustered')
ENGINE=CONNECT table_type=JSON file_name='cities.json' lrecl=128 option_list='pretty=0';
```

From Connect 1.6:

```
CREATE TABLE cities (
`_id` CHAR(5) KEY,
`city` CHAR(32),
`lat` DOUBLE(12,6) field_format='loc.0',
`long` DOUBLE(12,6) field_format='loc.1',
`pop` INT(8),
`state` CHAR(2) distrib='clustered')
ENGINE=CONNECT table_type=JSON file_name='cities.json' lrecl=128 option_list='pretty=0';
```

Until Connect 1.5:

```
CREATE TABLE cities (
`_id` CHAR(5) KEY,
`city` CHAR(32),
`long` DOUBLE(12,6) field_format='loc:[0]',
`lat` DOUBLE(12,6) field_format='loc:[1]',
`pop` INT(8),
`state` CHAR(2) distrib='clustered')
ENGINE=CONNECT table_type=JSON file_name='cities.json' lrecl=128 option_list='pretty=0';
```

Note the use of [n] (until Connect 1.5) or n (from Connect 1.6) array specifications for the longitude and latitude columns.

When using this format, the table is processed by CONNECT like a DOS, CSV or FMT table. Rows are retrieved and parsed by records and the table can be very large. Another advantage is that such a table can be indexed, which can be of great value for very large tables. The “distrib” option of the “state” column tells CONNECT to use block indexing when possible.

For such tables – as well as for *pretty=1* ones – the record size must be specified using the LRECL option. Be sure you don’t specify it too small as it is used to allocate the read/write buffers and the memory used for parsing the rows. If in doubt, be generous as it does not cost much in memory allocation.

Another format exists, noted by Pretty=1, which is similar to this one but has some additions to represent a JSON array. A header and a trailer records are added containing the opening and closing square bracket, and all records but the last are followed by a comma. It has the same advantages for reading and updating, but inserting and deleting are executed in the pretty=2 way.

Alternate Table Arrangement

We have seen that the most natural way to represent a table in a JSON file is to make it on an array of objects. However, other possibilities exist. A table can be an array of arrays, a one column table can be an array of values, or a one row table can be just one object or one value. Single row tables are internally handled by adding a one value array around them.

Let us see how to handle, for instance, a table that is an array of arrays. The file:

```
[
  [56, "Coucou", 500.00],
  [[2,0,1,4], "Hello World", 2.0316],
  ["1784", "John Doo", 32.4500],
  [1914, ["Nabucho","donosor"], 5.12],
  [7, "sept", [0.77,1.22,2.01]],
  [8, "huit", 13.0]
]
```

A table can be created on this file as:

From Connect 1.07.0002:

```
CREATE TABLE xjson (
`a` INT(6) jpath='1',
`b` CHAR(32) jpath='2',
`c` DOUBLE(10,4) jpath='3')
ENGINE=CONNECT table_type=JSON file_name='test.json' option_list='Pretty=1,Jmode=1,Base=1' lrecl=128;
```

From Connect 1.6:

```
CREATE TABLE xjson (
`a` INT(6) field_format='1',
`b` CHAR(32) field_format='2',
`c` DOUBLE(10,4) field_format='3')
ENGINE=CONNECT table_type=JSON file_name='test.json' option_list='Pretty=1,Jmode=1,Base=1' lrecl=128;
```

Until Connect 1.5:

```
CREATE TABLE xjson (
`a` INT(6) field_format='[1]',
`b` CHAR(32) field_format='[2]',
`c` DOUBLE(10,4) field_format='[3]')
ENGINE=CONNECT table_type=JSON file_name='test.json'
option_list='Pretty=1,Jmode=1,Base=1' lrecl=128;
```

Columns are specified by their position in the row arrays. By default, this is zero-based but for this table the base was set to 1 by the *Base* option of the option list. Another new option in the option list is Jmode=1.
It indicates what type of table this is. The Jmode values are:

1. An array of objects. This is the default.
2. An array of Array. Like this one.
3. An array of values.

When reading, this is not required as the type of the array items is specified for the columns; however, it is required when inserting new rows so CONNECT knows what to insert. For instance:

```
INSERT INTO xjson VALUES(25, 'Breakfast', 1.414);
```

After this, it is displayed as:

a	b	c
56	Coucou	500.0000
2	Hello World	2.0316
1784	John Doo	32.4500
1914	Nabucho	5.1200
7	sept	0.7700
8	huit	13.0000

a	b	c
25	Breakfast	1.4140

Unspecified array values are represented by their first element.

Getting and Setting JSON Representation of a Column

We have seen that columns corresponding to a Json object or array are retrieved by default as the concatenation of all its values separated by a blank. It is also possible to retrieve and display such column contains as the full JSON string corresponding to it in the JSON file. This is specified in the JPATH by a "*" where the object or array would be specified.

Note: When having columns generated by discovery, this can be specified by adding the STRINGIFY option to ON or 1 in the option list.

For instance:

From Connect 1.07.0002:

```
CREATE TABLE jsample2 (
ISBN CHAR(15),
Lng CHAR(2) jpath='LANG',
json_Author CHAR(255) jpath='AUTHOR.*',
Title CHAR(32) jpath='TITLE',
YEAR INT(4) jpath='DATEPUB')
ENGINE=CONNECT table_type=JSON file_name='biblio3.json';
```

From Connect 1.6:

```
CREATE TABLE jsample2 (
ISBN CHAR(15),
Lng CHAR(2) field_format='LANG',
json_Author CHAR(255) field_format='AUTHOR.*',
Title CHAR(32) field_format='TITLE',
YEAR INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=JSON file_name='biblio3.json';
```

Until Connect 1.5:

```
CREATE TABLE jsample2 (
ISBN CHAR(15),
Lng CHAR(2) field_format='LANG',
json_Author CHAR(255) field_format='AUTHOR:*',
Title CHAR(32) field_format='TITLE',
YEAR INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=JSON file_name='biblio3.json';
```

Now the query:

```
SELECT json_Author FROM jsample2;
```

will return and display :

json_Author
[{"FIRSTNAME":"Jean-Christophe","LASTNAME":"Bernadac"},{"FIRSTNAME":"François","LASTNAME":"Knab"}]
[{"FIRSTNAME":"William J.","LASTNAME":"Pardi"}]

Note: Prefixing the column name by *json_* is optional but is useful when using the column as argument to Connect UDF functions, making it to be surely recognized as valid Json without aliasing.

This also works on input, a column specified so that it can be directly set to a valid JSON string.

This feature is of great value as we will see below.

Create, Read, Update and Delete Operations on JSON Tables

The SQL commands INSERT, UPDATE and DELETE are fully supported for JSON tables except those returned by REST queries. For INSERT and UPDATE, if the target values are simple values, there are no problems.

However, there are some issues when the added or modified values are objects or arrays.

Concerning objects, the same problems exist that we have already seen with the XML type. The added or modified object will have the format described in the table definition, which can be different from the one of the JSON file. Modifications should be done using a file specifying the full path of modified objects.

New problems are raised when trying to modify the values of an array. Only updates can be done on the original table. First of all, for the values of the array to be distinct values, all update operations concerning array values must be done using a table expanding this array.

For instance, to modify the authors of the `biblio.json` based table, the `jsampex` table must be used. Doing so, updating and deleting authors is possible using standard SQL commands. For example, to change the first name of Knab from François to John:

```
UPDATE jsampex SET authorfn = 'John' WHERE authorln = 'Knab';
```

However It would be wrong to do:

```
UPDATE jsampex SET authorfn = 'John' WHERE isbn = '9782212090819';
```

Because this would change the first name of both authors as they share the same ISBN.

Where things become more difficult is when trying to delete or insert an author of a book. Indeed, a delete command will delete the whole book and an insert command will add a new complete row instead of adding a new author in the same array. Here we are penalized by the SQL language that cannot give us a way to specify this. Something like:

```
UPDATE jsampex ADD authorfn = 'Charles', authorln = 'Dickens'
WHERE title = 'XML en Action';
```

However this does not exist in SQL. Does this mean that it is impossible to do it? No, but it requires us to use a table specified on the same file but adapted to this task. One way to do it is to specify a table for which the authors are no more an expanded array. Supposing we want to add an author to the “XML en Action” book. We will do it on a table containing just the author(s) of that book, which is the second book of the table.

From Connect 1.6:

```
CREATE TABLE jauthor (
FIRSTNAME CHAR(64),
LASTNAME CHAR(64))
ENGINE=CONNECT table_type=JSON File_name='biblio3.json' option_list='Object=1.AUTHOR';
```

Until Connect 1.5

```
CREATE TABLE jauthor (
FIRSTNAME CHAR(64),
LASTNAME CHAR(64))
ENGINE=CONNECT table_type=JSON File_name='biblio3.json' option_list='Object=[1]:AUTHOR';
```

The command:

```
SELECT * FROM jauthor;
```

replies:

FIRSTNAME	LASTNAME
William J.	Pardi

It is a standard JSON table that is an array of objects in which we can freely insert or delete rows.

```
INSERT INTO jauthor VALUES('Charles','Dickens');
```

We can check that this was done correctly by:

```
SELECT * FROM jsampex;
```

This will display:

ISBN	Title	AuthorFN	AuthorLN	Year
9782212090819	Construire une application XML	Jean-Christophe	Bernadac	1999
9782212090819	Construire une application XML	John	Knab	1999
9782840825685	XML en Action	William J.	Pardi	1999
9782840825685	XML en Action	Charles	Dickens	1999

Note: If this table were a big table with many books, it would be difficult to know what the order of a specific book is in the table. This can be found by adding a special ROWID column in the table.

However, an alternate way to do it is by using direct JSON column representation as in the `JSAMPLE2` table. This can be done by:

```
UPDATE jsample2 SET json_Author =
'[{"FIRSTNAME":"William J.","LASTNAME":"Pardi"},
  {"FIRSTNAME":"Charles","LASTNAME":"Dickens"}]'
WHERE isbn = '9782840825685';
```

Here, we didn't have to find the index of the sub array to modify. However, this is not quite satisfying because we had to manually write the whole JSON value to set to the json_Author column.

Therefore we need specific functions to do so. They are introduced now.

JSON User Defined Functions

Although such functions written by other parties do exist,[2] CONNECT provides its own UDFs that are specifically adapted to the JSON table type and easily available because, being inside the CONNECT library or DLL, they require no additional module to be loaded (see CONNECT - Compiling JSON UDFs in a Separate Library to make these functions in a separate library module).

Here is the list of the CONNECT functions; more can be added if required.

Name	Type	Return	Description	Added
jbin_array	Function	STRING*	Make a JSON array containing its arguments.	MariaDB 10.1.9
jbin_array_add	Function	STRING*	Adds to its first array argument its second arguments.	MariaDB 10.1.9

Name	Type	Return	Description	Added
jbin_array_add_values	Function	STRING*	Adds to its first array argument all following arguments.	
jbin_array_delete	Function	STRING*	Deletes the nth element of its first array argument.	MariaDB 10.1.9
jbin_file	Function	STRING*	Returns of a (json) file contain.	MariaDB 10.1.9
jbin_get_item	Function	STRING*	Access and returns a json item by a JPATH key.	MariaDB 10.1.9
jbin_insert_item	Function	STRING	Insert item values located to paths.	
jbin_item_merge	Function	STRING*	Merges two arrays or two objects.	MariaDB 10.1.9
jbin_object	Function	STRING*	Make a JSON object containing its arguments.	MariaDB 10.1.9
jbin_object_nonull	Function	STRING*	Make a JSON object containing its not null arguments.	MariaDB 10.1.9
jbin_object_add	Function	STRING*	Adds to its first object argument its second argument.	MariaDB 10.1.9
jbin_object_delete	Function	STRING*	Deletes the nth element of its first object argument.	MariaDB 10.1.9
jbin_object_key	Function	STRING*	Make a JSON object for key/value pairs.	
jbin_object_list	Function	STRING*	Returns the list of object keys as an array.	MariaDB 10.1.9
jbin_set_item	Function	STRING	Set item values located to paths.	
jbin_update_item	Function	STRING	Update item values located to paths.	
jfile_bjson	Function	STRING	Convert a pretty=0 file to another BJson file.	MariaDB 10.5.9, MariaDB 10.4.18, MariaDB 10.3.28, MariaDB 10.2.36
jfile_convert	Function	STRING	Convert a Json file to another pretty=0 file.	MariaDB 10.5.9, MariaDB 10.4.18, MariaDB 10.3.28, MariaDB 10.2.36
jfile_make	Function	STRING	Make a json file from its json item first argument.	MariaDB 10.1.9
json_array	Function	STRING	Make a JSON array containing its arguments.	MariaDB 10.0.17 until Connect 1.5
json_array_add	Function	STRING	Adds to its first array argument its second arguments (before MariaDB 10.1.9, all following arguments).	
json_array_add_values	Function	STRING	Adds to its first array argument all following arguments.	MariaDB 10.1.9
json_array_delete	Function	STRING	Deletes the nth element of its first array argument.	
json_array_grp	Aggregate	STRING	Makes JSON arrays from coming argument.	
json_file	Function	STRING	Returns the contains of (json) file.	MariaDB 10.1.9

Name	Type	Return	Description	Added
json_get_item	Function	STRING	Access and returns a json item by a JPATH key.	MariaDB 10.1.9
json_insert_item	Function	STRING	Insert item values located to paths.	
json_item_merge	Function	STRING	Merges two arrays or two objects.	MariaDB 10.1.9
json_locate_all	Function	STRING	Returns the JPATH's of all occurrences of an element.	MariaDB 10.1.9
json_make_array	Function	STRING	Make a JSON array containing its arguments.	From Connect 1.6
json_make_object	Function	STRING	Make a JSON object containing its arguments.	From Connect 1.6
json_object	Function	STRING	Make a JSON object containing its arguments.	MariaDB 10.0.17 until Connect 1.5
json_object_delete	Function	STRING	Deletes the nth element of its first object argument.	MariaDB 10.1.9
json_object_grp	Aggregate	STRING	Makes JSON objects from coming arguments.	
json_object_list	Function	STRING	Returns the list of object keys as an array.	MariaDB 10.1.9
json_object_nonull	Function	STRING	Make a JSON object containing its not null arguments.	
json_serialize	Function	STRING	Serializes the return of a "Jbin" function.	MariaDB 10.1.9
json_set_item	Function	STRING	Set item values located to paths.	
json_update_item	Function	STRING	Update item values located to paths.	
jsonvalue	Function	STRING	Make a JSON value from its unique argument. Called json_value until MariaDB 10.0.22 and MariaDB 10.1.8.	MariaDB 10.0.17
jsoncontains	Function	INTEGER	Returns 0 or 1 if an element is contained in the document.	
jsoncontains_path	Function	INTEGER	Returns 0 or 1 if a JPATH is contained in the document.	
jsonget_string	Function	STRING	Access and returns a string element by a JPATH key.	MariaDB 10.1.9
jsonget_int	Function	INTEGER	Access and returns an integer element by a JPATH key.	MariaDB 10.1.9
jsonget_real	Function	REAL	Access and returns a real element by a JPATH key.	MariaDB 10.1.9
jsonlocate	Function	STRING	Returns the JPATH to access one element.	MariaDB 10.1.9

String values are mapped to JSON strings. These strings are automatically escaped to conform to the JSON syntax. The automatic escaping is bypassed when the value has an alias beginning with 'json_'. This is automatically the case when a JSON UDF argument is another JSON UDF whose name begins with "json_" (not case sensitive). This is why all functions that do not return a Json item are not prefixed by "json_".

Argument string values, for some functions, can alternatively be json file names. When this is ambiguous, alias them as *jfile_*. Full path should be used because UDF functions has no means to know what the current database is. Apparently, when the file name path is not full, it is based on the MariaDB data directory but I am not sure it is always true.

Numeric values are (big) integers, double floating point values or decimal values. Decimal values are character strings containing a numeric representation and are treated as strings. Floating point values contain a decimal point and/or an exponent. Integers are written without decimal points.

To install these functions execute the following commands :[3]

Note

Json function names are often written on this page with leading upper case letters for clarity. It is possible to do so in SQL queries because function names are case insensitive. However, when creating or dropping them, their names must match the case they are in the library module, which is in lower case.

On Unix systems (from Connect 1.7.02):

```
CREATE FUNCTION jsonvalue RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_make_array RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array_add_values RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_make_object RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_nonull RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_key RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_list RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_values RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonset_grp_size RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION jsonget_grp_size RETURNS INTEGER soname 'ha_connect.so';
CREATE AGGREGATE FUNCTION json_array_grp RETURNS STRING soname 'ha_connect.so';
CREATE AGGREGATE FUNCTION json_object_grp RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonlocate RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_locate_all RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsoncontains RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION jsoncontains_path RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION json_item_merge RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_get_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonget_string RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonget_int RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION jsonget_real RETURNS REAL soname 'ha_connect.so';
CREATE FUNCTION json_set_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_insert_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_update_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_file RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jfile_make RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jfile_convert RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jfile_bjson RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_serialize RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array_add_values RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_nonull RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_key RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_list RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_item_merge RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_get_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_set_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_insert_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_update_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_file RETURNS STRING soname 'ha_connect.so';
```

On Unix systems (from Connect 1.6):

```
CREATE FUNCTION jsonvalue RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_make_array RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array_add_values RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_make_object RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_nonull RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_key RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_list RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonset_grp_size RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION jsonget_grp_size RETURNS INTEGER soname 'ha_connect.so';
CREATE AGGREGATE FUNCTION json_array_grp RETURNS STRING soname 'ha_connect.so';
CREATE AGGREGATE FUNCTION json_object_grp RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonlocate RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_locate_all RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsoncontains RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION jsoncontains_path RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION json_item_merge RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_get_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonget_string RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonget_int RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION jsonget_real RETURNS REAL soname 'ha_connect.so';
CREATE FUNCTION json_set_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_insert_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_update_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_file RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jfile_make RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_serialize RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array_add_values RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_nonull RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_key RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_list RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_item_merge RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_get_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_set_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_insert_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_update_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_file RETURNS STRING soname 'ha_connect.so';
```

On Unix systems (until Connect 1.5):

```
CREATE FUNCTION jsonvalue RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array_add_values RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_array_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_nonull RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_key RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_object_list RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonset_grp_size RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION jsonget_grp_size RETURNS INTEGER soname 'ha_connect.so';
CREATE AGGREGATE FUNCTION json_array_grp RETURNS STRING soname 'ha_connect.so';
CREATE AGGREGATE FUNCTION json_object_grp RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonlocate RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_locate_all RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsoncontains RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION jsoncontains_path RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION json_item_merge RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_get_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonget_string RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jsonget_int RETURNS INTEGER soname 'ha_connect.so';
CREATE FUNCTION jsonget_real RETURNS REAL soname 'ha_connect.so';
CREATE FUNCTION json_set_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_insert_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_update_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_file RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jfile_make RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION json_serialize RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array_add_values RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_array_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_nonull RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_key RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_add RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_delete RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_object_list RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_item_merge RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_get_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_set_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_insert_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_update_item RETURNS STRING soname 'ha_connect.so';
CREATE FUNCTION jbin_file RETURNS STRING soname 'ha_connect.so';
```

On WIndows (from Connect 1.7.02):

```
CREATE FUNCTION jsonvalue RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_make_array RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array_add_values RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_make_object RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_nonull RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_key RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_list RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_values RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonset_grp_size RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION jsonget_grp_size RETURNS INTEGER soname 'ha_connect';
CREATE AGGREGATE FUNCTION json_array_grp RETURNS STRING soname 'ha_connect';
CREATE AGGREGATE FUNCTION json_object_grp RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonlocate RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_locate_all RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsoncontains RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION jsoncontains_path RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION json_item_merge RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_get_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonget_string RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonget_int RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION jsonget_real RETURNS REAL soname 'ha_connect';
CREATE FUNCTION json_set_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_insert_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_update_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_file RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jfile_make RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jfile_convert RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jfile_bjson RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_serialize RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array_add_values RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_nonull RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_key RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_list RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_item_merge RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_get_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_set_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_insert_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_update_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_file RETURNS STRING soname 'ha_connect';
```

On WIndows (from Connect 1.6):

```
CREATE FUNCTION jsonvalue RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_make_array RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array_add_values RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_make_object RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_nonull RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_key RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_list RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonset_grp_size RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION jsonget_grp_size RETURNS INTEGER soname 'ha_connect';
CREATE AGGREGATE FUNCTION json_array_grp RETURNS STRING soname 'ha_connect';
CREATE AGGREGATE FUNCTION json_object_grp RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonlocate RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_locate_all RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsoncontains RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION jsoncontains_path RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION json_item_merge RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_get_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonget_string RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonget_int RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION jsonget_real RETURNS REAL soname 'ha_connect';
CREATE FUNCTION json_set_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_insert_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_update_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_file RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jfile_make RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_serialize RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array_add_values RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_nonull RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_key RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_list RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_item_merge RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_get_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_set_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_insert_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_update_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_file RETURNS STRING soname 'ha_connect';
```

On WIndows (until Connect 1.5):

```
CREATE FUNCTION jsonvalue RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array_add_values RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_array_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_nonull RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_key RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_object_list RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonset_grp_size RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION jsonget_grp_size RETURNS INTEGER soname 'ha_connect';
CREATE AGGREGATE FUNCTION json_array_grp RETURNS STRING soname 'ha_connect';
CREATE AGGREGATE FUNCTION json_object_grp RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonlocate RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_locate_all RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsoncontains RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION jsoncontains_path RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION json_item_merge RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_get_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonget_string RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jsonget_int RETURNS INTEGER soname 'ha_connect';
CREATE FUNCTION jsonget_real RETURNS REAL soname 'ha_connect';
CREATE FUNCTION json_set_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_insert_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_update_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_file RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jfile_make RETURNS STRING soname 'ha_connect';
CREATE FUNCTION json_serialize RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array_add_values RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_array_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_nonull RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_key RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_add RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_delete RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_object_list RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_item_merge RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_get_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_set_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_insert_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_update_item RETURNS STRING soname 'ha_connect';
CREATE FUNCTION jbin_file RETURNS STRING soname 'ha_connect';
```

Jfile_Bjson

MariaDB starting with 10.5.9

JFile_Bjson was introduced in MariaDB.

```
Jfile_Bjson(in_file_name, out_file_name, lrecl)
```

Converts the first argument pretty=0 json file to Bjson file. B(inary)json is a pre-parsed json format. It is described below in the Performance chapter (available in next Connect versions).

Jfile_Convert

MariaDB starting with 10.5.9

JFile_Convert was introduced in MariaDB.

```
Jfile_Convert(in_file_name, out_file_name, lrecl)
```

Converts the first argument json file to another *pretty=0* json file. The third integer argument is the record length to use. This is often required to process huge json files that would be very slow if they were in *pretty=2* format.

This is done without completely parsing the file, is very fast and requires no big memory.

Jfile_Make

Jfile_Make was added in CONNECT 1.4

```
Jfile_Make(arg1, arg2, [arg3], …)
```

The first argument must be a json item (if it is just a string, Jfile_Make will try its best to see if it is a json item or an input file name). The following arguments are a string file name and an integer pretty value (defaulting to 2) in any order. This function creates a json file containing the first argument item.

The returned string value is the created file name. If not specified as an argument, the file name can in some cases be retrieved from the first argument; in such cases the file itself is modified.

This function can be used to create or format a json file. For instance, supposing we want to format the file tb.json, this can be done with the query:

```
SELECT Jfile_Make('tb.json' jfile_, 2);
```

The tb.json file are changed to:

```
[
  {
    "_id": 5,
    "type": "food",
    "ratings": [
      5,
      8,
      9
    ]
  },
  {
    "_id": 6,
    "type": "car",
    "ratings": [
      5,
      9
    ]
  }
]
```

Json_Array_Add

```
Json_Array_Add(arg1, arg2, [arg3][, arg4][, ...])
```

Note: The following describes this function for CONNECT version 1.4 only. The first argument must be a JSON array. The second argument is added as member of this array:

```
SELECT Json_Array_Add(Json_Array(56,3.1416,'machin',NULL),
'One more') ARRAY;
```

Array
[56,3.141600,“machin”,null,“One more”]

Note: The first array is not escaped, its (alias) name beginning with ‘json_’.

Now we can see how adding an author to the JSAMPLE2 table can alternatively be done:

```
UPDATE jsample2 SET
  json_author = json_array_add(json_author, json_object('Charles' FIRSTNAME, 'Dickens' LASTNAME))
  WHERE isbn = '9782840825685';
```

Note: Calling a column returning JSON a name prefixed by json_ (like json_author here) is good practice and removes the need to give it an alias to prevent escaping when used as an argument.

Additional arguments:
If a third integer argument is given, it specifies the position (zero based) of the added value:

```
SELECT Json_Array_Add('[5,3,8,7,9]' json_, 4, 2) ARRAY;
```

Array
[5,3,4,8,7,9]

If a string argument is added, it specifies the Json path to the array to be modified. For instance:

```
SELECT Json_Array_Add('{"a":1,"b":2,"c":[3,4]}' json_, 5, 1, 'c');
```

Json_Array_Add('{"a":1,"b":2,"c":[3, 4]}' json_, 5, 1, 'c')
{"a":1,"b":2,"c":[3,5,4]}

Json_Array_Add_Values

Json_Array_Add_Values added in CONNECT 1.4 replaces the function Json_Array_Add of CONNECT version 1.3.

```
Json_Array_Add_Values(arg, arglist)
```

The first argument must be a JSON array string. Then all other arguments are added as members of this array:

```
SELECT Json_Array_Add_Values
  (Json_Array(56, 3.1416, 'machin', NULL), 'One more', 'Two more') ARRAY;
```

Array
[56,3.141600,"machin",null,"One more","Two more"]

Json_Array_Delete

```
Json_Array_Delete(arg1, arg2 [,arg3] [...])
```

The first argument should be a JSON array. The second argument is an integer indicating the rank (0 based conforming to general json usage) of the element to delete:

```
SELECT Json_Array_Delete(Json_Array(56,3.1416,'foo',NULL),1) ARRAY;
```

Array
[56,"foo",null]

Now we can see how to delete the second author from the JSAMPLE2 table:

```
UPDATE jsample2 SET json_author = json_array_delete(json_author, 1)
  WHERE isbn = '9782840825685';
```

A Json path can be specified as a third string argument

Json_Array_Grp

```
Json_Array_Grp(arg)
```

This is an aggregate function that makes an array filled from values coming from the rows retrieved by a query. Let us suppose we have the pet table:

name	race	number
John	dog	2
Bill	cat	1
Mary	dog	1
Mary	cat	1
Lisbeth	rabbit	2
Kevin	cat	2
Kevin	bird	6
Donald	dog	1
Donald	fish	3

The query:

```
SELECT name, json_array_grp(race) FROM pet GROUP BY name;
```

will return:

name
Bill
Donald
John
Kevin
Lisbeth
Mary

One problem with the JSON aggregate functions is that they construct their result in memory and cannot know the needed amount of storage, not knowing the number of rows of the used table.

Therefore, the number of values for each group is limited. This limit is the value of JsonGrpSize whose default value is 10 but can be set using the JsonSet_Grp_Size function. Nevertheless, working on a larger table is possible, but only after setting JsonGrpSize to the ceiling of the number of rows per group for the table. Try not to set it to a very large value to avoid memory exhaustion.

JsonContains

```
JsonContains(json_doc, item [, int])<
```

This function can be used to check whether an item is contained in a document. Its arguments are the same than the ones of the JsonLocate function; only the return value changes. The integer returned value is 1 is the item is contained in the document or 0 otherwise.

JsonContains_Path

```
JsonContains_Path(json_doc, path)
```

This function can be used to check whether a Json path is contained in the document. The integer returned value is 1 is the path is contained in the document or 0 otherwise.

Json_File

```
Json_File(arg1, [arg2, [arg3]], …)
```

The first argument must be a file name. This function returns the text of the file that is supposed to be a json file. If only one argument is specified, the file text is returned without being parsed. Up to two additional arguments can be specified:

A string argument is the path to the sub-item to be returned. An integer argument specifies the pretty format value of the file.

This function is chiefly used to get the json item argument of other json functions from a json file. For instance, supposing the file tb.json is:

```
{ "_id" : 5, "type" : "food", "ratings" : [ 5, 8, 9 ] }
{ "_id" : 6, "type" : "car", "ratings" : [ 5, 9 ] }
```

Extracting a value from it can be done with a query such as:

```
SELECT JsonGet_String(Json_File('tb.json', 0), '$[1].type') "Type";
```

This query returns:

Type
car

However, we'll see that, most of the time, it is better to use Jbin_File or to directly specify the file name in queries. In particular this function should not be used for queries that must modify the json item because, even if the modified json is returned, the file itself would be unchanged.

Json_Get_Item

Json_Get_Item was added in CONNECT 1.4.

```
Json_Get_Item(arg1, arg2, …)
```

This function returns a subset of the json document passed as first argument. The second argument is the json path of the item to be returned and should be one returning a json item (terminated by a '*'). If not, the function will try to make it right but this is not foolproof. For instance:

```
SELECT Json_Get_Item(Json_Object('foo' AS "first", Json_Array('a', 33)
  AS "json_second"), 'second') AS "item";
```

The correct path should have been 'second.*'), but in this simple case the function was able to make it right. The returned item:

item
["a",33]

Note: The array is aliased “json_second” to indicate it is a json item and avoid escaping it. However, the “json_” prefix is skipped when making the object and must not be added to the path.

JsonGet_Grp_Size

```
JsonGet_Grp_Size(val)
```

This function returns the JsonGrpSize value.

JsonGet_String / JsonGet_Int / JsonGet_Real

JsonGet_String, JsonGet_Int and JsonGet_Real were added in CONNECT 1.4.

```
JsonGet_String(arg1, arg2, [arg3] …)
JsonGet_Int(arg1, arg2, [arg3] …)
JsonGet_Real(arg1, arg2, [arg3] …)
```

The first argument should be a JSON item. If it is a string with no alias, it are converted as a json item. The second argument is the path of the item to be located in the first argument and returned, eventually converted according to the used function:

```
SELECT
JsonGet_String('{"qty":7,"price":29.50,"garanty":null}','price') "String",
JsonGet_Int('{"qty":7,"price":29.50,"garanty":null}','price') "Int",
JsonGet_Real('{"qty":7,"price":29.50,"garanty":null}','price') "Real";
```

This query returns:

String	Int	Real
29.50	29	29.500000000000000

The function *JsonGet_Real* can be given a third argument to specify the number of decimal digits of the returned value. For instance:

```
SELECT
JsonGet_Real('{"qty":7,"price":29.50,"garanty":null}','price',4) "Real";
```

This query returns:

String
29.50

The given path can specify all operators for arrays except the “expand” [*] operator). For instance:

```
SELECT
JsonGet_Int(Json_Array(45,28,36,45,89), '[4]') "Rank",
JsonGet_Int(Json_Array(45,28,36,45,89), '[#]') "Number",
JsonGet_String(Json_Array(45,28,36,45,89), '[","]') "Concat",
JsonGet_Int(Json_Array(45,28,36,45,89), '[+]') "Sum",
JsonGet_Real(Json_Array(45,28,36,45,89), '[!]', 2) "Avg";
```

The result:

Rank	Number	Concat	Sum	Avg
89	5	45,28,36,45,89	243	48.60

Json_Item_Merge

```
Json_Item_Merge(arg1, arg2, …)
```

This function merges two arrays or two objects. For arrays, this is done by adding to the first array all the values of the second array. For instance:

```
SELECT Json_Item_Merge(Json_Array('a','b','c'), Json_Array('d','e','f')) AS "Result";
```

The function returns:

Result
[“a”,“b”,“c”,“d”,“e”,“f”]

For objects, the pairs of the second object are added to the first object if the key does not yet exist in it; otherwise the pair of the first object is set with the value of the matching pair of the second object. For instance:

```
SELECT Json_Item_Merge(Json_Object(1 "a", 2 "b", 3 "c"), Json_Object(4 "d",5 "b",6 "f"))
  AS "Result";
```

The function returns:

Result
{“a”:1,“b”:5,“c”:3,“d”:4,“f”:6}

JsonLocate

```
JsonLocate(arg1, arg2, [arg3], …):
```

The first argument must be a JSON tree. The second argument is the item to be located. The item to be located can be a constant or a json item. Constant values must be equal in type and value to be found. This is “shallow equality” – strings, integers and doubles won’t match.

This function returns the json path to the located item or null if it is not found:

```
SELECT JsonLocate('{"AUTHORS":[{"FN":"Jules", "LN":"Verne"},
  {"FN":"Jack", "LN":"London"}]}' json_, 'Jack') PATH;
```

This query returns:

Path
$.AUTHORS[1].FN

The path syntax is the same used in JSON CONNECT tables.

By default, the path of the first occurrence of the item is returned. The third parameter can be used to specify the occurrence whose path is to be returned. For instance:

```
SELECT
JsonLocate('[45,28,[36,45],89]',45) FIRST,
JsonLocate('[45,28,[36,45],89]',45,2) SECOND,
JsonLocate('[45,28,[36,45],89]',45.0) `wrong type`,
JsonLocate('[45,28,[36,45],89]','[36,45]' json_) JSON;
```

first	second	wrong type	json
$[0]	$[2][1]		$[2]

For string items, the comparison is case sensitive by default. However, it is possible to specify a string to be compared case insensitively by giving it an alias beginning by “ci”:

```
SELECT JsonLocate('{"AUTHORS":[{"FN":"Jules", "LN":"Verne"},
  {"FN":"Jack", "LN":"London"}]}' json_, 'VERNE' ci) PATH;
```

Path
$.AUTHORS[0].LN

Json_Locate_All

```
Json_Locate_All(arg1, arg2, [arg3], …):
```

The first argument must be a JSON item. The second argument is the item to be located. This function returns the paths to all locations of the item as an array of strings:

```
SELECT Json_Locate_All('[[45,28],[[36,45],89]]',45);
```

This query returns:

All paths
[“$[0][0]”,“$[1][0][1]”]

The returned array can be applied other functions. For instance, to get the number of occurrences of an item in a json tree, you can do:

```
SELECT JsonGet_Int(Json_Locate_All('[[45,28],[[36,45],89]]',45), '$[#]') "Nb of occurs";
```

The displayed result:

Nb of occurs
2

If specified, the third integer argument set the depth to search in the document. This means the maximum items in the paths. This value defaults to 10 but can be increased for complex documents or reduced to set the maximum wanted depth of the returned paths.

Json_Make_Array

```
Json_Make_Array(val1, …, valn)
```

Json_Make_Array returns a string denoting a JSON array with all its arguments as members:

```
SELECT Json_Make_Array(56, 3.1416, 'My name is "Foo"', NULL);
```

Json_Make_Array(56, 3.1416, ‘My name is “Foo”’,N ULL)
[56,3.141600,“My name is”Foo”“,null]

Note: The argument list can be void. If so, a void array is returned.

Json_Make_Object

```
Json_Make_Object(arg1, …, argn)
```

Json_Make_Object returns a string denoting a JSON object. For instance:

```
SELECT Json_Make_Object(56, 3.1416, 'machin', NULL);
```

The object is filled with pairs corresponding to the given arguments. The key of each pair is made from the argument (default or specified) alias.

Json_Make_Object(56, 3.1416, 'machin', NULL)
{"56":56,"3.1416":3.141600,"machin":"machin","NULL":null}

When needed, it is possible to specify the keys by giving an alias to the arguments:

```
SELECT Json_Make_Object(56 qty, 3.1416 price, 'machin' truc, NULL garanty);
```

Json_Make_Object(56 qty,3.1416 price,'machin' truc, NULL garanty)
{"qty":56,"price":3.141600,"truc":"machin","garanty":null}

If the alias is prefixed by 'json_' (to prevent escaping) the key name is stripped from that prefix.

This function is chiefly useful when entering values retrieved from a table, the key being by default the column name:

```
SELECT Json_Make_Object(matricule, nom, titre, salaire) FROM connect.employe WHERE nom = 'PANTIER';
```

Json_Make_Object(matricule, nom, titre, salaire)
{"matricule":40567,"nom":"PANTIER","titre":"DIRECTEUR","salaire":14000.000000}

Json_Object_Add

```
Json_Object_Add(arg1, arg2, [arg3] …)
```

The first argument must be a JSON object. The second argument is added as a pair to this object:

```
SELECT Json_Object_Add
  ('{"item":"T-shirt","qty":27,"price":24.99}' json_old,'blue' color) newobj;
```

newobj
{"item":"T-shirt","qty":27,"price":24.990000,"color":"blue"}

Note: If the specified key already exists in the object, its value is replaced by the new one.

The third string argument is a Json path to the target object.

Json_Object_Delete

```
Json_Object_Delete(arg1, arg2, [arg3] …):
```

The first argument must be a JSON object. The second argument is the key of the pair to delete:

```
SELECT Json_Object_Delete('{"item":"T-shirt","qty":27,"price":24.99}' json_old, 'qty') newobj;
```

newobj
{"item":"T-shirt","price":24.99}

The third string argument is a Json path to the object to be the target of deletion.

Json_Object_Grp

```
Json_Object_Grp(arg1,arg2)
```

This function works like Json_Array_Grp. It makes a JSON object filled with value pairs whose keys are passed from its first argument and values are passed from its second argument.

This can be seen with the query:

```
SELECT name, json_object_grp(NUMBER,race) FROM pet GROUP BY name;
```

This query returns:

name	json_object_grp(number,race)
Bill	{“cat”:1}
Donald	{“dog”:1,“fish”:3}
John	{“dog”:2}
Kevin	{“cat”:2,“bird”:6}
Lisbeth	{“rabbit”:2}
Mary	{“dog”:1,“cat”:1}

Json_Object_Key

```
Json_Object_Key([key1, val1 [, …, keyn, valn]])
```

Return a string denoting a JSON object. For instance:

```
SELECT Json_Object_Key('qty', 56, 'price', 3.1416, 'truc', 'machin', 'garanty', NULL);
```

The object is filled with pairs made from each key/value arguments.

Json_Object_Key(‘qty’, 56, ‘price’, 3.1416, ‘truc’, ‘machin’, ‘garanty’, NULL)
{“qty”:56,“price”:3.141600,“truc”:“machin”,“garanty”:null}

Json_Object_List

```
Json_Object_List(arg1, …):
```

The first argument must be a JSON object. This function returns an array containing the list of all keys existing in the object:

```
SELECT Json_Object_List(Json_Object(56 qty,3.1416 price,'machin' truc, NULL garanty))
  "Key List";
```

Key List
[“qty”,“price”,“truc”,“garanty”]

Json_Object_Nonull

```
Json_Object_Nonull(arg1, …, argn)
```

This function works like Json_Make_Object but “null” arguments are ignored and not inserted in the object. Arguments are regarded as “null” if they are JSON null values, void arrays or objects, or arrays or objects containing

only null members.

It is mainly used to avoid constructing useless null items when converting tables (see later).

Json_Object_Values

```
Json_Object_Values(json_object)
```

The first argument must be a JSON object. This function returns an array containing the list of all values existing in the object:

```
SELECT Json_Object_Values('{"One":1,"Two":2,"Three":3}') "Value List";
```

Value List
[1,2,3]

JsonSet_Grp_Size

```
JsonSet_Grp_Size(val)
```

This function is used to set the JsonGrpSize value. This value is used by the following aggregate functions as a ceiling value of the number of items in each group. It returns the JsonGrpSize value that can be its default value when passed 0 as argument.

Json_Set_Item / Json_Insert_Item / Json_Update_Item

```
Json_{Set | Insert | Update}_Item(json_doc, [item, path [, val, path …]])
```

These functions insert or update data in a JSON document and return the result. The value/path pairs are evaluated left to right. The document produced by evaluating one pair becomes the new value against which the next pair is evaluated.

- Json_Set_Item replaces existing values and adds non-existing values.
- Json_Insert_Item inserts values without replacing existing values.
- Json_Update_Item replaces only existing values.

Example:

```
SET @j = Json_Array(1, 2, 3, Json_Object_Key('quatre', 4));
SELECT Json_Set_Item(@j, 'foo', '$[1]', 5, '$[3].cinq') AS "Set",
Json_Insert_Item(@j, 'foo', '$[1]', 5, '$[3].cinq') AS "Insert",
Json_Update_Item(@j, 'foo', '$[1]', 5, '$[3].cinq') AS "Update";
```

This query returns:

Set	Insert	Update
[1,“foo”,3,{“quatre”:4,“cinq”:5}]	[1,2,3,{“quatre”:4,“cinq”:5}]	[1,“foo”,3,{“quatre”:4}]

JsonValue

```
JsonValue (val)
```

Returns a JSON value as a string, for instance:

```
SELECT JsonValue(3.1416);
```

JsonValue(3.1416)
3.141600

The “JBIN” Return Type

Almost all functions returning a json string - whose name begins with *Json_* - have a counterpart with a name beginning with *Jbin_*. This is both for performance (speed and memory) as well as for better control of what the functions should do.

This is due to the way CONNECT UDFs work internally. The Json functions, when receiving json strings as parameters, parse them and construct a binary tree in memory. They work on this tree and before returning; serialize this tree to return a new json string.

If the json document is large, this can take up a large amount of time and storage space. It is all right when one simple json function is called – it must be done anyway – but is a waste of time and memory when json functions are used as parameters to other json functions.

To avoid multiple serializing and parsing, the Jbin functions should be used as parameters to other functions. Indeed, they do not serialize the memory document tree, but return a structure allowing the receiving function to have direct access to the memory tree. This saves the serialize-parse steps otherwise needed to pass the argument and removes the need to reallocate the memory of the binary tree, which by the way is 6 to 7 times the size of the json string. For instance:

```
SELECT Json_Object(Jbin_Array_Add(Jbin_Array('a','b','c'), 'd') AS "Jbin_foo") AS "Result";
```

This query returns:

Result
{“foo”:[“a”,“b”,“c”,“d”]}

Here the binary json tree allocated by *Jbin_Array* is completed by *Jbin_Array_Add* and *Json_Object* and serialized only once to make the final result string. It would be serialized and parsed two more times if using “Json” functions.

Note that Jbin results are recognized as such because they are aliased beginning with “Jbin_”. This is why in the *Json_Object* function the alias is specified as “Jbin_foo”.

What happens if it is not recognized as such? These functions are declared as returning a string and to take care of this, the returned structure begins with a zero-terminated string. For instance:

```
SELECT Jbin_Array('a','b','c');
```

This query replies:

Jbin_Array(‘a’,‘b’,‘c’)
Binary Json array

Note: When testing, the tree returned by a “Jbin” function can be seen using the *Json_Serialize* function whose unique parameter must be a “Jbin” result. For instance:

```
SELECT Json_Serialize(Jbin_Array('a','b','c'));
```

This query returns:

Json_Serialize(Jbin_Array(‘a’,‘b’,‘c’))
[“a”,“b”,“c”]

Note: For this simple example, this is equivalent to using the *Json_Array* function.

Using a File as JSON UDF First Argument

We have seen that many json UDFs can have an additional argument not yet described. This is in the case where the json item argument was referring to a file. Then the additional integer argument is the pretty value of the json file. It matters only when the first argument is just a file name (to make the UDF understand this argument is a file name, it should be aliased with a name beginning with jfile_) or if the function modifies the file, in which case it are rewritten with this pretty format.

The json item is created by extracting the required part from the file. This can be the whole file but more often only some of it. There are two ways to specify the sub-item of the file to be used:

1. Specifying it in the Json_File or Jbin_File arguments.
2. Specifying it in the receiving function (not possible for all functions).

It doesn't make any difference when the *Jbin_File* is used but it does with *Json_File*. For instance:

```
SELECT Jfile_Make('{"a":1, "b":[44, 55]}' json_, 'test.json');
SELECT Json_Array_Add(Json_File('test.json', 'b'), 66);
```

The second query returns:

Json_Array_Add(Json_File('test.json', 'b'), 66)
[44,55,66]

It just returns the – modified – subset returned by the Json_File function, while the query:

```
SELECT Json_Array_Add(Json_File('test.json'), 66, 'b');
```

returns what was received from *Json_File* with the modification made on the subset.

Json_Array_Add(Json_File('test.json'), 66, 'b')
{"a":1,"b":[44,55,66]}

Note that in both case the test.json file is not modified. This is because the *Json_File* function returns a string representing all or part of the file text but no information about the file name. This is all right to check what would be the effect of the modification to the file.

However, to have the file modified, use the *Jbin_File* function or directly give the file name. *Jbin_File* returns a structure containing the file name, a pointer to the file parsed tree and eventually a pointer to the subset when a path is given as a second argument:

```
SELECT Json_Array_Add(Jbin_File('test.json', 'b'), 66);
```

This query returns:

Json_Array_Add(Jbin_File('test.json', 'b'), 66)
test.json

This time the file is modified. This can be checked with:

```
SELECT Json_File('test.json', 3);
```

Json_File('test.json', 3)
{"a":1,"b":[44,55,66]}

The reason why the first argument is returned by such a query is because of tables such as:

```
CREATE TABLE tb (
n INT KEY,
jfile_cols CHAR(10) NOT NULL);
INSERT INTO tb VALUES(1,'test.json');
```

In this table, the *jfile_cols* column just contains a file name. If we update it by:

```
UPDATE tb SET jfile_cols = SELECT Json_Array_Add(Jbin_File('test.json', 'b'), 66)
WHERE n = 1;
```

This is the test.json file that must be modified, not the jfile_cols column. This can be checked by:

```
SELECT JsonGet_String(jfile_cols, '[1]:*') FROM tb;
```

JsonGet_String(jfile_cols, '[1]:*')
{“a”:1,“b”:[44,55,66]}

Note: It was an important facility to name the second column of the table beginning by “jfile_” so the json functions knew it was a file name without obliging to specify an alias in the queries.

Using “Jbin” to Control the Query Execution

This is applying in particular when acting on json files. We have seen that a file was not modified when using the *Json_File* function as an argument to a modifying function because the modifying function just received a copy of the json file. This is not true when using the *Jbin_File* function that does not serialize the binary document and make it directly accessible. Also, as we have seen earlier, json functions that modify their first file parameter modify the file and return the file name. This is done by directly serializing the internal binary document as a file.

However, the “Jbin” counterpart of these functions does not serialize the binary document and thus does not modify the json file. For example let us compare these two queries:

/* First query */

```
SELECT Json_Object(Jbin_Object_Add(Jbin_File('bt2.json'), 4 AS "d") AS "Jbin_bt1")
  AS "Result";
```

/* Second query */

```
SELECT Json_Object(Json_Object_Add(Jbin_File('bt2.json'), 4 AS "d") AS "Jfile_bt1")
  AS "Result";
```

Both queries return:

Result
{“bt1”:{“a”:1,“b”:2,“c”:3,“d”:4}}

In the first query *Jbin_Object_Add* does not serialize the document (no “Jbin” functions do) and *Json_Object* just returns a serialized modified tree. Consequently, the file bt2.json is not modified. This query is all right to copy a modified version of the json file without modifying it.

However, in the second query *Json_Object_Add* does modify the json file and returns the file name. The *Json_Object* function receives this file name, reads and parses the file, makes an object from it and returns the serialized result. This modification can be done willingly but can be an unwanted side effect of the query.

Therefore, using “Jbin” argument functions, in addition to being faster and using less memory, are also safer when dealing with json files that should not be modified.

Using JSON as Dynamic Columns

The JSON nosql language has all the features to be used as an alternative to dynamic columns. For instance, take the following example of dynamic columns:

```
create table assets (
   item_name varchar(32) primary key, /* A common attribute for all items */
   dynamic_cols  blob  /* Dynamic columns are stored here */
 );

INSERT INTO assets VALUES
   ('MariaDB T-shirt', COLUMN_CREATE('color', 'blue', 'size', 'XL'));

INSERT INTO assets VALUES
   ('Thinkpad Laptop', COLUMN_CREATE('color', 'black', 'price', 500));

SELECT item_name, COLUMN_GET(dynamic_cols, 'color' as char) AS color FROM assets;
+-----------------+-------+
| item_name       | color |
+-----------------+-------+
| MariaDB T-shirt | blue  |
| Thinkpad Laptop | black |
+-----------------+-------+
```

/* Remove a column: */

```
UPDATE assets SET dynamic_cols=COLUMN_DELETE(dynamic_cols, "price")
  WHERE COLUMN_GET(dynamic_cols, 'color' AS CHAR)='black';
```

/* Add a column: */

```
UPDATE assets SET dynamic_cols=COLUMN_ADD(dynamic_cols, 'warranty', '3 years')
   WHERE item_name='Thinkpad Laptop';
```

/* You can also list all columns, or get them together with their values in JSON format: */

```
SELECT item_name, column_list(dynamic_cols) FROM assets;
+-----------------+--------------------------+
| item_name       | column_list(dynamic_cols) |
+-----------------+--------------------------+
| MariaDB T-shirt | `size`,`color`            |
| Thinkpad Laptop | `color`,`warranty`        |
+-----------------+--------------------------+

SELECT item_name, COLUMN_JSON(dynamic_cols) FROM assets;
+-----------------+---------------------------------------+
| item_name       | COLUMN_JSON(dynamic_cols)             |
+-----------------+---------------------------------------+
| MariaDB T-shirt | {"size":"XL","color":"blue"}          |
| Thinkpad Laptop | {"color":"black","warranty":"3 years"} |
+-----------------+---------------------------------------+
```

The same result can be obtained with json columns using the json UDF’s:

/* JSON equivalent */

```
create table jassets (
   item_name varchar(32) primary key, /* A common attribute for all items */
   json_cols varchar(512)  /* Jason columns are stored here */
 );

INSERT INTO jassets VALUES
   ('MariaDB T-shirt', Json_Object('blue' color, 'XL' size));

INSERT INTO jassets VALUES
   ('Thinkpad Laptop', Json_Object('black' color, 500 price));

SELECT item_name, JsonGet_String(json_cols, 'color') AS color FROM jassets;
+-----------------+-------+
| item_name       | color |
+-----------------+-------+
| MariaDB T-shirt | blue  |
| Thinkpad Laptop | black |
+-----------------+-------+
```

/* Remove a column: */

```
UPDATE jassets SET json_cols=Json_Object_Delete(json_cols, 'price')
 WHERE JsonGet_String(json_cols, 'color')='black';
```

/* Add a column */

```
UPDATE jassets SET json_cols=Json_Object_Add(json_cols, '3 years' warranty)
 WHERE item_name='Thinkpad Laptop';
```

/* You can also list all columns, or get them together with their values in JSON format: */

```
SELECT item_name, Json_Object_List(json_cols) FROM jassets;
+-----------------+-----------------------------+
| item_name       | Json_Object_List(json_cols) |
+-----------------+-----------------------------+
| MariaDB T-shirt | ["color","size"]            |
| Thinkpad Laptop | ["color","warranty"]        |
+-----------------+-----------------------------+

SELECT item_name, json_cols FROM jassets;
+-----------------+---------------------------------------+
| item_name       | json_cols                             |
+-----------------+---------------------------------------+
| MariaDB T-shirt | {"color":"blue","size":"XL"}          |
| Thinkpad Laptop | {"color":"black","warranty":"3 years"} |
+-----------------+---------------------------------------+
```

However, using JSON brings features not existing in dynamic columns:

- Use of a language used by many implementation and developers.
- Full support of arrays, currently missing from dynamic columns.
- Access of subpart of json by JPATH that can include calculations on arrays.
- Possible references to json files.

With more experience, additional UDFs can be easily written to support new needs.

New Set of BSON Functions

All these functions have been rewritten using the new JSON handling way and are temporarily available changing the J starting name to B. Then Json_Make_Array new style is called using Bson_Make_Array.
Some, such as Bson_Item_Delete, are new and some fix bugs found in their Json counterpart.

Converting Tables to JSON

The JSON UDF's and the direct Jpath "*" facility are powerful tools to convert table and files to the JSON format. For instance, the file `biblio3.json` we used previously can be obtained by converting the `xsample.xml file`. This can be done like this:

From Connect 1.07.0002

```
CREATE TABLE xj1 (ROW VARCHAR(500) jpath='*') ENGINE=CONNECT table_type=JSON file_name='biblio3.json'
option_list='jmode=2';
```

Before Connect 1.07.0002

```
CREATE TABLE xj1 (ROW VARCHAR(500) field_format='*')
 ENGINE=CONNECT table_type=JSON file_name='biblio3.json' option_list='jmode=2';
```

And then :

```
INSERT INTO xj1
  SELECT json_object_nonull(ISBN, LANGUAGE LANG, SUBJECT,
    json_array_grp(json_object(authorfn FIRSTNAME, authorln LASTNAME)) json_AUTHOR, TITLE,
    json_object(translated PREFIX, json_object(tranfn FIRSTNAME, tranln LASTNAME) json_TRANSLATOR)
    json_TRANSLATED, json_object(publisher NAME, LOCATION PLACE) json_PUBLISHER, DATE DATEPUB)
FROM xsampall2 GROUP BY isbn;
```

The xj1 table rows will directly receive the Json object made by the select statement used in the insert statement and the table file are made as shown (xj1 is pretty=2 by default) Its mode is Jmode=2 because the values inserted are strings even if they denote json objects.

Another way to do this is to create a table describing the file format we want before the `biblio3.json` file existed:

From Connect 1.07.0002

```
CREATE TABLE jsampall3 (
ISBN CHAR(15),
LANGUAGE CHAR(2) jpath='LANG',
SUBJECT CHAR(32),
AUTHORFN CHAR(128) jpath='AUTHOR:[X]:FIRSTNAME',
AUTHORLN CHAR(128) jpath='AUTHOR:[X]:LASTNAME',
TITLE CHAR(32),
TRANSLATED CHAR(32) jpath='TRANSLATOR:PREFIX',
TRANSLATORFN CHAR(128) jpath='TRANSLATOR:FIRSTNAME',
TRANSLATORLN CHAR(128) jpath='TRANSLATOR:LASTNAME',
PUBLISHER CHAR(20) jpath='PUBLISHER:NAME',
LOCATION CHAR(20) jpath='PUBLISHER:PLACE',
DATE INT(4) jpath='DATEPUB')
ENGINE=CONNECT table_type=JSON file_name='biblio3.json';
```

Before Connect 1.07.0002

```
CREATE TABLE jsampall3 (
ISBN CHAR(15),
LANGUAGE CHAR(2) field_format='LANG',
SUBJECT CHAR(32),
AUTHORFN CHAR(128) field_format='AUTHOR:[X]:FIRSTNAME',
AUTHORLN CHAR(128) field_format='AUTHOR:[X]:LASTNAME',
TITLE CHAR(32),
TRANSLATED CHAR(32) field_format='TRANSLATOR:PREFIX',
TRANSLATORFN CHAR(128) field_format='TRANSLATOR:FIRSTNAME',
TRANSLATORLN CHAR(128) field_format='TRANSLATOR:LASTNAME',
PUBLISHER CHAR(20) field_format='PUBLISHER:NAME',
LOCATION CHAR(20) field_format='PUBLISHER:PLACE',
DATE INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=JSON file_name='biblio3.json';
```

and to populate it by:

```
INSERT INTO jsampall3 SELECT * FROM xsampall;
```

This is a simpler method. However, the issue is that this method cannot handle the multiple column values. This is why we inserted from `xsampall` not from `xsampall2`. How can we add the missing multiple authors in this table? Here again we must create a utility table able to handle JSON strings.
From Connect 1.07.0002

```
CREATE TABLE xj2 (ISBN CHAR(15), author VARCHAR(150) jpath='AUTHOR:*') ENGINE=CONNECT table_type=JSON
file_name='biblio3.json' option_list='jmode=1';
```

Before Connect 1.07.0002

```
CREATE TABLE xj2 (ISBN CHAR(15), author VARCHAR(150) field_format='AUTHOR:*')
  ENGINE=CONNECT table_type=JSON file_name='biblio3.json' option_list='jmode=1';
```

```
UPDATE xj2 SET author =
(SELECT json_array_grp(json_object(authorfn FIRSTNAME, authorln LASTNAME))
  FROM xsampall2 WHERE isbn = xj2.isbn);
```

Voilà !

Converting JSON Files

We have seen that json files can be formatted differently depending on the pretty option. In particular, big data files should be formatted with pretty equal to 0 when used by a CONNECT json table. The best and simplest way to convert a file from one format to another is to use the *Jfile_Make* function. Indeed this function makes a file of specified format using the syntax:

```
Jfile_Make(json_document, [file_name], [pretty]);
```

The file name is optional when the json document comes from a Jbin_File function because the returned structure makes it available. For instance, to convert back the json file tb.json to pretty= 0, this can be simply done by:

```
SELECT Jfile_Make(Jbin_File('tb.json'), 0);
```

Performance Consideration

MySQL and PostgreSQL have a JSON data type that is not just text but an internal encoding of JSON data. This is to save parsing time when executing JSON functions. Of course, the parse must be done anyway when creating the data and serializing must be done to output the result.

CONNECT directly works on character strings impersonating JSON values with the need of parsing them all the time but with the advantage of working easily on external data. Generally, this is not too penalizing because JSON data are often of some or reasonable size. The only case where it can be a serious problem is when working on a big JSON file.

Then, the file should be formatted or converted to *pretty=0*.

From Connect 1.7.002, this easily done using the Jfile_Convert function, for instance:

```
SELECT jfile_convert('bibdoc.json','bibdoc0.json',350);
```

Such a json file should not be used directly by JSON UDFs because they parse the whole file, even when only a subset is used. Instead, it should be used by a JSON table created on it. Indeed, JSON tables do not parse the whole document but just the item corresponding to the row they are working on. In addition, indexing can be used by the table as explained previously on this page.

Generally speaking, the maximum flexibility offered by CONNECT is by using JSON tables and JSON UDFs together. Some things are better handled by tables, other by UDFs. The tools are there but it is up to you to discover the best way to resolve your problems.

Bjson Files

Starting with Connect 1.7.002, *pretty=0* json files can be converted to a binary format that is a pre-parsed representation of json. This can be done with the Jfile_Bjson UDF function, for instance:

```
SELECT jfile_bjson('bigfile.json','binfile.json',3500);
```

Here the third argument, the record length, must 6 to 10 times larger than the lrecl of the initial json file because the parsed representation is bigger than the original json text representation.

Tables using such Bjson files must specify ‘Pretty=-1’ in the option list.

It is probably similar to the BSON used by MongoDB and PostgreSQL and permits to process queries up to 10 times faster than working on text json files. Indexing is also available for tables using this format making even more performance improvement. For instance, some queries on a json table of half a million rows, that were previously done in more than 10 seconds, took only 0.1 second when converted and indexed.

Here again, this has been remade to use the new way Json is handled. The files made using the bfile_bjson function are only from two to four times the size of the source files. This new representation is not compatible with the old one. Therefore, these files must be used with BSON tables only.

Specifying a JSON Table Encoding

An important feature of JSON is that strings should in UNICODE. As a matter of fact, all examples we have found on the Internet seemed to be just ASCII. This is because UNICODE is generally encoded in JSON files using UTF8 or UTF16 or UTF32.

To specify the required encoding, just use the data_charset CONNECT option or the native DEFAULT CHARSET option.

Retrieving JSON Data From MongoDB

Classified as a NoSQL database program, MongoDB uses JSON-like documents (BSON) grouped in collections. The simplest way, and only method available before Connect 1.6, to access MongoDB data was to export a collection to a JSON file. This produces a file having the pretty=0 format. Viewed as SQL, a collection is a table and documents are table rows.

Since CONNECT version 1.6, it is now possible to directly access MongoDB collections via their MongoDB C Driver. This is the purpose of the MONGO table type described later. However, JSON tables can also do it in a somewhat different way (providing MONGO support is installed as described for MONGO tables).

It is achieved by specifying the MongoDB connection URI while creating the table. For instance:

From Connect 1.7.002

```
CREATE OR REPLACE TABLE jinvent (
_id CHAR(24) NOT NULL,
item CHAR(12) NOT NULL,
instock VARCHAR(300) NOT NULL jpath='instock.*')
ENGINE=CONNECT table_type=JSON tabname='inventory' lrecl=512
CONNECTION='mongodb://localhost:27017';
```

Before Connect 1.7.002

```
CREATE OR REPLACE TABLE jinvent (
_id CHAR(24) NOT NULL,
item CHAR(12) NOT NULL,
instock VARCHAR(300) NOT NULL field_format='instock.*')
ENGINE=CONNECT table_type=JSON tabname='inventory' lrecl=512
CONNECTION='mongodb://localhost:27017';
```

In this statement, the *file_name* option was replaced by the *connection* option. It is the URI enabling to retrieve data from a local or remote MongoDB server. The *tabname* option is the name of the MongoDB collection that are used and the *dbname* option could have been used to indicate the database containing the collection (it defaults to the current database).

The way it works is that the documents retrieved from MongoDB are serialized and CONNECT uses them as if they were read from a file. This implies serializing by MongoDB and parsing by CONNECT and is not the best performance wise. CONNECT tries its best to reduce the data transfer when a query contains a reduced column list and/or a where clause. This way makes all the possibilities of the JSON table type available, such as calculated arrays.

However, to work on large JSON collations, using the MONGO table type is generally the normal way.

Note: JSON tables using the MongoDB access accept the specific MONGO options colist, filter and pipeline. They are described in the MONGO table chapter.

Summary of Options and Variables Used With JSON Tables

Options and variables that can be used when creating Json tables are listed here:

Table Option	Type	Description
ENGINE	String	Must be specified as CONNECT.
TABLE_TYPE	String	Must be JSON or BSON.
FILE_NAME	String	The optional file (path) name of the Json file. Can be absolute or relative to the current data directory. If not specified, it defaults to the table name and json file type.
DATA_CHARSET	String	Set it to 'utf8' for most Unicode Json documents.
LRECL	Number	The file record size for pretty < 2 json files.
HTTP	String	The HTTP of the server of REST queries.
URI	String	THE URI of REST queries

Table Option	Type	Description
CONNECTION*	String	Specifies a connection to MONGODB.
ZIPPED	Boolean	True if the json file(s) is/are zipped in one or several zip files.
MULTIPLE	Number	Used to specify a multiple file table.
SEP_CHAR	String	Set it to ':' for old tables using the old json path syntax.
CATFUNC	String	The catalog function (column) used when creating a catalog table.
OPTION_LIST	String	Used to specify all other options listed below.

(*) For Json tables connected to MongoDB, Mongo specific options can also be used.

Other options must be specified in the option list:

Table Option	Type	Description
DEPTHLEVEL	Number	Specifies the depth in the document CONNECT looks when defining columns by discovery or in catalog tables
PRETTY	Number	Specifies the format of the Json file (-1 for Bjson files)
EXPAND	String	The name of the column to expand.
OBJECT	String	The json path of the sub-document used for the table.
BASE	Number	The numbering base for arrays: 0 (the default) or 1.
LIMIT	Number	The maximum number of array values to use when concatenating, calculating or expanding arrays. Defaults to 50 (>= Connect 1.7.0003), 10 (<= Connect 1.7.0002).
FULLARRAY	Boolean	Used when creating with Discovery. Make a column for each value of arrays (up to LIMIT).
JMODE	Number	The Json mode (array of objects, array of arrays, or array of values) Only used when inserting new rows.
ACCEPT	Boolean	Keep null columns (for discovery).
AVGLEN	Number	An estimate average length of rows. This is used only when indexing and can be set if indexing fails by miscalculating the table max size.
STRINGIFY	String	Ask discovery to make a column to return the Json representation of this object.

Column options:

Column Option	Type	Description
JPATHFIELD_FORMAT	String	Defaults to the column name.
DATE_FORMAT	String	Specifies the date format into the Json file when defining a DATE, DATETIME or TIME column.

Variables used with Json tables are:

- connect_default_depth
- connect_json_null
- connect_json_all_path

- connect_force_bson

Notes

1. ↑ The value n can be 0 based or 1 based depending on the base table option. The default is 0 to match what is the current usage in the Json world but it can be set to 1 for tables created in old versions.
2. ↑ See for instance: json-functions, lib_mysqludf_json#readme and json_udf_functions_version_04
3. ↑ This will not work when CONNECT is compiled embedded

This page is licensed: CC BY-SA / Gnu FDL

CONNECT MONGO Table Type: Accessing Collections from MongoDB

Classified as a NoSQL database program, MongoDB uses JSON-like documents (BSON) grouped in collections. The MONGO type is used to directly access MongoDB collections as tables.

Accessing MongDB from CONNECT

Accessing MongoDB from CONNECT can be done in different ways:

1. As a MONGO table via the MongoDB C Driver.
2. As a MONGO table via the MongoDB Java Driver.
3. As a JDBC table using some commercially available MongoDB JDBC drivers.
4. As a JSON table via the MongoDB C or Java Driver.

Using the MongoDB C Driver

This is currently not available from binary distributions but only for versions compiled from source. The preferred version of the MongoDB C Driver is 1.7, because they provide package recognition. What must be done is:

1. Install libbson and the MongoDB C Driver 1.7.
2. Configure, compile and install MariaDB.

With earlier versions of the Mongo C Driver, the additional include directories and libraries will have to be specified manually when compiling.

When possible, this is the preferred means of access because it does not require all the Java path settings etc. and is faster than using the Java driver.

Using the Mongo Java Driver

This is possible with all distributions including JDBC support, or compiling from source. With a binary distribution that does not enable the MONGO table type, it is possible to access MongoDB using an OEM module. See CONNECT OEM Table Example for details. The only additional things to do are:

1. Install the MongoDB Java Driver by downloading its jar file. Several versions are available. If possible use the latest version 3 one.
2. Add the path to it in the CLASSPATH environment variable or in the connect_class_path variable. This is like what is done to declare JDBC drivers.

Connection is established by new Java wrappers Mongo3Interface and Mongo2Interface. They are available in a JDBC distribution in the Mongo2.jar and Mongo3.jar files (previously JavaWrappers.jar). If version 2 of the Java Driver is used, specify “Version=2” in the option list when creating tables.

Using JDBC

See the documentation of the existing commercial JDBC Mongo drivers.

Using JSON

See the specific chapter of the JSON Table Type.

The following describes the MONGO table type.

CONNECT MONGO Tables

Creating and running MONGO tables requires a connection to a running local or remote MongoDB server.

A MONGO table is defined to access a MongoDB collection. The table rows are the collection documents. For instance, to create a table based on the MongoDB sample collection restaurants, you can do something such as the following:

```
CREATE TABLE resto (
_id VARCHAR(24) NOT NULL,
name VARCHAR(64) NOT NULL,
cuisine CHAR(200) NOT NULL,
borough CHAR(16) NOT NULL,
restaurant_id VARCHAR(12) NOT NULL)
ENGINE=CONNECT table_type=MONGO tabname='restaurants'
data_charset=utf8 CONNECTION='mongodb://localhost:27017';
```

Note: The used driver is by default the C driver if only the MongoDB C Driver is installed and the Java driver if only the MongoDB Java Driver is installed. If both are available, it can be specified by the DRIVER option to be specified in the option list and defaults to C.

Here we did not define all the items of the collection documents but only those that are JSON values. The database is test by default. The connection value is the URI used to establish a connection to a local or remote MongoDB server. The value shown in this example corresponds to a local server started with its default port. It is the default connection value for MONGO tables so we could have omit specifying it.

Using discovery is available. This table could have been created by:

```
CREATE TABLE resto
ENGINE=CONNECT table_type=MONGO tabname='restaurants'
data_charset=utf8 option_list='level=-1';
```

Here “depth=-1” is used to create only columns that are simple values (no array or object). Without this, with the default value “depth=0” the table had been created as:

```
CREATE TABLE `resto` (
  `_id` CHAR(24) NOT NULL,
  `address` VARCHAR(136) NOT NULL,
  `borough` CHAR(13) NOT NULL,
  `cuisine` CHAR(64) NOT NULL,
  `grades` VARCHAR(638) NOT NULL,
  `name` CHAR(98) NOT NULL,
  `restaurant_id` CHAR(8) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='MONGO' `TABNAME`='restaurants'
`DATA_CHARSET`='utf8';
```

Fixing Problems With mariadb-dump

In some case or some platforms, when CONNECT is set up for use with JDBC table types, this causes mariadb-dump with the –all-databases option to fail.

This was reported by Robert Dyas who found the cause of it and how to fix it (see MDEV-11238).

This occurs when the Java JRE “Usage Tracker” is enabled. In that case, Java creates a directory #mysql50#.oracle_jre_usage in the mysql data directory that shows up as a database but cannot be accessed via MySQL Workbench nor apparently backed up by mariadb-dump –all-databases.

Per the Oracle documentation () the
“Usage Tracker” is disabled by default. It is enabled only when creating the properties file /lib/management/usagetracker.properties. This turns out to be WRONG on some platforms as the file does exist by default on a new installation, and the existence of this file enables the usage tracker.

The solution on CentOS 7 with the Oracle JVM is to rename or delete the usagetracker.properties file (to disable it) and then delete the bogus folder it created in the mysql database directory, then restart.

For example, the following works:

```
sudo mv /usr/java/default/jre/lib/management/management.properties
/usr/java/default/jre/lib/management/management.properties.TRACKER-OFF
sudo reboot
sudo rm -rf  /var/lib/mysql/.oracle_jre_usage
sudo reboot
```

In this collection, the address column is a JSON object and the column grades is a JSON array. Unlike the JSON table, just specifying the column name with no Jpath result in displaying the JSON representation of them. For instance:

```
SELECT name, address FROM resto LIMIT 3;
```

name	address
Morris Park Bake Shop	{“building”:“1007”,“coord”:[-73.8561,40.8484], “street”:“Morris ParkAve”, “zipcode”:“10462”}
Wendy’S	{“building”:“469”,“coord”:[-73.9617,40.6629], “street”:“Flatbush Avenue”, “zipcode”:“11225”}
Reynolds Restaurant	{“building”:“351”,“coord”:[-73.9851,40.7677], “street”:“West 57Street”, “zipcode”:“10019”}

MongoDB Dot Notation

To address the items inside object or arrays, specify the Jpath in MongoDB syntax (if using Discovery, specify the Depth option accordingly):

From Connect 1.7.0002

```
CREATE TABLE newresto (
_id VARCHAR(24) NOT NULL,
name VARCHAR(64) NOT NULL,
cuisine CHAR(200) NOT NULL,
borough CHAR(16) NOT NULL,
street VARCHAR(65) jpath='address.street',
building CHAR(16) jpath='address.building',
zipcode CHAR(5) jpath='address.zipcode',
grade CHAR(1) jpath='grades.0.grade',
score INT(4) NOT NULL jpath='grades.0.score',
`date` DATE jpath='grades.0.date',
restaurant_id VARCHAR(255) NOT NULL)
ENGINE=CONNECT table_type=MONGO tabname='restaurants'
data_charset=utf8 CONNECTION='mongodb://localhost:27017';
```

Before Connect 1.7.0002

```
CREATE TABLE newresto (
_id VARCHAR(24) NOT NULL,
name VARCHAR(64) NOT NULL,
cuisine CHAR(200) NOT NULL,
borough CHAR(16) NOT NULL,
street VARCHAR(65) field_format='address.street',
building CHAR(16) field_format='address.building',
zipcode CHAR(5) field_format='address.zipcode',
grade CHAR(1) field_format='grades.0.grade',
score INT(4) NOT NULL field_format='grades.0.score',
`date` DATE field_format='grades.0.date',
restaurant_id VARCHAR(255) NOT NULL)
ENGINE=CONNECT table_type=MONGO tabname='restaurants'
data_charset=utf8 CONNECTION='mongodb://localhost:27017';
```

If this is not done, the Oracle JVM will start the usage tracker, which will create the hidden folder .oracle_jre_usage in the mysql home directory, which will cause a mariadb-dump of the server to fail.

```
SELECT name, street, score, DATE FROM newresto LIMIT 5;
```

name	street	score	date
Morris Park Bake Shop	Morris Park Ave	2	03/03/2014
Wendy'S	Flatbush Avenue	8	30/12/2014
Dj Reynolds Pub And Restaurant	West 57 Street	2	06/09/2014
Riviera Caterer	Stillwell Avenue	5	10/06/2014
Tov Kosher Kitchen	63 Road	20	24/11/2014

MONGO Specific Options

The MongoDB syntax for Jpath does not allow the CONNECT specific items on arrays. The same effect can still be obtained by a different way. For this, additional options are used when creating MONGO tables.

Option	Type	Description
Colist	String	Options to pass to the MongoDB cursor.
Filter	String	Query used by the MongoDB cursor.
Pipeline*	Boolean	If True, Colist is a pipeline.
Fullarray*	Boolean	Used when creating with Discovery.
Driver*	String	C or Java.
Version*	Integer	The Java Driver version (defaults to 3)

- : To be specified in the option list.

Note: For the content of these options, refer to the MongoDB documentation.

Colist Option

Used to pass different options when making the MongoDB cursor used to retrieve the collation documents. One of them is the projection, allowing to limit the items retrieved in documents. It is hardly useful because this limitation is made automatically by CONNECT. However, it can be used when using discovery to eliminate the _id (or another) column when you are not willing to keep it:

```
CREATE TABLE restest
ENGINE=CONNECT table_type=MONGO tabname='restaurants'
data_charset=utf8 option_list='depth=-1'
colist='{"projection":{"_id":0},"limit":5}';
```

In this example, we added another cursor option, the limit option that works like the limit SQL clause.

This additional option works only with the C driver. When using the Java driver, colist should be:

```
colist='{"_id":0}';
```

And limit would be specified with select statements.

Note: When used with a JSON table, to specify the projection list (or 'all' to get all columns) makes JPATH to be Connect Json paths, not MongoDB ones, allowing JPATH options not available to MongoDB.

Filter Option

This option is used to specify a "filter" that works as a where clause on the table. Supposing we want to create a table restricted to the restaurant making English cuisine that are not located in the Manhattan borough, we can do it by:

```
CREATE TABLE english
ENGINE=CONNECT table_type=MONGO tabname='restaurants'
data_charset=utf8
colist='{"projection":{"cuisine":0}}'
filter='{"cuisine":"English","borough":{"$ne":"Manhattan"}}'
option_list='Depth=-1';
```

And if we ask:

```
SELECT * FROM english;
```

This query will return:

_id	borough	name	restaurant_id
58ada47de5a51ddfcd5ee1f3	Brooklyn	The Park Slope Chipshop	40816202
58ada47de5a51ddfcd5ee999	Brooklyn	Chip Shop	41076583
58ada47ee5a51ddfcd5f13d5	Brooklyn	The Monro	41660253
58ada47ee5a51ddfcd5f176e	Brooklyn	Dear Bushwick	41690534
58ada47ee5a51ddfcd5f1e91	Queens	Snowdonia Pub	50000290

Pipeline Option

When this option is specified as true (by YES or 1) the Colist option contains a MongoDB pipeline applying to the table collation. This is a powerful mean for doing things such as expanding arrays like we do with JSON tables. For instance:

```
CREATE TABLE resto2 (
name VARCHAR(64) NOT NULL,
borough CHAR(16) NOT NULL,
DATE DATETIME NOT NULL,
grade CHAR(1) NOT NULL,
score INT(4) NOT NULL)
ENGINE=CONNECT table_type=MONGO tabname='restaurants' data_charset=utf8
colist='{"pipeline":[{"$match":{"cuisine":"French"}},{"$unwind":"$grades"},{"$project":
{"_id":0,"name":1,"borough":1,"date":"$grades.date","grade":"$grades.grade","score":"$grades.score"}}]}'
option_list='Pipeline=1';
```

In this pipeline "$match" is an early filter, "$unwind" means that the grades array are expanded (one Document for each array values) and "$project" eliminates the _id and cuisine columns and gives the Jpath for the date, grade and score columns.

```
SELECT name, grade, score, DATE FROM resto2
WHERE borough = 'Bronx';
```

This query replies:

name	grade	score	date
Bistro Sk	A	10	21/11/2014 01:00:00
Bistro Sk	A	12	19/02/2014 01:00:00
Bistro Sk	B	18	12/06/2013 02:00:00

This make possible to get things like we do with JSON tables:

```
SELECT name, AVG(score) average FROM resto2
GROUP BY name HAVING average >= 25;
```

Can be used to get the average score inside the grades array.

name	average
Bouley Botanical	25,0000
Cheri	46,0000
Graine De Paris	30,0000
Le Pescadeux	29,7500

Fullarray Option

This option, like the Depth option, is only interpreted when creating a table with Discovery (meaning not specifying the columns). It tells CONNECT to generate a column for all existing values in the array. For instance, let us see the MongoDB collection tar by:

From Connect 1.7.0002

```
CREATE TABLE seetar (
Collection VARCHAR(300) NOT NULL jpath='*')
ENGINE=CONNECT table_type=MONGO tabname=tar;
```

Before Connect 1.7.0002

```
CREATE TABLE seetar (
Collection VARCHAR(300) NOT NULL field_format='*')
ENGINE=CONNECT table_type=MONGO tabname=tar;
```

The format '*' indicates we want to see the Json documents. This small collection is:

Collection
{"_id":{"$oid":"58f63a5099b37d9c930f9f3b"},"item":"journal","prices":[87.0,45.0,63.0,12.0,78.0]}
{"_id":{"$oid":"58f63a5099b37d9c930f9f3c"},"item":"notebook","prices":[123.0,456.0,789.0]}

The Fullarray option can be used here to generate enough columns to see all the prices of the document prices array.

```
CREATE TABLE tar
ENGINE=CONNECT table_type=MONGO
colist='{"projection":{"_id":0}}'
option_list='depth=1,Fullarray=YES';
```

The table has been created as:

From Connect 1.7.0002

```
CREATE TABLE `tar` (
  `item` CHAR(8) NOT NULL,
  `prices_0` DOUBLE(12,6) NOT NULL `JPATH`='prices.0',
  `prices_1` DOUBLE(12,6) NOT NULL `JPATH`='prices.1',
  `prices_2` DOUBLE(12,6) NOT NULL `JPATH`='prices.2',
  `prices_3` DOUBLE(12,6) DEFAULT NULL `JPATH`='prices.3',
  `prices_4` DOUBLE(12,6) DEFAULT NULL `JPATH`='prices.4'
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='MONGO' `COLIST`='{"projection":{"_id":0}}'
`OPTION_LIST`='depth=1,Fullarray=YES';
```

Before Connect 1.7.0002

```
CREATE TABLE `tar` (
  `item` CHAR(8) NOT NULL,
  `prices_0` DOUBLE(12,6) NOT NULL `FIELD_FORMAT`='prices.0',
  `prices_1` DOUBLE(12,6) NOT NULL `FIELD_FORMAT`='prices.1',
  `prices_2` DOUBLE(12,6) NOT NULL `FIELD_FORMAT`='prices.2',
  `prices_3` DOUBLE(12,6) DEFAULT NULL `FIELD_FORMAT`='prices.3',
  `prices_4` DOUBLE(12,6) DEFAULT NULL `FIELD_FORMAT`='prices.4'
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='MONGO' `COLIST`='{"projection":{"_id":0}}'
`OPTION_LIST`='level=1,Fullarray=YES';
```

And is displayed as:

item	prices_0	prices_1	prices_2	prices_3	prices_4
journal	87.00	45.00	63.00	12.00	78.00
notebook	123.00	456.00	789.00	NULL	NULL

Create, Read, Update and Delete Operations

All modifying operations are supported. However, inserting into arrays must be done in a specific way. Like with the Fullarray option, we must have enough columns to specify the array values. For instance, we can create a new table by:

From Connect 1.7.0002

```
CREATE TABLE testing (
n INT NOT NULL,
m CHAR(12) NOT NULL,
surname CHAR(16) NOT NULL jpath='person.name.first',
name CHAR(16) NOT NULL jpath='person.name.last',
age INT(3) NOT NULL jpath='person.age',
price_1 DOUBLE(8,2) jpath='d.0',
price_2 DOUBLE(8,2) jpath='d.1',
price_3 DOUBLE(8,2) jpath='d.2')
ENGINE=CONNECT table_type=MONGO tabname='tin'
CONNECTION='mongodb://localhost:27017';
```

Before Connect 1.7.0002

```
CREATE TABLE testing (
n INT NOT NULL,
m CHAR(12) NOT NULL,
surname CHAR(16) NOT NULL field_format='person.name.first',
name CHAR(16) NOT NULL field_format='person.name.last',
age INT(3) NOT NULL field_format='person.age',
price_1 DOUBLE(8,2) field_format='d.0',
price_2 DOUBLE(8,2) field_format='d.1',
price_3 DOUBLE(8,2) field_format='d.2')
ENGINE=CONNECT table_type=MONGO tabname='tin'
CONNECTION='mongodb://localhost:27017';
```

Now it is possible to populate it by:

```
INSERT INTO testing VALUES
(1789, 'Welcome', 'Olivier','Bertrand',56, 3.14, 2.36, 8.45),
(1515, 'Hello', 'John','Smith',32, 65.17, 98.12, NULL),
(2014, 'Coucou', 'Foo','Bar',20, -1.0, 74, 81356);
```

The result are:

n	m	surname	name	age	price_1	price_2	price_3
1789	Welcome	Olivier	Bertrand	56	3,14	2,36	8,45
1515	Hello	John	Smith	32	65,17	98,12	NULL
2014	Coucou	Foo	Bar	20	-1	74	81356

Note: If the collection does not exist yet when creating the table and inserting in it, MongoDB creates it automatically.

It can be updated by queries such as:

```
UPDATE tintin SET price_3 = 83.36 WHERE n = 2014;
```

To look how the array is generated, let us create another table:

From Connect 1.7.0002

```
CREATE TABLE tintin (
n INT NOT NULL,
name CHAR(16) NOT NULL jpath='person.name.first',
prices VARCHAR(255) jpath='d')
ENGINE=CONNECT table_type=MONGO tabname='tin';
```

Before Connect 1.7.002

```
CREATE TABLE tintin (
n INT NOT NULL,
name CHAR(16) NOT NULL field_format='person.name.first',
prices VARCHAR(255) field_format='d')
ENGINE=CONNECT table_type=MONGO tabname='in';
```

This table is displayed as:

From Connect 1.7.0002

n	name	prices
1789	Olivier	[3.1400000000000001243,2.3599999999999998757,8.4499999999999992895]
1515	John	[65.170000000000001705,98.120000000000004547,null]
2014	Foo	[null,74.0,83.359999999999999432]

Before Connect 1.7.002

n	name	prices
1789	Olivier	[3.14, 2.36, 8.45]
1515	John	[65.17, 98.12]
2014	Foo	[, 74.0, 83.36]

Note: This last table can be used to make array calculations like with JSON tables using the JSON UDF functions. For instance:

```
SELECT name, jsonget_real(prices,'[+]') sum_prices, jsonget_real(prices,'[!]') avg_prices FROM tintin;
```

This query returns:

name	sum_prices	avg_prices
Olivier	13.95	4.65
John	163.29	81.64
Foo	157,36	78.68

Note: When calculating on arrays, null values are ignored.

Status of MONGO Table Type

This table type is still under development. It has significant advantages over the JSON type to access MongoDB collections. Firstly, the access being direct, tables are always up to date whether the collection has been modified by another application. Performance wise, it can be faster than JSON, because most processing is done by MongoDB on BSON, its internal representation of JSON data, which is designed to optimize all operations. Note that using the MongoDB C Driver can be faster than using the MongoDB Java Driver.

Current Restrictions

- Option “CATFUNC=tables” is not implemented yet.
- Options SRCDEF and EXECSRC do not apply to MONGO tables.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT MYSQL Table Type: Accessing MySQL/MariaDB Tables

This table type uses libmysql API to access a MySQL or MariaDB table or view. This table must be created on the current server or on another local or remote server. This is similar to what the FederatedX storage engine provides with some differences.

Currently the Federated-like syntax can be used to create such a table, for instance:

```
CREATE TABLE essai (
  num INTEGER(4) NOT NULL,
  line CHAR(15) NOT NULL)
ENGINE=CONNECT table_type=MYSQL
CONNECTION='mysql://root@localhost/test/people';
```

The connection string can have the same syntax as that used by FEDERATED

```
scheme://username:password@hostname:port/database/tablename
scheme://username@hostname/database/tablename
scheme://username:password@hostname/database/tablename
scheme://username:password@hostname/database/tablename
```

However, it can also be mixed with connect standard options. For instance:

```
CREATE TABLE essai (
  num INTEGER(4) NOT NULL,
  line CHAR(15) NOT NULL)
ENGINE=CONNECT table_type=MYSQL dbname=test tabname=people
CONNECTION='mysql://root@localhost';
```

It can also be specified as a reference to a federated server:

```
connection="connection_one"
connection="connection_one/table_foo"
```

The pure (deprecated) CONNECT syntax is also accepted:

```
CREATE TABLE essai (
  num INTEGER(4) NOT NULL,
  line CHAR(15) NOT NULL)
ENGINE=CONNECT table_type=MYSQL dbname=test tabname=people
option_list='user=root,host=localhost';
```

The specific connection items are:

Option	Default value	Description
Table	The table name	The name of the table to access.
Database	The current DB name	The database where the table is located.
Host	localhost*	The host of the server, a name or an IP address.
User	The current user	The connection user name.
Password	No password	An optional user password.
Port	The currently used port	The port of the server.
Quoted	0	1 if remote Tabname must be quoted.

- - When the host is specified as “localhost”, the connection is established on Linux using Linux sockets. On Windows, the connection is established by default using shared memory if it is enabled. If not, the TCP protocol is used. An alternative is to specify the host as “.” to use a named pipe connection (if it is enabled). This makes possible to use these table types with server skipping networking.

Caution: Take care not to refer to the MYSQL table itself to avoid an infinite loop!

MYSQL table can refer to the current server as well as to another server. Views can be referred by name or directly giving a source definition, for instance:

```
CREATE TABLE grp ENGINE=CONNECT table_type=mysql
CONNECTION='mysql://root@localhost/test/people'
SRCDEF='select title, count(*) as cnt from employees group by title';
```

When specified, the columns of the mysql table must exist in the accessed table with the same name, but can be only a subset of them and specified in a different order. Their type must be a type supported by CONNECT and, if it is not identical to the type of the accessed table matching column, a conversion can be done according to the rules given in Data type conversion.

Note: For columns prone to be targeted by a where clause, keep the column type compatible with the source table column type (numeric or character) to have a correct rephrasing of the where clause.

If you do not want to restrict or change the column definition, do not provide it and leave CONNECT get the column definition from the remote server. For instance:

```
CREATE TABLE essai ENGINE=CONNECT table_type=MYSQL
CONNECTION='mysql://root@localhost/test/people';
```

This will create the *essai* table with the same columns than the people table. If the target table contains CONNECT incompatible type columns, see Data type conversion to know how these columns can be converted or skipped.

Charset Specification

When accessing the remote table, CONNECT sets the connection charset set to the default local table charset as the FEDERATED engine does.

Do not specify a column character set if it is different from the table default character set even when it is the case on the remote table. This is because the remote column is translated to the local table character set when reading it. This is the default but it can be modified by the setting the character_set_results variable of the target server. If it must keep its setting, for instance to UTF8 when containing Unicode characters, specify the local default charset to its character set.

This means that it is not possible to correctly retrieve a remote table if it contains columns having different character sets. A solution is to retrieve it by several local tables, each accessing only columns with the same character set.

Indexing of MYSQL tables

Indexes are rarely useful with MYSQL tables. This is because CONNECT tries to access only the requested rows. For instance if you ask:

```
SELECT * FROM essai WHERE num = 23;
```

CONNECT will construct and send to the server the query:

```
SELECT num, line FROM people WHERE num = 23
```

If the *people* table is indexed on *num*, indexing are used on the remote server. This, in all cases, will limit the amount of data to retrieve on the network.

However, an index can be specified for columns that are prone to be used to join another table to the MYSQL table. For instance:

```
SELECT d.id, d.name, f.dept, f.salary
FROM loc_tab d STRAIGHT_JOIN cnc_tab f ON d.id = f.id
WHERE f.salary > 10000;
```

If the *id* column of the remote table addressed by the *cnc_tab* MYSQL table is indexed (which is likely if it is a key) you should also index the *id* column of the MYSQL *cnc_tab* table. If so, using “remote” indexing as does FEDERATED, only the useful rows of the remote table are retrieved during the join process. However, because these rows are retrieved by separate SELECT statements, this is useful only when retrieving a few rows of a big table.

In particular, you should not specify an index for columns not used for joining and above all DO NOT index a joined column if it is not indexed in the remote table. This would cause multiple scans of the remote table to retrieve the joined rows one by one.

Data Modifying Operations

The CONNECT MYSQL type supports SELECT and INSERT and a somewhat limited form of UPDATE and DELETE. These are described below.

The MYSQL type uses similar methods than the ODBC type to implement the INSERT,UPDATE and DELETE commands. Refer to the ODBC chapter for the restrictions concerning them.

For the UPDATE and DELETE commands, there are fewer restrictions because the remote server being a MySQL server, the syntax of the command are always acceptable by the remote server.

For instance, you can freely use keywords like IGNORE or LOW_PRIORITY as well as scalar functions in the SET and WHERE clauses.

However, there is still an issue on multi-table statements. Let us suppose you have a *t1* table on the remote server and want to execute a query such as:

```
UPDATE essai AS x SET line = (SELECT msg FROM t1 WHERE id = x.num)
WHERE num = 2;
```

When parsed locally, you will have errors if no *t1* table exists or if it does not have the referenced columns. When *t1* does not exist, you can overcome this issue by creating a local dummy *t1* table:

```
CREATE TABLE t1 (id INT, msg CHAR(1)) ENGINE=BLACKHOLE;
```

This will make the local parser happy and permit to execute the command on the remote server. Note however that having a local MySQL table defined on the remote *t1* table does not solve the problem unless it is also names *t1* locally.

This is why, to permit to have all types of commands executed by the data source without any restriction, CONNECT provides a specific MySQL table subtype described now.

Sending commands to a MariaDB Server

This can be done like for ODBC or JDBC tables by defining a specific table that are used to send commands and get the result of their execution..

```
  number int(5) not null flag=1,
CREATE TABLE send (
  command VARCHAR(128) NOT NULL,
  warnings INT(4) NOT NULL flag=3,
  message VARCHAR(255) flag=2)
ENGINE=CONNECT table_type=mysql
CONNECTION='mysql://user@host/database'
option_list='Execsrc=1,Maxerr=2';
```

The key points in this create statement are the EXECSRC option and the column definition.

The EXECSRC option tells that this table are used to send commands to the MariaDB server. Most of the sent commands do not return result set. Therefore, the table columns are used to specify the command to be executed and to get the result of the execution. The name of these columns can be chosen arbitrarily, their function coming from the FLAG value:

Flag=0:	The command to execute (the default)
Flag=1:	The number of affected rows, or the result number of columns if the command would return a result set.
Flag=2:	The returned (eventually error) message.
Flag=3:	The number of warnings.

How to use this table and specify the command to send? By executing a command such as:

```
SELECT * FROM send WHERE command = 'a command';
```

This will send the command specified in the WHERE clause to the data source and return the result of its execution. The syntax of the WHERE clause must be exactly as shown above. For instance:

```
SELECT * FROM send WHERE command =
'CREATE TABLE people (
num integer(4) primary key autoincrement,
line char(15) not null';
```

This command returns:

command	warnings	number	message
CREATE TABLE people (num integer(4) primary key aut…	0	0	Affected rows

Sending several commands in one call

It can be faster to execute because there are only one connection for all of them. To send several commands in one call, use the following syntax:

```
SELECT * FROM send WHERE command IN (
"update people set line = 'Two' where id = 2",
"update people set line = 'Three' where id = 3");
```

When several commands are sent, the execution stops at the end of them or after a command that is in error. To continue after n errors, set the option maxerr=*n* (0 by default) in the option list.

Note 1: It is possible to specify the SRCDEF option when creating an EXECSRC table. It are the command sent by default when a WHERE clause is not specified.

Note 2: Backslashes inside commands must be escaped. Simple quotes must be escaped if the command is specified between simple quotes, and double quotes if it is specified between double quotes.

Note 3: Sent commands apply in the specified database. However, they can address any table within this database.

Note 4: Currently, all commands are executed in mode AUTOCOMMIT.

Retrieving Warnings and Notes

If a sent command causes warnings to be issued, it is useless to resend a “show warnings” command because the MariaDB server is opened and closed when sending commands. Therefore, getting warnings requires a specific (and tricky) way.

To indicate that warning text must be added to the returned result, you must send a multi-command query containing “pseudo” commands that are not sent to the server but directly interpreted by the EXECSRC table. These “pseudo” commands are:

Warning	To get warnings
Note	To get notes
Error	To get errors returned as warnings (?)

Note that they must be spelled (case insensitive) exactly as above, no final “s”. For instance:

```
SELECT * FROM send WHERE command IN ('Warning','Note',
'drop table if exists try',
'create table try (id int key auto_increment, msg varchar(32) not
null) engine=aria',
"insert into try(msg) values('One'),(NULL),('Three') ",
"insert into try values(2,'Deux') on duplicate key update msg =
'Two'",
"insert into try(message) values('Four'),('Five'),('Six')",
'insert into try(id) values(NULL)',
"update try set msg = 'Four' where id = 4",
'select * from try');
```

This can return something like this:

command	warnings	number	message
drop table if exists try	1	0	Affected rows
Note	0	1051	Unknown table ‘try’
create table try (id int key auto_increment, msg…	0	0	Affected rows
insert into try(msg) values(‘One’),(NULL),(‘Three’)	1	3	Affected rows
Warning	0	1048	Column ‘msg’ cannot be null
insert into try values(2,‘Deux’) on duplicate key…	0	2	Affected rows
insert into try(msge) values(‘Four’),(‘Five’),(‘Six’)	0	1054	Unknown column ‘msge’ in ‘field list’
insert into try(id) values(NULL)	1	1	Affected rows
Warning	0	1364	Field ‘msg’ doesn’t have a default value
update try set msg = ‘Four’ where id = 4	0	1	Affected rows
select * from try	0	2	Result set columns

The execution continued after the command in error because of the MAXERR option. Normally this would have stopped the execution.

Of course, the last “select” command is useless here because it cannot return the table contain. Another MYSQL table without the EXECSRC option and with proper column definition should be used instead.

Connection Engine Limitations

Data types

There is a maximum key.index length of 255 bytes. You may be able to declare the table without an index and rely on the engine condition pushdown and remote schema.

The following types can’t be used:

- BIT
- BINARY
- TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB
- TINYTEXT, MEDIUMTEXT, LONGTEXT
- ENUM
- SET
- Geometry types

Note: TEXT is allowed. However, the handling depends on the values given to the connect_type_conv and connect_conv_size system variables, and by default no conversion of TEXT columns is permitted.

SQL Limitations

The following SQL queries are not supported

- REPLACE INTO
- INSERT ... ON DUPLICATE KEY UPDATE

CONNECT MYSQL versus FEDERATED

The CONNECT MYSQL table type should not be regarded as a replacement for the FEDERATED(X) engine. The main use of the MYSQL type is to access other engine tables as if they were CONNECT tables. This was necessary when accessing tables from some CONNECT table types such as TBL, XCOL, OCCUR, or PIVOT that are designed to access CONNECT tables only. When their target table is not a
CONNECT table, these types are silently using internally an intermediate MYSQL table.

However, there are cases where you can use MYSQL CONNECT tables yourself, for instance:

1. When the table are used by a TBL table. This enables you to specify the connection parameters for each sub-table and is more efficient than using a local FEDERATED sub-table.
2. When the desired returned data is directly specified by the SRCDEF option.
   This is great to let the remote server do most of the job, such as grouping and/or joining tables. This cannot be done with the FEDERATED engine.
3. To take advantage of the push_cond facility that adds a where clause to the command sent to the remote table. This restricts the size of the result set and can be crucial for big tables.
4. For tables with the EXECSRC option on.
5. When doing tests. For instance to check a connection string.

If you need multi-table updating, deleting, or bulk inserting on a remote table, you can alternatively use the FEDERATED engine or a “send” table specifying the EXECSRC option on.

See also

- Using the TBL and MYSQL types together

This page is licensed: GPLv2

CONNECT - NoSQL Table Types

They are based on files that do not match the relational format but often represent hierarchical data. CONNECT can handle JSON, INI-CFG, XML, and some HTML files.

The way it is done is different from what MySQL or PostgreSQL does. In addition to including in a table some column values of a specific data format (JSON, XML) to be handled by specific functions, CONNECT can directly use JSON, XML or INI files that are produced by other applications, and this is the table definition that describes where and how the contained information must be retrieved.

This is also different from what MariaDB does with dynamic columns, which is close to what MySQL and PostgreSQL do with the JSON column type.

Note: The LEVEL option used with these tables should, from Connect 1.07.0002, be specified as DEPTH. Also, what was specified with the FIELD_FORMAT column option should now also be specified using JPATH or XPATH.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT OCCUR Table Type

Similarly to the XCOL table type, `OCCUR` is an extension to the PROXY type when referring to a table or view having several columns containing the same kind of data. It enables having a different view of the table where the data from

these columns are put in a single column, eventually causing several rows to be generated from one row of the object table. For example, supposing we have a_pets_ table:

name	dog	cat	rabbit	bird	fish
John	2	0	0	0	0
Bill	0	1	0	0	0
Mary	1	1	0	0	0
Lisbeth	0	0	2	0	0
Kevin	0	2	0	6	0
Donald	1	0	0	0	3

We can create an occur table by:

```
CREATE TABLE xpet (
  name VARCHAR(12) NOT NULL,
  race CHAR(6) NOT NULL,
  NUMBER INT NOT NULL)
ENGINE=CONNECT table_type=occur tabname=pets
option_list='OccurCol=number,RankCol=race'
Colist='dog,cat,rabbit,bird,fish';
```

When displaying it by

```
SELECT * FROM xpet;
```

We will get the result:

name	race	number
John	dog	2
Bill	cat	1
Mary	dog	1
Mary	cat	1
Lisbeth	rabbit	2
Kevin	cat	2
Kevin	bird	6
Donald	dog	1
Donald	fish	3

First of all, the values of the column listed in the Colist option have been put in a unique column whose name is given by the OccurCol option. When several columns have non null (or pseudo-null) values, several rows are generated, with the other normal columns values repeated.

In addition, an optional special column was added whose name is given by the RankCol option. This column contains the name of the source column from which the value of the OccurCol column comes from. It permits here to know the race of the pets whose number is given in *number*.

This table type permit to make queries that would be more complicated to make on the original tables. For instance to know who as more than 1 pet of a kind, you can simply ask:

```
SELECT * FROM xpet WHERE NUMBER > 1;
```

You will get the result:

name	race	number
John	dog	2
Lisbeth	rabbit	2
Kevin	cat	2
Kevin	bird	6
Donald	fish	3

Note 1: Like for XCOL tables, no row multiplication for queries not implying the Occur column.

Note 2: Because the OccurCol was declared “not null” no rows were generated for null or pseudo-null values of the column list. If the OccurCol is declared as nullable, rows are also generated for columns containing null or pseudo-null values.

Occur tables can be also defined from views or source definition. Also, CONNECT is able to generate the column definitions if not specified:

```
CREATE TABLE ocsrc ENGINE=CONNECT table_type=occur
colist='january,february,march,april,may,june,july,august,september,
october,november,december' option_list='rankcol=month,occurcol=day'
srcdef='select ''Foo'' name, 8 january, 7 february, 2 march, 1 april,
  8 may, 14 june, 25 july, 10 august, 13 september, 22 october, 28
  november, 14 december';
```

This table is displayed as:

name	month	day
Foo	january	8
Foo	february	7
Foo	march	2
Foo	april	1
Foo	may	8
Foo	june	14
Foo	july	25
Foo	august	10
Foo	september	13
Foo	october	22
Foo	november	28
Foo	december	14

This page is licensed: CC BY-SA / Gnu FDL

CONNECT ODBC Table Type: Accessing Tables From Another DBMS

ODBC (Open Database Connectivity) is a standard API for accessing database management systems (DBMS). CONNECT uses this API to access data contained in other DBMS without having to implement a specific application for each one. An exception is the access to MySQL that should be done using the MYSQL table type.

Note: On Linux, unixODBC must be installed.

These tables are given the type ODBC. For example, if a “Customers” table is contained in an Access™ database you can define it with a command such as:

```
CREATE TABLE Customer (
  CustomerID VARCHAR(5),
  CompanyName VARCHAR(40),
  ContactName VARCHAR(30),
  ContactTitle VARCHAR(30),
  Address VARCHAR(60),
  City VARCHAR(15),
  Region VARCHAR(15),
  PostalCode VARCHAR(10),
  Country VARCHAR(15),
  Phone VARCHAR(24),
  Fax VARCHAR(24))
ENGINE=CONNECT table_type=ODBC block_size=10
tabname='Customers'
CONNECTION='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB;';
```

Tabname option defaults to the table name. It is required if the source table name is different from the name of the CONNECT table. Note also that for some data sources this name is case sensitive.

Often, because CONNECT can retrieve the table description using ODBC catalog functions, the column definitions can be unspecified. For instance this table can be simply created as:

```
CREATE TABLE Customer ENGINE=CONNECT table_type=ODBC
  block_size=10 tabname='Customers'
  CONNECTION='DSN=MS Access Database;DBQ=C:/Program Files/Microsoft Office/Office/1033/FPNWIND.MDB;';
```

The `BLOCK_SIZE` specification are used later to set the RowsetSize when retrieving rows from the ODBC table. A reasonably large RowsetSize can greatly accelerate the fetching process.

If you specify the column description, the column names of your table must exist in the data source table. However, you are not obliged to define all the data source columns and you can change the order of the columns. Some type conversion can also be done if appropriate. For instance, to access the FireBird sample table EMPLOYEE, you could define your table as:

```
CREATE TABLE empodbc (
  EMP_NO SMALLINT(5) NOT NULL,
  FULL_NAME VARCHAR(37) NOT NULL),
  PHONE_EXT VARCHAR(4) NOT NULL,
  HIRE_DATE DATE,
  DEPT_NO SMALLINT(3) NOT NULL,
  JOB_COUNTRY VARCHAR(15),
  SALARY DOUBLE(12,2) NOT NULL)
ENGINE=CONNECT table_type=ODBC tabname='EMPLOYEE'
CONNECTION='DSN=firebird';
```

This definition ignores the FIRST_NAME, LAST_NAME, JOB_CODE, and JOB_GRADE columns. It places the FULL_NAME last column of the original table in second position. The type of the HIRE_DATE column was changed from *timestamp* to *date* and the type of the DEPT_NO column was changed from *char* to *integer*.

Currently, some restrictions apply to ODBC tables:

1. Cursor type is forward only (sequential reading).
2. No indexing of ODBC tables (do not specify any columns as key). However, because CONNECT can often add a where clause to the query sent to the data source, indexing are used by the data source if it supports it. (Remote indexing is available with version 1.04, released with MariaDB 10.1.6)
3. CONNECT ODBC supports SELECT and INSERT. UPDATE and DELETE are also supported in a somewhat restricted way (see below). For other operations, use an ODBC table with the EXECSRC option (see below) to directly send proper commands to the data source.

Random Access of ODBC Tables

In CONNECT version 1.03 (until MariaDB 10.1.5) ODBC tables are not indexable. Version 1.04 (from MariaDB 10.1.6) adds remote indexing facility to the ODBC table type.

However, some queries require random access to an ODBC table; for instance when it is joined to another table or used in an order by queries applied to a long column or large tables.

There are several ways to enable random (position) access to a CONNECT ODBC table. They are dependent on the following table options:

Option	Type	Used For
Block_Size	Integer	Specifying the rowset size.
Memory*	Integer	Storing the result set in memory.
Scrollable*	Boolean	Using a scrollable cursor.

`*` - To be specified in the option_list.

When dealing with small tables, the simpler way to enable random access is to specify a rowset size equal or larger than the table size (or the result set size if a push down where clause is used). This means that the whole result is in memory on the first fetch and CONNECT will use it for further positional accesses.

Another way to have the result set in memory is to use the memory option. This option can be set to the following values:

0. No memory used (the default). Best when the table is read sequentially as in SELECT statements with only eventual WHERE clauses.**1.** Memory size required is calculated during the first sequential table read. The allocated memory is filled during the second sequential read. Then the table rows are retrieved from the memory. This should be used when the table are accessed several times randomly, such as in sub-selects or being the target table of a join.**2.** A first query is executed to get the result set size and the needed memory is allocated. It is filled on the first sequential reading. Then random access of the table is possible. This can be used in the case of ORDER BY clauses, when MariaDB uses position reading.

Note that the best way to handle ORDER BY is to set the max_length_for_sort_data variable to a larger value (its default value is 1024 that is pretty small). Indeed, it requires less memory to be used, particularly when a WHERE clause limits the retrieved data set. This is because in the case of an order by query, MariaDB firstly retrieves the sequentially the result set and the position of each records. Often the sort can be done from the result set if it is not too big. But if too big, or if it implies some “long” columns, only the positions are sorted and MariaDB retrieves the final result from the table read in random order. If setting the max_length_for_sort_data variable is not feasible or does not work, to be able to retrieve table data from memory after the first sequential read, the memory option must be set to 2.

For tables too large to be stored in memory another possibility is to make your table to use a scrollable cursor. In this case each randomly accessed row can be retrieved from the data source specifying its cursor position, which is reasonably fast. However, scrollable cursors are not supported by all data sources.

With CONNECT version 1.04 (from MariaDB 10.1.6), another way to provide random access is to specify some columns to be indexed. This should be done only when the corresponding column of the source table is also indexed. This should be used for tables too large to be stored in memory and is similar to the remote indexing used by the MYSQL table type and by the FEDERATED engine.

There remains the possibility to extract data from the external table and to construct another table of any file format from the data source. For instance to construct a fixed formatted DOS table containing the CUSTOMER table data, create the table as

```
CREATE TABLE Custfix ENGINE=CONNECT File_name='customer.txt'
  table_type=fix block_size=20 AS SELECT * FROM customer;
```

Now you can use *custfix* for fast database operations on the copied_customer_ table data.

Retrieving data from a spreadsheet

ODBC can also be used to create tables based on tabular data belonging to an
Excel spreadsheet:

```
CREATE TABLE XLCONT
ENGINE=CONNECT table_type=ODBC tabname='CONTACT'
CONNECTION='DSN=Excel Files;DBQ=D:/Ber/Doc/Contact_BP.xls;';
```

This supposes that a tabular zone of the sheet including column headers is defined as a table named CONTACT or using a “named reference”. Refer to the Excel documentation for how to specify tables inside sheets. Once done, you can ask:

```
SELECT * FROM xlcont;
```

This will extract the data from Excel and display:

Nom	Function	Societe
Boisseau Frederic		9 Telecom
Martelliere Nicolas		Vidal SA (Groupe UBM)
Remy Agathe		Price Minister
Du Halgouet Tanguy		Danone
Vandamme Anna		GDF
Thomas Willy		Europ Assistance France
Thomas Dominique		Acoss (DG des URSSAF)
Thomas Berengere	Responsible SI Decisionnel	DEXIA Credit Local
Husy Frederic	Responsible Decisionnel	Neuf Cegetel
Lemonnier Nathalie	Directeur Marketing Client	Louis Vuitton
Louis Loic	Reporting International Decisionnel	Accor
Menseau Eric		Orange France

Here again, the columns description was left to CONNECT when creating the table.

Multiple ODBC tables

The concept of multiple tables can be extended to ODBC tables when they are physically represented by files, for instance to Excel or Access tables. The condition is that the connect string for the table must contain a field DBQ=*filename*, in which wildcard characters can be included as for multiple=1 tables in their filename. For instance, a table contained in several
Excel files CA200401.xls, CA200402.xls, …CA200412.xls can be created by a command such as:

```
CREATE TABLE ca04mul (DATE CHAR(19), OPERATION VARCHAR(64),
  Debit DOUBLE(15,2), Credit DOUBLE(15,2))
ENGINE=CONNECT table_type=ODBC multiple=1
qchar= '"' tabname='bank account'
CONNECTION='DSN=Excel Files;DBQ=D:/Ber/CA/CA2004*.xls;';
```

Providing that in each file the applying information is internally set for
Excel as a table named “bank account”. This extension to ODBC does not support_multiple_=2. The *qchar* option was specified to make the identifiers quoted in the select statement sent to ODBC, in particular the when the table or column names contain blanks, to avoid SQL syntax errors.

Caution: Avoid accessing tables belonging to the currently running MariaDB server via the MySQL ODBC connector. This may not work and may cause the server to be restarted.

Performance consideration

To avoid extracting entire tables from an ODBC source, which can be a lengthy process, CONNECT extracts the “compatible” part of query WHERE clauses and adds it to the ODBC query. Compatible means that it must be understood by the data source. In particular, clauses involving scalar functions are not kept because the data source may have different functions than MariaDB or use a different syntax. Of course, clauses involving sub-select are also skipped. This will transfer eventual indexing to the data source.

Take care with clauses involving string items because you may not know whether they are treated by the data source as case sensitive or case insensitive. If in doubt, make your queries as if the data source was processing strings as case sensitive to avoid incomplete results.

Using ODBC Tables inside correlated sub-queries

Unlike not correlated subqueries that are executed only once, correlated subqueries are executed many times. It is what ODBC calls a “requery”. Several methods can be used by CONNECT to deal with this depending on the setting of the MEMORY or SCROLLABLE Boolean options:

Option	Description
Default	Implementing “requery” by discarding the current result set and re submitting the query (as MFC does)
Memory=1 or 2	Storing the result set in memory as MYSQL tables do.
Scrollable=Yes	Using a scrollable cursor.

Note: the MEMORY and SCROLLABLE options must be specified in the OPTION _ LIST.

Because the table is accessed several times, this can make queries last very long except for small tables and is almost unacceptable for big tables. However, if it cannot be avoided, using the memory method is the best choice and can be more than four times faster than the default method. If it is supported by the driver, using a scrollable cursor is slightly slower than using memory but can be an alternative to avoid memory problems when the sub-query returns a huge result set.

If the result set is of reasonable size, it is also possible to specify the block_size option equal or slightly larger than the result set. The whole result set being read on the first fetch, can be accessed many times without having to do anything else.

Another good workaround is to replace within the correlated sub-query the ODBC table by a local copy of it because MariaDB is often able to optimize the query and to provide a very fast execution.

Accessing specified views

Instead of specifying a source table name via the TABNAME option, it is possible to retrieve data from a “view” whose definition is given in a new option SRCDEF. For instance:

```
CREATE TABLE custnum (
  country VARCHAR(15) NOT NULL,
  customers INT(6) NOT NULL)
ENGINE=CONNECT TABLE_TYPE=ODBC BLOCK_SIZE=10
CONNECTION='DSN=MS Access Database;DBQ=C:/Program Files/Microsoft Office/Office/1033/FPNWIND.MDB;'
SRCDEF='select country, count(*) as customers from customers group by country';
```

Or simply, because CONNECT can retrieve the returned column definition:

```
CREATE TABLE custnum ENGINE=CONNECT TABLE_TYPE=ODBC BLOCK_SIZE=10
CONNECTION='DSN=MS Access Database;DBQ=C:/Program Files/Microsoft Office/Office/1033/FPNWIND.MDB;'
SRCDEF='select country, count(*) as customers from customers group by country';
```

Then, when executing for instance:

```
SELECT * FROM custnum WHERE customers > 3;
```

The processing of the group by is done by the data source, which returns only the generated result set on which only the where clause is performed locally.
The result:

country	customers
Brazil	9
France	11
Germany	11
Mexico	5
Spain	5
UK	7
USA	13
Venezuela	4

This makes possible to let the data source do complicated operations, such as joining several tables or executing procedures returning a result set. This minimizes the data transfer through ODBC.

Data Modifying Operations

The only data modifying operations are the INSERT , UPDATE and DELETE commands.
They can be executed successfully only if the data source database or tables are not read/only.

INSERT Command

When inserting values to an ODBC table, local values are used and sent to the ODBC table. This does not make any difference when the values are constant but in a query such as:

```
INSERT INTO t1 SELECT * FROM t2;
```

Where t1 is an ODBC table, t2 is a locally defined table that must exist on the local server. Besides, it is a good way to create a distant ODBC table from local data.

CONNECT does not directly support INSERT commands such as:

```
INSERT INTO t1 VALUES(2,'Deux') ON duplicate KEY UPDATE msg = 'Two';
```

Sure enough, the “on duplicate key update” part of it is ignored, and will result in error if the key value is duplicated.

UPDATE and DELETE Commands

Unlike the INSERT command, UPDATE and DELETE are supported in a simplified way. Only simple table commands are supported; CONNECT does not support multi-table commands, commands sent from a procedure, or issued via a trigger.

These commands are just rephrased to correspond to the data source syntax and sent to the data source for execution. Let us suppose we created the table:

```
CREATE TABLE tolite (
  id INT(9) NOT NULL,
  nom VARCHAR(12) NOT NULL,
  nais DATE DEFAULT NULL,
  rem VARCHAR(32) DEFAULT NULL)
ENGINE=CONNECT TABLE_TYPE=ODBC tabname='lite'
CONNECTION='DSN=SQLite3 Datasource;Database=test.sqlite3'
CHARSET=utf8 DATA_CHARSET=utf8;
```

We can populate it by:

```
INSERT INTO tolite VALUES(1,'Toto',NOW(),'First'),
(2,'Foo','2012-07-14','Second'),(4,'Machin','1968-05-30','Third');
```

The function `now()` are executed by MariaDB and it returned value sent to the ODBC table.

Let us see what happens when updating the table. If we use the query:

```
UPDATE tolite SET nom = 'Gillespie' WHERE id = 10;
```

CONNECT will rephrase the command as:

```
UPDATE lite SET nom = 'Gillespie' WHERE id = 10;
```

What it did is just to replace the local table name with the remote table name and change all the back ticks to blanks or to the data source identifier quoting characters if QUOTED is specified.
Then this command are sent to the data source to be executed by it.

This is simpler and can be faster than doing a positional update using a cursor and commands such as “select … for update of …” that are not supported by all data sources. However, there are some restrictions that must be understood due to the way it is handled by MariaDB.

1. MariaDB does not know about all the above. The command are parsed as if it were to be executed locally. Therefore, it must respect the MariaDB syntax.
2. Being executed by the data source, the (rephrased) command must also respect the data source syntax.
3. All data referenced in the SET and WHERE clause belongs to the data source.

This is possible because both MariaDB and the data source are using the SQL language. But you must use only the basic features that are part of the core
SQL language. For instance, keywords like IGNORE or LOW_PRIORITY will cause syntax error with many data source.

Scalar function names also can be different, which severely restrict the use of them. For instance:

```
UPDATE tolite SET nais = NOW() WHERE id = 2;
```

This will not work with SQLite3, the data source returning an “unknown scalar function” error message. Note that in this particular case, you can rephrase it to:

```
UPDATE tolite SET nais = DATE('now') WHERE id = 2;
```

This understood by both parsers, and even if this function would return NULL executed by MariaDB, it does return the current date when executed by SQLite3.
But this begins to become too trickery so to overcome all these restrictions, and permit to have all types of commands executed by the data source, CONNECT provides a specific ODBC table subtype described now.

Sending commands to a Data Source

This can be done using a special subtype of ODBC table. Let us see this in an example:

```
CREATE TABLE crlite (
  command VARCHAR(128) NOT NULL,
  NUMBER INT(5) NOT NULL flag=1,
  message VARCHAR(255) flag=2)
ENGINE=CONNECT table_type=odbc
CONNECTION='Driver=SQLite3 ODBC Driver;Database=test.sqlite3;NoWCHAR=yes'
option_list='Execsrc=1';
```

The key points in this create statement are the EXECSRC option and the column definition.

The EXECSRC option tells that this table are used to send a command to the data source. Most of the sent commands do not return result set. Therefore, the table columns are used to specify the command to be executed and to get the result of the execution. The name of these columns can be chosen arbitrarily, their function coming from the FLAG value:

Flag=0:	The command to execute.
Flag=1:	The affected rows, or -1 in case of error, or the result number of column if the command returns a result set.
Flag=2:	The returned (eventually error) message.

How to use this table and specify the command to send? By executing a command such as:

```
SELECT * FROM crlite WHERE command = 'a command';
```

This will send the command specified in the WHERE clause to the data source and return the result of its execution. The syntax of the WHERE clause must be exactly as shown above. For instance:

```
SELECT * FROM crlite WHERE command =
'CREATE TABLE lite (
ID integer primary key autoincrement,
name char(12) not null,
birth date,
rem varchar(32))';
```

This command returns:

command	number	message
CREATE TABLE lite (ID integer primary key autoincrement, name…	0	Affected rows

Now we can create a standard ODBC table on the newly created table:

```
CREATE TABLE tlite
ENGINE=CONNECT TABLE_TYPE=ODBC tabname='lite'
CONNECTION='Driver=SQLite3 ODBC Driver;Database=test.sqlite3;NoWCHAR=yes'
CHARSET=utf8 DATA_CHARSET=utf8;
```

We can populate it directly using the supported INSERT statement:

```
INSERT INTO tlite(name,birth) VALUES('Toto','2005-06-12');
INSERT INTO tlite(name,birth,rem) VALUES('Foo',NULL,'No ID');
INSERT INTO tlite(name,birth) VALUES('Truc','1998-10-27');
INSERT INTO tlite(name,birth,rem) VALUES('John','1968-05-30','Last');
```

And see the result:

```
SELECT * FROM tlite;
```

ID	name	birth	rem
1	Toto	2005-06-12	NULL
2	Foo	NULL	No ID
3	Truc	1998-10-27	NULL
4	John	1968-05-30	Last

Any command, for instance UPDATE, can be executed from the *crlite* table:

```
SELECT * FROM crlite WHERE command =
'update lite set birth = ''2012-07-14'' where ID = 2';
```

This command returns:

command	number	message
update lite set birth = '2012-07-15' where ID = 2	1	Affected rows

Let us verify it:

```
SELECT * FROM tlite WHERE ID = 2;
```

ID	name	birth	rem
2	Foo	2012-07-15	No ID

The syntax to send a command is rather strange and may seem unnatural. It is possible to use an easier syntax by defining a stored procedure such as:

```
CREATE PROCEDURE send_cmd(cmd VARCHAR(255))
MODIFIES SQL DATA
SELECT * FROM crlite WHERE command = cmd;
```

Now you can send commands like this:

```
call send_cmd('drop tlite');
```

This is possible only when sending one single command.

Sending several commands together

Grouping commands uses an easier syntax and is faster because only one connection is made for the all of them. To send several commands in one call, use the following syntax:

```
SELECT * FROM crlite WHERE command IN (
  'update lite set birth = ''2012-07-14'' where ID = 2',
  'update lite set birth = ''2009-08-10'' where ID = 3');
```

When several commands are sent, the execution stops at the end of them or after a command that is in error. To continue after *n* errors, set the option maxerr=*n* (0 by default) in the option list.

Note 1: It is possible to specify the SRCDEF option when creating an EXECSRC table. It are the command sent by default when a WHERE clause is not specified.

Note 2: Most data sources do not allow sending several commands separated by semi-colons.

Note 3: Quotes inside commands must be escaped. This can be avoided by using a different quoting character than the one used in the command

Note 4: The sent command must obey the data source syntax.

Note 5: Sent commands apply in the specified database. However, they can address any table within this database, or belonging to another database using the name syntax *schema.tabname*.

Connecting to a Data Source

There are two ways to establish a connection to a data source:

1. Using SQLDriverConnect and a Connection String
2. Using SQLConnect and a Data Source Name (DSN)

The first way uses a Connection String whose components describe what is needed to establish the connection. It is the most complete way to do it and by default CONNECT uses it.

The second way is a simplified way in which ODBC is just given the name of a DSN that must have been defined to ODBC or UnixOdbc and that contains the necessary information to establish the connection. Only the user name and password can be specified out of the DSN specification.

Defining the Connection String

Using the first way, the connection string must be specified. This is sometimes the most difficult task when creating ODBC tables because, depending on the operating system and the data source, this string can widely differ.

The format of the ODBC Connection String is:

```
connection-string::= empty-string[;] | attribute[;] | attribute; connection-string
empty-string ::=
attribute ::= attribute-keyword=attribute-value | DRIVER=[{]attribute-value[}]
attribute-keyword ::= DSN | UID | PWD | driver-defined-attribute-keyword
attribute-value ::= character-string
driver-defined-attribute-keyword = identifier
```

Where character-string has zero or more characters; identifier has one or more characters; attribute- keyword is not case-sensitive; attribute-value may be case-sensitive; and the value of the DSN keyword does not consist solely of blanks. Due to the connection string grammar, keywords and attribute values that contain the characters `[]{}(),;?*=!@` should be avoided. The value of the DSN keyword cannot consist only of blanks, and should not contain leading blanks. Because of the grammar of the system information, keywords and data source names cannot contain the backslash () character. Applications do not have to add braces around the attribute value after the DRIVER keyword unless the attribute contains a semicolon (;), in which case the braces are required.
If the attribute value that the driver receives includes the braces, the driver should not remove them, but they should be part of the returned connection string.

ODBC Defined Connection Attributes

The ODBC defined attributes are:

- DSN - the name of the data source to connect to. You must create this before attempting to refer to it. You create new DSNs through the ODBC Administrator (Windows), ODBCAdmin (unixODBC's GUI manager) or in the odbc.ini file.
- DRIVER - the name of the driver to connect to. You can use this in DSN-less connections.
- FILEDSN - the name of a file containing the connection attributes.
- UID/PWD - any username and password the database requires for authentication.
- SAVEFILE - request the DSN attributes are saved in this file.

Other attributes are DSN dependent attributes. The connection string can give the name of the driver in the DRIVER field or the data source in the DSN field (attention! meet the spelling and case) and has other fields that depend on

the data source. When specifying a file, the DBQ field must give the **full** path and name of the file containing the table. Refer to the specific ODBC connector documentation for the exact syntax of the connection string.

Using a Predefined DSN

This is done by specifying in the option list the Boolean option “UseDSN” as yes or 1. In addition, string options “user” and “password” can be optionally specified in the option list.

When doing so, the connection string just contains the name of the predefined Data Source. For instance:

```
CREATE TABLE tlite ENGINE=CONNECT TABLE_TYPE=ODBC tabname='lite'
CONNECTION='SQLite3 Datasource'
OPTION_LIST='UseDSN=Yes,User=me,Password=mypass';
```

Note: the connection data source name (limited to 32 characters) should not be preceded by “DSN=”.

ODBC Tables on Linux/Unix

In order to use ODBC tables, you will need to have unixODBC installed. Additionally, you will need the ODBC driver for your foreign server’s protocol. For example, for MS SQL Server or Sybase, you will need to have FreeTDS installed.

Make sure the user running mysqld (usually the mysql user) has permission to the ODBC data source configuration and the ODBC drivers.
If you get an error on Linux/Unix when using TABLE_TYPE=ODBC:

```
Error Code: 1105 [unixODBC][Driver Manager]Can't open lib
'/usr/cachesys/bin/libcacheodbc.so' : file not found
```

You must make sure that the user running mysqld (usually “mysql”) has enough permission to load the ODBC driver library. It can happen that the driver file does not have enough read privileges (use chmod to fix this), or loading is prevented by SELinux configuration (see below).

Try this command in a shell to check if the driver had enough permission:

```
sudo -u mysql ldd /usr/cachesys/bin/libcacheodbc.so
```

SELinux

SELinux can cause various problems. If you think SELinux is causing problems, check the system log (e.g. /var/log/messages) or the audit log (e.g. /var/log/audit/audit.log).

mysqld can’t load some executable code, so it can’t use the ODBC driver.

Example error:

```
Error Code: 1105 [unixODBC][Driver Manager]Can't open lib
'/usr/cachesys/bin/libcacheodbc.so' : file not found
```

Audit log:

```
type=AVC msg=audit(1384890085.406:76): avc: denied { execute }
for pid=1433 comm="mysqld"
path="/usr/cachesys/bin/libcacheodbc.so" dev=dm-0 ino=3279212
scontext=unconfined_u:system_r:mysqld_t:s0
tcontext=unconfined_u:object_r:usr_t:s0 tclass=file
```

mysqld can’t open TCP sockets on some ports, so it can’t connect to the foreign server.

Example error:

```
ERROR 1296 (HY000): Got error 174 '[unixODBC][FreeTDS][SQL Server]Unable to connect to data source' from
CONNECT
```

Audit log:

```
type=AVC msg=audit(1423094175.109:433): avc:  denied  { name_connect } for  pid=3193 comm="mysqld"
dest=1433 scontext=system_u:system_r:mysqld_t:s0 tcontext=system_u:object_r:mssql_port_t:s0
tclass=tcp_socket
```

ODBC Catalog Information

Depending on the version of the used ODBC driver, some additional information on the tables are existing, such as table QUALIFIER or OWNER for old versions, now named CATALOG or SCHEMA since version 3.

CATALOG is apparently rarely used by most data sources, but SCHEMA (formerly
OWNER) is and corresponds to the DATABASE information of MySQL.

The issue is that if no schema name is specified, some data sources return information for all schemas while some others only return the information of the “default” schema. In addition, the used “schema” or “database” is sometimes implied by the connection string and sometimes is not. Sometimes, it also can be included in a data source definition.

CONNECT offers two ways to specify this information:

1. When specified, the DBNAME create table option is regarded by ODBC tables as the SCHEMA name.
2. Table names can be specified as “cat.sch.tab” allowing to set the catalog and schema info.

When both are used, the qualified table name has precedence over DBNAME . For instance:

Tabname	DBname	Description
test.t1		The t1 table of the test schema.
test.t1	mydb	The t1 table of the test schema (test has precedence)
t1	mydb	The t1 table of the mydb schema
%.%.%		All tables in all catalogs and all schemas
t1		The t1 table in the default or all schema depending on the DSN
%.t1		The t1 table in all schemas for all DSN
test.%		All tables in the test schema

When creating a standard ODBC table, you should make sure only one source table is specified. Specifying more than one source table must be done only for
CONNECT catalog tables (with CATFUNC=tables or columns).

In particular, when column definition is left to the Discovery feature, if tables with the same name are present in several schemas and the schema name is not specified, several columns with the same name are generated. This will make the creation fail with a not very explicit error message.

Note: With some ODBC drivers, the DBNAME option or qualified table name is useless because the schema implied by the connection string or the definition of the data source has priority over the specified DBNAME .

Table name case

Another issue when dealing with ODBC tables is the way table and column names are handled regarding of the case.

For instance, Oracle follows to the SQL standard here. It converts non-quoted identifiers to upper case. This is correct and expected. PostgreSQL is not standard. It converts identifiers to lower case. MySQL/MariaDB is not

standard. They preserve identifiers on Linux, and convert to lower case on Windows.

Think about that if you fail to see a table or a column on an ODBC data source.

Non-ASCII Character Sets with Oracle

When connecting through ODBC, the MariaDB Server operates as a client to the foreign database management system. As such, it requires that you configure MariaDB as you would configure native clients for the given database server.

In the case of connecting to Oracle, when using non-ASCI character sets, you need to properly set the NLS_LANG environment variable before starting the MariaDB Server.

For instance, to test this on Oracle, create a table that contains a series of special characters:

```
CREATE TABLE t1 (letter VARCHAR(4000));

INSERT INTO t1 VALUES
   (UTL_RAW.CAST_TO_VARCHAR2(HEXTORAW('C4'))),
   (UTL_RAW.CAST_TO_VARCHAR2(HEXTORAW('C5'))),
   (UTL_RAW.CAST_TO_VARCHAR2(HEXTORAW('C6')));

SELECT letter, RAWTOHEX(letter) FROM t1;

letter | RAWTOHEX(letter)
-------|-----------------
Ä      | C4
Å      | C5
Æ      | C6
```

Then create a connecting table on MariaDB and attempt the same query:

```
CREATE TABLE t1 (
   letter VARCHAR(4000))
ENGINE=CONNECT
DEFAULT CHARSET=utf8mb4
CONNECTION='DSN=YOUR_DSN'
TABLE_TYPE = 'ODBC'
DATA_CHARSET = latin1
TABNAME = 'YOUR_SCHEMA.T1';

SELECT letter, HEX(letter) FROM t1;

+--------+-------------+
| letter | HEX(letter) |
+--------+-------------+
| A      |      41 |
| ?      |      3F |
| ?      |      3F |
+--------+-------------+
```

While the character set is defined in a way that satisfies MariaDB, it has not been defined for Oracle, (that is, setting the NLS_LANG environment variable). As a result, Oracle is not providing the characters you want to MariaDB and Connect.
The specific method of setting the NLS_LANG variable can vary depending on your operating system or distribution. If you're experiencing this issue, check your OS documentation for more details on how to properly set environment variables.

Using systemd

With Linux distributions that use systemd, you need to set the environment variable in the service file, (systemd doesn't read from the /etc/environment file).

This is done by setting the Environment variable in the [Service] unit. For instance,

```
# systemctl edit mariadb.service

[Service]
Environment=NLS_LANG=GERMAN_GERMANY.WE8ISO8859P1
```

Then restart MariaDB,

```
# systemctl restart mariadb.service
```

You can now retrieve the appropriate characters from Oracle tables:

```
SELECT letter, HEX(letter) FROM t1;

+--------+-------------+
| letter | HEX(letter) |
+--------+-------------+
| Ä      | C384        |
| Å      | C385        |
| Æ      | C386        |
+--------+-------------+
```

Using Windows

Microsoft Windows doesn't ignore environment variables the way systemd does on Linux, but it does require that you set the NLS_LANG environment variable on your system. In order to do so, you need to open an elevated command-prompt, (that is, Cmd.exe with administrative privileges).

From here, you can use the Setx command to set the variable. For instance,

```
Setx NLS_LANG GERMAN_GERMANY.WE8ISO8859P1 /m
```

Note: For more detail about this, see MDEV-17501.

`OPTION_LIST` Values Supported by the ODBC Tables

The following options can be given as comma-separated string to the `OPTION_LIST` value in the `CREATE TABLE` statement.

Name	Default	Description
MaxRes	0	Maximum number of rows returned by catalog functions
ConnectTimeout	-1	Connection timeout in seconds, unlimited by default
QueryTimeout	-1	Query timeout in seconds, unlimited by default
UseDSN	false	Use pre-configured DSN

This page is licensed: GPLv2

CONNECT PIVOT Table Type

This table type can be used to transform the result of another table or view (called the source table) into a pivoted table along "pivot" and "facts" columns. A pivot table is a great reporting tool that sorts and sums (by default) independent of the original data layout in the source table.

For example, let us suppose you have the following “Expenses” table:

Who	Week	What	Amount
Joe	3	Beer	18.00
Beth	4	Food	17.00
Janet	5	Beer	14.00
Joe	3	Food	12.00
Joe	4	Beer	19.00
Janet	5	Car	12.00
Joe	3	Food	19.00
Beth	4	Beer	15.00
Janet	5	Beer	19.00
Joe	3	Car	20.00
Joe	4	Beer	16.00
Beth	5	Food	12.00
Beth	3	Beer	16.00
Joe	4	Food	17.00
Joe	5	Beer	14.00
Janet	3	Car	19.00
Joe	4	Food	17.00
Beth	5	Beer	20.00
Janet	3	Food	18.00
Joe	4	Beer	14.00
Joe	5	Food	12.00
Janet	3	Beer	18.00
Janet	4	Car	17.00
Janet	5	Food	12.00

Pivoting the table contents using the ‘Who’ and ‘Week’ fields for the left columns, and the ‘What’ field for the top heading and summing the ‘Amount’ fields for each cell in the new table, gives the following desired result:

Who	Week	Beer	Car	Food
Beth	3	16.00	0.00	0.00
Beth	4	15.00	0.00	17.00
Beth	5	20.00	0.00	12.00
Janet	3	18.00	19.00	18.00
Janet	4	0.00	17.00	0.00
Janet	5	33.00	12.00	12.00
Joe	3	18.00	20.00	31.00
Joe	4	49.00	0.00	34.00
Joe	5	14.00	0.00	12.00

Note that SQL enables you to get the same result presented differently by using the “group by” clause, namely:

```
SELECT who, week, what, SUM(amount) FROM expenses
       GROUP BY who, week, what;
```

However there is no way to get the pivoted layout shown above just using SQL.
Even using embedded SQL programming for some DBMS is not quite simple and automatic.

The Pivot table type of CONNECT makes doing this much simpler.

Using the PIVOT Tables Type

To get the result shown in the example above, just define it as a new table with the statement:

```
CREATE TABLE pivex
ENGINE=connect table_type=pivot tabname=expenses;
```

You can now use it as any other table, for instance to display the result shown above, just say:

```
SELECT * FROM pivex;
```

The CONNECT implementation of the PIVOT table type does much of the work required to transform the source table:

1. Finding the “Facts” column, by default the last column of the source table. Finding “Facts” or “Pivot” columns work only for table based pivot tables. They do not for view or srcdef based pivot tables, for which they must be explicitly specified.
2. Finding the “Pivot” column, by default the last remaining column.
3. Choosing the aggregate function to use, “SUM” by default.
4. Constructing and executing the “Group By” on the “Facts” column, getting its result in memory.
5. Getting all the distinct values in the “Pivot” column and defining a “Data” column for each.
6. Spreading the result of the intermediate memory table into the final table.

The source table “Pivot” column must not be nullable (there are no such things as a “null” column) The creation are refused even is this nullable column actually does not contain null values.

If a different result is desired, Create Table options are available to change the defaults used by Pivot. For instance if we want to display the average expense for each person and product, spread in columns for each week, use the following statement:

```
CREATE TABLE pivex2
ENGINE=connect table_type=pivot tabname=expenses
option_list='PivotCol=Week,Function=AVG';
```

Now saying:

```
SELECT * FROM pivex2;
```

Will display the resulting table:

Who	What	3	4	5
Beth	Beer	16.00	15.00	20.00
Beth	Food	0.00	17.00	12.00
Janet	Beer	18.00	0.00	16.50
Janet	Car	19.00	17.00	12.00
Janet	Food	18.00	0.00	12.00

Who	What	3	4	5
Joe	Beer	18.00	16.33	14.00
Joe	Car	20.00	0.00	0.00
Joe	Food	15.50	17.00	12.00

Restricting the Columns in a Pivot Table

Let us suppose that we want a Pivot table from expenses summing the expenses for all people and products whatever week it was bought. We can do this just by removing from the pivex table the week column from the column list.

```
ALTER TABLE pivex DROP COLUMN week;
```

The result we get from the new table is:

Who	Beer	Car	Food
Beth	51.00	0.00	29.00
Janet	51.00	48.00	30.00
Joe	81.00	20.00	77.00

Note: Restricting columns is also needed when the source table contains extra columns that should not be part of the pivot table. This is true in particular for key columns that prevent a proper grouping.

PIVOT Create Table Syntax

The Create Table statement for PIVOT tables uses the following syntax:

```
CREATE TABLE pivot_table_name
[(column_definition)]
ENGINE=CONNECT table_type=PIVOT
{tabname='source_table_name' | srcdef='source_table_def'}
[option_list='pivot_table_option_list'];
```

The column definition has two sets of columns:

1. A set of columns belonging to the source table, not including the “facts” and “pivot” columns.
2. “Data” columns receiving the values of the aggregated “facts” columns named from the values of the “pivot” column. They are indicated by the “flag” option.

The **options** and **sub-options** available for Pivot tables are:

Option	Type	Description
Tabname	[DB.]Name	The name of the table to “pivot”. If not set SrcDef must be specified.
SrcDef	SQL_statement	The statement used to generate the intermediate mysql table.
DBname	name	The name of the database containing the source table. Defaults to the current database.
Function*	name	The name of the aggregate function used for the data columns, SUM by default.
PivotCol*	name	Specifies the name of the Pivot column whose values are used to fill the “data” columns having the flag option.
FncCol*	[func()name[)]	Specifies the name of the data “Facts” column. If the form func(name) is used, the aggregate function name is set to func.

Option	Type	Description
Groupby *	Boolean	Set it to True (1 or Yes) if the table already has a GROUP BY format.
Accept*	Boolean	To accept non matching Pivot column values.

- : These options must be specified in the OPTION_LIST.

Additional Access Options

There are four cases where pivot must call the server containing the source table or on which the SrcDef statement must be executed:

1. The source table is not a CONNECT table.
2. The SrcDef option is specified.
3. The source table is on another server.
4. The columns are not specified.

By default, pivot tries to call the currently used server using host=localhost, user=root not using password, and port=3306. However, this may not be what is needed, in particular if the local root user has a password in which case you can get an “access denied” error message when creating or using the pivot table.

Specify the host, user, password and/or port options in the option_list to override the default connection options used to access the source table, get column specifications, execute the generated group by or SrcDef query.

Defining a Pivot Table

There are principally two ways to define a PIVOT table:

1. From an existing table or view.
2. Directly giving the SQL statement returning the result to pivot.

Defining a Pivot Table from a Source Table

The **tabname** standard table option is used to give the name of the source table or view.

For tables, the internal Group By are internally generated, except when the `GROUP BY` option is specified as true. Do it only when the table or view has a valid `GROUP BY` format.

Directly Defining the Source of a Pivot Table in SQL

Alternatively, the internal source can be directly defined using the **SrcDef** option that must have the proper group by format.

As we have seen above, a proper Pivot Table is made from an internal intermediate table resulting from the execution of a `GROUP BY` statement. In many cases, it is simpler or desirable to directly specify this when creating the pivot table. This may be because the source is the result of a complex process including filtering and/or joining tables.

To do this, use the **SrcDef** option, often replacing all other options. For instance, suppose that in the first example we are only interested in weeks 4 and 5. We could of course display it by:

```
SELECT * FROM pivex WHERE week IN (4,5);
```

However, what if this table is a huge table? In this case, the correct way to do it is to define the pivot table as this:

```
CREATE TABLE pivex4
ENGINE=connect table_type=pivot
option_list='PivotCol=what,FncCol=amount'
SrcDef='select who, week, what, sum(amount) from expenses
where week in (4,5) group by who, week, what';
```

If your source table has millions of records and you plan to pivot only a small subset of it, doing so will make a lot of a difference performance wise. In addition, you have entire liberty to use expressions, scalar functions, aliases, join, where and having clauses in your SQL statement. The only constraint is that you are responsible for the result of this statement to have the correct format for the pivot processing.

Using SrcDef also permits to use expressions and/or scalar functions. For instance:

```
CREATE TABLE xpivot (
Who CHAR(10) NOT NULL,
What CHAR(12) NOT NULL,
First DOUBLE(8,2) flag=1,
Middle DOUBLE(8,2) flag=1,
Last DOUBLE(8,2) flag=1)
ENGINE=connect table_type=PIVOT
option_list='PivotCol=wk,FncCol=amnt'
Srcdef='select who, what, case when week=3 then ''First'' when
week=5 then ''Last'' else ''Middle'' end as wk, sum(amount) *
6.56 as amnt from expenses group by who, what, wk';
```

Now the statement:

```
SELECT * FROM xpivot;
```

Will display the result:

Who	What	First	Middle	Last
Beth	Beer	104.96	98.40	131.20
Beth	Food	0.00	111.52	78.72
Janet	Beer	118.08	0.00	216.48
Janet	Car	124.64	111.52	78.72
Janet	Food	118.08	0.00	78.72
Joe	Beer	118.08	321.44	91.84
Joe	Car	131.20	0.00	0.00
Joe	Food	203.36	223.04	78.72

Note 1: to avoid multiple lines having the same fixed column values, it is mandatory in **SrcDef** to place the pivot column at the end of the group by list.

Note 2: in the create statement **SrcDef**, it is mandatory to give aliases **to** the columns containing expressions so they are recognized by the other options.

Note 3: in the **SrcDef** select statement, quotes must be escaped because the entire statement is passed to MariaDB between quotes. Alternatively, specify it between double quotes.

Note 4: We could have left CONNECT do the column definitions. However, because they are defined from the sorted names, the Middle column had been placed at the end of them.

Specifying the Columns Corresponding to the Pivot Column

These columns must be named from the values existing in the “pivot” column. For instance, supposing we have the following *pet* table:

name	race	number
John	dog	2
Bill	cat	1
Mary	dog	1
Mary	cat	1
Lisbeth	rabbit	2
Kevin	cat	2
Kevin	bird	6
Donald	dog	1
Donald	fish	3

Pivoting it using *race* as the pivot column is done with:

```
CREATE TABLE pivet
ENGINE=connect table_type=pivot tabname=pet
option_list='PivotCol=race,groupby=1';
```

This gives the result:

name	dog	cat	rabbit	bird	fish
John	2	0	0	0	0
Bill	0	1	0	0	0
Mary	1	1	0	0	0
Lisbeth	0	0	2	0	0
Kevin	0	2	0	6	0
Donald	1	0	0	0	3

By the way, does this ring a bell? It shows that in a way PIVOT tables are doing the opposite of what OCCUR tables do.

We can alternatively define specifically the table columns but what happens if the Pivot column contains values that is not matching a “data” column? There are three cases depending on the specified options and flags.

First case: If no specific options are specified, this is an error an when trying to display the table. The query will abort with an error message stating that a non-matching value was met. Note that because the column list is established when creating the table, this is prone to occur if some rows containing new values for the pivot column are inserted in the source table. If this happens, you should re-create the table or manually add the new columns to the pivot table.

Second case: The accept option was specified. For instance:

```
CREATE TABLE xpivet2 (
name VARCHAR(12) NOT NULL,
dog INT NOT NULL DEFAULT 0 flag=1,
cat INT NOT NULL DEFAULT 0 flag=1)
ENGINE=connect table_type=pivot tabname=pet
option_list='PivotCol=race,groupby=1,Accept=1';
```

No error are raised and the non-matching values are ignored. This table are displayed as:

name	dog	cat
John	2	0

name	dog	cat
Bill	0	1
Mary	1	1
Lisbeth	0	0
Kevin	0	2
Donald	1	0

Third case: A “dump” column was specified with the flag value equal to 2. All non-matching values are added in this column. For instance:

```
CREATE TABLE xpivet (
name VARCHAR(12) NOT NULL,
dog INT NOT NULL DEFAULT 0 flag=1,
cat INT NOT NULL DEFAULT 0 flag=1,
other INT NOT NULL DEFAULT 0 flag=2)
ENGINE=connect table_type=pivot tabname=pet
option_list='PivotCol=race,groupby=1';
```

This table are displayed as:

name	dog	cat	other
John	2	0	0
Bill	0	1	0
Mary	1	1	0
Lisbeth	0	0	2
Kevin	0	2	6
Donald	1	0	3

It is a good idea to provide such a “dump” column if the source table is prone to be inserted new rows that can have a value for the pivot column that did not exist when the pivot table was created.

Pivoting Big Source Tables

This may sometimes be risky. If the pivot column contains too many distinct values, the resulting table may have too many columns. In all cases the process involved, finding distinct values when creating the table or doing the group by when using it, can be very long and sometimes can fail because of exhausted memory.

Restrictions by a where clause should be applied to the source table when creating the pivot table rather than to the pivot table itself. This can be done by creating an intermediate table or using as source a view or a srcdef option.

All PIVOT tables are read only.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT PROXY Table Type

A `PROXY` table is a table that accesses and reads the data of another table or view. To create a table based on the boys `FIX` table:

```
CREATE TABLE xboy ENGINE=CONNECT
  table_type=PROXY tabname=boys;
```

Simply, `PROXY` being the default type when `TABNAME` is specified:

```
CREATE TABLE xboy ENGINE=CONNECT tabname=boys;
```

Because the boys table can be directly used, what can be the use of a proxy table? Well, its main use is to be internally used by other table types such as TBL, XCOL, OCCUR, or PIVOT. Sure enough, PROXY tables are CONNECT tables, meaning that they can be based on tables of any engines and accessed by table types that need to access CONNECT tables.

Proxy on non-CONNECT Tables

When the sub-table is a view or not a CONNECT table, CONNECT internally creates a temporary CONNECT table of MYSQL type to access it. This connection uses the same default parameters as for a `MYSQL` table. It is also possible to specify them to the `PROXY` table using in the `PROXY` declaration the same `OPTION_LIST` options as for a `MYSQL` table. Of course, it is simpler and more natural to use directly the MYSQL type in this case.

Normally, the default parameters should enable the `PROXY` table to reconnect the server. However, an issue is when the current user was logged using a password. The security protocol prevents CONNECT to retrieve this password and requires it to be given in the `PROXY` table create statement. For instance adding to it:

```
... option_list='Password=mypass';
```

However, it is often not advisable to write in clear a password that can be seen by all user able to see the table declaration by show create table, in particular, if the table is used when the current user is root. To avoid this, a specific user should be created on the local host that are used by proxy tables to retrieve local tables. This user can have minimum grant options, for instance `SELECT` on desired directories, and needs no password. Supposing 'proxy' is such a user, the option list to add are:

```
... option_list='user=proxy';
```

Using a PROXY Table as a View

A `PROXY` table can also be used by itself to modify the way a table is viewed. For instance, a proxy table does not use the indexes of the object table. It is also possible to define its columns with different names or type, to use only some of them or to changes their order. For instance:

```
CREATE TABLE city (
  city VARCHAR(11),
  boy CHAR(12) flag=1,
  birth DATE)
ENGINE=CONNECT tabname=boys;
SELECT * FROM city;
```

This displays:

city	boy	birth
Boston	John	1986-01-25
Boston	Henry	1987-06-07
San Jose	George	1981-08-10
Chicago	Sam	1979-11-22
Dallas	James	1992-05-13
Boston	Bill	1986-09-11

Here we did not have to specify column format or offset because data are retrieved from the boys table, not directly from the boys.txt file. The flag option of the *boy* column indicates that it correspond to the first column of the boys table, the *name* column.

Avoiding PROXY Table Loop

CONNECT is able to test whether a `PROXY`, or `PROXY`-based, table refers directly or indirectly to itself. If a direct reference can tested at table creation, an indirect reference can only be tested when executing a query on the table. However, this is possible only for local tables. When using remote tables or views, a problem can occur if the remote table or the view refers back to one of the local tables of the chain. The same caution should be used than when using FEDERATEDX tables.

Note: All `PROXY` or `PROXY`-based tables are read-only in this version.

Modifying Operations

All INSERT / UPDATE / DELETE operations can be used with proxy tables. However, the same restrictions applying to the source table also apply to the proxy table.

Note: All PROXY and PROXY-based table types are not indexable.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT Table Types - Catalog Tables

A catalog table is one that returns information about another table, or data source. It is similar to what MariaDB commands such as `DESCRIBE` or `SHOW` do. Applied to local tables, this just duplicates what these commands do, with the noticeable difference that they are tables and can be used inside queries as joined tables or inside sub-selects.

But their main interest is to enable querying the structure of external tables that cannot be directly queried with description commands. Let's see an example:

Suppose we want to access the tables from a Microsoft Access database as an ODBC type table. The first information we must obtain is the list of tables existing in this data source. To get it, we will create a catalog table that will return it extracted from the result set of the SQLTables ODBC function:

```
CREATE TABLE tabinfo (
  table_name VARCHAR(128) NOT NULL,
  table_type VARCHAR(16) NOT NULL)
ENGINE=CONNECT table_type=ODBC catfunc=TABLES
CONNECTION='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB;';
```

The SQLTables function returns a result set having the following columns:

Field	Data Type	Null	Info Type	Flag Value
Table_Cat	char(128)	NO	FLD_CAT	17
Table_Name	char(128)	NO	FLD_SCHEM	18
Table_Name	char(128)	NO	FLD_NAME	1
Table_Type	char(16)	NO	FLD_TYPE	2
Remark	char(128)	NO	FLD_REM	5

Note: The Info Type and Flag Value are CONNECT interpretations of this result.

Here we could have omitted the column definitions of the catalog table or, as in the above example, chose the columns returning the name and type of the tables. If specified, the columns must have the exact name of the corresponding SQLTables result set, or be given a different name with the matching flag value specification.

(The Table_Type can be TABLE, SYSTEM TABLE, VIEW, etc.)

For instance, to get the tables we want to use we can ask:

```
SELECT TABLE_NAME FROM tabinfo WHERE table_type = 'TABLE';
```

This will return:

table_name
Categories
Customers
Employees
Products
Shippers
Suppliers

Now we want to create the table to access the CUSTOMERS table. Because CONNECT can retrieve the column description of ODBC tables, it not necessary to specify them in the create table statement:

```
CREATE TABLE Customers ENGINE=CONNECT table_type=ODBC
CONNECTION='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB;';
```

However, if we prefer to specify them (to eventually modify them) we must know what the column definitions of that table are. We can get this information with a catalog table. This is how to do it:

```
CREATE TABLE custinfo ENGINE=CONNECT table_type=ODBC
tabname=customers catfunc=columns
CONNECTION='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB;';
```

Alternatively it is possible to specify what columns of the catalog table we want:

```
CREATE TABLE custinfo (
  column_name CHAR(128) NOT NULL,
  type_name CHAR(20) NOT NULL,
  LENGTH INT(10) NOT NULL flag=7,
  prec SMALLINT(6) NOT NULL flag=9)
  NULLABLE SMALLINT(6) NOT NULL)
ENGINE=CONNECT table_type=ODBC tabname=customers
catfunc=columns
CONNECTION='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB;';
```

To get the column info:

```
SELECT * FROM custinfo;
```

which results in this table:

column_name	type_name	length	prec	nullable
CustomerID	VARCHAR	5	0	1
CompanyName	VARCHAR	40	0	1
ContactName	VARCHAR	30	0	1
ContactTitle	VARCHAR	30	0	1
Address	VARCHAR	60	0	1
City	VARCHAR	15	0	1
Region	VARCHAR	15	0	1

column_name	type_name	length	prec	nullable
PostalCode	VARCHAR	10	0	1
Country	VARCHAR	15	0	1
Phone	VARCHAR	24	0	1
Fax	VARCHAR	24	0	1

Now you can create the CUSTOMERS table as:

```
CREATE TABLE Customers (
  CustomerID VARCHAR(5),
  CompanyName VARCHAR(40),
  ContactName VARCHAR(30),
  ContactTitle VARCHAR(30),
  Address VARCHAR(60),
  City VARCHAR(15),
  Region VARCHAR(15),
  PostalCode VARCHAR(10),
  Country VARCHAR(15),
  Phone VARCHAR(24),
  Fax VARCHAR(24))
ENGINE=CONNECT table_type=ODBC block_size=10
CONNECTION='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB;';
```

Let us explain what we did here: First of all, the creation of the catalog table. This table returns the result set of an ODBC SQLColumns function sent to the ODBC data source. Columns functions always return a data set having some of the following columns, depending on the table type:

Field	Data Type	Null	Info Type	Flag Value	Returned by
Table_Cat*	char(128)	NO	FLD_CAT	17	ODBC, JDBC
Table_Schema*	char(128)	NO	FLD_SCEM	18	ODBC, JDBC
Table_Name	char(128)	NO	FLD_TABNAME	19	ODBC, JDBC
Column_Name	char(128)	NO	FLD_NAME	1	ALL
Data_Type	smallint(6)	NO	FLD_TYPE	2	ALL
Type_Name	char(30)	NO	FLD_TYPENAME	3	ALL
Column_Size*	int(10)	NO	FLD_PREC	4	ALL
Buffer_Length*	int(10)	NO	FLD_LENGTH	5	ALL
Decimal_Digits*	smallint(6)	NO	FLD_SCALE	6	ALL
Radix	smallint(6)	NO	FLD_RADIX	7	ODBC, JDBC, MYSQL
Nullable	smallint(6)	NO	FLD_NULL	8	ODBC, JDBC, MYSQL
Remarks	char(255)	NO	FLD_REM	9	ODBC, JDBC, MYSQL
Collation	char(32)	NO	FLD_CHARSET	10	MYSQL
Key	char(4)	NO	FLD_KEY	11	MYSQL
Default_value	N.A.		FLD_DEFAULT	12	
Privilege	N.A.		FLD_PRIV	13	
Date_fmt	char(32)	NO	FLD_DATEFMT	15	MYSQL
Xpath/Jpath	Varchar(256)	NO	FLD_FORMAT	16	XML/JSON

‘*’: These names have changed since earlier versions of CONNECT.

Note: ALL includes the ODBC, JDBC, MYSQL, DBF, CSV, PROXY, TBL, XML, JSON, XCOL, and WMI table types. More could be added later.

We chose among these columns the ones that were useful for our create statement, using the flag value when we gave them a different name (case insensitive).

The options used in this definition are the same as the one used later for the actual CUSTOMERS data tables except that:

1. The `TABNAME` option is mandatory here to specify what the queried table name is.
2. The `CATFUNC` option was added both to indicate that this is a catalog table, and to specify that we want column information.

Note: If the `TABNAME` option had not been specified, this table would have returned the columns of all the tables defined in the connected data source.

Currently the available `CATFUNC` are:

Function	Specified as:	Applies to table types:
FNC_TAB	tables	ODBC, JDBC, MYSQL
FNC_COL	columns	ODBC, JDBC, MYSQL, DBF, CSV, PROXY, XCOL, TBL, WMI
FNC_DSN	datasourcesdsnsqldatasources	ODBC
FNC_DRIVER	driverssqldrivers	ODBC, JDBC

Note: Only the bold part of the function name specification is required.

The `DATASOURCE` and `DRIVERS` functions respectively return the list of available data sources and ODBC drivers available on the system.

The SQLDataSources function returns a result set having the following columns:

Field	Data Type	Null	Info Type	Flag value
Name	varchar(256)	NO	FLD_NAME	1
Description	varchar(256)	NO	FLD_REM	9

To get the data source, you can do for instance:

```
CREATE TABLE datasources (
ENGINE=CONNECT table_type=ODBC catfunc=DSN;
```

The SQLDrivers function returns a result set having the following columns:

Field	Type	Null	Info Type	Flag value
Description	varchar(128)	YES	FLD_NAME	1
Attributes	varchar(256)	YES	FLD_REM	9

You can get the driver list with:

```
CREATE TABLE drivers
ENGINE=CONNECT table_type=ODBC catfunc=drivers;
```

Another Example, WMI table

To create a catalog table returning the attribute names of a WMI class, use the same table options as the ones used with the normal WMI table plus the additional option ‘catfunc=columns’. If specified, the columns of such a catalog table can be chosen among the following:

Name	Type	Flag	Description
Column_Name	CHAR	1	The name of the property
Data_Type	INT	2	The SQL data type
Type_Name	CHAR	3	The SQL type name
Column_Size	INT	4	The field length in characters
Buffer_Length	INT	5	Depends on the coding
Scale	INT	6	Depends on the type

If you wish to use a different name for a column, set the Flag column option.

For example, before creating the “csprod” table, you could have created the info table:

```
CREATE TABLE CSPRODCOL (
  Column_name CHAR(64) NOT NULL,
  Data_Type INT(3) NOT NULL,
  Type_name CHAR(16) NOT NULL,
  LENGTH INT(6) NOT NULL,
  Prec INT(2) NOT NULL flag=6)
ENGINE=CONNECT table_type='WMI' catfunc=col;
```

Now the query:

```
SELECT * FROM csprodcol;
```

will display the result:

Column_name	Data_Type	Type_name	Length	Prec
Caption	1	CHAR	255	1
Description	1	CHAR	255	1
IdentifyingNumber	1	CHAR	255	1
Name	1	CHAR	255	1
SKUNumber	1	CHAR	255	1
UUID	1	CHAR	255	1
Vendor	1	CHAR	255	1
Version	1	CHAR	255	1

This can help to define the columns of the matching normal table.

Note 1: The column length, for the Info table as well as for the normal table, can be chosen arbitrarily, it just must be enough to contain the returned information.

Note 2: The Scale column returns 1 for text columns (meaning case insensitive); 2 for float and double columns; and 0 for other numeric columns.

Catalog Table result size limit

Because catalog tables are processed like the information retrieved by “Discovery” when table columns are not specified in a Create Table statement, their result set is entirely retrieved and memory allocated.

By default, this allocation is done for a maximum return line number of:

Catfunc	Max lines
Drivers	256

Catfunc	Max lines
Data Sources	512
Columns	20,000
Tables	10,000

When the number of lines retrieved for a table is more than this maximum, a warning is issued by CONNECT. This is mainly prone to occur with columns (and also tables) with some data sources having many tables when the table name is not specified.

If this happens, it is possible to increase the default limit using the MAXRES option, for instance:

```
CREATE TABLE allcols ENGINE=CONNECT table_type=odbc
CONNECTION='DSN=ORACLE_TEST;UID=system;PWD=manager'
option_list='Maxres=110000' catfunc=columns;
```

Indeed, because the entire table result is memorized before the query is executed; the returned value would be limited even on a query such as:

```
SELECT COUNT(*) FROM allcols;
```

This page is licensed: GPLv2

CONNECT Table Types - Data Files

Most of the tables processed by CONNECT are just plain DOS or UNIX data files, logically regarded as tables thanks to the description given when creating the table. This description comes from the CREATE TABLE statement. Depending on the application, these tables can already exist as data files, used as is by CONNECT, or can have been physically made by CONNECT as the result of a `CREATE TABLE ... SELECT ...` and/or INSERT statement(s).

The file *path/name* is given by the `FILE_NAME` option. If it is a relative path/name, it are relative to the database directory, the one containing the table `.FRM` file.

Unless specified, the maturity of file table types is stable.

Multiple File Tables

A **multiple** file table is one that is physically contained in several files of the same type instead of just one. These files are processed sequentially during the process of a query and the result is the same as if all the table files were merged into one. This is great to process files coming from different sources (such as cash register log files) or made at different time periods (such as bank monthly reports) regarded as one table. Note that the operations on such files are restricted to sequential Select and Update; and that VEC multiple tables are not supported by CONNECT. The file list depends on the setting of the **multiple** option of the `CREATE TABLE` statement for that table.

Multiple tables are specified by the option MULTIPLE=*n*, which can take This storage engine has been deprecated.four values:

0	Not a multiple table (the default). This can be used in an ALTER TABLE statement.
1	The table is made from files located in the same directory. The FILE_NAME option is a pattern such as ‘cash*.log’ that all the table file path/names verify.
2	The FILE_NAME gives the name of a file that contains the path/names of all the table files. This file can be made using a DIR table.
3	Like multiple=1 but also including eligible files from the directory sub-folders.

The `FILEID` special column, described here, allows query pruning by filtering the file list or doing some grouping on the files that make a multiple table.

Note: Multiple was not initially implemented for XML tables. This restriction was removed in version 1.02.

Record Format

This characteristic applies to table files handled by the operating system input/output functions. It is **fixed** for table types FIX, BIN, DBF and VEC, and it is variable for DOS, VCT, FMT and some JSON tables.

For fixed tables, most I/O operations are done by block of BLOCK_SIZE rows. This diminishes the number of I/O's and enables block indexing.

Starting with CONNECT version 1.6.6, the BLOCK_SIZE option can also be specified for variable tables. Then, a file similar to the block indexing file is created by CONNECT that gives the size in bytes of each block of BLOCK_SIZE rows. This enables the use of block I/Os and block indexing to variable tables. It also enables CONNECT to return the exact row number for info commands

File Mapping

For file-based tables of reasonable size, processing time can be greatly enhanced under Windows(TM) and some flavors of UNIX or Linux by using the technique of "file mapping", in which a file is processed as if it were entirely in memory. Mapping is specified when creating the table by the use of the `MAPPED=YES` option. This does not apply to tables not handled by system I/O functions (XML and INI).

Big File Tables

Because all files are handled by the standard input/output functions of the operating system, their size is limited to 2GB, the maximum size handled by standard functions. For some table types, CONNECT can deal with files that are larger than 2GB, or prone to become larger than this limit. These are the FIX,BIN and VEC types. To tell connect to use input/output functions dealing with big files, specify the option `huge=1` or `huge=YES` for that table. Note however that CONNECT cannot randomly access tables having more than 2G records.

Compressed File Tables

CONNECT can make and process some tables whose data file is compressed. The only supported compression format is the gzlib format. Zip and zlib formats are supported differently. The table types that can be compressed are DOS,FIX,BIN,CSV and FMT. This can save some disk space at the cost of a somewhat longer processing time.

Some restrictions apply to compressed tables:

- Compressed tables are not indexable.
- Update and partial delete are not supported.

Use the numeric compress option to specify a compressed table:

1. Not compressed
2. Compressed in gzlib format.
3. Made of compressed blocks of block_size records (enabling block indexing)

Relational Formatted Tables

These are based on files whose records represent one table row. Only the column representation within each record can differ. The following relational formatted tables are supported:

- DOS and FIX Table Types
- DBF Table Type
- BIN Table Type
- VEC Table Type
- CSV and FMT Table Types

NoSQL Table Types

These are based on files that do not match the relational format but often represent hierarchical data. CONNECT can handle JSON, INI-CFG, XML and some HTML files..

The way it is done is different from what PostgreSQL does. In addition to including in a table some column values of a specific data format (JSON, XML) to be handled by specific functions, CONNECT can directly use JSON, XML or INI files that can be produced by other applications and this is the table definition that describes where and how the contained information must be retrieved.

This is also different from what MariaDB does with dynamic columns, which is close to what MySQL and PostgreSQL do with the JSON column type.

The following NoSQL types are supported:

- XML Table Type
- JSON Table Type
- INI Table Type

This page is licensed: GPLv2

CONNECT Table Types - OEM: Implemented in an External LIB

Although CONNECT provides a rich set of table types, specific applications may need to access data organized in a way that is not handled by its existing foreign data wrappers (FDW). To handle these cases, CONNECT features an interface that enables developers to implement in C++ the required table wrapper and use it as if it were part of the standard CONNECT table type list. CONNECT can use these additional handlers providing the corresponding external module (dll or shared lib) be available.

To create such a table on an existing handler, use a Create Table statement as shown below.

```
CREATE TABLE xtab (COLUMN definitions)
ENGINE=CONNECT table_type=OEM MODULE='libname'
subtype='MYTYPE' [standard table options]
Option_list='Myopt=foo';
```

The option module gives the name of the DLL or shared library implementing the OEM wrapper for the table type. This library must be located in the plugin directory like all other plugins or UDF's.

This library must export a function *GetMYTYPE*. The option subtype enables CONNECT to have the name of the exported function and to use the new table type. Other options are interpreted by the OEM type and can also be specified within the *option_list* option.

Column definitions can be unspecified only if the external wrapper is able to return this information. For this it must export a function ColMYTYPE returning these definitions in a format acceptable by the CONNECT discovery function.

Which and how options must be specified and the way columns must be defined may vary depending on the OEM type used and should be documented by the OEM type implementer(s).

An OEM Table Example

The OEM table REST described in Adding the REST Feature as a Library Called by an OEM Table permits using REST-like tables with MariaDB binary distributions containing but not enabling the REST table type

Of course, the mongo (dll or so) exporting the GetREST and colREST functions must be available in the plugin directory for all this to work.

Some Currently Available OEM Table Modules and Subtypes

Module	Subtype	Description
libhello	HELLO	A sample OEM wrapper displaying a one line table saying “Hello world”
mongo	MONGO	Enables using tables based on MongoDB collections.
Tabfic	FIC	Handles files having the Windev HyperFile format.
Tabofx	OFC	Handles Open Financial Connectivity files.
Tabofx	QIF	Handles Quicken Interchange Format files.
Cirpack	CRPK	Handles CDR’s from Cirpack UTP’s.
Tabplg	PLG	Access tables from the PlugDB DBMS.

How to implement an OEM handler is out of the scope of this document.

This page is licensed: GPLv2

CONNECT Table Types - Special “Virtual” Tables

The special table types supported by CONNECT are the Virtual table type (VIR - introduced in MariaDB 10.0.15), Directory Listing table type (DIR), the Windows Management Instrumentation Table Type (WMI), and the “Mac Address” type (MAC).

These tables are “virtual tables”, meaning they have no physical data but rather produce result data using specific algorithms. Note that this is close to what Views are, so they could be regarded as special views.

DIR Type

A table of type DIR returns a list of file name and description as a result set. To create a DIR table, use a Create Table statement such as:

```
CREATE TABLE SOURCE (
  DRIVE CHAR(2) NOT NULL,
  PATH VARCHAR(256) NOT NULL,
  FNAME VARCHAR(256) NOT NULL,
  FTYPE CHAR(4) NOT NULL,
  SIZE DOUBLE(12,0) NOT NULL flag=5,
  MODIFIED DATETIME NOT NULL)
ENGINE=CONNECT table_type=DIR file_name='..\\*.cc';
```

When used in a query, the table returns the same file information listing than the system “DIR `*.cc`” statement would return if executed in the same current directory (here supposedly ..)

For instance, the query:

```
SELECT fname, SIZE, modified FROM SOURCE
  WHERE fname like '%handler%';
```

Displays:

fname	size	modified
handler	152177	2011-06-13 18:08:29
sql_handler	25321	2011-06-13 18:08:31

Note: the important item in this table is the flag option value (set sequentially from 0 by default) because it determines which particular information item is returned in the column:

Flag value	Information
0	The disk drive (Windows)
1	The file path
2	The file name
3	The file type
4	The file attribute
5	The file size
6	The last write access date
7	The last read access date
8	The file creation date

The Subdir option

When specified in the create table statement, the subdir option indicates to list, in addition to the files contained in the specified directory, all the files verifying the filename pattern that are contained in sub-directories of the specified directory. For instance, using:

```
CREATE TABLE DATA (
  PATH VARCHAR(256) NOT NULL flag=1,
  FNAME VARCHAR(256) NOT NULL,
  FTYPE CHAR(4) NOT NULL,
  SIZE DOUBLE(12,0) NOT NULL flag=5)
ENGINE=CONNECT table_type=DIR file_name='*.frm'
option_list='subdir=1';

SELECT PATH, COUNT(*), SUM(SIZE) FROM DATA GROUP BY PATH;
```

You will get the following result set showing how many tables are created in the MariaDB databases and what is the total length of the FRM files:

path	count(*)	sum(size)
\CommonSource\mariadb-5.2.7\sql\data\connect\	30	264469
\CommonSource\mariadb-5.2.7\sql\data\mysql\	23	207168
\CommonSource\mariadb-5.2.7\sql\data\test\	22	196882

The Nodir option (Windows)

The Boolean Nodir option can be set to false (0 or no) to add directories that match the file name pattern from the listed files (it is true by default). This is an addition to CONNECT version 1.6. Previously, directory names matching pattern were listed on Windows. Directories were and are never listed on Linux.

Note: The way file names are retrieved makes positional access to them impossible. Therefore, DIR tables cannot be indexed or sorted when it is done using positions.

Be aware, in particular when using the subdir option, that queries on DIR tables are slow and can last almost forever if made on a directory that contains a great number of files in it and its sub-directories.

dir tables can be used to populate a list of files used to create a multiple=2 table. However, this is not as useful as it was when the multiple 3 did not exist.

Windows Management Instrumentation Table Type “WMI”

Note: This table type is available on Windows only.

WMI provides an operating system interface through which instrumented components provide information. Some Microsoft tools to retrieve information through WMI are the WMIC console command and the WMI CMI Studio application.

The CONNECT WMI table type enables administrators and operators not capable of scripting or programming on top of WMI to enjoy the benefit of WMI without even learning about it. It permits to present this information as tables that can be queried, transformed, copied in documents or other tables.

To create a WMI table displaying information coming from a WMI provider, you must provide the namespace and the class name that characterize the information you want to retrieve. The best way to find them is to use the WMI CIM Studio that have tools to browse namespaces and classes and that can display the names of the properties of that class.

The column names of the tables must be the names (case insensitive) of the properties you want to retrieve. For instance:

```
CREATE TABLE ALIAS (
  friendlyname CHAR(32) NOT NULL,
  target CHAR(50) NOT NULL)
ENGINE=CONNECT table_type='WMI'
option_list='Namespace=root\\cli,Class=Msft_CliAlias';
```

WMI tables returns one row for each instance of the related information. The above example is handy to get the class equivalent of the alias of the WMIC command and also to have a list of many classes commonly used.

Because most of the useful classes belong to the ‘root\cimv2’ namespace, this is the default value for WMI tables when the namespace is not specified. Some classes have many properties whose name and type may not be known when creating the table. To find them, you can use the WMI CMI Studio application but his are rarely required because CONNECT is able to retrieve them.

Actually, the class specification also has default values for some namespaces. For the ‘root\cli’ namespace the class name defaults to ‘Msft_CliAlias’ and for the ‘root_cimv2’ namespace the class default value is ‘Win32_ComputerSystemProduct’. Because many class names begin with ‘Win32_’ it is not necessary to say it and specifying the class as ‘Product’ will effectively use class ’Win32_Product’.

For example if you define a table as:

```
CREATE TABLE CSPROD ENGINE=CONNECT table_type='WMI';
```

It will return the information on the current machine, using the class ComputerSystemProduct of the CIMV2 namespace. For instance:

```
SELECT * FROM csprod;
```

Will return a result such as:

Column	Row 1
Caption	Computer system product
Description	Computer system product
IdentifyingNumber	LXAP50X32982327A922300
Name	Aspire 8920
SKUNumber	
UUID	00FC523D-B8F7-DC12-A70E-00B0D1A46136
Vendor	Acer

Column	Row 1
Version	Aspire 8920

Note: This is a transposed display that can be obtained with some GUI.

Getting Column Information

An issue, when creating a WMI table, is to make its column definition. Indeed, even when you know the namespace and the class for the wanted information, it is not easy to find what are the names and types of its properties. However, because CONNECT can retrieve this information from the WMI provider, you can simply omit defining columns and CONNECT will do the job.

Alternatively, you can get this information using a catalog table (see below).

Performance Consideration

Some WMI providers can be very slow to answer. This is not an issue for those that return few object instances, such as the ones returning computer, motherboard, or Bios information. They generally return only one row (instance). However, some can return many rows, in particular the “CIM_DataFile” class. This is why care must be taken about them.

Firstly, it is possible to limit the allocated result size by using the ‘Estimate’ create table option. To avoid result truncation, CONNECT allocates a result of 100 rows that is enough for almost all tables.The ‘Estimate’ option permits to reduce this size for all classes that return only a few rows, and in some rare case to increase it to avoid truncation.

However, it is not possible to limit the time taken by some WMI providers to answer, in particular the CIM_DATAFILE class. Indeed the Microsoft documentation says about it:

“Avoid enumerating or querying for all instances of CIM_DataFile on a computer because the volume of data is likely to either affect performance or cause the computer to stop responding.”

Sure enough, even a simple query such as:

```
SELECT COUNT(*) FROM cim WHERE drive = 'D:' AND PATH like '\\MariaDB\\%';
```

is prone to last almost forever (probably due to the LIKE clause). This is why, when not asking for some specific items, you should consider using the DIR table type instead.

Syntax of WMI queries

Queries to WMI providers are done using the WQL language, not the SQL language. CONNECT does the job of making the WQL query. However, because of the restriction of the WQL syntax, the WHERE clause are generated only when respecting the following restrictions:

1. No function.
2. No comparison between two columns.
3. No expression (currently a CONNECT restriction)
4. No BETWEEN and IN predicates.

Filtering with WHERE clauses not respecting these conditions will still be done by MariaDB only, except in the case of CIM_Datafile class for the reason given above.

However, there is one point that is not covered yet, the syntax used to specify dates in queries. WQL does not recognize dates as number items but translates them to its internal format dates specified as text. Many formats are recognized as described in the Microsoft documentation but only one is useful because common to WQL and MariaDB SQL. Here is an example of a query on a table named “cim” created by:

```
CREATE TABLE cim (
  Name VARCHAR(255) NOT NULL,
  LastModified DATETIME NOT NULL)
ENGINE=CONNECT table_type='WMI'
option_list='class=CIM_DataFile,estimate=5000';
```

The date must be specified with the format in which CIM DATETIME values are stored (WMI uses the date and time formats defined by the Distributed Management Task Force).

```
SELECT * FROM cim WHERE drive = 'D:' AND PATH = '\\PlugDB\\Bin\\'
     and lastmodified > '20120415000000.000000+120';
```

This syntax must be strictly respected. The text has the format:

```
yyyymmddHHMMSS.mmmmmmsUUU
```

It is: year, month, day, hour, minute, second, millisecond, and signed minute deviation from UTC. This format is locale-independent so you can write a query that runs on any machine.

Note 1: The WMI table type is available only in Windows versions of CONNECT.

Note 2: WMI tables are read only.

Note 3: WMI tables are not indexable.

Note 4: WMI consider all strings as case insensitive.

MAC Address Table Type “MAC”

Note: This table type is available on Windows only.

This type is used to display various general information about the computer and, in particular, about its network cards. To create such a table, the syntax to use is:

```
CREATE TABLE tabname (COLUMN definition)
ENGINE=CONNECT table_type=MAC;
```

Column names can be freely chosen because their signification, i.e. the values they will display, comes from the specified Flag option. The valid values for Flag are:

Flag	Valeur	Type
1	Host name	varchar(132)
2	Domain	varchar(132)
3	DNS address	varchar(24)
4	Node type	int(1)
5	Scope ID	varchar(256)
6	Routing	int(1)
7	Proxy	int(1)
8	DNS	int(1)
10	Name	varchar(260)
11	Description	varchar(132)
12	MAC address	char(24)
13	Type	int(3)
14	DHCP	int(1)

Flag	Valeur	Type
15	IP address	char(16)
16	SUBNET mask	char(16)
17	GATEWAY	char(16)
18	DHCP server	char(16)
19	Have WINS	int(1)
20	Primary WINS	char(16)
21	Secondary WINS	char(16)
22	Lease obtained	datetime
23	Lease expires	datetime

Note: The information of columns having a Flag value less than 10 are unique for the computer, the other ones are specific to the network cards of the computer.

For instance, you can define the table *macaddr* as:

```
CREATE TABLE MACADDR (
  Host VARCHAR(132) flag=1,
  Card VARCHAR(132) flag=11,
  Address CHAR(24) flag=12,
  IP CHAR(16) flag=15,
  Gateway CHAR(16) flag=17,
  Lease DATETIME flag=23)
ENGINE=CONNECT table_type=MAC;
```

If you execute the query:

```
SELECT host, address, ip, gateway, lease FROM MACADDR;
```

It will return, for example:

Host	Address	IP	Gateway	Lease
OLIVIER	00-A0-D1-A4-61-36	0.0.0.0	0.0.0.0	1970-01-01 00:00:00
OLIVIER	00-1D-E0-9B-90-0B	192.168.0.10	192.168.0.254	2011-09-18 10:28:5

This page is licensed: GPLv2

CONNECT Table Types - VIR

VIR Type

A VIR table is a virtual table having only Special or Virtual columns. Its only property is its “size”, or cardinality, meaning the number of virtual rows it contains. It is created using the syntax:

```
CREATE TABLE name [coldef] ENGINE=CONNECT TABLE_TYPE=VIR
[BLOCK_SIZE=n];
```

The optional `BLOCK_SIZE` option gives the size of the table, defaulting to 1 if not specified. When its columns are not specified, it is almost equivalent to a SEQUENCE table “seq_1_to_Size”.

Displaying constants or expressions

Many DBMS use a no-column one-line table to do this, often call “dual”. MySQL and MariaDB use syntax where no table is specified. With CONNECT, you can achieve the same purpose with a virtual table, with the noticeable

advantage of being able to display several lines:

```
CREATE TABLE virt ENGINE=CONNECT table_type=VIR block_size=10;
SELECT concat('The square root of ', n, ' is') what,
round(sqrt(n),16) value FROM virt;
```

This will return:

what	value
The square root of 1 is	1.0000000000000000
The square root of 2 is	1.4142135623730951
The square root of 3 is	1.7320508075688772
The square root of 4 is	2.0000000000000000
The square root of 5 is	2.2360679774997898
The square root of 6 is	2.4494897427831779
The square root of 7 is	2.6457513110645907
The square root of 8 is	2.8284271247461903
The square root of 9 is	3.0000000000000000
The square root of 10 is	3.1622776601683795

What happened here? First of all, unlike Oracle “dual” tableS that have no columns, a MariaDB table must have at least one column. By default, CONNECT creates VIR tables with one special column. This can be seen with the SHOW CREATE TABLE statement:

```
CREATE TABLE `virt` (
`n` INT(11) NOT NULL `SPECIAL`=ROWID,
PRIMARY KEY (`n`)
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='VIR'
`BLOCK_SIZE`=10
```

This special column is called “n” and its value is the row number starting from 1. It is purely a virtual table and no data file exists corresponding to it and to its index.
It is possible to specify the columns of a VIR table but they must be CONNECT special columns or virtual columns. For instance:

```
CREATE TABLE virt2 (
n INT KEY NOT NULL special=ROWID,
sig1 BIGINT AS ((n*(n+1))/2) virtual,
sig2 BIGINT AS(((2*n+1)*(n+1)*n)/6) virtual)
ENGINE=CONNECT table_type=VIR block_size=10000000;
SELECT * FROM virt2 LIMIT 995, 5;
```

This table shows the sum and the sum of the square of the n first integers:

n	sig1	sig2
996	496506	329845486
997	497503	330839495
998	498501	331835499
999	499500	332833500
1000	500500	333833500

Note that the size of the table can be made very big as there no physical data. However, the result should be limited in the queries. For instance:

```
SELECT * FROM virt2 WHERE n = 1664510;
```

Such a query could last very long if the rowid column were not indexed. Note that by default,
CONNECT declares the “n” column as a primary key. Actually, VIR tables can be indexed but only on the ROWID (or ROWNUM) columns of the table. This is a virtual index for which no data is stored.

Generating a Table filled with constant values

An interesting use of virtual tables, which often cannot be achieved with a table of any other type, is to generate a table containing constant values.
This is easily done with a virtual table. Let us define the table FILLER as:

```
CREATE TABLE filler ENGINE=CONNECT table_type=VIR block_size=5000000;
```

Here we choose a size larger than the biggest table we want to generate. Later if we need a table pre- filled with default and/or null values, we can do for example:

```
CREATE TABLE tp (
id INT(6) KEY NOT NULL,
name CHAR(16) NOT NULL,
salary FLOAT(8,2));
INSERT INTO tp SELECT n, 'unknown', NULL FROM filler WHERE n <= 10000;
```

This will generate a table having 10000 rows that can be updated later when needed. Note that a SEQUENCE table could have been used here instead of FILLING .

VIR tables vs. SEQUENCE tables

With just its default column, a VIR table is almost equivalent to a SEQUENCE table. The syntax used is the main difference, for instance:

```
SELECT * FROM seq_100_to_150_step_10;
```

can be obtained with a VIR table (of size >= 15) by:

```
SELECT n*10 FROM vir WHERE n BETWEEN 10 AND 15;
```

Therefore, the main difference is to be able to define the columns of VIR tables. Unfortunately, there are currently many limitations to virtual columns that hopefully should be removed in the future.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT TBL Table Type: Table List

This type allows defining a table as a list of tables of any engine and type. This is more flexible than multiple tables that must be all of the same file type. This type does, but is more powerful than, what is done with the MERGE engine.

The list of the columns of the TBL table may not necessarily include all the columns of the tables of the list. If the name of some columns is different in the sub-tables, the column to use can be specified by its position given by the `FLAG` option of the column. If the `ACCEPT` option is set to true (Y or 1) columns that do not exist in some of the sub-tables are accepted and their value are null or pseudo-null (this depends on the nullability of the column) for the tables not having this column. The column types can also be different and an automatic conversion are done if necessary.

Note: If not specified, the column definitions are retrieved from the first table of the table list.

The default database of the sub-tables is the current database or if not, can be specified in the DBNAME option. For the tables that are not in the default database, this can be specified in the table list. For instance, to create a table based on the French table *employe* in the current database and on the English table *employee* of the *db2* database, the syntax of the create statement can be:

```
CREATE TABLE allemp (
  SERIALNO char(5) NOT NULL flag=1,
  NAME varchar(12) NOT NULL flag=2,
  SEX smallint(1),
  TITLE varchar(15) NOT NULL flag=3,
  MANAGER char(5) DEFAULT NULL flag=4,
  DEPARTMENT char(4) NOT NULL flag=5,
  SECRETARY char(5) DEFAULT NULL flag=6,
  SALARY double(8,2) NOT NULL flag=7)
ENGINE=CONNECT table_type=TBL
table_list='employe,db2.employee' option_list='Accept=1';
```

The search for columns in sub tables is done by name and, if they exist with a different name, by their position given by a not null `FLAG` option. Column *sex* exists only in the English table (`FLAG` is `0`). Its values will null value for the French table.

For instance, the query:

```
SELECT name, sex, title, salary FROM allemp WHERE department = 318;
```

Can reply:

NAME	SEX	TITLE	SALARY
BARBOUD	NULL	VENDEUR	9700.00
MARCHANT	NULL	VENDEUR	8800.00
MINIARD	NULL	ADMINISTRATIF	7500.00
POUPIN	NULL	INGENIEUR	7450.00
ANTERPE	NULL	INGENIEUR	6850.00
LOULOUTE	NULL	SECRETAIRE	4900.00
TARTINE	NULL	OPERATRICE	2800.00
WERTHER	NULL	DIRECTEUR	14500.00
VOITURIN	NULL	VENDEUR	10130.00
BANCROFT	2	SALESMAN	9600.00
MERCHANT	1	SALESMAN	8700.00
SHRINKY	2	ADMINISTRATOR	7500.00
WALTER	1	ENGINEER	7400.00
TONGHO	1	ENGINEER	6800.00
HONEY	2	SECRETARY	4900.00
PLUMHEAD	2	TYPIST	2800.00
WERTHER	1	DIRECTOR	14500.00
WHEELFOR	1	SALESMAN	10030.00

The first 9 rows, coming from the French table, have a null for the *sex* value. They would have 0 if the sex column had been created NOT NULL.

Subtables of Non-CONNECT Engines

Sub-tables are accessed as PROXY tables. For not CONNECT sub-tables that are accessed via the MySQL API, it is possible like with `PROXY` to change the MYSQL default options. Of course, this will apply to all not CONNECT tables of the list.

Using the TABID Special Column

The TABID special column can be used to see from which table the rows come from and to restrict the access to only some of the sub-tables.

Let us see the following example where t1 and t2 are MyISAM tables similar to the ones given in the `MERGE` description:

```
CREATE TABLE xt1 (
  a INT(11) NOT NULL,
  message CHAR(20))
ENGINE=CONNECT table_type=MYSQL tabname='t1'
option_list='database=test,user=root';

CREATE TABLE xt2 (
  a INT(11) NOT NULL,
  message CHAR(20))
ENGINE=CONNECT table_type=MYSQL tabname='t2'
option_list='database=test,user=root';

CREATE TABLE toto (
  tabname CHAR(8) NOT NULL special='TABID',
  a INT(11) NOT NULL,
  message CHAR(20))
ENGINE=CONNECT table_type=TBL table_list='xt1,xt2';

SELECT * FROM total;
```

The result returned by the `SELECT` statement is:

tabname	a	message
xt1	1	Testing
xt1	2	table
xt1	3	t1
xt2	1	Testing
xt2	2	table
xt2	3	t2

Now if you send the query:

```
SELECT * FROM total WHERE tabname = 'xt2';
```

CONNECT will analyze the where clause and only read the *xt1* table. This can save time if you want to retrieve only a few sub-tables from a TBL table containing many sub-tables.

Parallel Execution

Parallel Execution is currently unavailable until some bugs are fixed.

When the sub-tables are located on different servers, it is possible to execute the remote queries simultaneously instead of sequentially. To enable this, set the thread option to yes.

Additional options available for this table type:

Option	Description
Maxerr	The max number of missing tables in the table list before an error is raised. Defaults to 0.
Accept	If true, missing columns are accepted and return null values. Defaults to false.
Thread	If true, enables parallel execution of remote sub-tables.

These options can be specified in the `OPTION_LIST`.

This page is licensed: GPLv2

CONNECT - Using the TBL and MYSQL Table Types Together

Used together, these types lift all the limitations of the FEDERATED and MERGE engines.

MERGE: Its limitation is obvious, the merged tables must be identical MyISAM tables, and MyISAM is not even the default engine for MariaDB. However, TBL accesses a collection of CONNECT tables, but because these tables can be user specified or internally created MYSQL tables, there is no limitation to the type of the tables that can be merged.

TBL is also much more flexible. The merged tables must not be “identical”, they just should have the columns defined in the TBL table. If the type of one column in a merged table is not the one of the corresponding column of the TBL table, the column value are converted. As we have seen, if one column of the TBL table of the TBL column does not exist in one of the merged table, the corresponding value are set to null. If columns in a sub-table have a different name, they can be accessed by position using the FLAG column option of CONNECT.

However, one limitation of the TBL type regarding MERGE is that TBL tables are currently read-only; INSERT is not supported by TBL. Also, keep using MERGE to access a list of identical MyISAM tables because it are faster, not passing by the MySQL API.

FEDERATED(X): The main limitation of FEDERATED is to access only MySQL/MariaDB tables. The MYSQL table type of CONNECT has the same limitation but CONNECT provides the ODBC table type and JDBC table type that can access tables of any RDBS providing an ODBC or JDBC driver (including MySQL even it is not really useful!)

Another major limitation of FEDERATED is to access only one table. By combining TBL and MYSQL tables, CONNECT enables to access a collection of local or remote tables as one table. Of course the sub-tables can be on different servers. With one SELECT statement, a company manager are able to interrogate results coming from all of his subsidiary computers. This is great for distribution, banking, and many other industries.

Remotely executing complex queries

Many companies or administrations must deal with distributed information. CONNECT enables to deal with it efficiently without having to copy it to a centralized database. Let us suppose we have on some remote network machines_m1, m2, … mn_ some information contained in two tables *t1* and *t2*.

Suppose we want to execute on all servers a query such as:

```
SELECT c1, SUM(c2) FROM t1 a, t2 b WHERE a.id = b.id GROUP BY c1;
```

This raises many problems. Returning the column values of the *t1* and *t2* tables from all servers can be a lot of network traffic. The group by on the possibly huge resulting tables can be a long process. In addition, the join on the *t1* and *t2* tables may be relevant only if the joined tuples belong to the same machine, obliging to add a condition on an additional tabid or servid special column.

All this can be avoided and optimized by forcing the query to be locally executed on each server and retrieving only the small results of the group by queries. Here is how to do it. For each remote machine, create a table that will retrieve the locally executed query. For instance for m1:

```
CREATE TABLE rt1 ENGINE=CONNECT option_list='host=m1'
srcdef='select c1, sum(c2) as sc2 from t1 a, t2 b where a.id = b.id group by c1';
```

Note the alias for the functional column. An alias would be required for the c1 column if its name was different on some machines. The t1 and t2 table names can also be eventually different on the remote machines. The true names must be used in the `SRCDEF` parameter. This will create a set of tables with two columns named c1 and sc2[1].

Then create the table that will retrieve the result of all these tables:

```
CREATE TABLE rtall ENGINE=CONNECT table_type=tbl
table_list='rt1,rt2,…,rtn' option_list='thread=yes';
```

Now you can retrieve the desired result by:

```
SELECT c1, SUM(sc2) FROM rtall;
```

Almost all the work are done on the remote machines, simultaneously thanks to the thread option, making this query super-fast even on big tables placed on many remote machines.

Thread is currently experimental. Use it only for test and report any malfunction on JIRA.

Providing a list of servers

An interesting case is when the query to run on remote machines is the same for all of them. It is then possible to avoid declaring all sub-tables. In this case, the table list option are used to specify the list of servers the `SRCDEF` query must be sent. This is a list of URL's and/or Federated server names.

For instance, supposing that federated servers srv1, srv2, … srv_n_ were created for all remote servers, it are possible to create a tbl table allowing getting the result of a query executed on all of them by:

```
CREATE TABLE qall [column definition]
ENGINE=CONNECT table_type=TBL srcdef='a query'
table_list='srv1,srv2,…,srvn' [option_list='thread=yes'];
```

For instance:

```
CREATE TABLE verall ENGINE=CONNECT table_type=TBL srcdef='select @@version' table_list=',server_one';
SELECT * FROM verall;
```

This reply:

@@**version**
10.0.3-MariaDB-debug
10.0.2-MariaDB

Here the server list specifies a void server corresponding to the local running MariaDB and a federated server named *server_one*.

1. ↑ To generate the columns from the `SRCDEF` query, CONNECT must execute it. This will make sure it is ok. However, if the remote server is not connected yet, or the remote table not existing yet, you can alternatively specify the columns in the create table statement.

This page is licensed: GPLv2

CONNECT VEC Table Type

> **Warning:**
>
> Avoid using this table type in production applications. This file format is specific to CONNECT and may not be supported in future versions.

Tables of type `VEC` are binary files that in some cases can provide good performance on read-intensive query workloads. CONNECT organizes their data on disk as columns of values from the same attribute, as opposed to storing it as rows of tabular records. This organization means that when a query needs to. access only a few columns of a particular table, only those columns need to be read from disk. Conversely, in a row-oriented table, all values in a table are typically read from disk, wasting I/O bandwidth.

CONNECT provides two integral VEC formats, in which each column's data is adjacent.

Integral Vector Formats

In these true vertical formats, the VEC files are made of all the data of the first column, followed by all the data of the second column etc. All this can be in one physical file or each column data can be in a separate file. In the first case, the option max_rows=m, where m is the estimate of the maximum size (number of rows) of the table, must be specified to be able to insert some new records. This leaves an empty space after each column area in which new data can be inserted. In the second case, the "Split" option can be specified[2] at table creation and each column are stored in a file named sequentially from the table file name followed by the rank of the column. Inserting new lines can freely augment such a table.

Differences Between Vector Formats

These formats correspond to different needs. The integral vector format provides the best performance gain. It are chosen when the speed of decisional queries must be optimized.

In the case of a unique file, inserting new data are limited but there will be only one open and close to do. However, the size of the table cannot be calculated from the file size because of the eventual unused space in the file. It must be kept in a header containing the maximum number of rows and the current number of valid rows in the table. To achieve this, specify the option Header=*n* when creating the table. If `n=1` the header are placed at the beginning of the file, if `n=2` it are a separate file with the type '.blk', and if `n=3` the header are place at the end of the file. This last value is provided because batch inserting is sometimes slower when the header is at the beginning of the file. If not specified, the header option will default to 2 for this table type.

On the other hand, the "Split" format with separate files have none of these issues, and is a much safer solution when the table must frequently inserted or shared among several users.

For instance:

```
CREATE TABLE vtab (
a INT NOT NULL,
b CHAR(10) NOT NULL)
ENGINE=CONNECT table_type=VEC file_name='vt.vec';
```

This table, split by default, will have the column values in files vt1.vec and vt2.vec.

For vector tables, the option *block_size=n* is used for block reading and writing; however, to have a file made of blocks of equal size, the internal value of the *max_rows=m* option is eventually increased to become a multiple of n.

Like for BIN tables, numeric values are stored using platform internal layout, the correspondence between column types and internal format being the same than the default ones given above for BIN. However, field formats are not available for VEC tables.

Header Option

This applies to VEC tables that are not split. Because the file size depends on the MAX_ROWS value, CONNECT cannot know how many valid records exist in the file. Depending on the value of the HEADER option, this information is stored in a header that can be placed at the beginning of the file, at the end of the file or in a separate file called fn.blk. The valid values for the HEADER option are:

0	Defaults to 2 for standard tables and to 3 for inward tables.
1	The header is at the beginning of the file.
2	The header is in a separate file.
3	The header is at the end of the file.

The value 2 can be used when dealing with files created by another application with no header. The value 3 makes sometimes inserting in the file faster than when the header is at the beginning of the file.

Note: VEC being a file format specific to CONNECT, no big endian / little endian conversion is provided. These files are not portable between machines using a different byte order setting.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT XCOL Table Type

`XCOL` tables are based on another table or view, like PROXY tables. This type can be used when the object table has a column that contains a list of values.

Suppose we have a *'children'* table that can be displayed as:

name	childlist
Sophie	Vivian, Antony
Lisbeth	Lucy,Charles,Diana
Corinne	
Claude	Marc
Janet	Arthur, Sandra, Peter, John

We can have a different view on these data, where each child are associated with his/her mother by creating an `XCOL` table by:

```
CREATE TABLE xchild (
  mother CHAR(12) NOT NULL,
  child CHAR(12) DEFAULT NULL flag=2
) ENGINE=CONNECT table_type=XCOL tabname='chlist'
option_list='colname=child';
```

The `COLNAME` option specifies the name of the column receiving the list items. This will return from:

```
SELECT * FROM xchild;
```

The requested view:

mother	child
Sophia	Vivian
Sophia	Antony
Lisbeth	Lucy
Lisbeth	Charles

mother	child
Lisbeth	Diana
Corinne	NULL
Claude	Marc
Janet	Arthur
Janet	Sandra
Janet	Peter
Janet	John

Several things should be noted here:

- When the original children field is void, what happens depends on the NULL specification of the “multiple” column. If it is nullable, like here, a void string will generate a NULL value. However, if the column is not nullable, no row are generated at all.
- Blanks after the separator are ignored.
- No copy of the original data was done. Both tables use the same source data.
- Specifying the column definitions in the `CREATE TABLE` statement is optional.

The “multiple” column *child* can be used as any other column. For instance:

```
SELECT * FROM xchild WHERE substr(child,1,1) = 'A';
```

This will return:

Mother	Child
Sophia	Antony
Janet	Arthur

If a query does not involve the “multiple” column, no row multiplication will be done. For instance:

```
SELECT mother FROM xchild;
```

This will just return all the mothers:

mother
Sophia
Lisbeth
Corinne
Claude
Janet

The same occurs with other types of select statements, for instance:

```
SELECT COUNT(*) FROM xchild;      -- returns 5
SELECT COUNT(child) FROM xchild;  -- returns 10
SELECT COUNT(mother) FROM xchild; -- returns 5
```

Grouping also gives different result:

```
SELECT mother, COUNT(*) FROM xchild GROUP BY mother;
```

Replies:

mother	count(*)
Claude	1
Corinne	1
Janet	1
Lisbeth	1
Sophia	1

While the query:

```
SELECT mother, COUNT(child) FROM xchild GROUP BY mother;
```

Gives the more interesting result:

mother	count(child)
Claude	1
Corinne	0
Janet	4
Lisbeth	3
Sophia	2

Some more options are available for this table type:

Option	Description
Sep_char	The separator character used in the “multiple” column, defaults to the comma.
Mult	Indicates the max number of multiple items. It is used to internally calculate the max size of the table and defaults to 10. (To be specified in OPTION_LIST).

Using Special Columns With XCOL

Special columns can be used in XCOL tables. The mostly useful one is ROWNUM that gives the rank of the value in the list of values. For instance:

```
CREATE TABLE xchild2 (
rank INT NOT NULL SPECIAL=ROWID,
mother CHAR(12) NOT NULL,
child CHAR(12) NOT NULL flag=2
) ENGINE=CONNECT table_type=XCOL tabname='chlist' option_list='colname=child';
```

This table are displayed as:

rank	mother	child
1	Sophia	Vivian
2	Sophia	Antony
1	Lisbeth	Lucy
2	Lisbeth	Charles
3	Lisbeth	Diana
1	Claude	Marc
1	Janet	Arthur

rank	mother	child
2	Janet	Sandra
3	Janet	Peter
4	Janet	John

To list only the first child of each mother you can do:

```
SELECT mother, child FROM xchild2 WHERE rank = 1 ;
```

returning:

mother	child
Sophia	Vivian
Lisbeth	Lucy
Claude	Marc
Janet	Arthur

However, note the following pitfall: trying to get the names of all mothers having more than 2 children cannot be done by:

```
SELECT mother FROM xchild2 WHERE rank > 2;
```

This is because with no row multiplication being done, the rank value is always 1. The correct way to obtain this result is longer but cannot use the ROWNUM column:

```
SELECT mother FROM xchild2 GROUP BY mother HAVING COUNT(child) > 2;
```

XCOL Tables Based on Specified Views

Instead of specifying a source table name via the `TABNAME` option, it is possible to retrieve data from a "view" whose definition is given in a new option `SRCDEF` . For instance:

```
CREATE TABLE xsvars ENGINE=CONNECT table_type=XCOL
srcdef='show variables like "optimizer_switch"'
option_list='Colname=Value';
```

Then, for instance:

```
SELECT value FROM xsvars LIMIT 10;
```

This will display something like:

value
index_merge=on
index_merge_union=on
index_merge_sort_union=on
index_merge_intersection=on
index_merge_sort_intersection=off
engine_condition_pushdown=off
index_condition_pushdown=on
derived_merge=on

value
derived_with_keys=on
firstmatch=on

Note: All XCOL tables are read only.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT XML Table Type

Overview

CONNECT supports tables represented by XML files. For these tables, the standard input/output functions of the operating system are not used but the parsing and processing of the file is delegated to a specialized library. Currently two such systems are supported: libxml2, a part of the GNOME framework, but which does not require GNOME and, on Windows, MS-DOM (DOMDOC), the Microsoft standard support of XML documents.

DOMDOC is the default for the Windows version of CONNECT and libxml2 is always used on other systems. On Windows the choice can be specified using the XMLSUP CREATE TABLE list option, for instance specifying `option_list='xmlsup=libxml2'`.

Creating XML Tables

First of all, it must be understood that XML is a very general language used to encode data having any structure. In particular, the tag hierarchy in an XML file describes a tree structure of the data. For instance, consider the file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BIBLIO SUBJECT="XML">
   <BOOK ISBN="9782212090819" LANG="fr" SUBJECT="applications">
      <AUTHOR>
         <FIRSTNAME>Jean-Christophe</FIRSTNAME>
         <LASTNAME>Bernadac</LASTNAME>
      </AUTHOR>
      <AUTHOR>
         <FIRSTNAME>François</FIRSTNAME>
         <LASTNAME>Knab</LASTNAME>
      </AUTHOR>
      <TITLE>Construire une application XML</TITLE>
      <PUBLISHER>
         <NAME>Eyrolles</NAME>
         <PLACE>Paris</PLACE>
      </PUBLISHER>
      <DATEPUB>1999</DATEPUB>
   </BOOK>
   <BOOK ISBN="9782840825685" LANG="fr" SUBJECT="applications">
      <AUTHOR>
         <FIRSTNAME>William J.</FIRSTNAME>
         <LASTNAME>Pardi</LASTNAME>
      </AUTHOR>
      <TRANSLATOR PREFIX="adapté de l'anglais par">
         <FIRSTNAME>James</FIRSTNAME>
         <LASTNAME>Guerin</LASTNAME>
      </TRANSLATOR>
      <TITLE>XML en Action</TITLE>
      <PUBLISHER>
         <NAME>Microsoft Press</NAME>
         <PLACE>Paris</PLACE>
      </PUBLISHER>
      <DATEPUB>1999</DATEPUB>
   </BOOK>
</BIBLIO>
```

It represents data having the structure:

```
<BIBLIO>
                          ___________|_________
                         |                     |
            <BOOK:ISBN,LANG,SUBJECT>           |
          _________________|_________________    |
         |        |         |           |     |
     <AUTHOR> <TITLE> <PUBLISHER> <DATEPUB> |
    ____|____            ___|____          |
   |    |    |          |        |          |
<FIRST> | <LAST>     <NAME>   <PLACE>       |
        |                                    |
     <AUTHOR>                  <BOOK:ISBN,LANG,SUBJECT>
    ____|____          _____________________|_________________
   |         |       |             |         |        |         |
<FIRST>   <LAST>  <AUTHOR> <TRANSLATOR> <TITLE> <PUBLISHER> <DATEPUB>
               _____|_        ___|___           ___|____
              |       |      |       |         |        |
            <FIRST> <LAST> <FIRST> <LAST>     <NAME>   <PLACE>
```

This structure seems at first view far from being tabular. However, modern database management systems, including MariaDB, implement something close to the relational model and work on tables that are structurally not hierarchical but tabular with rows and columns.

Nevertheless, CONNECT can do it. Of course, it cannot guess what you want to extract from the XML structure, but gives you the possibility to specify it when you create the table[1].

Let us take a first example. Suppose you want to make a table from the above document, displaying the node contents.

For this, you can define a table *xsamptag* as:

```
CREATE TABLE xsamptag (
  AUTHOR CHAR(50),
  TITLE CHAR(32),
  TRANSLATOR CHAR(40),
  PUBLISHER CHAR(40),
  DATEPUB INT(4))
ENGINE=CONNECT table_type=XML file_name='Xsample.xml';
```

It are displayed as:

AUTHOR	TITLE	TRANSLATOR	PUBLISHER	DATEPUB
Jean-Christophe Bernadac	Construire une application XML		Eyrolles Paris	1999
William J. Pardi	XML en Action	James Guerin	Microsoft Press Paris	1999

Let us try to understand what happened. By default the column names correspond to tag names. Because this file is rather simple, CONNECT was able to default the top tag of the table as the root node `<BIBLIO>` of the file, and the row tags as the `<BOOK>` children of the table tag. In a more complex file, this should have been specified, as we will see later. Note that we didn't have to worry about the sub-tags such as `<FIRSTNAME>` or `<LASTNAME>` because CONNECT automatically retrieves the entire text contained in a tag and its sub-tags[2].

Only the first author of the first book appears. This is because only the first occurrence of a column tag has been retrieved so the result has a proper tabular structure. We will see later what we can do about that.

How can we retrieve the values specified by attributes? By using a *Coltype* table option to specify the default column type. The value '@' means that column names match attribute names. Therefore, we can retrieve them by creating a table such as:

```
CREATE TABLE xsampattr (
  ISBN CHAR(15),
  LANG CHAR(2),
  SUBJECT CHAR(32))
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
option_list='Coltype=@';
```

This table returns the following:

ISBN	LANG	SUBJECT
9782212090819	fr	applications
9782840825685	fr	applications

Now to define a table that will give us all the previous information, we must specify the column type for each column. Because in the next statement the column type defaults to Node, the *field_format* column parameter was used to indicate which columns are attributes:

From Connect 1.7.0002

```
CREATE TABLE xsamp (
ISBN CHAR(15) xpath='@',
LANG CHAR(2) xpath='@',
SUBJECT CHAR(32) xpath='@',
AUTHOR CHAR(50),
TITLE CHAR(32),
TRANSLATOR CHAR(40),
PUBLISHER CHAR(40),
DATEPUB INT(4))
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK';
```

Before Connect 1.7.0002

```
CREATE TABLE xsamp (
  ISBN CHAR(15) field_format='@',
  LANG CHAR(2) field_format='@',
  SUBJECT CHAR(32) field_format='@',
  AUTHOR CHAR(50),
  TITLE CHAR(32),
  TRANSLATOR CHAR(40),
  PUBLISHER CHAR(40),
  DATEPUB INT(4))
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK';
```

Once done, we can enter the query:

```
SELECT subject, lang, title, author FROM xsamp;
```

This will return the following result:

SUBJECT	LANG	TITLE	AUTHOR
applications	fr	Construire une application XML	Jean-Christophe Bernadac
applications	fr	XML en Action	William J. Pardi

Note that we have been lucky. Because unlike SQL, XML is case sensitive and the column names have matched the node names only because the column names were given in upper case. Note also that the order of the columns in the table could have been different from the order in which the nodes appear in the XML file.

Using Xpaths With XML Tables

Xpath is used by XML to locate and retrieve nodes. The table's main node Xpath is specified by the `tabname` option. If just the node name is given, CONNECT constructs an Xpath such as `'BIBLIO'` in the example above that should retrieve the `BIBLIO` node wherever it is within the XML file.

The row nodes are by default the children of the table node. However, for instance to eliminate some children nodes that are not real row nodes, the row node name can be specified using the `rownode` sub-option of the `option_list` option.

The field_format options we used above can be specified to locate more precisely where and what information to retrieve using an Xpath-like syntax. For instance:

From Connect 1.7.0002

```
CREATE TABLE xsampall (
isbn CHAR(15) xpath='@ISBN',
LANGUAGE CHAR(2) xpath='@LANG',
subject CHAR(32) xpath='@SUBJECT',
authorfn CHAR(20) xpath='AUTHOR/FIRSTNAME',
authorln CHAR(20) xpath='AUTHOR/LASTNAME',
title CHAR(32) xpath='TITLE',
translated CHAR(32) xpath='TRANSLATOR/@PREFIX',
tranfn CHAR(20) xpath='TRANSLATOR/FIRSTNAME',
tranln CHAR(20) xpath='TRANSLATOR/LASTNAME',
publisher CHAR(20) xpath='PUBLISHER/NAME',
LOCATION CHAR(20) xpath='PUBLISHER/PLACE',
YEAR INT(4) xpath='DATEPUB')
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK';
```

Before Connect 1.7.0002

```
CREATE TABLE xsampall (
  isbn CHAR(15) field_format='@ISBN',
  LANGUAGE CHAR(2) field_format='@LANG',
  subject CHAR(32) field_format='@SUBJECT',
  authorfn CHAR(20) field_format='AUTHOR/FIRSTNAME',
  authorln CHAR(20) field_format='AUTHOR/LASTNAME',
  title CHAR(32) field_format='TITLE',
  translated CHAR(32) field_format='TRANSLATOR/@PREFIX',
  tranfn CHAR(20) field_format='TRANSLATOR/FIRSTNAME',
  tranln CHAR(20) field_format='TRANSLATOR/LASTNAME',
  publisher CHAR(20) field_format='PUBLISHER/NAME',
  LOCATION CHAR(20) field_format='PUBLISHER/PLACE',
  YEAR INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK';
```

This very flexible column parameter serves several purposes:

- To specify the tag name, or the attribute name if different from the column name.
- To specify the type (tag or attribute) by a prefix of '@' for attributes.
- To specify the path for sub-tags using the '/' character.

This path is always relative to the current context (the column top node) and cannot be specified as an absolute path from the document root, therefore a leading '/' cannot be used. The path cannot be variable in node names or depth, therefore using '`//`' is not allowed.

The query:

```
SELECT isbn, title, translated, tranfn, tranln, LOCATION FROM
    xsampall WHERE translated IS NOT NULL;
```

replies:

ISBN	TITLE	TRANSLATED	TRANFN	TRANLN	LOCATION
9782840825685	XML en Action	adapté de l'anglais par	James	Guerin	Paris

Libxml2 Default Name Space Issue

An issue with libxml2 is that some files can declare a default name space in their root node. Because Xpath only searches in that name space, the nodes will not be found if they are not prefixed. If this happens, specify the tabname option as an Xpath ignoring the current name space:

```
TABNAME="//*[local-name()='BIBLIO']"
```

This must also be done for the default of specified Xpath of the not attribute columns. For instance:

```
title char(32) field_format="*[local-name()='TITLE']",
```

Note: This raises an error (and is useless anyway) with DOMDOC.

Direct Access on XML Tables

Direct access is available on XML tables. This means that XML tables can be sorted and used in joins, even in the one-side of the join.

However, building a permanent index is not yet implemented. It is unclear whether this can be useful. Indeed, the DOM implementation that is used to access these tables firstly parses the whole file and constructs a node tree in memory. This may often be the longest part of the process, so the use of an index would not be of great value. Note also that this limits the XML files to a reasonable size. Anyway, when speed is important, this table type is not the best to use. Therefore, in these cases, it is probably better to convert the file to another type by inserting the XML table into another table of a more appropriate type for performance.

Accessing Tags With Namespaces

With the Windows DOMDOC support, this can be done using the prefix in the tabname column option and/or xpath column option. For instance, given the file gns.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx xmlns:gns="http:dummy">
<gns:trkseg>
<trkpt lon="-121.9822235107421875" lat="37.3884925842285156">
<gns:ele>6.610851287841797</gns:ele>
<time>2014-04-01T14:54:05.000Z</time>
</trkpt>
<trkpt lon="-121.9821929931640625" lat="37.3885803222656250">
<ele>6.787827968597412</ele>
<time>2014-04-01T14:54:08.000Z</time>
</trkpt>
<trkpt lon="-121.9821624755859375" lat="37.3886299133300781">
<ele>6.771987438201904</ele>
<time>2014-04-01T14:54:10.000Z</time>
</trkpt>
</gns:trkseg>
</gpx>
```

and the defined CONNECT table:

```
CREATE TABLE xgns (
`lon` DOUBLE(21,16) NOT NULL `xpath`='@',
`lat` DOUBLE(20,16) NOT NULL `xpath`='@',
`ele` DOUBLE(21,16) NOT NULL `xpath`='gns:ele',
`time` DATETIME date_format="YYYY-MM-DD 'T' hh:mm:ss '.000Z'"
)
  ENGINE=CONNECT DEFAULT CHARSET=latin1 `table_type`=XML
  `file_name`='gns.xml' tabname='gns:trkseg' option_list='xmlsup=domdoc';
```

```
SELECT * FROM xgns;
```

Displays:

lon	lat	ele	time
-121,982223510742	37,3884925842285	6,6108512878418	01/04/2014 14:54:05
-121,982192993164	37,3885803222656	0	01/04/2014 14:54:08
-121,982162475586	37,3886299133301	0	01/04/2014 14:54:10

Only the prefixed 'ele' tag is recognized.

However, this does not work with the libxml2 support. The solution is then to use a function ignoring the name space:

```
CREATE TABLE xgns2 (
`lon` DOUBLE(21,16) NOT NULL `xpath`='@',
`lat` DOUBLE(20,16) NOT NULL `xpath`='@',
`ele` DOUBLE(21,16) NOT NULL `xpath`="*[local-name()='ele']",
`time` DATETIME date_format="YYYY-MM-DD 'T' hh:mm:ss '.000Z'"
)
  ENGINE=CONNECT DEFAULT CHARSET=latin1 `table_type`=XML
  `file_name`='gns.xml' tabname="*[local-name()='trkseg']" option_list='xmlsup=libxml2';
```

Then :

```
SELECT * FROM xgns2;
```

Displays:

lon	lat	ele	time
-121,982223510742	37,3884925842285	6,6108512878418	01/04/2014 14:54:05
-121,982192993164	37,3885803222656	6.7878279685974	01/04/2014 14:54:08
-121,982162475586	37,3886299133301	6.7719874382019	01/04/2014 14:54:10

This time, all 'ele` tags are recognized. This solution does not work with DOMDOC.

Having Columns Defined by Discovery

It is possible to let the MariaDB discovery process do the job of column specification. When columns are not defined in the CREATE TABLE statement, CONNECT endeavours to analyze the XML file and to provide the column specifications. This is possible only for true XML tables, but not for HTML tables.

For instance, the *xsamp* table could have been created specifying:

```
CREATE TABLE xsamp
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK';
```

Let's check how it was actually specified using the SHOW CREATE TABLE statement:

```
CREATE TABLE `xsamp` (
  `ISBN` CHAR(13) NOT NULL `FIELD_FORMAT`='@',
  `LANG` CHAR(2) NOT NULL `FIELD_FORMAT`='@',
  `SUBJECT` CHAR(12) NOT NULL `FIELD_FORMAT`='@',
  `AUTHOR` CHAR(24) NOT NULL,
  `TRANSLATOR` CHAR(12) DEFAULT NULL,
  `TITLE` CHAR(30) NOT NULL,
  `PUBLISHER` CHAR(21) NOT NULL,
  `DATEPUB` CHAR(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='XML'
  `FILE_NAME`='E:/Data/Xml/Xsample.xml' `TABNAME`='BIBLIO' `OPTION_LIST`='rownode=BOOK';
```

It is equivalent except for the column sizes that have been calculated from the file as the maximum length of the corresponding column when it was a normal value. Also, all columns are specified as type CHAR because XML does not provide information about the node content data type. Nullable is set to true if the column is missing in some rows.

If a more complex definition is desired, you can ask CONNECT to analyse the XPATH up to a given level using the level option in the option list. The level value is the number of nodes that are taken in the XPATH. For instance:

```
CREATE TABLE xsampall
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK,Level=1';
```

This will define the table as:

From Connect 1.7.0002

```
CREATE TABLE `xsampall` (
  `ISBN` CHAR(13) NOT NULL `XPATH`='@',
  `LANG` CHAR(2) NOT NULL `XPATH`='@',
  `SUBJECT` CHAR(12) NOT NULL `XPATH`='@',
  `AUTHOR_FIRSTNAME` CHAR(15) NOT NULL `XPATH`='AUTHOR/FIRSTNAME',
  `AUTHOR_LASTNAME` CHAR(8) NOT NULL `XPATH`='AUTHOR/LASTNAME',
  `TRANSLATOR_PREFIX` CHAR(24) DEFAULT NULL `XPATH`='TRANSLATOR/@PREFIX',
  `TRANSLATOR_FIRSTNAME` CHAR(7) DEFAULT NULL `XPATH`='TRANSLATOR/FIRSTNAME',
  `TRANSLATOR_LASTNAME` CHAR(6) DEFAULT NULL `XPATH`='TRANSLATOR/LASTNAME',
  `TITLE` CHAR(30) NOT NULL,
  `PUBLISHER_NAME` CHAR(15) NOT NULL `XPATH`='PUBLISHER/NAME',
  `PUBLISHER_PLACE` CHAR(5) NOT NULL `XPATH`='PUBLISHER/PLACE',
  `DATEPUB` CHAR(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='XML' `FILE_NAME`='Xsample.xml' `TABNAME`='BIBLIO'
`OPTION_LIST`='rownode=BOOK,Depth=1';
<</SQL>>

BEFORE CONNECT 1.7.0002
<<SQL>>
CREATE TABLE `xsampall` (
  `ISBN` CHAR(13) NOT NULL `FIELD_FORMAT`='@',
  `LANG` CHAR(2) NOT NULL `FIELD_FORMAT`='@',
  `SUBJECT` CHAR(12) NOT NULL `FIELD_FORMAT`='@',
  `AUTHOR_FIRSTNAME` CHAR(15) NOT NULL `FIELD_FORMAT`='AUTHOR/FIRSTNAME',
  `AUTHOR_LASTNAME` CHAR(8) NOT NULL `FIELD_FORMAT`='AUTHOR/LASTNAME',
  `TRANSLATOR_PREFIX` CHAR(24) DEFAULT NULL `FIELD_FORMAT`='TRANSLATOR/@PREFIX',
  `TRANSLATOR_FIRSTNAME` CHAR(7) DEFAULT NULL `FIELD_FORMAT`='TRANSLATOR/FIRSTNAME',
  `TRANSLATOR_LASTNAME` CHAR(6) DEFAULT NULL `FIELD_FORMAT`='TRANSLATOR/LASTNAME',
  `TITLE` CHAR(30) NOT NULL,
  `PUBLISHER_NAME` CHAR(15) NOT NULL `FIELD_FORMAT`='PUBLISHER/NAME',
  `PUBLISHER_PLACE` CHAR(5) NOT NULL `FIELD_FORMAT`='PUBLISHER/PLACE',
  `DATEPUB` CHAR(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='XML' `FILE_NAME`='Xsample.xml'
  `TABNAME`='BIBLIO' `OPTION_LIST`='rownode=BOOK,Level=1';
<</SQL>>

This METHOD can be used AS a quick way TO make a “TEMPLATE” TABLE definition that can later be edited TO
make the desired definition. IN particular, COLUMN NAMES ARE constructed FROM ALL the nodes OF their PATH
IN ORDER TO have DISTINCT COLUMN names. This can be manually edited TO have the desired NAMES, provided
their XPATH IS NOT modified.

TO have a preview OF how columns are DEFINED, you can USE a CATALOG TABLE like this:

<<SQL>>
CREATE TABLE xsacol
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK,Level=1' catfunc=col;
<</SQL>>

AND WHEN asking:

<<SQL>>
SELECT COLUMN_NAME Name, type_name TYPE, column_size SIZE, NULLABLE, xpath FROM xsacol;
<</SQL>>

You GET the description OF what the TABLE columns are:

<<style CLASS="darkheader-nospace-borders">>
|= Name |= TYPE |= SIZE |= NULLABLE |= xpath |
| ISBN | CHAR | 13 | 0 | @ |
| LANG | CHAR | 2 | 0 | @ |
| SUBJECT | CHAR | 12 | 0 | @ |
| AUTHOR_FIRSTNAME | CHAR | 15 | 0 | AUTHOR/FIRSTNAME |
| AUTHOR_LASTNAME | CHAR | 8 | 0 | AUTHOR/LASTNAME |
| TRANSLATOR_PREFIX | CHAR | 24 | 1 | TRANSLATOR/@PREFIX |
| TRANSLATOR_FIRSTNAME | CHAR | 7 | 1 | TRANSLATOR/FIRSTNAME |
| TRANSLATOR_LASTNAME | CHAR | 6 | 1 | TRANSLATOR/LASTNAME |
| TITLE | CHAR | 30 | 0 |  |
```

```
| PUBLISHER_NAME | CHAR | 15 | 0 | PUBLISHER/NAME |
| PUBLISHER_PLACE | CHAR | 5 | 0 | PUBLISHER/PLACE |
| DATEPUB | CHAR | 4 | 0 | |
<</style>>

== WRITE operations ON XML TABLES
You can freely USE the UPDATE, DELETE AND INSERT commands WITH XML tables.
However, you must understand that the format OF the updated OR inserted DATA
follows the specifications OF the TABLE you created, NOT the ones OF the
original SOURCE file. FOR INSTANCE, let us suppose we INSERT a NEW book USING
the //xsamp// TABLE (NOT the //xsampall// TABLE) WITH the command:

<<code lang=mysql INLINE=FALSE>>
INSERT INTO xsamp
  (isbn, lang, subject, author, title, publisher,datepub)
  VALUES ('9782212090529','fr','général','Alain Michard',
         'XML, Language et Applications','Eyrolles Paris',1998);
```

Then if we ask:

```
SELECT subject, author, title, translator, publisher FROM xsamp;
```

Everything seems correct when we get the result:

SUBJECT	AUTHOR	TITLE	TRANSLATOR	PUBLISHER
applications	Jean-Christophe Bernadac	Construire une application XML		Eyrolles Paris
applications	William J. Pardi	XML en Action	James Guerin	Microsoft Press Paris
général	Alain Michard	XML, Language et Applications		Eyrolles Paris

However if we enter the apparently equivalent query on the *xsampall* table, based on the same file:

```
SELECT subject,
concat(authorfn, ' ', authorln) author , title,
concat(tranfn, ' ', tranln) translator,
concat(publisher, ' ', LOCATION) publisher FROM xsampall;
```

this returns an apparently wrong answer:

SUBJECT	AUTHOR	TITLE	TRANSLATOR	PUBLISHER
applications	Jean-Christophe Bernadac	Construire une application XML		Eyrolles Paris
applications	William J. Pardi	XML en Action	James Guerin	Microsoft Press Paris
général		XML, Language et Applications		

What happened here? Simply, because we used the *xsamp* table to do the Insert, what has been inserted within the XML file had the structure described for *xsamp*:

```
<BOOK ISBN="9782212090529" LANG="fr" SUBJECT="général">
      <AUTHOR>Alain Michard</AUTHOR>
      <TITLE>XML, Language et Applications</TITLE>
      <TRANSLATOR></TRANSLATOR>
      <PUBLISHER>Eyrolles Paris</PUBLISHER>
      <DATEPUB>1998</DATEPUB>
   </BOOK>
```

CONNECT cannot “invent” sub-tags that are not part of the *xsamp* table. Because these sub-tags do not exist, the *xsampall* table cannot retrieve the information that should be attached to them. If we want to be able to query the XML file by all the defined tables, the correct way to insert a new book to the file is to use the *xsampall* table, the only one that addresses all the components of the original document:

```
DELETE FROM xsamp WHERE isbn = '9782212090529';

INSERT INTO xsampall (isbn, LANGUAGE, subject, authorfn, authorln,
      title, publisher, LOCATION, YEAR)
   VALUES('9782212090529','fr','général','Alain','Michard',
      'XML, Language et Applications','Eyrolles','Paris',1998);
```

Now the added book, in the XML file, will have the required structure:

```
<BOOK ISBN="9782212090529" LANG="fr" SUBJECT="général">
      <AUTHOR>
         <FIRSTNAME>Alain</FIRSTNAME>
         <LASTNAME>Michard</LASTNAME>
      </AUTHOR>
      <TITLE>XML, Language et Applications</TITLE>
      <PUBLISHER>
         <NAME>Eyrolles</NAME>
         <PLACE>Paris</PLACE>
      </PUBLISHER>
      <DATEPUB>1998</DATEPUB>
   </BOOK>
```

Note: We used a column list in the Insert statements when creating the table to avoid generating a `<TRANSLATOR>` node with sub-nodes, all containing null values (this works on Windows only).

Multiple Nodes in the XML Document

Let us come back to the above example XML file. We have seen that the author node can be “multiple” meaning that there can be more than one author of a book. What can we do to get the complete information fitting the relational model? CONNECT provides you with two possibilities, but is restricted to only one such multiple node per table.

The first and most challenging one is to return as many rows than there are authors, the other columns being repeated as if we had make a join between the author column and the rest of the table. To achieve this, simply specify the
“multiple” node name and the “expand” option when creating the table. For instance, we can create the *xsamp2* table like this:

```
CREATE TABLE xsamp2 (
  ISBN CHAR(15) field_format='@',
  LANG CHAR(2) field_format='@',
  SUBJECT CHAR(32) field_format='@',
  AUTHOR CHAR(40),
  TITLE CHAR(32),
  TRANSLATOR CHAR(32),
  PUBLISHER CHAR(32),
  DATEPUB INT(4))
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO'
option_list='rownode=BOOK,Expand=1,Mulnode=AUTHOR,Limit=2';
```

In this statement, the Limit option specifies the maximum number of values that are expanded. If not specified, it defaults to `10`. Any values above the limit are ignored and a warning message issued[3]. Now you can enter a query such as:

```
SELECT isbn, subject, author, title FROM xsamp2;
```

This will retrieve and display the following result:

ISBN	SUBJECT	AUTHOR	TITLE
9782212090819	applications	Jean-Christophe Bernadac	Construire une application XML

ISBN	SUBJECT	AUTHOR	TITLE
9782212090819	applications	François Knab	Construire une application XML
9782840825685	applications	William J. Pardi	XML en Action
9782212090529	général	Alain Michard	XML, Language et Applications

In this case, this is as if the table had four rows. However if we enter the query:

```
SELECT isbn, subject, title, publisher FROM xsamp2;
```

this time the result are:

ISBN	SUBJECT	TITLE	PUBLISHER
9782212090819	applications	Construire une application XML	Eyrolles Paris
9782840825685	applications	XML en Action	Microsoft Press Paris
9782212090529	général	XML, Language et Applications	Eyrolles Paris

Because the author column does not appear in the query, the corresponding row was not expanded. This is somewhat strange because this would have been different if we had been working on a table of a different type. However, it is closer to the relational model for which there should not be two identical rows (tuples) in a table. Nevertheless, you should be aware of this somewhat erratic behavior. For instance:

```
SELECT COUNT(*) FROM xsamp2;               /* Replies 3 */
SELECT COUNT(author) FROM xsamp2;          /* Replies 4 */
SELECT COUNT(isbn) FROM xsamp2;            /* Replies 3 */
SELECT isbn, subject, title, publisher FROM xsamp2 WHERE author <> '';
```

This last query replies:

ISBN	SUBJECT	TITLE	PUBLISHER
9782212090819	applications	Construire une application XML	Eyrolles Paris
9782212090819	applications	Construire une application XML	Eyrolles Paris
9782840825685	applications	XML en Action	Microsoft Press Paris
9782212090529	général	XML, Language et Applications	Eyrolles Paris

Even though the author column does not appear in the result, the corresponding row was expanded because the multiple column was used in the where clause.

Intermediate Multiple Node

The “multiple” node can be an intermediate node. If we want to do the same expanding with the *xsampall* table, there are nothing more to do. The_xsampall2_ table can be created with:

From Connect 1.7.0002

```
CREATE TABLE xsampall2 (
isbn CHAR(15) xpath='@ISBN',
LANGUAGE CHAR(2) xpath='@LANG',
subject CHAR(32) xpath='@SUBJECT',
authorfn CHAR(20) xpath='AUTHOR/FIRSTNAME',
authorln CHAR(20) xpath='AUTHOR/LASTNAME',
title CHAR(32) xpath='TITLE',
translated CHAR(32) xpath='TRANSLATOR/@PREFIX',
tranfn CHAR(20) xpath='TRANSLATOR/FIRSTNAME',
tranln CHAR(20) xpath='TRANSLATOR/LASTNAME',
publisher CHAR(20) xpath='PUBLISHER/NAME',
LOCATION CHAR(20) xpath='PUBLISHER/PLACE',
YEAR INT(4) xpath='DATEPUB')
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK,Expand=1,Mulnode=AUTHOR,Limit=2';
```

Before Connect 1.7.0002

```
CREATE TABLE xsampall2 (
  isbn CHAR(15) field_format='@ISBN',
  LANGUAGE CHAR(2) field_format='@LANG',
  subject CHAR(32) field_format='@SUBJECT',
  authorfn CHAR(20) field_format='AUTHOR/FIRSTNAME',
  authorln CHAR(20) field_format='AUTHOR/LASTNAME',
  title CHAR(32) field_format='TITLE',
  translated CHAR(32) field_format='TRANSLATOR/@PREFIX',
  tranfn CHAR(20) field_format='TRANSLATOR/FIRSTNAME',
  tranln CHAR(20) field_format='TRANSLATOR/LASTNAME',
  publisher CHAR(20) field_format='PUBLISHER/NAME',
  LOCATION CHAR(20) field_format='PUBLISHER/PLACE',
  YEAR INT(4) field_format='DATEPUB')
ENGINE=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO'
option_list='rownode=BOOK,Expand=1,Mulnode=AUTHOR,Limit=2';
```

The only difference is that the “multiple” node is an intermediate node in the path. The resulting table can be seen with a query such as:

```
SELECT subject, LANGUAGE lang, title, authorfn FIRST, authorln
    LAST, YEAR FROM xsampall2;
```

This query displays:

SUBJECT	LANG	TITLE	FIRST	LAST	YEAR
applications	fr	Construire une application XML	Jean-Christophe	Bernadac	1999
applications	fr	Construire une application XML	François	Knab	1999
applications	fr	XML en Action	William J.	Pardi	1999
général	fr	XML, Language et Applications	Alain	Michard	1998

These composite tables, half array half tree, reserve some surprises for us when updating, deleting from or inserting into them. Insert just cannot generate this structure; if two rows are inserted with just a different author, two book nodes are generated in the XML file. Delete always deletes one book node and all its children nodes even if specified against only one author. Update is more complicated:

```
UPDATE xsampall2 SET authorfn = 'Simon' WHERE authorln = 'Knab';
UPDATE xsampall2 SET YEAR = 2002 WHERE authorln = 'Bernadac';
UPDATE xsampall2 SET authorln = 'Mercier' WHERE YEAR = 2002;
```

After these three updates, the first two responding “Affected rows: 1” and the last one responding “Affected rows: 2”, the last query answers:

subject	lang	title	first	last	year
applications	fr	Construire une application XML	Jean-Christophe	Mercier	2002
applications	fr	Construire une application XML	François	Knab	2002
applications	fr	XML en Action	William J.	Pardi	1999
général	fr	XML, Language et Applications	Alain	Michard	1998

What must be understood here is that the Update modifies node values in the XML file, not cell values in the relational table. The first update worked normally. The second update changed the year value of the book and this shows for the two expanded rows because there is only one DATEPUB node for that book. Because the third update applies to a row having a certain date value, both author names were updated.

Making a List of Multiple Values

Another way to see multiple values is to ask CONNECT to make a comma separated list of the multiple node values. This time, it can only be done if the “multiple” node is not intermediate. For example, we can modify the *xsamp2* table definition by:

```
ALTER TABLE xsamp2 option_list='rownode=BOOK,Mulnode=AUTHOR,Limit=3';
```

This time ‘Expand’ is not specified, and Limit gives the maximum number of items in the list. Now if we enter the query:

```
SELECT isbn, subject, author "AUTHOR(S)", title FROM xsamp2;
```

We will get the following result:

ISBN	SUBJECT	AUTHOR(S)	TITLE
9782212090819	applications	Jean-Christophe Bernadac, François Knab	Construire une application XML
9782840825685	applications	William J. Pardi	XML en Action
9782212090529	général	Alain Michard	XML, Language et Applications

Note that updating the “multiple” column is not possible because CONNECT does not know which of the nodes to update.

This could not have been done with the *xsampall2* table because the author node is intermediate in the path, and making two lists, one of first names and another one of last names would not make sense anyway.

What if a table contains several multiple nodes

This can be handled by creating several tables on the same file, each containing only one multiple node and constructing the desired result using joins.

Support of HTML Tables

Most tables included in HTML documents cannot be processed by CONNECT because the HTML language is often not compatible with the syntax of XML. In particular, XML requires all open tags to be matched by a closing tag while it is sometimes optional in HTML. This is often the case concerning column tags.

However, you can meet tables that respect the XML syntax but have some of the features of HTML tables. For instance:

```
<?xml version="1.0"?>
<Beers>
  <table>
    <th><td>Name</td><td>Origin</td><td>Description</td></th>
    <tr>
      <td><brandName>Huntsman</brandName></td>
      <td><origin>Bath, UK</origin></td>
      <td><details>Wonderful hop, light alcohol</details></td>
    </tr>
    <tr>
      <td><brandName>Tuborg</brandName></td>
      <td><origin>Danmark</origin></td>
      <td><details>In small bottles</details></td>
    </tr>
  </table>
</Beers>
```

Here the different column tags are included in `<td></td>` tags as for HTML tables. You cannot just add this tag in the Xpath of the columns, because the search is done on the first occurrence of each tag, and this would cause this search to fail for all columns except the first one. This case is handled by specifying the *Colnode* table option that gives the name of these column tags, for example:

From Connect 1.7.0002

```
CREATE TABLE beers (
`Name` CHAR(16) xpath='brandName',
`Origin` CHAR(16) xpath='origin',
`Description` CHAR(32) xpath='details')
ENGINE=CONNECT table_type=XML file_name='beers.xml'
tabname='table' option_list='rownode=tr,colnode=td';
```

Before Connect 1.7.0002

```
CREATE TABLE beers (
  `Name` CHAR(16) field_format='brandName',
  `Origin` CHAR(16) field_format='origin',
  `Description` CHAR(32) field_format='details')
ENGINE=CONNECT table_type=XML file_name='beers.xml'
tabname='table' option_list='rownode=tr,colnode=td';
```

The table are displayed as:

Name	Origin	Description
Huntsman	Bath, UK	Wonderful hop, light alcohol
Tuborg	Danmark	In small bottles

However, you can deal with tables even closer to the HTML model. For example the *coffee.htm* file:

```
<TABLE summary="This table charts the number of cups of coffe
                consumed by each senator, the type of coffee (decaf
                or regular), and whether taken with sugar.">
  <CAPTION>Cups of coffee consumed by each senator</CAPTION>
  <TR>
    <TH>Name</TH>
    <TH>Cups</TH>
    <TH>Type of Coffee</TH>
    <TH>Sugar?</TH>
  </TR>
  <TR>
    <TD>T. Sexton</TD>
    <TD>10</TD>
    <TD>Espresso</TD>
    <TD>No</TD>
  </TR>
  <TR>
    <TD>J. Dinnen</TD>
    <TD>5</TD>
    <TD>Decaf</TD>
    <TD>Yes</TD>
  </TR>
</TABLE>
```

Here column values are directly represented by the TD tag text. You cannot declare them as tags nor as attributes. In addition, they are not located using their name but by their position within the row. Here is how to declare such a table to CONNECT:

```
CREATE TABLE coffee (
  `Name` CHAR(16),
  `Cups` INT(8),
  `Type` CHAR(16),
  `Sugar` CHAR(4))
ENGINE=CONNECT table_type=XML file_name='coffee.htm'
tabname='TABLE' header=1 option_list='Coltype=HTML';
```

You specify the fact that columns are located by position by setting the_Coltype_ option to 'HTML'. Each column position (0 based) are the value of the *flag* column parameter that is set by default in sequence. Now we are able to display the table:

Name	Cups	Type	Sugar
T. Sexton	10	Espresso	No
J. Dinnen	5	Decaf	Yes

Note 1: We specified '`header=n`' in the create statement to indicate that the first n rows of the table are not data rows and should be skipped.

Note 2: In this last example, we did not specify the node names using the Rownode and Colnode options because when *Coltype* is set to 'HTML' they default to '`Rownode=TR`' and '`Colnode=TD`'.

Note 3: The *Coltype* option is a word only the first character of which is significant. Recognized values are:

T(ag) or N(ode)	Column names match a tag name (the default).
A(ttribute) or @	Column names match an attribute name.
H(tml) or C(ol) or P(os)	Column are retrieved by their position.

New File Setting

Some create options are used only when creating a table on a new file, for instance, when inserting into a file that does not exist yet. When specified, the 'Header' option will create a header row with the name of the table columns.

This is chiefly useful for HTML tables to be displayed on a web browser.

Some new list-options are used in this context:

Encoding	The encoding of the new document, defaulting to UTF-8.
Attribute	A list of ‘attname=attvalue’ separated by ‘;’ to add to the table node.
HeadAttr	An attribute list to be added to the header row node.

Let us see for instance, the following create statement:

```
CREATE TABLE handlers (
  handler CHAR(64),
  VERSION CHAR(20),
  author CHAR(64),
  description CHAR(255),
  maturity CHAR(12))
ENGINE=CONNECT table_type=XML file_name='handlers.htm'
tabname='TABLE' header=yes
option_list='coltype=HTML,encoding=ISO-8859-1,
attribute=border=1;cellpadding=5,headattr=bgcolor=yellow';
```

Supposing the table file does not exist yet, the first insert into that table, for instance by the following statement:

```
INSERT INTO handlers SELECT plugin_name, plugin_version,
  plugin_author, plugin_description, plugin_maturity FROM
  information_schema.plugins WHERE plugin_type = 'DAEMON';
```

will generate the following file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Created by CONNECT Version 3.05.0005 August 17, 2012 -->
<TABLE border="1" cellpadding="5">
  <TR bgcolor="yellow">
    <TH>handler</TH>
    <TH>version</TH>
    <TH>author</TH>
    <TH>description</TH>
    <TH>maturity</TH>
  </TR>
  <TR>
    <TD>Maria</TD>
    <TD>1.5</TD>
    <TD>Monty Program Ab</TD>
    <TD>Compatibility aliases for the Aria engine</TD>
    <TD>Gamma</TD>
  </TR>
</TABLE>
```

This file can be used to display the table on a web browser (encoding should be `ISO-8859-x`)

handler	version	author	description	maturity
Maria	1.5	Monty Program Ab	Compatibility aliases for the Aria engine	Gamma

Note: The XML document encoding is generally specified in the XML header node and can be different from the DATA_CHARSET, which is always UTF-8 for XML tables. Therefore the table DATA_CHARSET character set should be unspecified, or specified as UTF8. The Encoding specification is useful only for new XML files and ignored for existing files having their encoding already specified in the header node.

Notes

1. ↑ CONNECT does not claim to be able to deal with any XML document. Besides, those that can usefully be processed for data analysis are likely to have a structure that can easily be transformed into a table.
2. ↑ With libxml2, sub tags text can be separated by 0 or several blanks depending on the structure and indentation of the data file.
3. ↑ This may cause some rows to be lost because an eventual where clause on the “multiple” column is applied only on the limited number of retrieved rows.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT Zipped File Tables

Connect can work on table files that are compressed in one or several zip files.

The specific options used when creating tables based on zip files are:

Table Option	Type	Description
ZIPPED	Boolean	Required to be set as true.
ENTRY*	String	The optional name or pattern of the zip entry or entries to be used with the table. If not specified, all entries or only the first one are used depending on the mulentries option setting.
MULENTRIES*	Boolean	True if several entries are part of the table. If not specified, it defaults to false if the entry option is not specified. If the entry option is specified, it defaults to true if the entry name contains wildcard characters or false if it does not.
APPEND*	Boolean	Used when creating new zipped tables (see below)
LOAD*	String	Used when creating new zipped tables (see below)

Options marked with a ‘*’ must be specified in the option list.

Examples of use:

Example 1: Single CSV File Included in a Single ZIP File

Let’s suppose you have a CSV file from which you would create a table by:

```
ENGINE=connect table_type=CSV file_name='E:/Data/employee.csv'
CREATE TABLE emp
... optional COLUMN definition
sep_char=';' header=1;
```

If the CSV file is included in a ZIP file, the CREATE TABLE becomes:

```
CREATE TABLE empzip
... optional column definition
ENGINE=connect table_type=CSV file_name='E:/Data/employee.zip'
sep_char=';' header=1 zipped=1 option_list='Entry=emp.csv';
```

The *file_name* option is the name of the zip file. The *entry* option is the name of the entry inside the zip file. If there is only one entry file inside the zip file, this option can be omitted.

Example 2: Several CSV Files Included in a Single ZIP File

If the table is made from several files such as emp01.csv, emp02.csv, etc., the standard create table would be:

```
CREATE TABLE empmul (
... required column definition
) ENGINE=connect table_type=CSV file_name='E:/Data/emp*.csv'
sep_char=';' header=1 multiple=1;
```

But if these files are all zipped inside a unique zip file, it becomes:

```
CREATE TABLE empzmul
... required column definition
ENGINE=connect table_type=CSV file_name='E:/Data/emp.zip'
sep_char=';' header=1 zipped=1 option_list='Entry=emp*.csv';
```

Here the *entry* option is the pattern that the files inside the zip file must match. If all entry files are ok, the *entry* option can be omitted but the Boolean option *mulentries* must be specified as true.

Example 3: Single CSV File included in Multiple ZIP Files (Without considering subfolders)

If the table is created on several zip files, it is specified as for all other multiple tables:

```
CREATE TABLE zempmul (
... required column definition
) ENGINE=connect table_type=CSV file_name='E:/Data/emp*.zip'
sep_char=';' header=1 multiple=1 zipped=yes
option_list='Entry=employee.csv';
```

Here again the *entry* option is used to restrict the entry file(s) to be used inside the zip files and can be omitted if all are ok.

The column descriptions can be retrieved by the discovery process for table types allowing it. It cannot be done for multiple tables or multiple entries.

A catalog table can be created by adding *catfunc=columns*. This can be used to show the column definitions of multiple tables. *Multiple* must be set to false and the column definitions are the ones of the first table or entry.

This first implementation has some restrictions:

1. Zipped tables are read-only. UPDATE and DELETE are not supported. However, INSERT is supported in a specific way when making tables.
2. The inside files are decompressed into memory. Memory problems may arise with huge files.
3. Only file types that can be handled from memory are eligible for this. This includes DOS, FIX, BIN, CSV, FMT, DBF, JSON, and XML table types, as well as types based on these such as XCOL, OCCUR and PIVOT.

Optimization by indexing or block indexing is possible for table types supporting it. However, it applies to the uncompressed table. This means that the whole table is always uncompressed.

Partitioning is also supported. See how to do it in the section about partitioning.

Creating New Zipped Tables

Tables can be created to access already existing zip files. However, is it also possible to make the zip file from an existing file or table. Two ways are available to make the zip file:

Insert Method

insert can be used to make the table file for table types based on records (this excludes DBF, XML and JSON when pretty is not 0). However, the current implementation of the used package (minizip) does not support adding to an already existing zip entry. This means that when executing an insert statement the inserted records are not added but replace the existing ones. CONNECT protects existing data by not allowing such inserts, Therefore, only three ways are available to do so:

1. Using only one insert statement to make the whole table. This is possible only for small tables and is principally useful when making tests.
2. Making the table from the data of another table. This can be done by executing an “insert into table select * from another_table” or by specifying “as select * from another_table” in the create table statement.
3. Making the table from a file whose format enables to use the “load data infile” statement.

To add a new entry in an existing zip file, specify “append=YES” in the option list. When inserting several entries, use ALTER to specify the required options, for instance:

```
CREATE TABLE znumul (
Chiffre INT(3) NOT NULL,
Lettre CHAR(16) NOT NULL)
ENGINE=CONNECT table_type=CSV
file_name='C:/Data/FMT/mnum.zip' header=1 lrecl=20 zipped=1
option_list='Entry=Num1';
INSERT INTO znumul SELECT * FROM num1;
ALTER TABLE znumul option_list='Entry=Num2,Append=YES';
INSERT INTO znumul SELECT * FROM num2;
ALTER TABLE znumul option_list='Entry=Num3,Append=YES';
INSERT INTO znumul SELECT * FROM num3;
ALTER TABLE znumul option_list='Entry=Num*,Append=YES';
SELECT * FROM znumul;
```

The last ALTER is needed to display all the entries.

File Zipping Method

This method enables to make the zip file from another file when creating the table. It applies to all table types including DBF, XML and JSON. It is specified in the create table statement with the load option:

```
CREATE TABLE XSERVZIP (
NUMERO VARCHAR(4) NOT NULL,
LIEU VARCHAR(15) NOT NULL,
CHEF VARCHAR(5) NOT NULL,
FUNCTION VARCHAR(12) NOT NULL,
NOM VARCHAR(21) NOT NULL)
ENGINE=CONNECT table_type=XML file_name='E:/Xml/perso.zip' zipped=1
option_list='entry=services,load=E:/Xml/serv2.xml';
```

When executing this statement, the *serv2.xml* file are zipped as /perso.zip*. The entry name can be specified or defaults to the source file name.*

If the column descriptions are specified, the table can be used later to read from the zipped table, but they are not used when creating the zip file. Thus, a fake column (there must be one) can be specified and another table created to read the zip file. This one can take advantage of the discovery process to avoid providing the columns description for table types allowing it. For instance:

```
CREATE TABLE mkzq (whatever INT)
ENGINE=connect table_type=DBF zipped=1
file_name='C:/Data/EAUX/dbf/CQUART.ZIP'
option_list='Load=C:/Data/EAUX/dbf/CQUART.DBF';
```

```
CREATE TABLE zquart
ENGINE=connect table_type=DBF zipped=1
file_name='C:/Data/EAUX/dbf/CQUART.ZIP';
```

It is also possible to create a multi-entries table from several files:

```
CREATE TABLE znewcities (
  _id CHAR(5) NOT NULL,
  city CHAR(16) NOT NULL,
  lat DOUBLE(18,6) NOT NULL `FIELD_FORMAT`='loc:[0]',
  lng DOUBLE(18,6) NOT NULL `FIELD_FORMAT`='loc:[1]',
  pop INT(6) NOT NULL,
  state CHAR(2) NOT NULL
) ENGINE=CONNECT TABLE_TYPE=JSON FILE_NAME='E:/Json/newcities.zip' ZIPPED=1 LRECL=1000
OPTION_LIST='Load=E:/Json/city_*.json,mulentries=YES,pretty=0';
```

Here the files to load are specified with wildcard characters and the *mulentries* options must be specified. However, the *entry* option must not be specified, entry names are made from the file names. Provide a fake column description if the files have different column layout, but specific tables will have to be created to read each of them.

ZIP Table Type

A ZIP table type is also available. It is not meant to read the inside files but to display information about the zip file contents. For instance:

```
CREATE TABLE xzipinfo2 (
entry VARCHAR(256)NOT NULL,
cmpsize BIGINT NOT NULL flag=1,
uncsize BIGINT NOT NULL flag=2,
method INT NOT NULL flag=3,
date DATETIME NOT NULL flag=4)
ENGINE=connect table_type=ZIP file_name='E:/Data/Json/cities.zip';
```

This will display the name, compressed size, uncompressed size, and compress method of all entries inside the zip file. Column names are irrelevant; these are flag values that mean what information to retrieve.

It is possible to retrieve this information from several zip files by specifying the multiple option:

```
CREATE TABLE TestZip1 (
entry VARCHAR(260)NOT NULL,
cmpsize BIGINT NOT NULL flag=1,
uncsize BIGINT NOT NULL flag=2,
method INT NOT NULL flag=4,
date DATETIME NOT NULL flag=4,
zipname VARCHAR(256) special='FILEID')
ENGINE=connect table_type=ZIP multiple=1
file_name='C:/Data/Ziptest/CCAM06300_DBF_PART*.zip';
```

Here we added the special column zipname to get the name of the zip file for each entry.

This page is licensed: CC BY-SA / Gnu FDL

Inward and Outward Tables

There are two broad categories of file-based CONNECT tables. Inward and Outward. They are described below.

Outward Tables

Tables are “outward” when their file name is specified in the CREATE TABLE statement using the *file_name* option.

Firstly, remember that CONNECT implements MED (Management of External Data). This means that the “true” CONNECT tables – “outward tables” – are based on data that belongs to files that can be produced by other applications or data imported from another DBMS.

Therefore, their data is “precious” and should not be modified except by specific commands such as INSERT, UPDATE, or DELETE. For other commands such as CREATE, DROP, or ALTER their data is never modified or erased.

Outward tables can be created on existing files or external tables. When they are dropped, only the local description is dropped, the file or external table is not dropped or erased. Also, DROP TABLE does not erase the indexes.

ALTER TABLE produces the following warning, as a reminder:

```
Warning (Code 1105): This is an outward table, table data were not modified.
```

If the specified file does not exist, it is created when data is inserted into the table. If a SELECT is issued before the file is created, the following error is produced:

```
Warning (Code 1105): Open(rb) error 2 on <file_path>: No such file or directory
```

Altering Outward Tables

When an ALTER TABLE is issued, it just modifies the table definition accordingly without changing the data. ALTER can be used safely to, for instance, modify options such as MAPPED, HUGE or READONLY but with extreme care when modifying column definitions or order options because some column options such as FLAG should also be modified or may become wrong.

Changing the table type with ALTER often makes no sense. But many suspicious alterations can be acceptable if they are just meant to correct an existing wrong definition.

Translating a CONNECT table to another engine is fine but the opposite is forbidden when the target CONNECT table is not table based or when its data file exists (because when the target table data cannot be changed and if the source table is dropped, the table data would be lost). However, it can be done to create a new file-based tables when its file does not exist or is void.

Creating or dropping indexes is accepted because it does not modify the table data. However, it is often unsafe to do it with an ALTER TABLE statement that does other modifications.

Of course, all changes are acceptable for empty tables.

Note: Using outward tables requires the FILE privilege.

Inward Tables

A special type of file-based CONNECT tables are “inward” tables. They are file-based tables whose file name is not specified in the CREATE TABLE statement (no *file_name* option).

Their file are located in the current database directory and their name will default to tablename.type where tablename is the table name and type is the table type folded to lower case. When they are created without using a `CREATE TABLE ... SELECT ...` statement, an empty file is made at create time and they can be populated by further inserts.

They behave like tables of other storage engines and, unlike outward CONNECT tables, they are erased when the table is dropped. Of course they should not be read-only to be usable. Even though their utility is limited, they can be used for testing purposes or when the user does not have the FILE privilege.

Altering Inward Tables

One thing to know, because CONNECT builds indexes in a specific way, is that all index modifications are done using an “in-place” algorithm – meaning not using a temporary table. This is why, when indexing is specified in an ALTER TABLE statement containing other changes that cannot be done “in-place”, the statement cannot be executed and raises an error.

Converting an inward table to an outward table, using an `ALTER TABLE` statement specifying a new file name and/or a new table type, is restricted the same way it is when converting a table from another engine to an outward table. However there are no restrictions to convert another engine table to a CONNECT inward table.

This page is licensed: CC BY-SA / Gnu FDL

CONNECT Security

The use of the CONNECT engine requires the FILE privilege for "outward" tables. This should not be an important restriction. The use of CONNECT "outward" tables on a remote server seems of limited interest without knowing the files existing on it and must be protected anyway. On the other hand, using it on the local client machine is not an issue because it is always possible to create locally a user with the `FILE` privilege.

This page is licensed: GPLv2

CONNECT System Variables

> **Warning:**
>
> This storage engine has been deprecated.

This page documents system variables related to the CONNECT storage engine. See Server System Variables for instructions on setting them.

See also the Full list of MariaDB options, system and status variables.

`connect_class_path`

- Description: Java class path
- Command line: `--connect-class-path=value`
- Scope: Global
- Dynamic:
- Data Type: `string`
- Default Value:

`connect_cond_push`

- Description: Enable condition pushdown
- Command line: `--connect-cond-push={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`connect_conv_size`

- Description: The size of the VARCHAR created when converting from a TEXT type. See connect_type_conv.
- Command line: `--connect-conv-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - = MariaDB 10.4.8: `1024`
  - <= MariaDB 10.4.7: `8192`
- Range: `0` to `65500`

`connect_default_depth`

- Description: Default depth used by Json, XML and Mongo discovery.
- Command line: `--connect-default-depth=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `5`
- Range: `-1` to `16`
- Introduced: MariaDB 10.5.7, MariaDB 10.4.16

`connect_default_prec`

- Description: Default precision used for doubles.
- Command line: `--connect-default-prec=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `6`
- Range: `0` to `16`
- Introduced: MariaDB 10.5.9, MariaDB 10.4.18

`connect_enable_mongo`

- Description: Enable the Mongo table type.
- Command line: `--connect-enable-mongo={0|1}`
- Scope: Global, Session
- Dynamic:
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.3.2, MariaDB 10.2.9
- Removed: MariaDB 10.3.3

`connect_exact_info`

- Description: Whether the CONNECT engine should return an exact record number value to information queries. It is OFF by default because this information can take a very long time for large variable record length tables or for remote tables, especially if the remote server is not available. It can be set to ON when exact values are desired, for instance when querying the repartition of rows in a partition table.
- Command line: `--connect-exact-info={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`connect_force_bson`

- Description: Force using BSON for JSON tables. Starting with these releases, the internal way JSON was parsed and handled was changed. The main advantage of the new way is to reduce the memory required to parse JSON (from 6 to 10 times the size of the JSON source to now only 2 to 4 times). However, this is in Beta

mode and JSON tables are still handled using the old mode. To use the new mode, tables should be created with TABLE_TYPE=BSON, or by setting this session variable to 1 or ON. Then, all JSON tables are handled as BSON. This is temporary until the new way replaces the old way by default.

- Command line: `--connect-force-bson={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.5.9, MariaDB 10.4.18

`connect_indx_map`

- Description: Enable file mapping for index files. To accelerate the indexing process, CONNECT makes an index structure in memory from the index file. This can be done by reading the index file or using it as if it was in memory by "file mapping". Set to 0 (file read, the default) or 1 (file mapping).
- Command line: `--connect-indx-map=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`connect_java_wrapper`

- Description: Java wrapper.
- Command line: `--connect-java-wrapper=val`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `wrappers/JdbcInterface`

`connect_json_all_path`

- Description: Discovery to generate json path for all columns if ON (the default) or do not when the path is the column name.
- Command line: `--connect-json-all-path={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.5.7, MariaDB 10.4.16

`connect_json_grp_size`

- Description: Max number of rows for JSON aggregate functions.
- Command line: `--connect-json-grp-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`

- Default Value: `50` (>= Connect 1.7.0003), `10` (<= Connect 1.7.0002)
- Range: `1` to `2147483647`

`connect_json_null`

- Description: Representation of JSON null values.
- Command line: `--connect-json-null=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `<null>`

`connect_jvm_path`

- Description: Path to JVM library.
- Command line: `--connect-jvm_path=value`
- Scope: Global
- Dynamic:
- Data Type: `string`
- Default Value:

`connect_type_conv`

- Description: Determines the handling of TEXT columns.
  - `NO`: The default until Connect 1.06.005, no conversion takes place, and a TYPE_ERROR is returned, resulting in a “not supported” message.
  - `YES`: The default from Connect 1.06.006. The column is internally converted to a column declared as VARCHAR(n), `n` being the value of connect_conv_size.
  - `FORCE` (>= Connect 1.06.006): Also convert ODBC blob columns to TYPE_STRING.
  - `SKIP`: No conversion. When the column declaration is provided via Discovery (meaning the CONNECT table is created without a column description), this column is not generated. Also applies to ODBC tables.
- Command line: `--connect-type-conv=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Valid Values: `NO`, `YES`, `FORCE` or `SKIP`
- Default Value: `YES`

`connect_use_tempfile`

- Description:
  - `NO`: The first algorithm is always used. Because it can cause errors when updating variable record length tables, this value should be set only for testing.
  - `AUTO`: This is the default value. It leaves CONNECT to choose the algorithm to use. Currently it is equivalent to `NO`, except when updating variable record length tables (DOS, CSV or FMT) with file mapping forced to OFF.
  - `YES`: Using a temporary file is chosen with some exceptions. These are when file mapping is ON, for VEC tables and when deleting from DBF tables (soft delete). For variable record length tables, file mapping is forced to OFF.
  - `FORCE`: Like YES but forces file mapping to be OFF for all table types.

    - `TEST` : Reserved for CONNECT development.
- Command line: `--connect-use-tempfile=#`
- Scope: Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `AUTO`

`connect_work_size`

- Description: Size of the CONNECT work area used for memory allocation. Permits allocating a larger memory sub-allocation space when dealing with very large if sub-allocation fails. If the specified value is too big and memory allocation fails, the size of the work area remains but the variable value is not modified and should be reset.
- Command line: `--connect-work-size=#`
- Scope: Global, Session (Session-only from CONNECT 1.03.005)
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `67108864`
- Range: `4194304` upwards, depending on the physical memory size

`connect_xtrace`

- Description: Console trace value. Set to `0` (no trace), or to other values if a console tracing is desired. Note that to test this handler, MariaDB should be executed with the –console parameter because CONNECT prints some error and trace messages on the console. In some Linux versions, this is re-routed into the error log file. Console tracing can be set on the command line or later by names or values. Valid values (from Connect 1.06.006) include:
    - `0` : No trace
    - `YES` or `1` : Basic trace
    - `MORE` or `2` : More tracing
    - `INDEX` or `4` : Index construction
    - `MEMORY` or `8` : Allocating and freeing memory
    - `SUBALLOC` or `16` : Sub-allocating in work area
    - `QUERY` or `32` : Constructed query sent to external server
    - `STMT` or `64` : Currently executing statement
    - `HANDLER` or `128` : Creating and dropping CONNECT handlers
    - `BLOCK` or `256` : Creating and dropping CONNECT objects
    - `MONGO` or `512` : Mongo and REST (from Connect 1.06.0010) tracing :
    - `set global connect_xtrace=0; No trace`
    - `set global connect_xtrace='YES'; By name`
    - `set global connect_xtrace=1; By value`
    - `set global connect_xtrace='QUERY,STMT'; By name`
    - `set global connect_xtrace=96; By value`
    - `set global connect_xtrace=1023; Trace all`
- Command line: `--connect-xtrace=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `set`
- Default Value: `0`
- Valid Values: See description

This page is licensed: CC BY-SA / Gnu FDL

Adding the REST Feature as a Library Called by an OEM Table

If you are using a version of MariaDB that does not support REST, this is how the REST feature can be added as a library called by an OEM table.

Before making the REST OEM module, the Microsoft Casablanca package must be installed as for compiling MariaDB from source.

Even if this module is to be used with a binary distribution, you need some CONNECT source files in order to successfully make it. It is made with four files existing in the version 1.06.0010 of CONNECT: tabrest.cpp, restget.cpp, tabrest.h and mini-global.h. It also needs the CONNECT header files that are included in `tabrest.cpp` and the ones they can include. This can be obtained by going to a recent download site of a version of MariaDB that includes the REST feature, downloading the MariaDB source file `tar.gz` and extracting from it the CONNECT sources files in a directory that are added to the additional source directories if it is not the directory containing the above files.

On Windows, use a recent version of Visual Studio. Make a new empty DLL project and add the source files `tabrest.cpp` and `restget.cpp`. Visual studio should automatically generate all necessary connections to the `cpprest` SDK. Just edit the properties of the project to add the additional include directory (the one where the CONNECT source was downloaded) et the link to the ha_connect.lib of the binary version of MariaDB (in the same directory than `ha_connect.dll` in your binary distribution). Add the preprocessor definition `XML_SUPPORT`. Also set in the linker input page of the project property the Module definition File to the rest.def file (with its full path) also existing in the CONNECT source files. If you are making a debug configuration, make sure that in the C/C++ Code generation page the Runtime library line specifies Multi-threaded Debug DLL (/MDd) or your server crashes when using the feature.

This is not really simple but it is nothing compared with Linux! Someone having made an OEM module for its own application have written:

For whatever reason, g++ / ld on Linux are both extremely picky about what they will and won't consider a *"library"* for linking purposes. In order to get them to recognize and therefore find `ha_connect.so` as a "valid" linkable library, `ha_connect.so` must exist in a directory whose path is in `/etc/ld.so.conf` or `/etc/ld.so.conf.d/ha_connect.conf` *AND* its filename must begin with "lib".

On Fedora, you can make a link to ha_connect.so by:

```
$ sudo ln -s /..path to../ha_connect.so /usr/lib64/libconnect.so
```

This provides a library whose name begins with "lib". It was made in `/usr/lib64/` because it was the directory of the libcpprest.so Casablanca library. This solved the need of a file in `/etc/ld.so.conf.d` as this was already done for the `cpprest` library. Note that the -s parameter is a must, without it all sort of nasty errors are met when using the feature.

Then compile and install the OEM module with:

```
$ makdir oem
$ cd oem
$ makedir Release
$ make -f oemrest.mak
$ sudo cp rest.so /usr/local/mysql/lib/plugin
```

The `oemrest.mak` file:

```
#LINUX
CPP = g++
LD = g++
OD = ./Release/
SD = /home/olivier/MariaDB/server/storage/connect/
CD =/usr/lib64
# flags to compile object files that can be used in a dynamic library
CFLAGS= -Wall -c -O3 -std=c++11 -fPIC -fno-rtti -I$(SD) -DXML_SUPPORT
# Replace -03 by -g for debug
LDFLAGS = -L$(CD) -lcpprest -lconnect

# Flags to create a dynamic library.
DYNLINKFLAGS = -shared
# on some platforms, use '-G' instead.

# REST library's archive file
OEMREST = rest.so

SRCS_CPP = $(SD)tabrest.cpp $(SD)restget.cpp
OBJS_CPP = $(OD)tabrest.o $(OD)restget.o

# top-level rule
all: $(OEMREST)

$(OEMREST): $(OBJS_CPP)
  $(LD) $(OBJS_CPP) $(LDFLAGS) $(DYNLINKFLAGS) -o $@

#CPP Source files
$(OD)tabrest.o:   $(SD)tabrest.cpp   $(SD)mini-global.h $(SD)global.h $(SD)plgdbsem.h $(SD)xtable.h
$(SD)filamtxt.h $(SD)plgxml.h $(SD)tabdos.h  $(SD)tabfmt.h $(SD)tabjson.h $(SD)tabrest.h $(SD)tabxml.h
  $(CPP) $(CFLAGS) -o $@ $(SD)$(*F).cpp
$(OD)restget.o:   $(SD)restget.cpp   $(SD)mini-global.h $(SD)global.h
  $(CPP) $(CFLAGS) -o $@ $(SD)$(*F).cpp

# clean everything
clean:
  $(RM) $(OBJS_CPP) $(OEMREST)
```

The SD and CD variables are the directories of the CONNECT source files and the one containing the libcpprest.so lib. They can be edited to match those on your machine OD is the directory that was made to contain the object files.

A very important flag is -fno-rtti. Without it you would be in big trouble.

The resulting module, for instance rest.so or rest.dll, must be placed in the plugin directory of the MariaDB server. Then, you are able to use NoSQL tables simply replacing in the CREATE TABLE statement the TABLE_TYPE option =JSON or XML by TABLE_TYPE=OEM SUBTYPE=REST MODULE='rest.(so|dll)'. Actually, the module name, here supposedly 'rest', can be anything you like.

The file type is JSON by default. If not, it must be specified like this:

```
OPTION_LIST='Ftype=XML'
```

To be added to the create table statement. For instance:

```
CREATE  TABLE webw
ENGINE=CONNECT TABLE_TYPE=OEM MODULE='Rest.dll' SUBTYPE=REST
FILE_NAME='weatherdata.xml'
HTTP='https://samples.openweathermap.org/data/2.5/forecast?
q=London,us&mode=xml&appid=b6907d289e10d714a6e88b30761fae22'
OPTION_LIST='Ftype=XML,Depth=3,Rownode=weatherdata';
```

Note: this last example returns an XML file whose format was not recognized by old CONNECT versions. It is here the reason of the option ‘Rownode=weatherdata’.

If you have trouble making the module, you can post an issue on JIRA.

This page is licensed: CC BY-SA / Gnu FDL

Compiling JSON UDFs in a Separate Library

Although the JSON UDFs can be included in the CONNECT library module, there are cases when you may need to have them in a separate library.

This is when CONNECT is compiled embedded, or if you want to test or use these UDFs with other MariaDB versions not including them.

To make it, you need to have access to the most recent MariaDB source code. Then, make a project containing these files:

1. jsonudf.cpp
2. json.cpp
3. value.cpp
4. osutil.c
5. plugutil.cpp
6. maputil.cpp
7. jsonutil.cpp

`jsonutil.cpp` is not distributed with the source code, you will have to make it from the following:

```
#include "my_global.h"
#include "mysqld.h"
#include "plugin.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>

#include "global.h"

extern "C" int GetTraceValue(void) { return 0; }
uint GetJsonGrpSize(void) { return 100; }

/***********************************************************************/
/*  These replace missing function of the (not used) DTVAL class.      */
/***********************************************************************/
typedef struct _datpar   *PDTP;
PDTP MakeDateFormat(PGLOBAL, PSZ, bool, bool, int) { return NULL; }
int ExtractDate(char*, PDTP, int, int val[6]) { return 0; }

#ifdef __WIN__
my_bool CloseFileHandle(HANDLE h)
{
    return !CloseHandle(h);
} /* end of CloseFileHandle */

#else  /* UNIX */
my_bool CloseFileHandle(HANDLE h)
{
    return (close(h)) ? TRUE : FALSE;
}  /* end of CloseFileHandle */

int GetLastError()
{
    return errno;
}  /* end of GetLastError */

#endif  // UNIX

/***********************************************************************/
/*  Program for sub-allocating one item in a storage area.             */
/*  Note: This function is equivalent to PlugSubAlloc except that in   */
/*  case of insufficient memory, it returns NULL instead of doing a    */
/*  long jump. The caller must test the return value for error.        */
/***********************************************************************/
void *PlgDBSubAlloc(PGLOBAL g, void *memp, size_t size)
{
  PPOOLHEADER pph;                       // Points on area header.

  if (!memp)    //  Allocation is to be done in the Sarea
    memp = g->Sarea;

  size = ((size + 7) / 8) * 8;  /* Round up size to multiple of 8 */
  pph = (PPOOLHEADER)memp;

  if ((uint)size > pph->FreeBlk) { /* Not enough memory left in pool */
    sprintf(g->Message,
     "Not enough memory in Work area for request of %d (used=%d free=%d)",
            (int)size, pph->To_Free, pph->FreeBlk);
    return NULL;
  } // endif size

  // Do the suballocation the simplest way
  memp = MakePtr(memp, pph->To_Free);   // Points to sub_allocated block
  pph->To_Free += size;                 // New offset of pool free block
  pph->FreeBlk -= size;                 // New size   of pool free block
```

```
  return (memp);
} // end of PlgDBSubAlloc
```

You can create the file by copy/paste from the above.

Set all the additional include directories to the MariaDB include directories used in plugin compiling plus the reference of the storage/connect directories, and compile like any other UDF giving any name to the made library module (I used `jsonudf.dll` on Windows).

Then you can create the functions using this name as the soname parameter.

There are some restrictions when using the UDFs this way:

- The connect_json_grp_size variable cannot be accessed. The group size is set and retrieved using the jsonset_grp_size and jsonget_grp_size functions (previously 100).
- In case of error, warnings are replaced by messages sent to stderr.
- No trace.

This page is licensed: CC BY-SA / Gnu FDL

Making the GetRest Library

To enable the REST feature with binary distributions of MariaDB, the function calling the cpprestsdk package is not included in CONNECT, thus allowing CONNECT normal operation when the `cpprestsdk` package is not installed. Therefore, it must be compiled separately as a library (so or dll) that are loaded by CONNECT when needed.

This library will contain only one file shown here:

```
/************* Restget C++ Program Source Code File (.CPP) *************/
/* Adapted from the sample program of the Casablanca tutorial.         */
/* Copyright Olivier Bertrand 2019.                                    */
/***********************************************************************/
#include <cpprest/filestream.h>
#include <cpprest/http_client.h>

using namespace utility::conversions; // String conversions utilities
using namespace web;                  // Common features like URIs.
using namespace web::http;            // Common HTTP functionality
using namespace web::http::client;    // HTTP client features
using namespace concurrency::streams; // Asynchronous streams

typedef const char* PCSZ;

extern "C" int restGetFile(char* m, bool xt, PCSZ http, PCSZ uri, PCSZ fn);

/***********************************************************************/
/*  Make a local copy of the requested file.                           */
/***********************************************************************/
int restGetFile(char *m, bool xt, PCSZ http, PCSZ uri, PCSZ fn)
{
  int  rc = 0;
  auto fileStream = std::make_shared<ostream>();

  if (!http || !fn) {
        strcpy(m, "Missing http or filename");
        return 2;
  } // endif

    if (xt)
        fprintf(stderr, "restGetFile: fn=%s\n", fn);

  // Open stream to output file.
  pplx::task<void> requestTask = fstream::open_ostream(to_string_t(fn))
    .then([=](ostream outFile) {
      *fileStream= outFile;

            if (xt)
                fprintf(stderr, "Outfile isopen=%d\n", outFile.is_open());

      // Create http_client to send the request.
      http_client client(to_string_t(http));

      if (uri) {
        // Build request URI and start the request.
        uri_builder builder(to_string_t(uri));
        return client.request(methods::GET, builder.to_string());
      } else
        return client.request(methods::GET);
    })

    // Handle response headers arriving.
    .then([=](http_response response) {
            if (xt)
                fprintf(stderr, "Received response status code:%u\n",
                                  response.status_code());

      // Write response body into the file.
      return response.body().read_to_end(fileStream->streambuf());
    })

    // Close the file stream.
    .then([=](size_t n) {
            if (xt)
                fprintf(stderr, "Return size=%zu\n", n);

      return fileStream->close();
```

```
    });

  // Wait for all the outstanding I/O to complete and handle any exceptions
  try {
        if (xt)
            fprintf(stderr, "Waiting\n");

        requestTask.wait();
  } catch (const std::exception &e) {
        if (xt)
            fprintf(stderr, "Error exception: %s\n", e.what());

        sprintf(m, "Error exception: %s", e.what());
        rc= 1;
  } // end try/catch

    if (xt)
        fprintf(stderr, "restget done: rc=%d\n", rc);

  return rc;
} // end of restGetFile
```

This file exists in the source of CONNECT as `restget.cpp`. If you have no access to the source, use your favorite editor to make it by copy/pasting from the above.

Then, on Linux, compile the `GetRest.so` library:

```
g++ -o GetRest.so -O3 -Wall -std=c++11 -fPIC -shared restget.cpp -lcpprest
```

Note: You can replace `-O3` by `-g` to make a debug version.

This library should be placed where it can be accessed. A good place is the directory where the `libcpprest.so` is, for instance `/usr/lib64`. You can move or copy it there.

On windows, using Visual Studio, make an empty win32 dll project named GetRest and add it the above file. Also add it the module definition file `restget.def`:

```
LIBRARY REST
EXPORTS
   restGetFile     @1
```

> **Note:**
>
> This file must be specified in the property linker input page.

Once compiled, the release or debug versions can be copied in the `cpprestsdk` corresponding directories, bin or debug\bin.

That is all. It is a once-off job. Once done, it will work with all new MariaDB versions featuring CONNECT version 1.07.

Note: the xt tracing variable is true when connect_xtrace setting includes the value “MONGO” (512).

Caution: If your server crashes when using this feature, this is likely because the GetRest lib is linked to the wrong cpprestsdk lib (this may only apply on Windows). A Release version of GetRest must be linked to the release version of the `cpprestsdk lib` (cpprest_2_10.dll) but if you make a Debug version of GetRest, make sure it is linked to the Debug version of `cpprestsdk.lib` (or `cpprest_2_10d.dll`). This may be automatic if you use Visual Studio to make the GetRest.dll.

This page is licensed: CC BY-SA / Gnu FDL

OEM Table Example

This is an example showing how an OEM table can be implemented.

The header File `my_global.h`:

```
/*****************************************************************/
/*  Definitions needed by the included files.                    */
/*****************************************************************/
#if !defined(MY_GLOBAL_H)
#define MY_GLOBAL_H
typedef unsigned int uint;
typedef unsigned int uint32;
typedef unsigned short ushort;
typedef unsigned long ulong;
typedef unsigned long DWORD;
typedef char *LPSTR;
typedef const char *LPCSTR;
typedef int BOOL;
#if defined(__WIN__)
typedef void *HANDLE;
#else
typedef int HANDLE;
#endif
typedef char *PSZ;
typedef const char *PCSZ;
typedef unsigned char BYTE;
typedef unsigned char uchar;
typedef long long longlong;
typedef unsigned long long ulonglong;
typedef char my_bool;
struct charset_info_st {};
typedef const charset_info_st CHARSET_INFO;
#define FALSE 0
#define TRUE  1
#define Item char
#define MY_MAX(a,b) ((a>b)?(a):(b))
#define MY_MIN(a,b) ((a<b)?(a):(b))
#endif // MY_GLOBAL_H
```

Note:

This is a fake `my_global.h` that just contains what is useful for the `jmgoem.cpp` source file.

The source File `jmgoem.cpp`:

```
/************* jmgoem C++ Program Source Code File (.CPP) *************/
/* PROGRAM NAME: jmgoem    Version 1.0                                  */
/*  (C) Copyright to the author Olivier BERTRAND          2017         */
/*  This program is the Java MONGO OEM module definition.              */
/***********************************************************************/

/***********************************************************************/
/*  Definitions needed by the included files.                          */
/***********************************************************************/
#include "my_global.h"

/***********************************************************************/
/*  Include application header files:                                  */
/*  global.h    is header containing all global declarations.          */
/*  plgdbsem.h  is header containing the DB application declarations.  */
/*  (x)table.h  is header containing the TDBASE declarations.          */
/*  tabext.h    is header containing the TDBEXT declarations.          */
/*  mongo.h     is header containing the MONGO declarations.           */
/***********************************************************************/
#include "global.h"
#include "plgdbsem.h"
#if defined(HAVE_JMGO)
#include "csort.h"
#include "javaconn.h"
#endif   // HAVE_JMGO
#include "xtable.h"
#include "tabext.h"
#include "mongo.h"

/***********************************************************************/
/*  These functions are exported from the MONGO library.              */
/***********************************************************************/
extern "C" {
  PTABDEF __stdcall GetMONGO(PGLOBAL, void*);
  PQRYRES __stdcall ColMONGO(PGLOBAL, PTOS, void*, char*, char*, bool);
} // extern "C"

/***********************************************************************/
/*  DB static variables.                                               */
/***********************************************************************/
int TDB::Tnum;
int DTVAL::Shift;
#if defined(HAVE_JMGO)
int    CSORT::Limit = 0;
double CSORT::Lg2 = log(2.0);
size_t CSORT::Cpn[1000] = {0};          /* Precalculated cmpnum values */
#if defined(HAVE_JAVACONN)
char *JvmPath = NULL;
char *ClassPath = NULL;
char *GetPluginDir(void)
{return "C:/mongo-java-driver/mongo-java-driver-3.4.2.jar;"
        "C:/MariaDB-10.1/MariaDB/storage/connect/";}
char *GetJavaWrapper(void) {return (char*)"wrappers/Mongo3Interface";}
#else   // !HAVE_JAVACONN
HANDLE JAVAConn::LibJvm;              // Handle to the jvm DLL
CRTJVM JAVAConn::CreateJavaVM;
GETJVM JAVAConn::GetCreatedJavaVMs;
#if defined(_DEBUG)
GETDEF JAVAConn::GetDefaultJavaVMInitArgs;
#endif  //  _DEBUG
#endif  // !HAVE_JAVACONN
#endif   // HAVE_JMGO

/***********************************************************************/
/*  This function returns a Mongo definition class.                    */
/***********************************************************************/
PTABDEF __stdcall GetMONGO(PGLOBAL g, void *memp)
{
```

```
  return new(g, memp) MGODEF;
} // end of GetMONGO

#ifdef NOEXP
/***********************************************************************/
/* Functions to be defined if not exported by the CONNECT version.     */
/***********************************************************************/
bool IsNum(PSZ s)
{
  for (char *p = s; *p; p++)
    if (*p == ']')
      break;
    else if (!isdigit(*p) || *p == '-')
      return false;

  return true;
}   // end of IsNum
#endif

/***********************************************************************/
/*  Return the columns definition to MariaDB.                          */
/***********************************************************************/
PQRYRES __stdcall ColMONGO(PGLOBAL g, PTOS tp, char *tab,
                                               char *db, bool info)
{
#ifdef NOMGOCOL
  // Cannot use discovery
  strcpy(g->Message, "No discovery, MGOColumns is not accessible");
  return NULL;
#else
  return MGOColumns(g, db, NULL, tp, info);
#endif
} // end of ColMONGO
```

The file `mongo.def` : (required only on Windows)

```
LIBRARY     MONGO
EXPORTS
   GetMONGO     @1
   ColMONGO     @2
```

Compiling This OEM

To compile this OEM module, first make the two or three required files by copy/pasting from the above listings.

Even if this module is to be used with a binary distribution, you need some source files in order to successfully compile it. At least the CONNECT header files that are included in `jmgoem.cpp` and the ones they can include. This can be obtained by downloading the MariaDB source file tar.gz and extracting from it the CONNECT sources files in a directory that are added to the additional source directories if it is not the directory containing the above files.

The module must be linked to the `ha_connect.lib` of the binary version it will used with. Recent distributions add this lib in the plugin directory.

The resulting module, for instance `mongo.so` or `mongo.dll`, must be placed in the plugin directory of the MariaDB server. Then, you are able to use MONGO like tables simply replacing in the CREATE TABLE statement the option `TABLE_TYPE=MONGO` with `TABLE_TYPE=OEM SUBTYPE=MONGO MODULE='mongo.(so|dll)'`. Actually, the module name, here supposedly 'mongo', can be anything you like.

To implement for older versions that do not contain the MONGO type, you can add the corresponding source files, namely `javaconn.cpp`, `jmgfam.cpp`, `jmgoconn.cpp`, `mongo.cpp` and `tabjmg.cpp` that you should find in the CONNECT extracted source files if you downloaded a recent version. As they include `my_global.h`, this is the reason why the included file was named this way. In addition, your compiling should define `HAVE_JMGO` and `HAVE_JAVACONN`. Of course, this is possible only if `ha_connect.lib` is available.

This page is licensed: CC BY-SA / Gnu FDL

Current Status of the CONNECT Handler

The CONNECT handler is a GA (stable) release. It was written starting both from an aborted project written for MySQL in 2004 and from the `DBCONNECT` program. It was tested on all the examples described in this document, and is distributed with a set of 53 test cases. Here is a not limited list of future developments:

1. Adding more table types.
2. Make more tests files (53 are already made)
3. Adding more data types, in particular unsigned ones (done for unsigned).
4. Supporting indexing on nullable and decimal columns.
5. Adding more optimize tools (block indexing, dynamic indexing, etc.) (done)
6. Supporting MRR (done)
7. Supporting partitioning (done)
8. Getting NOSQL data from the Net as answers from REST queries (done)

No programs are bug free, especially new ones. Please report all bugs or documentation errors using the means provided by MariaDB.

This page is licensed: GPLv2

JSON Sample Files

Expense.json


```
[
  {
    "WHO": "Joe",
    "WEEK": [
      {
        "NUMBER": 3,
        "EXPENSE": [
          {
            "WHAT": "Beer",
            "AMOUNT": 18.00
          },
          {
            "WHAT": "Food",
            "AMOUNT": 12.00
          },
          {
            "WHAT": "Food",
            "AMOUNT": 19.00
          },
          {
            "WHAT": "Car",
            "AMOUNT": 20.00
          }
        ]
      },
      {
        "NUMBER": 4,
        "EXPENSE": [
          {
            "WHAT": "Beer",
            "AMOUNT": 19.00
          },
          {
            "WHAT": "Beer",
            "AMOUNT": 16.00
          },
          {
            "WHAT": "Food",
            "AMOUNT": 17.00
          },
          {
            "WHAT": "Food",
            "AMOUNT": 17.00
          },
          {
            "WHAT": "Beer",
            "AMOUNT": 14.00
          }
        ]
      },
      {
        "NUMBER": 5,
        "EXPENSE": [
          {
            "WHAT": "Beer",
            "AMOUNT": 14.00
          },
          {
            "WHAT": "Food",
            "AMOUNT": 12.00
          }
        ]
      }
    ]
```

```
  },
  {
    "WHO": "Beth",
    "WEEK": [
      {
        "NUMBER": 3,
        "EXPENSE": [
          {
            "WHAT": "Beer",
            "AMOUNT": 16.00
          }
        ]
      },
      {
        "NUMBER": 4,
        "EXPENSE": [
          {
            "WHAT": "Food",
            "AMOUNT": 17.00
          },
          {
            "WHAT": "Beer",
            "AMOUNT": 15.00
          }
        ]
      },
      {
        "NUMBER": 5,
        "EXPENSE": [
          {
            "WHAT": "Food",
            "AMOUNT": 12.00
          },
          {
            "WHAT": "Beer",
            "AMOUNT": 20.00
          }
        ]
      }
    ]
  },
  {
    "WHO": "Janet",
    "WEEK": [
      {
        "NUMBER": 3,
        "EXPENSE": [
          {
            "WHAT": "Car",
            "AMOUNT": 19.00
          },
          {
            "WHAT": "Food",
            "AMOUNT": 18.00
          },
          {
            "WHAT": "Beer",
            "AMOUNT": 18.00
          }
        ]
      },
      {
        "NUMBER": 4,
        "EXPENSE": [
          {
            "WHAT": "Car",
            "AMOUNT": 17.00
          }
        ]
```

```
      },
      {
        "NUMBER": 5,
        "EXPENSE": [
          {
            "WHAT": "Beer",
            "AMOUNT": 14.00
          },
          {
            "WHAT": "Car",
            "AMOUNT": 12.00
          },
          {
            "WHAT": "Beer",
            "AMOUNT": 19.00
          },
          {
            "WHAT": "Food",
            "AMOUNT": 12.00
          }
        ]
      }
    ]
  }
]
```

OEM Example

This is an example showing how an OEM table can be implemented. It is out of the scope of this document to explain how it works and to be a full guide on writing OEM tables for CONNECT.

tabfic.h

The header File `tabfic.h`:

```
// TABFIC.H     Olivier Bertrand    2008-2010
// External table type to read FIC files

#define TYPE_AM_FIC  (AMT)129

typedef class FICDEF *PFICDEF;
typedef class TDBFIC *PTDBFIC;
typedef class FICCOL *PFICCOL;

/* ------------------------ FIC classes ------------------------ */

/***********************************************************************/
/*  FIC: OEM table to read FIC files.                                  */
/***********************************************************************/

/***********************************************************************/
/*  This function is exported from the Tabfic.dll            */
/***********************************************************************/
extern "C" PTABDEF __stdcall GetFIC(PGLOBAL g, void *memp);

/***********************************************************************/
/*  FIC table definition class.                                        */
/***********************************************************************/
class FICDEF : public DOSDEF {        /* Logical table description */
  friend class TDBFIC;
 public:
  // Constructor
  FICDEF(void) {Pseudo = 3;}

  // Implementation
  virtual const char *GetType(void) {return "FIC";}

  // Methods
  virtual BOOL DefineAM(PGLOBAL g, LPCSTR am, int poff);
  virtual PTDB GetTable(PGLOBAL g, MODE m);

 protected:
  // No Members
}; // end of class FICDEF

/***********************************************************************/
/*  This is the class declaration for the FIC table.                   */
/***********************************************************************/
class TDBFIC : public TDBFIX {
  friend class FICCOL;
 public:
  // Constructor
  TDBFIC(PFICDEF tdp);

  // Implementation
  virtual AMT   GetAmType(void) {return TYPE_AM_FIC;}

  // Methods
  virtual void  ResetDB(void);
  virtual int   RowNumber(PGLOBAL g, BOOL b = FALSE);

  // Database routines
  virtual PCOL  MakeCol(PGLOBAL g, PCOLDEF cdp, PCOL cprec, int n);
  virtual BOOL  OpenDB(PGLOBAL g, PSQL sqlp);
  virtual int   ReadDB(PGLOBAL g);
  virtual int   WriteDB(PGLOBAL g);
  virtual int   DeleteDB(PGLOBAL g, int irc);

 protected:
  // Members
  int ReadMode;               // To read soft deleted lines
  int Rows;                           // Used for RowID
}; // end of class TDBFIC
```

```
/***********************************************************************/
/*  Class FICCOL: for Monetary columns.                                */
/***********************************************************************/
class FICCOL : public DOSCOL {
 public:
  // Constructors
  FICCOL(PGLOBAL g, PCOLDEF cdp, PTDB tdbp,
         PCOL cprec, int i, PSZ am = "FIC");

  // Implementation
  virtual int  GetAmType(void) {return TYPE_AM_FIC;}

  // Methods
  virtual void ReadColumn(PGLOBAL g);

 protected:
  // Members
  char Fmt;                   // The column format
}; // end of class FICCOL
```

tabfic.cpp

The source File `tabfic.cpp`:

```
/***********************************************************************/
/*  FIC: OEM table to read FIC files.                                  */
/***********************************************************************/
#if defined(WIN32)
#define WIN32_LEAN_AND_MEAN      // Exclude rarely-used stuff
#include <windows.h>
#endif   // WIN32
#include "global.h"
#include "plgdbsem.h"
#include "reldef.h"
#include "filamfix.h"
#include "tabfix.h"
#include "tabfic.h"

int TDB::Tnum;
int DTVAL::Shift;

/***********************************************************************/
/*  Initialize the CSORT static members.                               */
/***********************************************************************/
int    CSORT::Limit = 0;
double CSORT::Lg2 = log(2.0);
size_t CSORT::Cpn[1000] = {0};      /* Precalculated cmpnum values */

/* ------------- Implementation of the FIC subtype --------------- */

/***********************************************************************/
/*  This function is exported from the DLL.                            */
/***********************************************************************/
PTABDEF __stdcall GetFIC(PGLOBAL g, void *memp)
{
  return new(g, memp) FICDEF;
} // end of GetFIC

/* -------------- Implementation of the FIC classes -------------- */

/***********************************************************************/
/*  DefineAM: define specific AM block values from FIC file.           */
/***********************************************************************/
BOOL FICDEF::DefineAM(PGLOBAL g, LPCSTR am, int poff)
{
  ReadMode = GetIntCatInfo("Readmode", 0);

  // Indicate that we are a BIN format
  return DOSDEF::DefineAM(g, "BIN", poff);
} // end of DefineAM

/***********************************************************************/
/*  GetTable: makes a new TDB of the proper type.                      */
/***********************************************************************/
PTDB FICDEF::GetTable(PGLOBAL g, MODE m)
{
  return new(g) TDBFIC(this);
} // end of GetTable

/* ------------------------------------------------------------------ */

/***********************************************************************/
/*  Implementation of the TDBFIC class.                                */
/***********************************************************************/
TDBFIC::TDBFIC(PFICDEF tdp) : TDBFIX(tdp, NULL)
{
  ReadMode = tdp->ReadMode;
  Rows = 0;
} // end of TDBFIC constructor

/***********************************************************************/
/*  Allocate FIC column description block.                             */
```

```
/***********************************************************************/
PCOL TDBFIC::MakeCol(PGLOBAL g, PCOLDEF cdp, PCOL cprec, int n)
{
  PCOL colp;

  // BINCOL is alright except for the Monetary format
  if (cdp->GetFmt() && toupper(*cdp->GetFmt()) == 'M')
    colp = new(g) FICCOL(g, cdp, this, cprec, n);
  else
    colp = new(g) BINCOL(g, cdp, this, cprec, n);

  return colp;
} // end of MakeCol

/***********************************************************************/
/*  RowNumber: return the ordinal number of the current row.       */
/***********************************************************************/
int TDBFIC::RowNumber(PGLOBAL g, BOOL b)
{
  return (b) ? Txfp->GetRowID() : Rows;
} // end of RowNumber

/***********************************************************************/
/*  FIC Access Method reset table for re-opening.                  */
/***********************************************************************/
void TDBFIC::ResetDB(void)
{
  Rows = 0;
  TDBFIX::ResetDB();
} // end of ResetDB

/***********************************************************************/
/*  FIC Access Method opening routine.                             */
/***********************************************************************/
BOOL TDBFIC::OpenDB(PGLOBAL g, PSQL sqlp)
{
  if (Use == USE_OPEN) {
    // Table already open, just replace it at its beginning.
    return TDBFIX::OpenDB(g);
    } // endif use

  if (Mode != MODE_READ) {
    // Currently FIC tables cannot be modified.
    strcpy(g->Message, "FIC tables are read only");
    return TRUE;
    } // endif Mode

  /*********************************************************************/
  /*  Physically open the FIC file.                                  */
  /*********************************************************************/
  if (TDBFIX::OpenDB(g))
    return TRUE;

  Use = USE_OPEN;
  return FALSE;
} // end of OpenDB

/***********************************************************************/
/*  ReadDB: Data Base read routine for FIC access method.          */
/***********************************************************************/
int TDBFIC::ReadDB(PGLOBAL g)
{
  int rc;

  /*********************************************************************/
  /*  Now start the reading process.                                 */
  /*********************************************************************/
  do {
    rc = TDBFIX::ReadDB(g);
```

```
    } while (rc == RC_OK && ((ReadMode == 0 && *To_Line == '*') ||
                      (ReadMode == 2 && *To_Line != '*')));

  Rows++;
  return rc;
} // end of ReadDB

/*********************************************************************/
/*  WriteDB: Data Base write routine for FIC access methods.       */
/*********************************************************************/
int TDBFIC::WriteDB(PGLOBAL g)
{
  strcpy(g->Message, "FIC tables are read only");
  return RC_FX;
} // end of WriteDB

/*********************************************************************/
/*  Data Base delete line routine for FIC access methods.          */
/*********************************************************************/
int TDBFIC::DeleteDB(PGLOBAL g, int irc)
{
  strcpy(g->Message, "Delete not enabled for FIC tables");
  return RC_FX;
} // end of DeleteDB

// ---------------------- FICCOL functions -------------------------

/*********************************************************************/
/*  FICCOL public constructor.                                     */
/*********************************************************************/
FICCOL::FICCOL(PGLOBAL g, PCOLDEF cdp, PTDB tdbp, PCOL cprec, int i,
               PSZ am) : DOSCOL(g, cdp, tdbp, cprec, i, am)
{
  // Set additional FIC access method information for column.
  Fmt = toupper(*cdp->GetFmt());    // Column format
} // end of FICCOL constructor

/*********************************************************************/
/*  Handle the monetary value of this column. It is a big integer  */
/*  that represents the value multiplied by 1000.                  */
/*  In this function we translate it to a double float value.      */
/*********************************************************************/
void FICCOL::ReadColumn(PGLOBAL g)
{
  char   *p;
  int     rc;
  uint    n;
  double  fmon;
  PTDBFIC tdbp = (PTDBFIC)To_Tdb;

  /*********************************************************************/
  /*  If physical reading of the line was deferred, do it now.     */
  /*********************************************************************/
  if (!tdbp->IsRead())
    if ((rc = tdbp->ReadBuffer(g)) != RC_OK) {
      if (rc == RC_EF)
        sprintf(g->Message, MSG(INV_DEF_READ), rc);

      longjmp(g->jumper[g->jump_level], 11);
      } // endif

  p = tdbp->To_Line + Deplac;

  /*********************************************************************/
  /*  Set Value from the line field.                                 */
  /*********************************************************************/
  if (*(SHORT*)(p + 8) < 0) {
    n = ~*(SHORT*)(p + 8);
    fmon = (double)n;
```

```
    fmon *= 4294967296.0;
    n = ~*(int*)(p + 4);
    fmon += (double)n;
    fmon *= 4294967296.0;
    n = ~*(int*)p;
    fmon += (double)n;
    fmon++;
    fmon /= 1000000.0;
    fmon = -fmon;
  } else {
    fmon = ((double)*(USHORT*)(p + 8));
    fmon *= 4294967296.0;
    fmon += ((double)*(ULONG*)(p + 4));
    fmon *= 4294967296.0;
    fmon += ((double)*(ULONG*)p);
    fmon /= 1000000.0;
  } // enif neg

  Value->SetValue(fmon);
} // end of ReadColumn
```

tabfic.def

The file `tabfic.def` (required only on Windows):

```
LIBRARY     TABFIC
DESCRIPTION 'FIC files'
EXPORTS
   GetFIC       @1
```

JSON UDFs in a Separate Library

Although the JSON UDF's can be nicely included in the CONNECT library module, there are cases when you may need to have them in a separate library.

This is when CONNECT is compiled embedded, or if you want to test or use these UDF's with other MariaDB versions not including them.

To make it, you need to have access to the last MariaDB source code. Then, make a project containing these files:

1. jsonudf.cpp
2. json.cpp
3. value.cpp
4. osutil.c
5. plugutil.c
6. maputil.cpp
7. jsonutil.cpp

`jsonutil.cpp` is not distributed with the source code, you will have to make it from the following:

```
#include "my_global.h"
#include "mysqld.h"
#include "plugin.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>

#include "global.h"

extern "C" int GetTraceValue(void) { return 0; }
uint GetJsonGrpSize(void) { return 100; }

/***********************************************************************/
/*  These replace missing function of the (not used) DTVAL class.      */
/***********************************************************************/
typedef struct _datpar   *PDTP;
PDTP MakeDateFormat(PGLOBAL, PSZ, bool, bool, int) { return NULL; }
int ExtractDate(char*, PDTP, int, int val[6]) { return 0; }

#ifdef __WIN__
my_bool CloseFileHandle(HANDLE h)
{
    return !CloseHandle(h);
} /* end of CloseFileHandle */

#else  /* UNIX */
my_bool CloseFileHandle(HANDLE h)
{
    return (close(h)) ? TRUE : FALSE;
}  /* end of CloseFileHandle */

int GetLastError()
{
    return errno;
}  /* end of GetLastError */

#endif  // UNIX

/***********************************************************************/
/*  Program for sub-allocating one item in a storage area.             */
/*  Note: This function is equivalent to PlugSubAlloc except that in   */
/*  case of insufficient memory, it returns NULL instead of doing a    */
/*  long jump. The caller must test the return value for error.        */
/***********************************************************************/
void *PlgDBSubAlloc(PGLOBAL g, void *memp, size_t size)
{
  PPOOLHEADER pph;                       // Points on area header.

  if (!memp)    //  Allocation is to be done in the Sarea
    memp = g->Sarea;

  size = ((size + 7) / 8) * 8;  /* Round up size to multiple of 8 */
  pph = (PPOOLHEADER)memp;

  if ((uint)size > pph->FreeBlk) { /* Not enough memory left in pool */
    sprintf(g->Message,
     "Not enough memory in Work area for request of %d (used=%d free=%d)",
            (int)size, pph->To_Free, pph->FreeBlk);
    return NULL;
  } // endif size

  // Do the suballocation the simplest way
  memp = MakePtr(memp, pph->To_Free);   // Points to sub_allocated block
  pph->To_Free += size;                 // New offset of pool free block
  pph->FreeBlk -= size;                 // New size   of pool free block
```

```
  return (memp);
} // end of PlgDBSubAlloc
```

You can create the file by copy/paste from the above.

Set all the additional include directories to the MariaDB include directories used in plugin compiling plus the reference of the storage/connect directories, and compile like any other UDF giving any name to the made library module (I used `jsonudf.dll` on Windows)

Then you can create the functions using this name as the `soname` parameter.

There are some restrictions when using the UDF's this way:

- The `connect_json_grp_size` variable cannot be accessed. The group size is set to 100.
- In case of error, warnings are replaced by messages sent to stderr.
- No trace.

This page is licensed: CC BY-SA / Gnu FDL

CSV

See CSV Overview.

The CSV Storage Engine stores data in comma-separated values format text files, making it easy to exchange data with other applications.

See Checking and Repairing CSV Tables.

Learn how to use CHECK TABLE and REPAIR TABLE to identify and fix corruptions in CSV tables, discarding rows from the first error onwards.

This page is licensed: CC BY-SA / Gnu FDL

CSV Overview

The CSV Storage Engine can read and append to files stored in CSV (comma-separated-values) format.

The CSV storage engine and logging to tables

The CSV storage engine is the default storage engine when using logging of SQL queries to tables.

```
mysqld --log-output=table
```

CSV Storage Engine files

When you create a table using the CSV storage engine, three files are created:

- `<table_name>.frm`
- `<table_name>.CSV`
- `<table_name>.CSM`

The `.frm` file is the table format file.

The `.CSV` file is a plain text file. Data you enter into the table is stored as plain text in comma-separated-values format.

The `.CSM` file stores metadata about the table such as the state and the number of rows in the table.

Limitations

- CSV tables do not support indexing.

- CSV tables cannot be partitioned.
- Columns in CSV tables must be declared as NOT NULL.
- No transactions.
- The original CSV-format does not enable IETF-compatible parsing of embedded quote and comma characters. From MariaDB 10.1.8, it is possible to do so by setting the IETF_QUOTES option when creating a table.

Examples

Forgetting to add NOT NULL:

```
CREATE TABLE csv_test (x INT, y DATE, z CHAR(10)) ENGINE=CSV;
ERROR 1178 (42000): The storage engine for the table doesn't support nullable columns
```

Creating, inserting and selecting:

```
CREATE TABLE csv_test (
  x INT NOT NULL, y DATE NOT NULL, z CHAR(10) NOT NULL
  ) ENGINE=CSV;
```

```
INSERT INTO csv_test VALUES
    (1,CURDATE(),'one'),
    (2,CURDATE(),'two'),
    (3,CURDATE(),'three');
```

```
SELECT * FROM csv_test;
+---+------------+-------+
| x | y          | z     |
+---+------------+-------+
| 1 | 2011-11-16 | one   |
| 2 | 2011-11-16 | two   |
| 3 | 2011-11-16 | three |
+---+------------+-------+
```

Viewing in a text editor:

```
$ cat csv_test.CSV
1,"2011-11-16","one"
2,"2011-11-16","two"
3,"2011-11-16","three"
```

See Also

- Checking and Repairing CSV Tables

This page is licensed: CC BY-SA / Gnu FDL

Checking and Repairing CSV Tables

CSV tables support the CHECK TABLE and REPAIR TABLE statements.

`CHECK TABLE` marks the table as corrupt if it finds a problem, while `REPAIR TABLE` restores rows until the first corrupted row, discarding the rest.

Examples

```
CREATE TABLE csv_test (
  x INT NOT NULL, y DATE NOT NULL, z CHAR(10) NOT NULL
  ) ENGINE=CSV;

INSERT INTO csv_test VALUES
    (1,CURDATE(),'one'),
    (2,CURDATE(),'two'),
    (3,CURDATE(),'three');
```

```
SELECT * FROM csv_test;
+---+------------+-------+
| x | y          | z     |
+---+------------+-------+
| 1 | 2013-07-08 | one   |
| 2 | 2013-07-08 | two   |
| 3 | 2013-07-08 | three |
+---+------------+-------+
```

Using an editor, the actual file will look as follows

```
$ cat csv_test.CSV
1,"2013-07-08","one"
2,"2013-07-08","two"
3,"2013-07-08","three"
```

Let's introduce some corruption with an unwanted quote in the 2nd row:

```
1,"2013-07-08","one"
2","2013-07-08","two"
3,"2013-07-08","three"
```

```
CHECK TABLE csv_test;
+---------------+-------+----------+----------+
| Table         | Op    | Msg_type | Msg_text |
+---------------+-------+----------+----------+
| test.csv_test | check | error    | Corrupt  |
+---------------+-------+----------+----------+
```

We can repair this, but all rows from the corrupt row onwards are lost:

```
REPAIR TABLE csv_test;
+---------------+--------+----------+----------------------------------------+
| Table         | Op     | Msg_type | Msg_text                               |
+---------------+--------+----------+----------------------------------------+
| test.csv_test | repair | Warning  | Data truncated for column 'x' at row 2 |
| test.csv_test | repair | status   | OK                                     |
+---------------+--------+----------+----------------------------------------+

SELECT * FROM csv_test;
+---+------------+-----+
| x | y          | z   |
+---+------------+-----+
| 1 | 2013-07-08 | one |
+---+------------+-----+
```

This page is licensed: CC BY-SA / Gnu FDL

DuckDB

DuckDB Storage Engine

> **Important:**
>
> **Experimental — for evaluation only.** The DuckDB storage engine is at **alpha** maturity. It is not yet merged into MariaDB Server `main`, not built into MariaDB releases, and not signed in any official package. Interfaces, configuration variables, behavior, and supported operations are subject to change. **Do not use in production.**

The DuckDB storage engine embeds DuckDB — a columnar, vectorized, in-process analytical database engine — inside MariaDB Server as a loadable plugin (`ha_duckdb.so`). Tables created with `ENGINE=DuckDB` store data in DuckDB's native columnar format, and queries against them are executed by the DuckDB engine. Cross-engine joins between DuckDB and other engines (e.g. InnoDB) are supported in a single `SELECT`.

The engine borrows its design from Alibaba's AliSQL DuckDB integration. The original code was heavily refactored to use MariaDB's handler API and plugin system, and the engine links against vanilla upstream DuckDB.

Use Cases

- **HTAP (Hybrid Transactional/Analytical Processing)** — InnoDB serves the transactional workload, DuckDB serves analytics, both in the same MariaDB server.
- **Ad-hoc analytical queries** — joins, aggregations, and window functions over large datasets without exporting data to a separate system.
- **Eliminating ETL** — the analytical engine runs in-process; no separate cluster or data-movement pipeline.

Architecture

DuckDB 1.5.2 is statically linked into a single plugin shared library (`ha_duckdb.so`) along with the built-in ICU and JSON extensions. The engine integrates with MariaDB via the standard `handler` API and adds a `select_handler` for full query pushdown.

Cross-engine queries (a single `SELECT` joining a `DuckDB` table with, for example, an `InnoDB` table) are handled as follows: the entire query is pushed down to DuckDB. When DuckDB references a non-DuckDB table, a callback re-enters the MariaDB query pipeline through a cooperative fiber to stream rows back. The MariaDB optimizer still chooses the access path on the non-DuckDB side, and DuckDB's optimizer drives join order, hashing, and aggregation.

Supported Platforms

- **OS:** Linux only.
- **Architectures:** `x86_64` and `aarch64` (ARM64).
- **Distributions** (auto-detected by the build script): Ubuntu 22.04, Ubuntu 24.04, Debian 12, Rocky Linux 8, Rocky Linux 9. Fedora 42 builds are produced by the MariaDB CI.
- **Compiler:** GCC 12 or later (C++17 required).
- **MariaDB Server:** the engine is being prepared for MariaDB 11.4. A merge into the mainline source tree and the shipping version are not yet final.

Installing

The engine is **not** included in MariaDB releases. There are two ways to obtain it: prebuilt test packages from MariaDB CI, or building from source.

Option 1: Prebuilt Test Packages

MariaDB CI builds the engine for several Linux distributions and publishes the artifacts on ci.mariadb.org. Each CI run has its own job number and a subdirectory per distribution and architecture, for example `amd64-ubuntu-2404-deb-autobake/` or `aarch64-ubuntu-2404-deb-autobake/`.

> **Warning:**
>
> These packages are **unsigned** and intended for testing only.

The CI job number is **not stable** across rebuilds. For the current job URL, see the blog posts under See also.

Example for Ubuntu 24.04 (`amd64`), with a current job number `<JOB>`:

```
# Fetch the .deb files for the job
wget -r -np -nH --cut-dirs=2 -A '*.deb' \
  https://ci.mariadb.org/<JOB>/amd64-ubuntu-2404-deb-autobake/

# Install the server and the DuckDB plugin package
sudo dpkg -i mariadb-server*.deb mariadb-plugin-duckdb*.deb || sudo apt -f install
```

Option 2: Build From Source

The engine integration lives on the `11.4` branch of `MariaDB/server`. The engine source resides in the server repository under `storage/duckdb/`; a helper script there handles dependency installation and the build.

```
git clone --recurse-submodules -b 11.4 \
  https://github.com/MariaDB/server.git mariadb-server
cd mariadb-server

# Install build prerequisites (requires root)
./storage/duckdb/build.sh -D

# Build, install and start MariaDB server
./storage/duckdb/build.sh -S

# Optional: build a DEB or RPM instead
./storage/duckdb/build.sh -p
```

On Rocky Linux 8, pass `-R` to use `gcc-toolset-12`:

```
./storage/duckdb/build.sh -R
```

Enabling the Plugin

The DuckDB plugin is not loaded by default. Add a configuration file to the server's drop-in directory:

- **Debian / Ubuntu:** `/etc/mysql/mariadb.conf.d/duckdb.cnf`
- **Rocky / Fedora / openSUSE:** `/etc/my.cnf.d/duckdb.cnf`

```
[mysqld]
plugin-maturity=alpha
plugin-load-add=ha_duckdb.so
duckdb-memory-limit=8G
```

Each line is significant:

- `plugin-maturity=alpha` is **required**. The server's default maturity threshold rejects alpha plugins; this line is the explicit opt-in.
- `plugin-load-add=ha_duckdb.so` loads the engine at startup.

- `duckdb-memory-limit` sets the cap on memory DuckDB itself may use. The default (1 GiB) is too low for most analytical workloads — raise it to reflect the host's available RAM.

Restart the server, then verify the engine is registered:

```
SHOW ENGINES\G
```

The output should include a `DuckDB` row with `Support: YES`.

Verifying the Install

```
CREATE DATABASE analytics;
USE analytics;

CREATE TABLE orders (
  id BIGINT PRIMARY KEY,
  amount DECIMAL(15, 2),
  created_at TIMESTAMP
) ENGINE=DuckDB DEFAULT CHARSET=utf8mb4;

INSERT INTO orders VALUES (1, 199.99, NOW()), (2, 299.50, NOW());

SELECT SUM(amount) FROM orders;
```

DuckDB requires UTF-8: specify `DEFAULT CHARSET=utf8mb4` (or another UTF-8 charset) on every DuckDB table. Non-UTF8 charsets fall back to binary comparison.

Supported Operations

- **DDL:** `CREATE TABLE`, `DROP TABLE`, `ALTER TABLE`, `RENAME TABLE`. `ALTER TABLE ... ENGINE=DuckDB` migrates an existing table into DuckDB storage.
- **`INSERT`:** all major MariaDB column types.
- **`SELECT`:** full pushdown to DuckDB, including aggregations, joins, window functions, `GROUP BY`, and `ORDER BY`.
- **`UPDATE` and `DELETE`:** translated to DuckDB SQL.
- **Cross-engine `JOIN`:** a single `SELECT` can join DuckDB tables with InnoDB (or other) tables. Example:

  ```
  SELECT d.id, d.amount, i.name
    FROM analytics.orders d        -- ENGINE=DuckDB
    JOIN inventory.products i      -- ENGINE=InnoDB
      ON d.product_id = i.id
   WHERE d.amount > 1000;
  ```

- **User-defined functions:** DuckDB-side UDFs are registered with MariaDB.

Configuration Variables

The plugin adds a family of `duckdb_*` server variables. The most commonly tuned ones:

Variable	Type	Default	Purpose
`duckdb_memory_limit`	integer (bytes)	`1073741824` (1 GiB)	Maximum memory DuckDB may use. Raise for analytical workloads.
`duckdb_max_threads`	integer	(auto)	Maximum DuckDB execution threads.
`duckdb_temp_directory`	string	(auto)	Directory used by DuckDB for spilling.
`duckdb_require_primary_key`	boolean	`ON`	Reject `CREATE TABLE` without a `PRIMARY KEY`.

Variable	Type	Default	Purpose
`duckdb_max_temp_directory_size`	integer (bytes)	(unlimited)	Cap on temp-directory usage.

Additional tuning and debugging variables (batching, log categories, optimizer hints, `EXPLAIN` verbosity, etc.) are exposed by the plugin. List the full current set with:

```
SHOW VARIABLES LIKE 'duckdb%';
```

Limitations

The engine is at alpha maturity. Known limitations at the time of writing:

- **No `AUTO_INCREMENT`** and no `UUID` default-value generation.
- **`DECIMAL` precision capped at 38** — DuckDB's maximum. Wider MariaDB `DECIMAL` columns fail at DDL conversion.
- **Strict `GROUP BY`** — DuckDB rejects `SELECT` columns that are neither in the `GROUP BY` clause nor aggregated, even when MariaDB's `sql_mode` would allow it.
- **No XA `PREPARE`.**
- **UTF-8 only** — non-UTF8 charsets fall back to binary comparison. UCA-based MariaDB collations are approximated using DuckDB's `NOCASE` / `NOACCENT` collations.
- **Some MariaDB functions are not pushdown-compatible** — for example, `GROUP_CONCAT()`, `DATE_FORMAT()`, `JSON_CONTAINS()`, `FOUND_ROWS()`, and `LAST_INSERT_ID()` have no DuckDB equivalent or differ in syntax. Such queries fall back to MariaDB-side evaluation.
- **`ALTER COLUMN DROP DEFAULT`** isn't propagated to the DuckDB catalog.
- **`TIMESTAMP` is stored as `TIMESTAMPTZ`** — keep MariaDB and DuckDB timezone settings consistent to avoid value shifts.
- **Cross-engine scan is single-threaded** — only the DuckDB side of a cross-engine query is parallelized.
- **Cross-engine `INSERT` (DuckDB → InnoDB)** has restrictions; see the engine compatibility matrix.

A maintained compatibility matrix is published in the server repository at `storage/duckdb/docs/mariadb-duckdb-incompatibilities.md`.

See Also

- MDEV-39234 — upstream feature ticket.
- `11.4` branch — the MariaDB Server branch where the engine source resides, under `storage/duckdb/`.
- DuckDB Storage Engine for MariaDB: When the Sea Lion Learns to Quack — Roman Nozdrin, 2026-06-09.
- MariaDB DuckDB: A New Playground for Analytics — Frédéric Descamps, 2026-06-12.
- SHOW ENGINES
- INSTALL SONAME

This page is licensed: CC BY-SA / Gnu FDL

FederatedX

> **Warning:**
>
> This storage engine has been deprecated.

See About FederatedX.

An introduction to the FederatedX storage engine, a fork of MySQL's Federated engine, allowing access to remote tables as if they were local. This storage engine has been deprecated.

See Differences Between FederatedX and Federated.

This page outlines the key enhancements in FederatedX over the original Federated engine, including support for transactions and a refactored codebase. This storage engine has been deprecated.

This page is licensed: CC BY-SA / Gnu FDL

About FederatedX

> **Warning:**
>
> This storage engine has been deprecated.

The FederatedX storage engine is a fork of MySQL's Federated storage engine, which is no longer being developed by Oracle. The original purpose of FederatedX was to keep this storage engine's development progressing– to both add new features as well as fix old bugs.

Since MariaDB 10.0, the CONNECT storage engine also allows access to a remote database via MySQL or ODBC connection (table types: MYSQL, ODBC). However, in the current implementation there are several limitations.

What is the FederatedX storage engine?

The FederatedX Storage Engine is a storage engine that works with both MariaDB and MySQL. Where other storage engines are built as interfaces to lower-level file-based data stores, FederatedX uses libmysql to talk to the data source, the data source being a remote RDBMS. Currently, since FederatedX only uses libmysql, it can only talk to another MariaDB or MySQL RDBMS. The plan is of course to be able to use other RDBMS systems as a data source. There is an existing project Federated ODBC which was able to use PostgreSQL as a remote data source, and it is this type of functionality which are brought to FederatedX in subsequent versions.

History

The history of FederatedX is derived from the History of Federated. Cisco needed a MySQL storage engine that would allow them to consolidate remote tables on some sort of routing device, being able to interact with these remote tables as if they were local to the device, but not actually on the device, since the routing device had only so much storage space. The first prototype of the Federated Storage Engine was developed by JD (need to check on this- Brian Aker can verify) using the HANDLER interface. Brian handed the code to Patrick Galbraith and explained how it needed to work, and with Brian and Monty's tutelage and Patrick had a working Federated Storage Engine with MySQL 5.0. Eventually, Federated was released to the public in a MySQL 5.0 release.

When MySQL 5.1 became the production release of MySQL, Federated had more features and enhancements added to it, namely:

- New Federated SERVER added to the parser. This was something Cisco needed that made it possible to change the connection parameters for numerous Federated tables at once without having to alter or re-create the Federated tables.
- Basic Transactional support– for supporting remote transactional tables
- Various bugs that needed to be fixed from MySQL 5.0
- Plugin capability

In MariaDB 10.0.2 FederatedX got support for assisted table discovery.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'ha_federatedx';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = ha_federatedx
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'ha_federatedx';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

How FederatedX works

Every storage engine has to implement derived standard handler class API methods for a storage engine to work. FederatedX is no different in that regard. The big difference is that FederatedX needs to implement these handler methods in such as to construct SQL statements to run on the remote server and if there is a result set, process that result set into the internal handler format so that the result is returned to the user.

Internal workings of FederatedX

Normal database files are local and as such: You create a table called
'users', a file such as 'users.MYD' is created. A handler reads, inserts, deletes, updates data in this file. The data is stored in particular format, so to read, that data has to be parsed into fields, to write, fields have to be stored in this format to write to this data file.

With the FederatedX storage engine, there are no local files for each table's data (such as .MYD). A foreign database will store the data that would normally be in this file. This will necessitate the use of MySQL client API to read, delete, update, insert this data. The data will have to be retrieve via an SQL call
"`SELECT * FROM users`". Then, to read this data, it will have to be retrieved via `mysql_fetch_row` one row at a time, then converted from the column in this select into the format that the handler expects.

The basic functionality of how FederatedX works is:

- The user issues an SQL statement against the local federatedX table. This statement is parsed into an item tree
- FederatedX uses the mysql handler API to implement the various methods required for a storage engine. It has access to the item tree for the SQL statement issued, as well as the Table object and each of its Field members. At
- With this information, FederatedX constructs an SQL statement
- The constructed SQL statement is sent to the Foreign data source through libmysql using the mysql client API

- The foreign database reads the SQL statement and sends the result back through the mysql client API to the origin
- If the original SQL statement has a result set from the foreign data source, the FederatedX storage engine iterates through the result set and converts each row and column to the internal handler format
- If the original SQL statement only returns the number of rows returned (affected_rows), that number is added to the table stats which results in the user seeing how many rows were affected.

FederatedX table creation

The create table will simply create the .frm file, and within the `CREATE TABLE` SQL statement, there SHALL be any of the following :

```
connection=scheme://username:password@hostname:port/database/tablename
connection=scheme://username@hostname/database/tablename
connection=scheme://username:password@hostname/database/tablename
connection=scheme://username:password@hostname/database/tablename
```

Or using the syntax introduced in MySQL versions 5.1 for a Federated server (SQL/MED Spec xxxx)

```
connection="connection_one"
connection="connection_one/table_foo"
```

An example of a connect string specifying all the connection parameters would be:

```
connection=mysql://username:password@hostname:port/database/tablename
```

Or, using a Federated server, first a server is created:

```
CREATE SERVER 'server_one' FOREIGN DATA WRAPPER 'mysql' OPTIONS
  (HOST '127.0.0.1',
  DATABASE 'db1',
  USER 'root',
  PASSWORD '',
  PORT 3306,
  SOCKET '',
  OWNER 'root');
```

MariaDB starting with 10.11.12

You can also use ‘mariadb’ as a wrapper.

Then the FederatedX table is created specifying the newly created Federated server.
The following statements creates a federated table, `federated.t1` against the table `db1.t1` on the remote server.

```
CREATE TABLE federated.t1 (
  `id` INT(20) NOT NULL,
  `name` VARCHAR(64) NOT NULL DEFAULT ''
  )
ENGINE="FEDERATED" DEFAULT CHARSET=latin1
CONNECTION='server_one';
```

(Note that in MariaDB, the original Federated storage engine is replaced with the new FederatedX storage engine. And for backward compatibility, the old name “FEDERATED” is used in create table. So in MariaDB, the engine type should be given as “FEDERATED” without an extra “X”, not “FEDERATEDX”).

The equivalent of above, if done specifying all the connection parameters

```
CONNECTION="mysql://root@127.0.0.1:3306/db1/t1"
```

You can also change the server to point to a new schema:

```
ALTER SERVER 'server_one' OPTIONS(DATABASE 'db2');
```

All subsequent calls to any FederatedX table using the ‘server_one’ will now be against tables in `db2`! Guess what? You no longer have to perform an alter table in order to point one or more FederatedX tables to a new server!

This `connection="connection string"` is necessary for the handler to be able to connect to the foreign server, either by URL, or by server name.

Method calls

One way to see how the FederatedX storage engine works is to compile a debug build of MariaDB and turn on a trace log. Using a two column table, with one record, the following SQL statements shown below, can be analyzed for what internal methods they result in being called.

SELECT

If the query is for instance “`SELECT * FROM foo`”, then the primary methods you would see with debug turned on would be first:

```
ha_federatedx::info
ha_federatedx::scan_time:
ha_federatedx::rnd_init: share->select_query SELECT * FROM foo
ha_federatedx::extra
```

Then for every row of data retrieved from the foreign database in the result set:

```
ha_federatedx::rnd_next
ha_federatedx::convert_row_to_internal_format
ha_federatedx::rnd_next
```

After all the rows of data that were retrieved, you would see:

```
ha_federatedx::rnd_end
ha_federatedx::extra
ha_federatedx::reset
```

INSERT

If the query was “`INSERT INTO foo (id, ts) VALUES (2, now());`”, the trace would be:

```
ha_federatedx::write_row
ha_federatedx::reset
```

UPDATE

If the query was “`UPDATE foo SET ts = now() WHERE id = 1;`”, the resultant trace would be:

```
ha_federatedx::index_init
ha_federatedx::index_read
ha_federatedx::index_read_idx
ha_federatedx::rnd_next
ha_federatedx::convert_row_to_internal_format
ha_federatedx::update_row

ha_federatedx::extra
ha_federatedx::extra
ha_federatedx::extra
ha_federatedx::external_lock
ha_federatedx::reset
```

FederatedX capabilities and limitations

- Tables MUST be created on the foreign server prior to any action on those tables via the handler, first version. IMPORTANT: IF you MUST use the FederatedX storage engine type on the REMOTE end, make sure that the table you connect to IS NOT a table pointing BACK to your ORIGINAL table! You know and have heard the screeching of audio feedback? You know putting two mirrors in front of each other how the reflection continues for eternity? Well, need I say more?!
- There is no way for the handler to know if the foreign database or table has changed. The reason for this is that this database has to work like a data file that would never be written to by anything other than the database. The integrity of the data in the local table could be breached if there was any change to the foreign database.
- Support for SELECT, INSERT, UPDATE, DELETE indexes.
- No ALTER TABLE, DROP TABLE or any other Data Definition Language calls.
- Prepared statements will not be used in the first implementation, it remains to be seen whether the limited subset of the client API for the server supports this.
- This uses SELECT, INSERT, UPDATE, DELETE and not HANDLER for its implementation.
- This will not work with the query cache.
- FederatedX does not support GEOMETRY types. Such tables cannot be created explicitly, nor discovered.

How do you use FederatedX?

To use this handler, it's very simple. You must have two databases running, either both on the same host, or on different hosts.

First, on the foreign database you create a table, for example:

```
CREATE TABLE federated.test_table (
  id     INT(20) NOT NULL AUTO_INCREMENT,
  name   VARCHAR(32) NOT NULL DEFAULT '',
  other  INT(20) NOT NULL DEFAULT '0',
  PRIMARY KEY  (id),
  KEY name (name),
  KEY other_key (other))
DEFAULT CHARSET=latin1;
```

Then, on the server that are connecting to the foreign host (client), you create a federated table without specifying the table structure:

```
CREATE TABLE federated_test_table ENGINE=FEDERATED
  CONNECTION='mysql://root@127.0.0.1:9306/federated/test_table';
```

Notice the "ENGINE" and "CONNECTION" fields? This is where you respectively set the engine type, "FEDERATED" and foreign host information, this being the database your 'client' database will connect to and use as the "data file". Obviously, the foreign database is running on port 9306, so you want to start up your other

database so that it is indeed on port 9306, and your FederatedX database on a port other than that. In my setup, I use port 5554 for FederatedX, and port 5555 for the foreign database.

Alternatively (or if you're using MariaDB before version 10.0.2) you specify the federated table structure explicitly:

```
CREATE TABLE federated_test_table (
  id     INT(20) NOT NULL AUTO_INCREMENT,
  name   VARCHAR(32) NOT NULL DEFAULT '',
  other  INT(20) NOT NULL DEFAULT '0',
  PRIMARY KEY  (id),
  KEY name (name),
  KEY other_key (other))
ENGINE=FEDERATED
DEFAULT CHARSET=latin1
CONNECTION='mysql://root@127.0.0.1:9306/federated/test_table';
```

In this case the table structure must match exactly the table on the foreign server.

How to see the storage engine in action

When developing this handler, I compiled the FederatedX database with debugging:

```
./configure --with-federatedx-storage-engine \
  --prefix=/home/mysql/mysql-build/federatedx/ --with-debug
```

Once compiled, I did a 'make install' (not for the purpose of installing the binary, but to install all the files the binary expects to see in the directory I specified in the build with

```
--prefix=/home/code-dev/maria
```

Then, I started the foreign server:

```
/usr/local/mysql/bin/mysqld_safe \
  --user=mysql --log=/tmp/mysqld.5555.log -P 5555
```

Then, I went back to the directory containing the newly compiled mysqld `<builddir>/sql/`, started up gdb:

```
gdb ./mysqld
```

Then, within the (gdb) prompt:

```
(gdb) run --gdb --port=5554 --socket=/tmp/mysqld.5554 --skip-innodb --debug
```

Next, I open several windows for each:

1. Tail the debug trace: tail -f /tmp/mysqld.trace|grep ha_fed
2. Tail the SQL calls to the foreign database: tail -f /tmp/mysqld.5555.log
3. A window with a client open to the federatedx server on port 5554
4. A window with a client open to the federatedx server on port 5555

I would create a table on the client to the foreign server on port 5555, and then to the FederatedX server on port 5554. At this point, I would run whatever queries I wanted to on the FederatedX server, just always remembering that whatever changes I wanted to make on the table, or if I created new tables, that I would have to do that on the foreign server.

Another thing to look for is 'show variables' to show you that you have support for FederatedX handler support:

```
SHOW VARIABLES LIKE '%federat%';
```

and:

```
SHOW storage engines;
```

Both should display the federatedx storage handler.

How do I create a federated server?

A federated server is a way to have a foreign data source defined– with all connection parameters– so that you don't have to specify explicitly the connection parameters in a string.

For instance, if you wanted to connect to a table, `first_db.test_table`, using this definition:

```
CREATE TABLE federated_test_table ENGINE=FEDERATED
  CONNECTION='mysql://patg@192.168.1.123/first_db/test_table';
```

You could instead create this with a server:

```
CREATE SERVER 'server_one' FOREIGN DATA WRAPPER 'mysql' OPTIONS
  (HOST '192.168.1.123',
  DATABASE 'first_db',
  USER 'patg',
  PASSWORD '',
  PORT 3306,
  SOCKET '',
  OWNER 'root');
```

You could now specify the server instead of the full URL in the connection string:

```
CREATE TABLE federated_test_table ENGINE=FEDERATED
  CONNECTION='server_one/test_table';
```

On the server where you create this `federated_test_table` you will now have access to the table `test_table` in the `first_db` database on the remote server found on 192.168.1.123.

How does FederatedX differ from the old Federated Engine?

FederatedX from a user point of view is the same for the most part. What is different with FederatedX and Federated is the following:

- Rewrite of the main Federated source code from one single ha_federated.cc file into three main abstracted components:
  - ha_federatedx.cc - Core implementation of FederatedX
  - federated_io.cc - Parent connection class to be over-ridden by derived classes for each RDBMS/client lib
  - federatated_io_.cc - derived federated_io class for a given RDBMS
  - federated_txn.cc - New support for using transactional engines on the foreign server using a connection poll
- Various bugs fixed (need to look at opened bugs for Federated)

Where can I get FederatedX

FederatedX is part of MariaDB 5.1 and later. MariaDB merged with the latest FederatedX when there is a need to get a bug fixed. You can get the latest code/follow/participate in the project from the FederatedX home page.

What are the plans for FederatedX?

- Support for other RDBMS vendors using ODBC
- Support for pushdown conditions
- Ability to limit result set sizes

See Also

- CREATE SERVER
- ALTER SERVER
- DROP SERVER

This page is licensed: CC BY-SA / Gnu FDL

Differences Between FederatedX and Federated

> **Warning:**
>
> This storage engine has been deprecated.

The main differences are:

New features in FederatedX

- Transactions (beta feature)
- Supports partitions (alpha feature)
- New class structure which allows developers to write connection classes for other RDBMSs without having to modify base classes for FederatedX

Different behavior

- FederatedX is statically compiled into MariaDB by default.
- When you create a table with FederatedX, the connection are tested. The `CREATE` will fail if MariaDB can't connect to the remote host or if the remote table doesn't exist.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB

See InnoDB Introduction.

An overview of the InnoDB storage engine, detailing its support for ACID transactions, row-level locking, and crash recovery.

See InnoDB File Format.

Learn about the different file formats supported by InnoDB, such as Antelope and Barracuda, and how they impact table features and storage.

See InnoDB Limitations.

A list of constraints and limits within the InnoDB engine, including maximum table size, column counts, and index key lengths.

See AUTO_INCREMENT Handling in InnoDB.

This page explains how InnoDB manages AUTO_INCREMENT columns, including initialization behavior, gap handling, and potential restart effects.

See Binary Log Group Commit and InnoDB Flushing Performance.

Understand how group commit works with InnoDB to improve performance by reducing the number of disk syncs required during transaction commits.

See InnoDB Asynchronous I/O.

Explore how InnoDB uses asynchronous I/O on various operating systems to handle multiple read and write requests concurrently without blocking.

See InnoDB Buffer Pool.

Complete InnoDB Buffer Pool guide for MariaDB. Complete reference documentation for implementation, configuration, and usage with comprehensive examples and.

See InnoDB Change Buffering.

Learn about the change buffer, an optimization that delays secondary index writes to reduce I/O overhead for non-unique index modifications.

See InnoDB Data Scrubbing.

This feature allows for the secure deletion of data by overwriting deleted records in tablespaces and logs to prevent data recovery.

See InnoDB Doublewrite Buffer.

The doublewrite buffer is a storage area where InnoDB writes pages before writing them to the data file, preventing data corruption from partial page writes.

See InnoDB Flush Method.

Detailed description of the innodb_flush_method variable, its various settings, and effects.

See InnoDB Lock Modes.

InnoDB employs Row-Level Locking with Shared (S) and Exclusive (X) locks, along with Intention locks, to manage concurrent transaction access.

See InnoDB Log Archiving.

Retain the full InnoDB write-ahead log history (from MariaDB 13.0) by archiving log files, enabling point-in-time recovery and incremental backups.

See InnoDB Monitors.

InnoDB Monitors, such as the Standard, Lock, and Tablespace monitors, provide detailed internal state information to the error log for diagnostics.

See InnoDB Page Compression.

This feature enables transparent page-level compression for tables using algorithms like LZ4 or Zlib, reducing storage requirements.

See InnoDB Page Flushing.

Learn about the background processes that flush dirty pages from the buffer pool to disk, including adaptive flushing algorithms to optimize I/O.

See InnoDB Purge.

The purge process is a garbage collection mechanism that removes old row versions from the undo log that are no longer required for MVCC.

See InnoDB Undo Log.

The undo log stores the “before” image of data modified by active transactions, supporting rollbacks and consistent read views.

See InnoDB Redo Log.

The redo log is a disk-based transaction log used during crash recovery to replay incomplete transactions and ensure data durability.

See InnoDB Strict Mode.

InnoDB Strict Mode enforces stricter SQL compliance, returning errors instead of warnings for invalid CREATE TABLE options or potential data loss.

See InnoDB System Variables.

Complete guide to InnoDB system variables for MariaDB. Complete reference for buffer pool, I/O tuning, transaction settings, and optimization for production use.

See InnoDB Versions.

This page is outdated. It's left in place because release notes for old MariaDB versions refer to it (MariaDB < 10.3).

See InnoDB Architecture for MariaDB Enterprise Server.

Understand InnoDB's architecture for MariaDB Enterprise Server. This section details its components and their interactions, focusing on performance, scalability, and reliability for enterprise workloa

See InnoDB Online DDL.

Perform online DDL operations with InnoDB in MariaDB Server. Learn how to alter tables without blocking read/write access, ensuring high availability for your applications.

See InnoDB Row Formats.

Explore InnoDB row formats in MariaDB Server. Understand different formats like Compact, Dynamic, and Compressed, and how they impact storage efficiency and performance for your data.

See InnoDB Tablespaces.

Manage InnoDB tablespaces in MariaDB Server. Understand their role in data organization, performance, and recovery, including file-per-table and shared tablespaces.

See InnoDB Troubleshooting.

Troubleshoot InnoDB issues in MariaDB Server. Find solutions and best practices for common problems, ensuring your InnoDB-based applications run smoothly and efficiently.

See MariaDB Enterprise Server InnoDB Operations.

Learn about InnoDB operations in MariaDB Enterprise Server. This section covers critical management tasks, including configuration, performance tuning, and troubleshooting for enterprise-grade deploym

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Introduction

Overview

MariaDB Enterprise Server uses the InnoDB storage engine by default. InnoDB is a general purpose transactional storage engine that is performant, ACID-compliant, and well-suited for most workloads.

Benefits

The InnoDB storage engine:

- Is available with all versions of MariaDB Enterprise Server and MariaDB Community Server.
- Is a general purpose storage engine.
- Is transactional and well-suited for online transactional processing (OLTP) workloads.

- Is ACID-compliant.
- Performs well for mixed read-write workloads.
- Supports online DDL.

Feature Summary

Feature	Detail	Resources
Storage Engine	InnoDB	
Availability	All ES and CS versions	MariaDB Enterprise Server
Workload Optimization	Transactional	
Table Orientation	Row	
Default Row Format	Dynamic	InnoDB Row Formats InnoDB Dynamic Row Format
ACID-compliant	Yes	
XA Transactions	Yes	
Primary Keys	Yes	InnoDB Primary Keys
Auto-Increment	Yes	InnoDB AUTO_INCREMENT Columns
Sequences	Yes	InnoDB Sequences
Foreign Keys	Yes	InnoDB Foreign Keys
Indexes	Yes	InnoDB Indexes
Secondary Indexes	Yes	InnoDB Secondary Indexes
Unique Indexes	Yes	InnoDB Unique Indexes
Full-text Search	Yes	InnoDB Full-text Indexes
Spatial Indexes	Yes	InnoDB Spatial Indexes
Compression	Yes	Configure InnoDB Page Compression
Data-at-Rest Encryption	Yes	
High Availability (HA)	Yes	• MariaDB Replication • Galera Cluster
Main Memory Caching	Yes	InnoDB Buffer Pool
Transaction Logging	Yes	• InnoDB Redo Log (Crash Safety) • InnoDB Undo Log (MVCC)
Garbage Collection	Yes	InnoDB Purge Threads
Online Schema changes	Yes	InnoDB Schema Changes
Non-locking Reads	Yes	
Row Locking	Yes	

Examples

Creating an InnoDB Table

```
CREATE DATABASE hq_sales;
```

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
) ENGINE = InnoDB;
```

```
SELECT TABLE_SCHEMA, TABLE_NAME, ENGINE
FROM information_schema.TABLES
WHERE TABLE_SCHEMA='hq_sales'
AND TABLE_NAME='invoices';
+--------------+------------+--------+
| TABLE_SCHEMA | TABLE_NAME | ENGINE |
+--------------+------------+--------+
| hq_sales     | invoices   | InnoDB |
+--------------+------------+--------+
```

Resources

Architecture

- Background Thread Pool
- Buffer Pool
- I/O Threads
- Purge Threads
- Redo Log
- Row Formats
- Undo Log

Operations

- Configure Page Compression
- Configure the Buffer Pool
- Configure the I/O Threads
- Configure the Purge Threads
- Configure the Redo Log
- Configure the Undo Log
- Schema Changes

MariaDB documentation

- InnoDB

This page is: Copyright © 2026 MariaDB. All rights reserved.

InnoDB File Format

Setting a Table's File Format

A table's tablespace is tagged with the lowest InnoDB file format that supports the table's row format. So, even if the `Barracuda` file format is enabled, tables that use the `COMPACT` or `REDUNDANT` row formats are tagged with the `Antelope` file format in the information_schema.INNODB_SYS_TABLES table.

Supported File Formats

The InnoDB storage engine supports two different file formats:

- `Antelope`
- `Barracuda`

Antelope

In MariaDB 10.2.1 and before, the default file format is `Antelope`. In MariaDB 10.2.2 and later, the `Antelope` file format is deprecated.

`Antelope` is the original InnoDB file format. It supports the `COMPACT` and `REDUNDANT` row formats, but not the `DYNAMIC` or `COMPRESSED` row formats.

Barracuda

In MariaDB 10.1 and before, the `Barracuda` file format is only supported if the innodb_file_per_table system variable is set to `ON`. In MariaDB 10.2.2 and later, the default file format is `Barracuda` and `Antelope` is deprecated.

`Barracuda` is a newer InnoDB file format. It supports the `COMPACT`, `REDUNDANT`, `DYNAMIC` and `COMPRESSED` row formats. Tables with large BLOB or TEXT columns in particular could benefit from the dynamic row format.

Future Formats

InnoDB might use new file formats in the future. Each format will have an identifier from 0 to 25, and a name. The names have already been decided, and are animal names listed in an alphabetical order: Antelope, Barracuda, Cheetah, Dragon, Elk, Fox, Gazelle, Hornet, Impala, Jaguar, Kangaroo, Leopard, Moose, Nautilus, Ocelot, Porpoise, Quail, Rabbit, Shark, Tiger, Urchin, Viper, Whale, Xenops, Yak and Zebra.

Checking a Table's File Format.

The information_schema.INNODB_SYS_TABLES table can be queried to see the file format used by a table.

A table's tablespace is tagged with the lowest InnoDB file format that supports the table's row format. So, even if the `Barracuda` file format is enabled, tables that use the `COMPACT` or `REDUNDANT` row formats are tagged with the `Antelope` file format in the information_schema.INNODB_SYS_TABLES table.

Compatibility

Each tablespace is tagged with the id of the most recent file format used by one of its tables. All versions of InnoDB can read tables that use an older file format. However, it can not read from more recent formats. For this reason, each time InnoDB opens a table it checks the tablespace's format, and returns an error if a newer format is used.

This check can be skipped via the innodb_file_format_check variable. Beware that, is InnoDB tries to repair a table in an unknown format, the table are corrupted! This happens on restart if innodb_file_format_check is disabled and the server crashed, or it was closed with fast shutdown.

To downgrade a table from the Barracuda format to Antelope, the table's `ROW_FORMAT` can be set to a value supported by Antelope, via an ALTER TABLE statement. This recreates the indexes.

The Antelope format can be used to make sure that tables work on MariaDB and MySQL servers which are older than 5.5.

See Also

- InnoDB Storage Formats

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Limitations

The InnoDB storage engine has the following limitations.

Limitations on Schema

- InnoDB tables can have a maximum of 1,017 columns. This includes virtual generated columns.
- InnoDB tables can have a maximum of 64 secondary indexes.
- A multicolumn index on InnoDB can use a maximum of 32 columns. If you attempt to create a multicolumn index that uses more than 32 columns, MariaDB returns an Error 1070.

Limitations on Size

- With the exception of variable-length columns (that is, VARBINARY, VARCHAR, BLOB and TEXT), rows in InnoDB have a maximum length of roughly half the page size for 4KB, 8KB, 16KB and 32KB page sizes.
- The maximum size for BLOB and TEXT columns is 4GB. This also applies to LONGBLOB and LONGTEXT.
- MariaDB imposes a row-size limit of 65,535 bytes for the combined sizes of all columns. If the table contains BLOB or TEXT columns, these only count for 9 - 12 bytes in this calculation, given that their content is stored separately.
- 32-bit operating systems have a maximum file size limit of 2GB. When working with large tables using this architecture, configure InnoDB to use smaller data files.
- The maximum size for the combined InnoDB log files is 512GB.
- With tablespaces, the minimum size is 10MB, the maximum varies depending on the InnoDB Page Size.

InnoDB Page Size	Maximum Tablespace Size
4KB	16TB
8KB	32TB
16KB	64TB
32KB	128TB
64KB	256TB

Page Sizes

Using the innodb_page_size system variable, you can configure the size in bytes for InnoDB pages. Pages default to 16KB. There are certain limitations on how you use this variable.

- MariaDB instances using one page size cannot use data files or log files from an instance using a different page size.
- When using a Page Size of 4KB or 8KB, the maximum index key length is lowered proportionately.

InnoDB Page Size	Index Key Length
4KB	768B
8KB	1536B
16KB	3072B

Limitations on Tables

InnoDB has the following table-specific limitations.

- When you issue a DELETE statement, InnoDB doesn't regenerate the table, rather it deletes each row from the table one by one.
- When running MariaDB on Windows, InnoDB stores databases and tables in lowercase. When moving databases and tables in a binary format from Windows to a Unix-like system or from a Unix system to Windows,

you need to rename these to use lowercase.
- When using cascading foreign keys, operations in the cascade don't activate triggers.

Table Analysis

When running ANALYZE TABLE twice on a table in which statements or transactions are running, MariaDB blocks the second ANALYZE TABLE until the statement or transaction is complete. This occurs because the statement or transaction blocks the second ANALYZE TABLE statement from reloading the table definition, which it must do since the old one was marked as obsolete after the first statement.

Table Status

SHOW TABLE STATUS statements do not provide accurate statistics for InnoDB, except for the physical table size.

The InnoDB storage engine does not maintain internal row counts. Transactions isolate writes, which means that concurrent transactions will not have the same row counts.

Auto-incrementing Columns

- When defining an index on an auto-incrementing column, it must be defined in a way that allows the equivalent of `SELECT MAX(col)` lookups on the table.
- Restarting MariaDB may cause InnoDB to reuse old auto-increment values, such as in the case of a transaction that was rolled back.
- When auto-incrementing columns run out of values, INSERT statements generate duplicate-key errors.

Transactions and Locks

- You can modify data on a maximum of 96 * 1023 concurrent transactions that generate undo records.
- Of the 128 rollback segments, InnoDB assigns 32 to non-redo logs for transactions that modify temporary tables and related objects, reducing the maximum number of concurrent data-modifying transactions to 96,000, from 128.000.
- The limit is 32,000 concurrent transactions when all data-modifying transactions also modify temporary tables.
- Issuing a LOCK TABLES statement sets two locks on each table when the innodb_table_locks system variable is enabled (the default).
- When you commit or roll back a transaction, any locks set in the transaction are released. You don't need to issue LOCK TABLES statements when the autocommit variable is enabled, as InnoDB would immediately release the table locks.

This page is licensed: CC BY-SA / Gnu FDL

AUTO_INCREMENT Handling in InnoDB

AUTO_INCREMENT Lock Modes

The innodb_autoinc_lock_mode system variable determines the lock mode when generating AUTO_INCREMENT values for InnoDB tables. These modes allow InnoDB to make significant performance optimizations in certain circumstances.

The innodb_autoinc_lock_mode system variable may be removed in a future release. See MDEV-19577 for more information.

Traditional Lock Mode

When innodb_autoinc_lock_mode is set to `0`, InnoDB uses the traditional lock mode.

In this mode, InnoDB holds a table-level lock for all INSERT statements until the statement completes.

Consecutive Lock Mode

When innodb_autoinc_lock_mode is set to `1`, InnoDB uses the consecutive lock mode.

In this mode, InnoDB holds a table-level lock for all bulk INSERT statements (such as LOAD DATA or INSERT … SELECT) until the end of the statement. For simple INSERT statements, no table-level lock is held. Instead, a lightweight mutex is used which scales significantly better. This is the default setting.

Interleaved Lock Mode

When innodb_autoinc_lock_mode is set to `2`, InnoDB uses the interleaved lock mode.

In this mode, InnoDB does not hold any table-level locks at all. This is the fastest and most scalable mode, but is not safe for statement-based replication.

Setting AUTO_INCREMENT Values

The AUTO_INCREMENT value for an InnoDB table can be set for a table by executing the ALTER TABLE statement and specifying the AUTO_INCREMENT table option:

```
ALTER TABLE tab AUTO_INCREMENT=100;
```

However, in MariaDB 10.2.3 and before, InnoDB stores the table's AUTO_INCREMENT counter in memory. In these versions, when the server restarts, the counter is re-initialized to the highest value found in the table. This means that the above operation can be undone if the server is restarted before any rows are written to the table.

In MariaDB 10.2.4 and later, the AUTO_INCREMENT counter is persistent, so this restriction is no longer present. Persistent, however, does not mean transactional. Gaps may still occur in some cases, such as if a INSERT IGNORE statement fails, or if a user executes ROLLBACK or ROLLBACK TO SAVEPOINT.

For example:

```
CREATE TABLE t1 (pk INT AUTO_INCREMENT PRIMARY KEY, i INT, UNIQUE (i)) ENGINE=InnoDB;

INSERT INTO t1 (i) VALUES (1),(2),(3);
INSERT IGNORE INTO t1 (pk, i) VALUES (100,1);
Query OK, 0 rows affected, 1 warning (0.099 sec)

SELECT * FROM t1;
+----+------+
| pk | i    |
+----+------+
|  1 |    1 |
|  2 |    2 |
|  3 |    3 |
+----+------+

SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `pk` int(11) NOT NULL AUTO_INCREMENT,
  `i` int(11) DEFAULT NULL,
  PRIMARY KEY (`pk`),
  UNIQUE KEY `i` (`i`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1
```

If the server is restarted at this point, then the AUTO_INCREMENT counter will revert to `101`, which is the persistent value set as part of the failed INSERT IGNORE.

```
# Restart server
SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `pk` int(11) NOT NULL AUTO_INCREMENT,
  `i` int(11) DEFAULT NULL,
  PRIMARY KEY (`pk`),
  UNIQUE KEY `i` (`i`)
) ENGINE=InnoDB AUTO_INCREMENT=101 DEFAULT CHARSET=latin1
```

See Also

- AUTO_INCREMENT
- AUTO_INCREMENT FAQ
- LAST_INSERT_ID
- Sequences - an alternative to auto_increment available from MariaDB 10.3

This page is licensed: CC BY-SA / Gnu FDL

Binary Log Group Commit and InnoDB Flushing Performance

Overview

When both innodb_flush_log_at_trx_commit=1 (the default) is set and the binary log is enabled, there is now one less sync to disk inside InnoDB during commit (2 syncs shared between a group of transactions instead of 3).

Durability of commits is not decreased — this is because even if the server crashes before the commit is written to disk by InnoDB, it are recovered from the binary log at next server startup (and it is guaranteed that sufficient information is synced to disk so that such a recovery is always possible).

Switching to Old Flushing Behavior

The old behavior, with 3 syncs to disk per (group) commit (and consequently lower performance), can be selected with the new innodb_flush_log_at_trx_commit=3 option. There is normally no benefit to doing this, however there are a couple of edge cases to be aware of.

Non-durable Binary Log Settings

If innodb_flush_log_at_trx_commit=1 is set and the binary log is enabled, but sync_binlog=0 is set, then commits are not guaranteed durable inside InnoDB after commit. This is because if sync_binlog=0 is set and if the server crashes, then transactions that were not flushed to the binary log prior to the crash are missing from the binary log.

In this specific scenario, innodb_flush_log_at_trx_commit=3 can be set to ensure that transactions are durable in InnoDB, even if they are not necessarily durable from the perspective of the binary log.

One should be aware that if sync_binlog=0 is set, then a crash is nevertheless likely to cause transactions to be missing from the binary log. This will cause the binary log and InnoDB to be inconsistent with each other. This is also likely to cause any replication slaves to become inconsistent, since transactions are replicated through the binary log. Thus it is recommended to set sync_binlog=1. With the group commit improvements introduced in MariaDB 5.3, this setting has much less penalty in recent versions compared to older versions of MariaDB and MySQL.

Recent Transactions Missing from Backups

mariadb-backup and Percona XtraBackup only see transactions that have been flushed to the redo log. With the group commit improvements, there may be a small delay (defined by the binlog_commit_wait_usec system variable) between when a commit happens and when the commit are included in a backup.

Note that the backup will still be fully consistent with itself and the binary log. This problem is normally not an issue in practice. A backup usually takes a long time to complete (relative to the 1 second or so that binlog_commit_wait_usec is normally set to), and a backup usually includes a lot of transactions that were committed during the backup. With this in mind, it is not generally noticeable if the backup does not include transactions that were committed during the last 1 second or so of the backup process. It is just mentioned here for completeness.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Asynchronous I/O

From MariaDB 10.5, InnoDB uses asynchronous I/O to read from and write to disk asynchronously. This forms part of the InnoDB Background Thread Pool.

Stages

Each asynchronous IO operation goes through multiple stages:

1. SUBMITTED – The IO operation is initiated.

- For asynchronous writes, this typically occurs in the buffer pool flushing code.
- For asynchronous reads, this may happen during buffer pool loading at startup or in prefetching logic.

1. COMPLETED_IN_OS – The operating system notifies InnoDB that the I/O operation is complete.

- If using libaio or io_uring, a dedicated thread handles this notification.
- The completed IO operation is then submitted to InnoDB's internal thread pool (tpool).

1. EXECUTING_COMPLETION_TASK – A tpool thread processes the completion task for the IO operation.
2. COMPLETED – The IO operation is fully handled.

Resource Constraints and Queuing Mechanisms

Waiting for IO Slots

The total number of pending asynchronous IO operations is limited by:

```
total_count = number_of_IO_threads * 256
```

where number_of_IO_threads refers to either innodb_io_read_threads or innodb_io_write_threads.

Each IO operation is associated with an IO slot, which contains necessary metadata such as the file handle, operation type, offset, length, and any OS error codes. Initially, all total_count slots are free, but as pending IO requests accumulate, slots get occupied. If all slots are in use, additional IO requests must wait for a free slot.

Queuing Mechanism

The number of completion tasks (EXECUTING_COMPLETION_TASK stage) that can run in parallel is also limited by innodb_io_read_threads or innodb_io_write_threads. If too many IO operations complete simultaneously, they cannot all be processed in parallel and must be queued, respecting the thread limit.

Variables

From MariaDB 11.5, a number of status variables were added to give insight into the above operations:

- innodb_async_reads_pending – Number of read IO operations currently in progress (from SUBMITTED to COMPLETED).

- innodb_async_reads_tasks_running – Number of read IO operations currently in the EXECUTING_COMPLETION_TASK state.
- innodb_async_reads_total_count – Total number of read completion tasks that have finished execution.
- innodb_async_reads_queue_size – Current size of the queue (see Queuing Mechanism).
- innodb_async_reads_wait_slot_sec – Total wait time for a free IO slot (see Waiting for IO Slots).
- innodb_async_reads_total_enqueues – Total number of read operations that were queued (see Queuing Mechanism). Includes those still waiting and making up `innodb_async_reads_queue_size`.

Similar variables exist for write operations:

- innodb_async_writes_pending
- innodb_async_writes_tasks_running
- innodb_async_reads_total_count
- innodb_async_writes_queue_size
- innodb_async_writes_wait_slot_sec
- innodb_async_writes_total_enqueues

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Buffer Pool

The InnoDB storage engine in MariaDB Enterprise Server utilizes the Buffer Pool as a crucial in-memory cache. This Buffer Pool stores recently accessed data pages, enabling faster retrieval for subsequent requests. Recognizing patterns of access, InnoDB also employs predictive prefetching, caching nearby pages when sequential access is detected. To manage memory efficiently, a least recently used (LRU) algorithm is used to evict older, less frequently accessed pages.

To optimize server restarts, the Buffer Pool's contents can be preserved across shutdowns. At shutdown, the page numbers of all pages residing in the Buffer Pool are recorded. Upon the next startup, InnoDB reads this dump of page numbers and reloads the corresponding data pages from their respective data files, effectively avoiding a "cold" cache scenario.

The size of each individual page within the Buffer Pool is determined by the setting of the innodb_page_size system variable.

How the Buffer Pool Works

The buffer pool attempts to keep frequently-used blocks in the buffer, and so essentially works as two sublists, a *new* sublist of recently-used information, and an *old* sublist of older information. By default, 37% of the list is reserved for the old list.

When new information is accessed that doesn't appear in the list, it is placed at the top of the old list, the oldest item in the old list is removed, and everything else bumps back one position in the list.

When information is accessed that appears in the *old* list, it is moved to the top the new list, and everything above moves back one position.

innodb_buffer_pool_size

The most important server system variable is innodb_buffer_pool_size. This size should contain most of the active data set of your server so that SQL request can work directly with information in the buffer pool cache. Starting at several gigabytes of memory is a good starting point if you have that RAM available. Once warmed up to its normal load there should be very few innodb_buffer_pool_reads compared to innodb_buffer_pool_read_requests. Look how these values change over a minute. If the change in innodb_buffer_pool_reads is less than 1% of the change in innodb_buffer_pool_read_requests then you have a good amount of usage. If you are getting the status variable

innodb_buffer_pool_wait_free increasing then you don't have enough buffer pool (or your flushing isn't occurring frequently enough).

The larger the size, the longer it takes to initialize.

> **Note:**
>
> Make sure that the size is not too large, because this can cause swapping, which more than undoes the benefits of a large buffer pool.

> **Warning:**
>
> **Using ColumnStore?**
>
> The standard recommendation to use a large portion of your RAM for the InnoDB buffer pool does not apply if you are also running MariaDB ColumnStore on the same server. ColumnStore reserves 75% of system RAM by default. Over-allocating the InnoDB buffer pool in a combined environment will likely trigger the Linux OOM killer.
>
> Please refer to the ColumnStore Memory Requirements for accurate sizing and validation instructions.

The buffer pool can be set dynamically. See Setting Innodb Buffer Pool Size Dynamically.

Buffer Pool Changes

> **Warning:**
>
> From MariaDB 10.11.12 / 11.4.6 / 11.8.2, there are significant changes to the InnoDB buffer pool behavior.

MariaDB server deprecates and ignores the `innodb_buffer_pool_chunk_size`. Now, the buffer pool size is changed in arbitrary 1-megabyte increments, all the way up to `innodb_buffer_pool_size_max`, which must be specified at startup.

If `innodb_buffer_pool_size_max` is `0` or not specified, it defaults to the `innodb_buffer_pool_size` value.

The `innodb_buffer_pool_size_auto_min` variable specifies the minimum size the buffer pool can be shrunk to by a memory pressure event. When a memory pressure event occurs, MariaDB server attempts to shrink `innodb_buffer_pool_size` halfway between its current value and the `innodb_buffer_pool_size_auto_min` value. If `innodb_buffer_pool_size_auto_min` is not specified or `0`, its default value is adjusted to `innodb_buffer_pool_size` — in other words, memory pressure events are disregarded by default.

The minimum `innodb_buffer_pool_size` is 320 pages (256*5/4). With the default value of `innodb_page_size=16k`, this corresponds to 5 MiB. However, since `innodb_buffer_pool_size` includes the memory allocated for the block descriptors, the minimum is effectively `innodb_buffer_pool_size=6m`.

> **Tip:**
>
> When the buffer pool is shrunk, InnoDB tries to inform the operating system that the underlying memory for part of the virtual address range is no longer needed and may be zeroed out. On many POSIX-like systems this is done by `madvise(MADV_DONTNEED)` where available (Linux, FreeBSD, NetBSD, OpenBSD, Dragonfly BSD, IBM AIX, Apple macOS). On Microsoft Windows, `VirtualFree(MEM_DECOMMIT)` is invoked. On many systems, there is also `MADV_FREE`, which would be a deferred variant of `MADV_DONTNEED`, not freeing the virtual memory mapping immediately. We prefer immediate freeing so that the resident set size of the process reflects the current `innodb_buffer_pool_size` value. Shrinking the buffer pool is a rarely executed intensive operation, and the immediate configuration of the MMU mappings should not incur significant additional penalty.

The `Innodb_buffer_pool_resize_status` variable is removed. Issuing `SET GLOBAL innodb_buffer_pool_size` blocks until the buffer pool has been resized or the operation was aborted by a `KILL` or `SHUTDOWN` command, a client disconnect, or an interrupt.

innodb_old_blocks_pct and innodb_old_blocks_time

The default 37% reserved for the old list can be adjusted by changing the value of innodb_old_blocks_pct. It can accept anything between 5% and 95%.

The innodb_old_blocks_time variable specifies the delay before a block can be moved from the old to the new sublist. `0` means no delay, while the default has been set to `1000`.

Before changing either of these values from their defaults, make sure you understand the impact and how your system currently uses the buffer. Their main reason for existence is to reduce the impact of full table scans, which are usually infrequent, but large, and previously could clear everything from the buffer. Setting a non-zero delay could help in situations where full table scans are performed in quick succession.

Temporarily changing these values can also be useful to avoid the negative impact of a full table scan, as explained in InnoDB logical backups.

Dumping and Restoring the Buffer Pool

When the server starts, the buffer pool is empty. As it starts to access data, the buffer pool will slowly be populated. As more data are accessed, the most frequently accessed data are put into the buffer pool, and old data may be evicted. This means that a certain period of time is necessary before the buffer pool is really useful. This period of time is called the warmup.

InnoDB can dump the buffer pool before the server shuts down, and restore it when it starts again. If this feature is used, no warmup is necessary. Use the innodb_buffer_pool_dump_at_shutdown and innodb_buffer_pool_load_at_startup system variables to enable or disable the buffer pool dump at shutdown and the restore at startup respectively.

It is also possible to dump the InnoDB buffer pool at any moment while the server is running, and it is possible to restore the last buffer pool dump at any moment. To do this, the special innodb_buffer_pool_dump_now and innodb_buffer_pool_load_now system variables can be set to ON. When selected, their value is always OFF.

A buffer pool restore, both at startup or at any other moment, can be aborted by setting innodb_buffer_pool_load_abort to ON.

The file which contains the buffer pool dump is specified via the innodb_buffer_pool_filename system variable.

See Also

- InnoDB Change Buffering
- Information Schema INNODB_BUFFER_POOL_STATS Table
- Setting Innodb Buffer Pool Size Dynamically

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Change Buffering

> **Warning:**
>
> The change buffer has been disabled by default from MariaDB 10.5.15, MariaDB 10.6.7, MariaDB 10.7.3 and MariaDB 10.8.2 (MDEV-27734), was deprecated and ignored from MariaDB 10.9.0 (MDEV-27735), and was removed in MariaDB 11.0.0 (MDEV-29694).

Benchmarks attached to MDEV-19514 show that the change buffer sometimes reduces performance, and in the best case seem to bring a few per cent improvement to throughput. However, such improvement could come with a price: If the buffered changes are never merged (MDEV-19514, motivated by the reduction of random crashes and the removal of an innodb_force_recovery option that can inflict further corruption), then the InnoDB system tablespace can grow out of control (MDEV-21952).

`INSERT`, `UPDATE`, and `DELETE` statements can be particularly heavy operations to perform, as all indexes need to be updated after each change. For this reason these changes are often buffered.

Pages are modified in the buffer pool, and not immediately on disk. After all the records that cover the changes to a data page have been written to the InnoDB redo log, the changed page may be written (''flushed'') to a data file. Pages that have been modified in memory and not yet flushed are called dirty pages.

The Change Buffer is an optimization that allows some data to be modified even though the data page does not exist in the buffer pool. Instead of modifying the data in its final destination, we would insert a record into a special Change Buffer that resides in the system tablespace. When the page is read into the buffer pool for any reason, the buffered changes are applied to it.

The Change Buffer only contains changes to secondary index leaf pages.

In the days of old, only inserted rows could be buffered, so this buffer was called *Insert Buffer*. The old name still appears in several places, for example in the output of SHOW ENGINE INNODB STATUS.

Inserts to `UNIQUE` secondary indexes cannot be buffered unless unique_checks=0 is used. This may sometimes allow duplicates to be inserted into the `UNIQUE` secondary index. Much of the time, the `UNIQUE` constraint would be checked because the change buffer could only be used if the index page is not located in the buffer pool.

When rows are deleted, a flag is set, thus rows are not immediately deleted. Delete-marked records may be purged after the transaction has been committed and any read views that were created before the commit have been closed. Delete-mark and purge buffering of any secondary indexes is allowed.

`ROLLBACK` never makes use of the change buffer; it would force a merge of any changes that were buffered during the execution of the transaction.

The Change Buffer is an optimization because:

- Some random-access page reads are transformed into modifications of change buffer pages.
- A change buffer page can be modified several times in memory and be flushed to disk only once.
- Dirty pages are flushed together, so the number of IO operations is lower.

If the server crashes or is shut down, the Change Buffer might not be empty. The Change Buffer resides in the InnoDB system tablespace, covered by the write-ahead log, so they can be applied at server restart. A shutdown with innodb_fast_shutdown=0 will merge all buffered changes.

There is no background task that merges the change buffer to the secondary index pages. Changes are only merged on demand.

> **Warning:**
>
> The Change Buffer was removed in MariaDB 11.0 because it has been a prominent source of corruption bugs that have been extremely hard to reproduce.

The main server system variable here is innodb_change_buffering, which determines which form of change buffering, if any, to use.

The following settings are available:

- inserts
  - Only buffer insert operations
- deletes
  - Only buffer delete operations
- changes
  - Buffer both insert and delete operations

- purges
  - Buffer the actual physical deletes that occur in the background
- all
  - Buffer inserts, deletes and purges. Default setting from MariaDB 5.5 until MariaDB 10.5.14, MariaDB 10.6.6, MariaDB 10.7.2 and MariaDB 10.8.1.
- none
  - Don't buffer any operations. Default from MariaDB 10.5.15, MariaDB 10.6.7, MariaDB 10.7.3 and MariaDB 10.8.2.

Modifying the value of this variable only affects the buffering of new operations. The merging of already buffered changes is not affected.

The innodb_change_buffer_max_size system variable determines the maximum size of the change buffer, expressed as a percentage of the buffer pool.

See Also

- InnoDB Buffer Pool

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Data Scrubbing

> **Warning:**
>
> Most of the background and redo log scrubbing code has been removed in MariaDB 10.5.2. See MDEV-15528 and MDEV-21870.

Sometimes there is a requirement that when some data is deleted, it is really gone. This might be the case when one stores user's personal information or some other sensitive data. Normally though, when a row is deleted, the space is only marked as free on the page. It may eventually be overwritten, but there is no guarantee when that will happen. A copy of the deleted rows may also be present in the log files.

Support for InnoDB data scrubbing: Background threads periodically scan tablespaces and logs and remove all data that should be deleted. The number of background threads for tablespace scans is set by innodb-encryption-threads. Log scrubbing happens in a separate thread.

To configure scrubbing one can use the following variables:

innodb-background-scrub-data-check-interval	Seconds	Check at this interval if tablespaces needs scrubbing. Deprecated and ignored.
innodb-background-scrub-data-compressed	Boolean	Enable scrubbing of compressed data by background threads. Deprecated and ignored.
innodb-background-scrub-data-interval	Seconds	Scrub spaces that were last scrubbed longer than this many seconds ago. Deprecated and ignored.
innodb-background-scrub-data-uncompressed	Boolean	Enable scrubbing of uncompressed data by background threads. Deprecated and ignored.
innodb-immediate-scrub-data-uncompressed	Boolean	Enable scrubbing of uncompressed data.
innodb-scrub-log	Boolean	Enable redo log scrubbing. Deprecated and ignored.
innodb-scrub-log-speed	Bytes/sec	Redo log scrubbing speed in bytes/sec. Deprecated and ignored.

Redo log scrubbing did not fully work as intended, and was deprecated and ignored in MariaDB 10.5.2 (MDEV-21870). If old log contents should be kept secret, enabling innodb_encrypt_log or setting a smaller innodb_log_file_size could help.

The Information Schema INNODB_TABLESPACES_SCRUBBING table contains scrubbing information.

Thanks

- Scrubbing was donated to the MariaDB project by Google.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Doublewrite Buffer

The InnoDB doublewrite buffer was implemented to recover from half-written pages. This can happen when there's a power failure while InnoDB is writing a page to disk. On reading that page, InnoDB can discover the corruption from the mismatch of the page checksum. However, in order to recover, an intact copy of the page would be needed.

The double write buffer provides such a copy.

Whenever InnoDB flushes a page to disk, it is first written to the double write buffer. Only when the buffer is safely flushed to disk will InnoDB write the page to the final destination. When recovering, InnoDB scans the double write buffer and for each valid page in the buffer checks if the page in the data file is valid too.

Doublewrite Buffer Settings

To turn off the doublewrite buffer, set the innodb_doublewrite system variable to `0`. This is safe on filesystems that write pages atomically - that is, a page write fully succeeds or fails. But with other filesystems, it is not recommended for production systems. An alternative option is atomic writes. See atomic write support for more details.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Flush Method

> **Note:**
>
> The `innodb_flush_method` variable has been deprecated from MariaDB 11.0.

> **Note:**
>
> For the OS-level mechanisms behind the values used on this page (`fdatasync()`, `fsync()`, `O_DIRECT`, `O_DSYNC`, and their Windows counterparts), see Storage I/O: Buffering and Persistence.

Overview

innodb_flush_method is a variable that has been available in MariaDB (and MySQL) for a long time. It defines the method used to flush data to InnoDB data files and log files, which can affect I/O throughput quite a bit.

The default setting for this variable initially was `fsync`, but was changed to `O_DIRECT` in MariaDB 10.6. The variable has been deprecated in MariaDB 11.0 and replaced by other settings.

This page describes the main settings and examines when it makes sense or not to change the default values.

Options and Startup Variables

First of all, let's look at what the different options for `innodb_flush_method` mean, and how they map to the new Boolean options that can be changed while the server is running. Those new options are available from MariaDB

11.0. There are 2 sets of Boolean options, controlling how the InnoDB write-ahead log `ib_logfile0` as well as the persistent data files are written:

1. `innodb_data_file_write_through`, `innodb_log_file_write_through`: Controls if each write is made durable without an additional system call, such as `fdatasync()` or `fsync()`. This can be useful if there is a battery-backed cache, or if the storage stack can make efficient use of FUA[7].
2. `innodb_data_file_buffering`, `innodb_log_file_buffering`: Controls if the operating system's file system cache is used. Those options are disabled by default, to allow efficient writes by DMA[8] from an aligned memory buffer. Starting with MariaDB Server 10.8, the log block size adapts to the underlying physical block size (typically 512 or 4096 bytes), to avoid an inefficient read-before-write anomaly. This is handled via the `innodb_log_write_ahead_size` variable (MDEV-33894).

The non-default setting `innodb_flush_method=O_DSYNC` disables buffering and enables write-through on all persistent InnoDB files. On suitable hardware, this can provide the best write performance.

The old default setting `innodb_flush_method=fsync` disables write-through and enables `innodb_data_file_buffering=ON`. The setting of `innodb_log_file_buffering` is not affected. The settings `littlesync` and `nosync`, as well as the Microsoft Windows setting `normal` are treated in the same way.

The MariaDB Server 10.6 default setting `innodb_flush_method=O_DIRECT` leaves the new Boolean options at their default values. On Microsoft Windows, this option is also known as `unbuffered` and `async_unbuffered`.

The unsafe setting `innodb_flush_method =O_DIRECT_NO_FSYNC` is treated like `O_DIRECT`. You can disable any use of `fdatasync()` or `fsync()` system calls, but that means you lose all crash-safety guarantees. To do that, set `debug-no-sync =ON`. Starting with MariaDB Server 11.0, this unsafe variable affects InnoDB as well.

> **Note:**
>
> **Effects of bypassing the file system buffering:**
>
> When file system buffering is bypassed, InnoDB attempts to open the file in `O_DIRECT` mode (on Windows, with `FILE_FLAG_NO_BUFFERING`) if the file system supports that. When buffering is enabled or direct I/O is not supported, a file system cache can be maintained by the operating system kernel. This cache can make some read workloads much faster — some read workloads significantly benefit from setting `innodb_data_file_buffering=ON` (effectively disabling `O_DIRECT`).

Behavior Change in MariaDB 10.6

Let's now look at why the default behavior was changed in 10.6. Looking at how most modern file systems handle writes, it seems clear that the penalty of writing twice (to the file system cache and then to the disk) is not needed, and typically slows down the system. With `O_DIRECT`, there is no such double writing, as everything goes straight to disk.

However, there are still cases where reads are affected negatively by using `O_DIRECT`. Note that this only affects I/O-bound workloads. Ideally, a requested data page is cached in an internal MariaDB buffer (in this case the, InnoDB buffer pool), and the data can be accessed without involving the operating system, no matter how the data is written to disk.

There is a possibility that a data page is written from the buffer pool to the file system, evicted from the buffer pool, and again read from the file system. In this kind of scenario, the operating system could provide an additional layer of cache between the buffer pool and the storage, allowing the "read" to be a simple matter of copying the data from the kernel buffer to the buffer pool. The worst-case scenario is that the page needs to be read from actual storage, which can be orders of magnitude slower than copying from the file system cache.

If `innodb_buffer_pool_size` is very small compared to the available memory, the file system cache may effectively become the main cache. This can be useful when a MariaDB Server instance is small, or used sporadically on a

system that runs many other services. In such an environment, it can make sense to use `innodb_flush_method=fsync`, and to let the operating system tune the use of buffers according to the variable workload. From its early days decades ago, the Linux kernel tries to use all “free” memory for the file system cache.

Another thing to consider is read-ahead, or loading data into a cache in the anticipation of it being needed soon. While read-ahead mechanisms into the InnoDB buffer pool exist, the read-ahead into the Linux file system cache may perform much better. Part of the problem can be that MariaDB always reads InnoDB pages one at a time, giving an advantage to the operating system’s file system cache.

To summarize: Changing the default of `innodb_flush_method` to `O_DIRECT` in MariaDB 10.6 was the right choice — however, there are cases illustrated above where you might want to change this behavior back to the original one.

Behavior Change in MariaDB 11.0

> **Note:**
>
> The `innodb_flush_method` variable has been deprecated, but can still be used. The preferred way to use it is by setting one of the Boolean variables via the `SET GLOBAL` statement. The usage of those variables is explained in the Options and Startup Variables section.

From MariaDB 11.0, the flushing behavior of InnoDB should no longer be controlled by `innodb_flush_method`, but rather by the following variables, which can be modified with a `SET GLOBAL` statement while the server is running:

1. `innodb_log_file_buffering`: This is best left to its default value (`OFF`), except when backing up a write-heavy server.
2. `innodb_data_file_buffering`: The default is `OFF`; set it to `ON` if you encounter read performance problems.
3. `innodb_log_file_write_through`, `innodb_data_file_write_through`: The default is `OFF` for both variables; setting them to `ON` can improve or degrade write performance, depending on whether or not the hardware supports FUA (Forced Unit Access).

> **Note:**
>
> A `SET GLOBAL` change to `innodb_log_file_buffering` or `innodb_log_file_write_through` can be silently ignored — for example, while an `innodb_log_file_size` resize is in progress, or when the log is mapped to persistent memory (PMEM). Confirm the change took effect by reading the value back, as described for `innodb_log_file_buffering` and `innodb_log_file_write_through`.

FUA (Forced Unit Access)

FUA is a hardware-level flag used during write operations to ensure data integrity. When InnoDB sends a write command with the FUA flag, it tells the storage device (SSD or HDD) that the write must be written directly to the non-volatile, stable media before the device can report that the task is finished.

> **Warning:**
>
> FUA only works if your hardware (and its drivers) actually supports it. Some cheaper consumer SSDs claim to support FUA but secretly ignore it to make their benchmark speeds look faster — this can lead to data corruption during a power failure!

InnoDB is a transactional storage engine, meaning it promises that once a transaction is committed, it won’t be lost even if the power goes out. To keep this promise, InnoDB has to flush data from the computer’s RAM to the physical disk. Traditionally, this involved two steps:

1. Write: Send data to the disk (where it might sit in the disk’s own temporary cache).
2. `fsync`: Send a separate command telling the disk to push everything in the cache to the physical disk.

FUA combines these into one step. By using FUA, InnoDB instructs the operation system to write out data and demands that success is only reported when that data is safe on the physical media.

This has these benefits:

- Performance: It eliminates the need for a second `fsync` system call, which can reduce latency and CPU overhead.
- Efficiency: FUA is fine-grained. Instead of forcing the disk to flush its *entire* cache (which `fsync` does), FUA only forces the specific block of data you are currently writing.

Starting with MariaDB 10.6, the way InnoDB handles flushing changed significantly. See this section for details.

FUA can be detrimental for data file write performance, but beneficial for log writes. For the latter, set `innodb_data_file_write_through` to `OFF` to improve write performance.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Lock Modes

Locks are acquired by a transaction to prevent concurrent transactions from modifying, or even reading, some rows or ranges of rows. This is done to make sure that concurrent write operations never collide.

InnoDB supports a number of lock modes.

Shared and Exclusive Locks

The two standard row-level locks are *share locks*(S) and *exclusive locks*(X).

A shared lock is obtained to read a row, and allows other transactions to read the locked row, but not to write to the locked row. Other transactions may also acquire their own shared locks.

An exclusive lock is obtained to write to a row, and stops other transactions from locking the same row. It's specific behavior depends on the isolation level; the default (REPEATABLE READ), allows other transactions to read from the exclusively locked row.

Locks Apply to Index Records, Not Logical Table Rows

In InnoDB, when using `LOCK IN SHARE MODE` under the `READ COMMITTED` isolation level, it is critical to understand that locking reads are based on index records and not on abstract table rows. This difference is important, when using locking reads like these:

```
SELECT ... LOCK IN SHARE MODE;
SELECT ... FOR UPDATE;
```

InnoDB's locking engine works at the index record level, but a query logically focuses on a row. Therefore, based on the isolation level and data creation order, lock conflicts can be avoided if two simultaneous transactions access the same row through different indexes.

Index Reads and Lock Behavior

When a secondary index matches a transaction, InnoDB:

- Reads only the secondary index
- locks the secondary index records
- Avoids accessing the primary index record

If another transaction alters a column that is not included in the secondary index, only the primary index record can be locked. In that case, the two transactions lock different index records without blocking each other.

Example

The following scenario illustrates a case where `SELECT ... LOCK IN SHARE MODE` does not prevent a concurrent update, even though both statements refer to the same logical rows:

```
CREATE TABLE tbl_1 (
  col_1_1 TIMESTAMP,
  col_1_2 VARCHAR(100),
  col_1_3 MEDIUMINT,
  col_1_4 INT,
  col_1_5 DOUBLE,
  col_1_6 NUMERIC,
  pkid_1 INT,
  INDEX idx_1_1 (col_1_4, col_1_2, col_1_3, col_1_5),
  PRIMARY KEY (col_1_4, pkid_1)
) ENGINE=InnoDB;
```

Here:

- `PRIMARY KEY (col_1_4, pkid_1)` -> Clustered Index (Primary Index Record)
- `INDEX idx_1_1 (col_1_4, col_1_2, col_1_3, col_1_5)` -> Secondary Index (Secondary Index Record)

Sample Data

```
INSERT INTO tbl_1 VALUES
('2025-12-19 16:46:56','str_002',2,3,1,1,0),
('2025-12-19 16:46:56','str_002',2,3,2,1,1),
('2025-12-19 16:46:57','str_002',1,3,2,1,2),
('2025-12-19 16:46:56','str_002',1,3,2,1,3),
('2025-12-19 16:46:57','str_002',1,3,2,1,4),
('2025-12-19 16:46:56','str_002',1,1,2,1,5),
('2025-12-19 16:46:56','str_002',1,2,2,1,6);
```

Connection 1

```
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN;

SELECT col_1_4, col_1_2, pkid_1 FROM tbl_1 WHERE col_1_3 > 1 LOCK IN SHARE MODE;
```

What happens

- The query makes use of the secondary index `idx_1_1`.
- The query reads `col_1_4`, `col_1_2`, and `pkid_1`, which are part of `idx_1_1`.
- This is a covering or secondary index read.
- InnoDB skips the primary index and only applies shared locks to records in `idx_1_1`.

Default Connection (Concurrent)

```
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN;

UPDATE tbl_1
SET col_1_1 = '2025-12-19 16:46:56'
WHERE col_1_4 = 3;
```

What happens

- The `UPDATE` changes `col_1_1`, which is not present in `idx_1_1`.
- The PRIMARY KEY record must be locked in InnoDB.
- No changes to `idx_1_1` are required.

Why No Blocking Occurs

1. Connection 1's `SELECT` performs a "covering read" on the secondary index `idx_1_1`. Since the index contains all required data, InnoDB only uses shared locks on secondary index records.
2. Default Connection's `UPDATE` modifies column `col_1_1`, which is not part of that secondary index. Therefore, an exclusive lock is acquired on the Primary Index record (Clustered Index).
3. The Result: Since these are two different index records, there is no conflict between the shared and exclusive locks.

How to Ensure Blocking

To cause a lock conflict, the locking read must access the primary key record (Clustered Index). This ensures that the *shared lock* (S) is applied to the location where the update operation (exclusive lock, X) will later be required. This happens when:

- The SELECT contains a column that the secondary index does not cover (e.g., using `SELECT *` or selecting a non-indexed column), or
- If the secondary index itself is insufficient to satisfy the query, InnoDB must access the full row in the primary index, thereby placing the necessary locks to ensure blocking.

Example:

```
-- The UPDATE is blocked since col_1_1 is not included in the secondary index idx_1_1.
SELECT col_1_1, col_1_4, pkid_1
FROM tbl_1
WHERE col_1_3 > 1
LOCK IN SHARE MODE;
```

The UPDATE will now wait while the clustered index is accessed.

Intention Locks

InnoDB also permits table locking, and to allow locking at both table and row level to co-exist gracefully, a series of locks called *intention locks* exist.

An *intention shared lock*(IS) indicates that a transaction intends to set a shared lock.

An *intention exclusive lock*(IX) indicates that a transaction intends to set an exclusive lock.

Whether a lock is granted or not can be summarised as follows:

- An X lock is not granted if any other lock (X, S, IX, IS) is held.
- An S lock is not granted if an X or IX lock is held. It is granted if an S or IS lock is held.
- An IX lock is not granted if in X or S lock is held. It is granted if an IX or IS lock is held.
- An IS lock is not granted if an X lock is held. It is granted if an S, IX or IS lock is held.

AUTO_INCREMENT Locks

Locks are also required for auto-increments - see AUTO_INCREMENT handling in InnoDB.

Gap Locks

With the default isolation level, `REPEATABLE READ`, and, until MariaDB 10.4, the default setting of the innodb_locks_unsafe_for_binlog variable, a method called gap locking is used. When InnoDB sets a shared or exclusive lock on a record, it's actually on the index record. Records will have an internal InnoDB index even if they don't have a unique index defined. At the same time, a lock is held on the gap before the index record, so that another transaction cannot insert a new index record in the gap between the record and the preceding record.

The gap can be a single index value, multiple index values, or not exist at all depending on the contents of the index.

If a statement uses all the columns of a unique index to search for unique row, gap locking is not used.

Similar to the shared and exclusive intention locks described above, there can be a number of types of gap locks. These include the shared gap lock, exclusive gap lock, intention shared gap lock and intention exclusive gap lock.

Gap locks are disabled if the innodb_locks_unsafe_for_binlog system variable is set (until MariaDB 10.4), or the isolation level is set to `READ COMMITTED`.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Log Archiving

> **Note:**
>
> This functionality is available from MariaDB 13.0.

Overview

By default, the InnoDB write-ahead log is managed as a ring buffer in a single file (`ib_logfile0`). This is efficient for write performance, but log records are eventually overwritten as the ring wraps, which limits the options for point-in-time recovery (PITR) and incremental backups.

InnoDB log archiving replaces the ring buffer with a sequence of pre-allocated, sequentially-named log files. When one file fills up, the server creates the next. Once a checkpoint completes in a new file, the previous file is marked read-only — at which point you can safely move it to long-term storage. The result is a continuous log history for the period archiving was active.

> **Warning:**
>
> No shipped backup tool yet generates or restores backups in the `innodb_log_archive=ON` format. `mariadb-backup` only supports the legacy `ib_logfile0` format and fails when the server is running with `innodb_log_archive=ON`. A backup tool that uses this format is being worked on.

Enabling Log Archiving

To enable log archiving, set the `innodb_log_archive` system variable to `ON`. This variable is dynamic and can be changed while the server is running:

```
SET GLOBAL innodb_log_archive=ON;
```

When archiving is active, the server creates files in the data directory using the naming convention `ib_`*`lsn`*`.log`, where *lsn* is a 16-character hexadecimal value identifying the log sequence number that maps to file offset `0x3000` (12288) — the start of record payload within each file.

File Format

Each archive log file has a 12 KiB header. Completed checkpoints in the file are recorded in this header as 64-bit big-endian offsets to a `FILE_MODIFY` / `FILE_CHECKPOINT` mini-transaction within the file. With `innodb_encrypt_log=OFF`, the header can hold up to 1536 checkpoint slots; if a file produces more checkpoints than that, further checkpoints overwrite the last slot until the next log file is started.

Technical Constraints

- **Log file size:** When `innodb_log_archive` is `ON`, changes to `innodb_log_file_size` take effect when the current log file fills up and a new file is created. There is no special upper bound on `innodb_log_file_size` in

this mode.

- **Encryption:** While `innodb_log_archive` is `ON`, `innodb_encrypt_log` and related encryption parameters cannot be changed. Each `ib_lsn.log` file must use the same encryption parameters. To change encryption, set `innodb_log_archive=OFF` and restart the server — this permanently discards the archived log history.
- **Startup:** With `innodb_log_archive=ON`, the server refuses to start if `ib_logfile0` exists in the data directory.
- **Data Dictionary:** Log archiving tracks changes to InnoDB tables. It does not cover `.frm` files or other non-InnoDB metadata, which means point-in-time recovery of DDL operations is limited.

Inspecting the Archive State

When `innodb_log_archive=ON`, both the `innodb_log_archive` system variable and the `Innodb_lsn_archived` status variable remain constant by design. `Innodb_lsn_archived` reports the LSN since which a complete InnoDB log archive is available — that is, the first checkpoint in the first archived log file — and does not advance as new log records are written. It is the static lower bound of the archive.

What does advance during normal operation are the `Innodb_lsn_current` status variable (the LSN of the most recent log write) and the `Innodb_lsn_last_checkpoint` status variable (the LSN of the most recent completed checkpoint). Query these to observe ongoing log activity:

```
SELECT @@GLOBAL.innodb_log_archive AS archiving;

SELECT VARIABLE_NAME, VARIABLE_VALUE
FROM   INFORMATION_SCHEMA.GLOBAL_STATUS
WHERE  VARIABLE_NAME IN ('INNODB_LSN_ARCHIVED',
                         'INNODB_LSN_CURRENT',
                         'INNODB_LSN_LAST_CHECKPOINT');
```

Managing Archived Log Files

Once a checkpoint completes in a new log file, the previous file is marked read-only. You can safely move read-only `ib_lsn.log` files to offline storage.

> **Warning:**
>
> Before *deleting* an archived log file, check that `Innodb_lsn_last_checkpoint` is not older than the start LSN embedded in the first writable `ib_lsn.log` file. If it is, crash recovery still requires the previous log file — deleting it leaves the server unable to recover.

Performing Recovery from Archived Logs

The startup parameters `innodb_log_recovery_start` and `innodb_log_recovery_target` define the range of the log replay. Both default to `0`:

- `innodb_log_recovery_start=0` means start from the latest completed checkpoint (always present in one of the last two `ib_lsn.log` files).
- `innodb_log_recovery_target=0` means replay to the end of the available log.

See Point-In-Time Recovery (InnoDB log archiving) for the full procedure.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Monitors

The InnoDB Monitor refers to particular kinds of monitors included in MariaDB and since the early versions of MySQL.

There are four types: the standard InnoDB monitor, the InnoDB Lock Monitor, InnoDB Tablespace Monitor and the InnoDB Table Monitor.

Standard InnoDB Monitor

The standard InnoDB Monitor returns extensive InnoDB information, particularly lock, semaphore, I/O and buffer activity:

To enable the standard InnoDB Monitor, from MariaDB 10.0.14, set the innodb_status_output system variable to 1. Before MariaDB 10.0.14, running the following statement was the method used:

```
CREATE TABLE innodb_monitor (a INT) ENGINE=INNODB;
```

To disable the standard InnoDB monitor, either set the system variable to zero, or, before MariaDB 10.0.14, drop the table

```
DROP TABLE innodb_monitor;
```

The CREATE TABLE and DROP TABLE method of enabling and disabling the InnoDB Monitor has been deprecated, and may be removed in a future version of MariaDB.

For a description of the output, see SHOW ENGINE INNODB STATUS.

InnoDB Lock Monitor

The InnoDB Lock Monitor displays additional lock information.

To enable the InnoDB Lock Monitor, the standard InnoDB monitor must be enabled. Then, from MariaDB 10.0.14, set the innodb_status_output_locks system variable to 1.
Before MariaDB 10.0.14, running the following statement was the method used:

```
CREATE TABLE innodb_lock_monitor (a INT) ENGINE=INNODB;
```

To disable the standard InnoDB monitor, either set the system variable to zero, or, before MariaDB 10.0.14, drop the table

```
DROP TABLE innodb_lock_monitor;
```

The CREATE TABLE and DROP TABLE method of enabling and disabling the InnoDB Lock Monitor has been deprecated, and may be removed in a future version of MariaDB.

InnoDB Tablespace Monitor

The InnoDB Tablespace Monitor is deprecated, and may be removed in a future version of MariaDB.

Enabling the Tablespace Monitor outputs a list of file segments in the shared tablespace to the error log, and validates the tablespace allocation data structures.

To enable the Tablespace Monitor, run the following statement:

```
CREATE TABLE innodb_tablespace_monitor (a INT) ENGINE=INNODB;
```

To disable it, drop the table:

```
DROP TABLE innodb_tablespace_monitor;
```

InnoDB Table Monitor

The InnoDB Table Monitor is deprecated, and may be removed in a future version of MariaDB.

Enabling the Table Monitor outputs the contents of the InnoDB internal data dictionary to the error log every fifteen seconds.

To enable the Table Monitor, run the following statement:

```
CREATE TABLE innodb_table_monitor (a INT) ENGINE=INNODB;
```

To disable it, drop the table:

```
DROP TABLE innodb_table_monitor;
```

SHOW ENGINE INNODB STATUS

The SHOW ENGINE INNODB STATUS statement can be used to obtain the standard InnoDB Monitor output when required, rather than sending it to the error log. It will also display the InnoDB Lock Monitor information if the innodb_status_output_locks system variable is set to `1`.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Page Compression

Overview

InnoDB page compression provides a way to compress InnoDB tables.

Use Cases

- InnoDB page compression can be used on any storage device and any file system.
- InnoDB page compression is most efficient on file systems that support sparse files. See Saving Storage Space with Sparse Files for more information.
- InnoDB page compression is most beneficial on solid state drives (SSDs) and other flash storage. See Optimized for Flash Storage for more information.
- InnoDB page compression performs best when your storage device and file system support atomic writes, since that allows the InnoDB doublewrite buffer to be disabled. See Atomic Write Support for more information.

Comparison with the `COMPRESSED` Row Format

InnoDB page compression is a modern way to compress your InnoDB tables. It is similar to InnoDB's COMPRESSED row format, but it has many advantages. Some of the differences are:

- With InnoDB page compression, compressed pages are immediately decompressed after being read from the tablespace file, and only uncompressed pages are stored in the buffer pool. In contrast, with InnoDB's COMPRESSED row format, compressed pages are decompressed immediately after they are read from the tablespace file, and both the uncompressed and compressed pages are stored in the buffer pool. This means that the COMPRESSED row format uses more space in the buffer pool than InnoDB page compression does.
- With InnoDB page compression, pages are compressed just before being written to the tablespace file. In contrast, with InnoDB's COMPRESSED row format, pages are re-compressed immediately after any changes, and the compressed pages are stored in the buffer pool alongside the uncompressed pages. These changes are then occasionally flushed to disk. This means that the COMPRESSED row format re-compresses data more frequently than InnoDB page compression does.
- With InnoDB page compression, multiple compression algorithms are supported. In contrast, with InnoDB's COMPRESSED row format, zlib is the only supported compression algorithm. This means that the

COMPRESSED row format has less compression options than InnoDB page compression does.

In general, InnoDB page compression is superior to the COMPRESSED row format.

Comparison with Storage Engine-Independent Column Compression

- See Storage Engine-Independent Column Compression - Comparison with InnoDB Page Compression.

Configuring the InnoDB Page Compression Algorithm

There is not currently a table option to set different InnoDB page compression algorithms for individual tables.

However, the server-wide InnoDB page compression algorithm can be configured by setting the innodb_compression_algorithm system variable.

When this system variable is changed, the InnoDB page compression algorithm does not change for existing pages that were already compressed with a different InnoDB page compression algorithm. InnoDB is able to handle this situation without issues, because every page in an InnoDB tablespace contains metadata about the InnoDB page compression algorithm in the page header. This means that InnoDB supports having uncompressed pages and pages compressed with different InnoDB page compression algorithms in the same InnoDB tablespace at the same time.

This system variable can be set to one of the following values:

System Variable Value	Description
none	Pages are not compressed. This is the default value in MariaDB 10.2.3 and before, and MariaDB 10.1.21 and before.
zlib	Pages are compressed using the bundled zlib compression algorithm. This is the default value in MariaDB 10.2.4 and later, and MariaDB 10.1.22 and later.
lz4	Pages are compressed using the lz4 compression algorithm.
lzo	Pages are compressed using the lzo compression algorithm.
lzma	Pages are compressed using the lzma compression algorithm.
bzip2	Pages are compressed using the bzip2 compression algorithm.
snappy	Pages are compressed using the snappy algorithm.

However, on many distributions, the standard MariaDB builds do not support all InnoDB page compression algorithms by default. From MariaDB 10.7, algorithms can be installed as a plugin.

This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_compression_algorithm='lzma';
```

This system variable can also be set in a server option group in an option file prior to starting up the server:

```
[mariadb]
...
innodb_compression_algorithm=lzma
```

Checking Supported InnoDB Page Compression Algorithms

On many distributions, the standard MariaDB builds do not support all InnoDB page compression algorithms by default. Therefore, if you want to use a specific InnoDB page compression algorithm, then you should check whether your MariaDB build supports it.

The zlib compression algorithm is always supported. From MariaDB 10.7, algorithms can be installed as a plugin.

A MariaDB build's support for other InnoDB page compression algorithms can be checked by querying the following status variables with SHOW GLOBAL STATUS:

Status Variable	Description
Innodb_have_lz4	Whether InnoDB supports the lz4 compression algorithm.
Innodb_have_lzo	Whether InnoDB supports the lzo compression algorithm.
Innodb_have_lzma	Whether InnoDB supports the lzma compression algorithm.
Innodb_have_bzip2	Whether InnoDB supports the bzip2 compression algorithm.
Innodb_have_snappy	Whether InnoDB supports the snappy compression algorithm.

For example:

```
SHOW GLOBAL STATUS WHERE Variable_name IN (
   'Innodb_have_lz4',
   'Innodb_have_lzo',
   'Innodb_have_lzma',
   'Innodb_have_bzip2',
   'Innodb_have_snappy'
);
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| Innodb_have_lz4    | OFF   |
| Innodb_have_lzo    | OFF   |
| Innodb_have_lzma   | ON    |
| Innodb_have_bzip2  | OFF   |
| Innodb_have_snappy | OFF   |
+--------------------+-------+
```

Adding Support for an InnoDB Page Compression Algorithm

On many distributions, the standard MariaDB builds do not support all InnoDB page compression algorithms by default. From MariaDB 10.7, algorithms can be installed as a plugin, but in earlier versions, if you want to use certain InnoDB page compression algorithms, then you may need to do the following:

- Download the package for the desired compression library from the above links.
- Install the package for the desired compression library.
- Compile MariaDB from the source distribution.

The general steps for compiling MariaDB are:

- Download and unpack the source code distribution:

```
wget https://downloads.mariadb.com/MariaDB/mariadb-10.4.8/source/mariadb-10.4.8.tar.gz
tar -xvzf mariadb-10.4.8.tar.gz
cd mariadb-10.4.8/
```

- Configure the build using cmake:

```
cmake .
```

- Check CMakeCache.txt to confirm that it has found the desired compression library on your system.
- Compile the build:

```
make
```

- Either install the build:

```
make install
```

Or make a package to install:

```
make package
```

See Compiling MariaDB From Source for more information.

Enabling InnoDB Page Compression

InnoDB page compression is not enabled by default. However, InnoDB page compression can be enabled for just individual InnoDB tables or it can be enabled for all new InnoDB tables by default.

InnoDB page compression is also only supported if the InnoDB table is in a file per-table tablespace. Therefore, the innodb_file_per_table system variable must be set to `ON` to use InnoDB page compression.

InnoDB page compression is only supported if the InnoDB table uses the `Barracuda` file format.Therefore, in MariaDB 10.1 and before, the innodb_file_format system variable must be set to `Barracuda` to use InnoDB page compression.

InnoDB page compression is also only supported if the InnoDB table's row format is COMPACT or DYNAMIC.

Enabling InnoDB Page Compression by Default

In MariaDB 10.2.3 and later, InnoDB page compression can be enabled for all new InnoDB tables by default by setting the innodb_compression_default system variable to `ON`.

This system variable can be set to one of the following values:

System Variable Value	Description
OFF	New InnoDB tables do not use InnoDB page compression. This is the default value.
ON	New InnoDB tables use InnoDB page compression.

This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_compression_default=ON;
```

This system variable's session value can be changed dynamically with SET SESSION:

```
SET GLOBAL innodb_file_per_table=ON;

SET GLOBAL innodb_file_format='Barracuda';

SET GLOBAL innodb_default_row_format='dynamic';

SET GLOBAL innodb_compression_algorithm='lzma';

SET SESSION  innodb_compression_default=ON;

CREATE TABLE users (
   user_id INT NOT NULL,
   b VARCHAR(200),
   PRIMARY KEY(user_id)
)
   ENGINE=InnoDB;
```

This system variable can also be set in a server option group in an option file prior to starting up the server:

```
[mariadb]
...
innodb_compression_default=ON
```

Enabling InnoDB Page Compression for Individual Tables

InnoDB page compression can be enabled for individual tables by setting the PAGE_COMPRESSED table option to `1`:

```
SET GLOBAL innodb_file_per_table=ON;

SET GLOBAL innodb_file_format='Barracuda';

SET GLOBAL innodb_default_row_format='dynamic';

SET GLOBAL innodb_compression_algorithm='lzma';

CREATE TABLE users (
   user_id INT NOT NULL,
   b VARCHAR(200),
   PRIMARY KEY(user_id)
)
   ENGINE=InnoDB
   PAGE_COMPRESSED=1;
```

Configuring the Compression Level

Some InnoDB page compression algorithms support a compression level option, which configures how the InnoDB page compression algorithm will balance speed and compression.

The compression level's supported values range from `1` to `9`. The range goes from the fastest to the most compact, which means that `1` is the fastest and `9` is the most compact.

Only the following InnoDB page compression algorithms currently support compression levels:

- zlib
- lzma

If an InnoDB page compression algorithm does not support compression levels, then it ignores any provided compression level value.

Configuring the Default Compression Level

The default compression level can be configured by setting the innodb_compression_level system variable.

This system variable's default value is `6`.

This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_compression_level=9;
```

This system variable can also be set in a server option group in an option file prior to starting up the server:

```
[mariadb]
...
innodb_compression_level=9
```

Configuring the Compression Level for Individual Tables

The compression level for individual tables can also be configured by setting the PAGE_COMPRESSION_LEVEL table option for the table:

```
SET GLOBAL innodb_file_per_table=ON;

SET GLOBAL innodb_file_format='Barracuda';

SET GLOBAL innodb_default_row_format='dynamic';

SET GLOBAL innodb_compression_algorithm='lzma';

CREATE TABLE users (
   user_id INT NOT NULL,
   b VARCHAR(200),
   PRIMARY KEY(user_id)
)
   ENGINE=InnoDB
   PAGE_COMPRESSED=1
   PAGE_COMPRESSION_LEVEL=9;
```

Configuring the Failure Threshold and Maximum Padding

InnoDB page compression can encounter compression failures.

InnoDB page compression's failure threshold can be configured. If InnoDB encounters more compression failures than the failure threshold, then it pads pages with zeroed out bytes before attempting to compress them as a way to reduce failures. If the failure rate stays above the failure threshold, then InnoDB pads pages with more zeroed out bytes in 128 byte increments.

InnoDB page compression's maximum padding can also be configured.

Configuring the Failure Threshold

The failure threshold can be configured by setting the innodb_compression_failure_threshold_pct system variable.

This system variable's supported values range from `0` to `100`.

This system variable's default value is `5`.

This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_compression_failure_threshold_pct=10;
```

This system variable can also be set in a server option group in an option file prior to starting up the server:

```
[mariadb]
...
innodb_compression_failure_threshold_pct=10
```

Configuring the Maximum Padding

The maximum padding can be configured by setting the innodb_compression_pad_pct_max system variable.

This system variable's supported values range from `0` to `75`.

This system variable's default value is `50`.

This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_compression_pad_pct_max=75;
```

This system variable can also be set in a server option group in an option file prior to starting up the server:

```
[mariadb]
...
innodb_compression_pad_pct_max=75
```

Saving Storage Space with Sparse Files

When InnoDB page compression is used, InnoDB may still write the compressed page to the tablespace file with the original size of the uncompressed page, which would be equivalent to the value of the innodb_page_size system variable. This is done by design, because when InnoDB's I/O code needs to read the page from disk, it can only read the full page size. However, this is obviously not optimal.

On file systems that support sparse files, this problem is solved by writing the tablespace file as a sparse file using the *punch hole* technique. With the *punch hole* technique, InnoDB will only write the actual compressed page size to the tablespace file, aligned to sector size. The rest of the page is trimmed.

This *punch hole* technique allows InnoDB to read the compressed page from disk as the full page size, even though the compressed page really takes up less space on the file system.

There are some potential disadvantages to using sparse files:

- Some utilities may require special options in order to handle sparse files in an efficient manner.
- Most existing file systems are slow to unlink() sparse files. As a consequence, if a tablespace file is a sparse file, then dropping the table can be very slow.

Sparse File Support on Linux

On Linux, the following file systems support sparse files:

- `ext3`
- `ext4`
- `xfs`
- `btrfs`
- `nvmfs`

On Linux, file systems need to support the fallocate() system call with the `FALLOC_FL_PUNCH_HOLE` and `FALLOC_FL_KEEP_SIZE` flags:

```
fallocate(file_handle, FALLOC_FL_PUNCH_HOLE | FALLOC_FL_KEEP_SIZE, file_offset, remainder_len);
```

Some Linux utilities may require special options in order to work with sparse files efficiently:

- The ls utility will report the non-sparse size of the tablespace file when executed with default behavior, but ls -s will report the actual amount of storage allocated for the tablespace file.
- The cp utility is pretty good at auto-detecting sparse files, but it also provides the cp –sparse=always and cp –sparse=never options, if the auto-detection is not desired.
- The tar utility will archive sparse files with their non-sparse size when executed with default behavior, but tar –sparse will auto-detect sparse files, and archive them with their sparse size.

Sparse File Support on Windows

On Windows, the following file systems support sparse files:

- `NTFS`

On Windows, file systems need to support the DeviceIoControl() function with the FSCTL_SET_SPARSE and FSCTL_SET_ZERO_DATA control codes:

```
DeviceIoControl(file_handle, FSCTL_SET_SPARSE, inbuf, inbuf_size,
   outbuf, outbuf_size,  NULL, &overlapped)
...
DeviceIoControl(file_handle, FSCTL_SET_ZERO_DATA, inbuf, inbuf_size,
   outbuf, outbuf_size,  NULL, &overlapped)
```

Configuring InnoDB to use Sparse Files

In MariaDB 10.3 and later, InnoDB uses the *punch hole* technique to create sparse files used automatically when the underlying file system supports sparse files.

In MariaDB 10.2 and before, InnoDB can be configured to use the *punch hole* technique to create sparse files by configuring the innodb_use_trim and innodb_use_fallocate system variables. These system variables can be set in a server option group in an option file prior to starting up the server:

```
[mariadb]
...
innodb_use_trim=ON
innodb_use_fallocate=ON
```

Optimized for Flash Storage

InnoDB page compression was designed to be optimized on solid state drives (SSDs) and other flash storage.

InnoDB page compression was originally developed by collaborating with Fusion-io. As a consequence, it was originally designed to work best on FusionIO devices using NVMFS. Fusion-io has since been acquired by Western Digital, and they have decided not to continue supporting NVMFS.

However, InnoDB page compression is still likely to be most optimized on solid state drives (SSDs) and other flash storage.

InnoDB page compression works without any issues on hard disk drives (HDDs). However, since its compression relies on the use of sparse files, the data may be somewhat fragmented on disk. This fragmentation may hurt performance on HDDs, since they handle random reads and writes much more slowly than flash storage.

Configuring InnoDB Page Flushing

With InnoDB page compression, pages are compressed when they are flushed to disk. Therefore, it can be helpful to optimize the configuration of InnoDB's page flushing. See InnoDB Page Flushing for more information.

Monitoring InnoDB Page Compression

InnoDB page compression can be monitored by querying the following status variables with SHOW GLOBAL STATUS:

Status Variable	Description
Innodb_page_compression_saved	Bytes saved by compression
Innodb_page_compression_trim_sect512	Number of 512 sectors trimmed
Innodb_page_compression_trim_sect1024	Number of 1024 sectors trimmed
Innodb_page_compression_trim_sect2048	Number of 2048 sectors trimmed
Innodb_page_compression_trim_sect4096	Number of 4096 sectors trimmed
Innodb_page_compression_trim_sect8192	Number of 8192 sectors trimmed
Innodb_page_compression_trim_sect16384	Number of 16384 sectors trimmed
Innodb_page_compression_trim_sect32768	Number of 32768 sectors trimmed
Innodb_num_pages_page_compressed	Number of pages compressed

Status Variable	Description
Innodb_num_page_compressed_trim_op	Number of trim operations
Innodb_num_page_compressed_trim_op_saved	Number of trim operations saved
Innodb_num_pages_page_decompressed	Number of pages decompressed
Innodb_num_pages_page_compression_error	Number of compression errors

With InnoDB page compression, a page is only compressed when it is flushed to disk. This means that if you are monitoring InnoDB page compression via these status variables, then the status variables values will only get incremented when the dirty pages are flushed to disk, which does not necessarily happen immediately:

```
CREATE TABLE `tab` (
     `id` INT(11) NOT NULL,
     `str` VARCHAR(50) DEFAULT NULL,
     PRIMARY KEY (`id`)
   ) ENGINE=InnoDB;

INSERT INTO tab VALUES (1, 'str1');

SHOW GLOBAL STATUS LIKE 'Innodb_num_pages_page_compressed';
+----------------------------------+-------+
| Variable_name                    | Value |
+----------------------------------+-------+
| Innodb_num_pages_page_compressed | 0     |
+----------------------------------+-------+

SET GLOBAL innodb_file_per_table=ON;

SET GLOBAL innodb_file_format='Barracuda';

SET GLOBAL innodb_default_row_format='dynamic';

SET GLOBAL innodb_compression_algorithm='lzma';

ALTER TABLE tab PAGE_COMPRESSED=1;

SHOW GLOBAL STATUS LIKE 'Innodb_num_pages_page_compressed';
+----------------------------------+-------+
| Variable_name                    | Value |
+----------------------------------+-------+
| Innodb_num_pages_page_compressed | 0     |
+----------------------------------+-------+

SELECT SLEEP(10);
+-----------+
| SLEEP(10) |
+-----------+
|         0 |
+-----------+

SHOW GLOBAL STATUS LIKE 'Innodb_num_pages_page_compressed';
+----------------------------------+-------+
| Variable_name                    | Value |
+----------------------------------+-------+
| Innodb_num_pages_page_compressed | 3     |
+----------------------------------+-------+
```

Compatibility with Backup Tools

mariadb-backup supports InnoDB page compression.

Percona XtraBackup does not support InnoDB page compression.

Acknowledgements

- InnoDB page compression was developed by collaborating with Fusion-io. Special thanks especially to Dhananjoy Das and Torben Mathiasen.

See Also

- Storage-Engine Independent Column Compression
- Atomic Write Support
- MariaDB Introduces Atomic Writes
- Small Datum: Third day with InnoDB transparent page compression
- InnoDB holepunch compression vs the filesystem in MariaDB 10.1
- Significant performance boost with new MariaDB page compression on FusionIO
- INFLOW '14: NVM Compression—Hybrid Flash-Aware Application Level Compression

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Page Flushing

InnoDB uses a single `buf_flush_page_cleaner` thread to flush dirty pages—modified pages that have not yet been written to data files—from the InnoDB buffer pool to disk. This thread handles page flushing regardless of the number of buffer pool instances. These dirty pages are flushed using a least-recently used (LRU) algorithm, which manages memory efficiently by prioritizing the eviction of older, less frequently accessed pages.

innodb_max_dirty_pages_pct

The innodb_max_dirty_pages_pct variable specifies the target maximum percentage of unwritten (dirty) pages in the buffer pool. If this percentage is exceeded, flushing will take place.

innodb_max_dirty_pages_pct_lwm

The innodb_max_dirty_pages_pct_lwm variable determines the low-water mark percentage of dirty pages that will enable preflushing to lower the dirty page ratio. The default value is `0`.

> **Tip:**
>
> To make flushing more eager and ensure more consistent background I/O, you can set `innodb_max_dirty_pages_pct_lwm` to a very low value, such as `0.001`:
>
> ```
> SET GLOBAL innodb_max_dirty_pages_pct_lwm=0.001;
> ```

Configuring the InnoDB I/O Capacity

increasing the amount of I/O capacity available to InnoDB can help increase the performance of page flushing. The unit of innodb_io_capacity is the number of data pages (of the size defined by `innodb_page_size`) that can be written per second.

Scope of Throttling

It is critical to understand the restricted scope of this variable in modern versions of MariaDB:

- Checkpoint Flushing Only: innodb_io_capacity only throttles checkpoint flushing (background or idle flushing). It does not throttle LRU eviction flushing, which handles the removal of pages when the buffer pool is at capacity.
- Interaction with `innodb_flush_sync`: The `innodb_io_capacity` limit is only effective when innodb_flush_sync is set to `OFF`. When `innodb_flush_sync=ON` (the default), InnoDB may ignore this limit during aggressive "furious flushing" if a log checkpoint is urgently required to prevent the redo log from filling up.

- No Throttling for Buffer Pool Loading: As of MariaDB 10.5.19, 10.6.12, 10.11.2, and later, this parameter no longer throttles the loading of buffer pool dumps at startup (MDEV-25417). Startup loads are now performed at best-effort speed.
- Interaction with innodb_flush_sync: The `innodb_io_capacity` limit is only effective when innodb_flush_sync is set to `OFF`. When `innodb_flush_sync=ON` (the default), InnoDB may ignore this limit during aggressive "furious flushing" if a log checkpoint is urgently required to prevent the redo log from filling up.
- Shared Storage Consideration: If the InnoDB redo log resides on the same physical storage as the data files, ensure you leave some spare capacity for log writes so they are not blocked by background page flushing.

Adjusting I/O Capacity

The amount of I/O capacity available to InnoDB can be configured by setting the innodb_io_capacity system variable. This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_io_capacity=20000;
```

The maximum amount of I/O capacity available to InnoDB in an emergency defaults to either `2000` or twice `innodb_io_capacity`, whichever is higher, or can be directly configured by setting the innodb_io_capacity_max system variable.

Device-Specific Recommendations

When setting these variables, consider the physical limits of your storage hardware:

Storage Device Type	Typical IOPS Capability	Recommended innodb_io_capacity
SATA HDD	~100 – 200	100 – 200
SATA SSD	~50,000 – 100,000	2,000 – 20,000
NVMe SSD	500,000+	20,000 – 80,000+

For high-speed NVMe storage, a sensible value for `innodb_io_capacity` may be as high as 80,000.

Page Flushing Before MariaDB Server 10.5

InnoDB supported multiple partitions and threads to reduce internal mutex contention. These features were rendered unnecessary by architectural improvements—such as splitting the buffer pool mutex and implementing read-write locks for the page hash—and were removed to reduce context-switching overhead:

- innodb_page_cleaners: Previously allowed the use of multiple cleaner threads; it was deprecated and ignored in MariaDB 10.5.1 and removed entirely in MariaDB 10.6.
- innodb_buffer_pool_instances: Partitioned the buffer pool into multiple instances; this was removed in MariaDB 10.5 as the buffer pool now runs as a single instance.
- innodb_mtflush_threads: A Fusion-io specific parameter for multi-threaded flushing, removed in MariaDB 10.3.2.

See Also

- Significant performance boost with new MariaDB page compression on FusionIO
- InnoDB Redo Log
- InnoDB Buffer Pool
- InnoDB Background Thread Pool

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Purge

When a transaction updates a row in an InnoDB table, InnoDB's MVCC implementation keeps old versions of the row in the InnoDB undo log. The old versions are kept at least until all transactions older than the transaction that updated the row are no longer open. At that point, the old versions can be deleted. InnoDB has a purge process that is used to delete these old versions.

InnoDB Purge Threads

In MariaDB, the InnoDB storage engine uses Purge Threads to perform garbage collection in the background. The Purge Threads are related to multi-version concurrency control (MVCC).

The Purge Threads perform garbage collection of various items:

- The Purge Threads perform garbage collection of the InnoDB Undo Log. When a row is updated in the clustered index, InnoDB updates the values in the clustered index, and the old row version is added to the Undo Log. The Purge Threads scan the Undo Log for row versions that are not needed by open transactions and permanently delete them.
- The Purge Threads perform garbage collection of index records. When an indexed column is updated, InnoDB creates a new index record for the updated value in each affected index, and the old index records are delete-marked. When the primary key column is updated, InnoDB creates a new index record for the updated value in every index, and each old index record is delete-marked. The Purge Threads scan for delete-marked index records and permanently delete them.
- The Purge Threads perform garbage collection of freed overflow pages. BLOB, CHAR, TEXT, VARCHAR, VARBINARY, and related types are sometimes stored on overflow pages. When the value on the overflow page is deleted or updated, the overflow page is no longer needed. The Purge Threads delete these freed overflow pages.

Feature Summary

Feature	Detail	Resources
Thread	InnoDB Purge Threads	
Storage Engine	InnoDB	
Purpose		
Quantity	Set by innodb_purge_threads	Configure the InnoDB Purge Threads

Configuring the Purge Threads

The number of purge threads can be set by configuring the innodb_purge_threads system variable. This system variable can be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
innodb_purge_threads=8
```

```
SET GLOBAL innodb_purge_threads=8;

SHOW GLOBAL VARIABLES
   LIKE 'innodb_purge_threads';
```

```
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| innodb_purge_threads | 8     |
+----------------------+-------+
```

Optimizing Purge Performance

Configuring the Purge Batch Size

The purge batch size is defined as the number of InnoDB undo log records that must be written before triggering purge. The purge batch size can be set by configuring the innodb_purge_batch_size system variable. This system variable can be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
innodb_purge_batch_size = 50
```

Configuring the Max Purge Lag

If purge operations are lagging on a busy server, then this can be a tough situation to recover from. As a solution, InnoDB allows you to set the max purge lag. The max purge lag is defined as the maximum number of InnoDB undo logs that can be waiting to be purged from the history until InnoDB begins delaying DML statements.

The max purge lag can be set by configuring the innodb_max_purge_lag system variable. This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_max_purge_lag=1000;
```

This system variable can also be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
innodb_max_purge_lag = 1000
```

The maximum delay can be set by configuring the innodb_max_purge_lag_delay system variable. This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_max_purge_lag_delay=100;
```

This system variable can also be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
innodb_max_purge_lag_delay = 100
```

Configuring the Purge Rollback Segment Truncation Frequency

The purge rollback segment truncation frequency is defined as the number of purge loops that are run before unnecessary rollback segments are truncated. The purge rollback segment truncation frequency can be set by configuring the innodb_purge_rseg_truncate_frequency system variable. This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_purge_rseg_truncate_frequency=64;
```

This system variable can also be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
innodb_purge_rseg_truncate_frequency = 64
```

Configuring the Purge Undo Log Truncation

Purge undo log truncation occurs when InnoDB truncates an entire InnoDB undo log tablespace, rather than deleting individual InnoDB undo log records.

Purge undo log truncation can be enabled by configuring the innodb_undo_log_truncate system variable. This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_undo_log_truncate=ON;
```

This system variable can also be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
innodb_undo_log_truncate = ON
```

An InnoDB undo log tablespace is truncated when it exceeds the maximum size that is configured for InnoDB undo log tablespaces. The maximum size can be set by configuring the innodb_max_undo_log_size system variable. This system variable can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_max_undo_log_size='64M';
```

This system variable can also be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
innodb_max_undo_log_size = 64M
```

Purge's Effect on Row Metadata

An InnoDB table's clustered index has three hidden system columns that are automatically generated. These hidden system columns are:

- `DB_ROW_ID` - If the table has no other `PRIMARY KEY` or no other `UNIQUE KEY` defined as `NOT NULL` that can be promoted to the table's `PRIMARY KEY`, then InnoDB will use a hidden system column called `DB_ROW_ID`. InnoDB will automatically generate the value for the column from a global InnoDB-wide 48-bit sequence (instead of being table-local).
- `DB_TRX_ID` - The transaction ID of either the transaction that last changed the row or the transaction that currently has the row locked.
- `DB_ROLL_PTR` - A pointer to the InnoDB undo log that contains the row's previous record. The value of `DB_ROLL_PTR` is only valid if `DB_TRX_ID` belongs to the current read view. The oldest valid read view is the purge view.

If a row's last InnoDB undo log record is purged, this can obviously affect the value of the row's `DB_ROLL_PTR` column, because there would no longer be any InnoDB undo log record for the pointer to reference.

The purge process will set a row's `DB_TRX_ID` column to `0` after all of the row's associated InnoDB undo log records have been deleted. This change allows InnoDB to perform an optimization: if a query wants to read a row,

and if the row's `DB_TRX_ID` column is set to `0`, then it knows that no other transaction has the row locked. Usually, InnoDB needs to lock the transaction system's mutex in order to safely check whether a row is locked, but this optimization allows InnoDB to confirm that the row can be safely read without any heavy internal locking.

This optimization can speed up reads, but it comes at a noticeable cost at other times. For example, it can cause the purge process to use more I/O after inserting a lot of rows, since the value of each row's `DB_TRX_ID` column will have to be reset.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Undo Log

Overview

When a transaction writes data, it always inserts them in the table indexes or data (in the buffer pool or in physical files). No private copies are created. The old versions of data being modified by active InnoDB transactions are stored in the undo log. The original data can then be restored, or viewed by a consistent read.

Implementation Details

Before a row is modified, a diff is copied into the undo log. Each normal row contains a pointer to the most recent version of the same row in the undo log. Each row in the undo log contains a pointer to previous version, if any. So, each modified row has a history chain.

Rows are never physically deleted until a transaction ends. If they were deleted, the restore in `ROLLBACK` would be impossible. Thus, rows are simply marked for deletion.

Each transaction uses a *view* of the records. The transaction isolation level determines how this view is created. For example, `READ UNCOMMITTED` usually uses the current version of rows, even if they are not committed (*dirty reads*). Other isolation levels require that the most recent committed version of rows is searched in the undo log. `READ COMMITTED` uses a different view for each table, while `REPEATABLE READ` and `SERIALIZABLE` use the same view for all tables.

There is also a global history list of the data. When a transaction is committed, its history is added to this history list. The order of the list is the chronological order of the commits.

The purge thread deletes the rows in the undo log which are not needed by any existing view. The rows for which a most recent version exists are deleted, as well as the delete-marked rows.

If InnoDB needs to restore an old version, it will simply replace the newer version with the older one. When a transaction inserts a new row, there is no older version. However, in that case, the restore can be done by deleting the inserted rows.

Effects of Long-Running Transactions

Understanding how the undo log works helps with understanding the negative effects long transactions.

- Long transactions generate several old versions of the rows in the undo log. Those rows will probably be needed for a longer time, because other long transactions will need them. Since those transactions will generate more modified rows, a sort of combinatorial explosion can be observed. Thus, the undo log requires more space.
- Transaction may need to read very old versions of the rows in the history list, thus their performance will degrade.

Of course read-only transactions do not write more entries in the undo log; however, they delay the purging of existing entries.

Also, long transactions can more likely result in deadlocks, but this problem is not related to the undo log.

Feature Summary

Feature	Detail	Resources
Transaction Log	InnoDB Undo Log	
Storage Engine	InnoDB	
Purpose	Multi-Version Concurrency Control (MVCC)	
Availability	All ES and CS versions	MariaDB Enterprise Server
Location	By default, located in InnoDB system tablespace When innodb_undo_tablespaces	`innodb_undo_tablespaces` is set, located in directory set by innodb_undo_directory (Defaults to datadir)
Quantity	Set by innodb_undo_tablespaces	Configure the InnoDB Undo Log
Size	10 MB per tablespace by default (grows as needed)	

Configuration

System variables affecting undo logs include:

- innodb_max_undo_log_size
- innodb_undo_directory
- innodb_undo_log_truncate
- innodb_undo_logs
- innodb_undo_tablespaces
- innodb_purge_batch_size
- innodb_purge_rseg_truncate_frequency

The undo log is not a log file that can be viewed on disk in the usual sense, such as the error log or slow query log, but rather an area of storage.

Before MariaDB 11.0, the undo log is usually part of the physical system tablespace, but from MariaDB 10.0, the innodb_undo_directory and innodb_undo_tablespaces system variables can be used to split into different tablespaces and store in a different location (perhaps on a different storage device). From MariaDB 11.0, multiple undo tablespaces are enabled by default, and the innodb_undo_tablespaces default is changed to 3 so that the space occupied by possible bursts of undo log records can be reclaimed after innodb_undo_log_truncate is set.

Each insert or update portion of the undo log is known as a rollback segment. The innodb_undo_logs system variable allowed to reduce the number of rollback segments from the usual 128, to limit the number of concurrently active write transactions. innodb_undo_logs was deprecated and ignored in MariaDB 10.5 and removed in MariaDB 10.6, as it always makes sense to use the maximum number of rollback segments.

The related innodb_available_undo_logs status variable stores the total number of available InnoDB undo logs.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Redo Log

Directly editing or moving the redo logs can cause corruption, and should never normally be attempted.

Overview

The InnoDB storage engine in MariaDB Enterprise Server employs a **Redo Log** to ensure data is written to disk reliably, even in the event of a crash. This is achieved by the Redo Log acting as a transaction log.

Redo Log records are uniquely identified by a **Log Sequence Number (LSN)**. The Redo Log consists of circular log files of a fixed size, where older records are routinely overwritten by newer ones.

InnoDB periodically performs **checkpoints**. During a checkpoint operation, InnoDB writes (flushes) the Redo Log records to the InnoDB tablespace files.

In the event of a server crash, InnoDB utilizes the Redo Log for **crash recovery** during the subsequent server startup. It locates the last checkpoint within the Redo Log and then replays the Redo Log records generated since that checkpoint, effectively flushing these pending changes to the InnoDB tablespace files.

The redo log files are typically named `ib_logfileN`, where `N` is an integer. However, starting with MariaDB 10.5, there is a single redo log file, consistently named `ib_logfile0`. The location of these redo log files is determined by the `innodb_log_group_home_dir` system variable, if configured. If this variable is not set, the redo log files are created in the directory specified by the `datadir` system variable. The redo log plays a crucial role during crash recovery and in the background process of flushing transactions to the tablespaces.

Feature Summary

Feature	Detail	Resources
Transaction Log	InnoDB Redo Log	
Storage Engine	InnoDB	
Purpose	Crash Safety	
Availability	All ES and CS versions	MariaDB Enterprise Server
Location	Set by innodb_log_group_home_dir (Defaults to datadir)	
Quantity	1 (ES 10.5+, CS 10.5+)	Configure the InnoDB Redo Log
Size	Set by innodb_log_file_size (default varies)	Configure the InnoDB Redo Log

Basic Configuration

```
[mariadb]
...
innodb_log_file_size=2G
```

```
SET GLOBAL innodb_log_file_size=(2 * 1024 * 1024 * 1024);

SHOW GLOBAL VARIABLES
   LIKE 'innodb_log_file_size';
```

```
+----------------------+------------+
| Variable_name        | Value      |
+----------------------+------------+
| innodb_log_file_size | 2147483648 |
+----------------------+------------+
```

Flushing Effects on Performance and Consistency

The innodb_flush_log_at_trx_commit system variable determines how often the transactions are flushed to the redo log, and it is important to achieve a good balance between speed and reliability.

Binary Log Group Commit and Redo Log Flushing

When both innodb_flush_log_at_trx_commit=1 (the default) is set and the binary log is enabled, there is one less sync to disk inside InnoDB during commit (2 syncs shared between a group of transactions instead of 3). See Binary Log Group Commit and InnoDB Flushing Performance for more information.

Redo Log Group Capacity

The redo log group capacity is the total combined size of all InnoDB redo logs. The relevant factors are:

- From MariaDB 10.5, there is 1 redo log.
- The size of each redo log file is configured by the innodb_log_file_size system variable. This can safely be set to a much higher value from MariaDB 10.5. Before MariaDB 10.9, resizing required the server to be restarted. From MariaDB 10.9 the variable can be set dynamically.

The redo log group capacity is determined by the following calculation:

`innodb_log_group_capacity` = innodb_log_file_size * innodb_log_files_in_group

For example, if innodb_log_file_size is set to `2G` and innodb_log_files_in_group is set to `2`, then we would have the following:

- `innodb_log_group_capacity` = innodb_log_file_size * innodb_log_files_in_group
- = `2G` * `2`
- = `4G`

Changing the Redo Log Group Capacity

MariaDB starting with 10.5

The number of redo log files is fixed.

The size of redo log files can be changed with the following process:

- Stop the server.
- To change the log file size, configure innodb_log_file_size.
- Start the server.

Log Sequence Number (LSN)

Records within the InnoDB redo log are identified via a log sequence number (LSN).

Checkpoints

When InnoDB performs a checkpoint, it writes the LSN of the oldest dirty page in the InnoDB buffer pool to the InnoDB redo log. If a page is the oldest dirty page in the InnoDB buffer pool, then that means that all pages with lower LSNs have been flushed to the physical InnoDB tablespace files. If the server were to crash, then InnoDB would perform crash recovery by only applying log records with LSNs that are greater than or equal to the LSN of the oldest dirty page written in the last checkpoint.

Checkpoints are one of the tasks performed by the InnoDB master background thread. This thread schedules checkpoints 7 seconds apart when the server is very active, but checkpoints can happen more frequently when the server is less active.

Dirty pages are not actually flushed from the buffer pool to the physical InnoDB tablespace files during a checkpoint. That process happens asynchronously on a continuous basis by InnoDB's write I/O background threads configured by the innodb_write_io_threads system variable. If you want to make this process more aggressive, then you can decrease the value of the innodb_max_dirty_pages_pct system variable. You may also need to better tune InnoDB's I/O capacity on your system by setting the innodb_io_capacity system variable.

Determining the Checkpoint Age

The checkpoint age is the amount of data written to the InnoDB redo log since the last checkpoint.

Determining the Checkpoint Age in InnoDB

MariaDB starting with 10.5

MariaDB 10.5 reintroduced the Innodb_checkpoint_age status variable for determining the checkpoint age.

The checkpoint age can also be determined by the process shown below.

To determine the InnoDB checkpoint age, do the following:

- Query SHOW ENGINE INNODB STATUS.
- Find the `LOG` section:

```
---
LOG
---
Log sequence number 252794398789379
Log flushed up to 252794398789379
Pages flushed up to 252792767756840
Last checkpoint at 252792767756840
0 pending log flushes, 0 pending chkp writes
23930412 log i/o's done, 2.03 log i/o's/second
```

- Perform the following calculation:

`innodb_checkpoint_age` = `Log sequence number` - `Last checkpoint at`

In the example above, that would be:

- `innodb_checkpoint_age` = `Log sequence number` - `Last checkpoint at`
- = 252794398789379 - 252792767756840
- = 1631032539 bytes
- = 1631032539 byes / (1024 * 1024 * 1024) (GB/bytes)
- = 1.5 GB of redo log written since last checkpoint

Determining the Redo Log Occupancy

The redo log occupancy is the percentage of the InnoDB redo log capacity that is taken up by dirty pages that have not yet been flushed to the physical InnoDB tablespace files in a checkpoint. Therefore, it's determined by the following calculation:

`innodb_log_occupancy` = innodb_checkpoint_age / innodb_log_group_capacity

For example, if innodb_checkpoint_age is `1.5G` and innodb_log_group_capacity is `4G`, then we would have the following:

- `innodb_log_occupancy` = innodb_checkpoint_age / innodb_log_group_capacity
- = `1.5G` / `4G`
- = `0.375`

If the calculated value for redo log occupancy is too close to `1.0`, then the InnoDB redo log capacity may be too small for the current workload.

MariaDB 10.8 Updates

A number of redo log improvements were made in MariaDB 10.8:

- Autosize innodb_buffer_pool_chunk_size (MDEV-25342).
- Improve the redo log for concurrency (MDEV-14425).
- Remove FIL_PAGE_FILE_FLUSH_LSN (MDEV-27199).

Before MariaDB 10.8.1, mariadb-backup –prepare created a zero-length ib_logfile0 as a dummy placeholder. From MariaDB 10.8.1 (MDEV-14425), the size of that dummy file was increased to 12304 (0x3010) bytes, and all updates of FIL_PAGE_FILE_FLUSH_LSN in the first page of the system tablespace are removed.

From MariaDB 10.8.1, if the server is started up with a zero-sized ib_logfile0, it is assumed that an upgrade is being performed after a backup had been prepared. The start LSN will then be read from FIL_PAGE_FILE_FLUSH_LSN, and a new log file are created starting from exactly that LSN.

Manually creating a zero-sized ib_logfile0 without manually updating the FIL_PAGE_FILE_FLUSH_LSN in the system tablespace to a recent enough LSN may result in error messages such as “page LSN is in the future”. If a log was discarded while some changes had already been written to data pages, all sort of corruption may occur.

If the database was initialized with a server that never updates the FIL_PAGE_FILE_FLUSH_LSN field, then any server startup attempts with a zero-size ib_logfile0 are refused because of an invalid LSN. If that field was ever updated with a valid LSN by an older server, this safety mechanism cannot work, and the server may “rewind” to an earlier LSN.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Strict Mode

InnoDB strict mode is similar to SQL strict mode. When it is enabled, certain InnoDB warnings become errors instead.

Configuring InnoDB Strict Mode

InnoDB strict mode is enabled by default.

InnoDB strict mode can be enabled or disabled by configuring the innodb_strict_mode server system variable.

Its global value can be changed dynamically with SET GLOBAL:

```
SET GLOBAL innodb_strict_mode=ON;
```

Its value for the current session can also be changed dynamically with SET SESSION:

```
SET SESSION innodb_strict_mode=ON;
```

It can also be set in a server option group in an option file prior to starting up the server:

```
[mariadb]
...
innodb_strict_mode=ON
```

InnoDB Strict Mode Errors

Wrong Create Options

If InnoDB strict mode is enabled, and if a DDL statement is executed and invalid or conflicting table options are specified, then an error is raised. The error will only be a generic error that says the following:

```
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
```

However, more details about the error can be found by executing SHOW WARNINGS.

For example, the error is raised in the following cases:

- The KEY_BLOCK_SIZE table option is set to a non-zero value, but the ROW_FORMAT table option is set to some row format other than the COMPRESSED row format:

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
KEY_BLOCK_SIZE=4
ROW_FORMAT=DYNAMIC;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: cannot specify ROW_FORMAT = DYNAMIC with KEY_BLOCK_SIZE.      |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB       |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The KEY_BLOCK_SIZE table option is set to a non-zero value, but the configured value is larger than either `16` or the value of the innodb_page_size system variable, whichever is smaller.

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
KEY_BLOCK_SIZE=16;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: KEY_BLOCK_SIZE=16 cannot be larger than 8.                    |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB       |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The KEY_BLOCK_SIZE table option is set to a non-zero value, but the innodb_file_per_table system variable is not set to `ON`.

```
SET GLOBAL innodb_file_per_table=OFF;
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
KEY_BLOCK_SIZE=4;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: KEY_BLOCK_SIZE requires innodb_file_per_table.                |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB       |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The KEY_BLOCK_SIZE table option is set to a non-zero value, but it is not set to one of the supported values: [1, 2, 4, 8, 16].

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
KEY_BLOCK_SIZE=5;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: invalid KEY_BLOCK_SIZE = 5. Valid values are [1, 2, 4, 8, 16] |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options")    |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB       |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The ROW_FORMAT table option is set to the COMPRESSED row format, but the innodb_file_per_table system variable is not set to `ON`.

```
SET GLOBAL innodb_file_per_table=OFF;
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
ROW_FORMAT=COMPRESSED;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: ROW_FORMAT=COMPRESSED requires innodb_file_per_table.         |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB       |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The ROW_FORMAT table option is set to a value, but it is not set to one of the values supported by InnoDB: REDUNDANT, COMPACT, DYNAMIC, and COMPRESSED.

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
ROW_FORMAT=PAGE;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: invalid ROW_FORMAT specifier.                                 |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB    |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- Either the KEY_BLOCK_SIZE table option is set to a non-zero value or the ROW_FORMAT table option is set to the COMPRESSED row format, but the innodb_page_size system variable is set to a value greater than `16k`.

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
ROW_FORMAT=COMPRESSED;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: Cannot create a COMPRESSED table when innodb_page_size > 16k. |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options")    |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB       |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

- The DATA DIRECTORY table option is set, but the innodb_file_per_table system variable is not set to `ON`.

```
SET GLOBAL innodb_file_per_table=OFF;
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
DATA DIRECTORY='/mariadb';
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: DATA DIRECTORY requires innodb_file_per_table.                |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB    |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The DATA DIRECTORY table option is set, but the table is a temporary table.

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TEMPORARY TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
DATA DIRECTORY='/mariadb';
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: DATA DIRECTORY cannot be used for TEMPORARY tables.           |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB       |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The INDEX DIRECTORY table option is set.

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
INDEX DIRECTORY='/mariadb';
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning | 1478 | InnoDB: INDEX DIRECTORY is not supported                              |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB       |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The PAGE_COMPRESSED table option is set to `1`, so InnoDB page compression is enabled, but the ROW_FORMAT table option is set to some row format other than the COMPACT or DYNAMIC row formats.

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
PAGE_COMPRESSED=1
ROW_FORMAT=COMPRESSED;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning |  140 | InnoDB: PAGE_COMPRESSED table can't have ROW_TYPE=COMPRESSED          |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB       |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The PAGE_COMPRESSED table option is set to `1`, so InnoDB page compression is enabled, but the innodb_file_per_table system variable is not set to `ON`.

```
SET GLOBAL innodb_file_per_table=OFF;
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
PAGE_COMPRESSED=1;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning |  140 | InnoDB: PAGE_COMPRESSED requires innodb_file_per_table.               |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB    |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The PAGE_COMPRESSED table option is set to `1`, so InnoDB page compression is enabled, but the KEY_BLOCK_SIZE table option is also specified.

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
PAGE_COMPRESSED=1
KEY_BLOCK_SIZE=4;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning |  140 | InnoDB: PAGE_COMPRESSED table can't have key_block_size               |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB    |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

- The PAGE_COMPRESSION_LEVEL table option is set, but the PAGE_COMPRESSED table option is set to `0`, so InnoDB page compression is disabled.

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   id INT PRIMARY KEY,
   str VARCHAR(50)
)
PAGE_COMPRESSED=0
PAGE_COMPRESSION_LEVEL=9;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning |  140 | InnoDB: PAGE_COMPRESSION_LEVEL requires PAGE_COMPRESSED               |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB    |
+---------+------+-----------------------------------------------------------------------+
3 rows in set (0.000 sec)
```

COMPRESSED Row Format

If InnoDB strict mode is enabled, and if a table uses the COMPRESSED row format, and if the table's KEY_BLOCK_SIZE is too small to contain a row, then an error is returned by the statement.

Row Size Too Large

If InnoDB strict mode is enabled, and if a table exceeds its row format's maximum row size, then InnoDB will return an error.

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT OR BLOB may help. IN current row format, BLOB prefix of 0 bytes IS stored inline.
```

See Troubleshooting Row Size Too Large Errors with InnoDB for more information.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB System Variables

This page documents system variables related to the InnoDB storage engine. For options that are not system variables, see InnoDB Options.

See Server System Variables for a complete list of system variables and instructions on setting them.

Also see the Full list of MariaDB options, system and status variables.

`have_innodb`

- Description: If the server supports InnoDB tables, are set to `YES`, otherwise are set to `NO`. Removed in MariaDB 10.0, use the Information Schema PLUGINS table or SHOW ENGINES instead.
- Scope: Global
- Dynamic: No
- Removed: MariaDB 10.0

`ignore_builtin_innodb`

- Description: Setting this to `1` results in the built-in InnoDB storage engine being ignored. In some versions of MariaDB, XtraDB is the default and is always present, so this variable is ignored and setting it results in a warning. From MariaDB 10.0.1 to MariaDB 10.0.8, when InnoDB was the default instead of XtraDB, this variable needed to be set. Usually used in conjunction with the plugin-load=innodb=ha_innodb option to use the InnoDB plugin.
- Command line: `--ignore-builtin-innodb`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_adaptive_checkpoint`

- Description: Replaced with innodb_adaptive_flushing_method. Controls adaptive checkpointing. InnoDB's fuzzy checkpointing can cause stalls, as many dirty blocks are flushed at once as the checkpoint age nears the maximum. Adaptive checkpointing aims for more consistent flushing, approximately `modified age / maximum checkpoint age`. Can result in larger transaction log files
  - `reflex` Similar to innodb_max_dirty_pages_pct flushing but flushes blocks constantly and contiguously based on the oldest modified age. If the age exceeds 1/2 of the maximum age capacity, flushing are weak

contiguous. If the age exceeds 3/4, flushing are strong. Strength can be adjusted by the variable innodb_io_capacity.
    - `estimate` The default, and independent of innodb_io_capacity. If the oldest modified age exceeds 1/2 of the maximum age capacity, blocks are flushed every second at a rate determined by the number of modified blocks, LSN progress speed and the average age of all modified blocks.
    - `keep_average` Attempts to keep the I/O rate constant by using a shorter loop cycle of one tenth of a second. Designed for SSD cards.

- Command line: `--innodb-adaptive-checkpoint=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `estimate`
- Valid Values: `none` or `0`, `reflex` or `1`, `estimate` or `2`, `keep_average` or `3`
- Removed: XtraDB 5.5 - replaced with innodb_adaptive_flushing_method

`innodb_adaptive_flushing`

- Description: If set to `1`, the default, the server will dynamically adjust the flush rate of dirty pages in the InnoDB buffer pool. This assists to reduce brief bursts of I/O activity. If set to `0`, adaptive flushing will only take place when the limit specified by innodb_adaptive_flushing_lwm is reached.
- Command line: `--innodb-adaptive-flushing={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`innodb_adaptive_flushing_lwm`

- Description: Adaptive flushing is enabled when this low water mark percentage of the InnoDB redo log capacity is reached. Takes effect even if innodb_adaptive_flushing is disabled.
- Command line: `--innodb-adaptive-flushing-lwm=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `double`
- Default Value: `10.000000`
- Range: `0` to `70`

`innodb_adaptive_flushing_method`

- Description: Determines the method of flushing dirty blocks from the InnoDB buffer pool. If set to `native` or `0`, the original InnoDB method is used. The maximum checkpoint age is determined by the total length of all transaction log files. When the checkpoint age reaches the maximum checkpoint age, blocks are flushed. This can cause lag if there are many updates per second and many blocks with an almost identical age need to be flushed. If set to `estimate` or `1`, the default, the oldest modified age are compared with the maximum age capacity. If it's more than 1/4 of this age, blocks are flushed every second. The number of blocks flushed is determined by the number of modified blocks, the LSN progress speed and the average age of all modified blocks. It's therefore independent of the innodb_io_capacity for the 1-second loop, but not entirely so for the 10-second loop. If set to `keep_average` or `2`, designed specifically for SSD cards, a shorter loop cycle is used in

an attempt to keep the I/O rate constant. Removed in MariaDB 10.0/XtraDB 5.6 and replaced with InnoDB flushing method from MySQL 5.6.

- Command line: `innodb-adaptive-flushing-method=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `estimate`
- Valid Values: `native` or `0`, `estimate` or `1`, `keep_average` or `2`
- Removed: MariaDB 10.0 - replaced with InnoDB flushing method from MySQL 5.6

`innodb_adaptive_hash_index`

- Description: If set to `1`, the default until MariaDB 10.5, the InnoDB hash index is enabled. Based on performance testing (MDEV-17492), the InnoDB adaptive hash index helps performance in mostly read-only workloads, and could slow down performance in other environments, especially DROP TABLE, TRUNCATE TABLE, ALTER TABLE, or DROP INDEX operations. From MariaDB 13.1.1, this variable is an enumeration. The `IF_SPECIFIED` value enables the adaptive hash index only for tables and indexes whose ADAPTIVE_HASH_INDEX option is set to `YES`.
- Command line: `--innodb-adaptive-hash-index[={OFF|ON|IF_SPECIFIED}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration` (`boolean` before MariaDB 13.1.1)
- Valid Values: `OFF`, `ON`, `IF_SPECIFIED` (from MariaDB 13.1.1)
- Default Value: `OFF` (>= MariaDB 10.5), `ON` (<= MariaDB 10.4)

`innodb_adaptive_hash_index_cells`

- Description: Further improves the performance of the InnoDB adaptive hash index (AHI) when `innodb_adaptive_hash_index` is enabled. This variable allows for manual configuration of the hash table size to resolve contention on the `btr_sea::partition::latch` during high-concurrency workloads. The specified value is effectively multiplied by the number of partitions defined in `innodb_adaptive_hash_index_parts`, as each partition maintains its own hash table. Increasing this value can reduce the length of hash bucket chains, which helps avoid performance “hiccups” and “avalanche effects” caused by multiple threads spinning for AHI lookups.
- Command line: `--innodb-adaptive-hash-index-cells=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `134217728`
- Introduced: MariaDB 11.8.4, MariaDB 12.1.2, MariaDB 12.2.1

`innodb_adaptive_hash_index_partitions`

- Description: Specifies the number of partitions for use in adaptive searching. If set to `1`, no extra partitions are created. XtraDB-only. From MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB), this is an alias for innodb_adaptive_hash_index_parts to allow for easier upgrades.
- Command line: `innodb-adaptive-hash-index-partitions=#`
- Scope: Global
- Dynamic: No

- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `64`

`innodb_adaptive_hash_index_parts`

- Description: Specifies the number of partitions for use in adaptive searching. If set to `1`, no extra partitions are created.
- Command line: `innodb-adaptive-hash-index-parts=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `8`
- Range: `1` to `512`

`innodb_adaptive_max_sleep_delay`

- Description: Maximum time in microseconds to automatically adjust the innodb_thread_sleep_delay value to, based on the workload. Useful in extremely busy systems with hundreds of thousands of simultaneous connections. `0` disables any limit. Deprecated and ignored from MariaDB 10.5.5.
- Command line: `--innodb-adaptive-max-sleep-delay=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `0` (>= MariaDB 10.5.5)
  - `150000` (<= MariaDB 10.5.4)
- Range: `0` to `1000000`
- Introduced: MariaDB 10.0
- Deprecated: MariaDB 10.5.5
- Removed: MariaDB 10.6.0

`innodb_additional_mem_pool_size`

- Description: Size in bytes of the InnoDB memory pool used for storing information about internal data structures. Defaults to 8MB, if your application has many tables and a large structure, and this is exceeded, operating system memory are allocated and warning messages written to the error log, in which case you should increase this value. Deprecated in MariaDB 10.0 and removed in MariaDB 10.2 along with InnoDB's internal memory allocator.
- Command line: `--innodb-additional-mem-pool-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `8388608`
- Range: `2097152` to `4294967295`
- Deprecated: MariaDB 10.0
- Removed: MariaDB 10.2.2

`innodb_alter_copy_bulk`

- Description: Allow bulk insert operation for copy alter operation.
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.11.9, MariaDB 11.1.6, MariaDB 11.2.5, MariaDB 11.4.3, MariaDB 11.5.2, MariaDB 11.6.1

`innodb_api_bk_commit_interval`

- Description: Time in seconds between auto-commits for idle connections using the InnoDB memcached interface (not implemented in MariaDB).
- Command line: `--innodb-api-bk-commit-interval=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `5`
- Range: `1` to `1073741824`
- Introduced: MariaDB 10.0
- Removed: MariaDB 10.2.4

`innodb_api_disable_rowlock`

- Description: For use with MySQL's memcached (not implemented in MariaDB)
- Command line: `--innodb-api-disable-rowlock={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.0
- Removed: MariaDB 10.2.4

`innodb_api_enable_binlog`

- Description: For use with MySQL's memcached (not implemented in MariaDB)
- Command line: `--innodb-api-enable-binlog={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.0
- Removed: MariaDB 10.2.4

`innodb_api_enable_mdl`

- Description: For use with MySQL's memcached (not implemented in MariaDB)
- Command line: `--innodb-api-enable-mdl={0|1}`
- Scope: Global
- Dynamic: No

- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.0
- Removed: MariaDB 10.2.4

`innodb_api_trx_level`

- Description: For use with MySQL's memcached (not implemented in MariaDB)
- Command line: `--innodb-api-trx-level=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Introduced: MariaDB 10.0
- Removed: MariaDB 10.2.4

`innodb_auto_lru_dump`

- Description: Renamed innodb_buffer_pool_restore_at_startup since XtraDB 5.5.10-20.1, which was in turn replaced by innodb_buffer_pool_load_at_startup in MariaDB 10.0.
- Command line: `--innodb-auto-lru-dump=#`
- Removed: XtraDB 5.5.10-20.1

`innodb_autoextend_increment`

- Description: Size in MB to increment an auto-extending shared tablespace file when it becomes full. If innodb_file_per_table was set to `1`, this setting does not apply to the resulting per-table tablespace files, which are automatically extended in their own way.
- Command line: `--innodb-autoextend-increment=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `64` (from MariaDB 10.0) `8` (before MariaDB 10.0),
- Range: `1` to `1000`

`innodb_autoinc_lock_mode`

- Description: The lock mode that is used when generating AUTO_INCREMENT values for InnoDB tables.
  - Valid values are:
    - `0` is the traditional lock mode.
    - `1` is the consecutive lock mode.
    - `2` is the interleaved lock mode.
  - In order to use Galera Cluster, the lock mode needs to be set to `2`.
  - See AUTO_INCREMENT Handling in InnoDB: AUTO_INCREMENT Lock Modes for more information.
- Command line: `--innodb-autoinc-lock-mode=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1`

- Range: `0` to `2`

`innodb_background_scrub_data_check_interval`

- Description: Check if spaces needs scrubbing every innodb_background_scrub_data_check_interval seconds. See Data Scrubbing. Deprecated and ignored from MariaDB 10.5.2.
- Command line: `--innodb-background-scrub-data-check-interval=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `3600`
- Range: `1` to `4294967295`
- Deprecated: MariaDB 10.5.2
- Removed: MariaDB 10.6.0

`innodb_background_scrub_data_compressed`

- Description: Enable scrubbing of compressed data by background threads (same as encryption_threads). See Data Scrubbing. Deprecated and ignored from MariaDB 10.5.2.
- Command line: `--innodb-background-scrub-data-compressed={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`
- Deprecated: MariaDB 10.5.2
- Removed: MariaDB 10.6.0

`innodb_background_scrub_data_interval`

- Description: Scrub spaces that were last scrubbed longer than this number of seconds ago. See Data Scrubbing. Deprecated and ignored from MariaDB 10.5.2.
- Command line: `--innodb-background-scrub-data-interval=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `604800`
- Range: `1` to `4294967295`
- Deprecated: MariaDB 10.5.2
- Removed: MariaDB 10.6.0

`innodb_background_scrub_data_uncompressed`

- Description: Enable scrubbing of uncompressed data by background threads (same as encryption_threads). See Data Scrubbing. Deprecated and ignored from MariaDB 10.5.2.
- Command line: `--innodb-background-scrub-data-uncompressed={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`

- Deprecated: MariaDB 10.5.2
- Removed: MariaDB 10.6.0

`innodb_blocking_buffer_pool_restore`

- Description: If set to `1` (`0` is default), XtraDB will wait until the least-recently used (LRU) dump is completely restored upon restart before reporting back to the server that it has successfully started up. Available with XtraDB only, not InnoDB.
- Command line: `innodb-blocking-buffer-pool-restore={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.0.0

`innodb_buf_dump_status_frequency`

- Description: Determines how often (as a percent) the buffer pool dump status should be printed in the logs. For example, `10` means that the buffer pool dump status is printed when every 10% of the number of buffer pool pages are dumped. The default is `0` (only start and end status is printed).
- Command line: `--innodb-buf-dump-status-frequency=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `100`

`innodb_buffer_pool_chunk_size`

- Description: Chunk size used for dynamically resizing the buffer pool. Note that changing this setting can change the size of the buffer pool. When large-pages is used this value is effectively rounded up to the next multiple of large-page-size. See Setting Innodb Buffer Pool Size Dynamically. From MariaDB 10.8.0, the variable is autosized based on the buffer pool size.
- Command line: `--innodb-buffer-pool-chunk-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value:
  - `autosize (0)`, resulting in innodb_buffer_pool_size/64, if large_pages round down to multiple of largest page size, with 1MiB minimum (from MariaDB 10.8.1)
  - `134217728` (until MariaDB 10.8.0)
- Range:
  - `0`, as autosize, and then `1048576` to `18446744073709551615` (from MariaDB 10.8)
  - `1048576` to innodb_buffer_pool_size/innodb_buffer_pool_instances (until MariaDB 10.7)
- Block size: `1048576`
- Deprecated and ignored from MariaDB 10.11.12, MariaDB 11.4.6, MariaDB 11.8.2

`innodb_buffer_pool_dump_at_shutdown`

- Description: Whether to record pages cached in the buffer pool on server shutdown, which reduces the length of the warmup the next time the server starts. The related innodb_buffer_pool_load_at_startup specifies whether the buffer pool is automatically warmed up at startup.
- Command line: `--innodb-buffer-pool-dump-at-shutdown={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`innodb_buffer_pool_dump_now`

- Description: Immediately records pages stored in the buffer pool. The related innodb_buffer_pool_load_now does the reverse, and will immediately warm up the buffer pool.
- Command line: `--innodb-buffer-pool-dump-now={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.0

`innodb_buffer_pool_dump_pct`

- Description: Dump only the hottest N% of each buffer pool.
- Command line: `--innodb-buffer-pool-dump-pct={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value:
  - `25`
- Range: `1` to `100`

`innodb_buffer_pool_evict`

- Description: Evict pages from the buffer pool. If set to “uncompressed” then all uncompressed pages are evicted from the buffer pool. Variable to be used only for testing. Only exists in DEBUG builds.
- Command line: `--innodb-buffer-pool-evict=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `""`
- Valid Values: “” or “uncompressed”

`innodb_buffer_pool_filename`

- Description: The file that holds the buffer pool list of page numbers set by innodb_buffer_pool_dump_at_shutdown and innodb_buffer_pool_dump_now.
- Command line: `--innodb-buffer-pool-filename=file`
- Scope: Global

- Dynamic: No
- Data Type: `string`
- Default Value: `ib_buffer_pool`
- Introduced: MariaDB 10.0

`innodb_buffer_pool_instances`

- Description: If innodb_buffer_pool_size is set to more than 1GB, innodb_buffer_pool_instances divides the InnoDB buffer pool into the specified number of instances. The default was 1 in MariaDB 5.5, but for large systems with buffer pools of many gigabytes, many instances could help reduce contention concurrency through MariaDB 10.2. The default is 8 in MariaDB 10 (except on Windows 32-bit, where it varies according to innodb_buffer_pool_size, or from MariaDB 10.2.2, where it is set to 1 if innodb_buffer_pool_size < 1GB). Each instance manages its own data structures and takes an equal portion of the total buffer pool size, so for example if innodb_buffer_pool_size is 4GB and `innodb_buffer_pool_instances` is set to 4, each instance are 1GB. Each instance should ideally be at least 1GB in size.
  Starting with MariaDB 10.3, performance improvements intended to reduce the overhead of context-switching between buffer pools changed the recommended number of innodb_buffer_pool_instances to one for every 128GB of buffer pool size. Based on these changes, the variable is deprecated and ignored from MariaDB 10.5.1, where the buffer pool runs in a single instance regardless of size.
- Command line: `--innodb-buffer-pool-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: >= MariaDB 10.0.4: `8`, `1` (>= MariaDB 10.2.2 if innodb_buffer_pool_size < 1GB), or dependent on innodb_buffer_pool_size (Windows 32-bit)
- Deprecated: MariaDB 10.5.1
- Removed: MariaDB 10.6.0

`innodb_buffer_pool_load_abort`

- Description: Aborts the process of restoring buffer pool contents started by innodb_buffer_pool_load_at_startup or innodb_buffer_pool_load_now.
- Command line: `--innodb-buffer-pool-load-abort={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_buffer_pool_load_at_startup`

- Description: Specifies whether the buffer pool is automatically warmed up when the server starts by loading the pages held earlier. The related innodb_buffer_pool_dump_at_shutdown specifies whether pages are saved at shutdown. If the buffer pool is large and taking a long time to load, increasing innodb_io_capacity at startup may help.
- Command line: `--innodb-buffer-pool-load-at-startup={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`innodb_buffer_pool_load_now`

- Description: Immediately warms up the buffer pool by loading the stored data pages. The related innodb_buffer_pool_dump_now does the reverse, and immediately records pages stored in the buffer pool.
- Command line: `--innodb-buffer-pool-load-now={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_buffer_pool_load_pages_abort`

- Description: Number of pages during a buffer pool load to process before signaling innodb_buffer_pool_load_abort=1. Debug builds only.
- Command line: `--innodb-buffer-pool-load-pages-abort=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `9223372036854775807`
- Range: `1` to `9223372036854775807`

`innodb_buffer_pool_populate`

- Description: When set to `1` (`0` is default), XtraDB will preallocate pages in the buffer pool on starting up so that NUMA[9] allocation decisions are made while the buffer cache is still clean. XtraDB only. This option was made ineffective in MariaDB 10.0.23. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-buffer-pool-populate={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.0.23
- Removed: MariaDB 10.3.0

`innodb_buffer_pool_restore_at_startup`

- Description: Time in seconds between automatic buffer pool dumps. If set to a non-zero value, XtraDB will also perform an automatic restore of the buffer pool at startup. If set to `0`, automatic dumps are not performed, nor automatic restores on startup. Replaced by innodb_buffer_pool_load_at_startup in MariaDB 10.0.
- Command line: `innodb-buffer-pool-restore-at-startup`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range - 32 bit: `0` to `4294967295`
- Range - 64 bit: `0` to `18446744073709547520`
- Removed: MariaDB 10.0 - replaced by innodb_buffer_pool_load_at_startup

`innodb_buffer_pool_shm_checksum`

- Description: Used with Percona's SHM buffer pool patch in XtraDB 5.5. Was shortly deprecated and removed in XtraDB 5.6. XtraDB only.
- Command line: `innodb-buffer-pool-shm-checksum={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Removed: MariaDB 10.0

`innodb_buffer_pool_shm_key`

- Description: Used with Percona's SHM buffer pool patch in XtraDB 5.5. Later deprecated in XtraDB 5.5, and removed in XtraDB 5.6.
- Command line: `innodb-buffer-pool-shm-key={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `0`
- Removed: MariaDB 10.0

`innodb_buffer_pool_size`

- Description: InnoDB buffer pool size in bytes. The primary value to adjust on a database server with entirely/primarily InnoDB tables, can be set up to 80% of the total memory in these environments. See the InnoDB Buffer Pool for more on setting this variable, and also Setting InnoDB Buffer Pool Size Dynamically if doing so dynamically.
- Command line: `--innodb-buffer-pool-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `134217728` (128MiB)
- Range:
  - Minimum: `5242880` (5MiB) for InnoDB Page Size <= 16k otherwise `25165824` (24MiB) for InnoDB Page Size > 16k (for versions less than next line)
  - Minimum: `2MiB` InnoDB Page Size = 4k, `3MiB` InnoDB Page Size = 8k, `5MiB` InnoDB Page Size = 16k, `10MiB` InnoDB Page Size = 32k, `20MiB` InnoDB Page Size = 64k, (>= MariaDB 10.2.42, >= MariaDB 10.3.33, >= MariaDB 10.4.23, >= MariaDB 10.5.14, >= MariaDB 10.6.6, >= MariaDB 10.7.2)
  - Minimum: 1GiB for innodb_buffer_pool_instances > 1 (<= MariaDB 10.7)
  - Maximum: `9223372036854775807` (8192PB) (all versions)
- Block size: `1048576`

`innodb_buffer_pool_size_auto_min`

- Description: Minimum `innodb_buffer_pool_size` in bytes for dynamic shrinking on memory pressure. Only affects Linux. If a memory pressure event is reported by Linux, the `innodb_buffer_pool_size` may be automatically shrunk towards this value. By default, set to `innodb_buffer_pool_size_max`, that is, memory pressure events will be ignored. `0` sets no minimum value.
- Command line: `--innodb-buffer-pool-size-auto-min=#`

- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `134217728` (128MiB)
- Range: `0` to `18446744073701163008`
- Block size: `8388608` (8 MB on 64-bit systems)
- Introduced: MariaDB 10.11.12, MariaDB 11.4.6, MariaDB 11.8.2

`innodb_buffer_pool_size_max`

> **Important:**
>
> `innodb_buffer_pool_size_max` is a **read-only** variable. If not specified at startup, it defaults to the initial `innodb_buffer_pool_size`, which effectively disables upward dynamic resizing at runtime (attempts to increase size results in *Warning 1292*).

> **Warning:**
>
> Automatic upward dynamic resizing is not yet implemented (MDEV-36197). This variable currently serves only as a pre-allocated virtual address ceiling for **manual** resizing operations.

- Description: Maximum `innodb_buffer_pool_size` value. On 64-bit systems other than IBM AIX, the default is 8 TiB, and the minimum 8 MiB. On other systems, the default and minimum are `0`, and the value `0` is replaced with the initial `innodb_buffer_pool_size` rounded up to the allocation unit (2 MiB or 8 MiB). The maximum value is 4GiB-2MiB on 32-bit systems and 16EiB-8MiB on 64-bit systems. This maximum is likely to be limited further by the operating system.
  On 64-bit systems, while the default maximum is 8 TiB, the actual available address space may be restricted by operating system limits (such as `RLIMIT_AS` [10]) or specific hardware architectures. If MariaDB is unable to allocate the default 8 TiB of virtual address space at startup, it automatically attempts to fall back to a 128 GiB limit to ensure the server can still start. (128 GiB is often chosen as a fallback because it's a safe value that almost any modern 64-bit Linux kernel/CPU can handle without special configuration, while still being plenty large for the vast majority of database workloads.)
- Command line: `--innodb-buffer-pool-size-max=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: specified by the initial value of innodb_buffer_pool_size, rounded up to the block size of that variable. See the section about buffer pool changes in MariaDB 10.11.12, 11.4.6, and 11.8.2.
- Range: `0` to `18446744073701163008`
- Block size: `8388608` (8 MB on 64-bit systems)
- Introduced: MariaDB 10.11.12, MariaDB 11.4.6, MariaDB 11.8.2

`innodb_change_buffer_dump`

- Description: If set, causes the contents of the InnoDB change buffer to be dumped to the server error log at startup. Only available in debug builds.
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

- Introduced: MariaDB 10.2.28, MariaDB 10.3.19, MariaDB 10.4.9

`innodb_change_buffer_max_size`

- Description: Maximum size of the InnoDB Change Buffer as a percentage of the total buffer pool. The default is 25%, and this can be increased up to 50% for servers with high write activity, and lowered down to 0 for servers used exclusively for reporting.
- Command line: `--innodb-change-buffer-max-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `25`
- Range: `0` to `50`
- Introduced: MariaDB 10.0
- Deprecated: MariaDB 10.9.0
- Removed: MariaDB 11.0.0

`innodb_change_buffering`

- Description: Sets how InnoDB change buffering is performed. See InnoDB Change Buffering for details on the settings. Deprecated and ignored from MariaDB 10.9.0.
- Command line: `--innodb-change-buffering=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration` (>= MariaDB 10.3.7), `string` (<= MariaDB 10.3.6)
- Default Value:
  - = MariaDB 10.5.15, MariaDB 10.6.7, MariaDB 10.7.3, MariaDB 10.8.2: `none`
  - <= MariaDB 10.5.14, MariaDB 10.6.6, MariaDB 10.7.2, MariaDB 10.8.1: `all`
- Valid Values: `inserts`, `none`, `deletes`, `purges`, `changes`, `all`
- Deprecated: MariaDB 10.9.0
- Removed: MariaDB 11.0.0

`innodb_change_buffering_debug`

- Description: If set to `1`, an InnoDB Change Buffering debug flag is set. `1` forces all changes to the change buffer, while `2` causes a crash at merge. `0`, the default, indicates no flag is set. Only available in debug builds.
- Command line: `--innodb-change-buffering-debug=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `2`

`innodb_checkpoint_age_target`

- Description: The maximum value of the checkpoint age. If set to `0`, has no effect. Removed in MariaDB 10.0/XtraDB 5.6 and replaced with InnoDB flushing method from MySQL 5.6.
- Command line: `innodb-checkpoint-age-target=#`

- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` upwards
- Removed: MariaDB 10.0 - replaced with InnoDB flushing method from MySQL 5.6.

`innodb_checksum_algorithm`

- Description: Specifies how the InnoDB tablespace checksum is generated and verified.
  - `innodb`: Backwards compatible with earlier versions (<= MariaDB 5.5). Deprecated in MariaDB 10.3.29, MariaDB 10.4.19, MariaDB 10.5.10 and removed in MariaDB 10.6. If really needed, data files can still be converted with innochecksum.
  - `crc32`: A newer, faster algorithm, but incompatible with earlier versions. Tablespace blocks are converted to the new format over time, meaning that a mix of checksums may be present.
  - `full_crc32` and `strict_full_crc32`: From MariaDB 10.4.3. Permits encryption to be supported over a SPATIAL INDEX, which `crc32` does not support. Newly-created data files will carry a flag that indicates that all pages of the file will use a full CRC-32C checksum over the entire page contents (excluding the bytes where the checksum is stored, at the very end of the page). Such files will always use that checksum, no matter what parameter `innodb_checksum_algorithm` is assigned to. Even if `innodb_checksum_algorithm` is modified later, the same checksum will continue to be used. A special flag are set in the FSP_SPACE_FLAGS in the first data page to indicate the new format of checksum and encryption/page_compressed. ROW_FORMAT=COMPRESSED tables will only use the old format. These tables do not support new features, such as larger innodb_page_size or instant ADD/DROP COLUMN. Also cleans up the MariaDB tablespace flags - flags are reserved to store the page_compressed compression algorithm, and to store the compressed payload length, so that checksum can be computed over the compressed (and possibly encrypted) stream and can be validated without decrypting or decompressing the page. In the full_crc32 format, there no longer are separate before-encryption and after-encryption checksums for pages. The single checksum is computed on the page contents that is written to the file.See MDEV-12026 for details.
  - `none`: Writes a constant rather than calculate a checksum. Deprecated in MariaDB 10.3.29, MariaDB 10.4.19, MariaDB 10.5.10 and removed in MariaDB 10.6 as was mostly used to disable the original, slow, page checksum for benchmarking purposes.
  - `strict_crc32`, `strict_innodb` and `strict_none`: The options are the same as the regular options, but InnoDB will halt if it comes across a mix of checksum values. These are faster, as both new and old checksum values are not required, but can only be used when setting up tablespaces for the first time.
- Command line: `--innodb-checksum-algorithm=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value:
  - `full_crc32` (>= MariaDB 10.5.0)
  - `crc32` (>= MariaDB 10.2.2 to <= MariaDB 10.4)
  - `innodb` (<= MariaDB 10.2.1)
- Valid Values:
  - > = MariaDB 10.6.0: `crc32`, `full_crc32`, `strict_crc32`, `strict_full_crc32`

    - MariaDB 10.5, >= MariaDB 10.4.3: `innodb`, `crc32`, `full_crc32`, `none`, `strict_innodb`, `strict_crc32`, `strict_none`, `strict_full_crc32`
    - <= MariaDB 10.4.2: `innodb`, `crc32`, `none`, `strict_innodb`, `strict_crc32`, `strict_none`

`innodb_checksums`

- Description: By default, InnoDB performs checksum validation on all pages read from disk, which provides extra fault tolerance. You would usually want this set to `1` in production environments, although setting it to `0` can provide marginal performance improvements. Deprecated and functionality replaced by innodb_checksum_algorithm in MariaDB 10.0, and should be removed to avoid conflicts. `ON` is equivalent to `--innodb_checksum_algorithm=innodb` and `OFF` to `--innodb_checksum_algorithm=none`.
- Command line: `--innodb-checksums`, `--skip-innodb-checksums`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 10.0
- Removed: MariaDB 10.5.0

`innodb_cleaner_lsn_age_factor`

- Description: XtraDB has enhanced page cleaner heuristics, and with these in place, the default InnoDB adaptive flushing may be too aggressive. As a result, a new LSN age factor formula has been introduced, controlled by this variable. The default setting, `high_checkpoint`, uses the new formula, while the alternative, `legacy`, uses the original algorithm. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `--innodb-cleaner-lsn-age-factor=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value:
    - `deprecated`
- Valid Values:
    - `deprecated`, `high_checkpoint`, `legacy`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_cmp_per_index_enabled`

- Description: If set to `ON` (`OFF` is default), per-index compression statistics are stored in the INFORMATION_SCHEMA.INNODB_CMP_PER_INDEX table. These are expensive to record, so this setting should only be changed with care, such as for performance tuning on development or replica servers.
- Command line: `--innodb-cmp-per-index-enabled={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.0

`innodb_commit_concurrency`

- Description: Limit to the number of transaction threads that can commit simultaneously. 0, the default, imposes no limit. While you can change from one positive limit to another at runtime, you cannot set this variable to 0, or change it from 0, while the server is running. Deprecated and ignored from MariaDB 10.5.5.
- Command line: `--innodb-commit-concurrency=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `1000`
- Deprecated: MariaDB 10.5.5
- Removed: MariaDB 10.6.0

`innodb_compression_algorithm`

- Description: Compression algorithm used for InnoDB page compression. The supported values are:
  - `none`: Pages are not compressed.
  - `zlib`: Pages are compressed using the bundled zlib compression algorithm.
  - `lz4`: Pages are compressed using the lz4 compression algorithm.
  - `lzo`: Pages are compressed using the lzo compression algorithm.
  - `lzma`: Pages are compressed using the lzma compression algorithm.
  - `bzip2`: Pages are compressed using the bzip2 compression algorithm.
  - `snappy`: Pages are compressed using the snappy algorithm.
  - On many distributions, MariaDB may not support all page compression algorithms by default. From MariaDB 10.7, libraries can be installed as a plugin. See Compression Plugins.
  - See InnoDB Page Compression: Configuring the InnoDB Page Compression Algorithm for more information.
- Command line: `--innodb-compression-algorithm=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `zlib`
- Valid Values: `none`, `zlib`, `lz4`, `lzo`, `lzma`, `bzip2` or `snappy`

`innodb_compression_default`

- Description: Whether or not InnoDB page compression is enabled by default for new tables.
  - The default value is `OFF`, which means new tables are not compressed.
  - See InnoDB Page Compression: Enabling InnoDB Page Compression by Default for more information.
- Command line: `--innodb-compression-default={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_compression_failure_threshold_pct`

- Description: Specifies the percentage cutoff for expensive compression failures during updates to a table that uses InnoDB page compression, after which free space is added to each new compressed page, dynamically

adjusted up to the level set by innodb_compression_pad_pct_max. Zero disables checking of compression efficiency and adjusting padding.
    - See InnoDB Page Compression: Configuring the Failure Threshold and Padding for more information.
- Command line: `--innodb-compression-failure-threshold-pct=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `5`
- Range: `0` to `100`
- Introduced: MariaDB 10.0

`innodb_compression_level`

- Description: Specifies the default level of compression for tables that use InnoDB page compression.
    - Only a subset of InnoDB page compression algorithms support compression levels. If an InnoDB page compression algorithm does not support compression levels, then the compression level value is ignored.
    - The compression level can be set to any value between `1` and `9`. The default compression level is `6`. The range goes from the fastest to the most compact, which means that `1` is the fastest and `9` is the most compact.
    - See InnoDB Page Compression: Configuring the Default Compression Level for more information.
- Command line: `--innodb-compression-level=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `6`
- Range: `1` to `9`
- Introduced: MariaDB 10.0

`innodb_compression_pad_pct_max`

- Description: The maximum percentage of reserved free space within each compressed page for tables that use InnoDB page compression. Reserved free space is used when the page's data is reorganized and might be recompressed. Only used when innodb_compression_failure_threshold_pct is not zero, and the rate of compression failures exceeds its setting.
    - See InnoDB Page Compression: Configuring the Failure Threshold and Padding for more information.
- Command line: `--innodb-compression-pad-pct-max=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `50`
- Range: `0` to `75`
- Introduced: MariaDB 10.0

`innodb_concurrency_tickets`

- Description: Number of times a newly-entered thread can enter and leave InnoDB until it is again subject to the limitations of innodb_thread_concurrency and may possibly be queued. Deprecated and ignored from MariaDB 10.5.5.

- Command line: `--innodb-concurrency-tickets=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `0` (>= MariaDB 10.5.5)
  - `5000` (<= MariaDB 10.5.4)
- Range: `1` to `18446744073709551615`
- Deprecated: MariaDB 10.5.5
- Removed: MariaDB 10.6.0

`innodb_corrupt_table_action`

- Description: What action to perform when a corrupt table is found. XtraDB only.
  - When set to `assert`, the default, XtraDB will intentionally crash the server when it detects corrupted data in a single-table tablespace, with an assertion failure.
  - When set to `warn`, it will pass corruption as corrupt table instead of crashing, and disable all further I/O (except for deletion) on the table file.
  - If set to `salvage`, read access is permitted, but corrupted pages are ignored. innodb_file_per_table must be enabled for this option. Previously named `innodb_pass_corrupt_table`.
  - Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-corrupt-table-action=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value:
  - `assert` (<= MariaDB 10.1)
  - `deprecated` (<= MariaDB 10.2.6)
- Valid Values:
  - `deprecated`, `assert`, `warn`, `salvage`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_data_file_buffering`

- Description: Whether to enable the file system cache for data files. Set to `OFF` by default, are set to `ON` if innodb_flush_method is set to `fsync`, `littlesync`, `nosync`, or (Windows specific) `normal`.
- Command line: `--innodb-data-file-buffering={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 11.0.0

`innodb_data_file_path`

- Description: Individual InnoDB data files, paths and sizes. The value of innodb_data_home_dir is joined to each path specified by innodb_data_file_path to get the full directory path. If innodb_data_home_dir is an empty string, absolute paths can be specified here. A file size is specified (with K for kilobytes, M for megabytes and G for gigabytes). Also whether or not to `autoextend` the data file, and whether or not to autoshrink on startup may also be specified.
- Command line: `--innodb-data-file-path=name`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `ibdata1:12M:autoextend` (from MariaDB 10.0), `ibdata1:10M:autoextend` (before MariaDB 10.0)

`innodb_data_file_write_through`

- Description: Whether writes to InnoDB data files (including the temporary tablespace) are write through. Set to `OFF` by default, are set to `ON` if innodb_flush_method is set to `O_DSYNC`. On systems that support FUA it may make sense to enable write-through, to avoid extra system calls. See InnoDB Flush Method for a discussion of FUA and write-through behavior.
- Command line: `--innodb-data-file-write-through={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 11.0.0

`innodb_data_home_dir`

- Description: Directory path for all InnoDB data files in the shared tablespace (assuming innodb_file_per_table is not enabled). File-specific information can be added in innodb_data_file_path, as well as absolute paths if innodb_data_home_dir is set to an empty string.
- Command line: `--innodb-data-home-dir=path`
- Scope: Global
- Dynamic: No
- Data Type: `directory name`
- Default Value: `The MariaDB data directory`

`innodb_deadlock_detect`

- Description: By default, the InnoDB deadlock detector is enabled. If set to off, deadlock detection is disabled and MariaDB will rely on innodb_lock_wait_timeout instead. This may be more efficient in systems with high concurrency as deadlock detection can cause a bottleneck when a number of threads have to wait for the same lock.
- Command line: `--innodb-deadlock-detect`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`

`innodb_deadlock_report`

- Description: How to report deadlocks (if innodb_deadlock_detect=ON).
  - `off`: Do not report any details of deadlocks.
  - `basic`: Report transactions and waiting locks.
  - `full`: Default. Report transactions, waiting locks and blocking locks.
- Command line: `--innodb-deadlock-report=val`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `full`
- Valid Values: `off`, `basic`, `full`
- Introduced: MariaDB 10.6.0

`innodb_default_page_encryption_key`

- Description: Encryption key used for page encryption.
  - See Data-at-Rest Encryption and InnoDB Encryption Keys for more information.
- Command line: `--innodb-default-page-encryption-key=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `255`
- Introduced: MariaDB 10.1.3
- Removed: MariaDB 10.1.4

`innodb_default_encryption_key_id`

- Description: ID of encryption key used by default to encrypt InnoDB tablespaces.
  - See Data-at-Rest Encryption and InnoDB Encryption Keys for more information.
- Command line: `--innodb-default-encryption-key-id=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `4294967295`

`innodb_default_row_format`

- Description: Specifies the default row format to be used for InnoDB tables. The compressed row format cannot be set as the default.
  - See InnoDB Row Formats Overview: Default Row Format for more information.
- Command line: `--innodb-default-row-format=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `dynamic`

- Valid Values: `redundant`, `compact` or `dynamic`

`innodb_defragment`

- Description: When set to `1` (the default is `0`), InnoDB defragmentation is enabled. When set to FALSE, all existing defragmentation are paused and new defragmentation commands will fail. Paused defragmentation commands will resume when this variable is set to true again. See Defragmenting InnoDB Tablespaces.
- Command line: `--innodb-defragment={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 11.0.1
- Removed: MariaDB 11.1.0

`innodb_defragment_fill_factor`

- Description:. Indicates how full defragmentation should fill a page. Together with innodb_defragment_fill_factor_n_recs ensures defragmentation won't pack the page too full and cause page split on the next insert on every page. The variable indicating more defragmentation gain is the one effective. See Defragmenting InnoDB Tablespaces.
- Command line: `--innodb-defragment-fill-factor=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `double`
- Default Value: `0.9`
- Range: `0.7` to `1`
- Deprecated: MariaDB 11.0.1
- Removed: MariaDB 11.1.0

`innodb_defragment_fill_factor_n_recs`

- Description: Number of records of space that defragmentation should leave on the page. This variable, together with innodb_defragment_fill_factor, is introduced so defragmentation won't pack the page too full and cause page split on the next insert on every page. The variable indicating more defragmentation gain is the one effective. See Defragmenting InnoDB Tablespaces.
- Command line: `--innodb-defragment-fill-factor-n-recs=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `20`
- Range: `1` to `100`
- Deprecated: MariaDB 11.0.1
- Removed: MariaDB 11.1.0

`innodb_defragment_frequency`

- Description: Maximum times per second for defragmenting a single index. This controls the number of times the defragmentation thread can request X_LOCK on an index. The defragmentation thread will check whether 1/defragment_frequency (s) has passed since it last worked on this index, and put the index back in the queue if

not enough time has passed. The actual frequency can only be lower than this given number. See Defragmenting InnoDB Tablespaces.

- Command line: `--innodb-defragment-frequency=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `integer`
- Default Value: `40`
- Range: `1` to `1000`
- Deprecated: MariaDB 11.0.1
- Removed: MariaDB 11.1.0

`innodb_defragment_n_pages`

- Description: Number of pages considered at once when merging multiple pages to defragment. See Defragmenting InnoDB Tablespaces.
- Command line: `--innodb-defragment-n-pages=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `7`
- Range: `2` to `32`
- Deprecated: MariaDB 11.0.1
- Removed: MariaDB 11.1.0

`innodb_defragment_stats_accuracy`

- Description: Number of defragment stats changes there are before the stats are written to persistent storage. Defaults to zero, meaning disable defragment stats tracking. See Defragmenting InnoDB Tablespaces.
- Command line: `--innodb-defragment-stats-accuracy=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4294967295`
- Deprecated: MariaDB 11.0.1
- Removed: MariaDB 11.1.0

`innodb_dict_size_limit`

- Description: Size in bytes of a soft limit the memory used by tables in the data dictionary. Once this limit is reached, XtraDB will attempt to remove unused entries. If set to `0`, the default and standard InnoDB behavior, there is no limit to memory usage. Removed in MariaDB 10.0/XtraDB 5.6 and replaced by MySQL 5.6's new table_definition_cache implementation.
- Command line: `innodb-dict-size-limit=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`

- Default Value - 32 bit: `2147483648`
- Default Value - 64 bit: `9223372036854775807`
- Removed: MariaDB 10.0 - replaced by MySQL 5.6's table_definition_cache implementation.

`innodb_disable_sort_file_cache`

- Description: If set to `1` (`0` is default), the operating system file system cache for merge-sort temporary files is disabled.
- Command line: `--innodb-disable-sort-file-cache={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_disallow_writes`

- Description: Tell InnoDB to stop any writes to disk.
- Command line: None
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.3.35, MariaDB 10.4.25, MariaDB 10.5.16, MariaDB 10.6.8, MariaDB 10.7.4

`innodb_doublewrite`

> **CURRENT**
>
> - Description: If set to `ON`, the default, to improve fault tolerance InnoDB first stores data to a doublewrite buffer before writing it to data file. Disabling will provide a marginal performance improvement, and assumes that writes of innodb_page_size are atomic. `fast` is available from MariaDB 11.0.6, and is like `ON`, but writes are not synchronized to data files. The deprecated start-up parameter innodb_flush_method=NO_FSYNC will cause `innodb_doublewrite=ON` to be changed to `innodb_doublewrite=fast`, which will prevent InnoDB from making any durable writes to data files. This is normally done right before the log checkpoint LSN is updated. Depending on the file systems being used and their configuration, this may or may not be safe.
>
> The value `innodb_doublewrite=fast` differs from the previous combination of `innodb_doublewrite=ON` and `innodb_flush_method=O_DIRECT_NO_FSYNC` by always invoking `os_file_flush()` on the doublewrite buffer itself in `buf_dblwr_t::flush_buffered_writes_completed()`. This is safer when there are multiple doublewrite batches between checkpoints.
>
> Typically, once per second, `buf_flush_page_cleaner()` writes out up to `innodb_io_capacity` pages and advance the log checkpoint. Also typically, `innodb_io_capacity` > `128`, which is the size of the doublewrite buffer in pages. If `os_file_flush_func()` is not invoked between doublewrite batches, writes may be reordered in an unsafe way.
>
> The setting `innodb_doublewrite=fast` can be safe when the doublewrite buffer (the first file of the system tablespace) and the data files reside on the same file system.
> - Command line: `--innodb-doublewrite{=val}`, `--skip-innodb-doublewrite`
> - Scope: Global
> - Dynamic: Yes

- Data Type: `enum`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `fast`

< MARIADB 11.0.6

- Description: If set to `1`, the default, to improve fault tolerance InnoDB first stores data to a doublewrite buffer before writing it to data file. Disabling will provide a marginal performance improvement.
- Command line: `--innodb-doublewrite`, `--skip-innodb-doublewrite`
- Scope: Global
- Dynamic:No
- Data Type: `boolean`
- Default Value: `ON`

`innodb_doublewrite_file`

- Description: The absolute or relative path and filename to a dedicated tablespace for the doublewrite buffer. In heavy workloads, the doublewrite buffer can impact heavily on the server, and moving it to a different drive will reduce contention on random reads. Since the doublewrite buffer is mostly sequential writes, a traditional HDD is a better choice than SSD. This Percona XtraDB variable has not been ported to XtraDB 5.6.
- Command line: `innodb-doublewrite-file=filename`
- Scope: Global
- Dynamic: No
- Data Type: `filename`
- Default Value: `NULL`
- Removed: MariaDB 10.0

`innodb_empty_free_list_algorithm`

- Description: XtraDB 5.6.13-61 introduced an algorithm to assist with reducing mutex contention when the buffer pool free list is empty, controlled by this variable. If set to `backoff`, the new algorithm is used. If set to `legacy`, the original InnoDB algorithm is used. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades. See #1651657 for the reasons this was changed back to `legacy` in XtraDB 5.6.36-82.0.
- Command line: `innodb-empty-free-list-algorithm=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `deprecated`
- Valid Values: `deprecated`, `backoff`, `legacy`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_enable_unsafe_group_commit`

- Description: Unneeded after XtraDB 1.0.5. If set to `0`, the default, InnoDB keep transactions between the transaction log and binary log is in the same order. Safer, but slower. If set to `1`, transactions can be group-committed, but there is no guarantee of the order being kept, and a small risk of the two logs getting out of sync. In write-intensive environments, can lead to a significant improvement in performance.

- Command line: `--innodb-enable-unsafe-group-commit`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `1`
- Removed: Not needed after XtraDB 1.0.5

`innodb_encrypt_log`

- Description: Enables encryption of the InnoDB redo log. This also enables encryption of some temporary files created internally by InnoDB, such as those used for merge sorts and row logs.
  - See Data-at-Rest Encryption and Enabling InnoDB Encryption: Enabling Encryption for the Redo Log for more information.
- Command line: `--innodb-encrypt-log`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_encrypt_tables`

- Description: Enables automatic encryption of all InnoDB tablespaces.
  - `OFF` - Disables table encryption for all new and existing tables that have the ENCRYPTED table option set to `DEFAULT`.
  - `ON` - Enables table encryption for all new and existing tables that have the ENCRYPTED table option set to `DEFAULT`, but allows unencrypted tables to be created.
  - `FORCE` - Enables table encryption for all new and existing tables that have the ENCRYPTED table option set to `DEFAULT`, and doesn't allow unencrypted tables to be created (`CREATE TABLE ... ENCRYPTED=NO` fails).
  - See Data-at-Rest Encryption and Enabling InnoDB Encryption: Enabling Encryption for Automatically Encrypted Tablespaces for more information.
- Command line: `--innodb-encrypt-tables={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Valid Values: `ON`, `OFF`, `FORCE`

`innodb_encrypt_temporary_tables`

- Description: Enables automatic encryption of the InnoDB temporary tablespace.
  - See Data-at-Rest Encryption and InnoDB Enabling Encryption: Enabling Encryption for Temporary Tablespaces for more information.
- Command line: `--innodb-encrypt-temporary-tables={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

- Valid Values: `ON`, `OFF`
- Introduced: MariaDB 10.2.26, MariaDB 10.3.17, MariaDB 10.4.7

`innodb_encryption_rotate_key_age`

- Description: Re-encrypt in background any page having a key older than this number of key versions. When setting up encryption, this variable must be set to a non-zero value. Otherwise, when you enable encryption through innodb_encrypt_tables MariaDB won't be able to automatically encrypt any unencrypted tables.
  - See Data-at-Rest Encryption and InnoDB Encryption Keys: Key Rotation for more information.
- Command line: `--innodb-encryption-rotate-key-age=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `4294967295`

`innodb_encryption_rotation_iops`

- Description: Use this many iops for background key rotation operations performed by the background encryption threads.
  - See Data-at-Rest Encryption and InnoDB Encryption Keys: Key Rotation for more information.
- Command line: `--innodb-encryption-rotation_iops=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `0` to `4294967295`

`innodb_encryption_threads`

- Description: Number of background encryption threads performing background key rotation and scrubbing. When setting up encryption, this variable must be set to a non-zero value. Otherwise, when you enable encryption through innodb_encrypt_tables MariaDB won't be able to automatically encrypt any unencrypted tables. Recommended never be set higher than 255.
  - See Data-at-Rest Encryption and InnoDB Background Encryption Threads for more information.
- Command line: `--innodb-encryption-threads=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range:
  - `0` to `4294967295` (<= MariaDB 10.1.45, MariaDB 10.2.32, MariaDB 10.3.23, MariaDB 10.4.13, MariaDB 10.5.3)
  - `0` to `255` (>= MariaDB 10.1.46, MariaDB 10.2.33, MariaDB 10.3.24, MariaDB 10.4.14, MariaDB 10.5.4)

`innodb_extra_rsegments`

- Description: Removed in XtraDB 5.5 and replaced by innodb_rollback_segments. Usually there is one rollback segment protected by single mutex, a source of contention in high write environments. This option specifies a

number of extra user rollback segments. Changing the default will make the data readable by XtraDB only, and is incompatible with InnoDB. After modifying, the server must be slow-shutdown. If there is existing data, it must be dumped before changing, and re-imported after the change has taken effect.

- Command line: `--innodb-extra-rsegments=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `126`
- Removed: XtraDB 5.5 - replaced by innodb_rollback_segments

`innodb_extra_undoslots`

- Description: Usually, InnoDB has 1024 undo slots in its rollback segment, so 1024 transactions can run in parallel. New transactions will fail if all slots are used. Setting this variable to `1` expands the available undo slots to 4072. Not recommended unless you get the `Warning: cannot find a free slot for an undo log` error in the error log, as it makes data files unusable for ibbackup, or MariaDB servers not run with this option. See also undo log.
- Command line: `--innodb-extra-undoslots={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: XtraDB 5.5

`innodb_fake_changes`

- Description: From MariaDB 5.5 until MariaDB 10.1, XtraDB-only option that enables the fake changes feature. In replication, setting up or restarting a replica can cause a replication reads to perform more slowly, as MariaDB is single-threaded and needs to read the data before it can execute the queries. This can be speeded up by prefetching threads to warm the server, replaying the statements and then rolling back at commit. This however has an overhead from locking rows only then to undo changes at rollback. Fake changes attempts to reduce this overhead by reading the rows for `INSERT`, `UPDATE`, and `DELETE` statements but not updating them. The rollback is then very fast with little or nothing to do. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades. Not present in MariaDB 10.3 and beyond.
- Command line: `--innodb-fake-changes={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_fast_checksum`

- Description: Implements a more CPU efficient XtraDB checksum algorithm, useful for write-heavy loads with high I/O. If set to `1` on a server with tables that have been created with it set to `0`, reads are slower, so tables should be recreated (dumped and reloaded). XtraDB fails to start if set to `0` and there are tables created while set to `1`. Replaced with innodb_checksum_algorithm in MariaDB 10.0/XtraDB 5.6.

- Command line: `--innodb-fast-checksum={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.0/XtraDB 5.6 - replaced with innodb_checksum_algorithm

`innodb_fast_shutdown`

- Description: The shutdown mode:
  - `0` - InnoDB performs a slow shutdown, including full purge (before MariaDB 10.3.6, not always, due to MDEV-13603) and change buffer merge. Can be very slow, even taking hours in extreme cases.
  - `1` - the default, InnoDB performs a fast shutdown, not performing a full purge or an insert buffer merge.
  - `2`, the InnoDB redo log is flushed and a cold shutdown takes place, similar to a crash. The resulting startup then performs crash recovery. Extremely fast, in cases of emergency, but risks corruption. Not suitable for upgrades between major versions!
  - `3` (from MariaDB 10.3.6) - active transactions will not be rolled back, but all changed pages are written to data files. The active transactions are rolled back by a background thread on a subsequent startup. The fastest option that will not involve InnoDB redo log apply on subsequent startup. See MDEV-15832.
- Command line: `--innodb-fast-shutdown[=#]`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `3` (>= MariaDB 10.3.6), `0` to `2` (<= MariaDB 10.3.5)

`innodb_fatal_semaphore_wait_threshold`

- Description: In MariaDB, the fatal semaphore timeout is configurable. This variable sets the maximum number of seconds for semaphores to time out in InnoDB.
- Command line: `--innodb-fatal-semaphore-wait-threshold=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `600`
- Range: `1` to `4294967295`

`innodb_file_format`

- Description: File format for new InnoDB tables. Can either be `Antelope`, the default and the original format, or `Barracuda`, which supports compression. Note that this value is also used when a table is re-created with an ALTER TABLE which requires a table copy. See XtraDB/InnoDB File Format for more on the file formats. Removed in 10.3.1 and restored as a deprecated and unused variable in 10.4.3 for compatibility purposes.
- Command line: `--innodb-file-format=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value:
  - `Barracuda`

- Valid Values: `Antelope`, `Barracuda`
- Deprecated: MariaDB 10.2
- Removed: MariaDB 10.3.1
- Re-introduced: MariaDB 10.4.3 (for compatibility purposes)
- Removed: MariaDB 10.6.0

`innodb_file_format_check`

- Description: If set to `1`, the default, InnoDB checks the shared tablespace file format tag. If this is higher than the current version supported by XtraDB/InnoDB (for example Barracuda when only Antelope is supported), XtraDB/InnoDB will not start. If it the value is not higher, XtraDB/InnoDB starts correctly and the innodb_file_format_max value is set to this value. If `innodb_file_format_check` is set to `0`, no checking is performed. See XtraDB/InnoDB File Format for more on the file formats.
- Command line: `--innodb-file-format-check={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 10.2
- Removed: MariaDB 10.3.1

`innodb_file_format_max`

- Description: The highest XtraDB/InnoDB file format. This is set to the value of the file format tag in the shared tablespace on startup (see innodb_file_format_check). If the server later creates a higher table format, `innodb_file_format_max` is set to that value. See XtraDB/InnoDB File Format for more on the file formats.
- Command line: `--innodb-file-format-max=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `Antelope`
- Valid Values: `Antelope`, `Barracuda`
- Deprecated: MariaDB 10.2
- Removed: MariaDB 10.3.1

`innodb_file_per_table`

- Description: If set to `ON`, then new InnoDB tables are created with their own InnoDB file-per-table tablespaces. If set to `OFF`, then new tables are created in the InnoDB system tablespace instead. Page compression is only available with file-per-table tablespaces. Note that this value is also used when a table is re-created with an ALTER TABLE which requires a table copy. Deprecated in MariaDB 11.0 as there's no benefit to setting to `OFF`, the original InnoDB default.
- Command line: `--innodb-file-per-table`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 11.0.1

`innodb_fill_factor`

- Description: Percentage of B-tree page filled during bulk insert (sorted index build). Used as a hint rather than an absolute value. Setting to `70`, for example, reserves 30% of the space on each B-tree page for the index to grow in future.
- Command line: `--innodb-fill-factor=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `10` to `100`

`innodb_flush_log_at_timeout`

- Description: Interval in seconds to write and flush the InnoDB redo log. Before MariaDB 10, this was fixed at one second, which is still the default, but this can now be changed. It's usually increased to reduce flushing and avoid impacting performance of binary log group commit.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `2700`

`innodb_flush_log_at_trx_commit`

- Description: Set to `1`, along with sync_binlog=1 for the greatest level of fault tolerance. The value of innodb_use_global_flush_log_at_trx_commit determines whether this variable can be reset with a `SET` statement or not.
  - `1` The default, the log buffer is written to the InnoDB redo log file and a flush to disk performed after each transaction. This is required for full ACID[11] compliance.
  - `0` Nothing is done on commit; rather the log buffer is written and flushed to the InnoDB redo log once a second. This gives better performance, but a server crash can erase the last second of transactions.
  - `2` The log buffer is written to the InnoDB redo log after each commit, but flushing takes place every innodb_flush_log_at_timeout seconds (by default once a second). Performance is slightly better, but a OS or power outage can cause the last second's transactions to be lost.
  - `3` The log buffer is written to the InnoDB redo log file and flushed to disk at both the prepare and the commit phase of each transaction. This is slower than `1`, and the extra flush at prepare is usually redundant. Like `1`, it guarantees that after a crash, committed transactions are not lost and remain consistent with the binary log and other transactional engines. See Binlog group commit and innodb_flush_log_at_trx_commit.
- Command line: `--innodb-flush-log-at-trx-commit[=#]`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `1`
- Valid Values: `0`, `1`, `2` or `3`

Note: When the InnoDB-based Binary Log is enabled (`--binary-storage-engine=innodb`), this option manages the durability of commits for both binlog files and InnoDB table data. Also, in this configuration, there is no separate binlog `fsync` step and no two-phase commit between InnoDB and the binary log.

`innodb_flush_method`

CURRENT

- Description: InnoDB flushing method. **Deprecated from MariaDB 11.0.** The variable can still be set, but the preferred way to control flushing behavior is to use `SET GLOBAL` on the four replacement Boolean dynamic parameters, which can be changed while the server is running:
  - innodb_log_file_buffering (enable file system cache on the InnoDB write-ahead log; added in 10.8.4, 10.9.2)
  - innodb_data_file_buffering (enable file system cache on data files)
  - innodb_log_file_write_through (enable write-through on the log)
  - innodb_data_file_write_through (enable write-through on persistent data files)
- For the full discussion of buffering, write-through, FUA, and the flag names used as values, see InnoDB Flush Method and Storage I/O: Buffering and Persistence.
- When `innodb_flush_method` is set, the value is translated into the four Boolean parameters as follows:
  - `O_DSYNC`: innodb_log_file_write_through=ON, innodb_data_file_write_through=ON, innodb_data_file_buffering=OFF, and (if supported) innodb_log_file_buffering=OFF.
  - `fsync`, `littlesync`, `nosync`, or (on Windows) `normal`: innodb_log_file_write_through=OFF, innodb_data_file_write_through=OFF, and innodb_data_file_buffering=ON.
- Command line: `--innodb-flush-method=name`
- Scope: Global
- Dynamic: No
- Data Type: `enumeration`
- Default Value:
  - `O_DIRECT` (Unix, >= MariaDB 10.6.0)
  - `fsync` (only in MariaDB < 10.5)
- Valid Values:
  - Unix: `fsync`, `O_DSYNC`, `littlesync`, `nosync`, `O_DIRECT`, `O_DIRECT_NO_FSYNC`
  - Windows: `unbuffered`, `async_unbuffered`, `normal`
- Deprecated: MariaDB 11.0

< MARIADB 11.0

- Description: InnoDB flushing method. Windows always uses `async_unbuffered`, meaning that this variable has no effect. Adjusting this variable can give performance improvements, but behavior differs widely on different filesystems. Changing from the default value may cause problems in some situations, so test and benchmark carefully before adjusting. In MariaDB, Windows recognizes and correctly handles the Unix methods, but if no methods are specified, it uses its own default – unbuffered write (analog of `O_DIRECT`) plus syncs (for instance, `FileFlushBuffers()`) for all files.
- A detailed description of the variable and its effects can be found on this page.
  - `O_DSYNC` is used to open and flush logs, and `fsync()` to flush the data files.
  - `O_DIRECT` is used to open data files, and `fsync()` to flush data and logs. This is the default on Unix from MariaDB 10.6.
  - `O_DIRECT_NO_FSYNC` uses `O_DIRECT` during flushing I/O, but skips `fsync()` afterwards. Not suitable for XFS filesystems. Generally not recommended over `O_DIRECT`, as it does not get the benefit of innodb_use_native_aio=ON.
  - `unbuffered` - Windows-only default
  - `async_unbuffered` - Windows-only, alias for `unbuffered`

    - `normal` - Windows-only, alias for `fsync`
    - `littlesync` - for internal testing only
    - `nosync` - for internal testing only
- Command line: `--innodb-flush-method=name`
- Scope: Global
- Dynamic: No
- Data Type: `enumeration`
- Default Value:
    - `O_DIRECT` (Unix, >= MariaDB 10.6.0)
    - `fsync` (only in MariaDB < 10.5)
- Valid Values:
    - Unix: `fsync`, `O_DSYNC`, `littlesync`, `nosync`, `O_DIRECT`, `O_DIRECT_NO_FSYNC`
    - Windows: `unbuffered`, `async_unbuffered`, `normal`

`innodb_flush_neighbor_pages`

- Description: Determines whether, when dirty pages are flushed to the data file, neighboring pages in the data file are flushed at the same time. If set to `none`, the feature is disabled. If set to `area`, the default, the standard InnoDB behavior is used. For each page to be flushed, dirty neighboring pages are flushed too. If there's little head seek delay, such as SSD or large enough write buffer, one of the other two options may be more efficient. If set to `cont`, for each page to be flushed, neighboring contiguous blocks are flushed at the same time. Being contiguous, a sequential I/O is used, unlike the random I/O used in `area`. Replaced by innodb_flush_neighbors in MariaDB 10.0/XtraDB 5.6.
- Command line: `innodb-flush-neighbor-pages=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `area`
- Valid Values: `none` or `0`, `area` or `1`, `cont` or `2`
- Removed: MariaDB 10.0/XtraDB 5.6 - replaced by innodb_flush_neighbors

`innodb_flush_neighbors`

- Description: Determines whether flushing a page from the buffer pool will flush other dirty pages in the same group of pages (extent). In high write environments, if flushing is not aggressive enough, it can fall behind resulting in higher memory usage, or if flushing is too aggressive, cause excess I/O activity. SSD devices, with low seek times, would be less likely to require dirty neighbor flushing to be set. Since MariaDB 10.4.4 an attempt is made under Windows and Linux to determine SSD status which was exposed in information_schema.innodb_tablespaces_scrubbing_table. This variable is ignored for table spaces that are detected as stored on SSD (and the `0` behavior applies).
    - `1`: The default, flushes contiguous dirty pages in the same extent from the buffer pool.
    - `0`: No other dirty pages are flushed.
    - `2`: Flushes dirty pages in the same extent from the buffer pool.
- Command line: `--innodb-flush-neighbors=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`

- Default Value: `1`
- Valid Values: `0`, `1`, `2`

`innodb_flush_sync`

- Description: If set to `ON`, the default, the innodb_io_capacity setting is ignored for I/O bursts occurring at checkpoints.
- Command line: `--innodb-flush-sync={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`innodb_flushing_avg_loops`

- Description: Determines how quickly adaptive flushing will respond to changing workloads. The value is the number of iterations that a previously calculated flushing state snapshot is kept. Increasing the value smooths and slows the rate that the flushing operations change, while decreasing it causes flushing activity to spike quickly in response to workload changes.
- Command line: `--innodb-flushing-avg-loops=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `30`
- Range: `1` to `1000`

`innodb_force_load_corrupted`

- Description: Set to `0` by default, if set to `1`, InnoDB are permitted to load tables marked as corrupt. Only use this to recover data you can't recover any other way, or in troubleshooting. Always restore to `0` when the returning to regular use. Given that MDEV-11412 in MariaDB 10.5.4 aims to allow any metadata for a missing or corrupted table to be dropped, and given that MDEV-17567 and MDEV-25506 and related tasks made DDL operations crash-safe, the parameter no longer serves any purpose and was removed in MariaDB 10.6.6.
- Command line: `--innodb-force-load-corrupted`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.6.6

`innodb_force_primary_key`

- Description: If set to `1` (`0` is default) CREATE TABLEs without a primary or unique key where all keyparts are NOT NULL will not be accepted, and will return an error.
- Command line: `--innodb-force-primary-key`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_force_recovery`

- Description: InnoDB crash recovery mode. `0` is the default. The other modes are for recovery purposes only, and no data can be changed while another mode is active. Some queries relying on indexes are also blocked. See InnoDB Recovery Modes for more on mode specifics.
- Command line: `--innodb-force-recovery=#`
- Scope: Global
- Dynamic: No
- Data Type: `enumeration`
- Default Value: `0`
- Range: `0` to `6`

`innodb_foreground_preflush`

- Description: Before XtraDB 5.6.13-61.0, if the checkpoint age is in the sync preflush zone while a thread is writing to the XtraDB redo log, it will try to advance the checkpoint by issuing a flush list flush batch if this is not already being done. XtraDB has enhanced page cleaner tuning, and may already be performing furious flushing, resulting in the flush simply adding unneeded mutex pressure. Instead, the thread now waits for the flushes to finish, and then has two options, controlled by this variable. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
  - `exponential_backoff` - thread sleeps while it waits for the flush list flush to occur. The sleep time randomly progressively increases, periodically reset to avoid runaway sleeps.
  - `sync_preflush` - thread issues a flush list batch, and waits for it to complete. This is the same as is used when the page cleaner thread is not running.
- Command line: `innodb-foreground-preflush=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value:
  - `deprecated`
- Valid Values:
  - `deprecated`, `exponential_backoff`, `sync_preflush`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_ft_aux_table`

- Description: Diagnostic variable intended only to be set at runtime. It specifies the qualified name (for example `test/ft_innodb`) of an InnoDB table that has a FULLTEXT index, and after being set the INFORMATION_SCHEMA tables INNODB_FT_INDEX_TABLE, INNODB_FT_INDEX_CACHE, INNODB_FT_CONFIG, INNODB_FT_DELETED, and INNODB_FT_BEING_DELETED will contain search index information for the specified table.
- Command line: `--innodb-ft-aux-table=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`

`innodb_ft_cache_size`

- Description: Cache size available for a parsed document while creating an InnoDB FULLTEXT index.

- Command line: `--innodb-ft-cache-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `8000000`

`innodb_ft_enable_diag_print`

- Description: If set to `1`, additional full-text search diagnostic output is enabled.
- Command line: `--innodb-ft-enable-diag-print={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_ft_enable_stopword`

- Description: If set to `1`, the default, a set of stopwords is associated with an InnoDB FULLTEXT index when it is created. The stopword list comes from the table set by the session variable innodb_ft_user_stopword_table, if set, otherwise the global variable innodb_ft_server_stopword_table, if that is set, or the built-in list if neither variable is set.
- Command line: `--innodb-ft-enable-stopword={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`innodb_ft_max_token_size`

- Description: Maximum length of words stored in an InnoDB FULLTEXT index. A larger limit will increase the size of the index, slowing down queries, but permit longer words to be searched for. In most normal situations, longer words are unlikely search terms.
- Command line: `--innodb-ft-max-token-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `84`
- Range: `10` to `84`

`innodb_ft_min_token_size`

- Description: Minimum length of words stored in an InnoDB FULLTEXT index. A smaller limit will increase the size of the index, slowing down queries, but permit shorter words to be searched for. For data stored in a Chinese, Japanese or Korean character set, a value of 1 should be specified to preserve functionality.
- Command line: `--innodb-ft-min-token-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `3`
- Range: `1` to `16`

`innodb_ft_num_word_optimize`

- Description: Number of words processed during each OPTIMIZE TABLE on an InnoDB FULLTEXT index. To ensure all changes are incorporated, multiple OPTIMIZE TABLE statements could be run in case of a substantial change to the index.
- Command line: `--innodb-ft-num-word-optimize=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2000`
- Range: `1000` to `10000`

`innodb_ft_result_cache_limit`

- Description: Limit in bytes of the InnoDB FULLTEXT index query result cache per fulltext query. The latter stages of the full-text search are handled in memory, and limiting this prevents excess memory usage. If the limit is exceeded, the query returns an error.
- Command line: `--innodb-ft-result-cache-limit=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2000000000`
- Range: `1000000` to `18446744073709551615`

`innodb_ft_server_stopword_table`

- Description: Table name containing a list of stopwords to ignore when creating an InnoDB FULLTEXT index, in the format db_name/table_name. The specified table must exist before this option is set, and must be an InnoDB table with a single column, a VARCHAR named VALUE. See also innodb_ft_enable_stopword.
- Command line: `--innodb-ft-server-stopword-table=db_name/table_name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: Empty

`innodb_ft_sort_pll_degree`

- Description: Number of parallel threads used when building an InnoDB FULLTEXT index. See also innodb_sort_buffer_size.
- Command line: `--innodb-ft-sort-pll-degree=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `2`
- Range: `1` to `32`

`innodb_ft_total_cache_size`

- Description:Total memory allocated for the cache for all InnoDB FULLTEXT index tables. A force sync is triggered if this limit is exceeded.
- Command line: `--innodb-ft-total-cache-size=#`

- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `640000000`
- Range: `32000000` to `1600000000`
- Introduced: MariaDB 10.0.9

`innodb_ft_user_stopword_table`

- Description: Table name containing a list of stopwords to ignore when creating an InnoDB FULLTEXT index, in the format db_name/table_name. The specified table must exist before this option is set, and must be an InnoDB table with a single column, a VARCHAR named VALUE. See also innodb_ft_enable_stopword.
- Command line: `--innodb-ft-user-stopword-table=db_name/table_name`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: Empty

`innodb_ibuf_accel_rate`

- Description: Allows the insert buffer activity to be adjusted. The following formula is used: [real activity] = [default activity] * (innodb_io_capacity/100) * (innodb_ibuf_accel_rate/100). As `innodb_ibuf_accel_rate` is increased from its default value of `100`, the lowest setting, insert buffer activity is increased. See also innodb_io_capacity. This Percona XtraDB variable has not been ported to XtraDB 5.6.
- Command line: `innodb-ibuf-accel-rate=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `100` to `999999999`
- Removed: MariaDB 10.0

`innodb_ibuf_active_contract`

- Description: Specifies whether the insert buffer can be processed before it's full. If set to `0`, the standard InnoDB method is used, and the buffer is not processed until it's full. If set to `1`, the default, the insert buffer can be processed before it is full. This Percona XtraDB variable has not been ported to XtraDB 5.6.
- Command line: `innodb-ibuf-active-contract=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `1`
- Removed: MariaDB 10.0

`innodb_ibuf_max_size`

- Description: Maximum size in bytes of the insert buffer. Defaults to half the size of the buffer pool so you may want to reduce if you have a very large buffer pool. If set to `0`, the insert buffer is disabled, which will cause all

secondary index updates to be performed synchronously, usually at a cost to performance. This Percona XtraDB variable has not been ported to XtraDB 5.6.

- Command line: `innodb-ibuf-max-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: 1/2 the size of the InnoDB buffer pool
- Range: `0` to 1/2 the size of the InnoDB buffer pool
- Removed: MariaDB 10.0

`innodb_idle_flush_pct`

- Description: Up to what percentage of dirty pages should be flushed when innodb finds it has spare resources to do so. Has had no effect since merging InnoDB 5.7 from mysql-5.7.9 (MariaDB 10.2.2). Deprecated in MariaDB 10.2.37, MariaDB 10.3.28, MariaDB 10.4.18 and removed in MariaDB 10.5.9.
- Command line: `--innodb-idle-flush-pct=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `0` to `100`
- Deprecated: MariaDB 10.2.37, MariaDB 10.3.28, MariaDB 10.4.18
- Removed: MariaDB 10.5.9

`innodb_immediate_scrub_data_uncompressed`

- Description: Enable scrubbing of data. See Data Scrubbing.
- Command line: `--innodb-immediate-scrub-data-uncompressed={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_import_table_from_xtrabackup`

- Description: If set to `1`, permits importing of .ibd files exported with the XtraBackup –export option. Previously named `innodb_expand_import`. Removed in MariaDB 10.0/XtraDB 5.6 and replaced with MySQL 5.6's transportable tablespaces.
- Command line: `innodb-import-table-from-xtrabackup=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `1`
- Removed: MariaDB 10.0

`innodb_instant_alter_column_allowed`

- Description:

    - If a table is altered using `ALGORITHM=INSTANT`, it can force the table to use a non-canonical format: A hidden metadata record at the start of the clustered index is used to store each column's `DEFAULT` value. This makes it possible to add new columns that have default values without rebuilding the table. Starting with MariaDB 10.4, a `BLOB` in the hidden metadata record is used to store column mappings. This makes it possible to drop or reorder columns without rebuilding the table. This also makes it possible to add columns to any position or drop columns from any position in the table without rebuilding the table. If a column is dropped without rebuilding the table, old records will contain garbage in that column's former position, and new records are written with `NULL` values, empty strings, or dummy values.
    - This is generally not a problem. However, there may be cases where you want to avoid putting a table into this format. For example, to ensure that future `UPDATE` operations after an `ADD COLUMN` are performed in-place, to reduce write amplification. (Instantly added columns are essentially always variable-length.) Also avoid bugs similar to MDEV-19916, or to be able to export tables to older versions of the server.
    - This variable has been introduced as a result, with the following values:
    - `never` (0): Do not allow instant add/drop/reorder, to maintain format compatibility with MariaDB 10.x and MySQL 5.x. If the table (or partition) is not in the canonical format, then any ALTER TABLE (even one that does not involve instant column operations) will force a table rebuild.
    - `add_last` (1, default in 10.3): Store a hidden metadata record that allows columns to be appended to the table instantly (MDEV-11369).
      In 10.4 or later, if the table (or partition) is not in this format, then any ALTER TABLE (even one that does not involve column changes) will force a table rebuild.
    - `add_drop_reorder` (2, default): From MariaDB 10.4 only. Like 'add_last', but allow the metadata record to store a column map, to support instant add/drop/reorder of columns.
- Command line: `--innodb-instant-alter-column-allowed=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Valid Values:
    - <= MariaDB 10.3: `never`, `add_last`
    - = MariaDB 10.4: `never`, `add_last`, `add_drop_reorder`
- Default Value:
    - <= MariaDB 10.3: `add_last`
    - = MariaDB 10.4: `add_drop_reorder`
- Introduced: MariaDB 10.3.23, MariaDB 10.4.13, MariaDB 10.5.3

`innodb_instrument_semaphores`

- Description: Enable semaphore request instrumentation. This could have some effect on performance but allows better information on long semaphore wait problems.
- Command line: `--innodb-instrument-semaphores={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.2.5 (treated as if `OFF`)
- Removed: MariaDB 10.3.0

`innodb_io_capacity`

- Description: Limit on I/O activity for InnoDB background tasks, including merging data from the insert buffer and flushing pages. Should be set to around the number of I/O operations per second that system can handle, based on the type of drive/s being used. You can also set it higher when the server starts to help with the extra workload at that time, and then reduce for normal use. Ideally, opt for a lower setting, as at higher value data is removed from the buffers too quickly, reducing the effectiveness of caching. See also innodb_flush_sync.
  - See InnoDB Page Flushing: Configuring the InnoDB I/O Capacity for more information.
- Command line: `--innodb-io-capacity=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `200`
- Range: `100` to `4294967295` (2^{32}-1)

`innodb_io_capacity_max`

- Description: Upper limit to which InnoDB can extend innodb_io_capacity in case of emergency. See InnoDB Page Flushing: Configuring the InnoDB I/O Capacity for more information.
- Command line: `--innodb-io-capacity-max=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2000` or twice innodb_io_capacity, whichever is higher.
- Range : `100` to `4294967295` (2^{32}-1)

`innodb_kill_idle_transaction`

- Description: Time in seconds before killing an idle XtraDB transaction. If set to `0` (the default), the feature is disabled. Used to prevent accidental user locks. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `9223372036854775807`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_large_prefix`

- Description: If set to `1`, tables that use specific row formats are permitted to have index key prefixes up to 3072 bytes (for 16k pages, smaller otherwise). If not set, the limit is 767 bytes.
  - This applies to the DYNAMIC and COMPRESSED row formats.
  - Removed in 10.3.1 and restored as a deprecated and unused variable in 10.4.3 for compatibility purposes.
- Command line: `--innodb-large-prefix`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`

- Default Value:
  - `ON`
- Deprecated: MariaDB 10.2
- Removed: MariaDB 10.3.1
- Re-introduced: MariaDB 10.4.3 (for compatibility purposes)
- Removed: MariaDB 10.6.0

`innodb_lazy_drop_table`

- Description: Deprecated and removed in XtraDB 5.6. DROP TABLE processing can take a long time when innodb_file_per_table is set to 1 and there's a large buffer pool. If `innodb_lazy_drop_table` is set to `1` (`0` is default), XtraDB attempts to optimize DROP TABLE processing by deferring the dropping of related pages from the buffer pool until there is time, only initially marking them.
- Command line: `innodb-lazy-drop-table={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`
- Deprecated: XtraDB 5.5.30-30.2
- Removed: MariaDB 10.0.0

`innodb_lock_schedule_algorithm`

- Description: Removed in MariaDB 10.6.0 due to problems with the VATS implementation (MDEV-16664). Specifies the algorithm that InnoDB uses to decide which of the waiting transactions should be granted the lock once it has been released. The possible values are: `FCFS` (First-Come-First-Served) where locks are granted in the order they appear in the lock queue and `VATS` (Variance-Aware-Transaction-Scheduling) where locks are granted based on the Eldest-Transaction-First heuristic. Note that `VATS` should not be used with Galera, and InnoDB will refuse to start if `VATS` is used with Galera. It is also not recommended to set to `VATS` even in the general case (MDEV-16664). From MariaDB 10.2.12, the value was changed to `FCFS` and a warning produced when using Galera.
- Command line: `--innodb-lock-schedule-algorithm=#`
- Scope: Global
- Dynamic: No (>= MariaDB 10.2.12, MariaDB 10.1.30), Yes (<= MariaDB 10.2.11, MariaDB 10.1.29)
- Data Type: `enum`
- Valid Values: `FCFS`, `VATS`
- Default Value: `FCFS` (MariaDB 10.3.9, MariaDB 10.2.17), `VATS` (MariaDB 10.2.3), `FCFS` (MariaDB 10.1)
- Deprecated: MariaDB 10.5.7, MariaDB 10.4.16, MariaDB 10.3.26, MariaDB 10.2.35
- Removed: MariaDB 10.6.0

`innodb_lock_wait_timeout`

- Description: Time in seconds that an InnoDB transaction waits for an InnoDB record lock (or table lock) before giving up with the error `ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction`. When this occurs, the statement (not transaction) is rolled back. The whole transaction can be rolled back if the innodb_rollback_on_timeout option is used. Increase this for data warehousing applications or where other long-running operations are common, or decrease for OLTP and other highly interactive applications. This setting does not apply to deadlocks, which InnoDB detects immediately, rolling back a deadlocked transaction. `0`

means no wait. See WAIT and NOWAIT. Setting to `100000000` or more (from MariaDB 10.6.3, `100000000` is the maximum) means the timeout is infinite.

- Command line: `--innodb-lock-wait-timeout=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `INT UNSIGNED` (>= MariaDB 10.6.3), `BIGINT UNSIGNED` (<= MariaDB 10.6.2)
- Default Value: `50`
- Range:
  - `0` to `100000000` (>= MariaDB 10.6.3)
  - `0` to `1073741824` (>= MariaDB 10.3 to <= MariaDB 10.6.2)

`innodb_locking_fake_changes`

- Description: From MariaDB 5.5 to MariaDB 10.1, XtraDB-only option that if set to `OFF`, fake transactions (see innodb_fake_changes) don't take row locks. This is an experimental feature to attempt to deal with drawbacks in fake changes blocking real locks. It is not safe for use in all environments. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `--innodb-locking-fake-changes`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_locks_unsafe_for_binlog`

- Description: Set to `0` by default, in which case XtraDB/InnoDB uses gap locking. If set to `1`, gap locking is disabled for searches and index scans. Deprecated in MariaDB 10.0, and removed in MariaDB 10.5, use READ COMMITTED transaction isolation level instead.
- Command line: `--innodb-locks-unsafe-for-binlog`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.0
- Removed: MariaDB 10.5.0

`innodb_log_arch_dir`

- Description: The directory for XtraDB redo log archiving. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `--innodb-log-arch-dir=name`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `./`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_log_arch_expire_sec`

- Description: Time in seconds since the last change after which the archived XtraDB redo log should be deleted. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `--innodb-log-arch-expire-sec=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_log_archive`

- Description: Controls the InnoDB log archiving format. When set to `ON`, the InnoDB write-ahead log is written to a sequence of files named `ib_` *`lsn`* `.log` instead of the circular ring buffer (`ib_logfile0`). Each file is pre-allocated to `innodb_log_file_size`; when one fills up, the server creates and pre-allocates the next file. After the first checkpoint completes in a new file, the previous file is marked read-only, signaling that it can be moved to long-term storage. This format makes a continuous log history available, which is necessary for point-in-time recovery and incremental backups.
- Details:
  - File Naming: Archive files use the naming convention `ib_` *`lsn`* `.log`, where *lsn* is a 16-character hexadecimal representation of the Log Sequence Number (LSN) at offset 12288 (0x3000) of the file.
  - Log Resizing: When archiving is enabled, changes to `innodb_log_file_size` take effect when the current log file is filled and a new file is allocated. This differs from the standard resizing logic used when `innodb_log_archive` is `OFF`.
  - Encryption: While `innodb_log_archive` is `ON`, the value of `innodb_encrypt_log` and related encryption parameters cannot be changed. To change encryption, set `innodb_log_archive=OFF` and restart the server — this permanently discards the archived log history.
  - Startup: With `innodb_log_archive=ON`, the server refuses to start if `ib_logfile0` exists in the data directory.
  - Data Dictionary: This feature tracks InnoDB changes only. It does not cover `.frm` files or other non-InnoDB metadata.
  - Backup tooling: No shipped tool yet generates or restores backups in this format. `mariadb-backup` only supports the legacy `ib_logfile0` format and fails when the server is running with `innodb_log_archive=ON`.
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 13.0

`innodb_log_block_size`

- Description: Size in bytes of the XtraDB redo log records. Generally `512`, the default, or `4096`, are the only two useful values. If the server is restarted and this value is changed, all old log files need to be removed. Should be set to `4096` for SSD cards or if innodb_flush_method is set to `ALL_O_DIRECT` on ext4 filesystems. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-log-block-size=#`

- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `512`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_log_buffer_size`

- Description: Size in bytes of the buffer for writing InnoDB redo log files to disk. Increasing this means larger transactions can run without needing to perform disk I/O before committing.
- Command line: `--innodb-log-buffer-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `16777216` (16MB)
- Range: `262144` to `2147479552` (256KB to 2GB - 4K) (>= MariaDB 10.11.8)
- Range: `262144` to `18446744073709551615` (<= MariaDB 10.11.7) - limit to the above maximum because this is an operating system limit.
- Block size: `4096`

`innodb_log_checkpoint_now`

- Description: Write back dirty pages from the buffer pool and update the log checkpoint. Prior to MariaDB 10.11.12, MariaDB 11.4.6, MariaDB 11.8.2 was only available in debug builds. Introduced in order to force checkpoints before a backup, allowing mariadb-backup to create much smaller incremental backups. However, this comes at the cost of heavy I/O usage and it is now disabled by default.
- Command line: `--innodb-log-checkpoint{=1|0}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.11.12, MariaDB 11.4.6, MariaDB 11.8.2

`innodb_log_checksum_algorithm`

- Description: Experimental feature (as of MariaDB 10.0.9), this variable specifies how to generate and verify XtraDB redo log checksums. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
  - `none` - No checksum. A constant value is instead written to logs, and no checksum validation is performed.
  - `innodb` - The default, and the original InnoDB algorithm. This is inefficient, but compatible with all MySQL, MariaDB and Percona versions that don't support other checksum algorithms.
  - `crc32` - CRC32© is used for log block checksums, which also permits recent CPUs to use hardware acceleration (on SSE4.2 x86 machines and Power8 or later) for the checksums.
  - `strict_*` - Whether or not to accept checksums from other algorithms. If strict mode is used, checksums blocks are considered corrupt if they don't match the specified algorithm. Normally they are considered corrupt only if no other algorithm matches.
- Command line: `innodb-log-checksum-algorithm=value`
- Scope: Global

- Dynamic: Yes
- Data Type: `enum`
- Default Value:
  - `deprecated` (>= MariaDB 10.2.6)
  - `innodb` (<= MariaDB 10.1)
- Valid Values:
  - `deprecated`, `innodb`, `none`, `crc32`, `strict_none`, `strict_innodb`, `strict_crc32` (>= MariaDB 10.2.6)
  - `innodb`, `none`, `crc32`, `strict_none`, `strict_innodb`, `strict_crc32` (<= MariaDB 10.1)
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_log_checksums`

- Description: If set to `1`, the CRC32C for Innodb or `innodb_log_checksum_algorithm` for XtraDB algorithm is used for InnoDB redo log pages. If disabled, the checksum field contents are ignored. From MariaDB 10.5.0, the variable is deprecated, and checksums are always calculated, as previously, the InnoDB redo log used the slow innodb algorithm, but with hardware or SIMD assisted CRC-32C computation being available, there is no reason to allow checksums to be disabled on the redo log.
- Command line: `innodb-log-checksums={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 10.5.0
- Removed: MariaDB 10.6.0

`innodb_log_compressed_pages`

- Description: Whether or not images of recompressed pages are stored in the InnoDB redo log. Deprecated and ignored from MariaDB 10.5.3.
- Command line: `--innodb-log-compressed-pages={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value:
  - `ON`
- Deprecated: MariaDB 10.5.3
- Removed: MariaDB 10.6.0

`innodb_log_file_buffering`

- Description: Whether the file system cache for ib_logfile0 is enabled. In MariaDB 10.8.3, MariaDB disabled the file system cache on the InnoDB write-ahead log file (ib_logfile0) by default on Linux. With innodb_flush_trx_log_at_commit=2 in particular, writing to the log via the file system cache typically improves throughput, especially on slow storage or at a small number of concurrent transactions. For other values of innodb_flush_log_at_trx_commit, direct writes were observed to be mostly but not always faster. Whether it pays off to disable the file system cache on the log may depend on the type of storage, the workload, and the operating system kernel version. If the server is started up with innodb_flush_log_at_trx_commit=2, the value are changed to `ON`. Will be set to `OFF` if innodb_flush_method is set to `O_DSYNC`. On Linux, when the physical block

size cannot be determined to be a power of 2 between 64 and 4096 bytes, the file system cache cannot be disabled, and innodb_log_file_buffering=ON cannot be changed. Linux and Windows only.

- Command line: `--innodb-log-file-buffering={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.8.4, MariaDB 10.9.2

> **Note:**
>
> A change requested with `SET GLOBAL innodb_log_file_buffering` takes effect only when InnoDB is holding a single log file open, which is the case most of the time. It is ignored while a `SET GLOBAL innodb_log_file_size` resize is in progress on a circular-format log (`ib_logfile0`). From MariaDB 13.0 (MDEV-39862), it is also ignored while a log file switch is in progress with `innodb_log_archive` set to `ON` , because InnoDB then has more than one log file open; when this occurs, and how long it lasts, is hard to predict, as it depends on `innodb_log_file_size` and the workload. The setting also has no effect when the log is mapped to persistent memory (PMEM).
>
> Because the request can be silently ignored, confirm that it took effect by reading the value back:
>
> ```
> SELECT @@GLOBAL.innodb_log_file_buffering;
> ```

`innodb_log_file_mmap`

- Description: Whether ib_logfile0 resides in persistent memory or should initially be memory-mapped. When using the default innodb_log_buffer_size=2m, mariadb-backup –backup would spend a lot of time re-reading and re-parsing the log. For reading the log file during mariadb-backup –backup, it is beneficial to memory-map the entire ib_logfile0 to the address space (typically 48 bits or 256 TiB) and read it from there, both during –backup and –prepare. OFF by default on most platforms, to avoid aggressive read-ahead of the entire ib_logfile0 in when only a tiny portion would be accessed. On Linux and FreeBSD the default is innodb_log_file_mmap=ON, because those platforms define a specific mmap(2) option for enabling such read-ahead and therefore it can be assumed that the default wouldbe on-demand paging. This parameter will only have impact on the initial InnoDB startup and recovery. Any writes to the log will use regular I/O, except when the ib_logfile0 is stored in a specially configured file system that is backed by persistent memory (Linux “mount -o dax”).
- Command line: `--innodb-log-file-mmap{=0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON` (Linux, FreeBSD), `OFF` (Other platforms)
- Introduced: MariaDB 10.11.10, MariaDB 11.2.6, MariaDB 11.4.4, MariaDB 11.6.2, MariaDB 11.7.1

`innodb_log_file_size`

- Description: Size in bytes of each InnoDB redo log file in the log group. The combined size can be no more than 512GB. Larger values mean less disk I/O due to less flushing checkpoint activity, but also slower recovery from a crash. In MariaDB 10.5, crash recovery has been improved and shouldn’t run out of memory, so the default has been increased. It can safely be set higher to reduce checkpoint flushing, even larger than innodb_buffer_pool_size.From MariaDB 10.9 the variable is dynamic, and the server no longer needs to be restarted for the resizing to take place. Unless the log is located in a persistent memory file system (PMEM), an attempt to SET GLOBAL innodb_log_file_size to less than innodb_log_buffer_size are refused. Log resizing can be aborted by killing the connection that is executing the SET GLOBAL statement.

- Command line: `--innodb-log-file-size=#`
- Scope: Global
- Dynamic: Yes (>= MariaDB 10.9), No (<= MariaDB 10.8)
- Data Type: `numeric`
- Default Value: `100663296` (96MB) (>= MariaDB 10.5), `50331648` (48MB) (<= MariaDB 10.4)
- Range:
  - = MariaDB 10.8.3: `4194304` to `512GB` (4MB to 512GB)
  - <= MariaDB 10.8.2: `1048576` to `512GB` (1MB to 512GB)
- Block size: `4096`

`innodb_log_file_write_through`

- Description: Whether each write to ib_logfile0 is write through (disabling any caching, as in O_SYNC or O_DSYNC). Set to `OFF` by default, are set to `ON` if innodb_flush_method is set to `O_DSYNC`. On systems that support FUA it may make sense to enable write-through, to avoid extra system calls. See InnoDB Flush Method for a discussion of FUA and write-through behavior.
- Command line: `--innodb-log-file-write-through={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 11.0.0

> **Note:**
>
> A change requested with `SET GLOBAL innodb_log_file_write_through` takes effect only when InnoDB is holding a single log file open, which is the case most of the time. It is ignored while a `SET GLOBAL innodb_log_file_size` resize is in progress on a circular-format log (`ib_logfile0`). From MariaDB 13.0 (MDEV-39862), it is also ignored while a log file switch is in progress with `innodb_log_archive` set to `ON`, because InnoDB then has more than one log file open; when this occurs, and how long it lasts, is hard to predict, as it depends on `innodb_log_file_size` and the workload. The setting also has no effect when the log is mapped to persistent memory (PMEM).
>
> Because the request can be silently ignored, confirm that it took effect by reading the value back:
>
> ```
> SELECT @@GLOBAL.innodb_log_file_write_through;
> ```

`innodb_log_files_in_group`

- Description: Number of physical files in the InnoDB redo log. Deprecated and ignored from MariaDB 10.5.2
- Command line: `--innodb-log-files-in-group=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1` (>= MariaDB 10.5), `2` (<= MariaDB 10.4)
- Range: `1` to `100` (>= MariaDB 10.2.4)
- Deprecated: MariaDB 10.5.2
- Removed: MariaDB 10.6.0

`innodb_log_group_home_dir`

- Description: Path to the InnoDB redo log files. If none is specified, innodb_log_files_in_group files named ib_logfile0 and so on, with a size of innodb_log_file_size are created in the data directory.
- Command line: `--innodb-log-group-home-dir=path`
- Scope: Global
- Dynamic: No
- Data Type: `directory name`

`innodb_log_optimize_ddl`

- Description: Whether InnoDB redo log activity should be reduced when natively creating indexes or rebuilding tables. Reduced logging requires additional page flushing and interferes with mariadb-backup. Enabling this may slow down backup and cause delay due to page flushing. Deprecated and ignored from MariaDB 10.5.1. Deprecated (but not ignored) from MariaDB 10.4.16, MariaDB 10.3.26 and MariaDB 10.2.35.
- Command line: `--innodb-log-optimize-ddl={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value:
  - `OFF` (>= MariaDB 10.5.1, MariaDB 10.4.16, MariaDB 10.3.26, MariaDB 10.2.35)
  - `ON` (<= MariaDB 10.5.0, MariaDB 10.4.15, MariaDB 10.3.25, MariaDB 10.2.34)
- Introduced: MariaDB 10.2.17, MariaDB 10.3.9
- Deprecated: MariaDB 10.5.1, MariaDB 10.4.16, MariaDB 10.3.26, MariaDB 10.2.35
- Removed: MariaDB 10.6.0

`innodb_log_recovery_start`

- Description: Specifies the Log Sequence Number (LSN) at which the recovery process begins. Only meaningful when recovering a backup that contains log files in the `innodb_log_archive=ON` format.
- Usage: Set this variable to limit the scope of a recovery operation. The server expects to find an optional sequence of `FILE_MODIFY` records followed by a `FILE_CHECKPOINT` record at the specified *lsn*. This is typically used when restoring an incremental backup, by setting the variable to the end LSN of the previous restore.
- Special value: `0` (the default) means start recovery from the latest completed checkpoint. The latest checkpoint is guaranteed to live in one of the last two `ib_`*`lsn`*`.log` files in the data directory, typically the last one.
- Property: Set at startup
- Data Type: `numeric` (64-bit unsigned integer)
- Default Value: `0`
- Introduced: MariaDB 13.0

`innodb_log_recovery_target`

- Description: Specifies the target Log Sequence Number (LSN) at which the recovery process ends. Only meaningful when recovering a backup that contains log files in the `innodb_log_archive=ON` format.
- Usage: Use this variable to define a recovery point objective. The server replays archived logs up to the specified *lsn* and stops. When this variable is non-zero, all persistent InnoDB tables become read-only and no log writes are allowed.
- Special value: `0` (the default) performs an unlimited recovery — the server replays the log to its end.
- Property: Set at startup
- Data Type: `numeric` (64-bit unsigned integer)

- Default Value: `0`
- Introduced: MariaDB 13.0

> **Warning:**
>
> **No data file may carry an LSN newer than `innodb_log_recovery_target`.** Crash recovery's role is to bring every database page to the same logical point in time (the same LSN). If any data file is newer than the target, recovery completes in an inconsistent state where some pages carry an LSN past the requested target — that is, the database is corrupted. The server cannot validate every such impossible target, and the resulting corruption may not be detected until the affected pages are accessed (see MDEV-34830). Choose a target that lies at or beyond the highest page LSN you intend to retain.
>
> If you set a target that is unreachable in the other direction (for example, lower than the current checkpoint), the server terminates with an error message containing the available LSN range.

`innodb_log_spin_wait_delay`

- Description: Delay between log buffer spin lock polls (0 to use a blocking latch). Specifically, enables a spin lock that will execute that many MY_RELAX_CPU() operations (such as the x86 PAUSE instruction) between successive attempts of acquiring the spin lock. On some hardware with certain workloads (observed on write intensive workloads on NUMA[12] systems), the default setting results in a significant amount of time being spent in native_queued_spin_lock_slowpath() in the Linux kernel, plus context switching between user and kernel address space, in which case changing from the default (for example, setting to `50`), may result in a performance improvement.
- Command line: `--innodb-log-spin-wait-delay=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `6000`
- Introduced: MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5, MariaDB 11.2.4, MariaDB 11.4.2, MariaDB 11.5.1

`innodb_log_write_ahead_size`

- Description: InnoDB redo log write ahead unit size to avoid read-on-write. Should match the OS cache block IO size. Removed in MariaDB 10.8, and instead on Linux and Windows, the physical block size of the underlying storage is detected and used. Reintroduced in MariaDB 10.11.9 and later versions. On Linux and Windows, the default or the specified innodb_log_write_ahead_size are automatically adjusted to not be less than the physical block size (if it can be determined).
- Command line: `--innodb-log-write-ahead-size=#`
- Scope: Global
- Dynamic: No (>= MariaDB 10.11.9), Yes (<= MariaDB 10.7)
- Data Type: `numeric`
- Default Value: `512` (>= MariaDB 10.11.9), `8192` (<= MariaDB 10.7)
- Range:
  - `512` to `4096` (>= MariaDB 10.11.9)
  - `512` to innodb_page_size (<= MariaDB 10.7)
- Removed: MariaDB 10.8
- Re-introduced: MariaDB 10.11.9, MariaDB 11.1.6, MariaDB 11.2.5, MariaDB 11.4.3, MariaDB 11.5.2

`innodb_lru_flush_size`

- Description: Number of pages to flush on LRU eviction. Changes in MariaDB 10.6.18, MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5, MariaDB 11.2.4 and MariaDB 11.4.2 made this setting superfluous, and it is no longer used.
- Command line: `--innodb-lru-flush-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `0` (>= MariaDB 10.6.20, MariaDB 10.11.10, MariaDB 11.2.6, MariaDB 11.4.4. MariaDB 11.6.1, MariaDB 11.7.1)
  - `32` (<= MariaDB 10.6.19, MariaDB 10.11.9, MariaDB 11.2.5, MariaDB 11.4.3)
- Range: `1` to `18446744073709551615`
- Introduced: MariaDB 10.5.7
- Deprecated: MariaDB 10.6.20, MariaDB 10.11.10, MariaDB 11.2.6 and MariaDB 11.4.4. MariaDB 11.6.1, MariaDB 11.7.1

`innodb_lru_scan_depth`

- Description: Specifies how far down the buffer pool least-recently used (LRU) list the cleaning thread should look for dirty pages to flush. This process is performed once a second. In an I/O intensive-workload, can be increased if there is spare I/O capacity, or decreased if in a write-intensive workload with little spare I/O capacity.
  - See InnoDB Page Flushing for more information.
- Command line: `--innodb-lru-scan-depth=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `1536` (>= MariaDB 10.5.7)
  - `1024` (<= MariaDB 10.5.6)
- Range - 32bit: `100` to `2^32-1`
- Range - 64bit: `100` to `2^64-1`

`innodb_max_bitmap_file_size`

- Description: Limit in bytes of the changed page bitmap files. For faster incremental backup with Xtrabackup, XtraDB tracks pages with changes written to them according to the XtraDB redo log and writes the information to special changed page bitmap files. These files are rotated when the server restarts or when this limit is reached. XtraDB only. See also innodb_track_changed_pages and innodb_max_changed_pages.
  - Deprecated and ignored in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-max-bitmap-file-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4096` (4KB)
- Range: `4096` (4KB) to `18446744073709551615` (16EB)

- Deprecated: MariaDB 10.2.6

`innodb_max_changed_pages`

- Description: Limit to the number of changed page bitmap files (stored in the Information Schema INNODB_CHANGED_PAGES table). Zero is unlimited. See innodb_max_bitmap_file_size and innodb_track_changed_pages. Previously named `innodb_changed_pages_limit`. XtraDB only.
  - Deprecated and ignored in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-max-changed-pages=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1000000`
- Range: `0` to `18446744073709551615`
- Deprecated: MariaDB 10.2.6

`innodb_max_dirty_pages_pct`

- Description: Maximum percentage of unwritten (dirty) pages in the buffer pool.
  - See InnoDB Page Flushing for more information.
- Command line: `--innodb-max-dirty-pages-pct=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `90.000000` (>= MariaDB 10.5.7)
  - `75.000000` (<= MariaDB 10.5.6)
- Range: `0` to `99.999`

`innodb_max_dirty_pages_pct_lwm`

- Description: Low water mark percentage of dirty pages that will enable preflushing to lower the dirty page ratio. The value 0 (default) means 'refer to innodb_max_dirty_pages_pct'. (Note that 0 meant 0 in 10.5.7 to 10.5.8, but was then reverted back to "same as innodb_max_dirty_pages_pct" again in 10.5.9)
  - See InnoDB Page Flushing for more information.
- Command line: `--innodb-max-dirty-pages-pct-lwm=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `0` (>= MariaDB 10.2.1)
  - `0.001` (<= MariaDB 10.2.0)
- Range: `0` to `99.999`

`innodb_max_purge_lag`

- Description: When purge operations are lagging on a busy server, setting innodb_max_purge_lag can help. By default set to `0`, no lag, the figure is used to calculate a time lag for each INSERT, UPDATE, and DELETE when

the system is lagging. InnoDB keeps a list of transactions with delete-marked index records due to UPDATE and DELETE statements. The length of this list is `purge_lag`, and the calculation, performed every ten seconds, is as follows: ((purge_lag/innodb_max_purge_lag)×10)–5 microseconds.

- Command line: `--innodb-max-purge-lag=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4294967295`

`innodb_max_purge_lag_delay`

- Description: Maximum delay in milliseconds imposed by the innodb_max_purge_lag setting. If set to `0`, the default, there is no maximum.
- Command line: `--innodb-max-purge-lag-delay=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`

`innodb_max_purge_lag_wait`

- Description: Wait until History list length is below the specified limit.
- Command line: `--innodb-max-purge-wait=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4294967295`
- Range: `0` to `4294967295`
- Introduced: MariaDB 10.5.7, MariaDB 10.4.16, MariaDB 10.3.26, MariaDB 10.2.35

`innodb_max_undo_log_size`

- Description: If an undo tablespace is larger than this, it is marked for truncation if innodb_undo_log_truncate is set.
- Command line: `--innodb-max-undo-log-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `10485760`
- Range: `10485760` to `18446744073709551615`

`innodb_merge_sort_block_size`

- Description: Size in bytes of the block used for merge sorting in fast index creation. Replaced in MariaDB 10.0/XtraDB 5.6 by innodb_sort_buffer_size.
- Command line: `innodb-merge-sort-block-size=#`
- Scope: Global

- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1048576` (1M)
- Range: `1048576` (1M) to `1073741824` (1G)
- Removed: MariaDB 10.0 - replaced by innodb_sort_buffer_size

`innodb_mirrored_log_groups`

- Description: Unused. Restored as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Deprecated: MariaDB 10.0
- Removed: MariaDB 10.2.2 - MariaDB 10.2.5

`innodb_mtflush_threads`

- Description: Sets the number of threads to use in Multi-Threaded Flush operations. For more information, see Fusion-io Multi-threaded Flush.
  - InnoDB's multi-thread flush feature was deprecated in MariaDB 10.2.9 and removed from MariaDB 10.3.2. In later versions of MariaDB, use innodb_page_cleaners system variable instead.
  - See InnoDB Page Flushing: Page Flushing with Multi-threaded Flush Threads for more information.
- Command line: `--innodb-mtflush-threads=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `8`
- Range: `1` to `64`
- Deprecated: MariaDB 10.2.9
- Removed: MariaDB 10.3.2

`innodb_monitor_disable`

- Description: Disables the specified counters in the INFORMATION_SCHEMA.INNODB_METRICS table.
- Command line: `--innodb-monitor-disable=string`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`

`innodb_monitor_enable`

- Description: Enables the specified counters in the INFORMATION_SCHEMA.INNODB_METRICS table.
- Command line: `--innodb-monitor-enable=string`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`

`innodb_monitor_reset`

- Description: Resets the count value of the specified counters in the INFORMATION_SCHEMA.INNODB_METRICS table to zero.
- Command line: `--innodb-monitor-reset=string`

- Scope: Global
- Dynamic: Yes
- Data Type: `string`

`innodb_monitor_reset_all`

- Description: Resets all values for the specified counters in the INFORMATION_SCHEMA.INNODB_METRICS table.
- Command line: `---innodb-monitor-reset-all=string`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`

`innodb_numa_interleave`

- Description: Whether or not to use the NUMA[13] interleave memory policy to allocate the InnoDB buffer pool.
- Command line: `innodb-numa-interleave={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.2.23, MariaDB 10.3.14, MariaDB 10.4.4

`innodb_old_blocks_pct`

- Description: Percentage of the buffer pool to use for the old block sublist.
- Command line: `--innodb-old-blocks-pct=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `37`
- Range: `5` to `95`

`innodb_old_blocks_time`

- Description: Time in milliseconds an inserted block must stay in the old sublist after its first access before it can be moved to the new sublist. '0' means "no delay". Setting a non-zero value can help prevent full table scans clogging the buffer pool. See also innodb_old_blocks_pct.
- Command line: `--innodb-old-blocks-time=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1000`
- Range: `0` to `2^32-1`

`innodb_online_alter_log_max_size`

- Description: The maximum size for temporary log files during online DDL (data and index structure changes). The temporary log file is used for each table being altered, or index being created, to store data changes to the table while the process is underway. The table is extended by innodb_sort_buffer_size up to the limit set by this

variable. If this limit is exceeded, the online DDL operation fails and all uncommitted changes are rolled back. A lower value reduces the time a table could lock at the end of the operation to apply all the log's changes, but also increases the chance of the online DDL changes failing.

- Command line: `--innodb-online-alter-log-max-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `134217728`
- Range: `65536` to `2^64-1`

`innodb_open_files`

- Description: Maximum .ibd files MariaDB can have open at the same time. Only applies to systems with multiple XtraDB/InnoDB tablespaces, and is separate to the table cache and open_files_limit. The default, if innodb_file_per_table is disabled, is 300 or the value of table_open_cache, whichever is higher. It will also auto-size up to the default value if it is set to a value less than `10`.
- Command line: `--innodb-open-files=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `autosized`
- Range: `10` to `4294967295`

`innodb_optimize_fulltext_only`

- Description: When set to `1` (`0` is default), OPTIMIZE TABLE will only process InnoDB FULLTEXT index data. Only intended for use during fulltext index maintenance.
- Command line: `--innodb-optimize-fulltext-only={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_page_cleaners`

- Description: Number of page cleaner threads. The default is `4`, but the value are set to the number of innodb_buffer_pool_instances if this is lower. If set to `1`, only a single cleaner thread is used, as was the case until MariaDB 10.2.1. Cleaner threads flush dirty pages from the buffer pool, performing flush list and least-recently used (LRU) flushing. Deprecated and ignored from MariaDB 10.5.1, as the original reasons for splitting the buffer pool have mostly gone away.
  - See InnoDB Page Flushing: Page Flushing with Multiple InnoDB Page Cleaner Threads for more information.
- Command line: `--innodb-page-cleaners=#`
- Scope: Global
- Dynamic: Yes (>= MariaDB 10.3.3), No (<= MariaDB 10.3.2)
- Data Type: `numeric`
- Default Value: `4` (or set to innodb_buffer_pool_instances if lower)
- Range: `1` to `64`
- Deprecated: MariaDB 10.5.1
- Removed: MariaDB 10.6.0

`innodb_page_size`

- Description: Specifies the page size in bytes for all InnoDB tablespaces. The default, `16k`, is suitable for most uses.
  - A smaller InnoDB page size might work more effectively in a situation with many small writes (OLTP), or with SSD storage, which usually has smaller block sizes.
  - A larger InnoDB page size can provide a larger maximum row size.
  - InnoDB's page size can be as large as `64k` for tables using the following row formats: DYNAMIC, COMPACT, and REDUNDANT.
  - InnoDB's page size must still be `16k` or less for tables using the COMPRESSED row format.
  - This system variable's value cannot be changed after the `datadir` has been initialized. InnoDB's page size is set when a MariaDB instance starts, and it remains constant afterwards.
- Command line: `--innodb-page-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `enumeration`
- Default Value: `16384`
- Valid Values: `4k` or `4096`, `8k` or `8192`, `16k` or `16384`, `32k` and `64k`.

`innodb_pass_corrupt_table`

- Removed: XtraDB 5.5 - renamed innodb_corrupt_table_action.

`innodb_prefix_index_cluster_optimization`

- Description: Enable prefix optimization to sometimes avoid cluster index lookups. Deprecated and ignored from MariaDB 10.10, as the optimization is now always enabled.
- Command line: `--innodb-prefix-index-cluster-optimization={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.10.1

`innodb_print_all_deadlocks`

- Description: If set to `1` (`0` is default), all InnoDB transaction deadlock information is written to the error log.
- Command line: `--innodb-print-all-deadlocks={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_purge_batch_size`

- Description: Number of InnoDB undo log pages to purge in one batch from the history list. Together with innodb_purge_threads has a small effect on tuning.
- Command line: `--innodb-purge-batch-size=#`
- Scope: Global
- Dynamic: No

- Data Type: `numeric`
- Default Value:
  - `127` (>= MariaDB 10.6.20, MariaDB 10.11.10, MariaDB 11.2.6, MariaDB 11.4.4, MariaDB 11.6.2)
  - `1000` (>= MariaDB 10.6.16, MariaDB 10.10.7, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3 MariaDB 11.2.2)
  - `300` (<= MariaDB 10.6.15, MariaDB 10.10.6, MariaDB 10.11.5, MariaDB 11.0.3, MariaDB 11.1.2 MariaDB 11.2.1)
- Range: `1` to `5000`

`innodb_purge_rseg_truncate_frequency`

- Description: Frequency with which undo records are purged. Set by default to every 128 times, reducing this increases the frequency at which rollback segments are freed. See also innodb_undo_log_truncate. The motivation for introducing this in MySQL seems to have been to avoid stalls due to freeing undo log pages or truncating undo log tablespaces. In MariaDB, innodb_undo_log_truncate=ON should be a much lighter operation because it will not involve any log checkpoint, hence this is deprecated and ignored from MariaDB 10.6.16, MariaDB 10.10.7, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3 and MariaDB 11.2.2. (MDEV-32050)
- Command line: `-- innodb-purge-rseg-truncate-frequency=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `128`
- Range: `1` to `128`
- Deprecated: MariaDB 10.6.16, MariaDB 10.10.7, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3, MariaDB 11.2.2

`innodb_purge_threads`

- Description: Number of background threads dedicated to InnoDB purge operations. The range is `1` to `32`. At least one background thread is always used. Setting to a value greater than 1 creates that many separate purge threads. This can improve efficiency in some cases, such as when performing DML operations on many tables. See also innodb_purge_batch_size.
- Command line: `--innodb-purge-threads=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `4`
- Range: `1` to `32`

`innodb_random_read_ahead`

- Description: Originally, random read-ahead was always set as an optimization technique, but was removed in MariaDB 5.5. `innodb_random_read_ahead` permits it to be re-instated if set to `1` (`0`) is default.
- Command line: `--innodb-random-read-ahead={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_read_ahead`

- Description: If set to `linear`, the default, XtraDB/InnoDB will automatically fetch remaining pages if there are enough within the same extent that can be accessed sequentially. If set to `none`, read-ahead is disabled. `random` has been removed and is now ignored, while `both` sets to both `linear` and `random`. Also see innodb_read_ahead_threshold for more control on read-aheads. Removed in MariaDB 10.0/XtraDB 5.6 and replaced by MySQL 5.6's innodb_random_read_ahead.
- Command line: `innodb-read-ahead=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `linear`
- Valid Values: `none`, `random`, `linear`, `both`
- Removed: MariaDB 10.0/XtraDB 5.6 - replaced by MySQL 5.6's innodb_random_read_ahead

`innodb_read_ahead_threshold`

- Description: Minimum number of pages InnoDB must read sequentially from an extent of 64 before initiating an asynchronous read for the following extent.
- Command line: `--innodb-read-ahead-threshold=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `56`
- Range: `0` to `64`

`innodb_read_io_threads`

- Description: Prior to MariaDB 10.5, this was simply the number of I/O threads for InnoDB reads. From MariaDB 10.5, asynchronous I/O functionality in the InnoDB Background Thread Pool replaces the old InnoDB I/O Threads. This variable is now multiplied by 256 to determine the maximum number of concurrent asynchronous I/O read requests that can be completed by the Background Thread Pool. The default is therefore 4*256 = 1024 conccurrent asynchronous read requests. You may on rare occasions need to reduce this default on Linux systems running multiple MariaDB servers to avoid exceeding system limits, or increase if spending too much time waiting on I/O requests.
- Command line: `--innodb-read-io-threads=#`
- Scope: Global
- Dynamic: Yes (>= MariaDB 10.11), No (<= MariaDB 10.10)
- Data Type: `numeric`
- Default Value: `4`
- Range: `1` to `64`

`innodb_read_only`

- Description: If set to `1` (`0` is default), the server are read-only. For use in distributed applications, data warehouses or read-only media.
- Command line: `--innodb-read-only={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`

- Default Value: `OFF`

`innodb_read_only_compressed`

- Description: If set (the default before MariaDB 10.6.6), ROW_FORMAT=COMPRESSED tables are read-only. This was intended to be the first step towards removing write support and deprecating the feature, but this plan has been abandoned.
- Command line: `--innodb-read-only-compressed`, `--skip-innodb-read-only-compressed`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF` (>= MariaDB 10.6.6), `ON` (<= MariaDB 10.6.5)
- Introduced: MariaDB 10.6.0

`innodb_recovery_stats`

- Description: If set to `1` (`0` is default) and recovery is necessary on startup, the server will write detailed recovery statistics to the error log at the end of the recovery process. This Percona XtraDB variable has not been ported to XtraDB 5.6.
- Command line: No
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.0

`innodb_recovery_update_relay_log`

- Description: If set to `1` (`0` is default), the relay log info file are overwritten on crash recovery if the information differs from the InnoDB record. Should not be used if multiple storage engine types are being replicated. Previously named `innodb_overwrite_relay_log_info`. Removed in MariaDB 10.0/XtraDB 5.6 and replaced by MySQL 5.6's `relay-log-recovery`
- Command line: `innodb-recovery-update-relay-log={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.0 - replaced by MySQL 5.6's `relay-log-recovery`

`innodb_replication_delay`

- Description: Time in milliseconds for the replica server to delay the replication thread if innodb_thread_concurrency is reached. Deprecated and ignored from MariaDB 10.5.5.
- Command line: `--innodb-replication-delay=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4294967295`
- Deprecated: MariaDB 10.5.5

- Removed: MariaDB 10.6.0

`innodb_rollback_on_timeout`

- Description: InnoDB usually rolls back the last statement of a transaction that's been timed out (see innodb_lock_wait_timeout). If innodb_rollback_on_timeout is set to 1 (0 is default), InnoDB will roll back the entire transaction. Before MariaDB 5.5, rolling back the entire transaction was the default behavior.
- Command line: `--innodb-rollback-on-timeout`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `0`

`innodb_rollback_segments`

- Description: Specifies the number of rollback segments that XtraDB/InnoDB will use within a transaction (see undo log). Deprecated and replaced by innodb_undo_logs in MariaDB 10.0. Removed in MariaDB 10.5 as part of an InnoDB cleanup, as it makes sense to always create and use the maximum number of rollback segments. |
- Command line: `--innodb-rollback-segments=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `128`
- Range: `1` to `128`
- Deprecated: MariaDB 10.0
- Removed: MariaDB 10.5.0

`innodb_safe_truncate`

- Description: Use a backup-safe TRUNCATE TABLE implementation and crash-safe rename operations inside InnoDB. This is not compatible with hot backup tools other than mariadb-backup. Users who need to use such tools may set this to `OFF`.
- Command line: `--innodb-safe-truncate={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.2.19
- Removed: MariaDB 10.3.0

`innodb_scrub_log`

- Description: Enable InnoDB redo log scrubbing. See Data Scrubbing. Deprecated and ignored from MariaDB 10.5.2, as never really worked (MDEV-13019 and MDEV-18370). If old log contents should be kept secret, then enabling innodb_encrypt_log or setting a smaller innodb_log_file_size could help.
- Command line: `--innodb-scrub-log`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

- Deprecated: MariaDB 10.5.2
- Removed: MariaDB 10.6.0

`innodb_scrub_log_interval`

- Description: Used with Data Scrubbing in 10.1.3 only - replaced in 10.1.4 by innodb_scrub_log_speed. InnoDB redo log scrubbing interval in milliseconds.
- Command line: `--innodb-scrub-log-interval=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `56`
- Range: `0` to `50000`
- Introduced: MariaDB 10.1.3
- Removed: MariaDB 10.1.4

`innodb_scrub_log_speed`

- Description: InnoDB redo log scrubbing speed in bytes/sec. See Data Scrubbing. Deprecated and ignored from MariaDB 10.5.2.
- Command line: `--innodb-scrub-log-speed=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `256`
- Range: `1` to `50000`
- Deprecated: MariaDB 10.5.2
- Removed: MariaDB 10.6.0

`innodb_sched_priority_cleaner`

- Description: Set a thread scheduling priority for cleaner and least-recently used (LRU) manager threads. The range from `0` to `39` corresponds in reverse order to Linux nice values of `-20` to `19`. So `0` is the lowest priority (Linux nice value `19`) and `39` is the highest priority (Linux nice value `-20`). XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-sched-priority-cleaner=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `19`
- Range: `0` to `39`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_show_locks_held`

- Description: Specifies the number of locks held for each InnoDB transaction to be displayed in SHOW ENGINE INNODB STATUS output. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- Command line: `innodb-show-locks-held=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10`
- Range: `0` to `1000`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_show_verbose_locks`

- Description: If set to `1`, and innodb_status_output_locks is also ON, the traditional InnoDB behavior is followed and locked records are shown in INFORMATION_SCHEMA.INNODB_METRICS output. If set to `0`, the default, only high-level information about the lock is shown. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-show-verbose-locks=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `1`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_simulate_comp_failures`

- Description: Simulate compression failures. Used for testing robustness against random compression failures. XtraDB only.
- Command line: None
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `99`

`innodb_snapshot_isolation`

- Description: Whether or not to use snapshot isolation (write/write conflict detection within InnoDB).
  If enabled (set to `ON`), an error `DB_RECORD_CHANGED` (`HA_ERR_RECORD_CHANGED`, `ER_CHECKREAD`) is raised if an attempt is made to acquire a lock on a record that does not exist in the current read view. This error is treated in the same way as a deadlock, and the transaction is rolled back. This affects the default isolation level, REPEATABLE READ.
  In MariaDB 11.8 and later, changes to snapshot handling may affect how conflicts are detected in transactions that use the Repeatable Read isolation level. Specifically, `DELETE` and `UPDATE` statements can cause a rollback of the transaction, and return with ERROR 1020. This occurs when a transaction tries to modify rows using a snapshot that is inconsistent with the database's current state as a result of simultaneous modifications. In other words, this variable is designed to prevent non-repeatable reads and anomalies (like lost updates) by rejecting `UPDATE` or `DELETE` operations on rows that were modified by a concurrent transaction after the current transaction's snapshot was taken.

- Command line: `--innodb-snapshot-isolation={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON` (>= MariaDB 11.6.2), `OFF` (<= MariaDB 11.6.1)
- Introduced: MariaDB 10.6.18, MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5, MariaDB 11.2.4, MariaDB 11.4.2

`innodb_sort_buffer_size`

- Description: Size of the sort buffers used for sorting data when an InnoDB index is created, as well as the amount by which the temporary log file is extended during online DDL operations to record concurrent writes. The larger the setting, the fewer merge phases are required between buffers while sorting. When a CREATE TABLE or ALTER TABLE creates a new index, three buffers of this size are allocated, as well as pointers for the rows in the buffer.
- Command line: `--innodb-sort-buffer-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1048576` (1M)
- Range: `65536` to `67108864`

`innodb_spin_wait_delay`

- Description: Maximum delay (not strictly corresponding to a time unit) between spin lock polls. Default changed from `6` to `4` in MariaDB 10.3.5, as this was verified to give the best throughput by OLTP update index and read-write benchmarks on Intel Broadwell (2/20/40) and ARM (1/46/46).
- Command line: `--innodb-log-spin-wait-delay=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4` (>= MariaDB 10.3.5), `6` (<= MariaDB 10.3.4)
- Range: `0` to `4294967295`

`innodb_stats_auto_recalc`

- Description: If set to `1` (the default), persistent statistics are automatically recalculated when the table changes significantly (more than 10% of the rows). Affects tables created or altered with STATS_PERSISTENT=1 (see CREATE TABLE), or when innodb_stats_persistent is enabled. innodb_stats_persistent_sample_pages determines how much data to sample when recalculating. See InnoDB Persistent Statistics.
- Command line: `--innodb-stats-auto-recalc={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`innodb_stats_auto_update`

- Description: If set to `0` (`1` is default), index statistics will not be automatically calculated except when an ANALYZE TABLE is run, or the table is first opened. Replaced by innodb_stats_auto_recalc in MariaDB

10.0/XtraDB 5.6.

- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`
- Removed: MariaDB 10.0 - replaced by innodb_stats_auto_recalc.

`innodb_stats_include_delete_marked`

- Description: Include delete marked records when calculating persistent statistics.
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_stats_method`

- Description: Determines how NULLs are treated for InnoDB index statistics purposes.
  - `nulls_equal`: The default, all NULL index values are treated as a single group. This is usually fine, but if you have large numbers of NULLs the average group size is slanted higher, and the optimizer may miss using the index for ref accesses when it would be useful.
  - `nulls_unequal`: The opposite approach to `nulls_equal` is taken, with each NULL forming its own group of one. Conversely, the average group size is slanted lower, and the optimizer may use the index for ref accesses when not suitable.
  - `nulls_ignored`: Ignore NULLs altogether from index group calculations.
  - See also Index Statistics, aria_stats_method and myisam_stats_method.
- Command line: `--innodb-stats-method=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `nulls_equal`
- Valid Values: `nulls_equal`, `nulls_unequal`, `nulls_ignored`

`innodb_stats_modified_counter`

- Description: The number of rows modified before we calculate new statistics. If set to `0`, the default, current limits are used.
- Command line: `--innodb-stats-modified-counter=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`innodb_stats_on_metadata`

- Description: If set to `ON`, XtraDB/InnoDB updates statistics when accessing the INFORMATION_SCHEMA.TABLES or INFORMATION_SCHEMA.STATISTICS tables, and when running metadata statements such as SHOW INDEX or SHOW TABLE STATUS. If set to `OFF` (the default), statistics are

not updated at those times, which can reduce the access time for large schemas, as well as make execution plans more stable.

- Command line: `--innodb-stats-on-metadata`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_stats_persistent`

- Description: ANALYZE TABLE produces index statistics, and this setting determines whether they are stored on disk, or be required to be recalculated more frequently, such as when the server restarts. This information is stored for each table, and can be set with the STATS_PERSISTENT clause when creating or altering tables (see CREATE TABLE). See InnoDB Persistent Statistics.
- Command line: `--innodb-stats-persistent={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`innodb_stats_persistent_sample_pages`

- Description: Number of index pages sampled when estimating cardinality and statistics for indexed columns. Increasing this value will increases index statistics accuracy, but use more I/O resources when running ANALYZE TABLE. See InnoDB Persistent Statistics.
- Command line: `--innodb-stats-persistent-sample-pages=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `20`

`innodb_stats_sample_pages`

- Description: Gives control over the index distribution statistics by determining the number of index pages to sample. Higher values produce more disk I/O, but, especially for large tables, produce more accurate statistics and therefore make more effective use of the query optimizer. Lower values than the default are not recommended, as the statistics can be quite inaccurate.
  - If innodb_stats_traditional is enabled, then the exact number of pages configured by this system variable are sampled for statistics.
  - If innodb_stats_traditional is disabled, then the number of pages to sample for statistics is calculated using a logarithmic algorithm, so the exact number can change depending on the size of the table. This means that more samples may be used for larger tables.
  - If persistent statistics are enabled, then the innodb_stats_persistent_sample_pages system variable applies instead. persistent statistics are enabled with the innodb_stats_persistent system variable.
  - This system variable has been deprecated. The innodb_stats_transient_sample_pages system variable should be used instead.
- Command line: `--innodb-stats-sample-pages=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `8`
- Range: `1` to `2^64-1`
- Deprecated: MariaDB 10.0
- Removed: MariaDB 10.5.0

`innodb_stats_traditional`

- Description: This system variable affects how the number of pages to sample for transient statistics is determined, in particular how innodb_stats_transient_sample_pages is used.
  - If innodb_stats_traditional is enabled, then the exact number of pages configured by the system variable are sampled for statistics.
  - If innodb_stats_traditional is disabled, then the number of pages to sample for statistics is calculated using a logarithmic algorithm, so the exact number can change depending on the size of the table. This means that more samples may be used for larger tables.
  - This system variable does not affect the calculation of persistent statistics.
- Command line: `--innodb-stats-traditional={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`innodb_stats_transient_sample_pages`

- Description: Gives control over the index distribution statistics by determining the number of index pages to sample. Higher values produce more disk I/O, but, especially for large tables, produce more accurate statistics and therefore make more effective use of the query optimizer. Lower values than the default are not recommended, as the statistics can be quite inaccurate.
  - If innodb_stats_traditional is enabled, then the exact number of pages configured by this system variable are sampled for statistics.
  - If innodb_stats_traditional is disabled, then the number of pages to sample for statistics is calculated using a logarithmic algorithm, so the exact number can change depending on the size of the table. This means that more samples may be used for larger tables.
  - If persistent statistics are enabled, then the innodb_stats_persistent_sample_pages system variable applies instead. persistent statistics are enabled with the innodb_stats_persistent system variable.
- Command line: `--innodb-stats-transient-sample-pages=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `8`
- Range: `1` to `2^64-1`

`innodb_stats_update_need_lock`

- Description: Setting to `0` (`1` is default) may help reduce contention of the `&dict_operation_lock`, but also disables the Data_free option in SHOW TABLE STATUS. This Percona XtraDB variable has not been ported to XtraDB 5.6.
- Scope: Global
- Dynamic: Yes

- Data Type: `boolean`
- Default Value: `1`
- Removed: MariaDB 10.0/XtraDB 5.6

`innodb_status_output`

- Description: Enable InnoDB monitor output to the error log.
- Command line: `--innodb-status-output={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_status_output_locks`

- Description: Enable InnoDB lock monitor output to the error log and SHOW ENGINE INNODB STATUS. Also requires innodb_status_output=ON to enable output to the error log.
- Command line: `--innodb-status-output-locks={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_strict_mode`

- Description: If set to `1` (the default), InnoDB will return errors instead of warnings in certain cases, similar to strict SQL mode. See InnoDB Strict Mode for details.
- Command line: `--innodb-strict-mode={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`innodb_support_xa`

- Description: If set to `1`, the default, XA transactions are supported. XA support ensures data is written to the binary log in the same order to the actual database, which is critical for replication and disaster recovery, but comes at a small performance cost. If your database is set up to only permit one thread to change data (for example, on a replication replica with only the replication thread writing), it is safe to turn this option off. Removed in MariaDB 10.3, XA transactions are always supported.
- Command line: `--innodb-support-xa`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 10.2
- Removed: MariaDB 10.3.0

`innodb_sync_array_size`

- Description: By default `1`, can be increased to split internal thread co-ordinating, giving higher concurrency when there are many waiting threads.
- Command line: `--innodb-sync-array-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `1024`
- Removed: MariaDB 10.6.0

`innodb_sync_spin_loops`

- Description: The number of times a thread waits for an InnoDB mutex to be freed before the thread is suspended.
- Command line: `--innodb-sync-spin-loops=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `30`
- Range: `0` to `4294967295`

`innodb_table_locks`

- Description: If autocommit is set to `0` (`1` is default), setting innodb_table_locks to `1`, the default, will cause InnoDB to lock a table internally upon a LOCK TABLE.
- Command line: `--innodb-table-locks`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`innodb_thread_concurrency`

- Description: Once this number of threads is reached (excluding threads waiting for locks), XtraDB/InnoDB will place new threads in a wait state in a first-in, first-out queue for execution, in order to limit the number of threads running concurrently. A setting of `0`, the default, permits as many threads as necessary. A suggested setting is twice the number of CPU's plus the number of disks. Deprecated and ignored from MariaDB 10.5.5.
- Command line: `--innodb-thread-concurrency=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `1000`
- Deprecated: MariaDB 10.5.5
- Removed: MariaDB 10.6.0

`innodb_thread_concurrency_timer_based`

- Description: If set to `1`, thread concurrency are handled in a lock-free timer-based manner rather than the default mutex-based method. Depends on atomic op builtins being available. This Percona XtraDB variable has not been ported to XtraDB 5.6.
- Command line: `innodb-thread-concurrency-timer-based={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.0/XtraDB 5.6

`innodb_thread_sleep_delay`

- Description: Time in microseconds that InnoDB threads sleep before joining the queue. Setting to `0` disables sleep. Deprecated and ignored from MariaDB 10.5.5
- Command line: `--innodb-thread-sleep-delay=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `0` (>= MariaDB 10.5.5.)
  - `10000` (<= MariaDB 10.5.4)
- Range: `0` to `1000000`
- Deprecated: MariaDB 10.5.5
- Removed: MariaDB 10.6.0

`innodb_temp_data_file_path`

- Description: Path where to store data for InnoDB temporary tables. Argument is `filename:size` followed by options separated by ':' Multiple paths can be given separated by ';' A file size is specified (with K for kilobytes, M for megabytes and G for gigabytes). Also whether or not to `autoextend` the data file, `max` size and whether or not to autoshrink on startup may also be specified.
- Command line: `--innodb-temp-data-file-path=path`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `ibtmp1:12M:autoextend`
- Documentation & examples: innodb-temporary-tablespaces

`innodb_tmpdir`

- Description: Allows an alternate location to be set for temporary non-tablespace files. If not set (the default), files are created in the usual tmpdir location.
  Alternate location must be outside of `datadir`
- Command line: `--innodb-tmpdir=path`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`

- Default Value: Empty

`innodb_track_changed_pages`

- Description: For faster incremental backup with Xtrabackup, XtraDB tracks pages with changes written to them according to the XtraDB redo log and writes the information to special changed page bitmap files. This read-only variable is used for controlling this feature. See also innodb_max_changed_pages and innodb_max_bitmap_file_size. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-track-changed-pages={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.2.6

`innodb_track_redo_log_now`

- Description: Available on debug builds only. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-track-redo-log-now={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.2.6

`innodb_truncate_temporary_tablespace_now`

- Description: Set to ON to shrink the temporary tablespace.
- Command line: `innodb-truncate-temporary-tablespace-now={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 11.3.0

`innodb_undo_directory`

- Description: Path to the directory (relative or absolute) that InnoDB uses to create separate tablespaces for the undo logs. `.` (the default value before 10.2.2) leaves the undo logs in the same directory as the other log files. From MariaDB 10.2.2, the default value is NULL, and if no path is specified, undo tablespaces are created in the directory defined by datadir. Use together with innodb_undo_logs and innodb_undo_tablespaces. Undo logs are most usefully placed on a separate storage device.
- Command line: `--innodb-undo-directory=name`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: NULL

`innodb_undo_log_truncate`

- Description: When enabled, innodb_undo_tablespaces that are larger than innodb_max_undo_log_size are marked for truncation. See also innodb_purge_rseg_truncate_frequency. Enabling this setting may cause stalls during heavy write workloads.
- Command line: `--innodb-undo-log-truncate[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`innodb_undo_logs`

- Description: Specifies the number of rollback segments that XtraDB/InnoDB will use within a transaction (or the number of active undo logs). By default set to the maximum, `128`, it can be reduced to avoid allocating unneeded rollback segments. See the Innodb_available_undo_logs status variable for the number of undo logs available. See also innodb_undo_directory and innodb_undo_tablespaces. Replaced innodb_rollback_segments in MariaDB 10.0. The Information Schema XTRADB_RSEG Table contains information about the XtraDB rollback segments. Deprecated and ignored in MariaDB 10.5.0, as it always makes sense to use the maximum number of rollback segments.
- Command line: `--innodb-undo-logs=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `128`
- Range: `0` to `128`
- Deprecated: MariaDB 10.5.0
- Removed: MariaDB 10.6.0

`innodb_undo_tablespaces`

- Description: Number of tablespaces files used for dividing up the undo logs. Zero (the default before MariaDB 11.0) means that undo logs are all part of the system tablespace, which contains one undo tablespace more than the `innodb_undo_tablespaces` setting. A value of 1 is reset to 0 as 2 or more are needed for separate tablespaces. When the undo logs can grow large, splitting them over multiple tablespaces will reduce the size of any single tablespace. Until MariaDB 10.11.1, must be set before InnoDB is initialized, or else MariaDB will fail to start, with an error saying that `InnoDB did not find the expected number of undo tablespaces`. The files are created in the directory specified by innodb_undo_directory, and are named `undoN`, N being an integer. The default size of an undo tablespace is 10MB.From MariaDB 11.0, multiple undo tablespaces are enabled by default, and the default is changed to 3 so that the space occupied by possible bursts of undo log records can be reclaimed after innodb_undo_log_truncate is set. Before MariaDB 10.6, innodb_undo_logs must have a non-zero setting for `innodb_undo_tablespaces` to take effect.
- Command line: `--innodb-undo-tablespaces=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `3` (>= MariaDB 11.0), `0` (<= MariaDB 10.11)
- Range: `0`, or `2` to `95`

`innodb_use_atomic_writes`

- Description: Implement atomic writes on supported SSD devices. See atomic write support for other variables affected when this is set.
- Command line: `innodb-use-atomic-writes={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`innodb_use_fallocate`

- Description: Preallocate files fast, using operating system functionality. On POSIX systems, posix_fallocate system call is used.
  - Automatically set to `1` when innodb_use_atomic_writes is set - see FusionIO DirectFS atomic write support.
  - See InnoDB Page Compression: Saving Storage Space with Sparse Files for more information.
- Command line: `innodb-use-fallocate={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.2.5 (treated as if `ON`)
- Removed: MariaDB 10.3.0

`innodb_use_global_flush_log_at_trx_commit`

- Description: Determines whether a user can set the variable innodb_flush_log_at_trx_commit. If set to `1`, a user cannot reset the value with a SET command, while if set to `1`, a user can reset the value of `innodb_flush_log_at_trx_commit`. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `innodb-use-global-flush-log-at-trx_commit={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_use_mtflush`

- Description: Whether to enable Multi-Threaded Flush operations.
  For more information, see Fusion.
  - InnoDB's multi-thread flush feature was deprecated in MariaDB 10.2.9 and removed from MariaDB 10.3.2. In later versions of MariaDB, use innodb_page_cleaners system variable instead.
  - See InnoDB Page Flushing: Page Flushing with Multi-threaded Flush Threads for more information.
- Command line: `--innodb-use-mtflush={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`

- Default Value: `OFF`
- Deprecated: MariaDB 10.2.9
- Removed: MariaDB 10.3.2

`innodb_use_native_aio`

- Description: For Linux systems only, specified whether to use Linux's asynchronous I/O subsystem. Set to `ON` by default, it may be changed to `0` at startup if InnoDB detects a problem, or from MariaDB 10.6.5/MariaDB 10.7.1, if a 5.11 - 5.15 Linux kernel is detected, to avoid an io-uring bug/incompatibility (MDEV-26674). MariaDB-10.6.6/MariaDB-10.7.2 and later also consider 5.15.3+ as a fixed kernel and default to `ON`. To really benefit from the setting, the files should be opened in O_DIRECT mode (innodb_flush_method=O_DIRECT, default from MariaDB 10.6), to bypass the file system cache. In this way, the reads and writes can be submitted with DMA, using the InnoDB buffer pool directly, and no processor cycles need to be used for copying data.
- Command line: `--innodb-use-native-aio={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`innodb_use_purge_thread`

- Description: Usually with InnoDB, data changed by a transaction is written to an undo space to permit read consistency, and freed when the transaction is complete. Many, or large, transactions, can cause the main tablespace to grow dramatically, reducing performance. This option, introduced in XtraDB 5.1 and removed for 5.5, allows multiple threads to perform the purging, resulting in slower, but much more stable performance.
- Command line: `--innodb-use-purge-thread=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `32`
- Removed: XtraDB 5.5

`innodb_use_stacktrace`

- Description: If set to `ON` (`OFF` is default), a signal handler for SIGUSR2 is installed when the InnoDB server starts. When a long semaphore wait is detected at sync/sync0array.c, a SIGUSR2 signal is sent to the waiting thread and thread that has acquired the RW-latch. For both threads a full stacktrace is produced as well as if possible. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- Command line: `--innodb-use-stacktrace={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.2.6
- Removed: MariaDB 10.3.0

`innodb_use_sys_malloc`

- Description: If set the `1`, the default, XtraDB/InnoDB will use the operating system's memory allocator. If set to `0` it will use its own. Deprecated in MariaDB 10.0 and removed in MariaDB 10.2 along with InnoDB's internal memory allocator.
- Command line: `--innodb-use-sys-malloc={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 10.0
- Removed: MariaDB 10.2.2

`innodb_use_sys_stats_table`

- Description: If set to `1` (`0` is default), XtraDB will use the SYS_STATS system table for extra table index statistics. When a table is opened for the first time, statistics will then be loaded from SYS_STATS instead of sampling the index pages. Statistics are designed to be maintained only by running an ANALYZE TABLE. Replaced by MySQL 5.6's Persistent Optimizer Statistics.
- Command line: `innodb-use-sys-stats-table={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `0`
- Removed: MariaDB 10.0/XtraDB 5.6

`innodb_use_trim`

- Description: Use trim to free up space of compressed blocks.
  - See InnoDB Page Compression: Saving Storage Space with Sparse Files for more information.
- Command line: `--innodb-use-trim={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 10.2.4
- Removed: MariaDB 10.3.0

`innodb_version`

- Description: InnoDB version number. From MariaDB 10.3.7, as the InnoDB implementation in MariaDB has diverged from MySQL, the MariaDB version is instead reported. For example, the InnoDB version reported in MariaDB 10.1 (which is based on MySQL 5.6) included encryption and variable-size page compression before MySQL 5.7 introduced them. MariaDB 10.2 (based on MySQL 5.7) introduced persistent AUTO_INCREMENT (MDEV-6076) in a GA release before MySQL 8.0. MariaDB 10.3 (based on MySQL 5.7) introduced instant ADD COLUMN (MDEV-11369) before MySQL.
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Removed: MariaDB 10.10

`innodb_write_io_threads`

- Description: Prior to MariaDB 10.5, this was simply the number of I/O threads for InnoDB writes. From MariaDB 10.5, asynchronous I/O functionality in the InnoDB Background Thread Pool replaces the old InnoDB I/O Threads. This variable is now multiplied by 256 to determine the maximum number of concurrent asynchronous I/O write requests that can be completed by the Background Thread Pool. The default is therefore 4*256 = 1024 conccurrent asynchronous write requests. You may on rare occasions need to reduce this default on Linux systems running multiple MariaDB servers to avoid exceeding system limits, or increase if spending too much time waiting on I/O requests.
- Command line: `--innodb-write-io-threads=#`
- Scope: Global
- Dynamic: Yes (>= MariaDB 10.11), No (<= MariaDB 10.10)
- Data Type: `numeric`
- Default Value: `4`
- Range: `1` to `64`

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Versions

> **Warning:**
>
> This page is outdated. It's left in place because release notes for old MariaDB versions refer to it (MariaDB < 10.3).

The InnoDB implementation has diverged substantially from the InnoDB in MySQL. Therefore, in these versions, the InnoDB version is no longer associated with a MySQL release version.

The default InnoDB implementation is based on InnoDB from MySQL 5.7. See Why MariaDB uses InnoDB instead of XtraDB from MariaDB 10.2 for more information.

Note

XtraDB is a performance enhanced fork of InnoDB. For compatibility reasons, the system variables still retain their original `innodb` prefixes. If the documentation says that something applies to InnoDB, then it usually also applies to the XtraDB fork, unless explicitly stated otherwise. In these versions, it is still possible to use InnoDB instead of XtraDB. See Using InnoDB instead of XtraDB for more information.

Divergences

Some examples of divergences between MariaDB's InnoDB and MySQL's InnoDB are:

- MariaDB 10.1 (which is based on MySQL 5.6) included encryption and variable-size page compression before MySQL 5.7 introduced them.
- MariaDB 10.2 (based on MySQL 5.7) introduced persistent AUTO_INCREMENT (MDEV-6076) in a GA release before MySQL 8.0.
- MariaDB 10.3 (based on MySQL 5.7) introduced instant ADD COLUMN (MDEV-11369) before MySQL.

InnoDB Versions Included in MariaDB Releases

MariaDB 10.3

InnoDB Version	Introduced
No longer reported	MariaDB 10.3.7
InnoDB 5.7.20	MariaDB 10.3.3

InnoDB Version	Introduced
InnoDB 5.7.19	MariaDB 10.3.1
InnoDB 5.7.14	MariaDB 10.3.0

MariaDB 10.2

InnoDB Version	Introduced
InnoDB 5.7.29	MariaDB 10.2.33
InnoDB 5.7.23	MariaDB 10.2.17
InnoDB 5.7.22	MariaDB 10.2.15
InnoDB 5.7.21	MariaDB 10.2.13
InnoDB 5.7.20	MariaDB 10.2.10
InnoDB 5.7.19	MariaDB 10.2.8
InnoDB 5.7.18	MariaDB 10.2.7
InnoDB 5.7.14	MariaDB 10.2.2

MariaDB 10.1

InnoDB Version	Introduced
InnoDB 5.6.49	MariaDB 10.1.46
InnoDB 5.6.47	MariaDB 10.1.44
InnoDB 5.6.44	MariaDB 10.1.39
InnoDB 5.6.42	MariaDB 10.1.37
InnoDB 5.6.39	MariaDB 10.1.31
InnoDB 5.6.37	MariaDB 10.1.26
InnoDB 5.6.36	MariaDB 10.1.24
InnoDB 5.6.35	MariaDB 10.1.21
InnoDB 5.6.33	MariaDB 10.1.18
InnoDB 5.6.32	MariaDB 10.1.17
InnoDB 5.6.31	MariaDB 10.1.16
InnoDB 5.6.30	MariaDB 10.1.14
InnoDB 5.6.29	MariaDB 10.1.12

MariaDB 10.0

InnoDB Version	Introduced
InnoDB 5.6.43	MariaDB 10.0.38
InnoDB 5.6.42	MariaDB 10.0.37
InnoDB 5.6.40	MariaDB 10.0.35
InnoDB 5.6.39	MariaDB 10.0.34
InnoDB 5.6.38	MariaDB 10.0.33
InnoDB 5.6.37	MariaDB 10.0.32
InnoDB 5.6.36	MariaDB 10.0.31
InnoDB 5.6.35	MariaDB 10.0.29

InnoDB Version	Introduced
InnoDB 5.6.33	MariaDB 10.0.28
InnoDB 5.6.32	MariaDB 10.0.27
InnoDB 5.6.31	MariaDB 10.0.26
InnoDB 5.6.30	MariaDB 10.0.25
InnoDB 5.6.29	MariaDB 10.0.24
InnoDB 5.6.28	MariaDB 10.0.23
InnoDB 5.6.27	MariaDB 10.0.22
InnoDB 5.6.26	MariaDB 10.0.21
InnoDB 5.6.25	MariaDB 10.0.20
InnoDB 5.6.24	MariaDB 10.0.18
InnoDB 5.6.23	MariaDB 10.0.17
InnoDB 5.6.22	MariaDB 10.0.16
InnoDB 5.6.21	MariaDB 10.0.15
InnoDB 5.6.20	MariaDB 10.0.14
InnoDB 5.6.19	MariaDB 10.0.13
InnoDB 5.6.17	MariaDB 10.0.11
InnoDB 5.6.15	MariaDB 10.0.9
InnoDB 5.6.14	MariaDB 10.0.8

See Also

- Why MariaDB uses InnoDB instead of XtraDB from MariaDB 10.2
- XtraDB Versions

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Architecture for MariaDB Enterprise Server

See MariaDB Enterprise Server InnoDB Background Thread Pool.

This page details the dedicated thread pool in MariaDB Enterprise Server that manages InnoDB background tasks, improving scalability and performance.

See MariaDB Enterprise Server InnoDB I/O Threads.

Learn about the specialized I/O threads in MariaDB Enterprise Server's InnoDB engine that handle asynchronous read and write operations efficiently.

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Server InnoDB Background Thread Pool

Overview

Starting with MariaDB Enterprise Server 10.5 and MariaDB Community Server 10.5, InnoDB uses the InnoDB Background Thread Pool to perform internal operations in the background. In previous versions, the internal operations were performed by dedicated threads. By using the InnoDB Background Thread Pool instead of many dedicated threads, InnoDB can reduce context switching and use system resources more effectively.

The InnoDB Background Thread Pool performs internal operations in multiple categories: **tasks**, **timers**, and **asynchronous I/O**.

Tasks are used to perform internal operations that are triggered by some event. In ES 10.5 and later and CS 10.5 and later, the following threads have been replaced by **tasks** with the InnoDB Background Thread Pool:

- The InnoDB Buffer Pool Resize Thread
- The InnoDB Buffer Pool Dump Thread
- The InnoDB Full-Text Search (FTS) Optimization Thread

Timers are used to perform internal operations that are triggered periodically. In ES 10.5 and later and CS 10.5 and later, the following threads have been replaced by **timers** with the InnoDB Background Thread Pool:

- The InnoDB Master Thread
- The InnoDB Defragmentation Thread
- The InnoDB Monitor Thread
- The InnoDB Error Monitor Thread

Asynchronous I/O is used to read from and write to disk asynchronously. In ES 10.5 and later and CS 10.5 and later, the following threads have been replaced by **asynchronous I/O** with the InnoDB Background Thread Pool:

- The InnoDB I/O Threads

Feature Summary

Feature	Detail	Resources
Thread Pool	InnoDB Background Thread Pool	
Storage Engine	InnoDB	
Purpose	Handles background tasks for InnoDB	
Availability	• ES 10.5+ • CS 10.5+	MariaDB Enterprise Server

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Server InnoDB I/O Threads

Overview

Starting with MariaDB Enterprise Server 10.5 and MariaDB Community Server 10.5, the InnoDB I/O Threads were replaced by the asynchronous I/O functionality in the InnoDB Background Thread Pool.

Feature Summary

Feature	Detail	Resources
Thread	InnoDB I/O Threads	
Storage Engine	InnoDB	
Purpose	Reading data from disk / Writing data to disk	
Availability		MariaDB Enterprise Server
Quantity	Set by innodb_read_io_threads and innodb_write_io_threads	Configure the InnoDB I/O Threads

Basic Configuration

```
[mariadb]
...
innodb_read_io_threads=8
innodb_write_io_threads=8
```

```
SET GLOBAL innodb_read_io_threads=8;
SET GLOBAL innodb_write_io_threads=8;

SHOW GLOBAL VARIABLES
   LIKE 'innodb_%_io_threads';
```

```
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| innodb_read_io_threads  | 8     |
| innodb_write_io_threads | 8     |
+-------------------------+-------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

InnoDB Online DDL

See InnoDB Online DDL Overview.

An introduction to InnoDB's online DDL capabilities, detailing the ALGORITHM and LOCK clauses for controlling performance and concurrency during schema changes.

See InnoDB Online DDL Operations with the INPLACE Alter Algorithm.

Learn about operations supported by the INPLACE algorithm, which rebuilds the table but allows concurrent DML, offering a balance between performance and availability.

See InnoDB Online DDL Operations with the INSTANT Alter Algorithm.

Discover the INSTANT algorithm, which modifies table metadata without rebuilding the table, enabling extremely fast schema changes like adding columns.

See InnoDB Online DDL Operations with the NOCOPY Alter Algorithm.

Understand the NOCOPY algorithm, which avoids rebuilding the clustered index for certain operations like adding secondary indexes, significantly reducing I/O.

See Instant ADD COLUMN for InnoDB.

A focused guide on the Instant ADD COLUMN feature, explaining how it works by modifying metadata and its advantages over traditional table-rebuilding methods.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Online DDL Overview

InnoDB tables support online DDL, which permits concurrent DML and uses optimizations to avoid unnecessary table copying.

The ALTER TABLE statement supports two clauses that are used to implement online DDL:

- ALGORITHM - This clause controls how the DDL operation is performed.
- LOCK - This clause controls how much concurrency is allowed while the DDL operation is being performed.

Alter Algorithms

InnoDB supports multiple algorithms for performing DDL operations. This offers a significant performance improvement over previous versions. The supported algorithms are:

- `DEFAULT` - This implies the default behavior for the specific operation.
- `COPY`
- `INPLACE`
- `NOCOPY`
- `INSTANT`

Specifying an Alter Algorithm

The set of alter algorithms can be considered as a hierarchy. The hierarchy is ranked in the following order, with *least efficient* algorithm at the top, and *most efficient* algorithm at the bottom:

- `COPY`
- `INPLACE`
- `NOCOPY`
- `INSTANT`

When a user specifies an alter algorithm for a DDL operation, MariaDB does not necessarily use that specific algorithm for the operation. It interprets the choice in the following way:

- If the user specifies `COPY`, then InnoDB uses the `COPY` algorithm.
- If the user specifies any other algorithm, then InnoDB interprets that choice as the least efficient algorithm that the user is willing to accept. This means that if the user specifies `INPLACE`, then InnoDB will use the most efficient algorithm supported by the specific operation from the set (`INPLACE`, `NOCOPY`, `INSTANT`). Likewise, if the user specifies `NOCOPY`, then InnoDB will use the most efficient algorithm supported by the specific operation from the set (`NOCOPY`, `INSTANT`).

There is also a special value that can be specified:

- If the user specifies `DEFAULT`, then InnoDB uses its default choice for the operation. The default choice is to use the most efficient algorithm supported by the operation. The default choice will also be used if no algorithm is specified. Therefore, if you want InnoDB to use the most efficient algorithm supported by an operation, then you usually do not have to explicitly specify any algorithm at all.

Specifying an Alter Algorithm Using the ALGORITHM Clause

InnoDB supports the ALGORITHM clause.

The ALGORITHM clause can be used to specify the *least efficient* algorithm that the user is willing to accept. It is supported by the ALTER TABLE and CREATE INDEX statements.

For example, if a user wanted to add a column to a table, but only if the operation used an algorithm that is at least as efficient as the `INPLACE`, then they could execute the following:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

ALTER TABLE tab ADD COLUMN c VARCHAR(50), ALGORITHM=INPLACE;
```

The above operation should use the `INSTANT` algorithm, because the `ADD COLUMN` operation supports the `INSTANT` algorithm, and the `INSTANT` algorithm is more efficient than the `INPLACE` algorithm.

Specifying an Alter Algorithm Using System Variables

The alter_algorithm system variable can be used to pick the *least efficient* algorithm that the user is willing to accept.

For example, if a user wanted to add a column to a table, but only if the operation used an algorithm that is at least as efficient as the `INPLACE`, then they could execute the following:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD COLUMN c VARCHAR(50);
```

The above operation would actually use the `INSTANT` algorithm, because the `ADD COLUMN` operation supports the `INSTANT` algorithm, and the `INSTANT` algorithm is more efficient than the `INPLACE` algorithm.
<>

Supported Alter Algorithms

The supported algorithms are described in more details below.

DEFAULT Algorithm

The default behavior, which occurs if `ALGORITHM=DEFAULT` is specified, or if `ALGORITHM` is not specified at all, usually only makes a copy if the operation doesn't support being done in-place at all. In this case, the *most efficient* available algorithm will usually be used.

This means that, if an operation supports the `INSTANT` algorithm, then it will use that algorithm by default. If an operation does not support the `INSTANT` algorithm, but it does support the `NOCOPY` algorithm, then it will use that algorithm by default. If an operation does not support the `NOCOPY` algorithm, but it does support the `INPLACE` algorithm, then it will use that algorithm by default.

COPY Algorithm

The `COPY` algorithm refers to the original ALTER TABLE algorithm.

When the `COPY` algorithm is used, MariaDB essentially does the following operations:

```
-- Create a temporary table with the new definition
CREATE TEMPORARY TABLE tmp_tab (
...
);

-- Copy the data from the original table
INSERT INTO tmp_tab
   SELECT * FROM original_tab;

-- Drop the original table
DROP TABLE original_tab;

-- Rename the temporary table, so that it replaces the original one
RENAME TABLE tmp_tab TO original_tab;
```

This algorithm is very inefficient, but it is generic, so it works for all storage engines.

If the `COPY` algorithm is specified with the ALGORITHM clause or with the alter_algorithm system variable, then the `COPY` algorithm are used even if it is not necessary. This can result in a lengthy table copy. If multiple ALTER TABLE operations are required that each require the table to be rebuilt, then it is best to specify all operations in a single ALTER TABLE statement, so that the table is only rebuilt once.

Using the COPY Algorithm with InnoDB

If the `COPY` algorithm is used with an InnoDB table, then the following statements apply:

- The table are rebuilt using the current values of the innodb_file_per_table, innodb_file_format, and innodb_default_row_format system variables.
- The operation will have to create a temporary table to perform the table copy. This temporary table are in the same directory as the original table, and it's file name are in the format `#sql${PID}_${THREAD_ID}_${TMP_TABLE_COUNT}`, where `${PID}` is the process ID of `mysqld`, `${THREAD_ID}` is the connection ID, and `${TMP_TABLE_COUNT}` is the number of temporary tables that the connection has open. Therefore, the datadir may contain files with file names like `#sql1234_12_1.ibd`.
- The operation inserts one record at a time into each index, which is very inefficient.
- InnoDB does not use a sort buffer.
- The table copy operation creates a lot fewer InnoDB undo log writes. See MDEV-11415 for more information.
- The table copy operation creates a lot of InnoDB redo log writes.

INPLACE Algorithm

The `COPY` algorithm can be incredibly slow, because the whole table has to be copied and rebuilt. The `INPLACE` algorithm was introduced as a way to avoid this by performing operations in-place and avoiding the table copy and rebuild, when possible.

When the `INPLACE` algorithm is used, the underlying storage engine uses optimizations to perform the operation while avoiding the table copy and rebuild. However, `INPLACE` is a bit of a misnomer, since some operations may still require the table to be rebuilt for some storage engines. Regardless, several operations can be performed without a full copy of the table for some storage engines.

A more accurate name for the algorithm would have been the `ENGINE` algorithm, since the storage engine decides how to implement the algorithm.

If an ALTER TABLE operation supports the `INPLACE` algorithm, then it can be performed using optimizations by the underlying storage engine, but it may rebuilt.

If the `INPLACE` algorithm is specified with the ALGORITHM clause or with the alter_algorithm system variable and if the ALTER TABLE operation does not support the `INPLACE` algorithm, then an error are raised:

```
SET SESSION alter_algorithm='INPLACE';

ALTER TABLE tab MODIFY COLUMN c INT;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

In this case, raising an error is preferable, if the alternative is for the operation to make a copy of the table, and perform unexpectedly slowly.

Using the INPLACE Algorithm with InnoDB

If the `INPLACE` algorithm is used with an InnoDB table, then the following statements apply:

- The operation might have to write sort files in the directory defined by the innodb_tmpdir system variable.
- The operation might also have to write a temporary log file to track data changes by DML queries executed during the operation. The maximum size for this log file is configured by the innodb_online_alter_log_max_size system variable.
- Some operations require the table to be rebuilt, even though the algorithm is inaccurately called "in-place". This includes operations such as adding or dropping columns, adding a primary key, changing a column to NULL, etc.
- If the operation requires the table to be rebuilt, then the operation might have to create temporary tables.

    - It may have to create a temporary intermediate table for the actual table rebuild operation.
        - This temporary table are in the same directory as the original table, and it's file name are in the format `#sql${PID}_${THREAD_ID}_${TMP_TABLE_COUNT}`, where `${PID}` is the process ID of `mysqld`, `${THREAD_ID}` is the connection ID, and `${TMP_TABLE_COUNT}` is the number of temporary tables that the connection has open. Therefore, the datadir may contain files with file names like `#sql1234_12_1.ibd`.
    - When it replaces the original table with the rebuilt table, it may also have to rename the original table using a temporary table name.
        - The innodb_safe_truncate system variable is set to `OFF`, then the format will actually be `#sql-ib${TABLESPACE_ID}-${RAND}`, where `${TABLESPACE_ID}` is the table's tablespace ID within InnoDB and `${RAND}` is a randomly initialized number. Therefore, the datadir may contain files with file names like `#sql-ib230291-1363966925.ibd`.
- The storage needed for the above items can add up to the size of the original table, or more in some cases.
- Some operations are instantaneous, if they only require the table's metadata to be changed. This includes operations such as renaming a column, changing a column's DEFAULT value, etc.

Operations Supported by InnoDB with the `INPLACE` Algorithm

With respect to the allowed operations, the `INPLACE` algorithm supports a subset of the operations supported by the `COPY` algorithm, and it supports a superset of the operations supported by the `NOCOPY` algorithm.

See InnoDB Online DDL Operations with ALGORITHM=INPLACE for more information.

NOCOPY Algorithm

The `NOCOPY` algorithm is supported. The `INPLACE` algorithm can sometimes be surprisingly slow in instances where it has to rebuild the clustered index, because when the clustered index has to be rebuilt, the whole table has to be rebuilt. The `NOCOPY` algorithm was introduced as a way to avoid this.

If an ALTER TABLE operation supports the `NOCOPY` algorithm, then it can be performed without rebuilding the clustered index.

If the `NOCOPY` algorithm is specified with the ALGORITHM clause or with the alter_algorithm system variable and if the ALTER TABLE operation does not support the `NOCOPY` algorithm, then an error are raised:

```
SET SESSION alter_algorithm='NOCOPY';

ALTER TABLE tab MODIFY COLUMN c INT;
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

In this case, raising an error is preferable, if the alternative is for the operation to rebuild the clustered index, and perform unexpectedly slowly.

Operations Supported by InnoDB with the `NOCOPY` Algorithm

With respect to the allowed operations, the `NOCOPY` algorithm supports a subset of the operations supported by the `INPLACE` algorithm, and it supports a superset of the operations supported by the `INSTANT` algorithm.

See InnoDB Online DDL Operations with ALGORITHM=NOCOPY for more information.

INSTANT Algorithm

The `INSTANT` algorithm is supported. The `INPLACE` algorithm can sometimes be surprisingly slow in instances where it has to modify data files. The `INSTANT` algorithm was introduced as a way to avoid this.

If an ALTER TABLE operation supports the `INSTANT` algorithm, then it can be performed without modifying any data files.

If the `INSTANT` algorithm is specified with the ALGORITHM clause or with the alter_algorithm system variable and if the ALTER TABLE operation does not support the `INSTANT` algorithm, then an error are raised:

```
SET SESSION alter_algorithm='INSTANT';

ALTER TABLE tab MODIFY COLUMN c INT;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

In this case, raising an error is preferable, if the alternative is for the operation to modify data files, and perform unexpectedly slowly.

Operations Supported by InnoDB with the INSTANT Algorithm

With respect to the allowed operations, the `INSTANT` algorithm supports a subset of the operations supported by the `NOCOPY` algorithm.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT for more information.

Alter Locking Strategies

InnoDB supports multiple locking strategies for performing DDL operations. This offers a significant performance improvement over previous versions. The supported locking strategies are:

- `DEFAULT` - This implies the default behavior for the specific operation.
- `NONE`
- `SHARED`
- `EXCLUSIVE`

Regardless of which locking strategy is used to perform a DDL operation, InnoDB will have to exclusively lock the table for a short time at the start and end of the operation's execution. This means that any active transactions that may have accessed the table must be committed or aborted for the operation to continue. This applies to most DDL statements, such as ALTER TABLE, CREATE INDEX, DROP INDEX, OPTIMIZE TABLE, RENAME TABLE, etc.

Specifying an Alter Locking Strategy

Specifying an Alter Locking Strategy Using the `LOCK` Clause

The ALTER TABLE statement supports the LOCK clause.

The LOCK clause can be used to specify the locking strategy that the user is willing to accept. It is supported by the ALTER TABLE and CREATE INDEX statements.

For example, if a user wanted to add a column to a table, but only if the operation is non-locking, then they could execute the following:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

ALTER TABLE tab ADD COLUMN c VARCHAR(50), ALGORITHM=INPLACE, LOCK=NONE;
```

If the LOCK clause is not explicitly set, then the operation uses `LOCK=DEFAULT`.

Specifying an Alter Locking Strategy Using `ALTER ONLINE TABLE`

ALTER ONLINE TABLE is equivalent to `LOCK=NONE`. Therefore, the ALTER ONLINE TABLE statement can be used to ensure that your ALTER TABLE operation allows all concurrent DML.

Supported Alter Locking Strategies

The supported algorithms are described in more details below.

To see which locking strategies InnoDB supports for each operation, see the pages that describe which operations are supported for each algorithm:

- InnoDB Online DDL Operations with ALGORITHM=INPLACE
- InnoDB Online DDL Operations with ALGORITHM=NOCOPY
- InnoDB Online DDL Operations with ALGORITHM=INSTANT

DEFAULT Locking Strategy

The default behavior, which occurs if `LOCK=DEFAULT` is specified, or if `LOCK` is not specified at all, acquire the least restrictive lock on the table that is supported for the specific operation. This permits the maximum amount of concurrency that is supported for the specific operation.

NONE Locking Strategy

The `NONE` locking strategy performs the operation without acquiring any lock on the table. This permits **all** concurrent DML.

If this locking strategy is not permitted for an operation, then an error is raised.

SHARED Locking Strategy

The `SHARED` locking strategy performs the operation after acquiring a read lock on the table. This permit **read-only** concurrent DML.

If this locking strategy is not permitted for an operation, then an error is raised.

EXCLUSIVE Locking Strategy

The `EXCLUSIVE` locking strategy performs the operation after acquiring a write lock on the table. This does **not** permit concurrent DML.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Online DDL Operations with the INPLACE Alter Algorithm

Supported Operations by Inheritance

When the ALGORITHM clause is set to `INPLACE`, the supported operations are a superset of the operations that are supported when the ALGORITHM clause is set to `NOCOPY`. Similarly, when the ALGORITHM clause is set to `NOCOPY`, the supported operations are a superset of the operations that are supported when the ALGORITHM clause is set to `INSTANT`.

Therefore, when the ALGORITHM clause is set to `INPLACE`, some operations are supported by inheritance. See the following additional pages for more information about these supported operations:

- InnoDB Online DDL Operations with ALGORITHM=NOCOPY
- InnoDB Online DDL Operations with ALGORITHM=INSTANT

Column Operations

`ALTER TABLE ... ADD COLUMN`

InnoDB supports adding columns to a table with ALGORITHM set to `INPLACE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

With the exception of adding an auto-increment column, this operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD COLUMN c VARCHAR(50);
Query OK, 0 rows affected (0.006 sec)
```

This applies to ALTER TABLE … ADD COLUMN for InnoDB tables.

`ALTER TABLE ... DROP COLUMN`

InnoDB supports dropping columns from a table with ALGORITHM set to `INPLACE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP COLUMN c;
Query OK, 0 rows affected (0.021 sec)
```

This applies to ALTER TABLE … DROP COLUMN for InnoDB tables.

`ALTER TABLE ... MODIFY COLUMN`

This applies to ALTER TABLE … MODIFY COLUMN for InnoDB tables.

Reordering Columns

InnoDB supports reordering columns within a table with ALGORITHM set to `INPLACE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(50) AFTER a;
Query OK, 0 rows affected (0.022 sec)
```

Changing the Data Type of a Column

InnoDB does **not** support modifying a column's data type with ALGORITHM set to `INPLACE` in most cases. There are some exceptions:

- In MariaDB 10.2.2 and later, InnoDB supports increasing the length of `VARCHAR` columns with ALGORITHM set to `INPLACE`, unless it would require changing the number of bytes requires to represent the column's length. A `VARCHAR` column that is between 0 and 255 bytes in size requires 1 byte to represent its length, while a `VARCHAR` column that is 256 bytes or longer requires 2 bytes to represent its length. This means that the length of a column cannot be increased with ALGORITHM set to `INPLACE` if the original length was less than 256 bytes, and the new length is 256 bytes or more.
- In MariaDB 10.4.3 and later, InnoDB supports increasing the length of `VARCHAR` columns with ALGORITHM set to `INPLACE` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Changing the Data Type of a Column for more information.

For example, this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c INT;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

But this succeeds in MariaDB 10.2.2 and later, because the original length of the column is less than 256 bytes, and the new length is still less than 256 bytes:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) CHARACTER SET=latin1;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(100);
Query OK, 0 rows affected (0.005 sec)
```

But this fails in MariaDB 10.2.2 and later, because the original length of the column is less than 256 bytes, and the new length is greater than 256 bytes:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(255)
) CHARACTER SET=latin1;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(256);
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

Changing a Column to NULL

InnoDB supports modifying a column to allow NULL values with ALGORITHM set to `INPLACE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50) NOT NULL
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(50) NULL;
Query OK, 0 rows affected (0.021 sec)
```

Changing a Column to NOT NULL

InnoDB supports modifying a column to **not** allow NULL values with ALGORITHM set to `INPLACE`. It is required for strict mode to be enabled in SQL_MODE. The operation will fail if the column contains any `NULL` values. Changes that would interfere with referential integrity are also not permitted.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(50) NOT NULL;
Query OK, 0 rows affected (0.021 sec)
```

Adding a New `ENUM` Option

InnoDB supports adding a new ENUM option to a column with ALGORITHM set to `INPLACE`. In order to add a new ENUM option with ALGORITHM set to `INPLACE`, the following requirements must be met:

- It must be added to the end of the list.
- The storage requirements must not change.

This operation only changes the table's metadata, so the table does not have to be rebuilt..

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c ENUM('red', 'green')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c ENUM('red', 'green', 'blue');
Query OK, 0 rows affected (0.004 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c ENUM('red', 'green')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c ENUM('red', 'blue', 'green');
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

Adding a New `SET` Option

InnoDB supports adding a new SET option to a column with ALGORITHM set to `INPLACE`. In order to add a new SET option with ALGORITHM set to `INPLACE`, the following requirements must be met:

- It must be added to the end of the list.
- The storage requirements must not change.

This operation only changes the table's metadata, so the table does not have to be rebuilt..

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c SET('red', 'green')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c SET('red', 'green', 'blue');
Query OK, 0 rows affected (0.004 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c SET('red', 'green')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c SET('red', 'blue', 'green');
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

Removing System Versioning from a Column

In MariaDB 10.3.8 and later, InnoDB supports removing system versioning from a column with ALGORITHM set to `INPLACE`. In order for this to work, the system_versioning_alter_history system variable must be set to `KEEP`. See MDEV-16330 for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50) WITH SYSTEM VERSIONING
);

SET SESSION system_versioning_alter_history='KEEP';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(50) WITHOUT SYSTEM VERSIONING;
Query OK, 0 rows affected (0.005 sec)
```

`ALTER TABLE ... ALTER COLUMN`

This applies to ALTER TABLE … ALTER COLUMN for InnoDB tables.

Setting a Column's Default Value

InnoDB supports modifying a column's DEFAULT value with ALGORITHM set to `INPLACE`.

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.
For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ALTER COLUMN c SET DEFAULT 'NO value explicitly provided.';
Query OK, 0 rows affected (0.005 sec)
```

Removing a Column's Default Value

InnoDB supports removing a column's DEFAULT value with ALGORITHM set to `INPLACE`.

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50) DEFAULT 'NO value explicitly provided.'
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ALTER COLUMN c DROP DEFAULT;
Query OK, 0 rows affected (0.005 sec)
```

`ALTER TABLE ... CHANGE COLUMN`

InnoDB supports renaming a column with ALGORITHM set to `INPLACE`, unless the column's data type or attributes changed in addition to the name.

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab CHANGE COLUMN c str VARCHAR(50);
Query OK, 0 rows affected (0.006 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab CHANGE COLUMN c num INT;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

This applies to ALTER TABLE … CHANGE COLUMN for InnoDB tables.

Index Operations

`ALTER TABLE ... ADD PRIMARY KEY`

InnoDB supports adding a primary key to a table with ALGORITHM set to `INPLACE`.

If the new primary key column is not defined as NOT NULL, then it is highly recommended for strict mode to be enabled in SQL_MODE. Otherwise, `NULL` values are silently converted to the default value for the given data type, which is probably not the desired behavior in this scenario.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD PRIMARY KEY (a);
Query OK, 0 rows affected (0.021 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT,
   b VARCHAR(50),
   c VARCHAR(50)
);

INSERT INTO tab VALUES (NULL, NULL, NULL);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD PRIMARY KEY (a);
ERROR 1265 (01000): Data truncated for column 'a' at row 1
```

And this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT,
   b VARCHAR(50),
   c VARCHAR(50)
);

INSERT INTO tab VALUES (1, NULL, NULL);
INSERT INTO tab VALUES (1, NULL, NULL);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD PRIMARY KEY (a);
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

This applies to ALTER TABLE … ADD PRIMARY KEY for InnoDB tables.

`ALTER TABLE ... DROP PRIMARY KEY`

InnoDB does **not** support dropping a primary key with ALGORITHM set to `INPLACE` in most cases.

If you try to do so, then you will see an error. InnoDB only supports this operation with ALGORITHM set to `COPY`. Concurrent DML is *not* permitted.

However, there is an exception. If you are dropping a primary key, and adding a new one at the same time, then that operation can be performed with ALGORITHM set to `INPLACE`. This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP PRIMARY KEY;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Dropping a primary key is not allowed
without also adding a new primary key. Try ALGORITHM=COPY
```

But this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP PRIMARY KEY, ADD PRIMARY KEY (b);
Query OK, 0 rows affected (0.020 sec)
```

This applies to ALTER TABLE … DROP PRIMARY KEY for InnoDB tables.

`ALTER TABLE ... ADD INDEX` and `CREATE INDEX`

This applies to ALTER TABLE … ADD INDEX and CREATE INDEX for InnoDB tables.

Adding a Plain Index

InnoDB supports adding a plain index to a table with ALGORITHM set to `INPLACE`. The table is not rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD INDEX b_index (b);
Query OK, 0 rows affected (0.010 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
CREATE INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.011 sec)
```

Adding a Fulltext Index

InnoDB supports adding a FULLTEXT index to a table with ALGORITHM set to `INPLACE`. The table is not rebuilt in some cases.

However, there are some limitations, such as:

- Adding a FULLTEXT index to a table that does not have a user-defined `FTS_DOC_ID` column will require the table to be rebuilt once. When the table is rebuilt, the system adds a hidden `FTS_DOC_ID` column. From that point forward, adding additional FULLTEXT indexes to the same table will not require the table to be rebuilt when ALGORITHM is set to `INPLACE`.
- Only one FULLTEXT index may be added at a time when ALGORITHM is set to `INPLACE`.
- If a table has more than one FULLTEXT index, then it cannot be rebuilt by any ALTER TABLE operations when ALGORITHM is set to `INPLACE`.
- If a table has a FULLTEXT index, then it cannot be rebuilt by any ALTER TABLE operations when the LOCK clause is set to `NONE`.

This operation supports a read-only locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `SHARED`. When this strategy is used, read-only concurrent DML is permitted.

For example, this succeeds, but requires the table to be rebuilt, so that the hidden `FTS_DOC_ID` column can be added:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.055 sec)
```

And this succeeds in the same way as above:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
CREATE FULLTEXT INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.041 sec)
```

And this succeeds, and the second command does not require the table to be rebuilt:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.043 sec)

ALTER TABLE tab ADD FULLTEXT INDEX c_index (c);
Query OK, 0 rows affected (0.017 sec)
```

But this second command fails, because only one FULLTEXT index can be added at a time:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   d VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.041 sec)

ALTER TABLE tab ADD FULLTEXT INDEX c_index (c), ADD FULLTEXT INDEX d_index (d);
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: InnoDB presently supports one FULLTEXT
index creation at a time. Try ALGORITHM=COPY
```

And this third command fails, because a table cannot be rebuilt when it has more than one FULLTEXT index:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.040 sec)

ALTER TABLE tab ADD FULLTEXT INDEX c_index (c);
Query OK, 0 rows affected (0.015 sec)

ALTER TABLE tab FORCE;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: InnoDB presently supports one FULLTEXT
index creation at a time. Try ALGORITHM=COPY
```

Adding a Spatial Index

InnoDB supports adding a SPATIAL index to a table with ALGORITHM set to `INPLACE`.

However, there are some limitations, such as:

- If a table has a SPATIAL index, then it cannot be rebuilt by any ALTER TABLE operations when the LOCK clause is set to `NONE`.

This operation supports a read-only locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `SHARED`. When this strategy is used, read-only concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD SPATIAL INDEX c_index (c);
Query OK, 0 rows affected (0.006 sec)
```

And this succeeds in the same way as above:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='INPLACE';
CREATE SPATIAL INDEX c_index ON tab (c);
Query OK, 0 rows affected (0.006 sec)
```

`ALTER TABLE ... DROP INDEX` and `DROP INDEX`

InnoDB supports dropping indexes from a table with ALGORITHM set to `INPLACE`.

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   INDEX b_index (b)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP INDEX b_index;
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   INDEX b_index (b)
);

SET SESSION alter_algorithm='INPLACE';
DROP INDEX b_index ON tab;
```

This applies to ALTER TABLE … DROP INDEX and DROP INDEX for InnoDB tables.

`ALTER TABLE ... ADD FOREIGN KEY`

InnoDB supports adding foreign key constraints to a table with ALGORITHM set to `INPLACE`. In order to add a new foreign key constraint to a table with ALGORITHM set to `INPLACE`, the foreign_key_checks system variable needs to be set to `OFF`. If it is set to `ON`, then `ALGORITHM=COPY` is required.

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this fails:

```
CREATE OR REPLACE TABLE tab1 (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   d INT
);

CREATE OR REPLACE TABLE tab2 (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Adding foreign keys needs
foreign_key_checks=OFF. Try ALGORITHM=COPY
```

But this succeeds:

```
CREATE OR REPLACE TABLE tab1 (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   d INT
);

CREATE OR REPLACE TABLE tab2 (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

SET SESSION foreign_key_checks=OFF;
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
Query OK, 0 rows affected (0.011 sec)
```

This applies to ALTER TABLE … ADD FOREIGN KEY for InnoDB tables.

`ALTER TABLE ... DROP FOREIGN KEY`

InnoDB supports dropping foreign key constraints from a table with ALGORITHM set to `INPLACE`.

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab2 (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

CREATE OR REPLACE TABLE tab1 (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   d INT,
   FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab1 DROP FOREIGN KEY tab2_fk;
Query OK, 0 rows affected (0.005 sec)
```

This applies to ALTER TABLE … DROP FOREIGN KEY for InnoDB tables.

Table Operations

`ALTER TABLE ... AUTO_INCREMENT=...`

InnoDB supports changing a table's AUTO_INCREMENT value with ALGORITHM set to `INPLACE`. This operation should finish instantly. The table is not rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab AUTO_INCREMENT=100;
Query OK, 0 rows affected (0.004 sec)
```

This applies to ALTER TABLE … AUTO_INCREMENT=… for InnoDB tables.

`ALTER TABLE ... ROW_FORMAT=...`

InnoDB supports changing a table's row format with ALGORITHM set to `INPLACE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) ROW_FORMAT=DYNAMIC;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ROW_FORMAT=COMPRESSED;
Query OK, 0 rows affected (0.025 sec)
```

This applies to ALTER TABLE … ROW_FORMAT=… for InnoDB tables.

`ALTER TABLE ... KEY_BLOCK_SIZE=...`

InnoDB supports changing a table's KEY_BLOCK_SIZE with ALGORITHM set to `INPLACE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) ROW_FORMAT=COMPRESSED
  KEY_BLOCK_SIZE=4;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab KEY_BLOCK_SIZE=2;
Query OK, 0 rows affected (0.021 sec)
```

This applies to KEY_BLOCK_SIZE=... for InnoDB tables.

`ALTER TABLE ... PAGE_COMPRESSED=...` and `ALTER TABLE ... PAGE_COMPRESSION_LEVEL=...`

In MariaDB 10.3.10 and later, InnoDB supports setting a table's PAGE_COMPRESSED value to `1` with ALGORITHM set to `INPLACE`. InnoDB also supports changing a table's PAGE_COMPRESSED value from `1` to `0` with ALGORITHM set to `INPLACE`.

In these versions, InnoDB also supports changing a table's PAGE_COMPRESSION_LEVEL value with ALGORITHM set to `INPLACE`.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

See MDEV-16328 for more information.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab PAGE_COMPRESSED=1;
Query OK, 0 rows affected (0.006 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) PAGE_COMPRESSED=1;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab PAGE_COMPRESSED=0;
Query OK, 0 rows affected (0.020 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) PAGE_COMPRESSED=1
  PAGE_COMPRESSION_LEVEL=5;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab PAGE_COMPRESSION_LEVEL=4;
Query OK, 0 rows affected (0.006 sec)
```

This applies to PAGE_COMPRESSED=… and PAGE_COMPRESSION_LEVEL=… for InnoDB tables.

`ALTER TABLE ... DROP SYSTEM VERSIONING`

InnoDB supports dropping system versioning from a table with ALGORITHM set to `INPLACE`.

This operation supports the read-only locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `SHARED`. When this strategy is used, read-only concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) WITH SYSTEM VERSIONING;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP SYSTEM VERSIONING;
```

This applies to ALTER TABLE … DROP SYSTEM VERSIONING for InnoDB tables.

`ALTER TABLE ... DROP CONSTRAINT`

In MariaDB 10.3.6 and later, InnoDB supports dropping a CHECK constraint from a table with ALGORITHM set to `INPLACE`. See MDEV-16331 for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   CONSTRAINT b_not_empty CHECK (b != '')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP CONSTRAINT b_not_empty;
Query OK, 0 rows affected (0.004 sec)
```

This applies to ALTER TABLE … DROP CONSTRAINT for InnoDB tables.

`ALTER TABLE ... FORCE`

InnoDB supports forcing a table rebuild with ALGORITHM set to `INPLACE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab FORCE;
Query OK, 0 rows affected (0.022 sec)
```

This applies to ALTER TABLE … FORCE for InnoDB tables.

`ALTER TABLE ... ENGINE=InnoDB`

InnoDB supports forcing a table rebuild with ALGORITHM set to `INPLACE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ENGINE=InnoDB;
Query OK, 0 rows affected (0.022 sec)
```

This applies to ALTER TABLE … ENGINE=InnoDB for InnoDB tables.

`OPTIMIZE TABLE ...`

InnoDB supports optimizing a table with ALGORITHM set to `INPLACE`.

If the innodb_defragment system variable is set to `OFF`, and if the innodb_optimize_fulltext_only system variable is also set to `OFF`, then `OPTIMIZE TABLE` are equivalent to `ALTER TABLE … FORCE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

If either of the previously mentioned system variables is set to `ON`, then `OPTIMIZE TABLE` will optimize some data without rebuilding the table. However, the file size will not be reduced.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SHOW GLOBAL VARIABLES WHERE Variable_name IN('innodb_defragment', 'innodb_optimize_fulltext_only');
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| innodb_defragment              | OFF   |
| innodb_optimize_fulltext_only | OFF   |
+--------------------------------+-------+

SET SESSION alter_algorithm='INPLACE';
OPTIMIZE TABLE tab;
+---------+----------+----------+-------------------------------------------------------------------+
| Table   | Op       | Msg_type | Msg_text                                                          |
+---------+----------+----------+-------------------------------------------------------------------+
| db1.tab | optimize | note     | Table does not support optimize, doing recreate + analyze instead |
| db1.tab | optimize | status   | OK                                                                |
+---------+----------+----------+-------------------------------------------------------------------+
2 rows in set (0.026 sec)
```

And this succeeds, but the table is not rebuilt:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET GLOBAL innodb_defragment=ON;
SHOW GLOBAL VARIABLES WHERE Variable_name IN('innodb_defragment', 'innodb_optimize_fulltext_only');
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| innodb_defragment              | ON    |
| innodb_optimize_fulltext_only | OFF   |
+--------------------------------+-------+

SET SESSION alter_algorithm='INPLACE';
OPTIMIZE TABLE tab;
+---------+----------+----------+----------+
| Table   | Op       | Msg_type | Msg_text |
+---------+----------+----------+----------+
| db1.tab | optimize | status   | OK       |
+---------+----------+----------+----------+
1 row in set (0.004 sec)
```

This applies to OPTIMIZE TABLE for InnoDB tables.

`ALTER TABLE ... RENAME TO` and `RENAME TABLE ...`

InnoDB supports renaming a table with ALGORITHM set to `INPLACE`.

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the exclusive locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `EXCLUSIVE`. When this strategy is used, concurrent DML is **not** permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab RENAME TO old_tab;
Query OK, 0 rows affected (0.011 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
RENAME TABLE tab TO old_tab;
```

This applies to ALTER TABLE … RENAME TO and RENAME TABLE for InnoDB tables.

Limitations

Limitations Related to Fulltext Indexes

- If a table has more than one FULLTEXT index, then it cannot be rebuilt by any ALTER TABLE operations when ALGORITHM is set to `INPLACE`.
- If a table has a FULLTEXT index, then it cannot be rebuilt by any ALTER TABLE operations when the LOCK clause is set to `NONE`.

Limitations Related to Spatial Indexes

- If a table has a SPATIAL index, then it cannot be rebuilt by any ALTER TABLE operations when the LOCK clause is set to `NONE`.

Limitations Related to Generated (Virtual and Persistent/Stored) Columns

Generated columns do not currently support online DDL for all of the same operations that are supported for “real” columns.

See Generated (Virtual and Persistent/Stored) Columns: Statement Support for more information on the limitations.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Online DDL Operations with the INSTANT Alter Algorithm

Column Operations

`ALTER TABLE ... ADD COLUMN`

In MariaDB 10.3.2 and later, InnoDB supports adding columns to a table with ALGORITHM set to `INSTANT` if the new column is the last column in the table. See MDEV-11369 for more information. If the table has a hidden `FTS_DOC_ID` column is present, then this is not supported.

In MariaDB 10.4 and later, InnoDB supports adding columns to a table with ALGORITHM set to `INSTANT`, regardless of where in the column list the new column is added.

When this operation is performed with ALGORITHM set to `INSTANT`, the tablespace file will have a non-canonical storage format. See Non-canonical Storage Format Caused by Some Operations for more information.

With the exception of adding an auto-increment column, this operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD COLUMN c VARCHAR(50);
Query OK, 0 rows affected (0.004 sec)
```

And this succeeds in MariaDB 10.4 and later:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD COLUMN c VARCHAR(50) AFTER a;
Query OK, 0 rows affected (0.004 sec)
```

This applies to ALTER TABLE … ADD COLUMN for InnoDB tables.

See Instant ADD COLUMN for InnoDB for more information.

`ALTER TABLE ... DROP COLUMN`

In MariaDB 10.4 and later, InnoDB supports dropping columns from a table with ALGORITHM set to `INSTANT`. See MDEV-15562 for more information.

When this operation is performed with ALGORITHM set to `INSTANT`, the tablespace file will have a non-canonical storage format. See Non-canonical Storage Format Caused by Some Operations for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab DROP COLUMN c;
Query OK, 0 rows affected (0.004 sec)
```

This applies to ALTER TABLE … DROP COLUMN for InnoDB tables.

`ALTER TABLE ... MODIFY COLUMN`

This applies to ALTER TABLE … MODIFY COLUMN for InnoDB tables.

Reordering Columns

In MariaDB 10.4 and later, InnoDB supports reordering columns within a table with ALGORITHM set to `INSTANT`. See MDEV-15562 for more information.

When this operation is performed with ALGORITHM set to `INSTANT`, the tablespace file will have a non-canonical storage format. See Non-canonical Storage Format Caused by Some Operations for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(50) AFTER a;
Query OK, 0 rows affected (0.004 sec)
```

Changing the Data Type of a Column

InnoDB does **not** support modifying a column's data type with ALGORITHM set to `INSTANT` in most cases. There are some exceptions:

- InnoDB supports increasing the length of `VARCHAR` columns with ALGORITHM set to `INSTANT`, unless it would require changing the number of bytes requires to represent the column's length. A `VARCHAR` column that is between 0 and 255 bytes in size requires 1 byte to represent its length, while a `VARCHAR` column that is 256 bytes or longer requires 2 bytes to represent its length. This means that the length of a column cannot be increased with ALGORITHM set to `INSTANT` if the original length was less than 256 bytes, and the new length is 256 bytes or more.
- In MariaDB 10.4.3 and later, InnoDB supports increasing the length of `VARCHAR` columns with ALGORITHM set to `INSTANT` with no restrictions if the ROW_FORMAT table option is set to REDUNDANT. See MDEV-15563 for more information.
- In MariaDB 10.4.3 and later, InnoDB also supports increasing the length of `VARCHAR` columns with ALGORITHM set to `INSTANT` in a more limited manner if the ROW_FORMAT table option is set to COMPACT, DYNAMIC, or COMPRESSED. In this scenario, the following limitations apply:
  - The length can be increased with ALGORITHM set to `INSTANT` if the original length of the column is 127 bytes or less, and the new length of the column is 256 bytes or more.
  - The length can be increased with ALGORITHM set to `INSTANT` if the original length of the column is 255 bytes or less, and the new length of the column is still 255 bytes or less.
  - The length can be increased with ALGORITHM set to `INSTANT` if the original length of the column is 256 bytes or more, and the new length of the column is still 256 bytes or more.
  - The length can not be increased with ALGORITHM set to `INSTANT` if the original length was between 128 bytes and 255 bytes, and the new length is 256 bytes or more.
  - See MDEV-15563 for more information.

The supported operations in this category support the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c INT;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

But this succeeds because the original length of the column is less than 256 bytes, and the new length is still less than 256 bytes:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(100);
Query OK, 0 rows affected (0.005 sec)
```

But this fails because the original length of the column is between 128 bytes and 255 bytes, and the new length is greater than 256 bytes:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(255)
) CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(256);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

But this succeeds in MariaDB 10.4.3 and later because the table has `ROW_FORMAT=REDUNDANT`:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(200)
) ROW_FORMAT=REDUNDANT;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(300);
Query OK, 0 rows affected (0.004 sec)
```

And this succeeds in MariaDB 10.4.3 and later because the table has `ROW_FORMAT=DYNAMIC` and the column's original length is 127 bytes or less:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(127)
) ROW_FORMAT=DYNAMIC
  CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(300);
Query OK, 0 rows affected (0.003 sec)
```

And this succeeds in MariaDB 10.4.3 and later because the table has `ROW_FORMAT=COMPRESSED` and the column's original length is 127 bytes or less:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(127)
) ROW_FORMAT=COMPRESSED
  CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(300);
Query OK, 0 rows affected (0.003 sec)
```

But this fails even in MariaDB 10.4.3 and later because the table has `ROW_FORMAT=DYNAMIC` and the column's original length is between 128 bytes and 255 bytes:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(128)
) ROW_FORMAT=DYNAMIC
  CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(300);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

Changing a Column to NULL

In MariaDB 10.4.3 and later, InnoDB supports modifying a column to allow NULL values with ALGORITHM set to `INSTANT` if the ROW_FORMAT table option is set to REDUNDANT. See MDEV-15563 for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50) NOT NULL
) ROW_FORMAT=REDUNDANT;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(50) NULL;
Query OK, 0 rows affected (0.004 sec)
```

Changing a Column to NOT NULL

InnoDB does **not** support modifying a column to **not** allow NULL values with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) ROW_FORMAT=REDUNDANT;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(50) NOT NULL;
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE
```

Adding a New `ENUM` Option

InnoDB supports adding a new ENUM option to a column with ALGORITHM set to `INSTANT`. In order to add a new ENUM option with ALGORITHM set to `INSTANT`, the following requirements must be met:

- It must be added to the end of the list.
- The storage requirements must not change.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c ENUM('red', 'green')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c ENUM('red', 'green', 'blue');
Query OK, 0 rows affected (0.002 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c ENUM('red', 'green')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c ENUM('red', 'blue', 'green');
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

Adding a New `SET` Option

InnoDB supports adding a new SET option to a column with ALGORITHM set to `INSTANT`. In order to add a new SET option with ALGORITHM set to `INSTANT`, the following requirements must be met:

- It must be added to the end of the list.
- The storage requirements must not change.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c SET('red', 'green')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c SET('red', 'green', 'blue');
Query OK, 0 rows affected (0.002 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c SET('red', 'green')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c SET('red', 'blue', 'green');
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

Removing System Versioning from a Column

In MariaDB 10.3.8 and later, InnoDB supports removing system versioning from a column with ALGORITHM set to `INSTANT`. In order for this to work, the system_versioning_alter_history system variable must be set to `KEEP`. See MDEV-16330 for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50) WITH SYSTEM VERSIONING
);

SET SESSION system_versioning_alter_history='KEEP';
SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(50) WITHOUT SYSTEM VERSIONING;
Query OK, 0 rows affected (0.004 sec)
```

`ALTER TABLE ... ALTER COLUMN`

This applies to ALTER TABLE … ALTER COLUMN for InnoDB tables.

Setting a Column's Default Value

InnoDB supports modifying a column's DEFAULT value with ALGORITHM set to `INSTANT`.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ALTER COLUMN c SET DEFAULT 'NO value explicitly provided.';
Query OK, 0 rows affected (0.003 sec)
```

Removing a Column's Default Value

InnoDB supports removing a column's DEFAULT value with ALGORITHM set to `INSTANT`.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50) DEFAULT 'NO value explicitly provided.'
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ALTER COLUMN c DROP DEFAULT;
Query OK, 0 rows affected (0.002 sec)
```

`ALTER TABLE ... CHANGE COLUMN`

InnoDB supports renaming a column with ALGORITHM set to `INSTANT`, unless the column's data type or attributes changed in addition to the name.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab CHANGE COLUMN c str VARCHAR(50);
Query OK, 0 rows affected (0.004 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab CHANGE COLUMN c num INT;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try
ALGORITHM=COPY
```

This applies to ALTER TABLE … CHANGE COLUMN for InnoDB tables.

Index Operations

`ALTER TABLE ... ADD PRIMARY KEY`

InnoDB does **not** support adding a primary key to a table with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD PRIMARY KEY (a);
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … ADD PRIMARY KEY for InnoDB tables.

`ALTER TABLE ... DROP PRIMARY KEY`

InnoDB does **not** support dropping a primary key with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab DROP PRIMARY KEY;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Dropping a primary key is not allowed
without also adding a new primary key. Try ALGORITHM=COPY
```

This applies to ALTER TABLE … DROP PRIMARY KEY for InnoDB tables.

`ALTER TABLE ... ADD INDEX` and `CREATE INDEX`

This applies to ALTER TABLE … ADD INDEX and CREATE INDEX for InnoDB tables.

Adding a Plain Index

InnoDB does **not** support adding a plain index to a table with ALGORITHM set to `INSTANT`.

For example, this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD INDEX b_index (b);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

And this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
CREATE INDEX b_index ON tab (b);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

Adding a Fulltext Index

InnoDB does **not** support adding a FULLTEXT index to a table with ALGORITHM set to `INSTANT`.

For example, this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.042 sec)

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD FULLTEXT INDEX c_index (c);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

And this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
CREATE FULLTEXT INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.040 sec)

SET SESSION alter_algorithm='INSTANT';
CREATE FULLTEXT INDEX c_index ON tab (c);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

Adding a Spatial Index

InnoDB does **not** support adding a SPATIAL index to a table with ALGORITHM set to `INSTANT`.

For example, this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD SPATIAL INDEX c_index (c);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

And this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='INSTANT';
CREATE SPATIAL INDEX c_index ON tab (c);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

`ALTER TABLE ... ADD FOREIGN KEY`

InnoDB does **not** support adding foreign key constraints to a table with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab1 (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   d INT
);

CREATE OR REPLACE TABLE tab2 (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

SET SESSION foreign_key_checks=OFF;
SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

This applies to ALTER TABLE … ADD FOREIGN KEY for InnoDB tables.

`ALTER TABLE ... DROP FOREIGN KEY`

InnoDB supports dropping foreign key constraints from a table with ALGORITHM set to `INSTANT`.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab2 (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

CREATE OR REPLACE TABLE tab1 (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   d INT,
   FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab1 DROP FOREIGN KEY tab2_fk;
Query OK, 0 rows affected (0.004 sec)
```

This applies to ALTER TABLE … DROP FOREIGN KEY for InnoDB tables.

Table Operations

`ALTER TABLE ... AUTO_INCREMENT=...`

InnoDB supports changing a table's AUTO_INCREMENT value with ALGORITHM set to `INSTANT`.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab AUTO_INCREMENT=100;
Query OK, 0 rows affected (0.002 sec)
```

This applies to ALTER TABLE … AUTO_INCREMENT=… for InnoDB tables.

`ALTER TABLE ... ROW_FORMAT=...`

InnoDB does **not** support changing a table's row format with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) ROW_FORMAT=DYNAMIC;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ROW_FORMAT=COMPRESSED;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Changing table options requires the table
to be rebuilt. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … ROW_FORMAT=… for InnoDB tables.

`ALTER TABLE ... KEY_BLOCK_SIZE=...`

InnoDB does **not** support changing a table's KEY_BLOCK_SIZE with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) ROW_FORMAT=COMPRESSED
  KEY_BLOCK_SIZE=4;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab KEY_BLOCK_SIZE=2;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Changing table options requires the table
to be rebuilt. Try ALGORITHM=INPLACE
```

This applies to KEY_BLOCK_SIZE=… for InnoDB tables.

`ALTER TABLE ... PAGE_COMPRESSED=1` and `ALTER TABLE ... PAGE_COMPRESSION_LEVEL=...`

In MariaDB 10.3.10 and later, InnoDB supports setting a table's PAGE_COMPRESSED value to `1` with ALGORITHM set to `INSTANT`. InnoDB does **not** support changing a table's PAGE_COMPRESSED value from `1` to `0` with ALGORITHM set to `INSTANT`.

In these versions, InnoDB also supports changing a table's PAGE_COMPRESSION_LEVEL value with ALGORITHM set to `INSTANT`.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

See MDEV-16328 for more information.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab PAGE_COMPRESSED=1;
Query OK, 0 rows affected (0.004 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) PAGE_COMPRESSED=1
  PAGE_COMPRESSION_LEVEL=5;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab PAGE_COMPRESSION_LEVEL=4;
Query OK, 0 rows affected (0.004 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) PAGE_COMPRESSED=1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab PAGE_COMPRESSED=0;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Changing table options requires the table
to be rebuilt. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … PAGE_COMPRESSED=… and ALTER TABLE … PAGE_COMPRESSION_LEVEL=… for InnoDB tables.

`ALTER TABLE ... DROP SYSTEM VERSIONING`

InnoDB does **not** support dropping system versioning from a table with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) WITH SYSTEM VERSIONING;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab DROP SYSTEM VERSIONING;
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … DROP SYSTEM VERSIONING for InnoDB tables.

`ALTER TABLE ... DROP CONSTRAINT`

In MariaDB 10.3.6 and later, InnoDB supports dropping a CHECK constraint from a table with ALGORITHM set to `INSTANT`. See MDEV-16331 for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   CONSTRAINT b_not_empty CHECK (b != '')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab DROP CONSTRAINT b_not_empty;
Query OK, 0 rows affected (0.002 sec)
```

This applies to ALTER TABLE … DROP CONSTRAINT for InnoDB tables.

`ALTER TABLE ... FORCE`

InnoDB does **not** support forcing a table rebuild with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab FORCE;
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … FORCE for InnoDB tables.

`ALTER TABLE ... ENGINE=InnoDB`

InnoDB does **not** support forcing a table rebuild with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ENGINE=InnoDB;
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … ENGINE=InnoDB for InnoDB tables.

`OPTIMIZE TABLE ...`

InnoDB does **not** support optimizing a table with ALGORITHM set to `INSTANT`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SHOW GLOBAL VARIABLES WHERE Variable_name IN('innodb_defragment', 'innodb_optimize_fulltext_only');
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| innodb_defragment             | OFF   |
| innodb_optimize_fulltext_only | OFF   |
+-------------------------------+-------+
2 rows in set (0.001 sec)

SET SESSION alter_algorithm='INSTANT';
OPTIMIZE TABLE tab;
+---------+----------+----------+-------------------------------------------------------------------------------
------+
| Table   | Op       | Msg_type | Msg_text
|
+---------+----------+----------+-------------------------------------------------------------------------------
------+
| db1.tab | optimize | note     | Table does not support optimize, doing recreate + analyze instead
|
| db1.tab | optimize | error    | ALGORITHM=INSTANT is not supported for this operation. Try
ALGORITHM=INPLACE |
| db1.tab | optimize | status   | Operation failed
|
+---------+----------+----------+-------------------------------------------------------------------------------
------+
3 rows in set, 1 warning (0.002 sec)
```

This applies to OPTIMIZE TABLE for InnoDB tables.

`ALTER TABLE ... RENAME TO` and `RENAME TABLE ...`

InnoDB supports renaming a table with ALGORITHM set to `INSTANT`.

This operation supports the exclusive locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `EXCLUSIVE`. When this strategy is used, concurrent DML is **not** permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab RENAME TO old_tab;
Query OK, 0 rows affected (0.008 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INSTANT';
RENAME TABLE tab TO old_tab;
Query OK, 0 rows affected (0.008 sec)
```

This applies to ALTER TABLE … RENAME TO and RENAME TABLE for InnoDB tables.

Limitations

Limitations Related to Generated (Virtual and Persistent/Stored) Columns

Generated columns do not currently support online DDL for all of the same operations that are supported for “real” columns.

See Generated (Virtual and Persistent/Stored) Columns: Statement Support for more information on the limitations.

Non-canonical Storage Format Caused by Some Operations

Some operations cause a table’s tablespace file to use a non-canonical storage format when the `INSTANT` algorithm is used. The affected operations include:

- Adding a column.
- Dropping a column.
- Reordering columns.

These operations require the following non-canonical changes to the storage format:

- A hidden metadata record at the start of the clustered index is used to store each column’s DEFAULT value. This makes it possible to add new columns that have default values without rebuilding the table.
- A BLOB in the hidden metadata record is used to store column mappings. This makes it possible to drop or reorder columns without rebuilding the table. This also makes it possible to add columns to any position or drop columns from any position in the table without rebuilding the table.
- If a column is dropped, old records will contain garbage in that column’s former position, and new records are written with NULL values, empty strings, or dummy values.

This non-canonical storage format has the potential to incur some performance or storage overhead for all subsequent DML operations. If you notice some issues like this and you want to normalize a table’s storage format to avoid this problem, then you can do so by forcing a table rebuild by executing ALTER TABLE … FORCE with ALGORITHM set to `INPLACE`:

```
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab FORCE;
Query OK, 0 rows affected (0.008 sec)
```

However, keep in mind that there are certain scenarios where you may not be able to rebuild the table with ALGORITHM set to `INPLACE`. See InnoDB Online DDL Operations with ALGORITHM=INPLACE: Limitations for more information on those cases. If you hit one of those scenarios, but you still want to rebuild the table, then you would have to do so with ALGORITHM set to `COPY`.

Known Bugs

There are some known bugs that could lead to issues when an InnoDB DDL operation is performed using the INSTANT algorithm. This algorithm will usually be chosen by default if the operation supports the algorithm.

The effect of many of these bugs is that the table seems to *forget* that its tablespace file is in the non-canonical storage format.

If you are concerned that a table may be affected by one of these bugs, then your best option would be to normalize the table structure. This can be done by rebuilding the table:

```
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab FORCE;
Query OK, 0 rows affected (0.008 sec)
```

If you are concerned about these bugs, and you want to perform an operation that supports the INSTANT algorithm, but you want to avoid using that algorithm, then you can set the algorithm to INPLACE and add the `FORCE` keyword to the ALTER TABLE statement:

```
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD COLUMN c VARCHAR(50), FORCE;
```

Closed Bugs

- MDEV-20066: This bug could cause a table to become corrupt if a column was added instantly. It is fixed in MariaDB 10.3.18 and MariaDB 10.4.8.
- MDEV-20117: This bug could cause a table to become corrupt if a column was dropped instantly. It is fixed in MariaDB 10.4.9.
- MDEV-19743: This bug could cause a table to become corrupt during page reorganization if a column was added instantly. It is fixed in MariaDB 10.3.17 and MariaDB 10.4.7.
- MDEV-19783: This bug could cause a table to become corrupt if a column was added instantly. It is fixed in MariaDB 10.3.17 and MariaDB 10.4.7
- MDEV-20090: This bug could cause a table to become corrupt if columns were added, dropped, or reordered instantly. It is fixed in MariaDB 10.4.9.
- MDEV-18519: This bug could cause a table to become corrupt if a column was added instantly. It is fixed in MariaDB 10.6.9, MariaDB 10.7.5, MariaDB 10.8.4 and MariaDB 10.9.2.
- MDEV-18519: This bug could cause a table to become corrupt if a column was added instantly. This isn't and won't be fixed in versions less than MariaDB 10.6.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Online DDL Operations with the NOCOPY Alter Algorithm

Supported Operations by Inheritance

When the ALGORITHM clause is set to `NOCOPY`, the supported operations are a superset of the operations that are supported when the ALGORITHM clause is set to `INSTANT`.

Therefore, when the ALGORITHM clause is set to `NOCOPY`, some operations are supported by inheritance. See the following additional pages for more information about these supported operations:

- InnoDB Online DDL Operations with ALGORITHM=INSTANT

Column Operations

`ALTER TABLE ... ADD COLUMN`

In MariaDB 10.3.2 and later, InnoDB supports adding columns to a table with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE … ADD COLUMN for more information.

This applies to ALTER TABLE … ADD COLUMN for InnoDB tables.

`ALTER TABLE ... DROP COLUMN`

In MariaDB 10.4 and later, InnoDB supports dropping columns from a table with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE … DROP COLUMN for more information.

This applies to ALTER TABLE … DROP COLUMN for InnoDB tables.

`ALTER TABLE ... MODIFY COLUMN`

This applies to ALTER TABLE … MODIFY COLUMN for InnoDB tables.

Reordering Columns

In MariaDB 10.4 and later, InnoDB supports reordering columns within a table with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Reordering Columns for more information.

Changing the Data Type of a Column

InnoDB does **not** support modifying a column's data type with ALGORITHM set to `NOCOPY` in most cases. There are a few exceptions in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Changing the Data Type of a Column for more information.

Changing a Column to NULL

In MariaDB 10.4.3 and later, InnoDB supports modifying a column to allow NULL values with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Changing a Column to NULL for more information.

Changing a Column to NOT NULL

InnoDB does **not** support modifying a column to **not** allow NULL values with ALGORITHM set to `NOCOPY`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) ROW_FORMAT=REDUNDANT;

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab MODIFY COLUMN c VARCHAR(50) NOT NULL;
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE
```

Adding a New `ENUM` Option

InnoDB supports adding a new ENUM option to a column with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Adding a New ENUM Option for more information.

Adding a New `SET` Option

InnoDB supports adding a new SET option to a column with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Adding a New SET Option for more information.

Removing System Versioning from a Column

In MariaDB 10.3.8 and later, InnoDB supports removing system versioning from a column with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Removing System Versioning from a Column for more information.

`ALTER TABLE ... ALTER COLUMN`

This applies to ALTER TABLE … ALTER COLUMN for InnoDB tables.

Setting a Column's Default Value

InnoDB supports modifying a column's DEFAULT value with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Setting a Column's Default Value for more information.

Removing a Column's Default Value

InnoDB supports removing a column's DEFAULT value with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Removing a Column's Default Value for more information.

`ALTER TABLE ... CHANGE COLUMN`

InnoDB supports renaming a column with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE … CHANGE COLUMN for more information.

This applies to ALTER TABLE … CHANGE COLUMN for InnoDB tables.

Index Operations

`ALTER TABLE ... ADD PRIMARY KEY`

InnoDB does **not** support adding a primary key to a table with ALGORITHM set to `NOCOPY`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD PRIMARY KEY (a);
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … ADD PRIMARY KEY for InnoDB tables.

`ALTER TABLE ... DROP PRIMARY KEY`

InnoDB does **not** support dropping a primary key with ALGORITHM set to `NOCOPY`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab DROP PRIMARY KEY;
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Dropping a primary key is not allowed
without also adding a new primary key. Try ALGORITHM=COPY
```

This applies to ALTER TABLE … DROP PRIMARY KEY for InnoDB tables.

`ALTER TABLE ... ADD INDEX` and `CREATE INDEX`

This applies to ALTER TABLE … ADD INDEX and CREATE INDEX for InnoDB tables.

Adding a Plain Index

InnoDB supports adding a plain index to a table with ALGORITHM set to `NOCOPY`.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD INDEX b_index (b);
Query OK, 0 rows affected (0.009 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='NOCOPY';
CREATE INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.009 sec)
```

Adding a Fulltext Index

InnoDB supports adding a FULLTEXT index to a table with ALGORITHM set to `NOCOPY`.

However, there are some limitations, such as:

- Adding a FULLTEXT index to a table that does not have a user-defined `FTS_DOC_ID` column will require the table to be rebuilt once. When the table is rebuilt, the system adds a hidden `FTS_DOC_ID` column. This initial operation will have to be performed with ALGORITHM set to `INPLACE`.From that point forward, adding additional FULLTEXT indexes to the same table will not require the table to be rebuilt, and ALGORITHM can be set to `NOCOPY`.
- Only one FULLTEXT index may be added at a time when ALGORITHM is set to `NOCOPY`.

This operation supports a read-only locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `SHARED`. When this strategy is used, read-only concurrent DML is permitted.

For example, this succeeds, but the first operation requires the table to be rebuilt ALGORITHM set to `INPLACE`, so that the hidden `FTS_DOC_ID` column can be added:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.043 sec)

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD FULLTEXT INDEX c_index (c);
Query OK, 0 rows affected (0.017 sec)
```

And this succeeds in the same way as above:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
CREATE FULLTEXT INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.048 sec)

SET SESSION alter_algorithm='NOCOPY';
CREATE FULLTEXT INDEX c_index ON tab (c);
Query OK, 0 rows affected (0.016 sec)
```

But this second command fails, because only one FULLTEXT index can be added at a time:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   d VARCHAR(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.041 sec)

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD FULLTEXT INDEX c_index (c), ADD FULLTEXT INDEX d_index (d);
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: InnoDB presently supports one FULLTEXT
index creation at a time. Try ALGORITHM=COPY
```

Adding a Spatial Index

InnoDB supports adding a SPATIAL index to a table with ALGORITHM set to `NOCOPY`.

This operation supports a read-only locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `SHARED`. When this strategy is used, read-only concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD SPATIAL INDEX c_index (c);
Query OK, 0 rows affected (0.005 sec)
```

And this succeeds in the same way as above:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='NOCOPY';
CREATE SPATIAL INDEX c_index ON tab (c);
Query OK, 0 rows affected (0.005 sec)
```

`ALTER TABLE ... DROP INDEX` and `DROP INDEX`

InnoDB supports dropping indexes from a table with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE … DROP INDEX and DROP INDEX for more information.

This applies to ALTER TABLE … DROP INDEX and DROP INDEX for InnoDB tables.

`ALTER TABLE ... ADD FOREIGN KEY`

InnoDB does supports adding foreign key constraints to a table with ALGORITHM set to `NOCOPY`. In order to add a new foreign key constraint to a table with ALGORITHM set to `NOCOPY`, the foreign_key_checks system variable needs to be set to `OFF`. If it is set to `ON`, then `ALGORITHM=COPY` is required.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the LOCK clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this fails:

```
CREATE OR REPLACE TABLE tab1 (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   d INT
);

CREATE OR REPLACE TABLE tab2 (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Adding foreign keys needs
foreign_key_checks=OFF. Try ALGORITHM=COPY
```

But this succeeds:

```
CREATE OR REPLACE TABLE tab1 (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50),
   d INT
);

CREATE OR REPLACE TABLE tab2 (
   a INT PRIMARY KEY,
   b VARCHAR(50)
);

SET SESSION foreign_key_checks=OFF;
SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
Query OK, 0 rows affected (0.011 sec)
```

This applies to ALTER TABLE … ADD FOREIGN KEY for InnoDB tables.

`ALTER TABLE ... DROP FOREIGN KEY`

InnoDB supports dropping foreign key constraints from a table with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE … DROP FOREIGN KEY for more information.

This applies to ALTER TABLE … DROP FOREIGN KEY for InnoDB tables.

Table Operations

`ALTER TABLE ... AUTO_INCREMENT=...`

InnoDB supports changing a table's AUTO_INCREMENT value with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE … AUTO_INCREMENT=… for more information.

This applies to ALTER TABLE … AUTO_INCREMENT=… for InnoDB tables.

`ALTER TABLE ... ROW_FORMAT=...`

InnoDB does **not** support changing a table's row format with ALGORITHM set to `NOCOPY`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) ROW_FORMAT=DYNAMIC;

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ROW_FORMAT=COMPRESSED;
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Changing table options requires the table
to be rebuilt. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … ROW_FORMAT=… for InnoDB tables.

`ALTER TABLE ... KEY_BLOCK_SIZE=...`

InnoDB does **not** support changing a table's KEY_BLOCK_SIZE with ALGORITHM set to `NOCOPY`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) ROW_FORMAT=COMPRESSED
  KEY_BLOCK_SIZE=4;

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab KEY_BLOCK_SIZE=2;
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Changing table options requires the table
to be rebuilt. Try ALGORITHM=INPLACE
```

This applies to KEY_BLOCK_SIZE=… for InnoDB tables.

`ALTER TABLE ... PAGE_COMPRESSED=1` and `ALTER TABLE ... PAGE_COMPRESSION_LEVEL=...`

In MariaDB 10.3.10 and later, InnoDB supports setting a table's PAGE_COMPRESSED value to `1` with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

InnoDB does **not** support changing a table's PAGE_COMPRESSED value from `1` to `0` with ALGORITHM set to `NOCOPY`.

In these versions, InnoDB also supports changing a table's PAGE_COMPRESSION_LEVEL value with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause is set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE … PAGE_COMPRESSED=1 and ALTER TABLE … PAGE_COMPRESSION_LEVEL=… for more information.

This applies to ALTER TABLE … PAGE_COMPRESSED=… and ALTER TABLE … PAGE_COMPRESSION_LEVEL=… for InnoDB tables.

`ALTER TABLE ... DROP SYSTEM VERSIONING`

InnoDB does **not** support dropping system versioning from a table with ALGORITHM set to `NOCOPY`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
) WITH SYSTEM VERSIONING;

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab DROP SYSTEM VERSIONING;
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … DROP SYSTEM VERSIONING for InnoDB tables.

`ALTER TABLE ... DROP CONSTRAINT`

In MariaDB 10.3.6 and later, InnoDB supports dropping a CHECK constraint from a table with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE … DROP CONSTRAINT for more information.

This applies to ALTER TABLE … DROP CONSTRAINT for InnoDB tables.

`ALTER TABLE ... FORCE`

InnoDB does **not** support forcing a table rebuild with ALGORITHM set to `NOCOPY`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab FORCE;
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … FORCE for InnoDB tables.

`ALTER TABLE ... ENGINE=InnoDB`

InnoDB does **not** support forcing a table rebuild with ALGORITHM set to `NOCOPY`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ENGINE=InnoDB;
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to ALTER TABLE … ENGINE=InnoDB for InnoDB tables.

`OPTIMIZE TABLE ...`

InnoDB does **not** support optimizing a table with ALGORITHM set to `NOCOPY`.

For example:

```
CREATE OR REPLACE TABLE tab (
   a INT PRIMARY KEY,
   b VARCHAR(50),
   c VARCHAR(50)
);

SHOW GLOBAL VARIABLES WHERE Variable_name IN('innodb_defragment', 'innodb_optimize_fulltext_only');
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| innodb_defragment              | OFF   |
| innodb_optimize_fulltext_only | OFF   |
+--------------------------------+-------+
2 rows in set (0.001 sec)

SET SESSION alter_algorithm='NOCOPY';
OPTIMIZE TABLE tab;
+---------+----------+----------+-------------------------------------------------------------------------------
-----+
| Table   | Op       | Msg_type | Msg_text
|
+---------+----------+----------+-------------------------------------------------------------------------------
-----+
| db1.tab | optimize | note     | Table does not support optimize, doing recreate + analyze instead
|
| db1.tab | optimize | error    | ALGORITHM=NOCOPY is not supported for this operation. Try
ALGORITHM=INPLACE |
| db1.tab | optimize | status   | Operation failed
|
+---------+----------+----------+-------------------------------------------------------------------------------
-----+
3 rows in set, 1 warning (0.002 sec)
```

This applies to OPTIMIZE TABLE for InnoDB tables.

`ALTER TABLE ... RENAME TO` and `RENAME TABLE ...`

InnoDB supports renaming a table with ALGORITHM set to `NOCOPY` in the cases where the operation supports having the ALGORITHM clause set to `INSTANT`.

See InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE … RENAME TO and RENAME TABLE … for more information.

This applies to ALTER TABLE … RENAME TO and RENAME TABLE for InnoDB tables.

Limitations

Limitations Related to Generated (Virtual and Persistent/Stored) Columns

Generated columns do not currently support online DDL for all of the same operations that are supported for "real" columns.

See Generated (Virtual and Persistent/Stored) Columns: Statement Support for more information on the limitations.

This page is licensed: CC BY-SA / Gnu FDL

Instant ADD COLUMN for InnoDB

Normally, adding a column to a table requires the full table to be rebuilt. The complexity of the operation is proportional to the size of the table, or O(n·m) where n is the number of rows in the table and m is the number of indexes.

In MariaDB 10.0 and later, the ALTER TABLE statement supports online DDL for storage engines that have implemented the relevant online DDL algorithms and locking strategies.

The InnoDB storage engine has implemented online DDL for many operations. These online DDL optimizations allow concurrent DML to the table in many cases, even if the table needs to be rebuilt.

See InnoDB Online DDL Overview for more information about online DDL with InnoDB.

Allowing concurrent DML during the operation does not solve all problems. When a column was added to a table with the older in-place optimization, the resulting table rebuild could still significantly increase the I/O and memory consumption and cause replication lag.

In contrast, with the new instant ALTER TABLE … ADD COLUMN, all that is needed is an O(1) operation to insert a special hidden record into the table, and an update of the data dictionary. For a large table, instead of taking several hours, the operation would be completed in the blink of an eye. The ALTER TABLE … ADD COLUMN operation is only slightly more expensive than a regular INSERT, due to locking constraints.

In the past, some developers may have implemented a kind of "instant add column" in the application by encoding multiple columns in a single TEXT or BLOB column. MariaDB Dynamic Columns was an early example of that. A more recent example is JSON and related string manipulation functions.

Adding real columns has the following advantages over encoding columns into a single "expandable" column:

- Efficient storage in a native binary format
- Data type safety
- Indexes can be built natively
- Constraints are available: UNIQUE, CHECK, FOREIGN KEY
- DEFAULT values can be specified
- Triggers can be written more easily

With instant ALTER TABLE … ADD COLUMN, you can enjoy all the benefits of structured storage without the drawback of having to rebuild the table.

Instant ALTER TABLE … ADD COLUMN is available for both old and new InnoDB tables. Basically you can just upgrade from MySQL 5.x or MariaDB and start adding columns instantly.

Columns instantly added to a table exist in a separate data structure from the main table definition, similar to how InnoDB separates `BLOB` columns. If the table ever becomes empty, (such as from TRUNCATE or DELETE statements), InnoDB incorporates the instantly added columns into the main table definition. See InnoDB Online DDL Operations with ALGORITHM=INSTANT: Non-canonical Storage Format Caused by Some Operations for more information.

The operation is also crash safe. If the server is killed while executing an instant ALTER TABLE … ADD COLUMN, when the table is restored InnoDB integrates the new column, flattening the table definition.

Limitations

- In MariaDB 10.3, instant ALTER TABLE … ADD COLUMN only applies when the added columns appear last in the table. The place specifier `LAST` is the default. If `AFTER` col is specified, then col must be the last column, or the operation will require the table to be rebuilt. In MariaDB 10.4, this restriction was lifted.
- If the table contains a hidden `FTS_DOC_ID` column due to a FULLTEXT INDEX, then instant ALTER TABLE … ADD COLUMN will not be possible.
- InnoDB data files after instant ALTER TABLE … ADD COLUMN cannot be imported to older versions of MariaDB or MySQL without first being rebuilt.
- After using Instant ALTER TABLE … ADD COLUMN, any table-rebuilding operation such as ALTER TABLE … FORCE will incorporate instantaneously added columns into the main table body.
- Instant ALTER TABLE … ADD COLUMN is not available for ROW_FORMAT=COMPRESSED.
- In MariaDB 10.3, ALTER TABLE … DROP COLUMN requires the table to be rebuilt. In MariaDB 10.4, this restriction was lifted.

Example

```
CREATE TABLE t(id INT PRIMARY KEY, u INT UNSIGNED NOT NULL UNIQUE)
ENGINE=InnoDB;

INSERT INTO t(id,u) VALUES(1,1),(2,2),(3,3);

ALTER TABLE t ADD COLUMN
(d DATETIME DEFAULT current_timestamp(),
 p POINT NOT NULL DEFAULT ST_GeomFromText('POINT(0 0)'),
 t TEXT CHARSET utf8 DEFAULT 'The quick brown fox jumps over the lazy dog');

UPDATE t SET t=NULL WHERE id=3;

SELECT id,u,d,ST_AsText(p),t FROM t;

SELECT variable_value FROM information_schema.global_status
WHERE variable_name = 'innodb_instant_alter_column';
```

The above example illustrates that when the added columns are declared NOT NULL, a DEFAULT value must be available, either implied by the data type or set explicitly by the user. The expression need not be constant, but it must not refer to the columns of the table, such as DEFAULT u+1 (a MariaDB extension). The DEFAULT current_timestamp() would be evaluated at the time of the ALTER TABLE and apply to each row, like it does for non-instant ALTER TABLE. If a subsequent ALTER TABLE changes the DEFAULT value for subsequent INSERT, the values of the columns in existing records will naturally be unaffected.

The design was brainstormed in April by engineers from MariaDB Corporation, Alibaba and Tencent. A prototype was developed by Vin Chen (□□□) from the Tencent Game DBA Team.

See Also

- Other INSTANT operations in InnoDB

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Row Formats

See InnoDB Row Formats Overview.

An overview of the four InnoDB row formats (REDUNDANT, COMPACT, DYNAMIC, COMPRESSED), comparing their storage efficiency and feature support.

See InnoDB COMPACT Row Format.

Detailed information on the COMPACT row format, which reduces storage space by roughly 20% compared to REDUNDANT, handling NULLs and variable-length columns efficiently.

See InnoDB COMPRESSED Row Format.

Learn about the COMPRESSED row format, which compresses data and index pages using algorithms like zlib to minimize storage footprint at the cost of CPU.

See InnoDB DYNAMIC Row Format.

The DYNAMIC row format, default in modern MariaDB versions, optimizes storage for large BLOB/TEXT columns by storing them on separate overflow pages.

See InnoDB REDUNDANT Row Format.

Information on the legacy REDUNDANT row format, primarily maintained for backward compatibility with older MySQL versions.

See Troubleshooting Row Size Too Large Errors with InnoDB.

Complete InnoDB row size troubleshooting: innodb_strict_mode, ALTER TABLE ROW_FORMAT=DYNAMIC, VARCHAR/VARBINARY(256) overflow, and BLOB/TEXT solutions.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Row Formats Overview

The InnoDB storage engine supports four different row formats:

- REDUNDANT
- COMPACT
- DYNAMIC
- COMPRESSED

Default Row Format

The innodb_default_row_format system variable can be used to set the default row format for InnoDB tables. The possible values are:

- `redundant`
- `compact`
- `dynamic`

This system variable's default value is `dynamic`, which means that the default row format is `DYNAMIC`.

This system variable cannot be set to `compressed`, which means that the default row format cannot be `COMPRESSED`.

For example, the following statements would create a table with the `DYNAMIC` row format:

```
SET SESSION innodb_strict_mode=ON;

SET GLOBAL innodb_default_row_format='dynamic';

CREATE TABLE tab (
   id INT,
   str VARCHAR(50)
) ENGINE=InnoDB;
```

Setting a Table's Row Format

One way to specify an InnoDB table's row format is by setting the ROW_FORMAT table option to the relevant row format in a CREATE TABLE or ALTER TABLE statement:

```
SET SESSION innodb_strict_mode=ON;

SET GLOBAL innodb_file_per_table=ON;

SET GLOBAL innodb_file_format='Barracuda';

CREATE TABLE tab (
   id INT,
   str VARCHAR(50)
) ENGINE=InnoDB ROW_FORMAT=DYNAMIC;
```

Checking a Table's Row Format

The SHOW TABLE STATUS statement can be used to see the row format used by a table:

```
SHOW TABLE STATUS FROM db1 WHERE Name='tab'\G
*************************** 1. row ***************************
           Name: tab
         Engine: InnoDB
        Version: 10
     Row_format: Dynamic
           Rows: 0
 Avg_row_length: 0
    Data_length: 16384
Max_data_length: 0
   Index_length: 0
      Data_free: 0
 Auto_increment: NULL
    Create_time: 2019-04-18 20:24:04
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options: row_format=DYNAMIC
        Comment:
```

The information_schema.INNODB_SYS_TABLES table can also be queried to see the row format used by a table:

```
SELECT * FROM information_schema.INNODB_SYS_TABLES WHERE name='db1/tab'\G
*************************** 1. row ***************************
     TABLE_ID: 42
         NAME: db1/tab
         FLAG: 33
       N_COLS: 4
        SPACE: 27
  FILE_FORMAT: Barracuda
   ROW_FORMAT: Dynamic
ZIP_PAGE_SIZE: 0
   SPACE_TYPE: Single
```

A table's tablespace is tagged with the lowest InnoDB file format that supports the table's row format. So, even if the `Barracuda` file format is enabled, tables that use the `COMPACT` or `REDUNDANT` row formats are tagged with the `Antelope` file format in the information_schema.INNODB_SYS_TABLES table.

Row Formats

REDUNDANT Row Format

The `REDUNDANT` row format is the original non-compacted row format.

The `REDUNDANT` row format was the only available row format before MySQL 5.0.3. In that release, this row format was retroactively named the `REDUNDANT` row format. In the same release, the `COMPACT` row format was introduced as the new default row format.

See InnoDB REDUNDANT Row Format for more information.

COMPACT Row Format

Default row format in earlier versions `COMPACT`.

The `COMPACT` row format is similar to the `REDUNDANT` row format, but it stores data in a more compact manner that requires about 20% less storage.

See InnoDB COMPACT Row Format for more information.

DYNAMIC Row Format

`DYNAMIC` is the default row format.

The `DYNAMIC` row format is similar to the `COMPACT` row format, but tables using the `DYNAMIC` row format can store even more data on overflow pages than tables using the `COMPACT` row format. This results in more efficient data storage than tables using the `COMPACT` row format, especially for tables containing columns using the VARBINARY, VARCHAR, BLOB and TEXT data types. However, InnoDB tables using the `COMPRESSED` row format are more efficient.

See InnoDB DYNAMIC Row Format for more information.

COMPRESSED Row Format

An alternative way to compress InnoDB tables is by using InnoDB Page Compression.

The `COMPRESSED` row format is similar to the `COMPACT` row format, but tables using the `COMPRESSED` row format can store even more data on overflow pages than tables using the `COMPACT` row format. This results in more efficient data storage than tables using the `COMPACT` row format, especially for tables containing columns using the VARBINARY, VARCHAR, BLOB and TEXT data types.

The `COMPRESSED` row format also supports compression of all data and index pages.

See InnoDB COMPRESSED Row Format for more information.

Maximum Row Size

Several factors help determine the maximum row size of an InnoDB table.

First, MariaDB enforces a 65,535 byte limit on a table's maximum row size. The total size of a table's BLOB and TEXT columns do not count towards this limit. Only the pointers for a table's BLOB and TEXT columns count towards this limit. MariaDB enforces this limit for all storage engines, so this limit also applies to InnoDB tables. Therefore, this limit is the absolute maximum row size for an InnoDB table.

If you try to create a table that exceeds MariaDB's global limit on a table's maximum row size, then you will see an error like this:

```
ERROR 1118 (42000): Row size too large. The maximum row size for the used table type,
not counting BLOBs, is 65535. This includes storage overhead, check the manual. You
have to change some columns to TEXT or BLOBs
```

However, InnoDB also has its own limits on the maximum row size, so an InnoDB table's maximum row size could be smaller than MariaDB's global limit.

Second, the maximum amount of data that an InnoDB table can store in a row's main data page depends on the value of the innodb_page_size system variable. At most, the data that a single row can consume on the row's main data page is half of the value of the innodb_page_size system variable. With the default value of `16k`, that would mean that a single row can consume at most around 8 KB on the row's main data page. However, the limit on the row's main data page is not the absolute limit on the row's size.

Third, all InnoDB row formats can store certain kinds of data in overflow pages, so the maximum row size of an InnoDB table can be larger than the maximum amount of data that can be stored in the row's main data page.

Some row formats can store more data in overflow pages than others. For example, the `DYNAMIC` and `COMPRESSED` row formats can store the most data in overflow pages. To see how to determine the how the various InnoDB row formats can use overflow pages, see the following sections:

- InnoDB REDUNDANT Row Format: Overflow Pages with the REDUNDANT Row Format
- InnoDB COMPACT Row Format: Overflow Pages with the COMPACT Row Format
- InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format
- InnoDB COMPRESSED Row Format: Overflow Pages with the COMPRESSED Row Format

If a table's definition can allow rows that the table's InnoDB row format can't actually store, then InnoDB will raise errors or warnings in certain scenarios.

If the table were using the `REDUNDANT` or `COMPACT` row formats, then the error or warning would be the following:

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB or using ROW_FORMAT=DYNAMIC or ROW_FORMAT=COMPRESSED
may help. In current row format, BLOB prefix of 768 bytes is stored inline.
```

And if the table were using the `DYNAMIC` or `COMPRESSED` row formats, then the error or warning would be the following:

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored inline.
```

These messages are raised in the following cases:

- If InnoDB strict mode is enabled and if a DDL statement is executed that touches the table, such as CREATE TABLE or ALTER TABLE, then InnoDB will raise an error with this message
- If InnoDB strict mode is disabled and if a DDL statement is executed that touches the table, such as CREATE TABLE `or [ALTER TABLE](#pg-ence-sql-statements-data-definition-alter-alter-table-readme-a2aa59e4)`, then InnoDB will raise a warning with this message.
- Regardless of whether InnoDB strict mode is enabled, if a DML statement is executed that attempts to write a row that the table's InnoDB row format can't store, then InnoDB will raise an error with this message.

For information on how to solve the problem, see Troubleshooting Row Size Too Large Errors with InnoDB.

Feature Summary

Feature	Dynamic	Compressed	Compact	Redundant
Default	Yes	No	No	No
Recommended	Yes	No	No	No
Efficiently stores large columns	Yes	Yes	No	No
Efficiently utilizes buffer pool	Yes	No	Yes	Yes
Supported Page Sizes	• 64k • 32k • 16k • 8k • 4k	• 16k • 8k • 4k	• 64k • 32k • 16k • 8k • 4k	• 64k • 32k • 16k • 8k • 4k
Maximum size of indexed column values	• 3072 bytes (innodb_page_size >= 16k) • 1536 bytes (innodb_page_size == 8k) • 768 bytes (innodb_page_size == 4k)	• 3072 bytes (innodb_page_size >= 16k) • 1536 bytes (innodb_page_size == 8k) • 768 bytes (innodb_page_size == 4k)	767 bytes	767 bytes
Supports ADD/DROP column with INSTANT Algorithm	Yes	No	Yes	Yes

Known Issues

Upgrading Causes Row Size Too Large Errors

In earlier versions, MariaDB doesn't properly calculate the row sizes while executing DDL. In these versions, *unsafe* tables can be created, even if InnoDB strict mode is enabled. The calculations were fixed by MDEV-19292.

As a side effect, some tables that could be created or altered in previous versions may get rejected with the following error in these releases and any later releases.

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored inline.
```

And users could also see the following message as an error or warning in the error log:

```
[Warning] InnoDB: Cannot add field col in table db1.tab because after adding it, the row size is 8478
which is greater than maximum allowed size (8126) for a record on index leaf page.
```

InnoDB used the wrong calculations to determine row sizes for quite a long time, so a lot of users may unknowingly have *unsafe* tables that the InnoDB row format can't actually store.

InnoDB does not currently have an easy way to check which existing tables have this problem. See MDEV-20400 for more information.

For information on how to solve the problem, see Troubleshooting Row Size Too Large Errors with InnoDB.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB COMPACT Row Format

Note `COMPACT` was the default row format in prior versions of MariaDB. MariaDB has since transitioned to `DYNAMIC` as the default row format.

The `COMPACT` row format is similar to the `REDUNDANT` row format, but it stores data in a more compact manner that requires about 20% less storage.

Using the `COMPACT` Row Format

- If you performed a physical upgrade from older version of MariaDB Server or MySQL to a newer MariaDB Server version, then some of your tables may still use COMPACT row format.
- Compact row format does not store large columns as efficiently as the Dynamic row format.
- Compact row format limits indexing column values to 767 bytes, which is significant smaller than the Dynamic row format.

The easiest way to create an InnoDB table that uses the `COMPACT` row format is by setting the ROW_FORMAT table option to `COMPACT` in a CREATE TABLE or ALTER TABLE statement.

It is recommended to set the innodb_strict_mode system variable to `ON` when using this row format.

The `COMPACT` row format is supported by both the `Antelope` and the `Barracuda` file formats, so tables with this row format can be created regardless of the value of the innodb_file_format system variable.

For example:

```
SET SESSION innodb_strict_mode=ON;

CREATE TABLE tab (
   id INT,
   str VARCHAR(50)
) ENGINE=InnoDB ROW_FORMAT=COMPACT;
```

Index Prefixes with the `COMPACT` Row Format

The `COMPACT` row format supports index prefixes up to 767 bytes.

Overflow Pages with the `COMPACT` Row Format

All InnoDB row formats can store certain kinds of data in overflow pages. This allows for the maximum row size of an InnoDB table to be larger than the maximum amount of data that can be stored in the row's main data page. See Maximum Row Size for more information about the other factors that can contribute to the maximum row size for InnoDB tables.

In the `COMPACT` row format variable-length columns, such as columns using the VARBINARY, VARCHAR, BLOB and TEXT data types, can be partially stored in overflow pages.

InnoDB only considers using overflow pages if the table's row size is greater than half of innodb_page_size. If the row size is greater than this, then InnoDB chooses variable-length columns to be stored on overflow pages until the row size is less than half of innodb_page_size.

For VARBINARY, VARCHAR, BLOB and TEXT columns, only values longer than 767 bytes are considered for storage on overflow pages. Bytes that are stored to track a value's length do not count towards this limit. This limit is only based on the length of the actual column's data.

Fixed-length columns greater than 767 bytes are encoded as variable-length columns, so they can also be stored in overflow pages if the table's row size is greater than half of innodb_page_size. Even though a column using the CHAR data type can hold at most 255 characters, a CHAR column can still exceed 767 bytes in some cases. For example, a `char(255)` column can exceed 767 bytes if the character set is `utf8mb4`.

If a column is chosen to be stored on overflow pages, then the first 767 bytes of the column's value and a 20-byte pointer to the column's first overflow page are stored on the main page. Each overflow page is the size of innodb_page_size. If a column is too large to be stored on a single overflow page, then it is stored on multiple overflow pages. Each overflow page contains part of the data and a 20-byte pointer to the next overflow page, if a next page exists.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB COMPRESSED Row Format

An alternative (and usually superior) way to compress InnoDB tables is by using InnoDB Page Compression. See Comparison with the COMPRESSED Row Format.

The `COMPRESSED` row format is similar to the `COMPACT` row format, but tables using the `COMPRESSED` row format can store even more data on overflow pages than tables using the `COMPACT` row format. This results in more efficient data storage than tables using the `COMPACT` row format, especially for tables containing columns using the VARBINARY, VARCHAR, BLOB and TEXT data types.

The `COMPRESSED` row format also supports compression of all data and index pages.

Supported Block Sizes

The Compressed row format supports the following block sizes:

Block Size	KEY_BLOCK_SIZE Value
1 KB	1
2 KB	2
4 KB	4
8 KB	8
16 KB	16

If the `KEY_BLOCK_SIZE` table option is not specified, the block size defaults to half of innodb_page_size. With the default 16 KB page size, the block size defaults to 8 KB.

For compression to be effective, the chosen block size should be smaller than the servers value of the innodb_page_size system variable.

Supported Index Prefix Limits

The limit for indexing column values depends on the innodb_page_size value:

Page Size	Index Prefix Limit
16k	3072 bytes
8k	1536 bytes
4k	768 bytes

Using the `COMPRESSED` Row Format

The Compressed row format does not efficiently use the InnoDB buffer pool, so it is not the recommended way to achieve InnoDB table compression. For more information about how to compress InnoDB tables, see Configure InnoDB Page Compression.

An InnoDB table that uses the `COMPRESSED` row format can be created by setting the ROW_FORMAT table option to `COMPRESSED` and by setting the KEY_BLOCK_SIZE table option to one of the following values in a CREATE TABLE or ALTER TABLE statement, where the units are in `KB`.

`16k` is the default value of the innodb_page_size system variable, so using `16` will usually result in minimal compression unless one of the following is true:

- The table has many columns that can be stored in overflow pages, such as columns that use the VARBINARY, VARCHAR, BLOB and TEXT data types.
- The server is using a non-default innodb_page_size value that is greater than `16k`.

The value of the innodb_page_size system variable can be set to `32k` and `64k`. This is especially useful because the larger page size permits more columns using the VARBINARY, VARCHAR, BLOB and TEXT data types. Regardless, even when the value of the innodb_page_size system variable is set to some value higher than `16k`, `16` is still the maximum value for the KEY_BLOCK_SIZE table option for InnoDB tables using the `COMPRESSED` row format.

The `COMPRESSED` row format cannot be set as the default row format with the innodb_default_row_format system variable.

The `COMPRESSED` row format is only supported by the `Barracuda` file format. In earlier versions, the `COMPRESSED` row format is only supported if the InnoDB file format is `Barracuda`. Therefore, the innodb_file_format system variable must be set to `Barracuda` to use these row formats in those versions.

In earlier versions, the `COMPRESSED` row format is also only supported if the table is in a file per-table tablespace. Therefore, the innodb_file_per_table system variable must be set to `ON` to use this row format in those versions.

It is also recommended to set the innodb_strict_mode system variable to `ON` when using this row format.

InnoDB automatically uses the `COMPRESSED` row format for a table if the KEY_BLOCK_SIZE table option is set to some value in a CREATE TABLE or ALTER TABLE statement:

```
SET SESSION innodb_strict_mode=ON;

SET GLOBAL innodb_file_per_table=ON;

SET GLOBAL innodb_file_format='Barracuda';

CREATE TABLE tab (
   id INT,
   str VARCHAR(50)
) ENGINE=InnoDB KEY_BLOCK_SIZE=4;
```

If the KEY_BLOCK_SIZE table option is **not** set to some value, but the ROW_FORMAT table option is set to `COMPRESSED` in a CREATE TABLE or ALTER TABLE statement, then InnoDB uses a default value of `8` for the KEY_BLOCK_SIZE table option:

```
SET SESSION innodb_strict_mode=ON;

SET GLOBAL innodb_file_per_table=ON;

SET GLOBAL innodb_file_format='Barracuda';

CREATE TABLE tab (
   id INT,
   str VARCHAR(50)
) ENGINE=InnoDB ROW_FORMAT=COMPRESSED;
```

Create an InnoDB Table with the Compressed Row Format using ROW_FORMAT

The default block size for tables that use the Compressed row format is half of innodb_page_size. With the default 16 KB page size, the block size defaults to 8 KB. A compressed table with the default block size can be created by setting the ROW_FORMAT table option to COMPRESSED:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
) ROW_FORMAT = Compressed;
```

Step-by-Step Procedure

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. Create the table using the CREATE TABLE statement, and specify the Compressed row format using the `ROW_FORMAT` table option:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
) ROW_FORMAT = Compressed;
```

4. Confirm that the table uses the Compressed row format with an 8 KB block size by querying the information_schema.INNODB_SYS_TABLES table:

```
SELECT NAME, ROW_FORMAT, ZIP_PAGE_SIZE
FROM information_schema.INNODB_SYS_TABLES
WHERE NAME='hq_sales/invoices';
```

```
+-------------------+------------+---------------+
| NAME              | ROW_FORMAT | ZIP_PAGE_SIZE |
+-------------------+------------+---------------+
| hq_sales/invoices | Compressed |          8192 |
+-------------------+------------+---------------+
```

Create an InnoDB Table with the Compressed Row Format using KEY_BLOCK_SIZE

The default block size for tables that use the Compressed row format is half of innodb_page_size. With the default 16 KB page size, the block size defaults to 8 KB. A compressed table with a non-default block size can be created by setting the `KEY_BLOCK_SIZE` table option to the desired block size.

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
) KEY_BLOCK_SIZE = 4;
```

Step-by-Step Procedure

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. Create the table using the CREATE TABLE statement, and specify the block size using the `KEY_BLOCK_SIZE` table option, which implies the Compressed row format:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
) KEY_BLOCK_SIZE = 4;
```

4. Confirm that the table uses the Compressed row format with an 8 KB block size by querying the information_schema.INNODB_SYS_TABLES table:

```
SELECT NAME, ROW_FORMAT, ZIP_PAGE_SIZE
FROM information_schema.INNODB_SYS_TABLES
WHERE NAME='hq_sales/invoices';
```

```
+-------------------+------------+---------------+
| NAME              | ROW_FORMAT | ZIP_PAGE_SIZE |
+-------------------+------------+---------------+
| hq_sales/invoices | Compressed |          4096 |
+-------------------+------------+---------------+
```

Compression with the `COMPRESSED` Row Format

The `COMPRESSED` row format supports compression of all data and index pages.

To avoid compressing and uncompressing pages too many times, InnoDB tries to keep both compressed and uncompressed pages in the buffer pool when there is enough room. This results in a bigger cache. When there is

not enough room, an adaptive LRU algorithm is used to decide whether compressed or uncompressed pages should be evicted from the buffer: for CPU-bound workloads, the compressed pages are evicted first; for I/O-bound workloads, the uncompressed pages are evicted first. Of course, when necessary, both the compressed and uncompressed version of the same data can be evicted from the buffer.

Each compressed page has an uncompressed *modification log*, stored within the page itself. InnoDB writes small changes into it. When the space in the modification log runs out, the page is uncompressed, changes are applied, and the page is recompressed again. This is done to avoid some unnecessary decompression and compression operations.

Sometimes a *compression failure* might happen, because the data has grown too much to fit the page. When this happens, the page (and the index node) is split into two different pages. This process can be repeated recursively until the data fit the pages. This can be CPU-consuming on some busy servers which perform many write operations.

Before writing a compressed page into a data file, InnoDB writes it into the redo log. This ensures that the redo log can always be used to recover tables after a crash, even if the compression library is updated and some incompatibilities are introduced. But this also means that the redo log will grow faster and might need more space, or the frequency of checkpoints might need to increase.

Monitoring Performance of the `COMPRESSED` Row Format

The following `INFORMATION_SCHEMA` tables can be used to monitor the performances of InnoDB compressed tables:

- INNODB_CMP and INNODB_CMP_RESET
- INNODB_CMP_PER_INDEX and INNODB_CMP_PER_INDEX_RESET
- INNODB_CMPMEM and INNODB_CMPMEM_RESET

Index Prefixes with the `COMPRESSED` Row Format

The `COMPRESSED` row format supports index prefixes up to 3072 bytes. In MariaDB 10.2 and before, the innodb_large_prefix system variable is used to configure the maximum index prefix length. In these versions, if innodb_large_prefix is set to `ON`, then the maximum prefix length is 3072 bytes, and if it is set to `OFF`, then the maximum prefix length is 767 bytes.

Overflow Pages with the `COMPRESSED` Row Format

All InnoDB row formats can store certain kinds of data in overflow pages. This allows for the maximum row size of an InnoDB table to be larger than the maximum amount of data that can be stored in the row's main data page. See Maximum Row Size for more information about the other factors that can contribute to the maximum row size for InnoDB tables.

In the `COMPRESSED` row format variable-length columns, such as columns using the VARBINARY, VARCHAR, BLOB and TEXT data types, can be completely stored in overflow pages.

InnoDB only considers using overflow pages if the table's row size is greater than half of innodb_page_size. If the row size is greater than this, then InnoDB chooses variable-length columns to be stored on overflow pages until the row size is less than half of innodb_page_size.

For BLOB and TEXT columns, only values longer than 40 bytes are considered for storage on overflow pages. For VARBINARY and VARCHAR columns, only values longer than 255 bytes are considered for storage on overflow pages. Bytes that are stored to track a value's length do not count towards these limits. These limits are only based on the length of the actual column's data.

These limits differ from the limits for the `COMPACT` row format, where the limit is 767 bytes for all types.

Fixed-length columns greater than 767 bytes are encoded as variable-length columns, so they can also be stored in overflow pages if the table's row size is greater than half of innodb_page_size. Even though a column using the CHAR data type can hold at most 255 characters, a CHAR column can still exceed 767 bytes in some cases. For example, a `char(255)` column can exceed 767 bytes if the character set is `utf8mb4`.

If a column is chosen to be stored on overflow pages, then the entire value of the column is stored on overflow pages, and only a 20-byte pointer to the column's first overflow page is stored on the main page. Each overflow page is the size of innodb_page_size. If a column is too large to be stored on a single overflow page, then it is stored on multiple overflow pages. Each overflow page contains part of the data and a 20-byte pointer to the next overflow page, if a next page exists.

This behavior differs from the behavior of the `COMPACT` row format, which always stores the column prefix on the main page. This allows tables using the `COMPRESSED` row format to contain a high number of columns using the VARBINARY, VARCHAR, BLOB and TEXT data types.

Read-Only

MariaDB starting with 10.6

From MariaDB 10.6.0 until MariaDB 10.6.5, tables that are of the `COMPRESSED` row format are read-only by default. This was intended to be the first step towards removing write support and deprecating the feature.
This plan has been scrapped, and from MariaDB 10.6.6, `COMPRESSED` tables are no longer read-only by default.
From MariaDB 10.6.0 to MariaDB 10.6.5, set the innodb_read_only_compressed variable to `OFF` to make the tables writable.

See Also

- InnoDB Page Compression
- Storage-Engine Independent Column Compression

This page is licensed: CC BY-SA / Gnu FDL

InnoDB DYNAMIC Row Format

`DYNAMIC` is the default InnoDB row format.

The `DYNAMIC` row format is similar to the `COMPACT` row format, but tables using the `DYNAMIC` row format can store even more data on overflow pages than tables using the `COMPACT` row format. This results in more efficient data storage than tables using the `COMPACT` row format, especially for tables containing columns using the VARBINARY, VARCHAR, BLOB and TEXT data types. While InnoDB tables using the `COMPRESSED` row format can result in even greater space-efficiency, COMPRESSED requires substantially more memory and CPU to both read and write, so there is a significant performance and concurrency trade-off for that space-efficiency gain. COMPRESSED tables are not recommended for production use in most situations, while DYNAMIC row format scales well in high-performance environments.

Supported Index Prefix Limits

The limit for indexing column values depends on the innodb_page_size value:

Page Size	Index Prefix Limit
16k 32k 16k	3072 bytes
8k	1536 bytes
4k	768 bytes

Using the DYNAMIC Row Format

The default row format is `DYNAMIC`, as long as the innodb_default_row_format system variable has not been modified. Therefore, in these versions, the easiest way to create an InnoDB table that uses the `DYNAMIC` row format is by **not** setting the ROW_FORMAT table option at all in a CREATE TABLE or ALTER TABLE statement.

It is recommended to set the innodb_strict_mode system variable to `ON` when using this row format.

For example:

```
SET SESSION innodb_strict_mode=ON;

SET GLOBAL innodb_default_row_format='dynamic';

CREATE TABLE tab (
   id INT,
   str VARCHAR(50)
) ENGINE=InnoDB;
```

Create an InnoDB Table with the Dynamic Row Format by Default

InnoDB uses the Dynamic row format for new InnoDB tables by default, because the innodb_default_row_format system variable is dynamic by default.

Let's create an InnoDB table after confirming that the default storage engine is InnoDB and that InnoDB's default row format is Dynamic:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. Confirm that InnoDB's default row format is Dynamic by checking the innodb_default_row_format system variable using the SHOW GLOBAL VARIABLES statement:

```
SHOW GLOBAL VARIABLES
   LIKE 'innodb_default_row_format';
```

```
+---------------------------+---------+
| Variable_name             | Value   |
+---------------------------+---------+
| innodb_default_row_format | dynamic |
+---------------------------+---------+
```

4. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

5. Create the table using the CREATE TABLE statement:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
);
```

6. Confirm that the table uses the Dynamic row format by querying the information_schema.INNODB_SYS_TABLES table:

```
SELECT NAME, ROW_FORMAT
FROM information_schema.INNODB_SYS_TABLES
WHERE NAME='hq_sales/invoices';
```

```
+------------------+------------+
| NAME             | ROW_FORMAT |
+------------------+------------+
| hq_sales/invoices | Dynamic    |
+------------------+------------+
```

Create an InnoDB Table with the Dynamic Row Format using `ROW_FORMAT`

An InnoDB table that uses the Dynamic row format can be created using the `ROW_FORMAT` table option.

Let's create an InnoDB table after confirming that the default storage engine is InnoDB and that InnoDB's default row format is not Dynamic:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. Confirm that InnoDB's default row format is not Dynamic by checking the innodb_default_row_format system variable using the SHOW GLOBAL VARIABLES statement:

```
SHOW GLOBAL VARIABLES
   LIKE 'innodb_default_row_format';
```

```
+---------------------------+---------+
| Variable_name             | Value   |
+---------------------------+---------+
| innodb_default_row_format | compact |
+---------------------------+---------+
```

4. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

5. Create the table using the CREATE TABLE statement, and specify the Dynamic row format using the `ROW_FORMAT` table option:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
) ROW_FORMAT = DYNAMIC;
```

6. Confirm that the table uses the Dynamic row format by querying the information_schema.INNODB_SYS_TABLES table:

```
SELECT NAME, ROW_FORMAT
FROM information_schema.INNODB_SYS_TABLES
WHERE NAME='hq_sales/invoices';
```

```
+------------------+------------+
| NAME             | ROW_FORMAT |
+------------------+------------+
| hq_sales/invoices | Dynamic    |
+------------------+------------+
```

Convert InnoDB Tables to the Dynamic Row Format

If your database was physically upgraded from some older version of MariaDB Server or MySQL, then some of your tables may not be using the Dynamic row format. If you want to get the benefits of the Dynamic row format, then those tables will need to be converted to use it.

Let's convert some InnoDB tables to the Dynamic row format:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Search for InnoDB tables that do not use the Dynamic row format by querying the information_schema.INNODB_SYS_TABLES table:

```
SELECT NAME, ROW_FORMAT
FROM information_schema.INNODB_SYS_TABLES
WHERE NAME NOT LIKE 'SYS_%'
AND ROW_FORMAT != 'Dynamic';
```

```
+------------------+------------+
| NAME             | ROW_FORMAT |
+------------------+------------+
| hq_sales/invoices | Compact    |
+------------------+------------+
```

3. Alter the table using the ALTER TABLE statement, and specify the Dynamic row format using the `ROW_FORMAT` table option:

```
ALTER TABLE hq_sales.invoices
   ROW_FORMAT = DYNAMIC;
```

4. Confirm that the table uses the Dynamic row format by querying the information_schema.INNODB_SYS_TABLES table again:

```
SELECT NAME, ROW_FORMAT
FROM information_schema.INNODB_SYS_TABLES
WHERE NAME='hq_sales/invoices';
```

```
+------------------+------------+
| NAME             | ROW_FORMAT |
+------------------+------------+
| hq_sales/invoices | Dynamic    |
+------------------+------------+
```

Index Prefixes with the DYNAMIC Row Format

The `DYNAMIC` row format supports index prefixes up to 3072 bytes. In earlier versions of MariaDB, the innodb_large_prefix system variable is used to configure the maximum index prefix length. In these versions, if innodb_large_prefix is set to `ON`, then the maximum prefix length is 3072 bytes, and if it is set to `OFF`, then the maximum prefix length is 767 bytes.

Overflow Pages with the DYNAMIC Row Format

All InnoDB row formats can store certain kinds of data in overflow pages. This allows for the maximum row size of an InnoDB table to be larger than the maximum amount of data that can be stored in the row's main data page. See Maximum Row Size for more information about the other factors that can contribute to the maximum row size for InnoDB tables.

In the `DYNAMIC` row format variable-length columns, such as columns using the VARBINARY, VARCHAR, BLOB and TEXT data types, can be completely stored in overflow pages.

InnoDB only considers using overflow pages if the table's row size is greater than half of innodb_page_size. If the row size is greater than this, then InnoDB chooses variable-length columns to be stored on overflow pages until the row size is less than half of innodb_page_size.

For BLOB and TEXT columns, only values longer than 40 bytes are considered for storage on overflow pages. For VARBINARY and VARCHAR columns, only values longer than 255 bytes are considered for storage on overflow pages. Bytes that are stored to track a value's length do not count towards these limits. These limits are only based on the length of the actual column's data.

These limits differ from the limits for the `COMPACT` row format, where the limit is 767 bytes for all types.

Fixed-length columns greater than 767 bytes are encoded as variable-length columns, so they can also be stored in overflow pages if the table's row size is greater than half of innodb_page_size. Even though a column using the CHAR data type can hold at most 255 characters, a CHAR column can still exceed 767 bytes in some cases. For example, a `char(255)` column can exceed 767 bytes if the character set is `utf8mb4`.

If a column is chosen to be stored on overflow pages, then the entire value of the column is stored on overflow pages, and only a 20-byte pointer to the column's first overflow page is stored on the main page. Each overflow page is the size of innodb_page_size. If a column is too large to be stored on a single overflow page, then it is stored on multiple overflow pages. Each overflow page contains part of the data and a 20-byte pointer to the next overflow page, if a next page exists.

This behavior differs from the behavior of the `COMPACT` row format, which always stores the column prefix on the main page. This allows tables using the `DYNAMIC` row format to contain a high number of columns using the VARBINARY, VARCHAR, BLOB and TEXT data types.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB REDUNDANT Row Format

The `REDUNDANT` row format is the original non-compacted row format.

The `REDUNDANT` row format was the only available row format before MySQL 5.0.3. In that release, this row format was retroactively named the `REDUNDANT` row format. In the same release, the `COMPACT` row format was introduced as the new default row format.

Using the `REDUNDANT` Row Format

- Redundant row format should not be used in modern versions of MariaDB Server.
- Redundant row format does not store large columns as efficiently as the Dynamic row format.
- Redundant row format limits indexing column values to 767 bytes, which is significant smaller than the Dynamic row format.

The easiest way to create an InnoDB table that uses the `REDUNDANT` row format is by setting the ROW_FORMAT table option to `REDUNDANT` in a CREATE TABLE or ALTER TABLE statement.

It is recommended to set the innodb_strict_mode system variable to `ON` when using this format.

The `REDUNDANT` row format is supported by both the `Antelope` and the `Barracuda` file formats, so tables with this row format can be created regardless of the value of the innodb_file_format system variable.

For example:

```
SET SESSION innodb_strict_mode=ON;

CREATE TABLE tab (
   id INT,
   str VARCHAR(50)
) ENGINE=InnoDB ROW_FORMAT=REDUNDANT;
```

Index Prefixes with the `REDUNDANT` Row Format

The `REDUNDANT` row format supports index prefixes up to 767 bytes.

Overflow Pages with the `REDUNDANT` Row Format

All InnoDB row formats can store certain kinds of data in overflow pages. This allows for the maximum row size of an InnoDB table to be larger than the maximum amount of data that can be stored in the row's main data page. See Maximum Row Size for more information about the other factors that can contribute to the maximum row size for InnoDB tables.

In the `REDUNDANT` row format variable-length columns, such as columns using the VARBINARY, VARCHAR, BLOB and TEXT data types, can be partially stored in overflow pages.

InnoDB only considers using overflow pages if the table's row size is greater than half of innodb_page_size. If the row size is greater than this, then InnoDB chooses variable-length columns to be stored on overflow pages until the row size is less than half of innodb_page_size.

For VARBINARY, VARCHAR, BLOB and TEXT columns, only values longer than 767 bytes are considered for storage on overflow pages. Bytes that are stored to track a value's length do not count towards this limit. This limit is only based on the length of the actual column's data.

Fixed-length columns greater than 767 bytes are encoded as variable-length columns, so they can also be stored in overflow pages if the table's row size is greater than half of innodb_page_size. Even though a column using the CHAR data type can hold at most 255 characters, a CHAR column can still exceed 767 bytes in some cases. For example, a `char(255)` column can exceed 767 bytes if the character set is `utf8mb4`.

If a column is chosen to be stored on overflow pages, then the first 767 bytes of the column's value and a 20-byte pointer to the column's first overflow page are stored on the main page. Each overflow page is the size of [innodb-system-variables#innodb_page_size|innodb_page_size]]. If a column is too large to be stored on a single overflow page, then it is stored on multiple overflow pages. Each overflow page contains part of the data and a 20-byte pointer to the next overflow page, if a next page exists.

This page is licensed: CC BY-SA / Gnu FDL

Troubleshooting Row Size Too Large Errors with InnoDB

Overview

With InnoDB, users can see the following message as an error or warning:

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored
inline.
```

And they can also see the following message as an error or warning in the error log:

```
[Warning] InnoDB: Cannot add field col in table db1.tab because after adding it,
the row size is 8478 which is greater than maximum allowed size (8126) for a
record on index leaf page.
```

These messages indicate that the table definition allows rows that InnoDB row format can't store.

These messages are raised in the following cases:

- If InnoDB strict mode is enabled and if a DDL statement is executed that touches the table, such as CREATE TABLE or ALTER TABLE, InnoDB raises an error with the above message.
- If InnoDB strict mode is disabled and if a DDL statement is executed that touches the table, such as CREATE TABLE or ALTER TABLE, InnoDB raises a warning with the above message.
- Regardless of whether InnoDB strict mode is enabled, if a DML statement is executed that attempts to write a row that the table's InnoDB row format can't store, InnoDB raises an error with the above message.

Does the Problem Affect me?

The cause of the problem is described here. In very short, it affected old MariaDB versions (up to 10.2.26, 10.3.17, and 10.4.7). If you're on a newer version, the problem could yet still come up if the following applies:

- You created tables with a MariaDB version that has the problem.
- The tables weren't changed by any DML[14] statements in a newer MariaDB version since then.

If that's the case, you could still face the `Row size too large` error, particularly when issuing statements like ALTER TABLE or OPTIMIZE TABLE.

> **Note:**
>
> Take into account that the `Row size too large` error can come up for a good reason, for instance, when issuing DDL statements that actually exceed the row size.

> **Note:**
>
> For tables created in old MariaDB versions, an additional issue could come up: Tables were created whose row size wasn't calculated correctly. Creating those tables should have failed with the `Row size too large error`, but didn't.
>
> With such tables, you can get failures, both for DML (when inserted or updated data actually exceed the row size limit), and for DDL operations that should not affect the row size, like `TRUNCATE TABLE`, `CREATE TABLE LIKE`, or `OPTIMIZE TABLE`, or even dropping columns with `ALTER TABLE ... DROP COLUMN` which makes the row size shorter.

Example of the Problem

Here is an example of the problem:

```
SET GLOBAL innodb_default_row_format='dynamic';

SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
   col1 VARCHAR(40) NOT NULL,
   col2 VARCHAR(40) NOT NULL,
   col3 VARCHAR(40) NOT NULL,
   col4 VARCHAR(40) NOT NULL,
   col5 VARCHAR(40) NOT NULL,
   col6 VARCHAR(40) NOT NULL,
   col7 VARCHAR(40) NOT NULL,
   col8 VARCHAR(40) NOT NULL,
   col9 VARCHAR(40) NOT NULL,
   col10 VARCHAR(40) NOT NULL,
   col11 VARCHAR(40) NOT NULL,
   col12 VARCHAR(40) NOT NULL,
   col13 VARCHAR(40) NOT NULL,
   col14 VARCHAR(40) NOT NULL,
   col15 VARCHAR(40) NOT NULL,
   col16 VARCHAR(40) NOT NULL,
   col17 VARCHAR(40) NOT NULL,
   col18 VARCHAR(40) NOT NULL,
   col19 VARCHAR(40) NOT NULL,
   col20 VARCHAR(40) NOT NULL,
   col21 VARCHAR(40) NOT NULL,
   col22 VARCHAR(40) NOT NULL,
   col23 VARCHAR(40) NOT NULL,
   col24 VARCHAR(40) NOT NULL,
   col25 VARCHAR(40) NOT NULL,
   col26 VARCHAR(40) NOT NULL,
   col27 VARCHAR(40) NOT NULL,
   col28 VARCHAR(40) NOT NULL,
   col29 VARCHAR(40) NOT NULL,
   col30 VARCHAR(40) NOT NULL,
   col31 VARCHAR(40) NOT NULL,
   col32 VARCHAR(40) NOT NULL,
   col33 VARCHAR(40) NOT NULL,
   col34 VARCHAR(40) NOT NULL,
   col35 VARCHAR(40) NOT NULL,
   col36 VARCHAR(40) NOT NULL,
   col37 VARCHAR(40) NOT NULL,
   col38 VARCHAR(40) NOT NULL,
   col39 VARCHAR(40) NOT NULL,
   col40 VARCHAR(40) NOT NULL,
   col41 VARCHAR(40) NOT NULL,
   col42 VARCHAR(40) NOT NULL,
   col43 VARCHAR(40) NOT NULL,
   col44 VARCHAR(40) NOT NULL,
   col45 VARCHAR(40) NOT NULL,
   col46 VARCHAR(40) NOT NULL,
   col47 VARCHAR(40) NOT NULL,
   col48 VARCHAR(40) NOT NULL,
   col49 VARCHAR(40) NOT NULL,
   col50 VARCHAR(40) NOT NULL,
   col51 VARCHAR(40) NOT NULL,
   col52 VARCHAR(40) NOT NULL,
   col53 VARCHAR(40) NOT NULL,
   col54 VARCHAR(40) NOT NULL,
   col55 VARCHAR(40) NOT NULL,
   col56 VARCHAR(40) NOT NULL,
   col57 VARCHAR(40) NOT NULL,
   col58 VARCHAR(40) NOT NULL,
   col59 VARCHAR(40) NOT NULL,
   col60 VARCHAR(40) NOT NULL,
   col61 VARCHAR(40) NOT NULL,
   col62 VARCHAR(40) NOT NULL,
   col63 VARCHAR(40) NOT NULL,
```

```
    col64 VARCHAR(40) NOT NULL,
    col65 VARCHAR(40) NOT NULL,
    col66 VARCHAR(40) NOT NULL,
    col67 VARCHAR(40) NOT NULL,
    col68 VARCHAR(40) NOT NULL,
    col69 VARCHAR(40) NOT NULL,
    col70 VARCHAR(40) NOT NULL,
    col71 VARCHAR(40) NOT NULL,
    col72 VARCHAR(40) NOT NULL,
    col73 VARCHAR(40) NOT NULL,
    col74 VARCHAR(40) NOT NULL,
    col75 VARCHAR(40) NOT NULL,
    col76 VARCHAR(40) NOT NULL,
    col77 VARCHAR(40) NOT NULL,
    col78 VARCHAR(40) NOT NULL,
    col79 VARCHAR(40) NOT NULL,
    col80 VARCHAR(40) NOT NULL,
    col81 VARCHAR(40) NOT NULL,
    col82 VARCHAR(40) NOT NULL,
    col83 VARCHAR(40) NOT NULL,
    col84 VARCHAR(40) NOT NULL,
    col85 VARCHAR(40) NOT NULL,
    col86 VARCHAR(40) NOT NULL,
    col87 VARCHAR(40) NOT NULL,
    col88 VARCHAR(40) NOT NULL,
    col89 VARCHAR(40) NOT NULL,
    col90 VARCHAR(40) NOT NULL,
    col91 VARCHAR(40) NOT NULL,
    col92 VARCHAR(40) NOT NULL,
    col93 VARCHAR(40) NOT NULL,
    col94 VARCHAR(40) NOT NULL,
    col95 VARCHAR(40) NOT NULL,
    col96 VARCHAR(40) NOT NULL,
    col97 VARCHAR(40) NOT NULL,
    col98 VARCHAR(40) NOT NULL,
    col99 VARCHAR(40) NOT NULL,
    col100 VARCHAR(40) NOT NULL,
    col101 VARCHAR(40) NOT NULL,
    col102 VARCHAR(40) NOT NULL,
    col103 VARCHAR(40) NOT NULL,
    col104 VARCHAR(40) NOT NULL,
    col105 VARCHAR(40) NOT NULL,
    col106 VARCHAR(40) NOT NULL,
    col107 VARCHAR(40) NOT NULL,
    col108 VARCHAR(40) NOT NULL,
    col109 VARCHAR(40) NOT NULL,
    col110 VARCHAR(40) NOT NULL,
    col111 VARCHAR(40) NOT NULL,
    col112 VARCHAR(40) NOT NULL,
    col113 VARCHAR(40) NOT NULL,
    col114 VARCHAR(40) NOT NULL,
    col115 VARCHAR(40) NOT NULL,
    col116 VARCHAR(40) NOT NULL,
    col117 VARCHAR(40) NOT NULL,
    col118 VARCHAR(40) NOT NULL,
    col119 VARCHAR(40) NOT NULL,
    col120 VARCHAR(40) NOT NULL,
    col121 VARCHAR(40) NOT NULL,
    col122 VARCHAR(40) NOT NULL,
    col123 VARCHAR(40) NOT NULL,
    col124 VARCHAR(40) NOT NULL,
    col125 VARCHAR(40) NOT NULL,
    col126 VARCHAR(40) NOT NULL,
    col127 VARCHAR(40) NOT NULL,
    col128 VARCHAR(40) NOT NULL,
    col129 VARCHAR(40) NOT NULL,
    col130 VARCHAR(40) NOT NULL,
    col131 VARCHAR(40) NOT NULL,
    col132 VARCHAR(40) NOT NULL,
```

```
   col133 VARCHAR(40) NOT NULL,
   col134 VARCHAR(40) NOT NULL,
   col135 VARCHAR(40) NOT NULL,
   col136 VARCHAR(40) NOT NULL,
   col137 VARCHAR(40) NOT NULL,
   col138 VARCHAR(40) NOT NULL,
   col139 VARCHAR(40) NOT NULL,
   col140 VARCHAR(40) NOT NULL,
   col141 VARCHAR(40) NOT NULL,
   col142 VARCHAR(40) NOT NULL,
   col143 VARCHAR(40) NOT NULL,
   col144 VARCHAR(40) NOT NULL,
   col145 VARCHAR(40) NOT NULL,
   col146 VARCHAR(40) NOT NULL,
   col147 VARCHAR(40) NOT NULL,
   col148 VARCHAR(40) NOT NULL,
   col149 VARCHAR(40) NOT NULL,
   col150 VARCHAR(40) NOT NULL,
   col151 VARCHAR(40) NOT NULL,
   col152 VARCHAR(40) NOT NULL,
   col153 VARCHAR(40) NOT NULL,
   col154 VARCHAR(40) NOT NULL,
   col155 VARCHAR(40) NOT NULL,
   col156 VARCHAR(40) NOT NULL,
   col157 VARCHAR(40) NOT NULL,
   col158 VARCHAR(40) NOT NULL,
   col159 VARCHAR(40) NOT NULL,
   col160 VARCHAR(40) NOT NULL,
   col161 VARCHAR(40) NOT NULL,
   col162 VARCHAR(40) NOT NULL,
   col163 VARCHAR(40) NOT NULL,
   col164 VARCHAR(40) NOT NULL,
   col165 VARCHAR(40) NOT NULL,
   col166 VARCHAR(40) NOT NULL,
   col167 VARCHAR(40) NOT NULL,
   col168 VARCHAR(40) NOT NULL,
   col169 VARCHAR(40) NOT NULL,
   col170 VARCHAR(40) NOT NULL,
   col171 VARCHAR(40) NOT NULL,
   col172 VARCHAR(40) NOT NULL,
   col173 VARCHAR(40) NOT NULL,
   col174 VARCHAR(40) NOT NULL,
   col175 VARCHAR(40) NOT NULL,
   col176 VARCHAR(40) NOT NULL,
   col177 VARCHAR(40) NOT NULL,
   col178 VARCHAR(40) NOT NULL,
   col179 VARCHAR(40) NOT NULL,
   col180 VARCHAR(40) NOT NULL,
   col181 VARCHAR(40) NOT NULL,
   col182 VARCHAR(40) NOT NULL,
   col183 VARCHAR(40) NOT NULL,
   col184 VARCHAR(40) NOT NULL,
   col185 VARCHAR(40) NOT NULL,
   col186 VARCHAR(40) NOT NULL,
   col187 VARCHAR(40) NOT NULL,
   col188 VARCHAR(40) NOT NULL,
   col189 VARCHAR(40) NOT NULL,
   col190 VARCHAR(40) NOT NULL,
   col191 VARCHAR(40) NOT NULL,
   col192 VARCHAR(40) NOT NULL,
   col193 VARCHAR(40) NOT NULL,
   col194 VARCHAR(40) NOT NULL,
   col195 VARCHAR(40) NOT NULL,
   col196 VARCHAR(40) NOT NULL,
   col197 VARCHAR(40) NOT NULL,
   col198 VARCHAR(40) NOT NULL,
   PRIMARY KEY (col1)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored inline.
```

Root Cause of the Problem

The root cause is that InnoDB has a maximum row size that is roughly equivalent to half of the value of the innodb_page_size system variable. See InnoDB Row Formats Overview: Maximum Row Size for more information.

The InnoDB row formats work around this limit by storing certain kinds of variable-length columns on overflow pages. However, different row formats can store different types of data on overflow pages. Some row formats can store more data in overflow pages than others. For example, the DYNAMIC and COMPRESSED row formats can store the most data in overflow pages. To learn how the various InnoDB row formats use overflow pages, see the following pages:

- InnoDB REDUNDANT Row Format: Overflow Pages with the REDUNDANT Row Format
- InnoDB COMPACT Row Format: Overflow Pages with the COMPACT Row Format
- InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format
- InnoDB COMPRESSED Row Format: Overflow Pages with the COMPRESSED Row Format

Checking Existing Tables for the Problem

InnoDB does not currently have an easy way to check all existing tables to determine which tables have this problem. See MDEV-20400 for more information.

One method to check a single existing table for this problem is to enable InnoDB strict mode, and then try to create a duplicate of the table with CREATE TABLE … LIKE. If the table has this problem, then the operation fails:

```
SET SESSION innodb_strict_mode=ON;

CREATE TABLE tab_dup LIKE tab;
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored inline.
```

Finding All Tables That Currently Have the Problem

The following shell script will read through a MariaDB server to identify every table that has a row size definition that is too large for its row format and the server's page size. It runs on most common distributions of Linux.

To run the script, copy the code below to a shell-script named `rowsize.sh`, make it executable with the command `chmod 755 ./rowsize.sh`, and invoke it with the following parameters:

```
./rowsize.sh host user password
```

When the script runs, it displays the name of the temporary database it creates, so that if the script is interrupted before cleaning up, the database can be easily identified and removed manually.

As the script runs it will output one line reporting the database and tablename for each table it finds that has the oversize row problem. If it finds none, it will print the following message: "No tables with rows size too big found."

In either case, the script prints one final line to announce when it's done: `./rowsize.sh done.`

```
#!/bin/bash

[ -z "$3" ] && echo "Usage: $0 host user password" >&2 && exit 1

dt="tmp_$RANDOM$RANDOM"

mysql -h $1 -u $2 -p$3 -ABNe "create database $dt;"
[ $? -ne 0 ] && echo "Error: $0 terminating" >&2 exit 1

echo
echo "Created temporary database ${dt} on host $1"
echo

c=0
for d in $(mysql -h $1 -u $2 -p$3 -ABNe "show databases;" | egrep -iv
"information_schema|mysql|performance_schema|$dt")
do
    for t in $(mysql -h $1 -u $2 -p$3 -ABNe "show tables;" $d)
    do
        tc=$(mysql -h $1 -u $2 -p$3 -ABNe "show create table $t\\G" $d | egrep -iv "^\*|^$t")

        echo $tc | grep -iq "ROW_FORMAT"
        if [ $? -ne 0 ]
        then
            tf=$(mysql -h $1 -u $2 -p$3 -ABNe "select row_format from
information_schema.innodb_sys_tables where name = '${d}/${t}';")
            tc="$tc ROW_FORMAT=$tf"
        fi

        ef="/tmp/e$RANDOM$RANDOM"
        mysql -h $1 -u $2 -p$3 -ABNe "set innodb_strict_mode=1; set foreign_key_checks=0; ${tc};" $dt
>/dev/null  2>$ef
        [ $? -ne 0 ] && cat $ef | grep -q "Row size too large" && echo "${d}.${t}" && let c++ || mysql -h
$1 -u $2 -p$3 -ABNe "drop table if exists ${t};" $dt
        rm -f $ef
    done
done
mysql -h $1 -u $2 -p$3 -ABNe "set innodb_strict_mode=1; drop database $dt;"
[ $c -eq 0 ] && echo "No tables with rows size too large found." || echo && echo "$c tables found with
row size too large."
echo
echo "$0 done."
```

Solving the Problem

There are several potential solutions available to solve this problem.

Converting the Table to the DYNAMIC Row Format

If the table is using either the REDUNDANT or the COMPACT row format, then one potential solution to this problem is to convert the table to use the DYNAMIC row format instead.

If your tables were originally created on an older version of MariaDB or MySQL, then your table may be using one of InnoDB's older row formats:

- In MariaDB 10.1 and before, and in MySQL 5.6 and before, the COMPACT row format was the default row format.
- In MySQL 4.1 and before, the REDUNDANT row format was the default row format.

The DYNAMIC row format can store more data on overflow pages than these older row formats, so this row format may actually be able to store the table's data safely. See InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format for more information.

Therefore, a potential solution to the *Row size too large* error is to convert the table to use the DYNAMIC row format:

```
ALTER TABLE tab ROW_FORMAT=DYNAMIC;
```

You can use the INNODB_SYS_TABLES table in the information_schema database to find all tables that use the REDUNDANT or the COMPACT row formats. This is helpful if you would like to convert all of your tables that you still use the older row formats to the DYNAMIC row format. For example, the following query can find those tables, while excluding InnoDB's internal system tables:

```
SELECT NAME, ROW_FORMAT
FROM information_schema.INNODB_SYS_TABLES
WHERE ROW_FORMAT IN('Redundant', 'Compact')
AND NAME NOT IN('SYS_DATAFILES', 'SYS_FOREIGN', 'SYS_FOREIGN_COLS', 'SYS_TABLESPACES', 'SYS_VIRTUAL',
'SYS_ZIP_DICT', 'SYS_ZIP_DICT_COLS');
```

In MariaDB 10.2 and later, the DYNAMIC row format is the default row format. If your tables were originally created on one of these newer versions, then they may already be using this row format. In that case, you may need to try the next solution.

Fitting More Columns on Overflow Pages

If the table is already using the DYNAMIC row format, then another potential solution to this problem is to change the table schema, so that the row format can store more columns on overflow pages.

In order for InnoDB to store some variable-length columns on overflow pages, the length of those columns may need to be increased.

Therefore, a counter-intuitive solution to the *Row size too large* error in a lot of cases is actually to **increase** the length of some variable-length columns, so that InnoDB's row format can store them on overflow pages.

Some possible ways to change the table schema are listed below.

Converting Some Columns to `BLOB` or `TEXT`

For BLOB and TEXT columns, the DYNAMIC row format can store these columns on overflow pages. See InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format for more information.

Therefore, a potential solution to the *Row size too large* error is to convert some columns to the BLOB or TEXT data types.

Increasing the Length of `VARBINARY` Columns

For VARBINARY columns, the DYNAMIC row format can only store these columns on overflow pages if the maximum length of the column is 256 bytes or longer. See InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format for more information.

Therefore, a potential solution to the *Row size too large* error is to ensure that all VARBINARY columns are at least as long as `varbinary(256)`.

Increasing the Length of `VARCHAR` Columns

For VARCHAR columns, the DYNAMIC row format can only store these columns on overflow pages if the maximum length of the column is 256 bytes or longer. See InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format for more information.

The original table schema shown earlier on this page causes the *Row size too large* error, because all of the table's VARCHAR columns are smaller than 256 bytes, which means that they have to be stored on the row's main data

page.

Therefore, a potential solution to the *Row size too large* error is to ensure that all VARCHAR columns are at least as long as 256 bytes. The number of characters required to reach the 256 byte limit depends on the character set used by the column.

For example, when using InnoDB's DYNAMIC row format and a default character set of latin1 (which requires up to 1 byte per character), the 256 byte limit means that a VARCHAR column will only be stored on overflow pages if it is at least as large as a `varchar(256)`:

```
SET GLOBAL innodb_default_row_format='dynamic';
SET SESSION innodb_strict_mode=ON;
CREATE OR REPLACE TABLE tab (
   col1 VARCHAR(256) NOT NULL,
   col2 VARCHAR(256) NOT NULL,
   col3 VARCHAR(256) NOT NULL,
   col4 VARCHAR(256) NOT NULL,
   col5 VARCHAR(256) NOT NULL,
   col6 VARCHAR(256) NOT NULL,
   col7 VARCHAR(256) NOT NULL,
   col8 VARCHAR(256) NOT NULL,
   col9 VARCHAR(256) NOT NULL,
   col10 VARCHAR(256) NOT NULL,
   col11 VARCHAR(256) NOT NULL,
   col12 VARCHAR(256) NOT NULL,
   col13 VARCHAR(256) NOT NULL,
   col14 VARCHAR(256) NOT NULL,
   col15 VARCHAR(256) NOT NULL,
   col16 VARCHAR(256) NOT NULL,
   col17 VARCHAR(256) NOT NULL,
   col18 VARCHAR(256) NOT NULL,
   col19 VARCHAR(256) NOT NULL,
   col20 VARCHAR(256) NOT NULL,
   col21 VARCHAR(256) NOT NULL,
   col22 VARCHAR(256) NOT NULL,
   col23 VARCHAR(256) NOT NULL,
   col24 VARCHAR(256) NOT NULL,
   col25 VARCHAR(256) NOT NULL,
   col26 VARCHAR(256) NOT NULL,
   col27 VARCHAR(256) NOT NULL,
   col28 VARCHAR(256) NOT NULL,
   col29 VARCHAR(256) NOT NULL,
   col30 VARCHAR(256) NOT NULL,
   col31 VARCHAR(256) NOT NULL,
   col32 VARCHAR(256) NOT NULL,
   col33 VARCHAR(256) NOT NULL,
   col34 VARCHAR(256) NOT NULL,
   col35 VARCHAR(256) NOT NULL,
   col36 VARCHAR(256) NOT NULL,
   col37 VARCHAR(256) NOT NULL,
   col38 VARCHAR(256) NOT NULL,
   col39 VARCHAR(256) NOT NULL,
   col40 VARCHAR(256) NOT NULL,
   col41 VARCHAR(256) NOT NULL,
   col42 VARCHAR(256) NOT NULL,
   col43 VARCHAR(256) NOT NULL,
   col44 VARCHAR(256) NOT NULL,
   col45 VARCHAR(256) NOT NULL,
   col46 VARCHAR(256) NOT NULL,
   col47 VARCHAR(256) NOT NULL,
   col48 VARCHAR(256) NOT NULL,
   col49 VARCHAR(256) NOT NULL,
   col50 VARCHAR(256) NOT NULL,
   col51 VARCHAR(256) NOT NULL,
   col52 VARCHAR(256) NOT NULL,
   col53 VARCHAR(256) NOT NULL,
   col54 VARCHAR(256) NOT NULL,
   col55 VARCHAR(256) NOT NULL,
   col56 VARCHAR(256) NOT NULL,
   col57 VARCHAR(256) NOT NULL,
   col58 VARCHAR(256) NOT NULL,
   col59 VARCHAR(256) NOT NULL,
   col60 VARCHAR(256) NOT NULL,
   col61 VARCHAR(256) NOT NULL,
   col62 VARCHAR(256) NOT NULL,
   col63 VARCHAR(256) NOT NULL,
   col64 VARCHAR(256) NOT NULL,
   col65 VARCHAR(256) NOT NULL,
```

```
    col66 VARCHAR(256) NOT NULL,
    col67 VARCHAR(256) NOT NULL,
    col68 VARCHAR(256) NOT NULL,
    col69 VARCHAR(256) NOT NULL,
    col70 VARCHAR(256) NOT NULL,
    col71 VARCHAR(256) NOT NULL,
    col72 VARCHAR(256) NOT NULL,
    col73 VARCHAR(256) NOT NULL,
    col74 VARCHAR(256) NOT NULL,
    col75 VARCHAR(256) NOT NULL,
    col76 VARCHAR(256) NOT NULL,
    col77 VARCHAR(256) NOT NULL,
    col78 VARCHAR(256) NOT NULL,
    col79 VARCHAR(256) NOT NULL,
    col80 VARCHAR(256) NOT NULL,
    col81 VARCHAR(256) NOT NULL,
    col82 VARCHAR(256) NOT NULL,
    col83 VARCHAR(256) NOT NULL,
    col84 VARCHAR(256) NOT NULL,
    col85 VARCHAR(256) NOT NULL,
    col86 VARCHAR(256) NOT NULL,
    col87 VARCHAR(256) NOT NULL,
    col88 VARCHAR(256) NOT NULL,
    col89 VARCHAR(256) NOT NULL,
    col90 VARCHAR(256) NOT NULL,
    col91 VARCHAR(256) NOT NULL,
    col92 VARCHAR(256) NOT NULL,
    col93 VARCHAR(256) NOT NULL,
    col94 VARCHAR(256) NOT NULL,
    col95 VARCHAR(256) NOT NULL,
    col96 VARCHAR(256) NOT NULL,
    col97 VARCHAR(256) NOT NULL,
    col98 VARCHAR(256) NOT NULL,
    col99 VARCHAR(256) NOT NULL,
    col100 VARCHAR(256) NOT NULL,
    col101 VARCHAR(256) NOT NULL,
    col102 VARCHAR(256) NOT NULL,
    col103 VARCHAR(256) NOT NULL,
    col104 VARCHAR(256) NOT NULL,
    col105 VARCHAR(256) NOT NULL,
    col106 VARCHAR(256) NOT NULL,
    col107 VARCHAR(256) NOT NULL,
    col108 VARCHAR(256) NOT NULL,
    col109 VARCHAR(256) NOT NULL,
    col110 VARCHAR(256) NOT NULL,
    col111 VARCHAR(256) NOT NULL,
    col112 VARCHAR(256) NOT NULL,
    col113 VARCHAR(256) NOT NULL,
    col114 VARCHAR(256) NOT NULL,
    col115 VARCHAR(256) NOT NULL,
    col116 VARCHAR(256) NOT NULL,
    col117 VARCHAR(256) NOT NULL,
    col118 VARCHAR(256) NOT NULL,
    col119 VARCHAR(256) NOT NULL,
    col120 VARCHAR(256) NOT NULL,
    col121 VARCHAR(256) NOT NULL,
    col122 VARCHAR(256) NOT NULL,
    col123 VARCHAR(256) NOT NULL,
    col124 VARCHAR(256) NOT NULL,
    col125 VARCHAR(256) NOT NULL,
    col126 VARCHAR(256) NOT NULL,
    col127 VARCHAR(256) NOT NULL,
    col128 VARCHAR(256) NOT NULL,
    col129 VARCHAR(256) NOT NULL,
    col130 VARCHAR(256) NOT NULL,
    col131 VARCHAR(256) NOT NULL,
    col132 VARCHAR(256) NOT NULL,
    col133 VARCHAR(256) NOT NULL,
    col134 VARCHAR(256) NOT NULL,
```

```
   col135 VARCHAR(256) NOT NULL,
   col136 VARCHAR(256) NOT NULL,
   col137 VARCHAR(256) NOT NULL,
   col138 VARCHAR(256) NOT NULL,
   col139 VARCHAR(256) NOT NULL,
   col140 VARCHAR(256) NOT NULL,
   col141 VARCHAR(256) NOT NULL,
   col142 VARCHAR(256) NOT NULL,
   col143 VARCHAR(256) NOT NULL,
   col144 VARCHAR(256) NOT NULL,
   col145 VARCHAR(256) NOT NULL,
   col146 VARCHAR(256) NOT NULL,
   col147 VARCHAR(256) NOT NULL,
   col148 VARCHAR(256) NOT NULL,
   col149 VARCHAR(256) NOT NULL,
   col150 VARCHAR(256) NOT NULL,
   col151 VARCHAR(256) NOT NULL,
   col152 VARCHAR(256) NOT NULL,
   col153 VARCHAR(256) NOT NULL,
   col154 VARCHAR(256) NOT NULL,
   col155 VARCHAR(256) NOT NULL,
   col156 VARCHAR(256) NOT NULL,
   col157 VARCHAR(256) NOT NULL,
   col158 VARCHAR(256) NOT NULL,
   col159 VARCHAR(256) NOT NULL,
   col160 VARCHAR(256) NOT NULL,
   col161 VARCHAR(256) NOT NULL,
   col162 VARCHAR(256) NOT NULL,
   col163 VARCHAR(256) NOT NULL,
   col164 VARCHAR(256) NOT NULL,
   col165 VARCHAR(256) NOT NULL,
   col166 VARCHAR(256) NOT NULL,
   col167 VARCHAR(256) NOT NULL,
   col168 VARCHAR(256) NOT NULL,
   col169 VARCHAR(256) NOT NULL,
   col170 VARCHAR(256) NOT NULL,
   col171 VARCHAR(256) NOT NULL,
   col172 VARCHAR(256) NOT NULL,
   col173 VARCHAR(256) NOT NULL,
   col174 VARCHAR(256) NOT NULL,
   col175 VARCHAR(256) NOT NULL,
   col176 VARCHAR(256) NOT NULL,
   col177 VARCHAR(256) NOT NULL,
   col178 VARCHAR(256) NOT NULL,
   col179 VARCHAR(256) NOT NULL,
   col180 VARCHAR(256) NOT NULL,
   col181 VARCHAR(256) NOT NULL,
   col182 VARCHAR(256) NOT NULL,
   col183 VARCHAR(256) NOT NULL,
   col184 VARCHAR(256) NOT NULL,
   col185 VARCHAR(256) NOT NULL,
   col186 VARCHAR(256) NOT NULL,
   col187 VARCHAR(256) NOT NULL,
   col188 VARCHAR(256) NOT NULL,
   col189 VARCHAR(256) NOT NULL,
   col190 VARCHAR(256) NOT NULL,
   col191 VARCHAR(256) NOT NULL,
   col192 VARCHAR(256) NOT NULL,
   col193 VARCHAR(256) NOT NULL,
   col194 VARCHAR(256) NOT NULL,
   col195 VARCHAR(256) NOT NULL,
   col196 VARCHAR(256) NOT NULL,
   col197 VARCHAR(256) NOT NULL,
   col198 VARCHAR(256) NOT NULL,
   PRIMARY KEY (col1)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

And when using InnoDB's DYNAMIC row format and a default character set of utf8 (which requires up to 3 bytes per character), the 256 byte limit means that a VARCHAR column will only be stored on overflow pages if it is at least as large as a `varchar(86)`:

```
SET GLOBAL innodb_default_row_format='dynamic';
SET SESSION innodb_strict_mode=ON;
CREATE OR REPLACE TABLE tab (
   col1 VARCHAR(86) NOT NULL,
   col2 VARCHAR(86) NOT NULL,
   col3 VARCHAR(86) NOT NULL,
   col4 VARCHAR(86) NOT NULL,
   col5 VARCHAR(86) NOT NULL,
   col6 VARCHAR(86) NOT NULL,
   col7 VARCHAR(86) NOT NULL,
   col8 VARCHAR(86) NOT NULL,
   col9 VARCHAR(86) NOT NULL,
   col10 VARCHAR(86) NOT NULL,
   col11 VARCHAR(86) NOT NULL,
   col12 VARCHAR(86) NOT NULL,
   col13 VARCHAR(86) NOT NULL,
   col14 VARCHAR(86) NOT NULL,
   col15 VARCHAR(86) NOT NULL,
   col16 VARCHAR(86) NOT NULL,
   col17 VARCHAR(86) NOT NULL,
   col18 VARCHAR(86) NOT NULL,
   col19 VARCHAR(86) NOT NULL,
   col20 VARCHAR(86) NOT NULL,
   col21 VARCHAR(86) NOT NULL,
   col22 VARCHAR(86) NOT NULL,
   col23 VARCHAR(86) NOT NULL,
   col24 VARCHAR(86) NOT NULL,
   col25 VARCHAR(86) NOT NULL,
   col26 VARCHAR(86) NOT NULL,
   col27 VARCHAR(86) NOT NULL,
   col28 VARCHAR(86) NOT NULL,
   col29 VARCHAR(86) NOT NULL,
   col30 VARCHAR(86) NOT NULL,
   col31 VARCHAR(86) NOT NULL,
   col32 VARCHAR(86) NOT NULL,
   col33 VARCHAR(86) NOT NULL,
   col34 VARCHAR(86) NOT NULL,
   col35 VARCHAR(86) NOT NULL,
   col36 VARCHAR(86) NOT NULL,
   col37 VARCHAR(86) NOT NULL,
   col38 VARCHAR(86) NOT NULL,
   col39 VARCHAR(86) NOT NULL,
   col40 VARCHAR(86) NOT NULL,
   col41 VARCHAR(86) NOT NULL,
   col42 VARCHAR(86) NOT NULL,
   col43 VARCHAR(86) NOT NULL,
   col44 VARCHAR(86) NOT NULL,
   col45 VARCHAR(86) NOT NULL,
   col46 VARCHAR(86) NOT NULL,
   col47 VARCHAR(86) NOT NULL,
   col48 VARCHAR(86) NOT NULL,
   col49 VARCHAR(86) NOT NULL,
   col50 VARCHAR(86) NOT NULL,
   col51 VARCHAR(86) NOT NULL,
   col52 VARCHAR(86) NOT NULL,
   col53 VARCHAR(86) NOT NULL,
   col54 VARCHAR(86) NOT NULL,
   col55 VARCHAR(86) NOT NULL,
   col56 VARCHAR(86) NOT NULL,
   col57 VARCHAR(86) NOT NULL,
   col58 VARCHAR(86) NOT NULL,
   col59 VARCHAR(86) NOT NULL,
   col60 VARCHAR(86) NOT NULL,
   col61 VARCHAR(86) NOT NULL,
   col62 VARCHAR(86) NOT NULL,
   col63 VARCHAR(86) NOT NULL,
   col64 VARCHAR(86) NOT NULL,
   col65 VARCHAR(86) NOT NULL,
```

```
    col66 VARCHAR(86) NOT NULL,
    col67 VARCHAR(86) NOT NULL,
    col68 VARCHAR(86) NOT NULL,
    col69 VARCHAR(86) NOT NULL,
    col70 VARCHAR(86) NOT NULL,
    col71 VARCHAR(86) NOT NULL,
    col72 VARCHAR(86) NOT NULL,
    col73 VARCHAR(86) NOT NULL,
    col74 VARCHAR(86) NOT NULL,
    col75 VARCHAR(86) NOT NULL,
    col76 VARCHAR(86) NOT NULL,
    col77 VARCHAR(86) NOT NULL,
    col78 VARCHAR(86) NOT NULL,
    col79 VARCHAR(86) NOT NULL,
    col80 VARCHAR(86) NOT NULL,
    col81 VARCHAR(86) NOT NULL,
    col82 VARCHAR(86) NOT NULL,
    col83 VARCHAR(86) NOT NULL,
    col84 VARCHAR(86) NOT NULL,
    col85 VARCHAR(86) NOT NULL,
    col86 VARCHAR(86) NOT NULL,
    col87 VARCHAR(86) NOT NULL,
    col88 VARCHAR(86) NOT NULL,
    col89 VARCHAR(86) NOT NULL,
    col90 VARCHAR(86) NOT NULL,
    col91 VARCHAR(86) NOT NULL,
    col92 VARCHAR(86) NOT NULL,
    col93 VARCHAR(86) NOT NULL,
    col94 VARCHAR(86) NOT NULL,
    col95 VARCHAR(86) NOT NULL,
    col96 VARCHAR(86) NOT NULL,
    col97 VARCHAR(86) NOT NULL,
    col98 VARCHAR(86) NOT NULL,
    col99 VARCHAR(86) NOT NULL,
    col100 VARCHAR(86) NOT NULL,
    col101 VARCHAR(86) NOT NULL,
    col102 VARCHAR(86) NOT NULL,
    col103 VARCHAR(86) NOT NULL,
    col104 VARCHAR(86) NOT NULL,
    col105 VARCHAR(86) NOT NULL,
    col106 VARCHAR(86) NOT NULL,
    col107 VARCHAR(86) NOT NULL,
    col108 VARCHAR(86) NOT NULL,
    col109 VARCHAR(86) NOT NULL,
    col110 VARCHAR(86) NOT NULL,
    col111 VARCHAR(86) NOT NULL,
    col112 VARCHAR(86) NOT NULL,
    col113 VARCHAR(86) NOT NULL,
    col114 VARCHAR(86) NOT NULL,
    col115 VARCHAR(86) NOT NULL,
    col116 VARCHAR(86) NOT NULL,
    col117 VARCHAR(86) NOT NULL,
    col118 VARCHAR(86) NOT NULL,
    col119 VARCHAR(86) NOT NULL,
    col120 VARCHAR(86) NOT NULL,
    col121 VARCHAR(86) NOT NULL,
    col122 VARCHAR(86) NOT NULL,
    col123 VARCHAR(86) NOT NULL,
    col124 VARCHAR(86) NOT NULL,
    col125 VARCHAR(86) NOT NULL,
    col126 VARCHAR(86) NOT NULL,
    col127 VARCHAR(86) NOT NULL,
    col128 VARCHAR(86) NOT NULL,
    col129 VARCHAR(86) NOT NULL,
    col130 VARCHAR(86) NOT NULL,
    col131 VARCHAR(86) NOT NULL,
    col132 VARCHAR(86) NOT NULL,
    col133 VARCHAR(86) NOT NULL,
    col134 VARCHAR(86) NOT NULL,
```

```
   col135 VARCHAR(86) NOT NULL,
   col136 VARCHAR(86) NOT NULL,
   col137 VARCHAR(86) NOT NULL,
   col138 VARCHAR(86) NOT NULL,
   col139 VARCHAR(86) NOT NULL,
   col140 VARCHAR(86) NOT NULL,
   col141 VARCHAR(86) NOT NULL,
   col142 VARCHAR(86) NOT NULL,
   col143 VARCHAR(86) NOT NULL,
   col144 VARCHAR(86) NOT NULL,
   col145 VARCHAR(86) NOT NULL,
   col146 VARCHAR(86) NOT NULL,
   col147 VARCHAR(86) NOT NULL,
   col148 VARCHAR(86) NOT NULL,
   col149 VARCHAR(86) NOT NULL,
   col150 VARCHAR(86) NOT NULL,
   col151 VARCHAR(86) NOT NULL,
   col152 VARCHAR(86) NOT NULL,
   col153 VARCHAR(86) NOT NULL,
   col154 VARCHAR(86) NOT NULL,
   col155 VARCHAR(86) NOT NULL,
   col156 VARCHAR(86) NOT NULL,
   col157 VARCHAR(86) NOT NULL,
   col158 VARCHAR(86) NOT NULL,
   col159 VARCHAR(86) NOT NULL,
   col160 VARCHAR(86) NOT NULL,
   col161 VARCHAR(86) NOT NULL,
   col162 VARCHAR(86) NOT NULL,
   col163 VARCHAR(86) NOT NULL,
   col164 VARCHAR(86) NOT NULL,
   col165 VARCHAR(86) NOT NULL,
   col166 VARCHAR(86) NOT NULL,
   col167 VARCHAR(86) NOT NULL,
   col168 VARCHAR(86) NOT NULL,
   col169 VARCHAR(86) NOT NULL,
   col170 VARCHAR(86) NOT NULL,
   col171 VARCHAR(86) NOT NULL,
   col172 VARCHAR(86) NOT NULL,
   col173 VARCHAR(86) NOT NULL,
   col174 VARCHAR(86) NOT NULL,
   col175 VARCHAR(86) NOT NULL,
   col176 VARCHAR(86) NOT NULL,
   col177 VARCHAR(86) NOT NULL,
   col178 VARCHAR(86) NOT NULL,
   col179 VARCHAR(86) NOT NULL,
   col180 VARCHAR(86) NOT NULL,
   col181 VARCHAR(86) NOT NULL,
   col182 VARCHAR(86) NOT NULL,
   col183 VARCHAR(86) NOT NULL,
   col184 VARCHAR(86) NOT NULL,
   col185 VARCHAR(86) NOT NULL,
   col186 VARCHAR(86) NOT NULL,
   col187 VARCHAR(86) NOT NULL,
   col188 VARCHAR(86) NOT NULL,
   col189 VARCHAR(86) NOT NULL,
   col190 VARCHAR(86) NOT NULL,
   col191 VARCHAR(86) NOT NULL,
   col192 VARCHAR(86) NOT NULL,
   col193 VARCHAR(86) NOT NULL,
   col194 VARCHAR(86) NOT NULL,
   col195 VARCHAR(86) NOT NULL,
   col196 VARCHAR(86) NOT NULL,
   col197 VARCHAR(86) NOT NULL,
   col198 VARCHAR(86) NOT NULL,
   PRIMARY KEY (col1)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

And when using InnoDB's DYNAMIC row format and a default character set of utf8mb4 (which requires up to 4 bytes per character), the 256 byte limit means that a VARCHAR column will only be stored on overflow pages if it is at least as large as a `varchar(64)`:

```
SET GLOBAL innodb_default_row_format='dynamic';
SET SESSION innodb_strict_mode=ON;
CREATE OR REPLACE TABLE tab (
   col1 VARCHAR(64) NOT NULL,
   col2 VARCHAR(64) NOT NULL,
   col3 VARCHAR(64) NOT NULL,
   col4 VARCHAR(64) NOT NULL,
   col5 VARCHAR(64) NOT NULL,
   col6 VARCHAR(64) NOT NULL,
   col7 VARCHAR(64) NOT NULL,
   col8 VARCHAR(64) NOT NULL,
   col9 VARCHAR(64) NOT NULL,
   col10 VARCHAR(64) NOT NULL,
   col11 VARCHAR(64) NOT NULL,
   col12 VARCHAR(64) NOT NULL,
   col13 VARCHAR(64) NOT NULL,
   col14 VARCHAR(64) NOT NULL,
   col15 VARCHAR(64) NOT NULL,
   col16 VARCHAR(64) NOT NULL,
   col17 VARCHAR(64) NOT NULL,
   col18 VARCHAR(64) NOT NULL,
   col19 VARCHAR(64) NOT NULL,
   col20 VARCHAR(64) NOT NULL,
   col21 VARCHAR(64) NOT NULL,
   col22 VARCHAR(64) NOT NULL,
   col23 VARCHAR(64) NOT NULL,
   col24 VARCHAR(64) NOT NULL,
   col25 VARCHAR(64) NOT NULL,
   col26 VARCHAR(64) NOT NULL,
   col27 VARCHAR(64) NOT NULL,
   col28 VARCHAR(64) NOT NULL,
   col29 VARCHAR(64) NOT NULL,
   col30 VARCHAR(64) NOT NULL,
   col31 VARCHAR(64) NOT NULL,
   col32 VARCHAR(64) NOT NULL,
   col33 VARCHAR(64) NOT NULL,
   col34 VARCHAR(64) NOT NULL,
   col35 VARCHAR(64) NOT NULL,
   col36 VARCHAR(64) NOT NULL,
   col37 VARCHAR(64) NOT NULL,
   col38 VARCHAR(64) NOT NULL,
   col39 VARCHAR(64) NOT NULL,
   col40 VARCHAR(64) NOT NULL,
   col41 VARCHAR(64) NOT NULL,
   col42 VARCHAR(64) NOT NULL,
   col43 VARCHAR(64) NOT NULL,
   col44 VARCHAR(64) NOT NULL,
   col45 VARCHAR(64) NOT NULL,
   col46 VARCHAR(64) NOT NULL,
   col47 VARCHAR(64) NOT NULL,
   col48 VARCHAR(64) NOT NULL,
   col49 VARCHAR(64) NOT NULL,
   col50 VARCHAR(64) NOT NULL,
   col51 VARCHAR(64) NOT NULL,
   col52 VARCHAR(64) NOT NULL,
   col53 VARCHAR(64) NOT NULL,
   col54 VARCHAR(64) NOT NULL,
   col55 VARCHAR(64) NOT NULL,
   col56 VARCHAR(64) NOT NULL,
   col57 VARCHAR(64) NOT NULL,
   col58 VARCHAR(64) NOT NULL,
   col59 VARCHAR(64) NOT NULL,
   col60 VARCHAR(64) NOT NULL,
   col61 VARCHAR(64) NOT NULL,
   col62 VARCHAR(64) NOT NULL,
   col63 VARCHAR(64) NOT NULL,
   col64 VARCHAR(64) NOT NULL,
   col65 VARCHAR(64) NOT NULL,
```

```
    col66 VARCHAR(64) NOT NULL,
    col67 VARCHAR(64) NOT NULL,
    col68 VARCHAR(64) NOT NULL,
    col69 VARCHAR(64) NOT NULL,
    col70 VARCHAR(64) NOT NULL,
    col71 VARCHAR(64) NOT NULL,
    col72 VARCHAR(64) NOT NULL,
    col73 VARCHAR(64) NOT NULL,
    col74 VARCHAR(64) NOT NULL,
    col75 VARCHAR(64) NOT NULL,
    col76 VARCHAR(64) NOT NULL,
    col77 VARCHAR(64) NOT NULL,
    col78 VARCHAR(64) NOT NULL,
    col79 VARCHAR(64) NOT NULL,
    col80 VARCHAR(64) NOT NULL,
    col81 VARCHAR(64) NOT NULL,
    col82 VARCHAR(64) NOT NULL,
    col83 VARCHAR(64) NOT NULL,
    col84 VARCHAR(64) NOT NULL,
    col85 VARCHAR(64) NOT NULL,
    col86 VARCHAR(64) NOT NULL,
    col87 VARCHAR(64) NOT NULL,
    col88 VARCHAR(64) NOT NULL,
    col89 VARCHAR(64) NOT NULL,
    col90 VARCHAR(64) NOT NULL,
    col91 VARCHAR(64) NOT NULL,
    col92 VARCHAR(64) NOT NULL,
    col93 VARCHAR(64) NOT NULL,
    col94 VARCHAR(64) NOT NULL,
    col95 VARCHAR(64) NOT NULL,
    col96 VARCHAR(64) NOT NULL,
    col97 VARCHAR(64) NOT NULL,
    col98 VARCHAR(64) NOT NULL,
    col99 VARCHAR(64) NOT NULL,
    col100 VARCHAR(64) NOT NULL,
    col101 VARCHAR(64) NOT NULL,
    col102 VARCHAR(64) NOT NULL,
    col103 VARCHAR(64) NOT NULL,
    col104 VARCHAR(64) NOT NULL,
    col105 VARCHAR(64) NOT NULL,
    col106 VARCHAR(64) NOT NULL,
    col107 VARCHAR(64) NOT NULL,
    col108 VARCHAR(64) NOT NULL,
    col109 VARCHAR(64) NOT NULL,
    col110 VARCHAR(64) NOT NULL,
    col111 VARCHAR(64) NOT NULL,
    col112 VARCHAR(64) NOT NULL,
    col113 VARCHAR(64) NOT NULL,
    col114 VARCHAR(64) NOT NULL,
    col115 VARCHAR(64) NOT NULL,
    col116 VARCHAR(64) NOT NULL,
    col117 VARCHAR(64) NOT NULL,
    col118 VARCHAR(64) NOT NULL,
    col119 VARCHAR(64) NOT NULL,
    col120 VARCHAR(64) NOT NULL,
    col121 VARCHAR(64) NOT NULL,
    col122 VARCHAR(64) NOT NULL,
    col123 VARCHAR(64) NOT NULL,
    col124 VARCHAR(64) NOT NULL,
    col125 VARCHAR(64) NOT NULL,
    col126 VARCHAR(64) NOT NULL,
    col127 VARCHAR(64) NOT NULL,
    col128 VARCHAR(64) NOT NULL,
    col129 VARCHAR(64) NOT NULL,
    col130 VARCHAR(64) NOT NULL,
    col131 VARCHAR(64) NOT NULL,
    col132 VARCHAR(64) NOT NULL,
    col133 VARCHAR(64) NOT NULL,
    col134 VARCHAR(64) NOT NULL,
```

```
    col135 VARCHAR(64) NOT NULL,
    col136 VARCHAR(64) NOT NULL,
    col137 VARCHAR(64) NOT NULL,
    col138 VARCHAR(64) NOT NULL,
    col139 VARCHAR(64) NOT NULL,
    col140 VARCHAR(64) NOT NULL,
    col141 VARCHAR(64) NOT NULL,
    col142 VARCHAR(64) NOT NULL,
    col143 VARCHAR(64) NOT NULL,
    col144 VARCHAR(64) NOT NULL,
    col145 VARCHAR(64) NOT NULL,
    col146 VARCHAR(64) NOT NULL,
    col147 VARCHAR(64) NOT NULL,
    col148 VARCHAR(64) NOT NULL,
    col149 VARCHAR(64) NOT NULL,
    col150 VARCHAR(64) NOT NULL,
    col151 VARCHAR(64) NOT NULL,
    col152 VARCHAR(64) NOT NULL,
    col153 VARCHAR(64) NOT NULL,
    col154 VARCHAR(64) NOT NULL,
    col155 VARCHAR(64) NOT NULL,
    col156 VARCHAR(64) NOT NULL,
    col157 VARCHAR(64) NOT NULL,
    col158 VARCHAR(64) NOT NULL,
    col159 VARCHAR(64) NOT NULL,
    col160 VARCHAR(64) NOT NULL,
    col161 VARCHAR(64) NOT NULL,
    col162 VARCHAR(64) NOT NULL,
    col163 VARCHAR(64) NOT NULL,
    col164 VARCHAR(64) NOT NULL,
    col165 VARCHAR(64) NOT NULL,
    col166 VARCHAR(64) NOT NULL,
    col167 VARCHAR(64) NOT NULL,
    col168 VARCHAR(64) NOT NULL,
    col169 VARCHAR(64) NOT NULL,
    col170 VARCHAR(64) NOT NULL,
    col171 VARCHAR(64) NOT NULL,
    col172 VARCHAR(64) NOT NULL,
    col173 VARCHAR(64) NOT NULL,
    col174 VARCHAR(64) NOT NULL,
    col175 VARCHAR(64) NOT NULL,
    col176 VARCHAR(64) NOT NULL,
    col177 VARCHAR(64) NOT NULL,
    col178 VARCHAR(64) NOT NULL,
    col179 VARCHAR(64) NOT NULL,
    col180 VARCHAR(64) NOT NULL,
    col181 VARCHAR(64) NOT NULL,
    col182 VARCHAR(64) NOT NULL,
    col183 VARCHAR(64) NOT NULL,
    col184 VARCHAR(64) NOT NULL,
    col185 VARCHAR(64) NOT NULL,
    col186 VARCHAR(64) NOT NULL,
    col187 VARCHAR(64) NOT NULL,
    col188 VARCHAR(64) NOT NULL,
    col189 VARCHAR(64) NOT NULL,
    col190 VARCHAR(64) NOT NULL,
    col191 VARCHAR(64) NOT NULL,
    col192 VARCHAR(64) NOT NULL,
    col193 VARCHAR(64) NOT NULL,
    col194 VARCHAR(64) NOT NULL,
    col195 VARCHAR(64) NOT NULL,
    col196 VARCHAR(64) NOT NULL,
    col197 VARCHAR(64) NOT NULL,
    col198 VARCHAR(64) NOT NULL,
    PRIMARY KEY (col1)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Working Around the Problem

There are a few ways to work around this problem.

If you would like a solution for the problem instead of just working around it, then see the solutions mentioned in the previous section.

Refactoring the Table into Multiple Tables

A *safe* workaround is to refactor the single wide table, so that its columns are spread among multiple tables.

This workaround can even work if your table is so wide that the previous solutions have failed to solve them problem for your table.

Refactoring Some Columns into JSON

A *safe* workaround is to refactor some of the columns into a JSON document.

The JSON document can be queried and manipulated using MariaDB's JSON functions.

The JSON document can be stored in a column that uses one of the following data types:

- TEXT: The maximum size of a TEXT column is 64 KB.
- MEDIUMTEXT: The maximum size of a MEDIUMTEXT column is 16 MB.
- LONGTEXT: The maximum size of a LONGTEXT column is 4 GB.
- JSON: This is just an alias for the LONGTEXT data type.

This workaround can even work if your table is so wide that the previous solutions have failed to solve them problem for your table.

Disabling InnoDB Strict Mode

An *unsafe* workaround is to disable InnoDB strict mode. InnoDB strict mode can be disabled by setting the innodb_strict_mode system variable to `OFF`.

For example, even though the following table schema is too large for most InnoDB row formats to store, it can still be created when InnoDB strict mode is disabled:

```
SET GLOBAL innodb_default_row_format='dynamic';
SET SESSION innodb_strict_mode=OFF;
CREATE OR REPLACE TABLE tab (
   col1 VARCHAR(40) NOT NULL,
   col2 VARCHAR(40) NOT NULL,
   col3 VARCHAR(40) NOT NULL,
   col4 VARCHAR(40) NOT NULL,
   col5 VARCHAR(40) NOT NULL,
   col6 VARCHAR(40) NOT NULL,
   col7 VARCHAR(40) NOT NULL,
   col8 VARCHAR(40) NOT NULL,
   col9 VARCHAR(40) NOT NULL,
   col10 VARCHAR(40) NOT NULL,
   col11 VARCHAR(40) NOT NULL,
   col12 VARCHAR(40) NOT NULL,
   col13 VARCHAR(40) NOT NULL,
   col14 VARCHAR(40) NOT NULL,
   col15 VARCHAR(40) NOT NULL,
   col16 VARCHAR(40) NOT NULL,
   col17 VARCHAR(40) NOT NULL,
   col18 VARCHAR(40) NOT NULL,
   col19 VARCHAR(40) NOT NULL,
   col20 VARCHAR(40) NOT NULL,
   col21 VARCHAR(40) NOT NULL,
   col22 VARCHAR(40) NOT NULL,
   col23 VARCHAR(40) NOT NULL,
   col24 VARCHAR(40) NOT NULL,
   col25 VARCHAR(40) NOT NULL,
   col26 VARCHAR(40) NOT NULL,
   col27 VARCHAR(40) NOT NULL,
   col28 VARCHAR(40) NOT NULL,
   col29 VARCHAR(40) NOT NULL,
   col30 VARCHAR(40) NOT NULL,
   col31 VARCHAR(40) NOT NULL,
   col32 VARCHAR(40) NOT NULL,
   col33 VARCHAR(40) NOT NULL,
   col34 VARCHAR(40) NOT NULL,
   col35 VARCHAR(40) NOT NULL,
   col36 VARCHAR(40) NOT NULL,
   col37 VARCHAR(40) NOT NULL,
   col38 VARCHAR(40) NOT NULL,
   col39 VARCHAR(40) NOT NULL,
   col40 VARCHAR(40) NOT NULL,
   col41 VARCHAR(40) NOT NULL,
   col42 VARCHAR(40) NOT NULL,
   col43 VARCHAR(40) NOT NULL,
   col44 VARCHAR(40) NOT NULL,
   col45 VARCHAR(40) NOT NULL,
   col46 VARCHAR(40) NOT NULL,
   col47 VARCHAR(40) NOT NULL,
   col48 VARCHAR(40) NOT NULL,
   col49 VARCHAR(40) NOT NULL,
   col50 VARCHAR(40) NOT NULL,
   col51 VARCHAR(40) NOT NULL,
   col52 VARCHAR(40) NOT NULL,
   col53 VARCHAR(40) NOT NULL,
   col54 VARCHAR(40) NOT NULL,
   col55 VARCHAR(40) NOT NULL,
   col56 VARCHAR(40) NOT NULL,
   col57 VARCHAR(40) NOT NULL,
   col58 VARCHAR(40) NOT NULL,
   col59 VARCHAR(40) NOT NULL,
   col60 VARCHAR(40) NOT NULL,
   col61 VARCHAR(40) NOT NULL,
   col62 VARCHAR(40) NOT NULL,
   col63 VARCHAR(40) NOT NULL,
   col64 VARCHAR(40) NOT NULL,
   col65 VARCHAR(40) NOT NULL,
```

```
    col66 VARCHAR(40) NOT NULL,
    col67 VARCHAR(40) NOT NULL,
    col68 VARCHAR(40) NOT NULL,
    col69 VARCHAR(40) NOT NULL,
    col70 VARCHAR(40) NOT NULL,
    col71 VARCHAR(40) NOT NULL,
    col72 VARCHAR(40) NOT NULL,
    col73 VARCHAR(40) NOT NULL,
    col74 VARCHAR(40) NOT NULL,
    col75 VARCHAR(40) NOT NULL,
    col76 VARCHAR(40) NOT NULL,
    col77 VARCHAR(40) NOT NULL,
    col78 VARCHAR(40) NOT NULL,
    col79 VARCHAR(40) NOT NULL,
    col80 VARCHAR(40) NOT NULL,
    col81 VARCHAR(40) NOT NULL,
    col82 VARCHAR(40) NOT NULL,
    col83 VARCHAR(40) NOT NULL,
    col84 VARCHAR(40) NOT NULL,
    col85 VARCHAR(40) NOT NULL,
    col86 VARCHAR(40) NOT NULL,
    col87 VARCHAR(40) NOT NULL,
    col88 VARCHAR(40) NOT NULL,
    col89 VARCHAR(40) NOT NULL,
    col90 VARCHAR(40) NOT NULL,
    col91 VARCHAR(40) NOT NULL,
    col92 VARCHAR(40) NOT NULL,
    col93 VARCHAR(40) NOT NULL,
    col94 VARCHAR(40) NOT NULL,
    col95 VARCHAR(40) NOT NULL,
    col96 VARCHAR(40) NOT NULL,
    col97 VARCHAR(40) NOT NULL,
    col98 VARCHAR(40) NOT NULL,
    col99 VARCHAR(40) NOT NULL,
    col100 VARCHAR(40) NOT NULL,
    col101 VARCHAR(40) NOT NULL,
    col102 VARCHAR(40) NOT NULL,
    col103 VARCHAR(40) NOT NULL,
    col104 VARCHAR(40) NOT NULL,
    col105 VARCHAR(40) NOT NULL,
    col106 VARCHAR(40) NOT NULL,
    col107 VARCHAR(40) NOT NULL,
    col108 VARCHAR(40) NOT NULL,
    col109 VARCHAR(40) NOT NULL,
    col110 VARCHAR(40) NOT NULL,
    col111 VARCHAR(40) NOT NULL,
    col112 VARCHAR(40) NOT NULL,
    col113 VARCHAR(40) NOT NULL,
    col114 VARCHAR(40) NOT NULL,
    col115 VARCHAR(40) NOT NULL,
    col116 VARCHAR(40) NOT NULL,
    col117 VARCHAR(40) NOT NULL,
    col118 VARCHAR(40) NOT NULL,
    col119 VARCHAR(40) NOT NULL,
    col120 VARCHAR(40) NOT NULL,
    col121 VARCHAR(40) NOT NULL,
    col122 VARCHAR(40) NOT NULL,
    col123 VARCHAR(40) NOT NULL,
    col124 VARCHAR(40) NOT NULL,
    col125 VARCHAR(40) NOT NULL,
    col126 VARCHAR(40) NOT NULL,
    col127 VARCHAR(40) NOT NULL,
    col128 VARCHAR(40) NOT NULL,
    col129 VARCHAR(40) NOT NULL,
    col130 VARCHAR(40) NOT NULL,
    col131 VARCHAR(40) NOT NULL,
    col132 VARCHAR(40) NOT NULL,
    col133 VARCHAR(40) NOT NULL,
    col134 VARCHAR(40) NOT NULL,
```

```
   col135 VARCHAR(40) NOT NULL,
   col136 VARCHAR(40) NOT NULL,
   col137 VARCHAR(40) NOT NULL,
   col138 VARCHAR(40) NOT NULL,
   col139 VARCHAR(40) NOT NULL,
   col140 VARCHAR(40) NOT NULL,
   col141 VARCHAR(40) NOT NULL,
   col142 VARCHAR(40) NOT NULL,
   col143 VARCHAR(40) NOT NULL,
   col144 VARCHAR(40) NOT NULL,
   col145 VARCHAR(40) NOT NULL,
   col146 VARCHAR(40) NOT NULL,
   col147 VARCHAR(40) NOT NULL,
   col148 VARCHAR(40) NOT NULL,
   col149 VARCHAR(40) NOT NULL,
   col150 VARCHAR(40) NOT NULL,
   col151 VARCHAR(40) NOT NULL,
   col152 VARCHAR(40) NOT NULL,
   col153 VARCHAR(40) NOT NULL,
   col154 VARCHAR(40) NOT NULL,
   col155 VARCHAR(40) NOT NULL,
   col156 VARCHAR(40) NOT NULL,
   col157 VARCHAR(40) NOT NULL,
   col158 VARCHAR(40) NOT NULL,
   col159 VARCHAR(40) NOT NULL,
   col160 VARCHAR(40) NOT NULL,
   col161 VARCHAR(40) NOT NULL,
   col162 VARCHAR(40) NOT NULL,
   col163 VARCHAR(40) NOT NULL,
   col164 VARCHAR(40) NOT NULL,
   col165 VARCHAR(40) NOT NULL,
   col166 VARCHAR(40) NOT NULL,
   col167 VARCHAR(40) NOT NULL,
   col168 VARCHAR(40) NOT NULL,
   col169 VARCHAR(40) NOT NULL,
   col170 VARCHAR(40) NOT NULL,
   col171 VARCHAR(40) NOT NULL,
   col172 VARCHAR(40) NOT NULL,
   col173 VARCHAR(40) NOT NULL,
   col174 VARCHAR(40) NOT NULL,
   col175 VARCHAR(40) NOT NULL,
   col176 VARCHAR(40) NOT NULL,
   col177 VARCHAR(40) NOT NULL,
   col178 VARCHAR(40) NOT NULL,
   col179 VARCHAR(40) NOT NULL,
   col180 VARCHAR(40) NOT NULL,
   col181 VARCHAR(40) NOT NULL,
   col182 VARCHAR(40) NOT NULL,
   col183 VARCHAR(40) NOT NULL,
   col184 VARCHAR(40) NOT NULL,
   col185 VARCHAR(40) NOT NULL,
   col186 VARCHAR(40) NOT NULL,
   col187 VARCHAR(40) NOT NULL,
   col188 VARCHAR(40) NOT NULL,
   col189 VARCHAR(40) NOT NULL,
   col190 VARCHAR(40) NOT NULL,
   col191 VARCHAR(40) NOT NULL,
   col192 VARCHAR(40) NOT NULL,
   col193 VARCHAR(40) NOT NULL,
   col194 VARCHAR(40) NOT NULL,
   col195 VARCHAR(40) NOT NULL,
   col196 VARCHAR(40) NOT NULL,
   col197 VARCHAR(40) NOT NULL,
   col198 VARCHAR(40) NOT NULL,
   PRIMARY KEY (col1)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

But as mentioned above, if InnoDB strict mode is **disabled** and if a DDL statement is executed, then InnoDB will still raise a **warning** with this message. The SHOW WARNINGS statement can be used to view the warning:

```
SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------------------------------
-----------------------------------------------------+
| Level | Code | Message |
+---------+------+-----------------------------------------------------------------------------------------------
-----------------------------------------------------+
| Warning | 139 | Row size too large (> 8126). Changing some columns to TEXT or BLOB may help. In current
row format, BLOB prefix of 0 bytes is stored inline. |
+---------+------+-----------------------------------------------------------------------------------------------
-----------------------------------------------------+
1 row in set (0.000 sec)
```

As mentioned above, even though InnoDB is allowing the table to be created, there is still an opportunity for errors. Regardless of whether InnoDB strict mode is enabled, if a DML statement is executed that attempts to write a row that the table's InnoDB row format can't store, then InnoDB will raise an **error** with this message. This creates a somewhat *unsafe* situation, because it means that the application has the chance to encounter an additional error while executing DML.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Tablespaces

See InnoDB File-Per-Table Tablespaces.

Learn how to configure InnoDB to store each table in its own .ibd file, enabling features like table compression and easier space reclamation.

See InnoDB System Tablespaces.

This page explains the InnoDB system tablespace (ibdata1), which stores the data dictionary, doublewrite buffer, and undo logs, and how to resize it.

See InnoDB Temporary Tablespaces.

This page explains how InnoDB manages temporary tablespaces for non-compressed temporary tables, including configuration and sizing options.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB File-Per-Table Tablespaces

When you create a table using the InnoDB storage engine, data written to that table is stored on the file system in a data file called a tablespace. Tablespace files contain both the data and indexes.

When innodb_file_per_table=ON is set, InnoDB uses one tablespace file per InnoDB table. These tablespace files have the `.ibd` extension. When innodb_file_per_table=OFF is set, InnoDB stores all tables in the InnoDB system tablespace.

InnoDB versions in MySQL 5.7 and above also support an additional type of tablespace called general tablespaces that are created with CREATE TABLESPACE. However, InnoDB versions in MariaDB Server do not support general tablespaces or CREATE TABLESPACE.

File-Per-Table Tablespace Locations

By default, InnoDB's file-per-table tablespaces are created in the system's data directory, which is defined by the datadir system variable. The system variable innodb_data_home_dir will not change the location of file-per-table tablespaces.

In the event that you have a specific tablespace that you need stored in a dedicated path, you can set the location using the DATA DIRECTORY table option when you create the table.

For instance,

```
CREATE TABLE test.t1 (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(50)
) ENGINE=InnoDB
DATA DIRECTORY = "/data/contact";
```

MariaDB then creates a database directory on the configured path and the file-per-table tablespace are created inside that directory. On Unix-like operating systems, you can see the file using the ls command:

```
# ls -al /data/contact/test
drwxrwx--- 2 mysql mysql  4096 Dec 8 18:46 .
drwxr-xr-x 3 mysql mysql  4096 Dec 8 18:46 ..
-rw-rw---- 1 mysql mysql 98304 Dec 8 20:41 t1.ibd
```

Note, the system user that runs the MariaDB Server process (which is usually `mysql`) must have write permissions on the given path.

Copying Transportable Tablespaces

InnoDB's file-per-table tablespaces are transportable, which means that you can copy a file-per-table tablespace from one MariaDB Server to another server. You may find this useful in cases where you need to transport full tables between servers and don't want to use backup tools like mariadb-backup or mariadb-dump. In fact, this process can even be used with mariadb-backup in some cases, such as when restoring partial backups or when restoring individual tables or partitions from a backup.

Copying Transportable Tablespaces for Non-partitioned Tables

You can copy the transportable tablespace of a non-partitioned table from one server to another by exporting the tablespace file from the original server, and then importing the tablespace file into the new server.

MariaDB starting with 11.2.1

The workflow is simplified starting from MariaDB 11.2.1. On the source server, simply do:

```
FLUSH TABLES t1 FOR EXPORT;
# scp /data/contacts/test/t1.ibd target-server.com:/var/lib/mysql/test/
# scp /data/contacts/test/t1.cfg target-server.com:/var/lib/mysql/test/
# scp /data/contacts/test/t1.frm target-server.com:/var/lib/mysql/test/
UNLOCK TABLES;
```

On the destination server, simply do:

```
ALTER TABLE t1 IMPORT TABLESPACE;
```

Exporting Transportable Tablespaces for Non-partitioned Tables

You can export a non-partitioned table by locking the table and copying the table's `.ibd` and `.cfg` files from the relevant tablespace location for the table to a backup location. For example, the process would go like this:

- First, use the FLUSH TABLES … FOR EXPORT statement on the target table:

```
FLUSH TABLES test.t1 FOR EXPORT;
```

This forces the server to close the table and provides your connection with a read lock on the table.

- Then, while your connection still holds the lock on the table, copy the tablespace file and the metadata file to a safe directory:

```
# cp /data/contacts/test/t1.ibd /data/saved-tablespaces/
# cp /data/contacts/test/t1.cfg /data/saved-tablespaces/
```

- Then, once you've copied the files, you can release the lock with UNLOCK TABLES:

```
UNLOCK TABLES;
```

Importing Transportable Tablespaces for Non-partitioned Tables

You can import a non-partitioned table by discarding the table's original tablespace, copying the table's `.ibd` and `.cfg` files from the backup location to the relevant tablespace location for the table, and then telling the server to import the tablespace.

For example, the process would go like this:

- First, on the destination server, you need to create a copy of the table. Use the same CREATE TABLE statement that was used to create the table on the original server.

```
CREATE TABLE test.t1 (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(50)
) ENGINE=InnoDB;
```

- Then, use ALTER TABLE … DISCARD TABLESPACE to discard the new table's tablespace:

```
ALTER TABLE test.t1 DISCARD TABLESPACE;
```

- Then, copy the `.ibd` and `.cfg` files from the original server to the relevant directory on the target MariaDB Server.

```
# scp /data/tablespaces/t1.ibd target-server.com:/var/lib/mysql/test/
# scp /data/tablespaces/t1.cfg target-server.com:/var/lib/mysql/test/
```

File-per-table tablespaces can be imported with just the `.ibd` file in many cases. If you do not have the tablespace's `.cfg` file for whatever reason, then it is usually worth trying to import the tablespace with just the `.ibd` file.

- Then, once the files are in the proper directory on the target server, use ALTER TABLE … IMPORT TABLESPACE to import the new table's tablespace:

```
ALTER TABLE test.t1 IMPORT TABLESPACE;
```

Copying Transportable Tablespaces for Partitioned Tables

Currently, MariaDB does not directly support the transport of tablespaces from partitioned tables. See MDEV-10568 for more information about that. It is still possible to transport partitioned tables if we use a workaround. You can copy the transportable tablespaces of a partitioned table from one server to another by exporting the tablespace file of each partition from the original server, and then importing the tablespace file of each partition into the new server.

Exporting Transportable Tablespaces for Partitioned Tables

You can export a partitioned table by locking the table and copying the `.ibd` and `.cfg` files of each partition from the relevant tablespace location for the partition to a backup location. For example, the process would go like this:

- First, let's create a test table with some data on the original server:

```
CREATE TABLE test.t2 (
   employee_id INT,
   name VARCHAR(50)
) ENGINE=InnoDB
PARTITION BY RANGE (employee_id) (
   PARTITION p0 VALUES LESS THAN (6),
   PARTITION p1 VALUES LESS THAN (11),
   PARTITION p2 VALUES LESS THAN (16),
   PARTITION p3 VALUES LESS THAN MAXVALUE
);

INSERT INTO test.t2 (name, employee_id) VALUES
   ('Geoff Montee', 1),
   ('Chris Calendar', 6),
   ('Kyle Joiner', 11),
   ('Will Fong', 16);
```

- Then, we need to export the partitioned tablespace from the original server, which follows the same process as exporting non-partitioned tablespaces. That means that we need to use the FLUSH TABLES … FOR EXPORT statement on the target table:

```
FLUSH TABLES test.t2 FOR EXPORT;
```

This forces the server to close the table and provides your connection with a read lock on the table.

- Then, if we grep the database directory in the data directory for the newly created `t2` table, we can see a number of `.ibd` and `.cfg` files for the table:

```
# ls -l /var/lib/mysql/test/ | grep t2
total 428
-rw-rw---- 1 mysql mysql 827 Dec 5 16:08 t2.frm
-rw-rw---- 1 mysql mysql 48 Dec 5 16:08 t2.par
-rw-rw---- 1 mysql mysql 579 Dec 5 18:47 t2#P#p0.cfg
-rw-r----- 1 mysql mysql 98304 Dec 5 16:43 t2#P#p0.ibd
-rw-rw---- 1 mysql mysql 579 Dec 5 18:47 t2#P#p1.cfg
-rw-rw---- 1 mysql mysql 98304 Dec 5 16:08 t2#P#p1.ibd
-rw-rw---- 1 mysql mysql 579 Dec 5 18:47 t2#P#p2.cfg
-rw-rw---- 1 mysql mysql 98304 Dec 5 16:08 t2#P#p2.ibd
-rw-rw---- 1 mysql mysql 579 Dec 5 18:47 t2#P#p3.cfg
-rw-rw---- 1 mysql mysql 98304 Dec 5 16:08 t2#P#p3.ibd
```

- Then, while our connection still holds the lock on the table, we need to copy the tablespace files and the metadata files to a safe directory:

```
$ mkdir /tmp/backup
$ sudo cp /var/lib/mysql/test/t2*.ibd /tmp/backup
$ sudo cp /var/lib/mysql/test/t2*.cfg /tmp/backup
```

- Then, once we've copied the files, we can release the lock with UNLOCK TABLES:

```
UNLOCK TABLES;
```

Importing Transportable Tablespaces for Partitioned Tables

You can import a partitioned table by creating a placeholder table, discarding the placeholder table's original tablespace, copying the partition's `.ibd` and `.cfg` files from the backup location to the relevant tablespace location for the placeholder table, and then telling the server to import the tablespace. At that point, the server can exchange the tablespace for the placeholder table with the one for the partition. For example, the process would go like this:

- First, we need to copy the saved tablespace files from the original server to the target server:

```
$ scp /tmp/backup/t2* user@target-host:/tmp/backup
```

- Then, we need to import the partitioned tablespaces onto the target server. The import process for partitioned tables is more complicated than the import process for non-partitioned tables. To start with, if it doesn't already exist, then we need to create a partitioned table on the target server that matches the partitioned table on the original server:

```
CREATE TABLE test.t2 (
   employee_id INT,
   name VARCHAR(50)
) ENGINE=InnoDB
PARTITION BY RANGE (employee_id) (
   PARTITION p0 VALUES LESS THAN (6),
   PARTITION p1 VALUES LESS THAN (11),
   PARTITION p2 VALUES LESS THAN (16),
   PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

- Then, using this table as a model, we need to create a placeholder of this table with the same structure that does not use partitioning. This can be done with a CREATE TABLE… AS SELECT statement:

```
CREATE TABLE test.t2_placeholder LIKE test.t2;
ALTER TABLE test.t2_placeholder REMOVE PARTITIONING;
```

This statement will create a new table called `t2_placeholder` that has the same schema structure as `t2`, but it does not use partitioning and it contains no rows.

For Each Partition

From this point forward, the rest of our steps need to happen for each individual partition. For each partition, we need to do the following process:

- First, we need to use ALTER TABLE … DISCARD TABLESPACE to discard the placeholder table's tablespace:

```
ALTER TABLE test.t2_placeholder DISCARD TABLESPACE;
```

- Then, copy the `.ibd` and `.cfg` files for the next partition to the relevant directory for the `t2_placeholder` table on the target MariaDB Server:

```
# cp /tmp/backup/t2#P#p0.cfg /var/lib/mysql/test/t2_placeholder.cfg
# cp /tmp/backup/t2#P#p0.ibd /var/lib/mysql/test/t2_placeholder.ibd
# chown mysql:mysql /var/lib/mysql/test/t2_placeholder*
```

File-per-table tablespaces can be imported with just the `.ibd` file in many cases. If you do not have the tablepace's `.cfg` file for whatever reason, then it is usually worth trying to import the tablespace with just the `.ibd` file.

- Then, once the files are in the proper directory on the target server, we need to use ALTER TABLE … IMPORT TABLESPACE to import the new table's tablespace:

```
ALTER TABLE test.t2_placeholder IMPORT TABLESPACE;
```

The placeholder table now contains data from the `p0` partition on the source server.

```
SELECT * FROM test.t2_placeholder;

+-------------+--------------+
| employee_id | name         |
+-------------+--------------+
|           1 | Geoff Montee |
+-------------+--------------+
```

- Then, it's time to transfer the partition from the placeholder to the target table. This can be done with an ALTER TABLE… EXCHANGE PARTITION statement:

```
ALTER TABLE test.t2 EXCHANGE PARTITION p0 WITH TABLE test.t2_placeholder;
```

The target table now contains the first partition from the source table.

```
SELECT * FROM test.t2;

+-------------+--------------+
| employee_id | name         |
+-------------+--------------+
|           1 | Geoff Montee |
+-------------+--------------+
```

- Repeat this procedure for each partition you want to import. For each partition, we need to discard the placeholder table's tablespace, and then import the partitioned table's tablespace into the placeholder table, and then exchange the tablespaces between the placeholder table and the partition of our target table.

When this process is complete for all partitions, the target table will contain the imported data:

```
SELECT * FROM test.t2;

+-------------+----------------+
| employee_id | name           |
+-------------+----------------+
|           1 | Geoff Montee   |
|           6 | Chris Calendar |
|          11 | Kyle Joiner    |
|          16 | Will Fong      |
+-------------+----------------+
```

- Then, we can remove the placeholder table from the database:

```
DROP TABLE test.t2_placeholder;
```

Known Problems with Copying Transportable Tablespaces

Differing Storage Formats for Temporal Columns

MariaDB 10.1.2 added the mysql56_temporal_format system variable, which enables a new MySQL 5.6-compatible storage format for the TIME, DATETIME and TIMESTAMP data types.

If a file-per-tablespace file contains columns that use one or more of these temporal data types and if the tablespace file's original table was created with a certain storage format for these columns, then the tablespace file can only be imported into tables that were also created with the same storage format for these columns as the original table. Otherwise, you will see errors like the following:

```
ALTER TABLE dt_test IMPORT TABLESPACE;
ERROR 1808 (HY000): Schema mismatch (Column dt precise type mismatch.)
```

See MDEV-15225 for more information.

See the pages for the TIME, DATETIME and TIMESTAMP data types to determine how to update the storage format for temporal columns in tables that were created before MariaDB 10.1.2 or that were created with mysql56_temporal_format=OFF.

Differing ROW_FORMAT Values

InnoDB file-per-table tablespaces can use different row formats. A specific row format can be specified when creating a table either by setting the ROW_FORMAT table option or by the setting the innodb_default_row_format system variable. See Setting a Table's Row Format for more information on how to set an InnoDB table's row format.

If a file-per-tablespace file was created with a certain row format, then the tablespace file can only be imported into tables that were created with the same row format as the original table. Otherwise, you will see errors like the following:

```
ALTER TABLE t0 IMPORT TABLESPACE;
ERROR 1808 (HY000): Schema mismatch (Expected FSP_SPACE_FLAGS=0x21, .ibd file contains 0x0.)
```

The error message is a bit more descriptive in MariaDB 10.2.17 and later:

```
ALTER TABLE t0 IMPORT TABLESPACE;
ERROR 1808 (HY000): Schema mismatch (Table flags don't match, server table has 0x1 and the meta-data file
has 0x0; .cfg file uses ROW_FORMAT=REDUNDANT)
```

Be sure to check a tablespace's row format before moving it from one server to another. Keep in mind that the default row format can change between major versions of MySQL or MariaDB. See Checking a Table's Row Format for information on how to check an InnoDB table's row format.

See MDEV-15049 and MDEV-16851 for more information.

Foreign Key Constraints

DISCARD on a table with foreign key constraints is only possible after disabling foreign_key_checks:

```
SET SESSION foreign_key_checks=0;
ALTER TABLE t0 DISCARD TABLESPACE;
```

IMPORT on the other hand does not enforce foreign key constraints. So when importing tablespaces, referential integrity can only be guaranteed to import all tables bound by foreign key constraint at the same time, from an EXPORT of those tables taken with the same transactional state.

Tablespace Encryption

MariaDB supports data-at-rest encryption for the InnoDB storage engine. When enabled, the Server encrypts data before writing it to the tablespace and decrypts reads from the tablespace before returning result-sets. This means that a malicious user attempting to exfiltrate sensitive data won't be able to import the tablespace onto a different server as shown above without the encryption key.

For more information on data encryption, see Encrypting Data for InnoDB.

See Also

- Geoff Montee:Importing InnoDB Partitions in MySQL 5.6 and MariaDB 10.0/10.1

This page is licensed: CC BY-SA / Gnu FDL

InnoDB System Tablespaces

When InnoDB needs to store general information relating to the system as a whole, rather than a specific table, the specific file it writes to is the system tablespace. By default, this is the `ibdata1` file located in the data directory, (as defined by either the datadir or innodb_data_home_dir system variables). InnoDB uses the system tablespace to store the data dictionary, change buffer, and undo logs.

You can define the system tablespace filename or filenames, size and other options by setting the innodb_data_file_path system variable. This system variable can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
innodb_data_file_path=ibdata1:50M:autoextend
```

This system variable defaults to the file `ibdata1`, and it defaults to a minimum size of `12M`, and it defaults with the `autoextend` attribute enabled.

Changing Sizes

InnoDB defaults to allocating 12M to the `ibdata1` file for the system tablespace. While this is sufficient for most use cases, it may not be for all. You may find after using MariaDB for a while that the allocation is too small for the system tablespace or it grows too large for your disk. Fortunately, you can adjust this size as need later.

Increasing the Size

When setting the innodb_data_file_path system variable, you can define a size for each file given. In cases where you need a larger system tablespace, add the `autoextend` option to the last value.

```
[mariadb]
...
innodb_data_file_path=ibdata1:12M;ibdata2:50M:autoextend
```

Under this configuration, when the last system tablespace grows beyond the size allocation, InnoDB increases the size of the file by increments. To control the allocation increment, set the innodb_autoextend_increment system variable.

Decreasing the Size

MariaDB starting with 11.2

From MariaDB 11.2.0, when MariaDB starts up, unused InnoDB tablespace can be reclaimed, reducing the file size (MDEV-14795). This is disabled by default and is enabled by adding the `:autoshrink` attribute to the innodb_data_file_path system variable, e.g.:

```
[mariadb]
...
innodb_data_file_path=ibdata1:12M;ibdata2:50M:autoextend:autoshrink
```

Alternatively, starting with MariaDB 11.2.3, the shrinking can be set to happen during a slow shutdown (e.g. after `SET GLOBAL innodb_fast_shutdown=0`) (MDEV-32452).
Technically, how this works is:

1. Find the last used extent in the system tablespace by iterating through the BITMAP in the extent descriptor page.
2. Check whether the tablespace is being used within fixed size, and if the last used extent is less than the fixed size, then set the desired target size to fixed size.
3. Flush all pages belonging to the system tablespace in flush list.

4. Truncate the truncated pages from FSP_FREE and FSP_FREE_FRAG list.
5. Reset the bitmap in descriptor pages for the truncated pages.
6. Update the FSP_SIZE and FSP_FREE_LIMIT in header page.
7. In case of multiple files, calculate the truncated last file size and do the truncation in last file.

MariaDB until 11.2.0

In cases where the InnoDB system tablespace has grown too large, before MariaDB 11.2, the process to reduce it in size is a little more complicated than increasing the size. MariaDB does not allow you to remove data from the tablespace file itself. Instead you need to delete the tablespace files themselves, then restore the database from backups.
The backup utility mariadb-dump produces backup files containing the SQL statements needed to recreate the database. As a result, it restores a database with the bare minimum data rather than any additional information that might have built up in the tablespace file.
Use mariadb-dump to backup all of your InnoDB database tables, including the system tables in the `mysql` database that use InnoDB. You can find out what they are using the Information Schema.

```
SELECT TABLE_NAME FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'mysql' AND ENGINE = 'InnoDB';
```

If you only use InnoDB, you may find it easier to back up all databases and tables.

```
$ mariadb-dump -u root -p --all-databases > full-backup.sql
```

Then stop the MariaDB Server and remove the InnoDB tablespace files. In the data directory or the InnoDB data home directory, delete all the `ibdata` and `ib_log` files as well as any file with an `.ibd` or `.frm` extension.
Once this is done, restart the server and import the dump file:

```
$ mysql -u root -p < full-backup.sql
```

Using Raw Disk Partitions

Instead of having InnoDB write to the file system, you can set it to use raw disk partitions. On Windows and some Linux distributions, this allows you to perform non-buffered I/O without the file system overhead. Note that in many use cases this may not actually improve performance. Run tests to verify if there are any real gains for your application usage.

To enable a raw disk partition, first start MariaDB with the `newraw` option set on the tablespace:

```
[mariadb]
...
innodb_data_file_path=/dev/sdc:10Gnewraw
```

When the MariaDB Server starts, it initializes the partition. Don't create or change any data, (any data written to InnoDB at this stage are lost on restart). Once the server has successful started, stop it then edit the configuration file again, changing the `newraw` keyword to `raw`.

```
[mariadb]
...
innodb_data_file_path=/dev/sdc:10Graw
```

When you start MariaDB again, it'll read and write InnoDB data to the given disk partition instead of the file system.

Raw Disk Partitions on Windows

When defining a raw disk partition for InnoDB on the Windows operating system, use the same procedure as defined above, but when defining the path for the innodb_data_file_path system variable, use `./` at the start:

```
[mariadb]
...
innodb_data_file_path=//./E::10Graw
```

The given path is synonymous with the Windows syntax for accessing the physical drive.

System Tables within the InnoDB System Tablespace

InnoDB creates some system tables within the InnoDB System Tablespace:

- `SYS_DATAFILES`
- `SYS_FOREIGN`
- `SYS_FOREIGN_COLS`
- `SYS_TABLESPACES`
- `SYS_VIRTUAL`
- `SYS_ZIP_DICT`
- `SYS_ZIP_DICT_COLS`

These tables cannot be queried. However, you might see references to them in some places, such as in the INNODB_SYS_TABLES table in the information_schema database.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Temporary Tablespaces

When the user creates a temporary table using the CREATE TEMPORARY TABLE statement and the engine is set as InnoDB, MariaDB creates a temporary tablespace file. When the table is not compressed, MariaDB writes to a shared temporary tablespace as defined by the innodb_temp_data_file_path system variable. MariaDB does not allow the creation of `ROW_FORMAT=COMPRESSED` temporary tables. All temporary tables are uncompressed. MariaDB deletes temporary tablespaces when the server shuts down gracefully and is recreated when it starts again. It cannot be placed on a raw device.

Internal temporary tablespaces, (that is, temporary tables that cannot be kept in memory) use either Aria or MyISAM, depending on the aria_used_for_temp_tables system variable. You can set the default storage engine for user-created temporary tables using the default_tmp_storage_engine system variable.

Prior to MariaDB 10.2, temporary tablespaces existed as part of the InnoDB system tablespace or were file-per-table depending on the configuration of the innodb_file_per_table system variable.

Syntax for the value of the innodb_temp_data_file_path variable

The options for innodb_temp_data_file_path is one path or a set of paths, separated by ';'.

The syntax for each path is:

```
path:size[:autoextend][:max:size][:autoshrink]
```

`size` can have extensions 'G' (Gigabytes), 'M' (Megabytes) or 'K' (Kilobytes). If no extension is given, then megabytes is assumed.

- The first `size` argument is the initial size of the temporary table space.
- `autoextend` means that the file size will automatically increase if needed.

- `max` can be used to limit the total size of the temporary file if `autoextend` is used.
- `autoshrink` means that the file will shrink to original size when possible.

Only the last path can have the `autoextend` , `max` and `autoshrink` options.

Sizing Temporary Tablespaces

In order to size temporary tablespaces, use the innodb_temp_data_file_path system variable. This system variable can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
innodb_temp_data_file_path=ibtmp1:32M:autoextend
```

This system variable's syntax is the same as the innodb_data_file_path system variable. That is, a file name, size and option. By default, it writes a 12MB autoextending file to `ibtmp1` in the data directory.

To increase the size of the temporary tablespace, you can add a path to an additional tablespace file to the value of the innodb_temp_data_file_path system variable. Providing additional paths allows you to spread the temporary tablespace between multiple tablespace files. The last file can have the `autoextend` attribute, which ensures that you won't run out of space:

```
[mariadb]
...
innodb_temp_data_file_path=ibtmp1:32M;ibtmp2:32M:autoextend
```

Unlike normal tablespaces, temporary tablespaces are deleted when you stop MariaDB. To shrink temporary tablespaces to their minimum sizes, restart the server.

Shrinking the Tablespace

From MariaDB 11.3.0, the temporary tablespace can be shrunk by setting innodb_truncate_temporary_tablespace_now to `ON`:

```
SET GLOBAL innodb_truncate_temporary_tablespace_now=1;
```

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Troubleshooting

See InnoDB Troubleshooting Overview.

A starting point for diagnosing InnoDB issues, recommending checks on error logs, deadlocks, and table integrity using various tools.

See InnoDB Data Dictionary Troubleshooting.

Learn how to resolve inconsistencies between the InnoDB internal data dictionary and the file system, such as orphan .frm or .ibd files.

See InnoDB Recovery Modes.

Understand the different `innodb_force_recovery` levels, which allow you to start the server in read-only modes to recover data after a crash.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Troubleshooting Overview

As with most errors, first take a look at the contents of the MariaDB error log. If dealing with a deadlock, setting the innodb_print_all_deadlocks option (off by default) will output details of all deadlocks to the error log.

It can also help to enable the various InnoDB Monitors relating to the problem you are experiencing. There are four types: the standard InnoDB monitor, the InnoDB Lock Monitor, InnoDB Tablespace Monitor and the InnoDB Table Monitor.

Running CHECK TABLE will help determine whether there are errors in the table.

For problems with the InnoDB Data Dictionary, see InnoDB Data Dictionary Troubleshooting.

See Also

- InnoDB Data Dictionary Troubleshooting
- InnoDB Recovery Modes
- Error Codes

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Data Dictionary Troubleshooting

Can't Open File

If InnoDB returns something like the following error:

```
ERROR 1016: Can't open file: 'x.ibd'. (errno: 1)
```

it may be that an orphan `.frm` file exists. Something like the following may also appear in the error log:

```
InnoDB: Cannot find table test/x from the internal data dictionary
InnoDB: of InnoDB though the .frm file for the table exists. Maybe you
InnoDB: have deleted and recreated InnoDB data files but have forgotten
InnoDB: to delete the corresponding .frm files of InnoDB tables?
```

If this is the case, as the text describes, delete the orphan `.frm` file on the filesystem.

Could not find a valid tablespace file

In this case the table definition, the `.frm` file, is missing and the InnoDB dictionary expects it to be there. To remove the InnoDB dictionary entry, the existence of the file needs to faked and then dropped. The existence of the file is faked by creating the `tablename.frm` and potential the database directory if it is missing. Then a `DROP TABLE` or `DROP DATABASE` can be executed which will remove the InnoDB dictionary entry.

Use the query to identify potentially other tablespaces that are known but missing with:

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE NAME LIKE 'dbname/%';
```

Removing Orphan Intermediate Tables

An orphan intermediate table may prevent you from dropping the tablespace even if it is otherwise empty, and generally takes up unnecessary space.

It may come about if MariaDB exits in the middle of an ALTER TABLE … ALGORITHM=INPLACE operation. They are listed in the INFORMATION_SCHEMA.INNODB_SYS_TABLES table, and always start with an `#sql-ib` prefix. The accompanying `.frm` file also begins with `#sql-`, but has a different name.

To identify orphan tables, run:

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE NAME LIKE '%#sql%';
```

When innodb_file_per_table is set, the `#sql-*.ibd` file will also be visible in the database directory.

To remove an orphan intermediate table:

- Rename the `#sql-*.frm` file (in the database directory) to match the base name of the orphan intermediate table, for example:

```
mv #sql-36ab_2.frm #sql-ib87-856498050.frm
```

- Drop the table, for example:

```
DROP TABLE `#mysql50##sql-ib87-856498050`;
```

See Also

- InnoDB Troubleshooting Overview

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Recovery Modes

The InnoDB recovery mode is a mode used for recovering from emergency situations. You should ensure you have a backup of your database before making changes in case you need to restore it. The innodb_force_recovery server system variable sets the recovery mode. A mode of 0 is normal use, while the higher the mode, the more stringent the restrictions. Higher modes incorporate all limitations of the lower modes.

The recovery mode should never be set to a value other than zero except in an emergency situation.

Please note that recovery mode does not repair corruption. The corrupted files remain corrupted regardless of recovery mode. The sole purpose of recovery mode is to allow read access to the data, if at all possible.

Generally, it is best to start with a recovery mode of 1, and increase in single increments if needs be. With a recovery mode < 4, only corrupted pages should be lost. With 4, secondary indexes could be corrupted. With 5, results could be inconsistent and secondary indexes could be corrupted (even if they were not with 4). A value of 6 leaves pages in an obsolete state, which might cause more corruption.

Until MariaDB 10.2.7, mode `0` was the only mode permitting changes to the data. From MariaDB 10.2.7, write transactions are permitted with mode `3` or less.

To recover the tables, you can execute SELECTs to dump data, and DROP TABLE (when write transactions are permitted) to remove corrupted tables.

The following modes are available:

Recovery Modes

Recovery mode behaviour differs between versions (`server/storage/innobase/include/srv0srv.h`).

CURRENT

Mode	Description
0	The default mode while InnoDB is running normally. Write transactions are permitted with innodb_force_recovery<=4.

Mode	Description
1	(SRV_FORCE_IGNORE_CORRUPT) allows the server to keep running even if corrupt pages are detected. It does so by making redo log based recovery ignore certain errors, such as missing data files or corrupted data pages. Any redo log for affected files or pages are skipped. You can facilitate dumping tables by getting the SELECT * FROM table_name statement to jump over corrupt indexes and pages.
2	(SRV_FORCE_NO_BACKGROUND) stops the master thread from running, preventing a crash that occurs during a purge. No purge are performed, so the undo logs will keep growing.
3	(SRV_FORCE_NO_TRX_UNDO) does not roll back DML transactions after the crash recovery. Does not affect rollback of currently active DML transactions. Will also prevent some undo-generating background tasks from running. These tasks could hit a lock wait due to the recovered incomplete transactions whose rollback is being prevented.
4	(SRV_FORCE_NO_DDL_UNDO) does not roll back transactions after the crash recovery. Does not affect rollback of currently active transactions. Will also prevent some undo-generating background tasks from running. These tasks could hit a lock wait due to the recovered incomplete transactions whose rollback is being prevented.
5	(SRV_FORCE_NO_UNDO_LOG_SCAN) treats incomplete transactions as committed, and does not look at the undo logs when starting. Any DDL log for InnoDB tables are essentially ignored by InnoDB, but the server will start up
6	(SRV_FORCE_NO_LOG_REDO) does not perform redo log roll-forward as part of recovery. Running queries that require indexes are likely to fail with this mode active. However, if a table dump still causes a crash, you can try using a SELECT * FROM tab ORDER BY primary_key DESC to dump all the data portion after the corrupted part.

< 10.6.5

Mode	Description
0	The default mode while InnoDB is running normally. Write transactions are permitted with innodb_force_recovery<=4.
1	(SRV_FORCE_IGNORE_CORRUPT) allows the server to keep running even if corrupt pages are detected. It does so by making redo log based recovery ignore certain errors, such as missing data files or corrupted data pages. Any redo log for affected files or pages are skipped. You can facilitate dumping tables by getting the SELECT * FROM table_name statement to jump over corrupt indexes and pages.
2	(SRV_FORCE_NO_BACKGROUND) stops the master thread from running, preventing a crash that occurs during a purge. No purge are performed, so the undo logs will keep growing.
3	(SRV_FORCE_NO_TRX_UNDO) does not roll back transactions after the crash recovery. Does not affect rollback of currently active transactions. Will also prevent some undo-generating background tasks from running. These tasks could hit a lock wait due to the recovered incomplete transactions whose rollback is being prevented.
4	(SRV_FORCE_NO_IBUF_MERGE) The same as 3.
5	(SRV_FORCE_NO_UNDO_LOG_SCAN) treats incomplete transactions as committed, and does not look at the undo logs when starting.
6	(SRV_FORCE_NO_LOG_REDO) does not perform redo log roll-forward as part of recovery. Running queries that require indexes are likely to fail with this mode active. However, if a table dump still causes a crash, you can try using a SELECT * FROM tab ORDER BY primary_key DESC to dump all the data portion after the corrupted part.

< 10.5

Mode	Description
0	The default mode while InnoDB is running normally. Until MariaDB 10.2.7, it was the only mode permitting changes to the data. From MariaDB 10.2.7, write transactions are permitted with innodb_force_recovery<=3.
1	(SRV_FORCE_IGNORE_CORRUPT) allows the server to keep running even if corrupt pages are detected. It does so by making redo log based recovery ignore certain errors, such as missing data files or corrupted data pages. Any redo log for affected files or pages are skipped. You can facilitate dumping tables by getting the SELECT * FROM table_name statement to jump over corrupt indexes and pages.

Mode	Description
2	(SRV_FORCE_NO_BACKGROUND) stops the master thread from running, preventing a crash that occurs during a purge. No purge are performed, so the undo logs will keep growing.
3	(SRV_FORCE_NO_TRX_UNDO) does not roll back transactions after the crash recovery. Does not affect rollback of currently active transactions. Starting with MariaDB 10.2.7, will also prevent some undo-generating background tasks from running. These tasks could hit a lock wait due to the recovered incomplete transactions whose rollback is being prevented.
4	(SRV_FORCE_NO_IBUF_MERGE) does not calculate tables statistics and prevents insert buffer merges.
5	(SRV_FORCE_NO_UNDO_LOG_SCAN) treats incomplete transactions as committed, and does not look at the undo logs when starting.
6	(SRV_FORCE_NO_LOG_REDO) does not perform redo log roll-forward as part of recovery. Running queries that require indexes are likely to fail with this mode active. However, if a table dump still causes a crash, you can try using a SELECT * FROM tab ORDER BY primary_key DESC to dump all the data portion after the corrupted part.

Note also that XtraDB (<= MariaDB 10.2.6) by default will crash the server when it detects corrupted data in a single-table tablespace. This behaviour can be changed - see the innodb_corrupt_table_action system variable.

Fixing Things

Try to set innodb_force_recovery to 1 and start mariadb. If that fails, try a value of “2”. If a value of 2 works, then there is a chance the only corruption you have experienced is within the innodb “undo logs”. If that gets mariadb started, you should be able to dump your database with mariadb-dump. You can verify any other issues with any tables by running mariadb-check –all-databases.

If you were able to successfully dump your databases, or had previously known good backups, drop your database(s) from the mariadb command line like “DROP DATABASE yourdatabase”. Stop mariadb. Go to /var/lib/mysql (or wherever your mysql data directory is located) and “rm -i ib*”. Start mariadb, create the database(s) you dropped (“CREATE DATABASE yourdatabase”), and then import your most recent dumps: “mysql < mydatabasedump.sql”

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Enterprise Server InnoDB Operations

See Configure the InnoDB Buffer Pool.

A guide to configuring the size and instances of the InnoDB Buffer Pool to optimize memory usage and cache performance.

See Configure the InnoDB I/O Threads.

Instructions on tuning the number of InnoDB read and write I/O threads to match your system’s disk I/O capabilities.

See Configure the InnoDB Purge Threads.

Learn how to adjust the number of background purge threads to efficiently manage undo logs and prevent history list growth.

See Configure the InnoDB Redo Log.

A guide to configuring the size and number of InnoDB redo log files in MariaDB Enterprise Server to balance write performance and crash recovery time.

See Configure the InnoDB Undo Log.

Learn how to manage InnoDB undo logs in MariaDB Enterprise Server, including moving them to separate tablespaces and enabling truncation.

See Schema Changes.

An overview of supported online schema change operations in InnoDB, detailing which DDL statements can be performed without locking the table.

See InnoDB - Unmaintained.

This section provides information about unmaintained or deprecated features related to InnoDB in MariaDB Server. It is advisable to review this content for compatibility and migration planning.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Configure the InnoDB Buffer Pool

Overview

In MariaDB Enterprise Server, the InnoDB storage engine uses the Buffer Pool as an in-memory cache. The Buffer Pool caches pages that were recently accessed. If a lot of pages are being accessed sequentially, the Buffer Pool also preemptively caches nearby pages. Pages are evicted using a least recently used (LRU) algorithm.

The contents of the Buffer Pool can be reloaded at startup, so that InnoDB does not have to function with a “cold” cache after a restart. To support this, the page numbers of all pages in the Buffer Pool can be dumped at shutdown. During startup, the page numbers are read from the dump, and InnoDB uses the page numbers to reload each page from its corresponding data file.

The size of each page in the Buffer Pool depends on the value of the innodb_page_size system variable.

Starting with ES 10.5 and CS 10.5, the Buffer Pool always has a single instance.

For additional information, see “InnoDB Buffer Pool”.

This page describes how to configure the InnoDB Buffer Pool.

Configure the InnoDB Buffer Pool Size

The size of the InnoDB Buffer Pool can be configured by setting the innodb_buffer_pool_size system variable. On ES nodes that exclusively use the InnoDB storage engine, the InnoDB Buffer Pool should usually be between 50%-75% of the memory available.

Available Memory	Recommended InnoDB Buffer Pool Size
4 GB	2 GB
8 GB	4-8 GB
16 GB	8-12 GB
32 GB	16-24 GB
64 GB	32-56 GB
128 GB	64-96 GB
256 GB	128-192 GB

The method to configure the Buffer Pool size depends on whether a server restart are performed:

Product Versions	Server Restart?	Method
Any ES Any CS	No	Configure size with SET GLOBAL.
Any ES Any CS	No	Configure size in configuration file

Configure the InnoDB Buffer Pool Size with SET GLOBAL

The size of the InnoDB buffer pool can be changed dynamically by setting the innodb_buffer_pool_size system variable using the SET GLOBAL statement. The SET GLOBAL statement requires the SUPER privilege.

To ensure that the change survives server restarts, the innodb_buffer_pool_size system variable should also be set in a configuration file.

To configure the InnoDB Buffer Pool with the SET GLOBAL statement, use the following procedure:

1. Connect to the server using MariaDB Client as the `root@localhost` user account or another user account with the SUPER privilege:

```
$ mariadb --user=root
```

2. Set the innodb_buffer_pool_size system variable to the new size using the SET GLOBAL statement.

For example, to set the size to 2 GB:

```
SET GLOBAL innodb_buffer_pool_size=(2 * 1024 * 1024 * 1024);
```

3. Confirm that the resize operation has been completed by querying the Innodb_buffer_pool_resize_status status variable using the SHOW GLOBAL STATUS statement:

```
SHOW GLOBAL STATUS
   LIKE 'Innodb_buffer_pool_resize_status';
```

```
+----------------------------------+----------------------------------------------------+
| Variable_name                    | Value                                              |
+----------------------------------+----------------------------------------------------+
| Innodb_buffer_pool_resize_status | Completed resizing buffer pool at 200904 17:49:48. |
+----------------------------------+----------------------------------------------------+
```

4. Choose a configuration file for custom changes to system variables and options.
   It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the z- prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

5. Set the innodb_buffer_pool_size system variable in the configuration file.
   It needs to be set in a group that are read by MariaDB Server, such as [mariadb] or [server].
   When set in a configuration file, the value supports units, such as "M", "G", etc.

For example, to set the size to 2 GB:

```
[mariadb]
...
innodb_buffer_pool_size=2G
```

Configure the InnoDB Buffer Pool Size in a Configuration File

The size of the InnoDB Buffer Pool can be changed by setting the innodb_buffer_pool_size system variable in a configuration file.

To configure the InnoDB Buffer Pool in a configuration file, use the following procedure:

1. Choose a configuration file for custom changes to system variables and options.
   It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.
   Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the z- prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

2. Set the innodb_buffer_pool_size system variable in the configuration file.

It needs to be set in a group that are read by MariaDB Server, such as [mariadb] or [server].

When set in a configuration file, the value supports units, such as "M", "G", etc.

For example, to set the size to 2 GB:

```
[mariadb]
...
innodb_buffer_pool_size=2G
```

3. Restart the server:

```
$ sudo systemctl restart mariadb
```

The server can use the configuration change without a restart if you use SET GLOBAL.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Configure the InnoDB I/O Threads

Overview

The InnoDB Read I/O Threads handle completion of read I/O requests, and the InnoDB Write I/O Threads handle completion of write I/O requests.

Starting with MariaDB Enterprise Server 10.5 and MariaDB Community Server 10.5, the InnoDB I/O Threads were replaced by the asynchronous I/O functionality in the InnoDB Background Thread Pool.

For additional information, see "InnoDB I/O Threads".

This page describes how to configure the InnoDB I/O Threads.

Configure the Number of InnoDB I/O Threads

Starting with ES 10.5 and CS 10.5, the InnoDB I/O Threads have been replaced by the asynchronous I/O functionality in the InnoDB Background Thread Pool. In these versions, the innodb_read_io_threads and innodb_write_io_threads system variables have been repurposed. The value of each system variable is multiplied

by 256 to determine the maximum number of concurrent asynchronous I/O requests that can be completed by the Background Thread Pool.

For example, if `innodb_read_io_threads=2 and innodb_write_io_threads=4` are set, InnoDB are restricted to a maximum of 512 concurrent asynchronous read I/O requests and 1024 concurrent asynchronous write I/O requests.

Interaction with Asynchronous I/O

When asynchronous I/O is enabled, the InnoDB I/O Threads do not receive the initial I/O request from query threads. Instead, the query threads submit asynchronous I/O requests directly to the operating system, and after the operating system performs the operation, the InnoDB I/O Threads handle completion of the request.

Asynchronous I/O is enabled by the innodb_use_native_aio system variable, which is enabled by default.

Affected I/O Operations

The innodb_read_io_threads system variable affects completion of the following types of reads:

- Linear read-ahead (configured by innodb_read_ahead_threshold)
- Random read-ahead (configured by innodb_random_read_ahead)

The innodb_write_io_threads system variable affects completion of the following types of writes:

- Page flushing due to adaptive flushing (configured by innodb_adaptive_flushing and innodb_adaptive_flushing_lwm)
- Page flushing due to buffer pool capacity (configured by [innodb_max_dirty_pages_pct] and [innodb-system-variables/#innodb_max_dirty_pages_pct_lwm|innodb_max_dirty_pages_pct_lwm)
- Page flushing due to LRU page evictions (configured by innodb_lru_flush_size and innodb_lru_scan_depth)

Configuration Procedure

The method to configure the number of I/O threads depends on the server version and whether a server restart are performed:

Product Versions	Server Restart?	Method
ES 10.5 and Later	No	Configure maximum number of asynchronous I/O requests with SET GLOBAL
Any ES Any CS	Yes.	Configure number of I/O threads in configuration file

Configure InnoDB's Maximum Number of Asynchronous I/O Requests with SET GLOBAL (ES 10.5) and Later

Starting with MariaDB Enterprise Server 10.5, InnoDB's maximum number of asynchronous I/O requests can be changed dynamically by setting the innodb_read_io_threadsinnodb_write_io_threads system variables using the SET GLOBAL statement. The SET GLOBAL statement requires the SUPER privilege.

The value of each system variable is multiplied by 256 to determine the maximum number of asynchronous I/O requests that can be performed by the Background Thread Pool. For example, if you want to allow a maximum of 1024 concurrent asynchronous write I/O requests, the innodb_write_io_threads system variable should be set to 4 (since 1024/256=4).

To ensure that the change survives server restarts, the innodb_read_io_threads innodb_write_io_threads system variables should also be set in a configuration file.

To configure InnoDB's maximum number of asynchronous I/O requests with the SET GLOBAL statement, use the following procedure:

1. Connect to the server using MariaDB Client as the `root@localhost` user account or another user account with the SUPER privilege:

```
$ mariadb --user=root
```

2. Set the innodb_read_io_threads and innodb_write_io_threads system variables to the new values using the SET GLOBAL statement.

For example:

```
SET GLOBAL innodb_read_io_threads=8;
SET GLOBAL innodb_write_io_threads=8;
```

3. Choose a configuration file for custom changes to system variables and options.

It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the z- prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

4. Set the innodb_read_io_threads and innodb_write_io_threads system variables in the configuration file.

It needs to be set in a group that are read by MariaDB Server, such as [mariadb] or [server].

For example:

```
[mariadb]
...
innodb_read_io_threads=8
innodb_write_io_threads=8
```

Configure the Number of InnoDB I/O Threads in a Configuration File

The number of I/O threads is configured by the innodb_read_io_threads and innodb_write_io_threads system variables.

To configure the number of InnoDB I/O Threads in a configuration file, use the following procedure:

1. Choose a configuration file for custom changes to system variables and options.

It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the z- prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

2. Set the innodb_read_io_threads and innodb_write_io_threads system variables in the configuration file.

It needs to be set in a group that are read by MariaDB Server, such as `[mariadb]` or `[server]`.

For example:

```
[mariadb]
...
innodb_read_io_threads=8
innodb_write_io_threads=8
```

3. Restart the server:

```
$ sudo systemctl restart mariadb
```

Starting with MariaDB Enterprise Server 10.5, the server can use the configuration change without a restart if you use SET GLOBAL.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Configure the InnoDB Purge Threads

Overview

In MariaDB Enterprise Server, the InnoDB storage engine uses Purge Threads to perform garbage collection in the background. The Purge Threads are related to multi-version concurrency control (MVCC).

The Purge Threads perform garbage collection of various items:

The Purge Threads perform garbage collection of the InnoDB Undo Log. When a row is updated in the clustered index, InnoDB updates the values in the clustered index, and the old row version is added to the Undo Log. The Purge Threads scan the Undo Log for row versions that are not needed by open transactions and permanently delete them.

The Purge Threads perform garbage collection of index records. When an indexed column is updated, InnoDB creates a new index record for the updated value in each affected index, and the old index records are delete-marked. When the primary key column is updated, InnoDB creates a new index record for the updated value in every index, and each old index record is delete-marked. The Purge Threads scan for delete-marked index records and permanently delete them.

The Purge Threads perform garbage collection of freed overflow pages. BLOB, CHAR, TEXT, VARCHAR, VARBINARY, and related types are sometimes stored on overflow pages. When the value on the overflow page is deleted or updated, the overflow page is no longer needed. The Purge Threads delete these freed overflow pages.

For additional information, see “InnoDB Purge Threads”.

This page describes how to configure the InnoDB Purge Threads.

Configure the Number of InnoDB Purge Threads

The number of the InnoDB Purge Threads is configurable. If your server deletes or updates rows at a very high frequency, then you may need to increase the number of purge threads.

The method to configure the number of Purge Threads depends on the server version and whether a server restart are performed:

Product Versions	Server Restart?	Method
ES 10.5 and Later	No	Configure maximum number of asynchronous I/O requests with SET GLOBAL
Any ES Any CS	Yes.	Configure number of I/O threads in configuration file

Configure the Number of InnoDB Purge Threads with SET GLOBAL

> **Note:**
>
> This feature is available from MariaDB Enterprise Server 10.5.

The number of InnoDB purge threads can be changed dynamically by setting the innodb_purge_threads system variable using the SET GLOBAL statement. The SET GLOBAL statement requires the SUPER privilege.

To ensure that the change survives server restarts, the innodb_purge_threads system variable should also be set in a configuration file.

To configure the number of InnoDB Purge threads with the SET GLOBAL statement, use the following procedure:

1. Connect to the server using MariaDB Client as the `root@localhost` user account or another user account with the SUPER privilege:

```
$ mariadb --user=root
```

2. Set the innodb_purge_threads system variable to the new size using the SET GLOBAL statement.

For example:

```
SET GLOBAL innodb_purge_threads=8;
```

3. Choose a configuration file for custom changes to system variables and options.
   It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the `z-` prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

4. Set the innodb_purge_threads system variable in the configuration file.
   It needs to be set in a group that are read by MariaDB Server, such as [mariadb] or [server].

For example:

```
[mariadb]
...
innodb_purge_threads=8
```

Configure the Number of InnoDB Purge Threads in a Configuration File

The number of InnoDB Purge Threads can be configured by setting the innodb_purge_threads system variable in a configuration file.

To configure the number of innodb_purge_threads in a configuration file, use the following procedure:

1. Choose a configuration file for custom changes to system variables and options.
   It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

2. Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the `z-` prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

2. Set the innodb_purge_threads system variable in the configuration file.
   It needs to be set in a group that are read by MariaDB Server, such as `[mariadb]` or `[server]`.

For example:

```
[mariadb]
...
innodb_purge_threads=8
```

3. Restart the server:

```
$ sudo systemctl restart mariadb
```

The server can use the configuration change without a restart if you use SET GLOBAL.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Configure the InnoDB Redo Log

Overview

In MariaDB Enterprise Server, the InnoDB storage engine uses a Redo Log. The Redo Log is a transaction log that InnoDB uses to write data to disk in a crash-safe manner.

Redo Log records are identified using the Log Sequence Number (LSN). The Redo Log is a circular log file that is a constant size. Old Redo Log records are frequently overwritten by new Redo Log records. InnoDB regularly performs checkpoints. During a checkpoint, InnoDB flushes Redo Log records to the InnoDB tablespace files.

When the server crashes, InnoDB performs crash recovery during server startup using the Redo Log. During crash recovery, InnoDB finds the last checkpoint in the Redo Log and flushes the Redo Log records since the last checkpoint to the InnoDB tablespace files.

For additional information, see "InnoDB Redo Log".

This page describes how to configure the InnoDB Redo Log.

Configure the InnoDB Redo Log Size

> **Warning:**
>
> **Do not delete the `ib_logfile` files before startup to resize the redo log.** Starting with **MariaDB 10.8** — and **MariaDB Enterprise Server 11.4**, the first Enterprise Server with this behavior — the server fails to start if its existing redo log files are missing. The server must always be started with its existing redo log files. To resize the redo log, start the server normally, then use `SET GLOBAL innodb_log_file_size=…` (see below). Once the change is applied in the running instance, update the configuration file so the new size persists across restarts. This is part of the InnoDB redo log redesign tracked under MDEV-27199. Manual deletion of `ib_logfile` files was discouraged as early as MySQL 5.6.11 and became a hard startup failure in MariaDB 10.8 / Enterprise Server 11.4.

The size of the InnoDB Redo Log is configurable. If your server writes data at a very high frequency, then you may need to increase the redo log size, so that InnoDB does not have to perform checkpoints as frequently.

For the maximum capacity in the Redo Log, the Redo Log size should be the same as the innodb_buffer_pool_size, which is configured by the innodb_buffer_pool_size system variable.

The method to configure the Redo Log size depends on the server version and whether a server restart are performed:

Product Versions	Server Restart?	Method
ES 10.5 and Later	No	Configure size with SET GLOBAL
ES 10.5 and Later CS 10.5 and Later	Yes	Configure size in configuration file

Configure the InnoDB Redo Log Size with SET GLOBAL (ES 10.5 and Later)

Starting in MariaDB Enterprise Server 10.5, the size of the InnoDB Redo Log can be changed dynamically by setting the innodb_log_file_size system variable using the SET GLOBAL statement. The SET GLOBAL statement requires the SUPER privilege.

The resize operation is performed asynchronously in the background. If the server is restarted before the operation completes, the request may be ignored. To ensure that the change survives server restarts, the innodb_log_file_size system variable should also be set in a configuration file.

To configure the InnoDB Redo Log with the SET GLOBAL statement, use the following procedure:

1. Connect to the server using MariaDB Client as the `root@localhost` user account or another user account with the SUPER privilege:

```
$ mariadb --user=root
```

2. Set the innodb_log_file_size system variable to the new size using the SET GLOBAL statement.

For example, to set the size to 512 MB:

```
SET GLOBAL innodb_log_file_size=(512 * 1024 * 1024);
```

And to set the size to 2 GB:

```
SET GLOBAL innodb_log_file_size=(2 * 1024 * 1024 * 1024);
```

3. The resize operation is performed asynchronously in the background. Confirm that the resize operation is complete by querying the innodb_log_file_size system variable using the SHOW GLOBAL VARIABLES statement. The resize operation is complete when the output shows the new size as the value of the system variable.

Execute the following statement until it shows the new size:

```
SHOW GLOBAL VARIABLES
   LIKE 'innodb_log_file_size';
```

4. Choose a configuration file for custom changes to system variables and options.
   It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the `z-` prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

5. Set the innodb_log_file_size system variable in the configuration file.

It needs to be set in a group that are read by MariaDB Server, such as [mariadb] or [server].

When set in a configuration file, the value supports units, such as “M”, “G”, etc.

For example, to set the size to 512 MB:

```
[mariadb]
...
innodb_log_file_size=512M
```

And to set the size to 2 GB:

```
[mariadb]
...
innodb_log_file_size=2G
```

Configure the InnoDB Redo Log Size in a Configuration File (ES 10.5) and Later

Starting in MariaDB Enterprise Server 10.5, the size of the InnoDB Redo Log can be changed by setting the innodb_log_file_size system variable in a configuration file.

To configure the InnoDB Redo Log in a configuration file, use the following procedure:

1. Choose a configuration file for custom changes to system variables and options.

It is not recommended to make custom changes to Enterprise Server’s default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the `z-` prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

2. Set the innodb_log_file_size system variable in the configuration file.
   It needs to be set in a group that are read by MariaDB Server, such as [mariadb] or [server].
   When set in a configuration file, the value supports units, such as “M”, “G”, etc.

For example, to set the size to 512 MB:

```
[mariadb]
...
innodb_log_file_size=512M
```

And to set the size to 2 GB:

```
[mariadb]
...
innodb_log_file_size=2G
```

3. Starting in MariaDB Community Server 10.5, the server must be restarted for the configuration change to take effect:

```
$ sudo systemctl restart mariadb
```

4. Starting in MariaDB Enterprise Server 10.5, the server can use the configuration change without a restart if you use SET GLOBAL.

Emergency Recovery Across Major Versions

In MariaDB 11.x (and the corresponding MariaDB Enterprise Server releases), starting the server with data files from MariaDB 10.x is permitted **only** when innodb_force_recovery is set to `6`. This path is intended exclusively as a last-resort mechanism for extracting a logical dump from a corrupted or otherwise incompatible database; it is **not** a substitute for the normal upgrade path.

> **Important:**
>
> `innodb_force_recovery=6` places the database in a highly restricted state intended only for data salvage. Do not use it as a standard operational approach or a shortcut for the normal upgrade path.

When this emergency path is in effect:

- The database accepts no new writes or modifications.
- Use mariadb-dump (or another logical export tool) to extract the data, then re-import it into a fresh, supported instance.

For the standard upgrade procedure between major versions, see Upgrading Between Major MariaDB Versions.

This capability was added under MDEV-39303, available in MariaDB 11.4.11, MariaDB 11.8.7, and later releases.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Configure the InnoDB Undo Log

Overview

The InnoDB undo log is a transaction log used by InnoDB to keep track of multiple row versions for multi-version concurrency control (MVCC). When a row's value changes, InnoDB stores old versions of the row in the Undo Log.

When transactions are committed and the old row versions are no longer necessary, the InnoDB Purge Threads asynchronously delete old row versions from the Undo Log in the background.

When a transaction is rolled back, InnoDB uses the Undo Log to rollback the transaction's changes.

For additional information, see "InnoDB Undo Log".

This page describes how to configure the InnoDB Undo Log.

Configure InnoDB Undo Log Tablespaces

By default, the InnoDB undo log is located in the InnoDB system tablespace, which is defined by the innodb_data_file_path system variable. However, it can be helpful to configure separate undo log tablespaces to spread out I/O load between different files or storage devices.

InnoDB can be configured to use separate undo log tablespaces by setting the innodb_undo_tablespaces system variable. The separate undo log tablespaces will have file names of the format undoN, where N is an integer.

When you configure separate undo log tablespaces, you can also configure the separate undo log tablespaces to go to a specific directory by setting the innodb_undo_directory system variable. This is most helpful if you want to put the undo log tablespaces on a separate storage device.

Separate InnoDB undo log tablespaces must be configured prior to the initialization of the server's InnoDB data directory. If you try to configure separate InnoDB undo log tablespaces when the InnoDB data directory has already been initializes, you will see errors in the error log during startup similar to the following:

```
[ERROR] InnoDB: Expected to open innodb_undo_tablespaces=8 but was able to find only 0
[ERROR] InnoDB: Plugin initialization aborted with error Generic error
```

To safely configure separate InnoDB undo log tablespaces:

1. If you have preexisting data, backup your data with MariaDB Dump.
2. Ensure that the server is stopped:

```
$ sudo systemctl stop mariadb
```

3. Choose a configuration file for custom changes to system variables and options.

It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the `z-` prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

4. Set the innodb_undo_tablespaces system variable in the configuration file.

It needs to be set in a group that are read by MariaDB Server, such as [mariadb] or [server].

For example, to set the number of tablespaces to 8:

```
[mariadb]
...
innodb_undo_tablespaces=8
```

5. If you want your InnoDB undo log tablespaces to be in a specific directory, then also set the innodb_undo_directory system variable in the configuration file:

For example, to set the directory to /innodb/undo:

```
[mariadb]
...
innodb_undo_directory=/innodb/undo
```

6. If you want your InnoDB undo log tablespaces to be in a specific directory, then also create the directory, and give it the proper permissions:

```
$ sudo mkdir -p /innodb/undo
$ sudo chown mysql:mysql /innodb/undo
```

7. Delete the current contents of the datadir and innodb_data_home_dir.

For example, if the default value of /var/lib/mysql is used for both:

```
$ sudo rm -fr /var/lib/mysql/*
```

8. Reinitialize the data directory using the MariaDB Install DB command.
9. Start the server:

```
$ sudo systemctl start mariadb
```

10. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

11. If your server had preexisting data, then reload the backup taken at the beginning of the procedure.
12. Confirm that the configuration changes were properly applied by checking the values of the system variables using the SHOW GLOBAL VARIABLES statement:

```
SHOW GLOBAL VARIABLES
   WHERE Variable_name IN (
      'innodb_undo_tablespaces',
      'innodb_undo_directory'
   );
```

```
+-------------------------+--------------+
| Variable_name           | Value        |
+-------------------------+--------------+
| innodb_undo_directory   | /innodb/undo |
| innodb_undo_tablespaces | 8            |
+-------------------------+--------------+
```

13. Consider also enabling undo log truncation to increase performance of the InnoDB Purge Threads.

Enable InnoDB Undo Log Truncation

If a server is configured to have 2 or more separate InnoDB undo log files, then InnoDB undo log truncation can be enabled by setting the innodb_undo_log_truncate system variable using the SET GLOBAL statement. The SET GLOBAL statement requires the SUPER privilege.

When InnoDB undo log truncation is enabled, the InnoDB purge threads can truncate an entire undo log at once, rather than individually freeing each rollback segment within the undo log.

An undo log is truncated when its size exceeds the innodb_max_undo_log_size system variable.

The frequency at which the InnoDB purge threads check for undo logs to truncate is configured by setting the innodb_purge_rseg_truncate_frequency system variable using the SET GLOBAL statement.

To ensure that the changes survive server restarts, the system variables should also be set in a configuration file.

To enable InnoDB undo log truncation:

1. Connect to the server using MariaDB Client as the `root@localhost` user account or another user account with the SUPER privilege:

```
$ mariadb --user=root
```

2. Set the innodb_undo_log_truncate system variable to ON using the SET GLOBAL statement.

For example:

```
SET GLOBAL innodb_undo_log_truncate=ON;
```

3. If you would like to change the size at which undo logs are truncated, then also set the innodb_max_undo_log_size system variable to the new size using the SET GLOBAL statement.

For example, to set the size to 2 GB:

```
SET GLOBAL innodb_max_undo_log_size=(2 * 1024 * 1024 * 1024);
```

4. If you would like the InnoDB purge threads to check the undo logs more frequently, then also set the innodb_purge_rseg_truncate_frequency system variable to a lower value using the SET GLOBAL statement.

For example, to configure the purge threads to check the undo logs for truncation every 64 iterations:

```
SET GLOBAL innodb_purge_rseg_truncate_frequency=64;
```

5. Choose a configuration file for custom changes to system variables and options.
   It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the `z-` prefix in the file name.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

6. Set the system variables in the configuration file.
   It needs to be set in a group that are read by MariaDB Server, such as [mariadb] or [server].
   When set in a configuration file, the innodb_max_undo_log_size value supports units, such as "M", "G", etc.

For example:

```
[mariadb]
...
innodb_undo_log_truncate=ON
innodb_max_undo_log_size=2G
innodb_purge_rseg_truncate_frequency=64
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Schema Changes

Overview

In MariaDB Enterprise Server, InnoDB supports many different schema change operations. Many of the operations can be performed online with concurrent DML using little or no locking.

DDL Statements

InnoDB schema changes are performed using the following DDL statements:

- ALTER TABLE
- CREATE INDEX
- CREATE SPATIAL INDEX
- CREATE UNIQUE INDEX
- DROP INDEX
- RENAME TABLE

About InnoDB Schema Changes and Online DDL

InnoDB schema changes and online DDL are performed with a wide range of statements:

Each operation supports a subset of the following algorithms: `INSTANT, NOCOPY, INPLACE, or COPY`.

By default, InnoDB will use the most efficient algorithm supported by an operation. This behavior can be changed by using the `ALGORITHM` clause with the ALTER TABLE statement or by changing the value of the alter_algorithm system variable.

Each operation supports a subset of the following locking strategies: `NONE, SHARED, or EXCLUSIVE`.

By default, InnoDB will use the most permissive locking strategy supported by an operation. This behavior can be changed by using the LOCK clause with the ALTER TABLE statement.

Feature	Detail	Resources
Operations support instant algorithm	Yes	InnoDB Schema Changes with the INSTANT Algorithm
Operations support no-copy algorithm	Yes	InnoDB Schema Changes with the NOCOPY Algorithm
Operations support in-place algorithm	Yes	InnoDB Schema Changes with the INPLACE Algorithm
Adding a column	Yes	
Dropping a column	Yes	
Reordering columns	Yes	
Changing a column to NULL	Yes	
Changing a column to NOT NULL	Yes	
Adding a new ENUM option	Yes	
Adding a new SET option	Yes	
Adding system versioning	Yes	
Removing system versioning	Yes	
Setting a column's DEFAULT	Yes	
Removing a column's DEFAULT	Yes	
Adding a primary key	Yes	
Dropping a primary key	Yes	
Adding an index	Yes	
Dropping an index	Yes	
Adding a foreign key	Yes	
Dropping a foreign key	Yes	
Setting the next AUTO_INCREMENT value	Yes	
Setting the row format	Yes	
Setting the block size for the Compressed row format	Yes	
Enabling page compression	Yes	

Feature	Detail	Resources
Setting the page compression level	Yes	
Disabling page compression	Yes	
Enabling data-at-rest encryption	Yes	
Setting the encryption key ID	Yes	
Disabling data-at-rest encryption	Yes	
Adding a constraint	Yes	
Dropping a constraint	Yes	
Rebuilding the table	Yes	
Renaming the table	Yes	

This page is: Copyright © 2026 MariaDB. All rights reserved.

InnoDB Schema Changes

MariaDB Enterprise Server InnoDB Schema Changes with the INPLACE Algorithm

Changes in enterprise databases and applications are inevitable. Schema changes often block other workloads, incur unique production situations that cannot be fully simulated for testing, and may have unpredictable execution times.

MariaDB Enterprise Server includes the In-place `ALTER` functionality for the InnoDB storage engine, such that:

When possible, schema change operations are performed `INPLACE`, minimizing impact on other workloads.

When the `alter_algorithm` system variable is set to `INPLACE`, schema change operations will not run unless they can be performed `INPLACE`, minimizing the risk of unpredictable behavior.

Additional information is available here.

MariaDB Enterprise Server InnoDB Schema Changes with the INSTANT Algorithm

Changes in enterprise databases and applications are inevitable. Schema changes often block other workloads, incur unique production situations that cannot be fully simulated for testing, and may have unpredictable execution times.

MariaDB Enterprise Server includes the Instant `ALTER` functionality for the InnoDB storage engine, such that:

When possible, schema change operations are performed `INSTANT`, minimizing impact on other workloads.

When the `alter_algorithm` system variable is set to `INSTANT`, schema change operations will not run unless they can be performed `INSTANT`, minimizing the risk of unpredictable behavior.

Additional information is available here.

MariaDB Enterprise Server InnoDB Schema Changes with the NOCOPY Algorithm

Changes in enterprise databases and applications are inevitable. Schema changes often block other workloads, incur unique production situations that cannot be fully simulated for testing, and may have unpredictable execution times.

MariaDB Enterprise Server includes the No-copy `ALTER` functionality for the InnoDB storage engine, such that:

When possible, schema change operations are performed `NOCOPY`, minimizing impact on other workloads.

When the alter_algorithm system variable is set to `NOCOPY`, schema change operations will not run unless they can be performed `NOCOPY`, minimizing the risk of unpredictable behavior.

Additional information is available here.

This page is: Copyright © 2026 MariaDB. All rights reserved.

InnoDB - Unmaintained

See About XtraDB.

Information about the XtraDB storage engine that was used in old MariaDB versions.

See Using InnoDB Instead of XtraDB.

Information about XtraDB, an storage engine used in old MariaDB versions.

This page is licensed: CC BY-SA / Gnu FDL

About XtraDB

Percona XtraDB was an enhanced version of the InnoDB storage engine used in MariaDB before MariaDB 10.2, designed to better scale on modern hardware, and it includes a variety of other features useful in high-performance environments.

It is fully backwards compatible, and it identifies itself to MariaDB as "`ENGINE=InnoDB`" (just like InnoDB), and so can be used as a drop-in replacement for standard InnoDB.

Percona XtraDB includes all of InnoDB's robust, reliable ACID-compliant design and advanced MVCC architecture, and builds on that solid foundation with more features, more tunability, more metrics, and more scalability. In particular, it is designed to scale better on many cores, to use memory more efficiently, and to be more convenient and useful. The new features are especially designed to alleviate some of InnoDB's limitations. We choose features and fixes based on customer requests and on our best judgment of real-world needs as a high-performance consulting company.

XtraDB was also available in MariaDB for Windows.

Percona XtraDB versions in MariaDB

MariaDB 10.1

- XtraDB from Percona Server 5.6.49-89.0 in MariaDB 10.1.46
- XtraDB from Percona Server 5.6.46-86.2 in MariaDB 10.1.44
- XtraDB from Percona Server 5.6.43-84.3 in MariaDB 10.1.39
- XtraDB from Percona Server 5.6.41-84.1 in MariaDB 10.1.36
- XtraDB from Percona Server 5.6.38-83.0[1] in MariaDB 10.1.31
- XtraDB from Percona Server 5.6.37-82.2[2]in MariaDB 10.1.27
- XtraDB from Percona Server 5.6.36-82.1 in MariaDB 10.1.26
- XtraDB from Percona Server 5.6.36-82.0 in MariaDB 10.1.24
- XtraDB from Percona Server 5.6.35-80.0 in MariaDB 10.1.22
- XtraDB from Percona Server 5.6.34-79.1 in MariaDB 10.1.20
- XtraDB from Percona Server 5.6.33-79.0[3] in MariaDB 10.1.19
- XtraDB from Percona Server 5.6.32-78.1 in MariaDB 10.1.18
- XtraDB from Percona Server 5.6.31-77.0 in MariaDB 10.1.17
- XtraDB from Percona Server 5.6.30-76.3 in MariaDB 10.1.15
- XtraDB from Percona Server 5.6.29-76.2 in MariaDB 10.1.14

- XtraDB from Percona Server 5.6.28-76.1 in MariaDB 10.1.12
- XtraDB from Percona Server 5.6.26-76.0 in MariaDB 10.1.10
- XtraDB from Percona Server 5.6.26-74.0 in MariaDB 10.1.8
- XtraDB from Percona Server 5.6.25-73.1 in MariaDB 10.1.7
- XtraDB from Percona Server 5.6.24-72.2 in MariaDB 10.1.6
- XtraDB from Percona Server 5.6.23-72.1 in MariaDB 10.1.5
- XtraDB from Percona Server 5.6.22-72.0 in MariaDB 10.1.4
- XtraDB from Percona Server 5.6.21-70.0 in MariaDB 10.1.2
- XtraDB from Percona Server 5.6.17-65.0 in MariaDB 10.1.1

MariaDB 10.0

- XtraDB from Percona Server 5.6.42-84.2 in MariaDB 10.0.38
- XtraDB from Percona Server 5.6.41-84.1 in MariaDB 10.0.37
- XtraDB from Percona Server 5.6.39-83.1 in MariaDB 10.0.35
- XtraDB from Percona Server 5.6.38-83.0[4]in MariaDB 10.0.34
- XtraDB from Percona Server 5.6.37-82.2[5]in MariaDB 10.0.33
- XtraDB from Percona Server 5.6.36-82.1 in MariaDB 10.0.32
- XtraDB from Percona Server 5.6.36-82.0 in MariaDB 10.0.31
- XtraDB from Percona Server 5.6.35-80.0 in MariaDB 10.0.30
- XtraDB from Percona Server 5.6.34-79.1 in MariaDB 10.0.29
- XtraDB from Percona Server 5.6.33-79.0[6] in MariaDB 10.0.28
- XtraDB from Percona Server 5.6.31-77.0 in MariaDB 10.0.27
- XtraDB from Percona Server 5.6.30-76.3 in MariaDB 10.0.26
- XtraDB from Percona Server 5.6.29-76.2 in MariaDB 10.0.25
- XtraDB from Percona Server 5.6.28-76.1 in MariaDB 10.0.24
- XtraDB from Percona Server 5.6.27-76.0 in MariaDB 10.0.23
- XtraDB from Percona Server 5.6.26-74.0 in MariaDB 10.0.22
- XtraDB from Percona Server 5.6.25-73.1 in MariaDB 10.0.21
- XtraDB from Percona Server 5.6.24-72.2 in MariaDB 10.0.20
- XtraDB from Percona Server 5.6.23-72.1 in MariaDB 10.0.18
- XtraDB from Percona Server 5.6.22-72.0 in MariaDB 10.0.17
- XtraDB from Percona Server 5.6.22-71.0 in MariaDB 10.0.16
- XtraDB from Percona Server 5.6.21-70.0 in MariaDB 10.0.15
- XtraDB from Percona Server 5.6.20-68.0 in MariaDB 10.0.14
- XtraDB from Percona Server 5.6.19-67.0 in MariaDB 10.0.13
- XtraDB from Percona Server 5.6.17-65.0 in MariaDB 10.0.11
- XtraDB from Percona Server 5.6.14-rel62.0 in MariaDB 10.0.7

MariaDB 5.5

- XtraDB from Percona Server 5.5.61-38.13 in MariaDB 5.5.62
- XtraDB from Percona Server 5.5.59-38.11 in MariaDB 5.5.60
- XtraDB from Percona Server 5.5.58-38.10 in MariaDB 5.5.59
- XtraDB from Percona Server 5.5.55-38.9 in MariaDB 5.5.58
- XtraDB from Percona Server 5.5.55-38.8 in MariaDB 5.5.57
- XtraDB from Percona Server 5.5.52-38.3 in MariaDB 5.5.53

- XtraDB from Percona Server 5.5.50-38.0 in MariaDB 5.5.51
- XtraDB from Percona Server 5.5.49-37.9 in MariaDB 5.5.50
- XtraDB from Percona Server 5.5.48-37.8 in MariaDB 5.5.49
- XtraDB from Percona Server 5.5.46-37.7 in MariaDB 5.5.48
- XtraDB from Percona Server 5.5.46-37.6 in MariaDB 5.5.47
- XtraDB from Percona Server 5.5.45-37.4 in MariaDB 5.5.46
- XtraDB from Percona Server 5.5.44-37.3 in MariaDB 5.5.45
- XtraDB from Percona Server 5.5.42-37.2 in MariaDB 5.5.44
- XtraDB from Percona Server 5.5.42-37.1 in MariaDB 5.5.43
- XtraDB from Percona Server 5.5.40-36.1 in MariaDB 5.5.40
- XtraDB from Percona Server 5.5.38-35.2 in MariaDB 5.5.39
- XtraDB from Percona Server 5.5.37-35.0 in MariaDB 5.5.38
- XtraDB from Percona Server 5.5.36-34.0 in MariaDB 5.5.37
- XtraDB from Percona Server 5.5.35-33.0 in MariaDB 5.5.35
- XtraDB from Percona Server 5.5.34-32.0 in MariaDB 5.5.34
- XtraDB from Percona Server 5.5.33-31.1 in MariaDB 5.5.33
- XtraDB from Percona Server-5.5.32-31.0 in MariaDB 5.5.32

MariaDB 5.2 and MariaDB 5.3

- MariaDB 5.2 and 5.3 include the latest XtraDB version from MariaDB 5.1 at the time they were released.

MariaDB 5.1

- version 5.1.59-13 in MariaDB 5.1.60
- version 5.1.54-12.5 in MariaDB 5.1.55
- version 5.1.52-11.6 in MariaDB 5.2.4 and 5.1.53
- version 5.1.49-12 in MariaDB 5.1.50
- version 5.1.47-11.2 in MariaDB 5.1.49
- version 1.0.6-10 in MariaDB 5.1.47
- version 1.0.6-9 in MariaDB 5.1.42,5.1.44, and 5.1.44b.
- version 1.0.4-8 in MariaDB 5.1.41 RC
- version 1.0.3-8 in MariaDB 5.1.39 Beta
- version 1.0.3-6 in MariaDB 5.1.38 Beta

Notes

1. ↑ Misidentifies itself as 5.6.36-83.0 in MariaDB 10.1.31
2. ↑ Misidentifies itself as 5.6.36-82.2 from MariaDB 10.1.27 to MariaDB 10.1.30
3. ↑ Misidentifies itself as 5.6.32-79.0 in MariaDB 10.1.19
4. ↑ Misidentifies itself as 5.6.36-83.0 in MariaDB 10.0.34
5. ↑ Misidentifies itself as 5.6.36-82.2 in MariaDB 10.0.33
6. ↑ Misidentifies itself as 5.6.32-79.0 in MariaDB 10.0.28

See Also

More information can be found in the Percona documentation.

This page is licensed: CC BY-SA / Gnu FDL

Using InnoDB Instead of XtraDB

XtraDB, previously the default InnoDB replacement in MariaDB, is no longer included in standard distributions. MariaDB now uses InnoDB by default.

The reasons you may want to use InnoDB instead of XtraDB in earlier versions of MariaDB are:

- You want to benchmark the difference between InnoDB/XtraDB
- You hit a bug in XtraDB
- You got a table space crash in XtraDB and recovery doesn't work. In some cases InnoDB may do a better job to recover data.

See Also

- Compiling with the InnoDB plugin from Oracle

This page is licensed: CC BY-SA / Gnu FDL

MEMORY

Contents of the MEMORY storage engine (previously known as HEAP) are stored in memory rather than on disk.

It is best-used for read-only caches of data from other tables, or for temporary work areas.

Since the data is stored in memory, it is highly vulnerable to power outages or hardware failure, and is unsuitable for permanent data storage. In fact, after a server restart, `MEMORY` tables are recreated (because the definition file is stored on disk), but they are empty. It is possible to re-populate them with a query using the `--init-file` server startup option.

Variable-length types like VARCHAR can be used in MEMORY tables. BLOB or TEXT columns are not supported for MEMORY tables.

Memory Usage

The maximum total size of MEMORY tables cannot exceed the max_heap_table_size system server variable. When a table is created this value applies to that table, and when the server is restarted this value applies to existing tables. Changing this value has no effect on existing tables. However, executing a `ALTER TABLE ... ENGINE=MEMORY` statement applies the current value of `max_heap_table_size` to the table. Also, it is possible to change the session value of `max_heap_table_size` before creating a table, to make sure that tables created by other sessions are not affected.

The `MAX_ROWS` table option provides a hint about the number of rows you plan to store in them. This is not a hard limit that cannot be exceeded, and does not allow to exceed `max_heap_table_size`. The storage engine uses max_heap_table_size and MAX_ROWS to calculate the maximum memory that could be allocated for the table.

Memory allocated to a MEMORY table is freed by running DROP TABLE or TRUNCATE TABLE, or rebuilding with ALTER TABLE tbl_name ENGINE = MEMORY. When rows are deleted, space is not automatically freed.

Replication

The data in MEMORY tables are lost when a server restarts. To reflect this result in replication, the first time a server uses a MEMORY table after a restart, it writes a TRUNCATE statement for that table with a "generated by server" comment to the binary log, so its replicas are also emptied.

> **Warning:**
>
> In non-circular replication, restarting a replica will diverge from its primaries in both MEMORY table data and GTID. The contents and replication must be resynchronized to fix this issue. Alternatively, if consistency is not required (e.g., for cache data), then the replica should have a different Domain ID than its primaries to satisfy GTID strict mode.
>
> If the MEMORY tables are not required for replication at all, then excluding them from replication avoids this problem entirely.

Index Type

The MEMORY storage engine permits indexes to be either B-tree or Hash. Hash is the default type for MEMORY. See Storage Engine index types for more on their characteristics.

A MEMORY table can have up to 64 indexes, 16 columns for each index and a maximum key length of 3072 bytes.

Example

The following example shows how to create a `MEMORY` table with a given maximum size, as described above.

```
SET max_heap_table_size = 1024*516;

CREATE TABLE t (a VARCHAR(10), b INT) ENGINE = MEMORY;

SET max_heap_table_size = @@max_heap_table_size;
```

See Also

- Performance of MEMORY tables

This page is licensed: CC BY-SA / Gnu FDL

MERGE

Description

The MERGE storage engine, also known as the MRG_MyISAM engine, is a collection of identical MyISAM tables that can be used as one. “Identical” means that all tables have identical column and index information. You cannot merge MyISAM tables in which the columns are listed in a different order, do not have exactly the same columns, or have the indexes in different order. However, any or all of the MyISAM tables can be compressed with myisampack. Columns names and indexes names can be different, as long as data types and `NULL` / `NOT NULL` clauses are the same. Differences in table options such as `AVG_ROW_LENGTH`, `MAX_ROWS`, or `PACK_KEYS` do not matter.

Each index in a `MERGE` table must match an index in underlying `MyISAM` tables, but the opposite is not true. Also, a `MERGE` table cannot have a `PRIMARY KEY` or `UNIQUE` indexes, because it cannot enforce uniqueness over all underlying tables.

The following options are meaningful for `MERGE` tables:

- `UNION`. This option specifies the list of the underlying `MyISAM` tables. The list is enclosed between parentheses and separated with commas.
- `INSERT_METHOD`. This options specifies whether, and how, `INSERT`s are allowed for the table. Allowed values are: `NO` (`INSERT`s are not allowed), `FIRST` (new rows are written into the first table specified in the `UNION` list), `LAST` (new rows are written into the last table specified in the `UNION` list). The default value is `NO`.

If you define a `MERGE` table with a definition which is different from the underlying `MyISAM` tables, or one of the underlying tables is not `MyISAM`, the `CREATE TABLE` statement will not return any error. But any statement which involves the table will produce an error like the following:

```
ERROR 1168 (HY000): Unable to open underlying table which is differently defined
  or of non-MyISAM type or doesn't exist
```

A CHECK TABLE will show more information about the problem.

The error is also produced if the table is properly define, but an underlying table's definition changes at some point in time.

If you try to insert a new row into a `MERGE` table with `INSERT_METHOD=NO`, you will get an error like the following:

```
ERROR 1036 (HY000): Table 'tbl_name' is read only
```

It is possible to build a `MERGE` table on `MyISAM` tables which have one or more virtual columns. `MERGE` itself does not support virtual columns, thus such columns are seen as regular columns. The data types and sizes will still need to be identical, and they cannot be `NOT NULL`.

Examples

```
CREATE TABLE t1 (
    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    message CHAR(20)) ENGINE=MyISAM;

CREATE TABLE t2 (
    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    message CHAR(20)) ENGINE=MyISAM;

INSERT INTO t1 (message) VALUES ('Testing'),('table'),('t1');

INSERT INTO t2 (message) VALUES ('Testing'),('table'),('t2');

CREATE TABLE total (
    a INT NOT NULL AUTO_INCREMENT,
    message CHAR(20), INDEX(a))
    ENGINE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;

SELECT * FROM total;
+---+---------+
| a | message |
+---+---------+
| 1 | Testing |
| 2 | table   |
| 3 | t1      |
| 1 | Testing |
| 2 | table   |
| 3 | t2      |
+---+---------+
```

In the following example, we'll create three `MyISAM` tables, and then a `MERGE` table on them. However, one of them uses a different data type for the column b, so a `SELECT` will produce an error:

```
CREATE TABLE t1 (
  a INT,
  b INT
) ENGINE = MyISAM;

CREATE TABLE t2 (
  a INT,
  b INT
) ENGINE = MyISAM;

CREATE TABLE t3 (
  a INT,
  b TINYINT
) ENGINE = MyISAM;

CREATE TABLE t_mrg (
  a INT,
  b INT
) ENGINE = MERGE,UNION=(t1,t2,t3);

SELECT * FROM t_mrg;
ERROR 1168 (HY000): Unable to open underlying table which is differently defined
 or of non-MyISAM type or doesn't exist
```

To find out what's wrong, we'll use a `CHECK TABLE`:

```
CHECK TABLE t_mrg\G
*************************** 1. row ***************************
   Table: test.t_mrg
      Op: check
Msg_type: Error
Msg_text: Table 'test.t3' is differently defined or of non-MyISAM type or doesn't exist
*************************** 2. row ***************************
   Table: test.t_mrg
      Op: check
Msg_type: Error
Msg_text: Unable to open underlying table which is differently defined or of non-MyISAM type or doesn't
exist
*************************** 3. row ***************************
   Table: test.t_mrg
      Op: check
Msg_type: error
Msg_text: Corrupt
```

Now, we know that the problem is in `t3`'s definition.

This page is licensed: GPLv2, originally from fill_help_tables.sql

Mroonga

> **Note:**
>
> Mroonga is a community contribution to MariaDB Server, developed and maintained by the Mroonga project.

See Mroonga Overview.

Learn about the Mroonga storage engine, which provides fast CJK-ready full-text searching using column-based storage and supports Groonga's features.

See About Mroonga.

This page provides version history, installation instructions, and limitations for the Mroonga storage engine, highlighting its full-text search capabilities.

See Mroonga Status Variables.

A list of status variables specific to Mroonga, used to monitor internal operations like fast line counts and order-by-limit optimizations.

See Mroonga System Variables.

System variables for the Mroonga storage engine, and how to set them.

See Mroonga User-Defined Functions.

Extend Mroonga's functionality in MariaDB Server with user-defined functions. Learn how to create custom functions to enhance full-text search and data processing capabilities.

This page is licensed: CC BY-SA / Gnu FDL

Mroonga Overview

Once Mroonga has been installed (see About Mroonga), its basic usage is similar to that of a regular fulltext index.

For example:

```
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy)) ENGINE=Mroonga;

INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
    ('There was a wicked witch'), ('Who ate everybody up');

SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('wicked');
+--------------------------+
| copy                     |
+--------------------------+
| There was a wicked witch |
+--------------------------+
```

Score

Mroonga can also order by weighting. For example, first add another record:

```
INSERT INTO ft_mroonga(copy) VALUES ('She met a wicked, wicked witch');
```

Records can be returned by weighting, for example, the newly added record has two occurrences of the word 'wicked' and a higher weighting:

```
SELECT *, MATCH(copy) AGAINST('wicked') AS score FROM ft_mroonga
   WHERE MATCH(copy) AGAINST('wicked') ORDER BY score DESC;
+--------------------------------+--------+
| copy                           | score  |
+--------------------------------+--------+
| She met a wicked, wicked witch | 299594 |
| There was a wicked witch       | 149797 |
+--------------------------------+--------+
```

Parser

Mroonga permits you to set a different parser for searching by specifying the parser in the `CREATE TABLE` statement as a comment or, in older versions, changing the value of the mroonga_default_parser system variable.

For example:

```
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenDelimitNull"')
  ENGINE=Mroonga;,
```

or

```
SET GLOBAL mroonga_default_parser = 'TokenBigramSplitSymbol';
```

The following parser settings are available:

Setting	Description
off	No tokenizing is performed.
TokenBigram	Default value. Continuous alphabetical characters, numbers or symbols are treated as a token.
TokenBigramIgnoreBlank	Same as TokenBigram except that white spaces are ignored.
TokenBigramIgnoreBlankSplitSymbol	Same as TokenBigramSplitSymbol. except that white spaces are ignore.
TokenBigramIgnoreBlankSplitSymbolAlpha	Same as TokenBigramSplitSymbolAlpha except that white spaces are ignored.
TokenBigramIgnoreBlankSplitSymbolAlphaDigit	Same as TokenBigramSplitSymbolAlphaDigit except that white spaces are ignored.
TokenBigramSplitSymbol	Same as TokenBigram except that continuous symbols are not treated as a token, but tokenised in bigram.
TokenBigramSplitSymbolAlpha	Same as TokenBigram except that continuous alphabetical characters are not treated as a token, but tokenised in bigram.
TokenDelimit	Tokenises by splitting on white spaces.
TokenDelimitNull	Tokenises by splitting on null characters (\0).
TokenMecab	Tokenise using MeCab. Required Groonga to be built with MeCab support.
TokenTrigram	Tokenises in trigrams but continuous alphabetical characters, numbers or symbols are treated as a token.
TokenUnigram	Tokenises in unigrams but continuous alphabetical characters, numbers or symbols are treated as a token.

Examples

TokenBigram vs TokenBigramSplitSymbol

`TokenBigram` failing to match partial symbols which `TokenBigramSplitSymbol` matches, since `TokenBigramSplitSymbol` does not treat continuous symbols as a token.

```
DROP TABLE ft_mroonga;
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenBigram"')
  ENGINE=Mroonga;
INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
  ('There was a wicked witch'),
  ('Who ate everybody up'),
  ('She met a wicked, wicked witch'),
  ('A really wicked, wicked witch!!?!');
SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('!?');
Empty set (0.00 sec)

DROP TABLE ft_mroonga;
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenBigramSplitSymbol"')
  ENGINE=Mroonga;
INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
  ('There was a wicked witch'),
  ('Who ate everybody up'),
  ('She met a wicked, wicked witch'),
  ('A really wicked, wicked witch!!?!');
SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('!?');
+----------------------------------+
| copy                             |
+----------------------------------+
| A really wicked, wicked witch!!?! |
+----------------------------------+
```

TokenBigram vs TokenBigramSplitSymbolAlpha

```
DROP TABLE ft_mroonga;
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenBigram"')
  ENGINE=Mroonga;
INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
  ('There was a wicked witch'),
  ('Who ate everybody up'),
  ('She met a wicked, wicked witch'),
  ('A really wicked, wicked witch!!?!');
SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('ick');
Empty set (0.00 sec)

DROP TABLE ft_mroonga;
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenBigramSplitSymbolAlpha"')
  ENGINE=Mroonga;
INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
  ('There was a wicked witch'),
  ('Who ate everybody up'),
  ('She met a wicked, wicked witch'),
  ('A really wicked, wicked witch!!?!');
SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('ick');
+----------------------------------+
| copy                             |
+----------------------------------+
| There was a wicked witch         |
| She met a wicked, wicked witch   |
| A really wicked, wicked witch!!?! |
+----------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

About Mroonga

Mroonga is a full-text search storage engine based on Groonga, which is an open-source CJK-ready (Chinese, Japanese, and Korean) fulltext search engine using column base. See groonga.org for more.

With Mroonga, you can have a CJK-ready full text search feature, and it is faster than the MyISAM and InnoDB full text search for both updating and searching.

Mroonga also supports Groonga's fast geolocation search by using MariaDB's geolocation SQL syntax.

Mroonga only supports Linux x86_64 (Intel64/AMD64).

How to Install

Enable Mroonga with the following statement:

```
INSTALL SONAME 'ha_mroonga';
```

On Debian and Ubuntu mroonga engine are installed with

```
sudo apt-get install mariadb-plugin-mroonga
```

See Plugin overview for details on installing and uninstalling plugins.

SHOW ENGINES can be used to check whether Mroonga is installed correctly:

```
SHOW ENGINES;
...
*************************** 8. row ***************************
      Engine: Mroonga
     Support: YES
     Comment: CJK-ready fulltext search, column store
Transactions: NO
          XA: NO
  Savepoints: NO
...
```

Once the plugin is installed, add a UDF (User-Defined Function) named "last_insert_grn_id", that returns the record ID assigned by groonga in INSERT, by the following SQL.

```
CREATE FUNCTION last_insert_grn_id RETURNS INTEGER SONAME 'ha_mroonga.so';
```

Limitations

- The maximum size of a single key is 4096 bytes.
- The maximum size of all keys is 4GB.
- The maximum number of records in a fulltext index is 268,435,455
- The maximum number of distinct terms in a fulltext index is 268,435,455
- The maximum size of a fulltext index is 256GB

Note that the maximum sizes are not hard limits, and may vary according to circumstance.

For more details, see limitations.html.

Available Character Sets

Mroonga supports a limited number of character sets. These include:

- ASCII
- BINARY
- CP932
- EUCJPMS
- KOI8R
- LATIN1
- SJIS
- UJIS

- UTF8
- UTF8MB4

More Information

Further documentation for Mroonga can be found at

This page is licensed: CC BY-SA / Gnu FDL

Mroonga Status Variables

This page documents status variables related to the Mroonga storage engine. See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

`Mroonga_count_skip`

- Description: Incremented each time the 'fast line count feature' is used. Can be used to check if the feature is working after enabling it.
- Data Type: `numeric`

`Mroonga_fast_order_limit`

- Description: Incremented each time the 'fast ORDER BY LIMIT feature' is used. Can be used to check if the feature is working after enabling it.
- Data Type: `numeric`

This page is licensed: CC BY-SA / Gnu FDL

Mroonga System Variables

This page documents system variables related to the Mroonga storage engine. See Server System Variables for instructions on setting them.

See also the Full list of MariaDB options, system and status variables.

`mroonga_action_on_fulltext_query_error`

- Description: Action to take when encountering a Mroonga fulltext error.
  - `ERROR`: Report an error without logging.
  - `ERROR_AND_LOG`: Report an error with logging (the default)
  - `IGNORE`: No logging or reporting - the error is ignored.
  - `IGNORE_AND_LOG`: Log the error without reporting it.
- Command line: `--mroonga-action-on-fulltext-query-error=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `ERROR_AND_LOG`

`mroonga_boolean_mode_syntax_flags`

- Description: Flags to customize syntax in BOOLEAN MODE searches. Available flags:
  - `DEFAULT`: (=SYNTAX_QUERY,ALLOW_LEADING_NOT)
  - `ALLOW_COLUMN`: Allows `COLUMN:...` syntax in query syntax, an incompatible change to the regular BOOLEAN MODE syntax. Permits multiple indexes in one `MATCH () AGAINST ()`. Can be used in other

operations besides full-text search, such as equal, and prefix search. See Groonga query syntax for more details.
    - `ALLOW_LEADING_NOT` Permits using the `NOT_INCLUDED_KEYWORD` syntax in the query syntax.
    - `ALLOW_UPDATE`: Permits updating values with the `COLUMN:=NEW_VALUE` syntax in the query syntax.
    - `SYNTAX_QUERY`: Mroonga will use Groonga's query syntax, compatible with MariaDB's BOOLEAN MODE syntax. Unless `SYNTAX_SCRIPT` is specified, this mode is always in use.
    - `SYNTAX_SCRIPT`: Mroonga will use Groonga's script syntax, a JavaScript-like syntax. If both `SYNTAX_QUERY` and `SYNTAX_SCRIPT` are specified, `SYNTAX_SCRIPT` will take precedence..
- Command line: `--mroonga-boolean-mode-syntax-flags=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `DEFAULT`

`mroonga_database_path_prefix`

- Description: The database path prefix.
- Command line: `--mroonga-database-path-prefix=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: (Empty)

`mroonga_default_parser`

- Description: The fulltext default parser, for example `TokenBigramSplitSymbolAlphaDigit` or `TokenBigram` (the default). See the list of options at Mroonga Overview:Parser. Deprecated since Mroonga 5.04, use mroonga_default_tokenizer instead.
- Command line: `--mroonga-default-parser=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `TokenBigram`
- Deprecated: MariaDB 10.1.6, Mroonga 5.0.4

`mroonga_default_tokenizer`

- Description: The fulltext default parser, for example `TokenBigramSplitSymbolAlphaDigit` or `TokenBigram` (the default). See the list of options at Mroonga Overview:Parser.
- Command line: `--mroonga-default-tokenizer=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `TokenBigram`
- Introduced: MariaDB 10.1.6, Mroonga 5.0.4

`mroonga_default_wrapper_engine`

- Description: The default engine for wrapper mode.

- Command line: `--mroonga-default-wrapper-engine=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: (Empty)

`mroonga_dry_write`

- Description: If set to `on`, (`off` is default), data is not actually written to the Groonga database. Only really useful to change for benchmarking.
- Command line: `--mroonga-dry-write[={0|1}]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `off`

`mroonga_enable_operations_recording`

- Description: Whether recording operations for recovery to the Groonga database is enabled (default) or not. Requires reopening the database with FLUSH TABLES after changing the variable.
- Command line: `--mroonga-enable-operations-recording={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.2.11, MariaDB 10.1.29

`mroonga_enable_optimization`

- Description: If set to `on` (the default), optimization is enabled. Only really useful to change for benchmarking.
- Command line: `--mroonga-enable-optimization={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `on`

`mroonga_libgroonga_embedded`

- Description: Whether libgroonga is embedded or not.
- Command line: None
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.1.6

`mroonga_libgroonga_support_lz4`

- Description: Whether libgroonga supports lz4 or not.
- Command line: None

- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`mroonga_libgroonga_support_zlib`

- Description: Whether libgroonga supports zlib or not.
- Command line: None
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`mroonga_libgroonga_support_zstd`

- Description: Whether libgroonga supports Zstandard or not.
- Command line: None
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.2.11, MariaDB 10.1.29

`mroonga_libgroonga_version`

- Description: Groonga library version.
- Command line: None
- Scope: Global
- Dynamic: No
- Data Type: `string`

`mroonga_lock_timeout`

- Description: Lock timeout used in Groonga.
- Command line: `--mroonga-lock-timeout=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `900000`
- Range: `-1` to `2147483647`

`mroonga_log_file`

- Description: Name and path of the Mroonga log file.
- Command line: `--mroonga-log-file=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `groonga.log`

`mroonga_log_level`

- Description: Mroonga log file output level, which determines what is logged. Valid levels include:
  - `NONE` No output.
  - `EMERG`: Only emergency error messages, such as database corruption.
  - `ALERT`: Alert messages, such as internal errors.
  - `CRIT`: Critical error messages, such as deadlocks.
  - `ERROR`: Errors, such as API errors.
  - `WARNING`: Warnings, such as invalid arguments.
  - `NOTICE`: Notices, such as a change in configuration or a status change.
  - `INFO`: Information messages, such as file system operations.
  - `DEBUG`: Debug messages, suggested for developers or testers.
  - `DUMP`: Dump messages.
- Command line: `--mroonga-log-level=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `NOTICE`

`mroonga_match_escalation_threshold`

- Description: The threshold to determine whether the match method is escalated. `-1` means never escalate.
- Command line: `--mroonga-match-escalation-threshold=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `-1` to `9223372036854775807`

`mroonga_max_n_records_for_estimate`

- Description: The max number of records to estimate the number of matched records
- Command line: `--mroonga-max-n-records-for-estimate=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1000`
- Range: `-1` to `2147483647`

`mroonga_query_log_file`

- Description: Query log file for Mroonga.
- Command line: `--mroonga-query-log-file=filename`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: (Empty string)
- Introduced: MariaDB 10.2.11

`mroonga_vector_column_delimiter`

- Description: Delimiter to use when outputting a vector column. The default is a white space.
- Command line: `--mroonga-vector-column-delimiter=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `` (white space)

`mroonga_version`

- Description: Mroonga version
- Command line: None
- Scope: Global
- Dynamic: No
- Data Type: `string`

This page is licensed: CC BY-SA / Gnu FDL

Mroonga User-Defined Functions

See Creating Mroonga User-Defined Functions.

Instructions on how to manually create Mroonga's UDFs if they were not automatically installed, ensuring full functionality.

See last_insert_grn_id.

This UDF returns the unique ID assigned by Groonga for the last inserted record, useful for tracking internal record identifiers.

See mroonga_command.

Execute raw Groonga commands directly from MariaDB using this UDF, allowing for advanced administration and inspection of the Groonga database.

See mroonga_escape.

This function escapes special characters in a string to make it safe for use in Mroonga full-text search queries.

See mroonga_highlight_html.

Highlight keywords within a text string using HTML tags, making it easy to display search results with matched terms emphasized.

See mroonga_normalize.

This UDF normalizes a given string using Groonga's normalizers, ensuring consistent text processing for accurate indexing and searching.

See mroonga_snippet.

This function extracts a snippet of text surrounding a keyword from a document, providing necessary context for search result displays.

See mroonga_snippet_html.

Similar to mroonga_snippet, this function generates HTML-formatted snippets, automatically wrapping matched keywords in tags for web display.

This page is licensed: CC BY-SA / Gnu FDL

Creating Mroonga User-Defined Functions

The Mroonga storage engine includes a number of user-defined functions that need to be created before they can be used. If these are not created already during Mroonga setup, you will need to do so yourself. The full list of available functions and the statements to create them are found in `share/mroonga/install.sql`, for example, as of Mroonga 7.07 running on Linux:

```
DROP FUNCTION IF EXISTS last_insert_grn_id;
CREATE FUNCTION last_insert_grn_id RETURNS INTEGER
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_snippet;
CREATE FUNCTION mroonga_snippet RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_command;
CREATE FUNCTION mroonga_command RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_escape;
CREATE FUNCTION mroonga_escape RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_snippet_html;
CREATE FUNCTION mroonga_snippet_html RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_normalize;
CREATE FUNCTION mroonga_normalize RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_highlight_html;
CREATE FUNCTION mroonga_highlight_html RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_query_expand;
CREATE FUNCTION mroonga_query_expand RETURNS STRING
  SONAME 'ha_mroonga.so';
```

This page is licensed: CC BY-SA / Gnu FDL

last_insert_grn_id

Syntax

```
last_insert_grn_id()
```

Description

`last_insert_grn_id` is a user-defined function (UDF) included with the Mroonga storage engine. It returns the unique Groonga id of the last-inserted record. See Creating Mroonga User-Defined Functions for details on creating this UDF if required.

Examples

```
SELECT last_insert_grn_id();
+----------------------+
| last_insert_grn_id() |
+----------------------+
|                    3 |
+----------------------+
```

See Also

- Creating Mroonga User-Defined Functions

This page is licensed: CC BY-SA / Gnu FDL

mroonga_command

Syntax

```
mroonga_command (command)
```

Description

`mroonga_command` is a user-defined function (UDF) included with the Mroonga storage engine. It passes a command to Groonga for execution. See Creating Mroonga User-Defined Functions for details on creating this UDF if required.

- `command` - string, required parameter specifying the command to pass that are executed by Groonga. See the Groonga reference for a list of commands.

Returns the result of the Groonga command.

Note As a administrative function that sends commands directly into Groonga, it completely bypassed MariaDB privilege control. Do not create this UDF on a multi-user system that relies on privilege separation of accounts.

Example

```
SELECT mroonga_command('status');
+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+
| mroonga_command('status')
|
+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+
|
{"alloc_count":593,"starttime":1512022368,"start_time":1512022368,"uptime":13510,"version":"7.0.7","n_que
ries":0,"cache_hit_rate":0.0,"command_version":1,"default_command_version":1,"max_command_version":3} |
```

See Also

- Creating Mroonga User-Defined Functions

This page is licensed: CC BY-SA / Gnu FDL

mroonga_escape

Syntax

```
mroonga_escape (string [,special_characters])
```

- `string` - required parameter specifying the text you want to escape
- `special_characters` - optional parameter specifying the characters to escape

Description

`mroonga_escape` is a user-defined function (UDF) included with the Mroonga storage engine, used for escaping a string. See Creating Mroonga User-Defined Functions for details on creating this UDF if required.

If no `special_characters` parameter is provided, by default `+-<>*()":` are escaped.

Returns the escaped string.

Example

```
SELECT mroonga_escape("+-<>~*()\"\:");
'\\+\\-\\<\\>\\~\\*\\(\\)\\"\\:
```

See Also

- Creating Mroonga User-Defined Functions

This page is licensed: CC BY-SA / Gnu FDL

mroonga_highlight_html

Syntax

```
mroonga_highlight_html(text[[, query AS query]])

mroonga_highlight_html(text[[, keyword1, ..., keywordN]])
```

Description

`mroonga_highlight_html` is a user-defined function (UDF) included with the Mroonga storage engine. It highlights the specified keywords in the target text. See Creating Mroonga User-Defined Functions for details on creating this UDF if required.

The optional parameter can either be one or more *keywords*, or a Groonga *query*.

The function highlights the specified keywords in the target text by surrounding each keyword with `<span class="keyword">...</span>`, and escaping special HTML characters such as `<` and `>`.

Returns highlighted HTML.

Examples

```
SELECT mroonga_highlight_html('<p>MariaDB includes the Mroonga storage engine</p>.')
  AS highlighted_html;
+-------------------------------------------------------------------+
| highlighted_html                                                  |
+-------------------------------------------------------------------+
| &lt;p&gt;MariaDB includes the Mroonga storage engine&lt;/p&gt;. |
+-------------------------------------------------------------------+
```

Highlighting the words `MariaDB` and `Mroonga` in a given text:

```
SELECT mroonga_highlight_html('MariaDB includes the Mroonga storage engine.', 'MariaDB', 'Mroonga')
  AS highlighted_html;
+-----------------------------------------------------------------------------------------------------------------
+
| highlighted_html
|
+-----------------------------------------------------------------------------------------------------------------
+
| <span class="keyword">MariaDB</span> includes the <span class="keyword">Mroonga</span> storage engine.
|
+-----------------------------------------------------------------------------------------------------------------
+
```

The same outcome, formulated as a Groonga query:

```
SELECT mroonga_highlight_html('MariaDB includes the Mroonga storage engine.', 'MariaDB OR Mroonga'
  AS query) AS highlighted_text;
+-----------------------------------------------------------------------------------------------------------------
+
| highlighted_text
|
+-----------------------------------------------------------------------------------------------------------------
+
| <span class="keyword">MariaDB</span> includes the <span class="keyword">Mroonga</span> storage engine.
|
+-----------------------------------------------------------------------------------------------------------------
+
```

See Also

- Creating Mroonga User-Defined Functions

This page is licensed: CC BY-SA / Gnu FDL

mroonga_normalize

Syntax

```
mroonga_normalize(string[, normalizer_name])
```

Description

`mroonga_normalize` is a user-defined function (UDF) included with the Mroonga storage engine. It uses Groonga's normalizer to normalize text. See Creating Mroonga User-Defined Functions for details on creating this UDF if required.

Given a string, returns the normalized text.

See the Groonga Normalizer Reference for details on the Groonga normalizers. The default if no normalizer is provided is `NormalizerAuto`.

Examples

```
SELECT mroonga_normalize("AB▫▫");
+------------------------------+
| mroonga_normalize("AB▫▫")   |
+------------------------------+
| ab▫▫▫▫▫                  |
+------------------------------+
```

See Also

- Creating Mroonga User-Defined Functions

This page is licensed: CC BY-SA / Gnu FDL

mroonga_snippet

Syntax

```
mroonga_snippet document,
                max_length,
                max_count,
                encoding,
                skip_leading_spaces,
                html_escape,
                snippet_prefix,
                snippet_suffix,
                word1, word1_prefix, word1_suffix
                ...
                [wordN wordN_prefix wordN_suffix]
```

Description

`mroonga_snippet` is a user-defined function (UDF) included with the Mroonga storage engine. It provides a keyword with surrounding text, or the keyword in context. See Creating Mroonga User-Defined Functions for details on creating this UDF if required.

The required parameters include:

- `document` - Column name or string value.
- `max_length` - Maximum length of the snippet, in bytes.
- `max_count` - Maximum snippet elements (N word).
- `encoding` - Encoding of the document, for example `cp932_japanese_ci`
- `skip_leading_spaces` - `1` to skip leading spaces, `0` to not skip.
- `html_escape` = `1` to enable HTML espape, `0` to disable.
- `prefix` - Snippet start text.
- `suffix` - Snippet end text.

The optional parameters include:

- `wordN` - A word.
- `wordN_prefix` - wordN start text.
- `wordN_suffix` - wordN end text

It can be used in both storage and wrapper mode.

Returns the snippet string.

Example

```
```

See Also

- Creating Mroonga User-Defined Functions

This page is licensed: CC BY-SA / Gnu FDL

mroonga_snippet_html

Description

`mroonga_snippet_html` is a user-defined function (UDF) included with the Mroonga storage engine. It provides a keyword with surrounding text, or the keyword in context. It is still considered experimental. See Creating Mroonga User-Defined Functions for details on creating this UDF if required.

See Also

- Creating Mroonga User-Defined Functions
- mroonga_snippet

This page is licensed: CC BY-SA / Gnu FDL

MyISAM

See MyISAM Overview.

A detailed overview of the MyISAM storage engine, including its file structure, features, and limitations compared to newer engines like Aria.

See MyISAM Index Storage Space.

This page explains how MyISAM stores and compresses indexes, detailing space usage for different key types and compression strategies.

See MyISAM Storage Formats.

Learn about the three storage formats supported by MyISAM: FIXED (static), DYNAMIC (variable length), and COMPRESSED (read-only).

See MyISAM System Variables.

A reference for system variables that configure MyISAM behavior, such as key cache sizes, recovery modes, and concurrent insert settings.

This page is licensed: CC BY-SA / Gnu FDL

MyISAM Overview

The MyISAM storage engine was the default storage engine from MySQL 3.23 until it was replaced as default by InnoDB in MariaDB and MySQL 5.5. Historically, MyISAM is a replacement for the older ISAM engine, removed in MySQL 4.1.

It's a light, non-transactional engine with great performance, is easy to copy between systems and has a small data footprint.

You're encouraged to rather use the Aria storage engine for new applications, which has even better performance in most cases and the goal of being crash-safe.

A MyISAM table is stored in three files on disk. There's a table definition file with an extension of `.frm`, a data file with the extension `.MYD`, and an index file with the extension `.MYI`.

MyISAM features

- Does not support transactions.
- Does not support foreign keys.
- Supports FULLTEXT indexes.
- Supports GIS data types.

- Storage limit of 256TB.
- Maximum of 64 indexes per table.
- Maximum of 32 columns per index.
- Maximum index length of 1000 bytes.
- Limit of (232)2 (1.844E+19) rows per table.
- Supports large files up to 63-bits in length where the underlying system supports this.
- All data is stored with the low byte first, so all files will still work if copied to other systems or other machines.
- The data file and the index file can be placed on different devices to improve speed.
- Supports table locking, not row locking.
- Supports a key buffer that is segmented in MariaDB.
- Supports concurrent inserts.
- Supports fixed length, dynamic and compressed formats - see MyISAM Storage Formats.
- Numeric index values are stored with the high byte first, which enables more efficient index compression.
- Data values are stored with the low byte first, making it mostly machine and operating system independent. The only exceptions are if a machine doesn't use two's-complement signed integers and the IEEE floating-point format.
- Can be copied between databases or systems with normal system tools, as long as the files are not open on either system. Use FLUSH_TABLES to ensure files are not in use.
- There are a number of tools for working with MyISAM tables. These include:
  - mariadb-check for checking or repairing
  - myisamchk for checking or repairing
  - myisampack for compressing
- It is possible to build a MERGE table on the top of one or more MyISAM tables.

This page is licensed: CC BY-SA / Gnu FDL

MyISAM Index Storage Space

Regular MyISAM tables make use of B-tree indexes.

String indexes are space-compressed, which reduces the size of VARCHARs that don't use the full length, or a string that has trailing spaces. String indexes also make use of prefix-compression, where strings with identical prefixes are compressed.

Numeric indexes can also be prefix-compressed if the PACK_KEYS=1 option is used. Regardless, the high byte is always stored first, which allows a reduced index size.

In the worst case, with no strings being space-compressed, the total index storage space are (index_length+4)/0.67 per index.

This page is licensed: CC BY-SA / Gnu FDL

MyISAM Storage Formats

The MyISAM storage engine supports three different table storage formats.

These are FIXED, DYNAMIC and COMPRESSED. FIXED and DYNAMIC can be set with the ROW FORMAT option in the CREATE TABLE statement, or are chosen automatically depending on the columns the table contains. COMPRESSED can only be set via the myisampack tool.

The SHOW TABLE STATUS statement can be used to see the storage format used by a table. Note that `COMPRESSED` tables are reported as `DYNAMIC` in that context.

Fixed-length

Fixed-length (or static) tables contain records of a fixed-length. Each column is the same length for all records, regardless of the actual contents. It is the default format if a table has no BLOB, TEXT, VARCHAR or VARBINARY fields, and no ROW FORMAT is provided. You can also specify a fixed table with ROW_FORMAT=FIXED in the table definition.

Tables containing BLOB or TEXT fields cannot be FIXED, as by design these are both dynamic fields. However, no error or warning are raised if you specify FIXED.

Fixed-length tables have a number of characteristics

- fast, since MariaDB will always know where a record begins
- easy to repair: myisamchk is always able to recover all rows, except for the last one if it is not entirely written
- easy to cache
- take up more space than dynamic or compressed tables, as the maximum amount of storage space are allocated to each record.
- reconstructing after a crash is uncomplicated due to the fixed positions
- no fragmentation or need to re-organize, unless records have been deleted and you want to free the space up.

Dynamic

Dynamic tables contain records of a variable length. It is the default format if a table has any BLOB, TEXT, VARCHAR or VARBINARY fields, and no ROW FORMAT is provided. You can also specify a DYNAMIC table with ROW_FORMAT=DYNAMIC in the table definition. If the table contains BLOB or TEXT columns, its format is always DYNAMIC, and the ROW FORMAT option is ignored.

Dynamic tables have a number of characteristics

- Each row contains a header indicating the length of the row.
- Rows tend to become fragmented easily. UPDATING a record to be longer will likely ensure it is stored in different places on the disk. Use OPTIMIZE TABLE when the fragmentation is too high.
- All string columns with a length of four or more are dynamic.
- They require much less space than fixed-length tables.
- Restoring after a crash is more complicated than with FIXED tables. Some fragments may be lost.

If a DYNAMIC table has some frequently-accessed fixed-length columns, it could be a good idea to move them into a separate FIXED table to avoid fragmentation.

Compressed

Compressed tables are a read-only format, created with the myisampack tool. This can be done while the server is running, but external lock must not be disabled. myisamchk is used to uncompress them.

Compressed tables have a number of characteristics:

- while the data is read-only, DDL statements such as DROP TABLE and TRUNCATE TABLE will still function.
- take much less space than fixed or dynamic tables. Each data has usually a 40-70% compression ratio
- rows are compressed separately, reducing access overhead.
- row headers are from one to three bytes.
- rows can be compressed with different compression types, including
  - prefix space compression
  - suffix space compression
  - columns with small sets of values are converted to ENUM
  - numeric zeros are stored with only one bit

- integer columns are reduced to the smallest int type that can hold the contents

See Also

- Why we still need MyISAM (for read-only tables) describes an important use case for MyISAM compressed tables.

This page is licensed: CC BY-SA / Gnu FDL

MyISAM System Variables

This page documents system variables related to the MyISAM storage engine. For options, see MyISAM Options.

See Server System Variables for a complete list of system variables and instructions on setting them.

See also the Full list of MariaDB options, system and status variables.

`key_buffer_size`

- Description: Size of the buffer for the index blocks used by MyISAM tables and shared for all threads. See Optimizing key_buffer_size for more on selecting the best value.
- Command line: `--key-buffer-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `134217728`
- Range: `8` upwards (upper limit determined by operating system per process limit)

`key_cache_age_threshold`

- Description: The lower the setting, the more quickly buffers move from the hot key cache sublist to the warm sublist.
- Command line: `--key-cache-age-threshold=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `300`
- Range: `100` to `4294967295`

`key_cache_block_size`

- Description: MyISAM key cache block size in bytes .
- Command line: `--key-cache-block-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1024`
- Range: `512` to `16384`

`key_cache_division_limit`

- Description: Percentage to use for the warm key cache buffer list (the remainder is allocated between the hot and cold caches).
- Command line: `--key-cache-division-limit=#`

- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `1` to `100`

`key_cache_file_hash_size`

- Description: Number of hash buckets for open and changed files. If you have many MyISAM files open you should increase this for faster flushing of changes. A good value is probably 1/10th of the number of possible open MyISAM files.
- Command line: `--key-cache-file-hash-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `512`
- Range: `128` to `16384`

`key_cache_segments`

- Description: The number of segments in a key cache. See Segmented Key Cache.
- Command line: `--key-cache-segments=#`
- Scope: Global
- Dynamic: Yes
- Type: numeric
- Default Value: `0` (non-segmented)
- Range: `0` to `64`

`myisam_block_size`

- Description: Block size to be used for MyISAM index pages.
- Command line: `--myisam-block-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1024`

`myisam_data_pointer_size`

- Description: Size in bytes of the default pointer, used in a MyISAM CREATE TABLE with no MAX_ROWS option.
- Command line: `--myisam-data-pointer-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `6`
- Range: `2` to `7`

`myisam_max_extra_sort_file_size`

- Description: Removed in MySQL 5.0.6, was used as a way to force long character keys in large tables to use the key cache method.
- Removed: MySQL 5.0.6

`myisam_max_sort_file_size`

- Description: Maximum size in bytes of the temporary file used while recreating a MyISAM index. If the this size is exceeded, the slower process of using the key cache is done instead.
- Command line: `--myisam-max-sort-file-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value - 32 bit: `2147483648`
- Default Value - 64 bit: `9223372036854775807`

`myisam_mmap_size`

- Description: Maximum memory in bytes that can be used for memory mapping compressed MyISAM files. Too high a value may result in swapping if there are many compressed MyISAM tables.
- Command line: `--myisam-mmap-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value - 32 bit: `4294967295`
- Default Value - 64 bit: `18446744073709547520`
- Range - 32-bit: `7` to `4294967295`
- Range - 64-bit: `7` to `18446744073709547520`

`myisam_recover_options`

- Description: MyISAM recovery mode. Multiple options can be selected, comma-delimited. Using no argument is equivalent to specifying `DEFAULT`, while specifying “” is equivalent to `OFF`. If enabled each time the server opens a MyISAM table, it checks whether it has been marked as crashed, or wasn’t closed properly. If so, mysqld will run a check and then attempt to repair the table, writing to the error log beforehand.
  - OFF: No recovery.
  - BACKUP: If the data file is changed while recovering, saves a backup of the .MYD data file. t.MYD are saved as t.MYD-datetime.BAK.
  - BACKUP_ALL: Same as `BACKUP` but also backs up the .MYI index file. t.MYI are saved as t.MYI-datetime.BAK.
  - DEFAULT: Recovers without backing up, forcing, or quick checking.
  - FORCE: Runs the recovery even if it determines that more than one row from the data file are lost.
  - QUICK: Does not check rows in the table if there are no delete blocks.
- Command line: `--myisam-recover-options[=name]`
- Scope: Global
- Dynamic: No
- Data Type: `enumeration`
- Default Value:

    - `BACKUP,QUICK` (>= MariaDB 10.2.4)
    - `DEFAULT` (<= MariaDB 10.2.3)
    - `OFF`
- Valid Values: `OFF`, `DEFAULT`, `BACKUP`, `BACKUP_ALL`, `FORCE` or `QUICK`

`myisam_repair_threads`

- Description: If set to more than `1`, the default, MyISAM table indexes each have their own thread during repair and sorting. Increasing from the default will usually result in faster repair, but will use more CPU and memory.
- Command line: `--myisam-repair-threads=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range - 32-bit: `1` to `4294967295`
- Range - 64-bit: `1` to `18446744073709547520`

`myisam_sort_buffer_size`

- Description: Size in bytes of the buffer allocated when creating or sorting indexes on a MyISAM table. Increase for better myisamchk -r, –recover performance.
- Command line: `--myisam-sort-buffer-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `BIGINT UNSIGNED`
- Default Value: `134216704`
- Range:
    - (>= MariaDB 10.11.5, MariaDB 10.6.15) : `4096` to `1152921504606846975`
    - (< = MariaDB 10.11.4, MariaDB 10.6.14) : `4096` to `18446744073709551615`

`myisam_stats_method`

- Description: Determines how NULLs are treated for the purposes of MyISAM index statistics. If set to `nulls_equal`, all NULL index values are treated as a single group. This is usually fine, but if you have large numbers of NULLs the average group size is slanted higher, and the optimizer may miss using the index for ref accesses when it would be useful. If set to `nulls_unequal`, the default, the opposite approach is taken, with each NULL forming its own group of one. Conversely, the average group size is slanted lower, and the optimizer may use the index for ref accesses when not suitable. Setting to `nulls_ignored` ignores NULLs altogether from index group calculations. See also Index Statistics, aria_stats_method, innodb_stats_method.
- Command line: `--myisam-stats-method=name`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `NULLS_UNEQUAL`
- Valid Values: `NULLS_UNEQUAL, NULLS_EQUAL, NULLS_IGNORED`

`myisam_use_mmap`

- Description: If set to `1` (0 is default), memory mapping are used to reading and writing MyISAM tables.

- Command line: `--myisam-use-mmap`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

This page is licensed: CC BY-SA / Gnu FDL

MyRocks

> **Note:**
>
> MyRocks is a community contribution to MariaDB Server. It is based on RocksDB and on Facebook's MyRocks, which are developed and maintained upstream, outside MariaDB.

See About MyRocks for MariaDB.

MyRocks is a storage engine based on RocksDB, optimized for high-write workloads and flash storage, offering superior compression and reduced write amplification.

See Building MyRocks in MariaDB.

Learn how to build the MyRocks storage engine from source within the MariaDB Server build process, including necessary dependencies and configuration options.

See Differences Between MyRocks Variants.

Differences between variants of the MyRocks storage engine.

See Getting Started with MyRocks.

A guide to installing and configuring MyRocks, including enabling the plugin, setting up basic tables, and understanding key configuration parameters.

See Loading Data Into MyRocks.

Best practices and methods for efficiently loading large datasets into MyRocks tables, including using bulk loading features to improve performance.

See MyRocks Bloom Filters.

Learn how to configure and use Bloom filters in MyRocks to speed up point lookups by probabilistically determining if a key exists in a data file.

See MyRocks CHECK TABLE.

Details on using the CHECK TABLE statement with MyRocks to verify the integrity of tables and indexes, and how it differs from other engines.

See MyRocks Data Compression.

Explore the data compression options available in MyRocks, including different algorithms (Zstd, LZ4, etc.) and how to configure them per column family.

See MyRocks Group Commit with Binary log.

Understand how MyRocks implements group commit to coordinate with the binary log, ensuring data consistency and crash safety for replicated transactions.

See MyRocks Index-Only Scans.

MyRocks storage engine scans that only use indexes.

See MyRocks and Replication.

This page details how MyRocks integrates with MariaDB replication, specifically addressing limitations with statement-based replication and lack of Gap Lock support.

See MyRocks START TRANSACTION WITH CONSISTENT SNAPSHOT.

MyRocks storage engine transactions with consistent snapshot.

See MyRocks Column Families.

Learn about MyRocks column families, a mechanism for grouping data similar to tablespaces, to optimize compression and Bloom filter settings per family.

See MyRocks Performance Troubleshooting.

A guide to diagnosing and resolving performance issues in MyRocks using status variables, `SHOW ENGINE ROCKSDB STATUS`, and RocksDB performance context.

See Optimizer Statistics in MyRocks.

MyRocks storage engine statistics for the query optimizer.

See MyRocks Transactional Isolation.

MyRocks storage engine transactional isolatioin.

See MyRocks Status Variables.

A list of MyRocks-specific status variables providing metrics on cache hits, compaction statistics, block cache usage, and other internal operations.

See MyRocks System Variables.

A comprehensive reference for MyRocks system variables, allowing fine-tuning of performance, memory usage, compaction, and other internal behaviors.

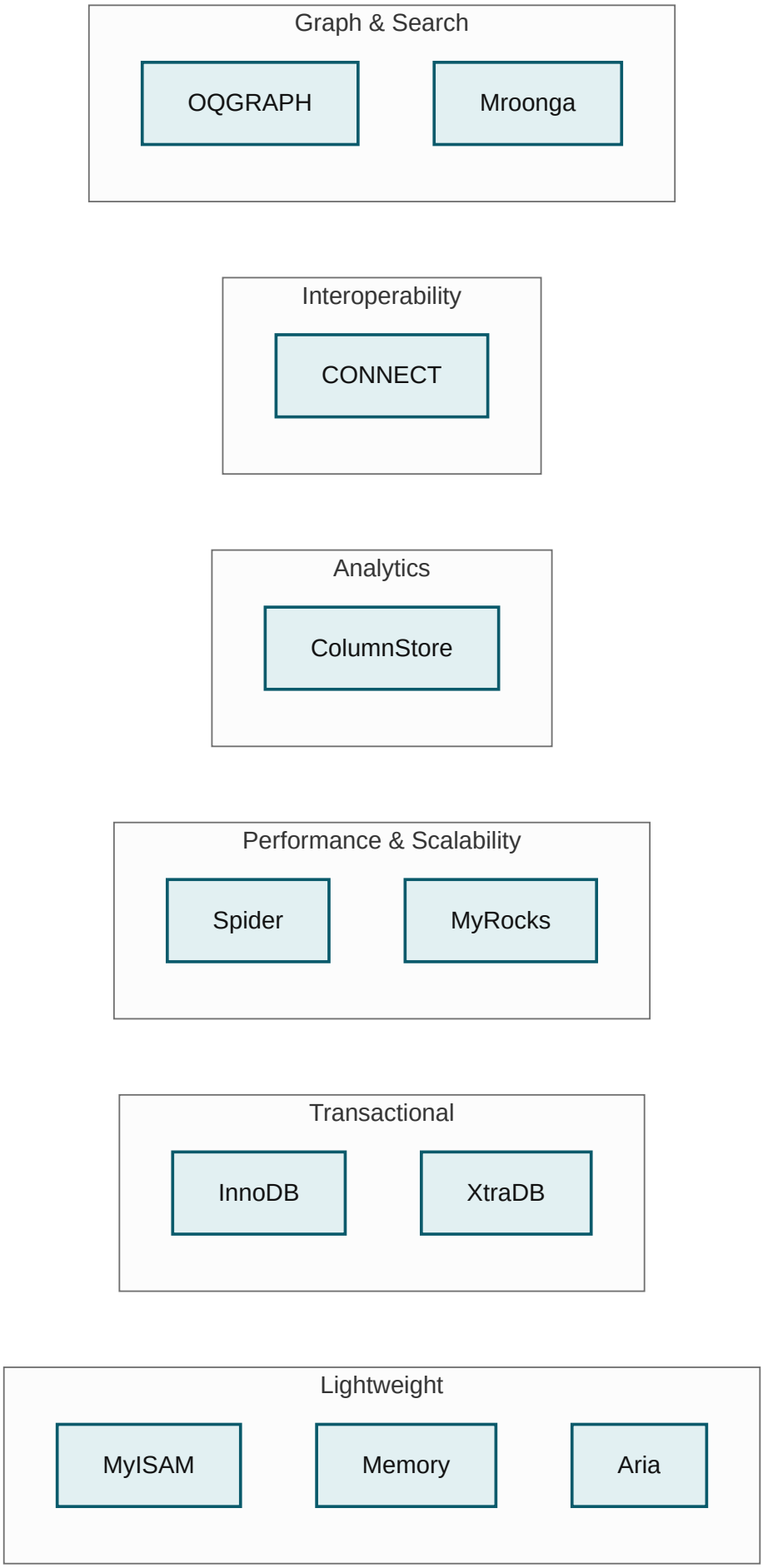
This page is licensed: CC BY-SA / Gnu FDL

About MyRocks for MariaDB

About MyRocks for MariaDB

MyRocks is an open source storage engine that was originally developed by Facebook.

MyRocks has been extended by the MariaDB engineering team to be a pluggable storage engine that you use in your MariaDB solutions. It works seamlessly with MariaDB features. This openness in the storage layer allows you to use the right storage engine to optimize your usage requirements, which provides optimum performance. Community contributions are one of MariaDB's greatest advantages over other databases. Under the lead of our developer Sergey Petrunia, MyRocks in MariaDB is occasionally being merged with upstream MyRocks from Facebook.
See more at: facebook-myrocks-mariadb#sthash.ZlEr7kNq.dpuf


Graph & Search
OQGRAPH
Mroonga
Interoperability
CONNECT
Analytics
ColumnStore
Performance & Scalability
Spider
MyRocks
Transactional
InnoDB
XtraDB
Lightweight
MyISAM
Memory
Aria

MariaDB's pluggable storage layer: engines grouped by purpose, with MyRocks among the performance-and-scalability engines.

MyRocks, typically, gives greater performance for web scale type applications. It can be an ideal storage engine solution when you have workloads that require greater compression and IO efficiency. It uses a Log Structured Merge (LSM) architecture, which has advantages over B-Tree algorithms, to provide efficient data ingestion, like read-free replication slaves, or fast bulk data loading.
MyRocks distinguishing features include:

- compaction filter
- merge operator
- backup
- column families
- bulk loading
- persistent cache

For more MyRocks features see: Features-Not-in-LevelDB

Benefits

On production workloads, MyRocks was tested to prove that it provides:

Greater Space Efficiency

- 2x more compression
  MyRocks has 2x better compression compared to compressed InnoDB, 3-4x better compression compared to uncompressed InnoDB, meaning you use less space.

Greater Writing Efficiency

- 2x lower write rates to storage
  MyRocks has a 10x less write amplification compared to InnoDB, giving you better endurance of flash storage and improving overall throughput.

Faster Data Loading

- faster database loads
  MyRocks writes data directly onto the bottommost level, which avoids all compaction overheads when you enable faster data loading for a session.

Faster Replication

- No random reads for updating secondary keys, except for unique indexes. The Read-Free Replication option does away with random reads when updating primary keys, regardless of uniqueness, with a row-based binary logging format.

myrocks.iofacebook-myrocks-mariadb

Requirements and Limitations

- MyRocks is included from MariaDB 10.2.5.
- MyRocks is available in the MariaDB Server packages for Linux and Windows.
- Maria DB optimistic parallel replication may not be supported.
- MyRocks is not available for 32-bit platforms
- Galera Cluster is tightly integrated into InnoDB storage engine (it also supports Percona's XtraDB which is a modified version of InnoDB). Galera Cluster does not work with any other storage engines, including MyRocks.

MyRocks builds are available on platforms that support a sufficiently modern compiler, for example:

- Ubuntu Trusty, Xenial, (amd64 and ppc64el)
- Ubuntu Yakkety (amd64)
- Debian Jessie, stable (amd64, ppc64el)
- Debian Stretch, Sid (testing and unstable) (amd64)
- CentOS/RHEL 7 (amd64)
- Centos/RHEL 7.3 (amd64)
- Fedora 24 and 25 (amd64)
- OpenSUSE 42 (amd64)
- Windows 64 (zip and MSI)

This page is licensed: CC BY-SA / Gnu FDL

Building MyRocks in MariaDB

This page describes how to get MyRocks in MariaDB when compiling MariaDB from source. (See Build-Steps for instructions how to build the upstream.)

Build Process and Requirements

MariaDB compile process will compile MyRocks into `ha_rocksdb.so` by default if the platform supports it (That is, no WITH_ROCKSDB switch is necessary).

Platform requirements:

- A 64-bit platform (due to some 32 bit compilers having difficulties with RocksDB)
- git installed (or git submodules fetched somehow)
- A sufficiently recent compiler:
  - gcc >= 4.8, or
  - clang >= 3.3, or
  - MS Visual Studio 2015 or newer

Building on Ubuntu 16.04

The steps were checked on a fresh install of Ubuntu 16.04.2 LTS Xenial.

```
sudo apt-get update
sudo apt-get -y install g++ cmake libbz2-dev libaio-dev bison zlib1g-dev libsnappy-dev
sudo apt-get -y install libgflags-dev libreadline6-dev libncurses5-dev libssl-dev liblz4-dev gdb git
;
```

```
git clone https://github.com/MariaDB/server.git mariadb-10.2
cd mariadb-10.2
git checkout 10.2
git submodule init
git submodule update
cmake .
make -j10
```

This should produce `storage/rocksdb/ha_rocksdb.so` which is MyRocks storage engine in the loadable form.

Starting MyRocks

MyRocks does not require any special way to initialize the data directory.
Minimal my.cnf flle:

```
cat > ~/my1.cnf <<EOF
[mysqld]

datadir=../mysql-test/var/install.db
plugin-dir=../storage/rocksdb
language=./share/english
socket=/tmp/mysql.sock
port=3307

plugin-load=ha_rocksdb
default-storage-engine=rocksdb
EOF
```

Run the server like this

```
(cd mysql-test; ./mtr alias)
cp -r mysql-test/var/install.db ~/data1
cd ../sql
./mysqld --defaults-file=~/my1.cnf
```

Compression libraries.
Supported compression libraries are listed in rocksdb_supported_compression_types. Compiling like the above, I get:

```
Snappy,Zlib,LZ4,LZ4HC
```

This page is licensed: CC BY-SA / Gnu FDL

Differences Between MyRocks Variants

MyRocks is available in

- Facebook's (FB) MySQL branch (originally based on MySQL 5.6)
- MariaDB (from 10.2 and 10.3)
- Percona Server from 5.7

This page lists differences between these variants.

This is a work in progress. The contents are not final

RocksDB Data Location

FB and Percona store RocksdDB files in $datadir/`.rocksdb`. MariaDB puts them in $datadir/`#rocksdb`. This is more friendly for packaging and OS scripts.

Compression Algorithms

- FB's branch doesn't provide binaries. One needs to compile it with appropriate compression libraries.
- In MariaDB, available compression algorithms can be seen in the rocksdb_supported_compression_types variable. From MariaDB 10.7, algorithms can be installed as a plugin. In earlier versions, the set of supported compression algorithms depends on the platform.
  - On Ubuntu 16.04 (current LTS) it is `Snappy,Zlib,LZ4,LZ4HC` .
  - On CentOS 7.4 it is `Snappy,Zlib`.
  - In the bintar tarball it is `Snappy,Zlib`.
- Percona Server supports: `Zlib, ZSTD, LZ4 (the default), LZ4HC`. Unsupported algorithms: `Snappy, BZip2, XPress`.

RocksDB Version Information

- FB's branch provides the `rocksdb_git_hash` *status* variable.
- MariaDB provides the `@@rocksdb_git_hash` *system* variable.
- Percona Server doesn't provide either.

RocksDB Version

- Facebook's branch uses RocksDB 5.10.0 (the version number can be found in `include/rocksdb/version.h`)

```
commit ba295cda29daee3ffe58549542804efdfd969784
Author: Andrew Kryczka <andrewkr@fb.com>
Date:   Fri Jan 12 11:03:55 2018 -0800
```

- MariaDB currently uses 5.8.0

```
commit 9a970c81af9807071bd690f4c808c5045866291a
Author: Yi Wu <yiwu@fb.com>
Date:   Wed Sep 13 17:21:35 2017 -0700
```

- Percona Server uses 5.8.0

```
commit ab0542f5ec6e7c7e405267eaa2e2a603a77d570b
Author: Maysam Yabandeh <myabandeh@fb.com>
Date:   Fri Sep 29 07:55:22 2017 -0700
```

Binlog Position in information_schema.rocksdb_global_info

- FB branch provides information_schema.rocksdb_global_info type=BINLOG, NAME={FILE, POS, GTID}.
- Percona Server doesn't provide it.
- MariaDB doesn't provide it.

One use of that information is to take the output of `myrocks_hotbackup` and make it a new master.

Gap Lock Detector

- FB branch has a "Gap Lock Detector" feature. It is at the SQL layer. It can be controlled with `gap_lock_XXX` variables and is disabled by default (gap-lock-raise-error=false, gap-lock-write-lock=false).
- Percona Server has gap lock checking ON but doesn't seem to have any way to control it?
  Queries that use Gap Lock on MyRocks fail with an error like this:

```
INSERT INTO tbl2 SELECT * FROM tbl1;
ERROR 1105 (HY000): Using Gap Lock without full unique key in multi-table or multi-statement transactions
is not allowed. You need to either rewrite queries to use all unique key columns in WHERE equal
conditions,
or rewrite to single-table, single-statement transaction.  Query: insert into tbl2 select * from tbl1
```

- MariaDB doesn't include the Gap Lock Detector.

Generated Columns

- Both MariaDB and Percona Server support generated columns, but neither one supports them for the MyRocks storage engine (attempts to create a table will produce an error).
- Invisible columns in MariaDB 10.3 are supported (as they are an SQL layer feature).

rpl_skip_tx_api

Facebook's branch has a performance feature for replication slaves, `rpl_skip_tx_api`. It is not available in MariaDB or in Percona Server.

Details

The above comparison was made using

- FB/MySQL 5.6.35
- Percona Server 5.7.20-19-log
- MariaDB 10.2.13 (MyRocks is beta)

This page is licensed: CC BY-SA / Gnu FDL

Getting Started with MyRocks

MyRocks is a storage engine that adds the RocksDB database to MariaDB. RocksDB is an LSM database with a great compression ratio that is optimized for flash storage.

The storage engine must be installed before it can be used.

Installing the Plugin's Package

The MyRocks storage engine's shared library is included in MariaDB packages as the `ha_rocksdb.so` or `ha_rocksdb.dll` shared library on systems where it can be built.

Installing on Linux

The MyRocks storage engine is included in binary tarballs on Linux.

Installing with a Package Manager

The MyRocks storage engine can also be installed via a package manager on Linux. In order to do so, your system needs to be configured to install from one of the MariaDB repositories.

You can configure your package manager to install it from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script.

You can also configure your package manager to install it from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool.

Installing with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using yum or dnf. Starting with RHEL 8 and Fedora 22, `yum` has been replaced by `dnf`, which is the next major version of `yum`. However, `yum` commands still work on many systems that use `dnf`:

```
sudo yum install MariaDB-rocksdb-engine
```

Installing with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant DEB package from MariaDB's repository using apt-get:

```
sudo apt-get install mariadb-plugin-rocksdb
```

Installing with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using zypper:

```
sudo zypper install MariaDB-rocksdb-engine
```

Installing on Windows

The MyRocks storage engine is included in MSI and ZIP packages on Windows.

Installing the Plugin

Once the shared library is in place, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'ha_rocksdb';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = ha_rocksdb
```

Note: When installed with a package manager, an option file that contains the –plugin-load-add option may also be installed. The RPM package installs it as `/etc/my.cnf.d/rocksdb.cnf`, and the DEB package installs it as `/etc/mysql/mariadb.conf.d/rocksdb.cnf`

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'ha_rocksdb';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Verifying the Installation

After installing MyRocks you will see RocksDB in the list of plugins:

```
SHOW PLUGINS;
+-------------------------------+----------+--------------------+---------------+---------+
| Name                          | Status   | Type               | Library       | License |
+-------------------------------+----------+--------------------+---------------+---------+
...
| ROCKSDB                       | ACTIVE   | STORAGE ENGINE     | ha_rocksdb.so | GPL     |
| ROCKSDB_CFSTATS               | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_DBSTATS               | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_PERF_CONTEXT          | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_PERF_CONTEXT_GLOBAL   | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_CF_OPTIONS            | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_COMPACTION_STATS      | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_GLOBAL_INFO           | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_DDL                   | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_INDEX_FILE_MAP        | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_LOCKS                 | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
| ROCKSDB_TRX                   | ACTIVE   | INFORMATION SCHEMA | ha_rocksdb.so | GPL     |
...
+-------------------------------+----------+--------------------+---------------+---------+
```

Compression

Supported compression types are listed in the rocksdb_supported_compression_types variable:

```
SHOW VARIABLES LIKE 'rocksdb_supported_compression_types';
+-----------------------------------+-------------+
| Variable_name                     | Value       |
+-----------------------------------+-------------+
| rocksdb_supported_compression_types | Snappy,Zlib |
+-----------------------------------+-------------+
```

See MyRocks and Data Compression for more.

System and Status Variables

All MyRocks system variables and status variables are prefaced with “rocksdb”, so you can query them with, for example:

```
SHOW VARIABLES LIKE 'rocksdb%';
SHOW STATUS LIKE 'rocksdb%';
```

This page is licensed: CC BY-SA / Gnu FDL

Loading Data Into MyRocks

Being a write-optimized storage engine, MyRocks has special ways to load data much faster than normal INSERTs would.

See:

- the section about “Migrating from InnoDB to MyRocks in production” has some clues.
- Data-Loading covers the topic in greater detail.

> **Note:**
>
> When loading data with rocksdb_bulk_load=1 and the data conflicts with the data already in the database, one may get non-trivial errors, for example:

```
ERROR 1105 (HY000): [./.rocksdb/test.t1_PRIMARY_2_0.bulk_load.tmp] bulk load error:
  Invalid argument: External file requires flush
```

This page is licensed: CC BY-SA / Gnu FDL

MyRocks Bloom Filters

Bloom filters are used to reduce read amplification. Bloom filters can be set on a per-column family basis (see myrocks-column-families).

Bloom Filter Parameters

- How many bits to use
- whole_key_filtering=true/false
- Whether the bloom filter is for the entire key or for the prefix. In case of a prefix, you need to look at the index definition and compute the desired prefix length.

Computing Prefix Length

- It’s 4 bytes for `index_nr`

- Then, for fixed-size columns (integer, date[time], decimal) it is key_length as shown by `EXPLAIN`. For VARCHAR columns, determining the length is tricky (It depends on the values stored in the table. Note that MyRocks encodes VARCHARs with "Variable-Length Space-Padded Encoding" format).

Configuring Bloom Filter

To enable 10-bit bloom filter for 8-byte prefix length for column family "cf1", put this into my.cnf:

```
rocksdb_override_cf_options='cf1={block_based_table_factory=
{filter_policy=bloomfilter:10:false;whole_key_filtering=0;};prefix_extractor=capped:8};'
```

and restart the server.

Check if the column family actually uses the bloom filter:

```
SELECT *
FROM information_schema.rocksdb_cf_options
WHERE
  cf_name='cf1' AND
  option_type IN ('TABLE_FACTORY::FILTER_POLICY','PREFIX_EXTRACTOR');
```

```
+---------+-------------------------------+---------------------------+
| CF_NAME | OPTION_TYPE                   | VALUE                     |
+---------+-------------------------------+---------------------------+
| cf1     | PREFIX_EXTRACTOR              | rocksdb.CappedPrefix.8    |
| cf1     | TABLE_FACTORY::FILTER_POLICY | rocksdb.BuiltinBloomFilter |
+---------+-------------------------------+---------------------------+
```

Checking if Bloom Filter is Useful

Watch these status variables:

```
SHOW status LIKE '%bloom%';
+-----------------------------------+-------+
| Variable_name                     | Value |
+-----------------------------------+-------+
| Rocksdb_bloom_filter_prefix_checked | 1     |
| Rocksdb_bloom_filter_prefix_useful  | 0     |
| Rocksdb_bloom_filter_useful         | 0     |
+-----------------------------------+-------+
```

Other useful variables are:

- `rocksdb_force_flush_memtable_now` - bloom filter is only used when reading data from disk. If you are doing testing, flush the data to disk first.
- `rocksdb_skip_bloom_filter_on_read` - skip using the bloom filter (default is FALSE).

This page is licensed: CC BY-SA / Gnu FDL

MyRocks CHECK TABLE

MyRocks supports the CHECK TABLE command.

The command will do a number of checks to verify that the table data is self-consistent.

The details about the errors are printed into the error log.
If log_warnings > 2, the error log will also have some informational messages which can help with troubleshooting.

Besides this, RocksDB has its own (low-level) log in `#rocksdb/LOG` file.

This page is licensed: CC BY-SA / Gnu FDL

MyRocks Data Compression

MyRocks supports several compression algorithms.

Supported Compression Algorithms

Supported compression algorithms can be checked like so:

```
SHOW variables LIKE 'rocksdb%compress%';
+-----------------------------------+----------------------------------+
| Variable_name                     | Value                            |
+-----------------------------------+----------------------------------+
| rocksdb_supported_compression_types | Snappy,Zlib,LZ4,LZ4HC,ZSTDNotFinal |
+-----------------------------------+----------------------------------+
```

Another way to make the check is to look into `#rocksdb/LOG` file in the data directory. It should have lines like:

```
2019/04/12-14:08:23.869919 7f839188b540 Compression algorithms supported:
2019/04/12-14:08:23.869920 7f839188b540         kZSTDNotFinalCompression supported: 1
2019/04/12-14:08:23.869922 7f839188b540         kZSTD supported: 1
2019/04/12-14:08:23.869923 7f839188b540         kXpressCompression supported: 0
2019/04/12-14:08:23.869924 7f839188b540         kLZ4HCCompression supported: 1
2019/04/12-14:08:23.869924 7f839188b540         kLZ4Compression supported: 1
2019/04/12-14:08:23.869925 7f839188b540         kBZip2Compression supported: 0
2019/04/12-14:08:23.869926 7f839188b540         kZlibCompression supported: 1
2019/04/12-14:08:23.869927 7f839188b540         kSnappyCompression supported: 1
```

Compression Settings

Compression is set on a per-Column Family basis. See MyRocks Column Families.

Checking Compression Settings

To check current compression settings for a column family one can use a query like so:

```
SELECT * FROM information_schema.rocksdb_cf_options
WHERE option_type LIKE '%ompression%' AND cf_name='DEFAULT';
```

The output are like:

```
+---------+---------------------------------------+---------------------------+
| CF_NAME | OPTION_TYPE                           | VALUE                     |
+---------+---------------------------------------+---------------------------+
| default | COMPRESSION_TYPE                      | kSnappyCompression        |
| default | COMPRESSION_PER_LEVEL                 | NUL                       |
| default | COMPRESSION_OPTS                      | -14:32767:0               |
| default | BOTTOMMOST_COMPRESSION                | kDisableCompressionOption |
| default | TABLE_FACTORY::VERIFY_COMPRESSION     | 0                         |
| default | TABLE_FACTORY::ENABLE_INDEX_COMPRESSION | 1                       |
+---------+---------------------------------------+---------------------------+
```

Current column family settings are used for the new SST files.

Modifying Compression Settings

Compression settings are not dynamic parameters, one cannot change them by setting rocksdb_update_cf_options.

The procedure to change compression settings is as follows:

- Edit `my.cnf` to set rocksdb_override_cf_options.

Example:

```
rocksdb-override-cf-options='cf1={compression=kZSTD;bottommost_compression=kZSTD;}'
```

- Restart the server.

The data will not be re-compressed immediately. However, all new SST files will use the new compression settings, so as data gets inserted/updated the column family will gradually start using the new option.

Caveat: Syntax Errors

Please note that `rocksdb-override-cf-options` syntax is quite strict. Any typos will result in the parse error, and MyRocks plugin will not be loaded. Depending on your configuration, the server may still start. If it does start, you can use this command to check if the plugin is loaded:

```
SELECT * FROM information_schema.plugins WHERE plugin_name='ROCKSDB'
```

(note that you need the “ROCKSDB” plugin. Other auxiliary plugins like “ROCKSDB_TRX” might still get loaded).

Another way is to detect the error is check the error log. When option parsing fails, it will contain messages like so:

```
2019-04-16 11:07:57 140283675678016 [Warning] Invalid cf config for cf1 in override options (options:
cf1={compression=kLZ4Compression;bottommost_compression=kZSTDCompression;})
2019-04-16 11:07:57 140283675678016 [ERROR] RocksDB: Failed to initialize CF options map.
2019-04-16 11:07:57 140283675678016 [ERROR] Plugin 'ROCKSDB' init function returned error.
2019-04-16 11:07:57 140283675678016 [ERROR] Plugin 'ROCKSDB' registration as a STORAGE ENGINE failed.
```

Checking How the Data is Compressed

A query to check what compression is used in the SST files that store the data for a given table (test.t1):

```
SELECT
  SP.sst_name, SP.compression_algo
FROM
  information_schema.rocksdb_sst_props SP,
  information_schema.rocksdb_ddl D,
  information_schema.rocksdb_index_file_map IFM
WHERE
  D.table_schema='test' AND D.table_name='t1' AND
  D.index_number= IFM.index_number AND
  IFM.sst_name=SP.sst_name;
```

Example output:

```
+------------+------------------+
| sst_name   | compression_algo |
+------------+------------------+
| 000028.sst | Snappy           |
| 000028.sst | Snappy           |
| 000026.sst | Snappy           |
| 000026.sst | Snappy           |
+------------+------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

MyRocks Group Commit with Binary log

MyRocks supports group commit with the binary log (MDEV-11934).

Counter Values to Watch

(The following is only necessary if you are studying MyRocks internals)

MariaDB's group commit counters are:

Binlog_commits - how many transactions were written to the binary log

Binlog_group_commits - how many group commits happened. (e.g. if each group had two transactions, this is twice as small as Binlog_commits)

On the RocksDB side, there is one relevant counter:Rocksdb_wal_synced - How many times RocksDB's WAL file was synced. (TODO: this is after group commit happened, right?)

On the Value of rocksdb_wal_group_syncs

FB/MySQL-5.6 has a rocksdb_wal_group_syncs counter (The counter is provided by MyRocks, it is not a view of a RocksDB counter). It is increased in rocksdb_flush_wal() when doing the rdb->FlushWAL() call.

rocksdb_flush_wal() is called by MySQL's Group Commit when it wants to make the effect of several rocksdb_prepare() calls persistent.

So, the value of rocksdb_wal_group_syncs in FB/MySQL-5.6 is similar to Binlog_group_commits in MariaDB.

MariaDB doesn't have that call, each rocksdb_prepare() call takes care of being persistent on its own.

Because of that, rocksdb_wal_group_syncs is zero for MariaDB. (Currently, it is only incremented when the binlog is rotated).

Examples

So for a workload with concurrency=50, n_queries=10K, one gets

- Binlog_commits=10K
- Binlog_group_commits=794
- Rocksdb_wal_synced=8362

This is on a RAM disk

For a workload with concurrency=50, n_queries=10K, rotating laptop hdd, one gets

- Binlog_commits= 10K
- Binlog_group_commits=1403
- Rocksdb_wal_synced=400

The test took 38 seconds, Number of syncs was 1400+400=1800, which gives 45 syncs/sec which looks normal for this slow rotating desktop hdd.

Note that the WAL was synced fewer times than there were binlog commit groups (?)

This page is licensed: CC BY-SA / Gnu FDL

MyRocks Index-Only Scans

This article is about MyRocks and index-only scans on secondary indexes. It applies to MariaDB's MyRocks, Facebook's MyRocks, and other variants.

Secondary Keys Only

The primary key in MyRocks is always the clustered key, that is, the index record is THE table record and so it's not possible to do "index only" because there isn't anything that is not in the primary key's (Key,Value) pair.

Secondary keys may or may not support index-only scans, depending on the datatypes of the columns that the query is trying to read.

Background: Mem-Comparable Keys

MyRocks indexes store “mem-comparable keys” (that is, the key values are compared with `memcmp`). For some datatypes, it is easily possible to convert between the column value and its mem-comparable form, while for others the conversion is one-way.

For example, in case-insensitive collations capital and regular letters are considered identical, i.e. ‘c’ =‘C’. For some datatypes, MyRocks stores some extra data which allows it to restore the original value back. (For the `latin1_general_ci` collation and character ‘c’, for example, it will store one bit which says whether the original value was a small ‘c’ or a capital letter ‘C’). This doesn’t work for all datatypes, though.

Index-Only Support for Various Datatypes

Index-only scans are supported for numeric and date/time datatypes. For CHAR and VAR[CHAR], it depends on which collation is used, see below for details.

Index-only scans are currently not supported for less frequently used datatypes, like

- BIT(n)
- SET(…)
- ENUM(…)
  It is actually possible to add support for those, feel free to write a patch or at least make a case why a particular datatype is important

Index-Only Support for Various Collations

As far as Index-only support is concerned, MyRocks distinguishes three kinds of collations:

1. Binary (Reversible) Collations

These are `binary`, `latin1_bin`, and `utf8_bin`.

For these collations, it is possible to convert a value back from its mem-comparable form. Hence, one can restore the original value back from its index record, and index-only scans are supported.

2. Restorable Collations

These are collations where one can store some extra information which helps to restore the original value.

Criteria (from storage/rocksdb/rdb_datadic.cc, rdb_is_collation_supported()) are:

- The charset should use 1-byte characters (so, unicode-based collations are not included)
- strxfrm(1 byte) = {one 1-byte weight value always}
- no binary sorting
- PAD attribute

The examples are: `latin1_general_ci`, `latin1_general_cs`, `latin1_swedish_ci`, etc.

Index-only scans are supported for these collations.

3. All Other Collations

For these collations, there is no known way to restore the value from its mem-comparable form, and so index-only scans are not supported.

MyRocks needs to fetch the clustered PK record to get the field value.

Covering Secondary Key Lookups for VARCHARs

TODO: there is also this optimization:

303f349c95848e92b5b27b44f0e57194100eb0997e7

document it.

This page is licensed: CC BY-SA / Gnu FDL

MyRocks and Replication

Details about how MyRocks works with replication.

MyRocks and Statement-Based Replication

Statement-based replication (SBR) works as follows: SQL statements are executed on the master (possibly concurrently). They are written into the binlog (this fixes their ordering, “a serialization”). The slave then reads the binlog and executes the statements in their binlog order.

In order to prevent data drift, serial execution of statements on the slave must have the same effect as concurrent execution of these statements on the master. In other words, transaction isolation on the master must be close to `SERIALIZABLE` transaction isolation level (This is not a strict mathematical proof but shows the idea).

InnoDB achieves this by (almost) supporting `SERIALIZABLE` transactional isolation level. It does so by supporting “Gap Locks”. MyRocks doesn’t support `SERIALIZABLE` isolation, and it doesn’t support gap locks.

Because of that, generally one cannot use MyRocks and statement-based replication.

Updating a MyRocks table while having SBR on, will result in an error as follow:

```
ERROR 4056 (HY000): Can't execute updates on master with binlog_format != ROW.
```

Can One Still Use SBR with MyRocks?

Yes. In many cases, database applications run a restricted set of SQL statements, and it’s possible to prove that lack of Gap Lock support is not a problem and data skew will not occur.

In that case, one can set `@@rocksdb_unsafe_for_binlog=1` and MyRocks will work with SBR. The user is however responsible for making sure their queries are not causing a data skew.

Differences From Upstream MyRocks

MyRocks upstream (that is, Facebook’s MySQL branch) has a number of unique replication enhancements. These are available in upstream’s version of MyRocks but not in MariaDB’s version of MyRocks.

- Read-Free Replication (see Read-Free-Replication) TODO
- `<<unique_check_lag_threshold>>`. This is FB/MySQL-5.6 feature where unique checks are disabled if replication lag exceeds a certain threshold.
- `<<slave_gtid_info=OPTIMIZED>>`. This is said to be:

```
<<quote>>
"Whether SQL threads update mysql.slave_gtid_info table. If this value "
"is OPTIMIZED, updating the table is done inside storage engines to "
"avoid MySQL layer's performance overhead",
<</quote>>
```

This page is licensed: CC BY-SA / Gnu FDL

MyRocks START TRANSACTION WITH CONSISTENT SNAPSHOT

FB/MySQL has added new syntax:

```
START TRANSACTION WITH CONSISTENT ROCKSDB|INNODB SNAPSHOT;
```

The statement returns the binlog coordinates pointing at the snapshot.

MariaDB (and Percona Server) support extension to the regular

```
START TRANSACTION WITH CONSISTENT SNAPSHOT;
```

syntax as documented in Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT.

After issuing the statement, one can examine the binlog_snapshot_file and binlog_snapshot_position status variables to see the binlog position that corresponds to the snapshot.

See Also

- START TRANSACTION
- Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT

This page is licensed: CC BY-SA / Gnu FDL

MyRocks Column Families

MyRocks stores data in column families. These are similar to tablespaces.
By default, the data is stored in the `default` column family.

One can specify which column family the data goes to by using index comments:

```
INDEX index_name(col1, col2, ...) COMMENT 'column_family_name'
```

If the column name starts with `rev:`, the column family is reverse-ordered.

Reasons for Column Families

Storage parameters like

- Bloom filter settings
- Compression settings
- Whether the data is stored in reverse order

are specified on a per-column family basis.

Creating a Column Family

When creating a table or index, you can specify the name of the column family for it. If the column family doesn't exist, it are automatically created.

Dropping a Column Family

There is currently no way to drop a column family.
RocksDB supports this internally but MyRocks doesn't provide any way to do it.

Setting Column Family Parameters

Use these variables:

- rocksdb_default_cf_options - a my.cnf parameter specifying default options for all column families.
- rocksdb_override_cf_options - a my.cnf parameter specifying per-column family option overrides.
- rocksdb_update_cf_options - a dynamically-settable variable which allows to change parameters online. Not all parameters can be changed.

rocksdb_override_cf_options

This parameter allows one to override column family options for specific column families.
Here is an example of how to set option1=value1 and option2=value for column family cf1, and option3=value3 for column family cf3:

```
rocksdb_override_cf_options='cf1={option1=value1;option2=value2};cf2={option3=value3}'
```

One can check the contents of `INFORMATION_SCHEMA.ROCKSDB_CF_OPTIONS` to see what options are available.

Options that are frequently configured are:

- Data compression. See myrocks-and-data-compression.
- Bloom Filters. See myrocks-and-bloom-filters.

Examining Column Family Parameters

See the INFORMATION_SCHEMA.ROCKSDB_CF_OPTIONS table.

This page is licensed: CC BY-SA / Gnu FDL

MyRocks Performance Troubleshooting

MyRocks exposes its performance metrics through several interfaces:

- Status variables
- `SHOW ENGINE ROCKSDB STATUS`
- RocksDB's perf context

the contents slightly overlap, but each source has its own unique information, so be sure to check all three.

Status Variables

Check the output of

```
SHOW STATUS LIKE 'Rocksdb%'
```

See MyRocks Status Variables for more information.

SHOW ENGINE ROCKSDB STATUS

This produces a lot of information.

One particularly interesting part is compaction statistics. It shows the amount of data on each SST level and other details:

```
*************************** 4. row ***************************
  Type: CF_COMPACTION
  Name: default
Status:
** Compaction Stats [default] **
Level    Files   Size     Score Read(GB)  Rn(GB) Rnp1(GB) Write(GB) Wnew(GB) Moved(GB) W-Amp Rd(MB/s)
Wr(MB/s) Comp(sec) Comp(cnt) Avg(sec) KeyIn KeyDrop
------------------------------------------------------------------------------------------------------------
------------------------------------------------
  L0      3/0   30.16 MB   1.0      0.0     0.0      0.0      11.9     11.9       0.0   1.0      0.0
76.6       159       632    0.251       0      0
  L1      5/0   247.54 MB   1.0      0.7     0.2      0.5       0.5      0.0      11.6   2.6     58.5
44.1        12         4    2.926     30M    10M
  L2    112/0    2.41 GB   1.0      0.6     0.0      0.6       0.5     -0.1      11.4  43.4     55.2
45.9        11         1   10.827     21M  3588K
  L3    466/0    8.91 GB   0.4      0.0     0.0      0.0       0.0      0.0       8.9   0.0      0.0
0.0         0         0    0.000       0      0
 Sum    586/0   11.59 GB   0.0      1.3     0.2      1.0      12.8     11.8      32.0   1.1      7.1
72.6       181       637    0.284     52M    13M
 Int      0/0    0.00 KB   0.0      0.9     0.1      0.8       0.8      0.0       0.1  20.5     48.4
45.3        19         6    3.133     33M  3588K
```

Performance Context

RocksDB has an internal mechanism called “perf context”. The counter values are exposed through two tables:

- INFORMATION_SCHEMA.ROCKSDB_PERF_CONTEXT_GLOBAL - global counters
- INFORMATION_SCHEMA.ROCKSDB_PERF_CONTEXT - Per-table/partition counters

By default statistics are NOT collected. One needs to set rocksdb_perf_context_level to some value (e.g. 3) to enable collection.

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Statistics in MyRocks

This article describes how MyRocks storage engine provides statistics to the query optimizer.

There are three kinds of statistics:

- Table statistics (number of rows in the table, average row size)
- Index cardinality (how distinct values are in the index)
- records-in-range estimates (how many rows are in a certain range “const1 < tbl.key < const2”.

How MyRocks computes statistics

MyRocks (actually RocksDB) uses LSM files which are written once and never updated. When an LSM file is written, MyRocks will compute index cardinalities and number-of-rows for the data in the file. (The file generally has rows, index records and/or tombstones for multiple tables/indexes).

For performance reasons, statistics are computed based on a fraction of rows in the LSM file. The percentage of rows used is controlled by rocksdb_table_stats_sampling_pct; the default value is 10%.

Before the data is dumped into LSM file, it is stored in the MemTable. MemTable doesn’t allow computing index cardinalities, but it can provide an approximate number of rows in the table. Use of MemTable data for statistics is controlled by rocksdb_force_compute_memtable_stats; the default value is `ON`.

Are index statistics predictable?

Those who create/run MTR tests, need to know whether EXPLAIN output is deterministic.
For MyRocks tables, the answer is NO (just like for InnoDB).

Statistics are computed using sampling and GetApproximateMemTableStats() which means that the #rows column in the EXPLAIN output may vary slightly.

Records-in-range estimates

MyRocks uses RocksDB's GetApproximateSizes() call to produce an estimate for the number of rows in the certain range. The data in MemTable is also taken into account by issuing a GetApproximateMemTableStats call.

ANALYZE TABLE

ANALYZE TABLE will possibly flush the MemTable (depending on the rocksdb_flush_memtable_on_analyze and rocksdb_pause_background_work settings).

After that, it will re-read statistics from the SST files and re-compute the summary numbers
(TODO: and if the data was already on disk, the result should not be different from the one we had before ANALYZE?)

Debugging helper variables

There are a few variables that will cause MyRocks to report certain pre-defined estimate numbers to the optimizer:

- @@rocksdb_records_in_range - if not 0, report that any range has this many rows
- @@rocksdb_force_index_records_in_range - if not 0, and FORCE INDEX hint is used, report that any range has this many rows.
- @@rocksdb_debug_optimizer_n_rows - if not 0, report that any MyRocks table has this many rows.

This page is licensed: CC BY-SA / Gnu FDL

MyRocks Transactional Isolation

- MyRocks uses snapshot isolation.
- Support do `READ-COMMITTED` and `REPEATABLE-READ` .
- `SERIALIZABLE` is not supported.
  - There is no "gap locking" which makes Statement Based Replication unsafe (see MyRocks and Replication).

This page is licensed: CC BY-SA / Gnu FDL

MyRocks Status Variables

This page documents status variables related to the MyRocks storage engine. See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

See also the Full list of MariaDB options, system and status variables.

`Rocksdb_block_cache_add`

- Description: Number of blocks added to the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cache_add_failures`

- Description: Number of failures when adding blocks to Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_bytes_read`

- Description: Bytes read from Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_bytes_write`

- Description: Bytes written to Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_data_add`

- Description: Number of data blocks added to the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_data_bytes_insert`

- Description: Bytes added to the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_data_hit`

- Description: Number of hits when accessing the data block from the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cache_data_miss`

- Description: Number of misses when accessing the data block from the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cache_filter_add`

- Description: Number of bloom filter blocks added to the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_filter_bytes_evict`

- Description: Bytes of bloom filter blocks evicted from the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_filter_bytes_insert`

- Description: Bytes of bloom filter blocks added to the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_filter_hit`

- Description: Number of hits when accessing the filter block from the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cache_filter_miss`

- Description: Number of misses when accessing the filter block from the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cache_hit`

- Description: Total number of hits for the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cache_index_add`

- Description: Number of index blocks added to Block Cache index.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_index_bytes_evict`

- Description: Bytes of index blocks evicted from the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_index_bytes_insert`

- Description: Bytes of index blocks added to the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_block_cache_index_hit`

- Description: Number of hits for the Block Cache index.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cache_index_miss`

- Description: Number of misses for the Block Cache index.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cache_miss`

- Description: Total number of misses for the Block Cache.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cachecompressed_hit`

- Description: Number of hits for the compressed Block Cache.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_block_cachecompressed_miss`

- Description: Number of misses for the compressed Block Cache.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_bloom_filter_full_positive`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.18, MariaDB 10.3.10

`Rocksdb_bloom_filter_full_true_positive`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.18, MariaDB 10.3.10

`Rocksdb_bloom_filter_prefix_checked`

- Description: Number of times the Bloom Filter checked before creating an iterator on a file.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_bloom_filter_prefix_useful`

- Description: Number of times the Bloom Filter check used to avoid creating an iterator on a file.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_bloom_filter_useful`

- Description: Number of times the Bloom Filter used instead of reading form file.
- Scope: Global, Session

- Data Type: `numeric`

`Rocksdb_bytes_read`

- Description: Total number of uncompressed bytes read from memtables, cache or table files.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_bytes_written`

- Description: Total number of uncompressed bytes written.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_compact_read_bytes`

- Description: Number of bytes read during compaction.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_compact_write_bytes`

- Description: Number of bytes written during compaction.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_compaction_key_drop_new`

- Description: Number of keys dropped during compaction due their being overwritten by new values.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_compaction_key_drop_obsolete`

- Description: Number of keys dropped during compaction due to their being obsolete.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_compaction_key_drop_user`

- Description: Number of keys dropped during compaction due to user compaction.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_covered_secondary_key_lookups`

- Description: Incremented when avoiding reading a record via a keyread. This indicates lookups that were performed via a secondary index containing a field that is only a prefix of the VARCHAR column, and that could return all requested fields directly from the secondary index.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_flush_write_bytes`

- Description: Number of bytes written during flush.

- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_get_hit_l0`

- Description: Number of times reads got data from the L0 compaction layer.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_get_hit_l1`

- Description: Number of times reads got data from the L1 compaction layer.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_get_hit_l2_and_up`

- Description: Number of times reads got data from the L2 and up compaction layer.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_getupdatessince_calls`

- Description: Number of calls to the `GetUpdatesSince` function. You may find this useful when monitoring refreshes of the transaction log.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_iter_bytes_read`

- Description: Total uncompressed bytes read from an iterator, including the size of both key and value.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_l0_num_files_stall_micros`

- Description: Shows how long in microseconds throttled due to too many files in L0.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.3.1, MariaDB 10.2.8

`Rocksdb_l0_slowdown_micros`

- Description: Total time spent waiting in microseconds while performing L0-L1 compactions.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.3.1, MariaDB 10.2.8

`Rocksdb_manual_compactions_processed`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.18, MariaDB 10.3.10

`Rocksdb_manual_compactions_running`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.18, MariaDB 10.3.10

`Rocksdb_memtable_compaction_micros`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.3.1, MariaDB 10.2.8

`Rocksdb_memtable_hit`

- Description: Number of memtable hits.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_memtable_miss`

- Description: Number of memtable misses.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_memtable_total`

- Description: Memory used, in bytes, of all memtables.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_memtable_unflushed`

- Description: Memory used, in bytes, of all unflushed memtables.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_no_file_closes`

- Description: Number of times files were closed.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_no_file_errors`

- Description: Number of errors encountered while trying to read data from an SST file.

- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_no_file_opens`

- Description: Number of times files were opened.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_num_iterators`

- Description: Number of iterators currently open.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_block_not_compressed`

- Description: Number of uncompressed blocks.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_db_next`

- Description: Number of `next` calls.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_number_db_next_found`

- Description: Number of `next` calls that returned data.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_number_db_prev`

- Description: Number of `prev` calls.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_number_db_prev_found`

- Description: Number of `prev` calls that returned data.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_number_db_seek`

- Description: Number of `seek` calls.
- Scope: Global, Session

- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_number_db_seek_found`

- Description: Number of `seek` calls that returned data.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_number_deletes_filtered`

- Description: Number of deleted records were not written to storage due to a nonexistent key.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_keys_read`

- Description: Number of keys have been read.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_keys_updated`

- Description: Number of keys have been updated.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_keys_written`

- Description: Number of keys have been written.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_merge_failures`

- Description: Number of failures encountered while performing merge operator actions.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_multiget_bytes_read`

- Description: Number of bytes read during RocksDB `MultiGet()` calls.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_multiget_get`

- Description: Number of RocksDB `MultiGet()` requests made.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_multiget_keys_read`

- Description: Number of keys read through RocksDB `MultiGet()` calls.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_reseeks_iteration`

- Description: Number of reseeks that have occurred inside an iteration that skipped over a large number of keys with the same user key.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_sst_entry_delete`

- Description: Number of delete markers written.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_sst_entry_merge`

- Description: Number of merge keys written.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_sst_entry_other`

- Description: Number of keys written that are not delete, merge or put keys.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_sst_entry_put`

- Description: Number of put keys written.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_sst_entry_singledelete`

- Description: Number of single-delete keys written.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_superversion_acquires`

- Description: Number of times the superversion structure acquired. This is useful when tracking files for the database.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_number_superversion_cleanups`

- Description: Number of times the superversion structure performed cleanups.
- Scope: Global, Session

- Data Type: `numeric`

`Rocksdb_number_superversion_releases`

- Description: Number of times the superversion structure released.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_queries_point`

- Description: Number of single-row queries.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_queries_range`

- Description: Number of multi-row queries.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_row_lock_deadlocks`

- Description: Number of deadlocks.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_row_lock_wait_timeouts`

- Description: Number of row lock wait timeouts.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_rows_deleted`

- Description: Number of rows deleted.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_rows_deleted_blind`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_rows_expired`

- Description: Number of expired rows.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_rows_filtered`

- Description: Number of TTL filtered rows.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.15, MariaDB 10.3.7

`Rocksdb_rows_inserted`

- Description: Number of rows inserted.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_rows_read`

- Description: Number of rows read.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_rows_updated`

- Description: Number of rows updated.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_snapshot_conflict_errors`

- Description: Number of snapshot conflict errors that have occurred during transactions that forced a rollback.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_stall_l0_file_count_limit_slowdowns`

- Description: Write slowdowns due to L0 being near to full.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_stall_l0_file_count_limit_stops`

- Description: Write stops due to L0 being to full.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_stall_locked_l0_file_count_limit_slowdowns`

- Description: Write slowdowns due to L0 being near to full and L0 compaction in progress.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_stall_locked_l0_file_count_limit_stops`

- Description: Write stops due to L0 being full and L0 compaction in progress.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_stall_memtable_limit_slowdowns`

- Description: Write slowdowns due to approaching maximum permitted number of memtables.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.10, MariaDB 10.3.3

`Rocksdb_stall_memtable_limit_stops`

- Description: * Description: Write stops due to reaching maximum permitted number of memtables.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.2.10, MariaDB 10.3.3

`Rocksdb_stall_micros`

- Description: Time in microseconds that the writer had to wait for the compaction or flush to complete.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_stall_pending_compaction_limit_slowdowns`

- Description: Write slowdowns due to nearing the limit for the maximum number of pending compaction bytes.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_stall_pending_compaction_limit_stops`

- Description: Write stops due to reaching the limit for the maximum number of pending compaction bytes.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_stall_total_slowdowns`

- Description: Total number of write slowdowns.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_stall_total_stops`

- Description: Total number of write stops.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_system_rows_deleted`

- Description: Number of rows deleted from system tables.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_system_rows_inserted`

- Description: Number of rows inserted into system tables.
- Scope: Global, Session

- Data Type: `numeric`

`Rocksdb_system_rows_read`

- Description: Number of rows read from system tables.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_system_rows_updated`

- Description: Number of rows updated for system tables.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_wal_bytes`

- Description: Number of bytes written to WAL.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_wal_group_syncs`

- Description: Number of group commit WAL file syncs have occurred. This is provided by MyRocks and is not a view of a RocksDB counter. Increased in `rocksdb_flush_wal()` when doing the `rdb->FlushWAL()` call.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_wal_synced`

- Description: Number of syncs made on RocksDB WAL file.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_write_other`

- Description: Number of writes processed by a thread other than the requesting thread.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_write_self`

- Description: Number of writes processed by requesting thread.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_write_timedout`

- Description: Number of writes that timed out.
- Scope: Global, Session
- Data Type: `numeric`

`Rocksdb_write_wal`

- Description: Number of write calls that requested WAL.

- Scope: Global, Session
- Data Type: `numeric`

This page is licensed: CC BY-SA / Gnu FDL

MyRocks System Variables

This page documents system variables related to the MyRocks storage engine. See Server System Variables for a complete list of system variables and instructions on setting them.

See also the Full list of MariaDB options, system and status variables.

`rocksdb_access_hint_on_compaction_start`

- Description: DBOptions::access_hint_on_compaction_start for RocksDB. Specifies the file access pattern, applied to all input files, once a compaction starts.
- Command line: `--rocksdb-access-hint-on-compaction-start=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `3`

`rocksdb_advise_random_on_open`

- Description: DBOptions::advise_random_on_open for RocksDB.
- Command line: `--rocksdb-advise-random-on-open={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_allow_concurrent_memtable_write`

- Description: DBOptions::allow_concurrent_memtable_write for RocksDB.
- Command line: `--rocksdb-allow-concurrent-memtable-write={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_allow_mmap_reads`

- Description: DBOptions::allow_mmap_reads for RocksDB
- Command line: `--rocksdb-allow-mmap-reads={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_allow_mmap_writes`

- Description: DBOptions::allow_mmap_writes for RocksDB

- Command line: `--rocksdb-allow-mmap-writes={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_allow_to_start_after_corruption`

- Description: Allow server still to start successfully even if RocksDB corruption is detected.
- Command line: `--rocksdb-allow-to-start-after-corruption={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_background_sync`

- Description: Turns on background syncs for RocksDB
- Command line: `--rocksdb-background-sync={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_base_background_compactions`

- Description: DBOptions::base_background_compactions for RocksDB
- Command line: `--rocksdb-base-background-compactions=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1`
- Range: `-1` to `64`
- Removed: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_blind_delete_primary_key`

- Description: Deleting rows by primary key lookup, without reading rows (Blind Deletes). Blind delete is disabled if the table has secondary key.
- Command line: `--rocksdb-blind-delete-primary-key={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_block_cache_size`

- Description: Block_cache size for RocksDB (block size 1024)
- Command line: `--rocksdb-block-cache-size=#`

- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `536870912`
- Range: `1024` to `9223372036854775807`

To see the statistics of block cache usage, check `SHOW ENGINE ROCKSDB STATUS` output (search for lines starting with `rocksdb.block.cache`).

One can check the size of data of the block cache in `DB_BLOCK_CACHE_USAGE` column of the `INFORMATION_SCHEMA.ROCKSDB_DBSTATS` table.

`rocksdb_block_restart_interval`

- Description: BlockBasedTableOptions::block_restart_interval for RocksDB
- Command line: `--rocksdb-block-restart-interval=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `16`
- Range: `1` to `2147483647`

`rocksdb_block_size`

- Description: BlockBasedTableOptions::block_size for RocksDB
- Command line: `--rocksdb-block-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `4096`
- Range: `1` to `18446744073709551615`

`rocksdb_block_size_deviation`

- Description: BlockBasedTableOptions::block_size_deviation for RocksDB
- Command line: `--rocksdb-block-size-deviation=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `10`
- Range: `0` to `2147483647`

`rocksdb_bulk_load`

- Description: Use bulk-load mode for inserts. This disables unique_checks and enables rocksdb_commit_in_the_middle.
- Command line: `--rocksdb-bulk-load={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_bulk_load_allow_sk`

- Description: Allow bulk loading of sk keys during bulk-load. Can be changed only when bulk load is disabled.
- Command line: `--rocksdb-bulk-load_allow_sk={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_bulk_load_allow_unsorted`

- Description: Allow unsorted input during bulk-load. Can be changed only when bulk load is disabled.
- Command line: `--rocksdb-bulk-load_allow_unsorted={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_bulk_load_size`

- Description: Maximum number of records in a batch for bulk-load mode.
- Command line: `--rocksdb-bulk-load-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1000`
- Range: `1` to `1073741824`

`rocksdb_bytes_per_sync`

- Description: DBOptions::bytes_per_sync for RocksDB.
- Command line: `--rocksdb-bytes-per-sync=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`rocksdb_cache_dump`

- Description: Include RocksDB block cache content in core dump.
- Command line: `--rocksdb-cache-dump={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_cache_high_pri_pool_ratio`

- Description: Specify the size of block cache high-pri pool.
- Command line: `--rocksdb-cache-high-pri-pool-ratio=#`

- Scope: Global
- Dynamic: Yes
- Data Type: `double`
- Default Value: `0.000000`
- Range: `0` to `1`

`rocksdb_cache_index_and_filter_blocks`

- Description: BlockBasedTableOptions::cache_index_and_filter_blocks for RocksDB.
- Command line: `--rocksdb-cache-index-and-filter-blocks={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_cache_index_and_filter_with_high_priority`

- Description: cache_index_and_filter_blocks_with_high_priority for RocksDB.
- Command line: `--rocksdb-cache-index-and-filter-with-high-priority={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_checksums_pct`

- Description: Percentage of rows to be checksummed.
- Command line: `--rocksdb-checksums-pct=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `0` to `100`

`rocksdb_collect_sst_properties`

- Description: Enables collecting SST file properties on each flush.
- Command line: `--rocksdb-collect-sst-properties={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_commit_in_the_middle`

- Description: Commit rows implicitly every rocksdb_bulk_load_size, on bulk load/insert, update and delete.
- Command line: `--rocksdb-commit-in-the-middle={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`

- Default Value: `OFF`

`rocksdb_commit_time_batch_for_recovery`

- Description: TransactionOptions::commit_time_batch_for_recovery for RocksDB.
- Command line: `--rocksdb-commit-time-batch-for-recovery={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_compact_cf`

- Description: Compact column family.
- Command line: `--rocksdb-compact-cf=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: (Empty)

`rocksdb_compaction_readahead_size`

- Description: DBOptions::compaction_readahead_size for RocksDB.
- Command line: `--rocksdb-compaction-readahead-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`rocksdb_compaction_sequential_deletes`

- Description: RocksDB will trigger compaction for the file if it has more than this number sequential deletes per window.
- Command line: `--rocksdb-compaction-sequential-deletes=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `2000000`

`rocksdb_compaction_sequential_deletes_count_sd`

- Description: Counting SingleDelete as rocksdb_compaction_sequential_deletes.
- Command line: `--rocksdb-compaction-sequential-deletes-count-sd={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_compaction_sequential_deletes_file_size`

- Description: Minimum file size required for compaction_sequential_deletes.
- Command line: `--rocksdb-compaction-sequential-deletes-file-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `-1` to `9223372036854775807`

`rocksdb_compaction_sequential_deletes_window`

- Description: Size of the window for counting rocksdb_compaction_sequential_deletes.
- Command line: `--rocksdb-compaction-sequential-deletes-window=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `2000000`

`rocksdb_concurrent_prepare`

- Description: DBOptions::concurrent_prepare for RocksDB.
- Command line: `--rocksdb-coconcurrent-prepare={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `1`
- Removed: MariaDB 10.3.7, MariaDB 10.2.15

`rocksdb_create_checkpoint`

- Description: Checkpoint directory.
- Command line: `--rocksdb-create-checkpoint=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: (Empty)

`rocksdb_create_if_missing`

- Description: DBOptions::create_if_missing for RocksDB.
- Command line: `--rocksdb-create-if-missing={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_create_missing_column_families`

- Description: DBOptions::create_missing_column_families for RocksDB.

- Command line: `--rocksdb-create-missing-column-families={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_datadir`

- Description: RocksDB data directory.
- Command line: `--rocksdb-datadir[=value]`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `./#rocksdb`

`rocksdb_db_write_buffer_size`

- Description: DBOptions::db_write_buffer_size for RocksDB.
- Command line: `--rocksdb-db-write-buffer-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`rocksdb_deadlock_detect`

- Description: Enables deadlock detection.
- Command line: `--rocksdb-deadlock-detect={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_deadlock_detect_depth`

- Description: Number of transactions deadlock detection will traverse through before assuming deadlock.
- Command line: `--rocksdb-deadlock-detect-depth=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `50`
- Range: `2` to `18446744073709551615`

`rocksdb_debug_manual_compaction_delay`

- Description: For debugging purposes only. Sleeping specified seconds for simulating long running compactions.
- Command line: `--rocksdb-debug_manual_compaction_delay=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4294967295`

`rocksdb_debug_optimizer_no_zero_cardinality`

- Description: If cardinality is zero, override it with some value.
- Command line: `--rocksdb-debug-optimizer-no-zero-cardinality={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_debug_ttl_ignore_pk`

- Description: For debugging purposes only. If true, compaction filtering will not occur on PK TTL data. This variable is a no-op in non-debug builds.
- Command line: `--rocksdb-debug-ttl-ignore-pk={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_debug_ttl_read_filter_ts`

- Description: For debugging purposes only. Overrides the TTL read filtering time to time + debug_ttl_read_filter_ts. A value of 0 denotes that the variable is not set. This variable is a no-op in non-debug builds.
- Command line: `--rocksdb-debug-ttl-read-filter-ts=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `-3600` to `3600`

`rocksdb_debug_ttl_rec_ts`

- Description: For debugging purposes only. Overrides the TTL of records to now() + debug_ttl_rec_ts. The value can be +/- to simulate a record inserted in the past vs a record inserted in the ‘future’. A value of 0 denotes that the variable is not set. This variable is a no-op in non-debug builds.
- Command line: `--rocksdb-debug-ttl-read-filter-ts=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `-3600` to `3600`

`rocksdb_debug_ttl_snapshot_ts`

- Description: For debugging purposes only. Sets the snapshot during compaction to now() + debug_set_ttl_snapshot_ts. The value can be positive or negative to simulate a snapshot in the past vs a

snapshot created in the 'future'. A value of 0 denotes that the variable is not set. This variable is a no-op in non-debug builds.

- Command line: `--rocksdb-debug-ttl-snapshot-ts=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `-3600` to `3600`
- Introduced: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_default_cf_options`

- Description: Default cf options for RocksDB.
- Command line: `--rocksdb-default-cf-options=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: (Empty)

`rocksdb_delayed_write_rate`

- Description: DBOptions::delayed_write_rate.
- Command line: `--rocksdb-delayed-write-rate=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0` (Previously `16777216`)
- Range: `0` to `18446744073709551615`

`rocksdb_delete_cf`

- Description: Delete column family.
- Command line: `--rocksdb-delete-cf=val`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: (Empty string)
- Introduced: MariaDB 10.4.7, MariaDB 10.3.17, MariaDB 10.2.26

`rocksdb_delete_obsolete_files_period_micros`

- Description: DBOptions::delete_obsolete_files_period_micros for RocksDB.
- Command line: `--rocksdb-delete-obsolete-files-period-micros=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `21600000000`
- Range: `0` to `9223372036854775807`

`rocksdb_enable_2pc`

- Description: Enable two phase commit for MyRocks. When set, MyRocks will keep its data consistent with the binary log (in other words, the server are a crash-safe master). The consistency is achieved by doing two-phase XA commit with the binary log.
- Command line: `--rocksdb-enable-2pc={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_enable_bulk_load_api`

- Description: Enables using SstFileWriter for bulk loading.
- Command line: `--rocksdb-enable-bulk-load-api={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_enable_insert_with_update_caching`

- Description: Whether to enable optimization where we cache the read from a failed insertion attempt in INSERT ON DUPLICATE KEY UPDATE.
- Command line: `--rocksdb-enable-insert-with-update-caching={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.4.7, MariaDB 10.3.17, MariaDB 10.2.26

`rocksdb_enable_thread_tracking`

- Description: DBOptions::enable_thread_tracking for RocksDB.
- Command line: `--rocksdb-enable-thread-tracking={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_enable_ttl`

- Description: Enable expired TTL records to be dropped during compaction.
- Command line: `--rocksdb-enable-ttl={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_enable_ttl_read_filtering`

- Description: For tables with TTL, expired records are skipped/filtered out during processing and in query results. Disabling this will allow these records to be seen, but as a result rows may disappear in the middle of transactions as they are dropped during compaction. Use with caution.
- Command line: `--rocksdb-enable-ttl-read-filtering={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_enable_write_thread_adaptive_yield`

- Description: DBOptions::enable_write_thread_adaptive_yield for RocksDB.
- Command line: `--rocksdb-enable-write-thread-adaptive-yield={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_error_if_exists`

- Description: DBOptions::error_if_exists for RocksDBB.
- Command line: `--rocksdb-error-if-exists={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_error_on_suboptimal_collation`

- Description: Raise an error instead of warning if a sub-optimal collation is used.
- Command line: `--rocksdb-error-on-suboptimal-collation={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.3.10, MariaDB 10.2.18

`rocksdb_flush_log_at_trx_commit`

- Description: Sync on transaction commit. Similar to innodb_flush_log_at_trx_commit. One can check the flushing by examining the rocksdb_wal_synced and rocksdb_wal_bytes status variables.
  - 1: Always sync on commit (the default).
  - 0: Never sync.
  - 2: Sync based on a timer controlled via rocksdb-background-sync.
- Command line: `--rocksdb-flush-log-at-trx-commit=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `2`

`rocksdb_flush_memtable_on_analyze`

- Description: Forces memtable flush on ANALYZE table to get accurate cardinality.
- Command line: `--rocksdb-flush-memtable-on-analyze={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Removed: MariaDB 10.3.7, MariaDB 10.2.15

`rocksdb_force_compute_memtable_stats`

- Description: Force to always compute memtable stats.
- Command line: `--rocksdb-force-compute-memtable-stats={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_force_compute_memtable_stats_cachetime`

- Description: Time in usecs to cache memtable estimates.
- Command line: `--rocksdb-force-compute-memtable-stats-cachetime=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `60000000`
- Range: `0` to `2147483647`

`rocksdb_force_flush_memtable_and_lzero_now`

- Description: Acts similar to force_flush_memtable_now, but also compacts all L0 files.
- Command line: `--rocksdb-force-flush-memtable-and-lzero-now={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_force_flush_memtable_now`

- Description: Forces memstore flush which may block all write requests so be careful.
- Command line: `--rocksdb-force-flush-memtable-now={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`

- Default Value: `OFF`

`rocksdb_force_index_records_in_range`

- Description: Used to override the result of records_in_range() when FORCE INDEX is used.
- Command line: `--rocksdb-force-index-records-in-range=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `2147483647`

`rocksdb_git_hash`

- Description: Git revision of the RocksDB library used by MyRocks.
- Command line: `--rocksdb-git-hash=value=#`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: As per git revision.

`rocksdb_hash_index_allow_collision`

- Description: BlockBasedTableOptions::hash_index_allow_collision for RocksDB.
- Command line: `--rocksdb-hash-index-allow-collision={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_ignore_unknown_options`

- Description: Enable ignoring unknown options passed to RocksDB.
- Command line: `--rocksdb-ignore-unknown-options={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.3.7, MariaDB 10.2.15

`rocksdb_index_type`

- Description: BlockBasedTableOptions::index_type for RocksDB.
- Command line: `--rocksdb-index-type=value`
- Scope: Global
- Dynamic: No
- Data Type: `enum`
- Default Value: `kBinarySearch`
- Valid Values: `kBinarySearch`, `kHashSearch`

`rocksdb_info_log_level`

- Description: Filter level for info logs to be written mysqld error log. Valid values include 'debug_level', 'info_level', 'warn_level', 'error_level' and 'fatal_level'.
- Command line: `--rocksdb-info-log-level=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `error_level`
- Valid Values: `error_level`, `debug_level`, `info_level`, `warn_level`, `fatal_level`

`rocksdb_io_write_timeout`

- Description: Timeout for experimental I/O watchdog.
- Command line: `--rocksdb-io-write-timeout=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Valid Values: `0` to `4294967295`
- Introduced: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_is_fd_close_on_exec`

- Description: DBOptions::is_fd_close_on_exec for RocksDB.
- Command line: `--rocksdb-is-fd-close-on-exec={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_keep_log_file_num`

- Description: DBOptions::keep_log_file_num for RocksDB.
- Command line: `--rocksdb-keep-log-file-num=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1000`
- Range: `0` to `18446744073709551615`

`rocksdb_large_prefix`

- Description: Support large index prefix length of 3072 bytes. If off, the maximum index prefix length is 767.
- Command line: `--rocksdb-large_prefix={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_lock_scanned_rows`

- Description: Take and hold locks on rows that are scanned but not updated.
- Command line: `--rocksdb-lock-scanned-rows={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_lock_wait_timeout`

- Description: Number of seconds to wait for lock.
- Command line: `--rocksdb-lock-wait-timeout=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `1073741824`

`rocksdb_log_dir`

- Description: DBOptions::log_dir for RocksDB. Where the log files are stored. An empty value implies `rocksdb_datadir` is used as the directory.
- Command line: `--rocksdb-log-dir=#`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: (Empty)
- Introduced: MariaDB 10.9.1

`rocksdb_log_file_time_to_roll`

- Description: DBOptions::log_file_time_to_roll for RocksDB.
- Command line: `--rocksdb-log-file-time-to_roll=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`rocksdb_manifest_preallocation_size`

- Description: DBOptions::manifest_preallocation_size for RocksDB.
- Command line: `--rocksdb-manifest-preallocation-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `4194304`
- Range: `0` to `18446744073709551615`

`rocksdb_manual_compaction_threads`

- Description: How many rocksdb threads to run for manual compactions.
- Command line: `--rocksdb-manual-compation-threads=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `128`
- Introduced: MariaDB 10.3.10, MariaDB 10.2.18

`rocksdb_manual_wal_flush`

- Description: DBOptions::manual_wal_flush for RocksDB.
- Command line: `--rocksdb-manual-wal-flush={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_master_skip_tx_api`

- Description: Skipping holding any lock on row access. Not effective on slave.
- Command line: `--rocksdb-master-skip-tx-api={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_max_background_compactions`

- Description: DBOptions::max_background_compactions for RocksDB.
- Command line: `--rocksdb-max-background-compactions=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `64`
- Removed: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_max_background_flushes`

- Description: DBOptions::max_background_flushes for RocksDB.
- Command line: `--rocksdb-max-background-flushes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `64`
- Removed: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_max_background_jobs`

- Description: DBOptions::max_background_jobs for RocksDB.
- Command line: `--rocksdb-max-background-jobs=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2`
- Range: `-1` to `64`
- Introduced: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_max_latest_deadlocks`

- Description: Maximum number of recent deadlocks to store.
- Command line: `--rocksdb-max-latest-deadlocks=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `5`
- Range: `0` to `4294967295`

`rocksdb_max_log_file_size`

- Description: DBOptions::max_log_file_size for RocksDB.
- Command line: `--rocksdb-max-log-file-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`rocksdb_max_manifest_file_size`

- Description: DBOptions::max_manifest_file_size for RocksDB.
- Command line: `--rocksdb-manifest-log-file-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1073741824`
- Range: `0` to `18446744073709551615`

`rocksdb_max_manual_compactions`

- Description: Maximum number of pending + ongoing number of manual compactions..
- Command line: `--rocksdb-manual_compactions=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10`
- Range: `0` to `4294967295`

- Introduced: MariaDB 10.3.10, MariaDB 10.2.18

`rocksdb_max_open_files`

- Description: DBOptions::max_open_files for RocksDB.
- Command line: `--rocksdb-max-open-files=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-2`
- Range: `-2` to `2147483647`

`rocksdb_max_row_locks`

- Description: Maximum number of locks a transaction can have.
- Command line: `--rocksdb-max-row-locks=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1048576`
- Range:
  - `1` to `1073741824` (>= MariaDB 10.3.10, MariaDB 10.2.18)
  - `1` to `1048576` (<= MariaDB 10.3.9, MariaDB 10.2.17)

`rocksdb_max_subcompactions`

- Description: DBOptions::max_subcompactions for RocksDB.
- Command line: `--rocksdb-max-subcompactions=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `64`

`rocksdb_max_total_wal_size`

- Description: DBOptions::max_total_wal_size for RocksDB. The maximum size limit for write-ahead-log files. Once this limit is reached, RocksDB forces the flushing of memtables.
- Command line: `--rocksdb-max-total-wal-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `9223372036854775807`

`rocksdb_merge_buf_size`

- Description: Size to allocate for merge sort buffers written out to disk during inplace index creation.
- Command line: `--rocksdb-merge-buf-size=#`
- Scope: Global, Session

- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `67108864`
- Range: `100` to `18446744073709551615`

`rocksdb_merge_combine_read_size`

- Description: Size that we have to work with during combine (reading from disk) phase of external sort during fast index creation.
- Command line: `--rocksdb-merge-combine-read-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1073741824`
- Range: `100` to `18446744073709551615`

`rocksdb_merge_tmp_file_removal_delay_ms`

- Description: Fast index creation creates a large tmp file on disk during index creation. Removing this large file all at once when index creation is complete can cause trim stalls on Flash. This variable specifies a duration to sleep (in milliseconds) between calling chsize() to truncate the file in chunks. The chunk size is the same as merge_buf_size.
- Command line: `--rocksdb-merge-tmp-file-removal-delay-ms=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`rocksdb_new_table_reader_for_compaction_inputs`

- Description: DBOptions::new_table_reader_for_compaction_inputs for RocksDB.
- Command line: `--rocksdb-new-table-reader-for-compaction-inputs={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_no_block_cache`

- Description: BlockBasedTableOptions::no_block_cache for RocksDB.
- Command line: `--rocksdb-no-block-cache={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_override_cf_options`

- Description: Option overrides per cf for RocksDB. Note that the `rocksdb-override-cf-options` syntax is quite strict, and any typos will result in a parse error, and the MyRocks plugin will not be loaded. Depending on your configuration, the server may still start. If it does start, you can use this command to check if the plugin is loaded: `select * from information_schema.plugins where plugin_name='ROCKSDB'` (note that you need the "ROCKSDB" plugin. Other auxiliary plugins like "ROCKSDB_TRX" might still get loaded). Another way is to detect the error is check the error log.
- Command line: `--rocksdb-override-cf-options=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: (Empty)

`rocksdb_paranoid_checks`

- Description: DBOptions::paranoid_checks for RocksDB.
- Command line: `--rocksdb-paranoid-checks={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_pause_background_work`

- Description: Disable all rocksdb background operations.
- Command line: `--rocksdb-pause-background-work={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_perf_context_level`

- Description: Perf Context Level for rocksdb internal timer stat collection.
- Command line: `--rocksdb-perf-context-level=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `5`

`rocksdb_persistent_cache_path`

- Description: Path for BlockBasedTableOptions::persistent_cache for RocksDB.
- Command line: `--rocksdb-persistent-cache-path=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: (Empty)

`rocksdb_persistent_cache_size_mb`

- Description: Size of cache in MB for BlockBasedTableOptions::persistent_cache for RocksDB.
- Command line: `--rocksdb-persistent-cache-size-mb=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`rocksdb_pin_l0_filter_and_index_blocks_in_cache`

- Description: pin_l0_filter_and_index_blocks_in_cache for RocksDB.
- Command line: `--rocksdb-pin-l0-filter-and-index-blocks-in-cache={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_print_snapshot_conflict_queries`

- Description: Logging queries that got snapshot conflict errors into *.err log.
- Command line: `--rocksdb-print-snapshot-conflict-queries={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_rate_limiter_bytes_per_sec`

- Description: DBOptions::rate_limiter bytes_per_sec for RocksDB.
- Command line: `--rocksdb-rate-limiter-bytes-per-sec=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `9223372036854775807`

`rocksdb_read_free_rpl_tables`

- Description: List of tables that will use read-free replication on the slave (i.e. not lookup a row during replication).
- Command line: `--rocksdb-read-free-rpl-tables=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: (Empty)
- Removed: MariaDB 10.4.7, MariaDB 10.3.17, MariaDB 10.2.26

`rocksdb_records_in_range`

- Description: Used to override the result of records_in_range(). Set to a positive number to override.

- Command line: `--rocksdb-records-in-range=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `2147483647`

`rocksdb_remove_mariadb-backup_checkpoint`

- Description: Remove mariadb-backup checkpoint.
- Command line: `--rocksdb-remove-mariadb-backup-checkpoint={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.3.8, MariaDB 10.2.16

`rocksdb_reset_stats`

- Description: Reset the RocksDB internal statistics without restarting the DB.
- Command line: `--rocksdb-reset-stats={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_rollback_on_timeout`

- Description: Whether to roll back the complete transaction or a single statement on lock wait timeout (a single statement by default).
- Command line: `--rocksdb-rollback-on-timeout={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.4.7, MariaDB 10.3.17, MariaDB 10.2.26

`rocksdb_seconds_between_stat_computes`

- Description: Sets a number of seconds to wait between optimizer stats recomputation. Only changed indexes are refreshed.
- Command line: `--rocksdb-seconds-between-stat-computes=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `3600`
- Range: `0` to `4294967295`

`rocksdb_signal_drop_index_thread`

- Description: Wake up drop index thread.
- Command line: `--rocksdb-signal-drop-index-thread={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_sim_cache_size`

- Description: Simulated cache size for RocksDB.
- Command line: `--rocksdb-sim-cache-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `9223372036854775807`

`rocksdb_skip_bloom_filter_on_read`

- Description: Skip using bloom filter for reads.
- Command line: `--rocksdb-skip-bloom-filter-on_read={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_skip_fill_cache`

- Description: Skip filling block cache on read requests.
- Command line: `--rocksdb-skip-fill-cache={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_skip_unique_check_tables`

- Description: Skip unique constraint checking for the specified tables.
- Command line: `--rocksdb-skip-unique-check-tables=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `.*`

`rocksdb_sst_mgr_rate_bytes_per_sec`

- Description: DBOptions::sst_file_manager rate_bytes_per_sec for RocksDB
- Command line: `--rocksdb-sst-mgr-rate-bytes-per-sec=#`
- Scope: Global

- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`rocksdb_stats_dump_period_sec`

- Description: DBOptions::stats_dump_period_sec for RocksDB.
- Command line: `--rocksdb-stats-dump-period-sec=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `600`
- Range: `0` to `2147483647`

`rocksdb_stats_level`

- Description: Statistics Level for RocksDB. Default is 0 (kExceptHistogramOrTimers).
- Command line: `--rocksdb-stats-level=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4`
- Introduced: MariaDB 10.4.7, MariaDB 10.3.17, MariaDB 10.2.26

`rocksdb_stats_recalc_rate`

- Description: The number of indexes per second to recalculate statistics for. 0 to disable background recalculation.
- Command line: `--rocksdb-stats-recalc_rate=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4294967295`
- Introduced: MariaDB 10.3.10 MariaDB 10.2.18

`rocksdb_store_row_debug_checksums`

- Description: Include checksums when writing index/table records.
- Command line: `--rocksdb-store-row-debug-checksums={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_strict_collation_check`

- Description: Enforce case sensitive collation for MyRocks indexes.

- Command line: `--rocksdb-strict-collation-check={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_strict_collation_exceptions`

- Description: List of tables (using regex) that are excluded from the case sensitive collation enforcement.
- Command line: `--rocksdb-strict-collation-exceptions=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: (Empty)

`rocksdb_supported_compression_types`

- Description: Compression algorithms supported by RocksDB. Note that RocksDB does not make use of MariaDB 10.7 compression-plugins.
- Command line: `--rocksdb-supported-compression-types=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `Snappy,Zlib,ZSTDNotFinal`

`rocksdb_table_cache_numshardbits`

- Description: DBOptions::table_cache_numshardbits for RocksDB.
- Command line: `--rocksdb-table-cache-numshardbits=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `6`
- Range: `0` to `19`

`rocksdb_table_stats_sampling_pct`

- Description: Percentage of entries to sample when collecting statistics about table properties. Specify either 0 to sample everything or percentage [1..100]. By default 10% of entries are sampled.
- Command line: `--rocksdb-table-stats-sampling-pct=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10`
- Range: `0` to `100`

`rocksdb_tmpdir`

- Description: Directory for temporary files during DDL operations.
- Command line: `--rocksdb-tmpdir[=value]`

- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: (Empty)

`rocksdb_trace_sst_api`

- Description: Generate trace output in the log for each call to the SstFileWriter.
- Command line: `--rocksdb-trace-sst-api={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_two_write_queues`

- Description: DBOptions::two_write_queues for RocksDB.
- Command line: `--rocksdb-two-write-queues={0|1}`
- Scope: Global,
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.3.7, MariaDB 10.2.15

`rocksdb_unsafe_for_binlog`

- Description: Allowing statement based binary logging which may break consistency.
- Command line: `--rocksdb-unsafe-for-binlog={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_update_cf_options`

- Description: Option updates per column family for RocksDB.
- Command line: `--rocksdb-update-cf-options=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `varchar`
- Default Value: (Empty)
- Introduced: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_use_adaptive_mutex`

- Description: DBOptions::use_adaptive_mutex for RocksDB.
- Command line: `--rocksdb-use-adaptive-mutex={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`

- Default Value: `OFF`

`rocksdb_use_clock_cache`

- Description: Use ClockCache instead of default LRUCache for RocksDB.
- Command line: `--rocksdb-use-clock-cache={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_use_direct_io_for_flush_and_compaction`

- Description: DBOptions::use_direct_io_for_flush_and_compaction for RocksDB.
- Command line: `--rocksdb-use-direct-io-for-flush-and-compaction={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_use_direct_reads`

- Description: DBOptions::use_direct_reads for RocksDB.
- Command line: `--rocksdb-use-direct-reads={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_use_direct_writes`

- Description: DBOptions::use_direct_writes for RocksDB.
- Command line: `--rocksdb-use-direct-reads={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.3.1, MariaDB 10.2.8

`rocksdb_use_fsync`

- Description: DBOptions::use_fsync for RocksDB.
- Command line: `--rocksdb-use-fsync={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_validate_tables`

- Description: Verify all .frm files match all RocksDB tables (0 means no verification, 1 means verify and fail on error, and 2 means verify but continue.
- Command line: `--rocksdb-validate-tables=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `2`

`rocksdb_verify_row_debug_checksums`

- Description: Verify checksums when reading index/table records.
- Command line: `--rocksdb-verify-row-debug-checksums={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_wal_bytes_per_sync`

- Description: DBOptions::wal_bytes_per_sync for RocksDB.
- Command line: `--rocksdb-wal-bytes-per-sync=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`rocksdb_wal_dir`

- Description: DBOptions::wal_dir for RocksDB. Directory where the write-ahead-log files are stored.
- Command line: `--rocksdb-wal-dir=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: (Empty)

`rocksdb_wal_recovery_mode`

- Description: DBOptions::wal_recovery_mode for RocksDB. Default is kAbsoluteConsistency. Records that are not yet committed are stored in the Write-Ahead-Log (WAL). If the server is not cleanly shut down, the recovery mode will determine the WAL recovery behavior.
  - 1: kAbsoluteConsistency. Will not start if any corrupted entries (including incomplete writes) are detected (the default).
  - 0: kTolerateCorruptedTailRecords. Ignores any errors found at the end of the WAL
  - 2: kPointInTimeRecovery. Replay of the WAL is halted after finding an error. The system are recovered to the latest consistent point-in-time. Data from a replica can used to replay past the point-in-time.
  - 3: kSkipAnyCorruptedRecords. A risky option where any corrupted entries are skipped while subsequent healthy WAL entries are applied.

- Command line: `--rocksdb-wal-recovery-mode=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `3`

`rocksdb_wal_size_limit_mb`

- Description: DBOptions::WAL_size_limit_MB for RocksDB. Write-ahead-log files are moved to an archive directory once their memtables are flushed. This variable specifies the largest size, in MB, that the archive can be.
- Command line: `--rocksdb-wal-size-limit-mb=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `9223372036854775807`

`rocksdb_wal_ttl_seconds`

- Description: DBOptions::WAL_ttl_seconds for RocksDB. Oldest time, in seconds, that a write-ahead-log file should exist.
- Command line: `--rocksdb-wal-ttl-seconds=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `9223372036854775807`

`rocksdb_whole_key_filtering`

- Description: BlockBasedTableOptions::whole_key_filtering for RocksDB. If set (the default), the bloomfilter to use the whole key (rather than only the prefix) for filtering is enabled. Lookups should use the whole key for matching to make best use of this setting.
- Command line: `--rocksdb-whole-key-filtering={0|1}`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`rocksdb_write_batch_max_bytes`

- Description: Maximum size of write batch in bytes. 0 means no limit.
- Command line: `--rocksdb-write-batch-max-bytes=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`

- Range: `0` to `18446744073709551615`

`rocksdb_write_disable_wal`

- Description: WriteOptions::disableWAL for RocksDB.
- Command line: `--rocksdb-write-disable-wal={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_write_ignore_missing_column_families`

- Description: WriteOptions::ignore_missing_column_families for RocksDB.
- Command line: `--rocksdb-write-ignore-missing-column-families={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rocksdb_write_policy`

- Description: DBOptions::write_policy for RocksDB.
- Command line: `--rocksdb-write-policy=val`
- Scope: Global
- Dynamic: No
- Data Type: `enum`
- Default Value: `write_committed`
- Valid Values: `write_committed`, `write_prepared`, `write_unprepared`
- Introduced: MariaDB 10.3.10, MariaDB 10.2.18

This page is licensed: CC BY-SA / Gnu FDL

OQGRAPH

> **Note:**
>
> OQGRAPH is a community contribution to MariaDB Server, developed and maintained by its contributors: Arjen Lentz and Antony T Curtis of Open Query, and Andrew McDonnell.

See OQGRAPH Overview.

An overview of the OQGRAPH storage engine, which allows for handling hierarchies and graph structures using SQL.

See Installing OQGRAPH.

This guide provides instructions for installing the OQGRAPH storage engine from package repositories or enabling the plugin.

See Compiling OQGRAPH.

Learn about the dependencies and build steps required to compile the OQGRAPH storage engine from source.

See Building OQGRAPH Under Windows.

Specific instructions and prerequisites for building the OQGRAPH storage engine on Windows systems using Visual Studio.

See OQGRAPH Examples.

Practical examples demonstrating how to create OQGRAPH tables and perform graph operations like finding shortest paths and leaf nodes.

This page is licensed: CC BY-SA / Gnu FDL

OQGRAPH Overview

The Open Query GRAPH computation engine, or OQGRAPH as the engine itself is called, allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions).

OQGRAPH is unlike other storage engines, consisting of an entirely different architecture to a regular storage engine such as Aria, MyISAM or InnoDB.

It is intended to be used for retrieving hierarchical information, such as those used for graphs, routes or social relationships, in plain SQL.

Installing

See Installing OQGRAPH. Note that if the query cache is enabled, OQGRAPH will not use it.

Creating a Table

The following documentation is based upon MariaDB 10.0.7 and OQGRAPH v3.

Example with origin and destination nodes only

To create an OQGRAPH v3 table, a backing table must first be created. This backing table will store the actual data, and are used for all INSERTs, UPDATEs and so on. It must be a regular table, not a view. Here's a simple example to start with:

```
CREATE TABLE oq_backing (
  origid INT UNSIGNED NOT NULL,
  destid INT UNSIGNED NOT NULL,
  PRIMARY KEY (origid, destid),
  KEY (destid)
);
```

Some data can be inserted into the backing table to test with later:

```
INSERT INTO oq_backing(origid, destid)
 VALUES (1,2), (2,3), (3,4), (4,5), (2,6), (5,6);
```

Now the read-only OQGRAPH table is created. The CREATE statement must match the format below - any difference will result in an error.

```
CREATE TABLE oq_graph (
  latch VARCHAR(32) NULL,
  origid BIGINT UNSIGNED NULL,
  destid BIGINT UNSIGNED NULL,
  weight DOUBLE NULL,
  seq BIGINT UNSIGNED NULL,
  linkid BIGINT UNSIGNED NULL,
  KEY (latch, origid, destid) USING HASH,
  KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';
```

An older format (prior to MariaDB 10.0.7) has the latch field as a SMALLINT rather than a VARCHAR. The format gives an error:

```
CREATE TABLE oq_old (
  latch SMALLINT UNSIGNED NULL,
  origid BIGINT UNSIGNED NULL,
  destid BIGINT UNSIGNED NULL,
  weight DOUBLE NULL,
  seq BIGINT UNSIGNED NULL,
  linkid BIGINT UNSIGNED NULL,
  KEY (latch, origid, destid) USING HASH,
  KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';

ERROR 1005 (HY000): Can't create table `test`.`oq_old` (errno: 140 "Wrong create options")
```

The old, deprecated format can still be used prior to MariaDB 11.5 if the value of the oqgraph_allow_create_integer_latch system variable is changed from its default, `FALSE`, to `TRUE`.

```
SET GLOBAL oqgraph_allow_create_integer_latch=1;

CREATE TABLE oq_old (
  latch SMALLINT UNSIGNED NULL,
  origid BIGINT UNSIGNED NULL,
  destid BIGINT UNSIGNED NULL,
  weight DOUBLE NULL,
  seq BIGINT UNSIGNED NULL,
  linkid BIGINT UNSIGNED NULL,
  KEY (latch, origid, destid) USING HASH,
  KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';
Query OK, 0 rows affected, 1 warning (0.19 sec)

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------------------------------
-------------------------------------------+
| Level   | Code | Message
|
+---------+------+-----------------------------------------------------------------------------------------------
-------------------------------------------+
| Warning | 1287 | 'latch SMALLINT UNSIGNED NULL' is deprecated and are removed in a future release.
Please use 'latch VARCHAR(32) NULL' instead |
+---------+------+-----------------------------------------------------------------------------------------------
-------------------------------------------+
```

Data is only inserted into the backing table, not the OQGRAPH table.

Now, having created the `oq_graph` table linked to a backing table, it is now possible to query the `oq_graph` table directly. The `weight` field, since it was not specified in this example, defaults to `1`.

```
SELECT * FROM oq_graph;
+-------+--------+--------+--------+------+--------+
| latch | origid | destid | weight | seq  | linkid |
+-------+--------+--------+--------+------+--------+
|  NULL |      1 |      2 |      1 | NULL |   NULL |
|  NULL |      2 |      3 |      1 | NULL |   NULL |
|  NULL |      2 |      6 |      1 | NULL |   NULL |
|  NULL |      3 |      4 |      1 | NULL |   NULL |
|  NULL |      4 |      5 |      1 | NULL |   NULL |
|  NULL |      5 |      6 |      1 | NULL |   NULL |
+-------+--------+--------+--------+------+--------+
```

The data here represents one-directional starting and ending nodes. So node 2 has paths to node 3 and node 6, while node 6 has no paths to any other node.

Manipulating Weight

There are three fields which can be manipulated: `origid`, `destid` (the example above uses these two), as well as `weight`. To create a backing table with a `weight` field as well, the following syntax can be used:

```
CREATE TABLE oq2_backing (
  origid INT UNSIGNED NOT NULL,
  destid INT UNSIGNED NOT NULL,
  weight DOUBLE NOT NULL,
  PRIMARY KEY (origid, destid),
  KEY (destid)
);
```

```
INSERT INTO oq2_backing(origid, destid, weight)
 VALUES (1,2,1), (2,3,1), (3,4,3), (4,5,1), (2,6,10), (5,6,2);
```

```
CREATE TABLE oq2_graph (
  latch VARCHAR(32) NULL,
  origid BIGINT UNSIGNED NULL,
  destid BIGINT UNSIGNED NULL,
  weight DOUBLE NULL,
  seq BIGINT UNSIGNED NULL,
  linkid BIGINT UNSIGNED NULL,
  KEY (latch, origid, destid) USING HASH,
  KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq2_backing' origid='origid' destid='destid' weight='weight';
```

```
SELECT * FROM oq2_graph;
+-------+--------+--------+--------+------+--------+
| latch | origid | destid | weight | seq  | linkid |
+-------+--------+--------+--------+------+--------+
| NULL  |      1 |      2 |      1 | NULL |   NULL |
| NULL  |      2 |      3 |      1 | NULL |   NULL |
| NULL  |      2 |      6 |     10 | NULL |   NULL |
| NULL  |      3 |      4 |      3 | NULL |   NULL |
| NULL  |      4 |      5 |      1 | NULL |   NULL |
| NULL  |      5 |      6 |      2 | NULL |   NULL |
+-------+--------+--------+--------+------+--------+
```

See OQGRAPH Examples for OQGRAPH usage examples.

This page is licensed: CC BY-SA / Gnu FDL

Installing OQGRAPH

The Open Query GRAPH computation engine, or OQGRAPH as the engine itself is called, allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions).

Installation

The OQGRAPH storage engine exists as a separate package in the repositories. On Ubuntu and Debian the package is called `mariadb-oqgraph-engine-10.0` or `mariadb-plugin-oqgraph`. On Red Hat, CentOS, and Fedora the package is called `MariaDB-oqgraph-engine`. To install the plugin, first install the appropriate package and then install the plugin using the INSTALL SONAME or INSTALL PLUGIN commands.

Debian and Ubuntu

On Debian and Ubuntu, install the package as follows:

```
sudo apt-get install mariadb-oqgraph-engine-10.0
```

Fedora/Red Hat/CentOS

Note that OQGRAPH v3 requires libjudy, which is not in the official Red Hat/Fedora repositories. This needs to be installed first, for example:

```
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
rpm -Uvh epel-release-6-8.noarch.rpm
```

Then install the package, as follows:

```
sudo yum install MariaDB-oqgraph-engine
```

Installing the Plugin

On either system you can then launch the `mysql` command-line client and install the plugin in MariaDB as follows:

```
INSTALL SONAME 'ha_oqgraph';
```

See Also

More information on this engine is found on the OpenQuery website: graph-engine

This page is licensed: CC BY-SA / Gnu FDL

Compiling OQGRAPH

To compile OQGraph v2 you need to have the boost library with the versions not earlier than 1.40 and not later than 1.55 and gcc version not earlier than 4.5.

Note

OQGraph v3 compiles fine with the newer boost libraries, but it additionally needs the Judy library installed.

When all build prerequisites are met, OQGraph is enabled and compiled automatically. To enable or disable OQGRAPH explicitly, see the generic plugin build instructions.

Finding Out Why OQGRAPH Didn't Compile

If OQGRAPH gets compiled properly, there should be a file like:

```
storage/oqgraph/ha_oqgraph.so
```

If this is not the case, then you can find out if there is any modules missing that are required by OQGRAPH by doing:

```
cmake . -LAH | grep -i oqgraph
```

This page is licensed: CC BY-SA / Gnu FDL

Building OQGRAPH Under Windows

OQGRAPH v3 can be built on Windows. This has been tested using Windows 7, Microsoft Visual Studio Express 2010 (32-bit), Microsoft Windows 64-bit Platform SDK 7.1 (64-bit), the Boost library >= 1.55 and Judy 1.0.5.

Note

Other recent versions of Boost, Judy or MSVC may work but these combinations have not been tested.

- Download the source package for Boost 1.55 from the Boost project website, www.boost.org
- Download the source package for Judy 1.05 via
- Follow the documented instructions for building under Windows from the command line: Building_MariaDB_on_Windows
  - Ensure that the following variable is set to CMAKE: `JUDY_ROOT=path\to\judy\unzipped`
  - See also comments in `storage/oqgraph/cmake/FindJudy.cmake`

This page is licensed: CC BY-SA / Gnu FDL

OQGRAPH Examples

Creating a Table with origid, destid Only

```
CREATE TABLE oq_backing (
  origid INT UNSIGNED NOT NULL,
  destid INT UNSIGNED NOT NULL,
  PRIMARY KEY (origid, destid),
  KEY (destid)
);
```

Some data can be inserted into the backing table to test with later:

```
INSERT INTO oq_backing(origid, destid)
 VALUES (1,2), (2,3), (3,4), (4,5), (2,6), (5,6);
```

Now the read-only OQGRAPH table is created.

You can use the following syntax:

```
CREATE TABLE oq_graph
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';
```

Creating a Table with Weight

For the examples on this page, we'll create a second OQGRAPH table and backing table, this time with `weight` as well.

```
CREATE TABLE oq2_backing (
  origid INT UNSIGNED NOT NULL,
  destid INT UNSIGNED NOT NULL,
  weight DOUBLE NOT NULL,
  PRIMARY KEY (origid, destid),
  KEY (destid)
);
```

```
INSERT INTO oq2_backing(origid, destid, weight)
 VALUES (1,2,1), (2,3,1), (3,4,3), (4,5,1), (2,6,10), (5,6,2);
```

```
CREATE TABLE oq2_graph (
  latch VARCHAR(32) NULL,
  origid BIGINT UNSIGNED NULL,
  destid BIGINT UNSIGNED NULL,
  weight DOUBLE NULL,
  seq BIGINT UNSIGNED NULL,
  linkid BIGINT UNSIGNED NULL,
  KEY (latch, origid, destid) USING HASH,
  KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq2_backing' origid='origid' destid='destid' weight='weight';
```

Shortest Path

A `latch` value of `'dijkstras'` and an `origid` and `destid` is used for finding the shortest path between two nodes, for example:

```
SELECT * FROM oq_graph WHERE latch='breadth_first' AND origid=1 AND destid=6;
+----------+--------+--------+--------+------+--------+
| latch    | origid | destid | weight | seq  | linkid |
+----------+--------+--------+--------+------+--------+
| dijkstras|      1 |      6 |   NULL |    0 |      1 |
| dijkstras|      1 |      6 |      1 |    1 |      2 |
| dijkstras|      1 |      6 |      1 |    2 |      6 |
+----------+--------+--------+--------+------+--------+
```

Note that nodes are uni-directional, so there is no path from node 6 to node 1:

```
SELECT * FROM oq_graph WHERE latch='dijkstras' AND origid=6 AND destid=1;
Empty set (0.00 sec)
```

Using the GROUP_CONCAT function can produce more readable results, for example:

```
SELECT GROUP_CONCAT(linkid ORDER BY seq) AS path FROM oq_graph
 WHERE latch='dijkstras' AND origid=1 AND destid=6;
+-------+
| path  |
+-------+
| 1,2,6 |
+-------+
```

Using the table `oq2_graph`, the shortest path is different:

```
SELECT GROUP_CONCAT(linkid ORDER BY seq) AS path FROM oq2_graph
 WHERE latch='dijkstras' AND origid=1 AND destid=6;
+-------------+
| path        |
+-------------+
| 1,2,3,4,5,6 |
+-------------+
```

The reason is the weight between nodes 2 and 6 is `10` in `oq_graph2`, so the shortest path taking into account `weight` is now across more nodes.

Possible Destinations

```
SELECT GROUP_CONCAT(linkid) AS dests FROM oq_graph WHERE latch='dijkstras' AND origid=2;
+-----------+
| dests     |
+-----------+
| 5,4,6,3,2 |
+-----------+
```

Note that this returns all possible destinations along the path, not just immediate links.

Leaf Nodes

MariaDB supports the `leaves` latch value. A `latch` value of `'leaves'` and either `origid` or `destid` is used for finding leaf nodes at the beginning or end of a graph.

```
INSERT INTO oq_backing(origid, destid)
 VALUES (1,2), (2,3), (3,5), (4,5), (5,6), (6,7), (6,8), (2,8);
```

For example, to find all reachable nodes from `origid` that only have incoming edges:

```
SELECT * FROM oq_graph WHERE latch='leaves' AND origid=2;
+--------+--------+--------+--------+------+--------+
| latch  | origid | destid | weight | seq  | linkid |
+--------+--------+--------+--------+------+--------+
| leaves |      2 |   NULL |      4 |    2 |      7 |
| leaves |      2 |   NULL |      1 |    1 |      8 |
+--------+--------+--------+--------+------+--------+
```

And to find all nodes from which a path can be found to `destid` that only have outgoing edges:

```
SELECT * FROM oq_graph WHERE latch='leaves' AND destid=5;
+--------+--------+--------+--------+------+--------+
| latch  | origid | destid | weight | seq  | linkid |
+--------+--------+--------+--------+------+--------+
| leaves |   NULL |      5 |      3 |    2 |      1 |
| leaves |   NULL |      5 |      1 |    1 |      4 |
+--------+--------+--------+--------+------+--------+
```

Summary of Implemented Latch Commands

Latch	Alternative	additional where clause fields	Graph operation
NULL	(unspecified)	(none)	List original data
(empty string)	0	(none extra)	List all vertices in linkid column
(empty string)	0	origid	List all first hop vertices from origid in linkid column
dijkstras	1	origid, destid	Find shortest path using Dijkstras algorithm between origid and destid, with traversed vertex ids in linkid column
dijkstras	1	origid	Find all vertices reachable from origid, listed in linkid column, and report sum of weights of vertices on path to given vertex in weight
dijkstras	1	destid	Find all vertices from which a path can be found to destid, listed in linkid column, and report sum of weights of vertices on path to given vertex in weight

Latch	Alternative	additional where clause fields	Graph operation
breadth_first	2	origid	List vertices reachable from origid in linkid column
breadth_first	2	destid	List vertices from which a path can be found to destid in linkid column
breadth_first	2	origid, destid	Find shortest path between origid and destid, report in linkid column
leaves	4	origid	List vertices reachable from origid, that only have incoming edges
leaves	4	destid	List vertices from which a path can be found to destid, that only have outgoing edges
leaves	4	origid, destid	Not supported, will return an empty result

Note

The use of integer latch commands is deprecated and may be phased out in a future release. Currently, numeric values in the strings are interpreted as aliases, and use of an integer column can be optionally allowed, for the latch commands column.

The use of integer latches is controlled using the oqgraph_allow_create_integer_latch system variable.

See Also

- OQGRAPH Overview

This page is licensed: CC BY-SA / Gnu FDL

PERFORMANCE_SCHEMA

If you run SHOW ENGINES, you'll see the following storage engine listed.

```
SHOW ENGINES\G
...
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
...
```

The PERFORMANCE_SCHEMA is not a regular storage engine for storing data, it's a mechanism for implementing the Performance Schema feature.

The SHOW ENGINE PERFORMANCE_SCHEMA STATUS statement is also available, which shows how much memory is used by the tables and internal buffers.

See Performance Schema for more details.

This page is licensed: CC BY-SA / Gnu FDL

SEQUENCE Storage Engine

This article is about the Sequence storage engine. For details about sequence objects, see Sequences.

A **Sequence** engine allows the creation of ascending or descending sequences of numbers (positive integers) with a given starting value, ending value and increment.

It creates completely virtual, ephemeral tables automatically when you need them. There is no way to create a Sequence table explicitly. Nor are they ever written to disk or create `.frm` files. They are read-only, transactional, and support XA.

Installing

The Sequence engine is installed by default, and SHOW ENGINES will list the Sequence storage engine as supported:

```
SHOW ENGINES\G
...
*************************** 5. row ***************************
      Engine: MyISAM
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 6. row ***************************
      Engine: SEQUENCE
     Support: YES
     Comment: Generated tables filled with sequential values
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 7. row ***************************
      Engine: MRG_MyISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO

...
```

Usage and Examples

To use a Sequence table, you simply select from it, as in

```
SELECT * FROM seq_1_to_5;
+-----+
| seq |
+-----+
|   1 |
|   2 |
|   3 |
|   4 |
|   5 |
+-----+
```

To use a sequence in a statement, you select from the table named by a pattern **seq**_`FROM`***to***`TO` or **seq**_`FROM`***to***`TO`***step***`STEP`.

In the case of an odd step, the sequence will commence with the `FROM`, and end at the final result before `TO`.

```
SELECT * FROM seq_1_to_15_step_3;
+-----+
| seq |
+-----+
|   1 |
|   4 |
|   7 |
|  10 |
|  13 |
+-----+
```

A sequence can go backwards too. In this case the final value will always be the `TO` value, so that a descending sequence has the same values as an ascending sequence:

```
SELECT * FROM seq_5_to_1_step_2;
+-----+
| seq |
+-----+
|   5 |
|   3 |
|   1 |
+-----+
```

```
SELECT * FROM seq_15_to_1_step_3;
+-----+
| seq |
+-----+
|  13 |
|  10 |
|   7 |
|   4 |
|   1 |
+-----+
```

```
SELECT * FROM seq_15_to_2_step_3;
+-----+
| seq |
+-----+
|  14 |
|  11 |
|   8 |
|   5 |
|   2 |
+-----+
```

This engine is particularly useful with joins and subqueries. For example, this query finds all prime numbers below 50:

```
SELECT seq FROM seq_2_to_50 s1 WHERE 0 NOT IN
     (SELECT s1.seq % s2.seq FROM seq_2_to_50 s2 WHERE s2.seq <= sqrt(s1.seq));
+-----+
| seq |
+-----+
|   2 |
|   3 |
|   5 |
|   7 |
|  11 |
|  13 |
|  17 |
|  19 |
|  23 |
|  29 |
|  31 |
|  37 |
|  41 |
|  43 |
|  47 |
+-----+
```

And almost (without 2, the only even prime number) the same result with joins:

```
SELECT s1.seq FROM seq_2_to_50 s1 JOIN seq_2_to_50 s2
  WHERE s1.seq > s2.seq AND s1.seq % s2.seq <> 0
  GROUP BY s1.seq HAVING s1.seq - COUNT(*) = 2;
+-----+
| seq |
+-----+
|   3 |
|   5 |
|   7 |
|  11 |
|  13 |
|  17 |
|  19 |
|  23 |
|  29 |
|  31 |
|  37 |
|  41 |
|  43 |
|  47 |
+-----+
```

Sequence tables can also be useful in date calculations. For example, to find the day of the week that a particular date has fallen on over a 40 year period (perhaps for birthday planning ahead!):

```
SELECT DAYNAME('1980-12-05' + INTERVAL (seq) YEAR) day,
    '1980-12-05' + INTERVAL (seq) YEAR DATE FROM seq_0_to_40;
+-----------+------------+
| day       | date       |
+-----------+------------+
| Friday    | 1980-12-05 |
| Saturday  | 1981-12-05 |
| Sunday    | 1982-12-05 |
...
| Friday    | 2014-12-05 |
| Saturday  | 2015-12-05 |
| Monday    | 2016-12-05 |
| Tuesday   | 2017-12-05 |
| Wednesday | 2018-12-05 |
| Thursday  | 2019-12-05 |
| Saturday  | 2020-12-05 |
+-----------+------------+
```

Although Sequence tables can only directly make use of positive integers, they can indirectly be used to return negative results by making use of the CAST statement:

```
SELECT CAST(seq AS INT) - 5 x FROM seq_5_to_1;
+----+
| x  |
+----+
|  0 |
| -1 |
| -2 |
| -3 |
| -4 |
+----+
```

CAST is required to avoid a `BIGINT UNSIGNED value is out of range` error.

Sequence tables, while virtual, are still tables, so they must be in a database. This means that a default database must be selected (for example, via the USE command) to be able to query a Sequence table. The information_schema database cannot be used as the default for a Sequence table.

Table Name Conflicts

If the SEQUENCE storage engine is installed, it is not possible to create a table with a name which follows the SEQUENCE pattern:

```
CREATE TABLE seq_1_to_100 (col INT) ENGINE = InnoDB;
ERROR 1050 (42S01): Table 'seq_1_to_100' already exists
```

However, a SEQUENCE table can be converted to another engine and the new table can be referred in any statement:

```
ALTER TABLE seq_1_to_100 ENGINE = BLACKHOLE;

SELECT * FROM seq_1_to_100;
Empty set (0.00 sec)
```

While a SEQUENCE table cannot be dropped, it is possible to drop the converted table. The SEQUENCE table with the same name will still exist:

```
DROP TABLE seq_1_to_100;

SELECT COUNT(*) FROM seq_1_to_100;
+----------+
| COUNT(*) |
+----------+
|      100 |
+----------+
1 row in set (0.00 sec)
```

A temporary table with a SEQUENCE-like name can always be created and used:

```
CREATE TEMPORARY TABLE seq_1_to_100 (col INT) ENGINE = InnoDB;

SELECT * FROM seq_1_to_100;
Empty set (0.00 sec)
```

See Also

- For details about sequence objects, see Sequences.
- Sometimes its the little things - Dean Ellis tries out the Sequence engine.

- MariaDB's Sequence Storage Engine - Federico Razzoli writes more in-depth on the engine.

This page is licensed: CC BY-SA / Gnu FDL

S3 Storage Engine

See Using the S3 Storage Engine.

This guide covers typical use cases for the S3 engine, such as archiving inactive tables, and details supported operations like ALTER TABLE and SELECT.

See S3 Status Variables.

A list of status variables for monitoring the S3 engine's interactions with the cloud service, although specific variables are not extensively documented here.

See S3 System Variables.

This page lists system variables to configure the S3 engine, including AWS credentials, bucket names, page cache sizes, and connection protocols.

See S3 Engine Internals.

Learn about the internal architecture of the S3 engine, which inherits from Aria code but redirects reads to S3, using a dedicated page cache.

See Testing Connections to S3.

Instructions on how to verify your S3 configuration using tools like `aria_s3_copy` and the `mysql-test-run` suite to ensure proper connectivity.

See aria_s3_copy.

A reference for the `aria_s3_copy` tool, which is used to manually copy Aria tables to and from S3 storage for testing and data migration.

This page is licensed: CC BY-SA / Gnu FDL

Using the S3 Storage Engine

The S3 storage engine is read only and allows one to archive MariaDB tables in Amazon S3, or any third-party public or private cloud that implements S3 API (of which there are many), but still have them accessible for reading in MariaDB.

Installing the Plugin

The S3 storage engine is gamma maturity, so the following step can be omitted.

On earlier releases, when it was alpha maturity, it does not load by default on a stable release of the server due to the default value of the plugin_maturity variable. Set to `alpha` (or below) in your config file to permit installation of the plugin:

```
[mariadbd]
plugin-maturity = alpha
```

and restart the server.

Now install the plugin library, for example:

```
INSTALL SONAME 'ha_s3';
```

If the library is not available, for example:

```
INSTALL SONAME 'ha_s3';
ERROR 1126 (HY000): Can't open shared library '/var/lib/mysql/lib64/mysql/plugin/ha_s3.so'
  (errno: 13, cannot open shared object file: No such file or directory)
```

you may need to install a separate package for the S3 storage engine, for example:

```
shell> yum install MariaDB-s3-engine
```

or for Debian/Ubuntu

```
shell> apt install mariadb-plugin-s3
```

Creating an S3 Table

As S3 tables are read only, one cannot create a S3 table with `CREATE TABLE`. You should use instead use `ALTER TABLE old_table ENGINE=S3` to convert an existing table to be stored on S3.

Moving Data to S3

To move data from an existing table to S3, you can run:

```
ALTER TABLE old_table ENGINE=S3 COMPRESSION_ALGORITHM=zlib
```

To get data back to a 'normal' table, you can do:

```
ALTER TABLE s3_table ENGINE=INNODB
```

New Options for ALTER TABLE

- `S3_BLOCK_SIZE` : Set to 4M as default. This is the block size for all index and data pages stored in S3.
- `COMPRESSION_ALGORITHM` : Set to 'none' as default. Which compression algorithm to use for block stored in S3. Options are: `none` or `zlib`.

ALTER TABLE can be used on S3 tables as normal to add columns or change column definitions.

mariadbd Startup Options for S3

To be able to use S3 for storage one ***must*** define how to access S3 and where data are stored in S3:

- s3_access_key: The AWS access key to access your data
- s3_secret_key: The AWS secret key to access your data
- s3_bucket: The AWS bucket where your data should be stored. All MariaDB table data is stored in this bucket.
- s3_region: The AWS region where your data should be stored.

For compatibility tweaks with different providers:

- s3_provider: Enable S3 provider specific compatibility tweaks. "Default", "Amazon", or "Huawei". From MariaDB 11.6.2.

If you are using an S3 service that is using HTTP to connect (like) you also need the set the following variables:

- s3_port: Port number to connect to (0 means use default)
- s3_use_http: If true, force use of HTTP protocol

If you are going to use a primary-replica setup, you should look at the following variables:

- s3_replicate_alter_as_create_select: When converting an S3 table to local table, log all rows in binary log. Defaults to `TRUE`. This allows the replica to replicate `CREATE TABLE .. SELECT FROM s3_table` even it the

replica doesn't have access to the original `s3_table`.

- s3_slave_ignore_updates: Should be set if primary and replica share the same S3 instance. This tells the replica that it can ignore any updates to the S3 tables as they are already applied on the primary. Defaults to `FALSE`.

The above defaults assume that the primary and replica don't share the same S3 instance.

Other, less critical options, are:

- s3_host_name: Hostname for the S3 service. "s3.amazonaws.com", Amazon S3 service, by default.
- s3_protocol_version: Protocol used to communication with S3. One of "Auto", "Amazon" or "Original" where "Auto" is the default. If you get errors like "8 Access Denied" when you are connecting to another service provider, then try to change this option. The reason for this variable is that Amazon has changed some parts of the S3 protocol since they originally introduced it but other service providers are still using the original protocol.
- s3_block_size: Set to 4M as default. This is the default block size for a table, if not specified in CREATE TABLE.
- s3_pagecache_buffer_size: Default 128M. The size of the buffer used for data and index blocks for S3 tables. Increase this to get better index handling (for all reads and multiple writes) to as much as you can afford.
- s3_ssl_no_verify: If true, SSL certificate verification for the S3 endpoint is disabled. From MariaDB 11.6.2.
- ss3_no_content_type: If true (false is default), disables the Content-Type header, required for some providers. From MariaDB 11.6.2.

Last some options you probably don't have to ever touch:

- s3_pagecache_age_threshold : Default 300: This characterizes the number of hits a hot block has to be untouched until it is considered aged enough to be downgraded to a warm block. This specifies the percentage ratio of that number of hits to the total number of blocks in the page cache.
- s3_pagecache_division_limit: Default 100. The minimum percentage of warm blocks in key cache.
- s3_pagecache_file_hash_size: Default 512. Number of hash buckets for open files. If you have a lot of S3 files open you should increase this for faster flush of changes. A good value is probably 1/10 of number of possible open S3 files.
- s3_debug: Default 0. Generates a trace file from libmarias3 on stderr (mysqld.err) for debugging the S3 protocol.

Typical my.cnf Entry for connecting to Amazon S3 service

```
[mariadb]
s3=ON
s3-bucket=mariadb
s3-access-key=xxxx
s3-secret-key=xxx
s3-region=eu-north-1
s3-host-name=s3.amazonaws.com
# The following is useful if you want to use minio as a S3 server. (https://min.io/)
#s3-port=9000
#s3-use-http=ON

# Primary and replica share same S3 tables.
s3-slave-ignore-updates=1

[aria_s3_copy]
s3-bucket=mariadb
s3-access-key=xxxx
s3-secret-key=xxx
s3-region=eu-north-1
s3-host-name=s3.amazonaws.com
# The following is useful if you want to use minio as a S3 server. (https://min.io/)
#s3-port=9000
#s3-use-http=ON
```

Typical my.cnf entry for connecting to a minio S3 server

```
[mariadb]
s3=ON
s3-host-name="127.0.0.1"
s3-bucket=storage-engine
s3-access-key=minio
s3-secret-key=minioadmin
s3-port=9000
s3-use-http=ON

[aria_s3_copy]
s3=ON
s3-host-name="127.0.0.1"
s3-bucket=storage-engine
s3-access-key=minio
s3-secret-key=minioadmin
s3-port=9000
s3-use-http=ON
```

Typical Usage Case for S3 Tables

The typical use case would be that there exists tables that after some time would become fairly inactive, but are still important so that they can not be removed. In that case, an option is to move such a table to an archiving service, which is accessible through an S3 API.

Notice that S3 means the Cloud Object Storage API defined by Amazon AWS. Often the whole of Amazon's Cloud Object Storage is referred to as S3. In the context of the S3 archive storage engine, it refers to the API itself that defines how to store objects in a cloud service, being it Amazon's or someone else's. OpenStack for example provides an S3 API for storing objects.

The main benefit of storing things in an S3 compatible storage is that the cost of storage is much cheaper than many other alternatives. Many S3 implementations also provide reliable long-term storage.

Operations Allowed on S3 Tables

- ALTER TABLE S3 supports all types, keys and other options that are supported by the Aria engine. One can also perform ALTER TABLE on an S3 table to add or modify columns etc.
- DROP TABLE
- SELECT Any SELECT operations you can perform on a normal table should work with an S3 table.
- SHOW TABLES will show all tables that exist in the current defined S3 location.
- S3 tables can be part of partitions. See Discovery below.

Discovery

The S3 storage engine supports full MariaDB discovery. This means that if you have the S3 storage engine enabled and properly configured, the table stored in S3 will automatically be discovered when it's accessed with SHOW TABLES, SELECT or any other operation that tries to access it. In the case of SELECT, the .frm file from S3 will be copied to the local storage to speed up future accesses.

When an S3 table is opened for the first time (it's not in the table cache) and there is a local .frm file, the S3 engine will check if it's still relevant, and if not, update or delete the .frm file.

This means that if the table definition changes on S3 and it's in the local cache, one has to execute FLUSH TABLES to get MariaDB to notice the change and update the .frm file.

If partitioning S3 tables are used, the partition definitions will also be stored on S3 storage and are discovered by other servers.

Discovery of S3 tables is not done for tables in the mysql databases to make mariadbd boot faster and more securely.

Replication

S3 works with replication. One can use replication in two different scenarios:

- The primary and replica share the same S3 storage. In this case the primary will make all changes to the S3 data and the replica will ignore any changes in the replication stream to S3 data . This scenario is achieved by setting s3_slave_ignore_updates to 1.
- The primary and replica don't share the same S3 storage or the replica uses another storage engine for the S3 tables. This scenario is achieved by setting s3_slave_ignore_updates to 0.

aria_s3_copy

aria_s3_copy is an external tool that one can use to copy Aria tables to and from S3. Use `aria_s3_copy --help` to get the options of how to use it.

mariadb-dump

- mariadb-dump will by default ignore S3 tables. If `mariadb-dump` is run with the `--copy-s3-tables` option, the resulting file will contain a CREATE statement for a similar Aria table, followed by the table data and ending with an `ALTER TABLE xxx ENGINE=S3`.

ANALYZE TABLE

As of MariaDB 10.5.14, ANALYZE TABLE is supported for S3 tables.
As the S3 tables are read-only, a normal `ANALYZE TABLE` will not do anything. However using `ANALYZE TABLE table_name PERSISTENT FOR...` will now work.

CHECK TABLE

As of MariaDB 10.5.14, CHECK TABLE will work. As S3 tables are read only it is very unlikely that they can become corrupted. The only known way an S3 table could be corrupted if either the original table copied to S3 was corrupted or the process of copying the original table to S3 was somehow interrupted.

Current Limitations

- mysql-test-run doesn't by default test the S3 engine as we can't embed AWS keys into mysql-test-run.
- Replicas should not access S3 tables while they are ALTERed! This is because there is no locking implemented to S3 between servers. However, after a table (either the original S3 table or the partitioned S3 table) is changed on the primary, the replica will notice this on the next access and update its local definition.

Limitations in ALTER .. PARTITION

All ALTER PARTITION operations are supported on S3 partitioning tables except:

- REBUILD PARTITION
- TRUNCATE PARTITION
- REORGANIZE PARTITION

Performance Considerations

Depending on your connection speed to your S3 provider, there can be some notable slowdowns in some operations.

Discovery

As S3 is supporting discovery (automatically making tables available that are in S3) this can cause some small performance problems if the S3 engine is enabled. Partitioning S3 tables also support discovery.

- CREATE TABLE is a bit slower as the S3 engine has to check if the to-be-created table is already S3.
- Queries on information_schema tables are slower as S3 has to check if there is new tables in S3.
- DROP of non existing tables are slower as S3 has to check if the table is in S3.

There are no performance degradation's when accessing existing tables on the server. Accessing the S3 table the first time will copy the .frm file from S3 to the local disk, speeding up future accesses to the table.

Caching

- Accessing a table on S3 can take some time , especially if you are using big packets (s3_block_size). However the second access to the same data should be fast as it's then cached in the S3 page cache.

Things to Try to Increase Performance

If you have performance problems with the S3 engine, here are some things you can try:

- Decreasing s3_block_size. This can be done both globally and per table.
- Use COMPRESSION_ALGORITHM=zlib when creating the table. This will decrease the amount of data transferred from S3 to the local cache.
- Increasing the size of the s3 page cache: s3_pagecache_buffer_size

Try also to execute the query twice to check if the problem is that the data was not properly cached. When data is cached locally the performance should be excellent.

Future Development Ideas

- Store aws keys and region in the mysql.servers table (as Spider and FederatedX). This will allow one to have different tables on different S3 servers.
- Store s3 bucket, access_key and secret key in a cache to better be able to better to reuse connections. This would save some memory and make some S3 accesses a bit faster as we could reuse old connections.

Troubleshooting S3 on SELinux

If you get errors such as:

```
ERROR 3 (HY000): Got error from put_object(bubu/produkt/frm): 5 Couldn't connect to server
```

one reason could be that your system doesn't allow MariaDB to connect to ports other than 3306.
To procedure to enable other ports is the following:

Search for the ports allowed for MariaDB:

```
$ sudo semanage port -l | grep mysqld_port_t
mysqld_port_t                tcp   1186, 3306, 63132-63164
```

Say you want to allow MariaDB to connect to port 32768:

```
$ sudo semanage port -a -t mysqld_port_t -p tcp 32768
```

You can verify that the new port, 32768, is now allowed for MariaDB:

```
$ sudo semanage port -l | grep mysqld_port_t
mysqld_port_t                tcp   32768,1186, 3306, 63132-63164
```

See Also

- S3 storage engine internals

This page is licensed: CC BY-SA / Gnu FDL

S3 Status Variables

> **Note:**
>
> The S3 storage engine is available from MariaDB 10.5.4.

This page documents status variables related to the S3 storage engine. See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

See also the Full list of MariaDB options, system and status variables.

`S3_pagecache_blocks_not_flushed`

- Description:
- Scope:
- Data Type:
- Introduced: MariaDB 10.5

`S3_pagecache_blocks_unused`

- Description:
- Scope:
- Data Type:
- Introduced: MariaDB 10.5

`S3_pagecache_blocks_used`

- Description:
- Scope:
- Data Type:
- Introduced: MariaDB 10.5

`S3_pagecache_reads`

- Description:
- Scope:
- Data Type:
- Introduced: MariaDB 10.5

This page is licensed: CC BY-SA / Gnu FDL

S3 System Variables

> **Note:**
>
> The S3 storage engine is available from MariaDB 10.5.4.

This page documents system variables related to the S3 storage engine.

See Server System Variables for a complete list of system variables and instructions on setting system variables.

Also see the Full list of MariaDB options, system and status variables

Variables

`s3_access_key`

- Description: The AWS access key to access your data. See mariadbd startup options for S3.
- Command line: `--s3-access-key=val`
- Scope: Global
- Dynamic: No
- Data Type: String
- Default Value: (Empty)
- Introduced: MariaDB 10.5.4

`s3_block_size`

- Description: The default block size for a table, if not specified in CREATE TABLE. Set to 4M as default. See mariadbd startup options for S3.
- Command line: `--s3-block-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: Numeric
- Default Value: `4194304`
- Range: `4194304` to `16777216`
- Introduced: MariaDB 10.5.4

`s3_bucket`

- Description: The AWS bucket where your data should be stored. All MariaDB table data is stored in this bucket. See mariadbd startup options for S3.
- Command line: `--s3-bucket=val`
- Scope: Global
- Dynamic: No
- Data Type: String
- Default Value: `MariaDB`
- Introduced: MariaDB 10.5.4

`s3_debug`

- Description: Generates a trace file from libmarias3 on stderr (mysqld.err) for debugging the S3 protocol.
- Command line: `--s3-debug=[0|1]`
- Scope: Global
- Dynamic:
  - Yes (>= MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3, MariaDB 11.3.2, MariaDB 11.4.1)
  - No (<= MariaDB 10.6.16, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3, MariaDB 11.2.2, MariaDB 11.3.1)
- Data Type: Boolean
- Valid Values: 0 or 1
- Default Value: `0`

- Introduced: MariaDB 10.5.4

`s3_host_name`

- Description: Hostname for the S3 service. “s3.amazonaws.com”, Amazon S3 service, by default
- Command line: `--s3-host-name=val`
- Scope: Global
- Dynamic: No
- Data Type: String
- Default Value: `s3.amazonaws.com`
- Introduced: MariaDB 10.5.4

`s3_no_content_type`

- Description: If true (false is default), disables the Content-Type header, required for some providers.
- Command line: `--s3-no-content-type=[0|1]`
- Scope: Global
- Dynamic: No
- Data Type: Boolean
- Default Value: `0`
- Introduced: MariaDB 11.6.2

`s3_pagecache_age_threshold`

- Description: This characterizes the number of hits a hot block has to be untouched until it is considered aged enough to be downgraded to a warm block. This specifies the percentage ratio of that number of hits to the total number of blocks in the page cache.
- Command line: `--s3-pagecache-age-threshold=val`
- Scope: Global
- Dynamic: Yes
- Data Type: Numeric
- Default Value: `300`
- Range: `100` to `18446744073709551615`
- Introduced: MariaDB 10.5.4

`s3_pagecache_buffer_size`

- Description: The size of the buffer used for index blocks for S3 tables. Increase this to get better index handling (for all reads and multiple writes) to as much as you can afford. Size can be adjusted in blocks of `8192`.
- Command line: `--s3-pagecache-buffer-size=val`
- Scope: Global
- Dynamic: No
- Data Type: Numeric
- Default Value: `134217728` (128M)
- Range: `33554432` to `18446744073709551615`
- Introduced: MariaDB 10.5.4

`s3_pagecache_division_limit`

- Description: The minimum percentage of warm blocks in key cache.

- Command line: `--s3-pagecache-division-limit=val`
- Scope: Global
- Dynamic: Yes
- Data Type: Numeric
- Default Value: `100`
- Range: `1` to `100`
- Introduced: MariaDB 10.5.4

`s3_pagecache_file_hash_size`

- Description: Number of hash buckets for open files. Default 512. If you have a lot of S3 files open you should increase this for faster flush of changes. A good value is probably 1/10 of number of possible open S3 files.
- Command line: `--s3-pagecache-file-hash-size=#`
- Scope: Global
- Dynamic: No
- Data Type: Numeric
- Default Value: `512`
- Range: `32` to `16384`
- Introduced: MariaDB 10.5.4

`s3_port`

- Description: The TCP port number on the S3 host to connect to. A values of 0 means determine automatically.
- Command line: `--s3-port=#`
- Scope: Global
- Dynamic: No
- Data Type: Numeric
- Default Value: `0`
- Range: `0` to `65535`
- Introduced: MariaDB 10.5.7

`s3_protocol_version`

- Description: Protocol used to communication with S3. “Auto” is the default. If you get errors like “8 Access Denied” when you are connecting to another service provider, then try to change this option. The reason for this variable is that Amazon has changed some parts of the S3 protocol since they originally introduced it but other service providers are still using the original protocol. Can also be set numerically.
  - `Auto` (`0`):
  - `Original` or (`1`): Same as “Auto”. Deprecated from MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3, MariaDB 11.3.2, MariaDB 11.4.1.
  - `Amazon` (`2`) Same as “Auto”. Deprecated from MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3, MariaDB 11.3.2, MariaDB 11.4.1.
  - `Legacy` (`3`): v1 protocol. From MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3, MariaDB 11.3.2, MariaDB 11.4.1
  - `Path` (`4`): From MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3, MariaDB 11.3.2, MariaDB 11.4.1
  - `Domain` (`5`): From MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3, MariaDB 11.3.2, MariaDB 11.4.1
- Command line: `--s3-protocol-version=val`

- Scope: Global
- Dynamic: Yes
- Data Type: Enum
- Valid Values:
  - `Auto` (`0`), `Original` (`1`), `Amazon` (`2`), `Legacy` (`3`), `Path` (`4`), `Domain` (`5`) (>= MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3, MariaDB 11.3.2, MariaDB 11.4.1)
  - `Auto`, `Original`, `Amazon` (<= MariaDB 10.6.16, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3, MariaDB 11.2.2, MariaDB 11.3.1)
- Default Value: `Auto`
- Introduced: MariaDB 10.5.4

`s3_provider`

- Description: Enable S3 provider specific compatibility tweaks. “Default”, “Amazon”, or “Huawei”.
  - `Default`:
  - `Amazon`: Effectively sets s3_protocol_version to `5` (`Domain`), overriding any other setting it may have.
  - `Huawei`: Effectively sets s3_no_content_type to 1, overriding any other setting it may have.
- Command line: `--s3-provider=val`
- Scope: Global
- Dynamic: Yes
- Data Type: enum
- Valid Values: `Default`, `Amazon`, `Huawei`
- Default Value: `Default`
- Introduced: MariaDB 10.6.20, MariaDB 10.11.10, MariaDB 11.2.6, MariaDB 11.4.4, MariaDB 11.6.2

`s3_region`

- Description: The AWS region where your data should be stored. See mariadbd startup options for S3.
- Command line: `--s3-region=val`
- Scope: Global
- Dynamic: No
- Data Type: String
- Default Value: (Empty)
- Introduced: MariaDB 10.5.4

`s3_replicate_alter_as_create_select`

- Description: When converting S3 table to local table, log all rows in binary log. This allows the slave to replicate `CREATE TABLE .. SELECT FROM s3_table` even it the slave doesn’t have access to the original `s3_table`.
- Command line: `--s3-replicate-alter-as-create-select=[0|1]`
- Scope: Global
- Dynamic: No
- Data Type: Boolean
- Default Value: `1`
- Introduced: MariaDB 10.5.4

`s3_secret_key`

- Description: The AWS secret key to access your data. See mariadbd startup options for S3.

- Command line: `--s3-secret-key=val`
- Scope: Global
- Dynamic: No
- Data Type: String
- Default Value: (Empty)
- Introduced: MariaDB 10.5.4

`s3_slave_ignore_updates`

- Description: Should be set if master and slave share the same S3 instance. This tells the slave that it can ignore any updates to the S3 tables as they are already applied on the master.
- Command line: `--s3-slave-ignore-updates=[0|1]`
- Scope: Global
- Dynamic: No
- Data Type: Boolean
- Default Value: `0`
- Introduced: MariaDB 10.5.4

`s3_ssl_no_verify`

- Description: If true, SSL certificate verification for the S3 endpoint is disabled
- Command line: `--s3-ssl-no-verify=[0|1]`
- Scope: Global
- Dynamic: No
- Data Type: Boolean
- Default Value: `0`
- Introduced: MariaDB 10.6.20, MariaDB 10.11.10, MariaDB 11.2.6, MariaDB 11.4.4, MariaDB 11.6.2

`s3_use_http`

- Description: If enabled, HTTP are used instead of HTTPS.
- Command line: `--s3-use-http=[0|1]`
- Scope: Global
- Dynamic: No
- Data Type: Boolean
- Default Value: `0`
- Introduced: MariaDB 10.5.7

See Also

Using the S3 Storage Engine

This page is licensed: CC BY-SA / Gnu FDL

S3 Engine Internals

> **Note:**
>
> The S3 storage engine is available from MariaDB 10.5.4.

The S3 storage engine is based on the Aria code. Internally, the S3 storage inherits from the Aria code, with hooks that change reads, so that instead of reading data from the local disk it reads things from S3.

The S3 engine uses it's own page cache, modified to be able to handle reading blocks from S3 (of size `s3_block_size`). Internally the S3 page cache uses pages of aria-block-size for splitting the blocks read from S3.

ALTER TABLE

ALTER TABLE will first create a local table in the normal Aria on disk format and then move both index and data to S3 in buckets of `S3_BLOCK_SIZE`. The .frm file is also copied to S3 for discovery to support discovery for other MariaDB servers. You can also use `ALTER TABLE` to change the structure of an S3 table.

Partitioning Tables

S3 tables can also be used with Partitioning tables. All ALTER PARTITION operations are supported except:

- `REBUILD PARTITION`
- `TRUNCATE PARTITION`
- `REORGANIZE PARTITION`

Big Reads

One of the properties of many S3 implementations is that they favor large reads. It's said that 4M gives the best performance, which is why the default value for `S3_BLOCK_SIZE` is 4M.

Compression

If compression (`COMPRESSION_ALGORITHM=zlib`) is used, then all index blocks and data blocks are compressed. The `.frm` file and Aria definition header (first page/pages in the index file) are not compressed as these are used by discovery/open.

If compression is used, then the local block size is `S3_BLOCK_SIZE`, but the block stored in S3 are the size of the compressed block.

Typical compression we have seen is in the range of 80% saved space.

Structure Stored on S3

The table are copied in S3 into the following locations:

```
frm file (for discovery):
s3_bucket/database/table/frm

First index block (contains description of the Aria file):
s3_bucket/database/table/aria

Rest of the index file:
s3_bucket/database/table/index/block_number

Data file:
s3_bucket/database/table/data/block_number
```

`block_number` is a 6-digit decimal number, prefixed with `0`. (It can be larger than 6 numbers – the prefix is just for nice output.)

Using the awsctl Python Tool to Examine Data

Installing awsctl on Linux

```
# install python-pip (on an OpenSuse distribution)
# use the appropriate command for your distribution
zypper install python-pip
pip install --upgrade pip

# the following installs awscli tools in ~/.local/bin
pip install --upgrade --user awscli
export PATH=~/.local/bin:$PATH

# configure your aws credentials
aws configure
```

Using the awsctl Tool

One can use the `aws` python tool to see how things are stored on S3:

```
shell> aws s3 ls --recursive s3://mariadb-bucket/
2019-05-10 17:46:48       8192 foo/test1/aria
2019-05-10 17:46:49    3227648 foo/test1/data/000001
2019-05-10 17:46:48        942 foo/test1/frm
2019-05-10 17:46:48    1015808 foo/test1/index/000001
```

To delete an obsolete table `foo.test1` one can do:

```
shell> ~/.local/bin/aws s3 rm --recursive s3://mariadb-bucket/foo/test1
delete: s3://mariadb-bucket/foo/test1/aria
delete: s3://mariadb-bucket/foo/test1/data/000001
delete: s3://mariadb-bucket/foo/test1/frm
delete: s3://mariadb-bucket/foo/test1/index/000001
```

See Also

- Using the S3 storage engine

This page is licensed: CC BY-SA / Gnu FDL

Testing Connections to S3

> **Note:**
>
> The S3 storage engine is available from MariaDB 10.5.4.

If you can't get the S3 storage engine to work, here are some steps to help verify where the problem could be.

S3 Connection Variables

In most cases the problem is to correctly set the S3 connection variables.

Key S3 variables are:

- s3_access_key: The AWS access key to access your data
- s3_secret_key: The AWS secret key to access your data
- s3_bucket: The AWS bucket where your data should be stored. All MariaDB table data is stored in this bucket.
- s3_region: The AWS region where your data should be stored.
- s3_host_name: Hostname for the S3 service.
- s3_provider: Enable S3 provider specific compatibility tweaks. "Default", "Amazon", or "Huawei".

- s3_protocol_version: Protocol used to communicate with S3. One of “Amazon” or “Original”

There are several ways to ensure you get them right, detailed in the following sections.

Using aria_s3_copy to Test the Connection

aria_s3_copy is a tool that allows you to copy aria tables to and from S3. It’s useful for testing the connection as it allows you to specify all s3 options on the command line.

Execute the following sql commands to create a trivial sql table:

```
USE test;
CREATE TABLE s3_test (a INT) ENGINE=aria row_format=page transactional=0;
INSERT INTO s3_test VALUES (1),(2);
FLUSH TABLES s3_test;
```

Now you can use the aria_s3_copy tool to copy this to S3 from your shell/the command line:

```
shell> cd mariadb-data-directory/test
shell> aria_s3_copy --op=to --verbose --force --**other*options* s3_test.frm

Copying frm file s3_test.frm
Copying aria table: test.s3_test to s3
Creating aria table information test/s3_test/aria
Copying index information test/s3_test/index
Copying data information test/s3_test/data
```

As you can see from the above, aria_s3_copy is using the current directory as the database name.

You can also set the aria_s3_copy options in your my.cnf file to avoid some typing.

Using mariadb-test-run to Test the Connection and the S3 Storage Engine

One can use the MariaDB test system to run all default S3 test against your S3 storage.

To do that you have to locate the `mysql-test` directory in your system and `cd` to it.

The config file for the S3 test system can be found at `suite/s3/my.cnf`.
To enable testing you have to edit this file and add the s3 connection options to the end of the file. It should look something like this after editing:

```
!include include/default_mysqld.cnf
!include include/default_client.cnf

[mysqld.1]
s3=ON
#s3-host-name=s3.amazonaws.com
#s3-protocol-version=Amazon
s3-bucket=MariaDB
s3-access-key=
s3-secret-key=
s3-region=
```

You must give values for `s3-access-key`, `s3-secret-key` and `s3-region` that reflects your S3 provider. The `s3-bucket` name is defined by your administrator.

If you are not using Amazon Web Services as your S3 provider you must also specify `s3-hostname` and possibly change `s3-protocol-version` to “Original”.

Now you can test the configuration:

```
shell> cd **mysql-test** directory
shell> ./mysql-test-run --suite=s3
...
s3.no_s3                                 [ pass ]      5
s3.alter                                 [ pass ]  11073
s3.arguments                             [ pass ]   2667
s3.basic                                 [ pass ]   2757
s3.discovery                             [ pass ]   7851
s3.select                                [ pass ]   1325
s3.unsupported                           [ pass ]    363
```

Note that there may be more tests in your output as we are constantly adding more tests to S3 when needed.

Create a trace of the S3 connection to see what goes wrong

One can use the `s3_debug` variable to get a trace of the S3 engines interaction with the S3 storage. The trace is sent to the error log.

Here follows one example on can use to get a trace if `ALTER TABLE .. ENGINE=S3` fails:

```
USE test;
CREATE TABLE s3_test (a INT) ENGINE=aria row_format=page transactional=0;
INSERT INTO s3_test VALUES (1),(2);
SET @@global.s3_debug=1;
ALTER TABLE s3_test ENGINE=S3;
SET @@global.s3_debug=0;
```

If you have problems deciper the trace, you can always create a ticket on MariaDB Jira and explain the problem you have, including any errors. Don't forget to include the trace!

What to do when you have got things to work

When you got the connection to work, you should add the options to your global my.cnf file. Now you can start testing S3 from your mariadb command client by converting some existing table to S3 with ALTER TABLE … ENGINE=S3.

See Also

- Using the S3 Storage Engine
- Using MinIO with mysql-test-run to test the S3 storage engine

This page is licensed: CC BY-SA / Gnu FDL

aria_s3_copy

> **Note:**
>
> The S3 storage engine is available from MariaDB 10.5.4.

`aria_s3_copy` is a tool for copying an Aria table to and from S3.

The Aria table must be non transactional and have ROW_FORMAT=PAGE.

For `aria_s3_copy` to work reliably, the table should not be changed by the MariaDB server during the copy, and one should have first performed FLUSH TABLES to ensure that the table is properly closed.

Example of properly created Aria table:

```
CREATE TABLE test1 (a INT) transactional=0 row_format=PAGE engine=aria;
```

> **Note:**
>
> ALTER TABLE table_name ENGINE=S3 works for any kind of table. It internally converts the table to an Aria table, then moves it to S3 storage.

Main Arguments

Option	Description
-?, –help	Display this help and exit.
-k, –s3-access-key=name	AWS access key ID
-r, –s3-region=name	AWS region
-K, –s3-secret-key=name	AWS secret access key ID
-b, –s3-bucket=name	AWS prefix for tables
-h, –s3-host-name=name	Host name to S3 provider
-c, –compress	Use compression
-o, –op=name	Operation to execute. One of ‘from_s3’, ‘to_s3’ or ‘delete_from_s3’
-d, –database=name	Database for copied table (second prefix). If not given, the directory of the table file is used
-B, –s3-block-size=#	Block size for data/index blocks in s3
-L, –s3-protocol-version=name	Protocol used to communication with S3. One of “Amazon” or “Original”.
-f, –force	Force copy even if target exists
-v, –verbose	Write more information
-V, –version	Print version and exit.
-#, –debug[=name]	Output debug log. Often this is ‘d:t:o,filename’.
–s3-debug	Output debug log from marias3 to stdout

-p, –s3-port=#	Port number to connect to (0 means use default)
-P, –s3-use-http	If true, force use of HTTP protocol

Typical Configuration in a my.cnf File

```
[aria_s3_copy]
s3-bucket=mariadb
s3-access-key=xxxx
s3-secret-key=xxx
s3-region=eu-north-1
#s3-host-name=s3.amazonaws.com
#s3-protocol-version=Amazon
verbose=1
op=to
```

Example Usage

The following code will copy an existing Aria table named `test1` to S3.
If the `--database` option is not given, then the directory name where the table files exist are used as the database.

```
shell> aria_s3_copy --force --op=to --database=foo --compress --verbose --s3_block_size=4M test1
Delete of aria table: foo.test1
Delete of index information foo/test1/index
Delete of data information foo/test1/data
Delete of base information and frm
Copying frm file test1.frm
Copying aria table: foo.test1 to s3
Creating aria table information foo/test1/aria
Copying index information foo/test1/index
.
Copying data information foo/test1/data
.
```

When using `--verbose`, `aria_s3_copy` will write a dot for each #/79 part of the file copied.

See Also

Using the S3 storage engine. This pages has examples of .my.cnf entries for using `aria_s3_copy`.

This page is licensed: CC BY-SA / Gnu FDL

SphinxSE

> **Note:**
>
> SphinxSE is a community contribution to MariaDB Server, developed and maintained by the Sphinx developers. The Sphinx search daemon that it connects to is separate software, and isn't part of MariaDB Server.

See About SphinxSE.

SphinxSE is a storage engine that connects MariaDB to the Sphinx search daemon, enabling advanced full-text search capabilities within MariaDB.

See Installing Sphinx.

Instructions for installing the standalone Sphinx search server on various operating systems, a prerequisite for using the SphinxSE storage engine.

See Installing and Testing SphinxSE with MariaDB.

Learn how to install the SphinxSE plugin in MariaDB and run tests to verify the connection between the database and the Sphinx daemon.

See Configuring Sphinx.

A guide to configuring the Sphinx daemon (`searchd`) to index data from MariaDB, including setting up `sphinx.conf` and creating necessary users.

This page is licensed: CC BY-SA / Gnu FDL

About SphinxSE

The Sphinx storage engine (SphinxSE) is a storage engine that talks to `searchd` (the Sphinx daemon) to enable text searching. Sphinx and SphinxSE is used as a faster and more customizable alternative to MariaDB's built-in full-text search.

Sphinx does not depend on MariaDB, and can run independently, but SphinxSE provides a convenient interface to the underlying Sphinx daemon.

Enabling SphinxSE in MariaDB

The Sphinx storage engine is included in the source, binaries, and packages of MariaDB.

> **Note:**
>
> However, make sure the Sphinx daemon is installed on your system. Instructions for installing it are here.

SphinxSE is built as a dynamically loadable .so plugin. To use it, you need to perform a one-time install:

```
INSTALL SONAME 'ha_sphinx';
```

If you're getting a `Symbol not found` error, this is due to a server/plugin mismatch.

Once installed, SphinxSE will show up in the list of installed storage engines:

```
SHOW ENGINES;
+------------+---------+----------------------------------------------+--------------+------+------------+
| Engine     | Support | Comment                                      | Transactions | XA   | Savepoints |
+------------+---------+----------------------------------------------+--------------+------+------------+
...
| SPHINX     | YES     | Sphinx storage engine 0.9.9                  | NO           | NO   | NO         |
...
+------------+---------+----------------------------------------------+--------------+------+------------+
```

This is a one-time step and will not need to be performed again.

Note: SphinxSE is just the storage engine part of Sphinx. You will have to install Sphinx itself in order to make use of SphinxSE in MariaDB.

> **Note:**
>
> Despite the name, SphinxSE does not actually store any data itself. It is just a built-in client which allows MariaDB to talk to Sphinx, run search queries, and obtain search results. All indexing and searching happens outside MariaDB.

Some SphinxSE applications include:

- easier porting of MariaDB/MySQL FTS applications to Sphinx
- allowing Sphinx use with programming languages for which native APIs are not available yet
- optimizations when additional Sphinx result set processing on the MariaDB side is required (eg. JOINs with original document tables, additional
  MariaDB-side filtering, and etc…)

Using SphinxSE

Basic Usage

To search via SphinxSE, you would need to create a special `ENGINE=SPHINX` "search table", and then `SELECT` from it with full text query put into the `WHERE` clause for query column.

Here is an example create statement and search query:

```
CREATE TABLE t1
(
    id          BIGINT UNSIGNED NOT NULL,
    weight      INTEGER NOT NULL,
    query       VARCHAR(3072) NOT NULL,
    group_id    INTEGER,
    INDEX(query)
) ENGINE=SPHINX CONNECTION="sphinx://127.0.0.1:9312/test1";

SELECT * FROM t1 WHERE query='test it;mode=any';
```

The first three columns of the search table must have a type of `BIGINT` for the 1st column (document id), `INTEGER` or `BIGINT` for the 2nd column (match weight), and `VARCHAR` or `TEXT` for the 3rd column (your query), respectively. This mapping is fixed; you cannot omit any of these three required columns, or move them around, or change types. Also, the query column must be indexed; all the others must be kept unindexed. Column names are ignored so you can use arbitrary ones.

Additional columns must be either `INTEGER`, `TIMESTAMP`, `BIGINT`, `VARCHAR`, or `FLOAT`. They are bound to the attributes provided in the Sphinx result set by name, so their names must match the attribute names specified in `sphinx.conf`. If there's no such attribute name in the Sphinx search results, the additional columns will have `NULL` values.

Special "virtual" attribute names can also be bound to SphinxSE columns. `_sph_` needs to be used instead of `@` for that. For instance, to obtain the values of '`@groupby`', '`@count`', or '`@distinct`' virtual attributes, use '`_sph_groupby`', '`_sph_count`' or '`_sph_distinct`' column names, respectively.

The `CONNECTION` string parameter is used to specify the default `searchd` host, port, and indexes for queries issued using this table. If no connection string is specified in `CREATE TABLE`, index name '`*`' (ie. search all indexes) and '`127.0.0.1:9312`' are assumed. The connection string syntax is as follows:

```
CONNECTION="sphinx://HOST:PORT/INDEXNAME"
```

You can change the default connection string later like so:

```
ALTER TABLE t1 CONNECTION="sphinx://NEWHOST:NEWPORT/NEWINDEXNAME";
```

You can also override all these parameters per-query.

Note: To use Linux sockets you can modify the searchd section of the Sphinx configuration file, setting the listen parameter to a socket file. Instruct SphinxSE about the socket using CONNECTION="unix:*unix/domain/socket[:index]".*

Search Options

As seen in the example above, both query text and search options should be put into the '`WHERE`' clause of the search query column (i.e. the 3rd column); the options are separated by semicolons ('`;`') and separate names from values using an equals sign ('`=`'). Any number of options can be specified. Available options are:

- query - query text;
- mode - matching mode. Must be one of "all", "any", "phrase", "boolean", or "extended". Default is "all";
- sort - match sorting mode. Must be one of "relevance", "attr_desc", "attr_asc", "time_segments", or "extended". In all modes besides "relevance" attribute name (or sorting clause for "extended") is also required after a colon:

```
... WHERE query='test;sort=attr_asc:group_id';
... WHERE query='test;sort=extended:@weight desc, group_id asc';
```

- offset - offset into result set, default is 0;
- limit - amount of matches to retrieve from result set, default is 20;
- index - names of the indexes to search:

```
... WHERE query='test;index=test1;';
... WHERE query='test;index=test1,test2,test3;';
```

- minid, maxid - min and max document ID to match;
- weights - comma-separated list of weights to be assigned to Sphinx full-text fields:

```
... WHERE query='test;weights=1,2,3;';
```

- filter, !filter - comma-separated attribute name and a set of values to match:

```
# only include groups 1, 5 and 19
... WHERE query='test;filter=group_id,1,5,19;';

# exclude groups 3 and 11
... WHERE query='test;!filter=group_id,3,11;';
```

- range, !range - comma-separated attribute name, min and max value to match:

```
# include groups from 3 to 7, inclusive
... WHERE query='test;range=group_id,3,7;';

# exclude groups from 5 to 25
... WHERE query='test;!range=group_id,5,25;';
```

- maxmatches - per-query max matches value:

```
... WHERE query='test;maxmatches=2000;';
```

- groupby - group-by function and attribute:

```
... WHERE query='test;groupby=day:published_ts;';
... WHERE query='test;groupby=attr:group_id;';
```

- groupsort - group-by sorting clause:

```
... WHERE query='test;groupsort=@count desc;';
```

- indexweights - comma-separated list of index names and weights to use when searching through several indexes:

```
... WHERE query='test;indexweights=idx_exact,2,idx_stemmed,1;';
```

- comment - a string to mark this query in query log (mapping to $comment parameter in Query() API call):

```
... WHERE query='test;comment=marker001;';
```

- select - a string with expressions to compute (mapping to SetSelect() API call):

```
... WHERE query='test;select=2*a+3*b as myexpr;';
```

Note

It is much more efficient to allow Sphinx to perform sorting, filtering, and slicing of the result set than to raise max matches count and use '`WHERE`', '`ORDER BY`', and '`LIMIT`' clauses on the MariaDB side. This is for two reasons:

1. Sphinx does a number of optimizations and performs better than MariaDB/MySQL on these tasks.
2. Less data would need to be packed by `searchd`, and transferred and unpacked by SphinxSE.

SHOW ENGINE SPHINX STATUS

Starting with version 0.9.9-rc1, additional query info besides the result set can be retrieved with the '`SHOW ENGINE SPHINX STATUS`' statement:

```
SHOW ENGINE SPHINX STATUS;
+--------+-------+-----------------------------------------------+
| Type   | Name  | Status                                        |
+--------+-------+-----------------------------------------------+
| SPHINX | stats | total: 25, total found: 25, time: 126, words: 2 |
| SPHINX | words | sphinx:591:1256 soft:11076:15945              |
+--------+-------+-----------------------------------------------+
```

This information can also be accessed through status variables. Note that this method does not require super-user privileges.

```
SHOW STATUS LIKE 'sphinx_%';
+--------------------+---------------------------------+
| Variable_name      | Value                           |
+--------------------+---------------------------------+
| sphinx_total       | 25                              |
| sphinx_total_found | 25                              |
| sphinx_time        | 126                             |
| sphinx_word_count  | 2                               |
| sphinx_words       | sphinx:591:1256 soft:11076:15945 |
+--------------------+---------------------------------+
```

JOINs with SphinxSE

You can perform `JOIN`s on a SphinxSE search table and tables using other engines. Here's an example with "documents" from example.sql:

```
SELECT content, date_added FROM test.documents docs
    JOIN t1 ON (docs.id=t1.id)
    WHERE query="one document;mode=any";
+-----------------------------------+---------------------+
| content                           | docdate             |
+-----------------------------------+---------------------+
| this is my test document number two | 2006-06-17 14:04:28 |
| this is my test document number one | 2006-06-17 14:04:28 |
+-----------------------------------+---------------------+

SHOW ENGINE SPHINX STATUS;
+--------+-------+-----------------------------------------------+
| Type   | Name  | Status                                        |
+--------+-------+-----------------------------------------------+
| SPHINX | stats | total: 2, total found: 2, time: 0, words: 2 |
| SPHINX | words | one:1:2 document:2:2                          |
+--------+-------+-----------------------------------------------+
```

Building snippets (excerpts) via MariaDB

Starting with version 0.9.9-rc2, SphinxSE also includes a UDF function that lets you create snippets through MariaDB. The functionality is fully similar to the BuildExcerprts API call but is accessible through MariaDB+SphinxSE.

The function name must be '`sphinx_snippets`', you can not use an arbitrary name. Function arguments are as follows:

```
Prototype: function sphinx_snippets ( document, index, words, [options] );
```

Document and words arguments can be either strings or table columns. Options must be specified like this: `'value' AS option_name`. For a list of supported options, refer to the BuildExcerprts() API call. The only UDF-specific additional option is named 'sphinx' and lets you specify searchd location (host and port).

Usage examples:

```
SELECT sphinx_snippets('hello world doc', 'main', 'world',
    'sphinx://192.168.1.1/' AS sphinx, true AS exact_phrase,
    '[b]' AS before_match, '[/b]' AS after_match)
FROM documents;

SELECT title, sphinx_snippets(text, 'index', 'mysql php') AS text
    FROM sphinx, documents
    WHERE query='mysql php' AND sphinx.id=documents.id;
```

More Information

More information on Sphinx and SphinxSE is available on the Sphinx website.

This page is licensed: CC BY-SA / Gnu FDL

Installing Sphinx

In order to use the Sphinx Storage Engine, it is necessary to install the Sphinx daemon.

Many Linux distributions have Sphinx in their repositories. These can be used to install Sphinx instead of following the instructions below, but these are usually quite old versions and don't all include API's for easy integration. Ubuntu users can use the updated repository at sphinxsearch-rel21 (see instructions below).

Debian and Ubuntu

Ubuntu users can make use of the repository, as follows:

```
sudo add-apt-repository ppa:builds/sphinxsearch-rel21
sudo apt-get update
sudo apt-get install sphinxsearch
```

Alternatively, install as follows:

- The Sphinx package and daemon are named `sphinxsearch`.
- `sudo apt-get install unixodbc libpq5 mariadb-client`
- `sudo dpkg -i sphinxsearch*.deb`
- Configure Sphinx as required
- You may need to check `/etc/default/sphinxsearch` to see that `START=yes`
- Start with `sudo service sphinxsearch start` (and stop with `sudo service sphinxsearch stop`)

Red Hat and CentOS

- The package name is `sphinx` and the daemon `searchd`.
- `sudo yum install postgresql-libs unixODBC`
- `sudo rpm -Uhv sphinx*.rpm`
- Configure Sphinx as required
- `service searchd start`

Windows

- Unzip and extract the downloaded zip file
- Move the extracted directory to `C:\Sphinx`
- Configure Sphinx as required
- Install as a service:
  - `C:\Sphinx\bin> C:\Sphinx\bin\searchd --install --config C:\Sphinx\sphinx.conf.in --servicename SphinxSearch`

Once Sphinx has been installed, it will need to be configured.

Full instructions, including details on compiling Sphinx yourself, are available at current.html.

This page is licensed: CC BY-SA / Gnu FDL

Installing and Testing SphinxSE with MariaDB

To use SphinxSE with MariaDB you need to first download and install Sphinx.

Complete Sphinx documentation is available on the Sphinx website.

Tips for Installing Sphinx

libexpat

One library we know you will need on Linux before you can install Sphinx is `libexpat`. If it is not installed, you will get the warning `checking for libexpat... not found`. On Suse Linux, the package is called `libexpat-devel`, on Ubuntu the package is called `libexpat1-dev`.

MariaDB detection

If you run into problems with MariaDB not being detected, use the following options:

```
--with-mysql            compile with MySQL support (default is enabled)
 --with-mysql-includes   path to MySQL header files
 --with-mysql-libs       path to MySQL libraries
```

The above will tell the `configure` script where your MySQL/MariaDB installation is.

Testing Sphinx

After installing Sphinx, you can check that things are working in MariaDB by doing the following:

```
cd installation-dir/mysql-test
./mysql-test-run --suite=sphinx
```

If the above test doesn't pass, check the logs in the `'var'` directory. If there is a problem with the sphinx installation, the reason can probably be found in the log file at `var/log/sphinx.sphinx/searchd/sphinx.log`.

This page is licensed: CC BY-SA / Gnu FDL

Configuring Sphinx

Before you can get Sphinx working with the Sphinx Storage Engine on MariaDB, you need to configure it.

- The default configuration file is called `sphinx.conf`, usually located in `/etc/sphinxsearch` (Debian/Ubuntu), `/etc/sphinx/sphinx.conf.` (Red Hat/CentOS) or `C:\Sphinx\sphinx.conf` (Windows).

If it doesn't already exist, you can use the sample configuration file, `sphinx.conf.dist`. There is also sample data supplied that we can use for testing. Load the sample data (which creates two tables, `documents` and `tags` in the `test` database), for example:

`mysql -u test < /usr/local/sphinx/etc/example.sql` (Red Hat, CentOS) `mysql -u test < /usr/share/doc/sphinxsearch/example-conf/example.sql` (Debian/Ubuntu)

The sample configuration file documents the available options. You will need to make at least a few changes. A MariaDB user with permission to access the database must be created:

```
CREATE USER 'sphinx'@localhost
  IDENTIFIED BY 'sphinx_password';
GRANT SELECT ON test.* TO 'sphinx'@localhost;
```

Add these details to the `mysql` section of the config file:

```
sql_host = localhost
sql_user = sphinx
sql_pass = sphinx_password
sql_db   = test
sql_port = 3306
```

On Windows, the `path` and `pid` lines will need to be changed to reflect a valid path, usually as follows:

```
path = C:\Sphinx\docsidx
...
pid_file = C:\Sphinx\sphinx.pid
```

The query in the configuration files is the query that are used for building the index. In the sample data, this is:

```
sql_query = \
  SELECT id, group_id, UNIX_TIMESTAMP(date_added) AS date_added, title, content \
  FROM documents
```

This page is licensed: CC BY-SA / Gnu FDL

Spider

See Spider Storage Engine Overview.

An introduction to the Spider storage engine, which provides built-in sharding by linking to tables on remote MariaDB servers, supporting partitioning and XA transactions.

See Spider Storage Engine Introduction.

Explains the architecture of Spider, where a “Spider node” processes queries and distributes them to one or more “Data nodes” that actually store the data.

See Spider Differences Between SpiderForMySQL and MariaDB.

This page outlines the differences between the standalone SpiderForMySQL distribution and the version integrated into MariaDB Server, including version correspondence and feature availability.

See Spider Feature Matrix.

A matrix listing the features supported by Spider, including sharding, partitioning, XA transactions, and support for various SQL statements and functions.

See Spider Use Cases.

Describes common use cases for Spider, such as horizontal sharding for scalability, consolidating data from multiple sources, and migrating data between servers.

See Spider Installation.

A guide to installing the Spider storage engine on Debian/Ubuntu and other Linux distributions, including loading the plugin and configuring data nodes.

See Spider Cluster Management.

Covers advanced management topics like executing direct SQL on backends, copying tables between nodes, and monitoring the cluster using status variables.

See Spider Case Studies.

A list of real-world companies and projects using Spider for high-volume data handling, gaming, and analytics, illustrating its scalability.

See Spider Benchmarks.

Performance benchmark results for Spider, demonstrating its throughput and latency characteristics under various workloads compared to other configurations.

See Information Schema SPIDER_WRAPPER_PROTOCOLS Table.

Describes the SPIDER_WRAPPER_PROTOCOLS table, which lists the available foreign data wrappers (like `mysql`) that Spider can use to connect to remote servers.

See Spider Storage Engine Core Concepts.

Explains the fundamental concepts behind Spider, including its architecture as a proxy storage engine, sharding capabilities, and support for XA transactions across data nodes.

See Spider System Variables.

Comprehensive list of system variables to configure Spider globally or per session, affecting connection timeouts, buffering, and query pushdown strategies.

See Spider Table Parameters.

A reference for table-level parameters in Spider, which can be set via the COMMENT or CONNECTION string to control connection settings, monitoring, and query behavior.

See Spider Functions.

Explore Spider functions in MariaDB Server. Learn about the specialized functions that enhance data access and manipulation across sharded and distributed databases using the Spider storage engine.

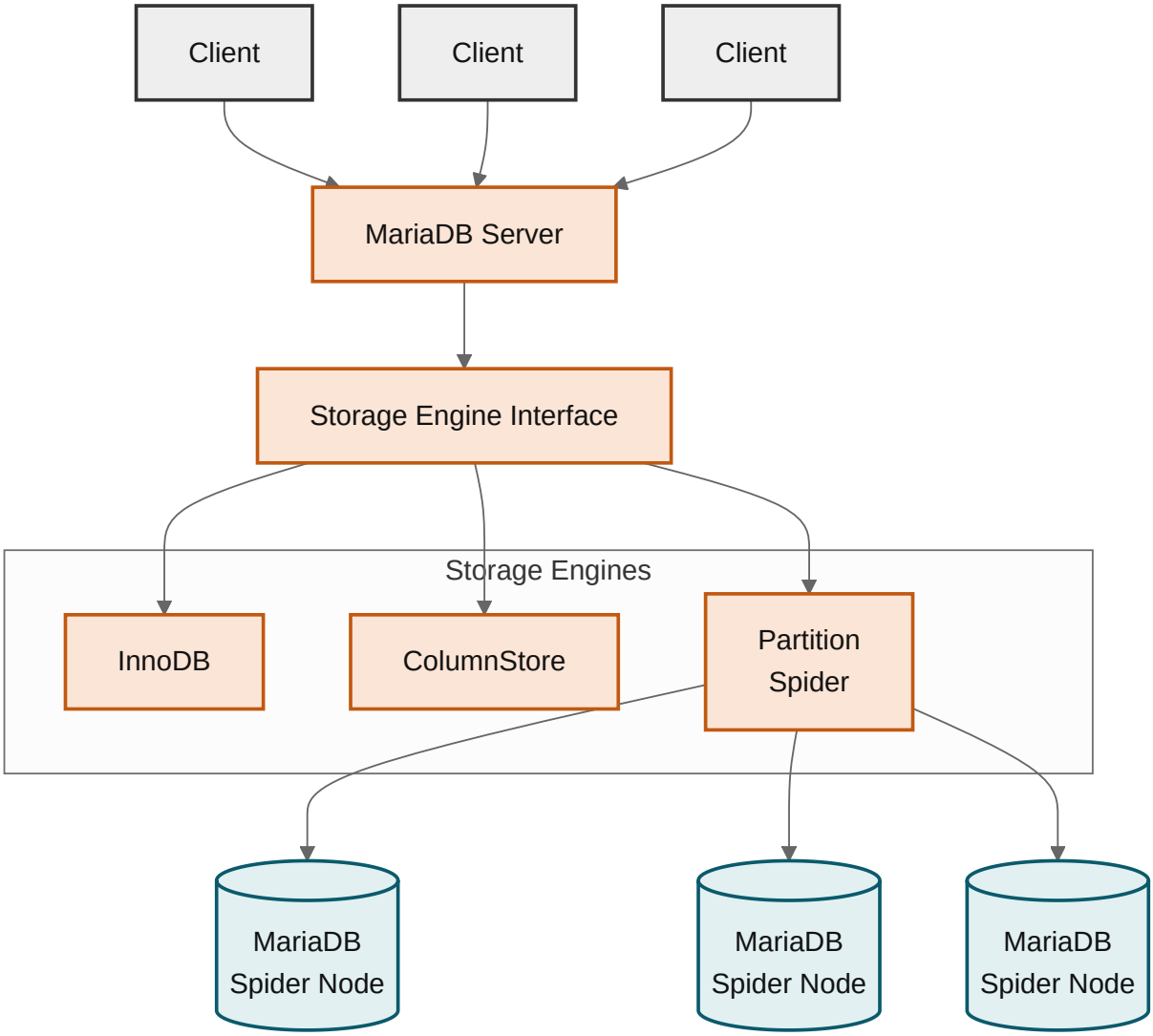
See Spider FAQ.

Frequently asked questions about Spider, covering troubleshooting common errors, configuration best practices, and architectural questions regarding HA and sharding.

This page is licensed: CC BY-SA / Gnu FDL

Spider Storage Engine Overview

About



The Spider storage engine sits alongside InnoDB and ColumnStore in the MariaDB Server's storage engine interface, and links out to multiple remote MariaDB Spider Node servers via Partition.

The Spider storage engine is a storage engine with built-in sharding features. It supports partitioning and xa transactions, and allows tables of different MariaDB instances to be handled as if they were on the same instance. It refers to one possible implementation of ISO/IEC 9075-9:2008 SQL/MED.

When a table is created with the Spider storage engine, the table links to the table on a remote server. The remote table can be of any storage engine. The table link is concretely achieved by the establishment of the connection from a local MariaDB server to a remote MariaDB server. The link is shared for all tables that are part of a the same transaction.

The Spider documentation on the MariaDB documentation is currently incomplete. See the Spider website for more:, as well as the spider-1.0-doc and spider-2.0-doc repositories.

Spider Versions in MariaDB

Spider Version	Introduced	Maturity
Spider 3.3.15	MariaDB 10.5.7, MariaDB 10.4.6	Stable

Spider Version	Introduced	Maturity
Spider 3.3.15	MariaDB 10.5.4	Gamma
Spider 3.3.14	MariaDB 10.4.3, MariaDB 10.3.13	Stable
Spider 3.3.13	MariaDB 10.3.7	Stable
Spider 3.3.13	MariaDB 10.3.3	Gamma
Spider 3.2.37	MariaDB 10.1.10, MariaDB 10.0.23	Gamma
Spider 3.2.21	MariaDB 10.1.5, MariaDB 10.0.18	Gamma
Spider 3.2.18	MariaDB 10.0.17	Gamma
Spider 3.2.11	MariaDB 10.0.14	Gamma
Spider 3.2.4	MariaDB 10.0.12	Gamma
Spider 3.2	MariaDB 10.0.11	Gamma
Spider 3.0	MariaDB 10.0.4	Beta

Usage

Basic Usage

To create a table in the Spider storage engine format, the COMMENT and/or CONNECTION clauses of the CREATE TABLE statement are used to pass connection information about the remote server.

For example, the following table exists on a remote server (in this example, the remote node was created with the MySQL Sandbox tool, an easy way to test with multiple installations)::

```
node1 >CREATE TABLE s(
  id INT NOT NULL AUTO_INCREMENT,
  code VARCHAR(10),
  PRIMARY KEY(id));
```

On the local server, a Spider table can be created as follows:

```
CREATE TABLE s(
  id INT NOT NULL AUTO_INCREMENT,
  code VARCHAR(10),
  PRIMARY KEY(id)
)
ENGINE=SPIDER
COMMENT='host "127.0.0.1", user "msandbox", password "msandbox", port "8607"';
```

Records can now be inserted on the local server, and they are stored on the remote server:

```
INSERT INTO s(code) VALUES ('a');

node1 > SELECT * FROM s;
+----+------+
| id | code |
+----+------+
|  1 | a    |
+----+------+
```

MariaDB starting with 10.8.1

Alternative to specifying the data node information in the COMMENT, certain information (server, database, table) can also be specified using Table Options, like so:

```
CREATE SERVER srv FOREIGN DATA WRAPPER mysql OPTIONS(
  HOST '127.0.0.1',
  USER 'msandbox',
  PASSWORD 'msandbox',
  PORT 8607);

CREATE TABLE s(
  id INT NOT NULL AUTO_INCREMENT,
  code VARCHAR(10),
  PRIMARY KEY(id)
)
ENGINE=SPIDER REMOTE_SERVER="srv" REMOTE_DATABASE="db" REMOTE_TABLE="s";
```

Further Examples

Preparing 10M record table using the sysbench utility

```
/usr/local/skysql/sysbench/bin/sysbench --test=oltp  --db-driver=mysql  --mysql-table-engine=innodb --
mysql-user=skysql --mysql-password=skyvodka --mysql-host=192.168.0.202 --mysql-port=5054 --oltp-table-
size=10000000  --mysql-db=test  prepare
```

Make a first read only benchmark to check the initial single node performance.

```
/usr/local/skysql/sysbench/bin/sysbench --test=oltp  --db-driver=mysql  --mysql-table-engine=innodb --
mysql-user=skysql --mysql-password=skyvodka --mysql-host=192.168.0.202 --mysql-port=5054 --mysql-db=test
--oltp-table-size=10000000 --num-threads=4 --max-requests=100000 --oltp-read-only=on run
```

```
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 4

Doing OLTP test.
Running mixed OLTP test
Doing read-only test
Using Special distribution (12 iterations,  1 pct of values are returned in 75 pct cases)
Using "BEGIN" for starting transactions
Using auto_inc on the id column
Maximum number of requests for OLTP test is limited to 100000
Threads started!
Done.

OLTP test statistics:
    queries performed:
        read:                            1400196
        write:                           0
        other:                           200028
        total:                           1600224
    transactions:                        100014 (1095.83 per sec.)
    deadlocks:                           0      (0.00 per sec.)
    read/write requests:                 1400196 (15341.58 per sec.)
    other operations:                    200028 (2191.65 per sec.)

Test execution summary:
    total time:                          91.2681s
    total number of events:              100014
    total time taken by event execution: 364.3693
    per-request statistics:
         min:                                  1.85ms
         avg:                                  3.64ms
         max:                                 30.70ms
         approx.  95 percentile:               4.66ms

Threads fairness:
    events (avg/stddev):           25003.5000/84.78
    execution time (avg/stddev):   91.0923/0.00
```

Define an easy way to access the nodes from the MariaDB or MySQL client.

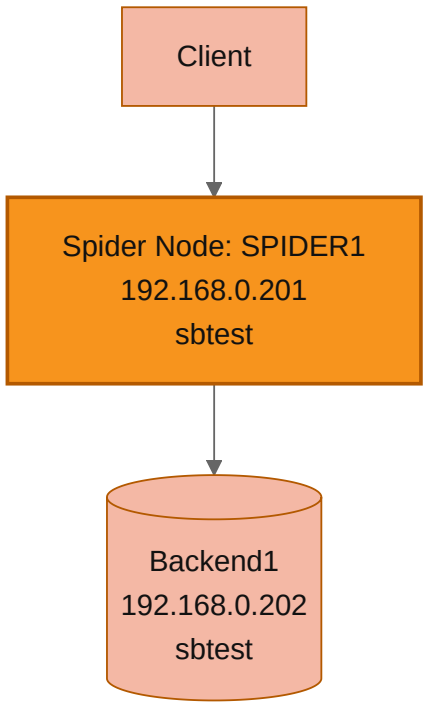
```
alias backend1='/usr/local/skysql/mysql-client/bin/mysql  --user=skysql --password=skyvodka --
host=192.168.0.202 --port=5054'
alias backend2='/usr/local/skysql/mysql-client/bin/mysql  --user=skysql --password=skyvodka --
host=192.168.0.203 --port=5054'
alias spider1='/usr/local/skysql/mysql-client/bin/mysql  --user=skysql --password=skyvodka --
host=192.168.0.201 --port=5054'
```

Create the empty tables to hold the data and repeat for all available backend nodes.

```
backend1 << EOF
CREATE DATABASE backend;
CREATE TABLE backend.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
EOF

backend2 << EOF
CREATE DATABASE backend;
CREATE TABLE backend.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
EOF
```

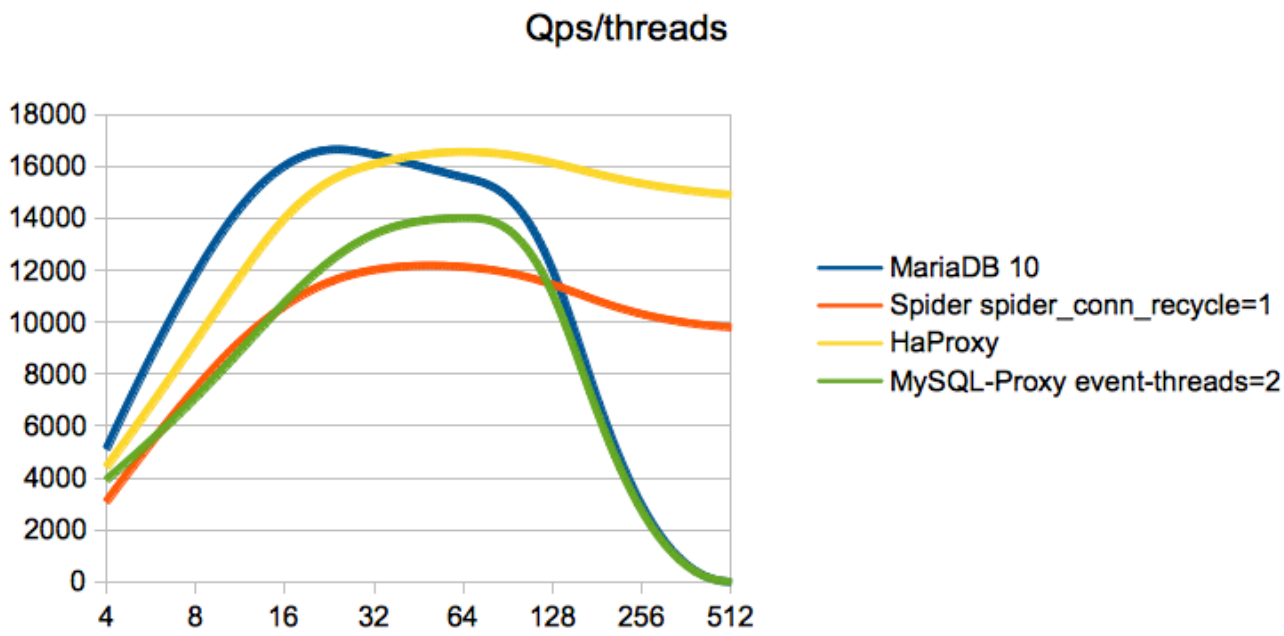
Federation Setup



A client talks to a single Spider node (SPIDER1), which federates the sbtest table to one remote backend server (BACKEND1).

```
spider1 << EOF
CREATE SERVER backend
  FOREIGN DATA WRAPPER mysql
OPTIONS(
  HOST '192.168.0.202',
  DATABASE 'test',
  USER 'skysql',
  PASSWORD 'skyvodka',
  PORT 5054
);

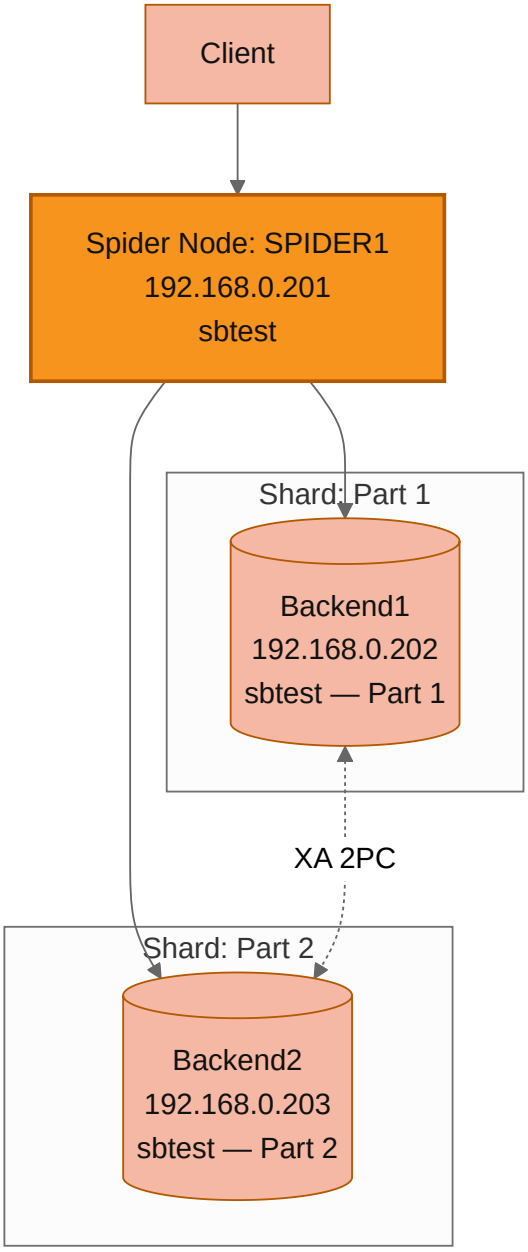
CREATE  TABLE test.sbtest
(
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=spider COMMENT='wrapper "mysql",srv "backend"';
SELECT * FROM test.sbtest LIMIT 10;
EOF
```



spbench10

Without connection pool or MariaDB thread pool, HaProxy and Spider have been protecting the tcp socket overflow without specific TCP tuning. In reality with a well tuned TCP stack or thread pool the curve should not decrease so abruptly to 0. Refer to the MariaDB Thread Pool to explore this feature.

Sharding Setup



A single Spider node (SPIDER1) shards sbtest across two backends, Backend1 (Part 1) and Backend2 (Part 2), coordinated by XA 2PC.

Create the spider table on the Spider Node

```
#spider1 << EOF
CREATE SERVER backend1
  FOREIGN DATA WRAPPER mysql
OPTIONS(
  HOST '192.168.0.202',
  DATABASE 'backend',
  USER 'skysql',
  PASSWORD 'skyvodka',
  PORT 5054
);
CREATE SERVER backend2
  FOREIGN DATA WRAPPER mysql
OPTIONS(
  HOST '192.168.0.203',
  DATABASE 'backend',
  USER 'skysql',
  PASSWORD 'skyvodka',
  PORT 5054
);
CREATE DATABASE IF NOT EXISTS backend;
CREATE  TABLE backend.sbtest
(
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=spider COMMENT='wrapper "mysql", TABLE "sbtest"'
 PARTITION BY KEY (id)
(
 PARTITION pt1 COMMENT = 'srv "backend1"',
 PARTITION pt2 COMMENT = 'srv "backend2"'
) ;
EOF
```

Copy the data from the original sysbench table to the spider table

```
#/usr/local/skysql/mariadb/bin/mysqldump  --user=skysql --password=skyvodka --host=192.168.0.202 --
port=5054 --no-create-info test sbtest | spider1 backend

#backend2 -e"SELECT count(*) FROM backend.sbtest;"
+----------+
| count(*) |
+----------+
|  3793316 |
+----------+
#backend1 -e"SELECT count(*) FROM backend.sbtest;"
+----------+
| count(*) |
+----------+
|  6206684 |
+----------+
```

We observe a common issue with partitioning is a non uniform distribution of data between the backends. based on the partition key hashing algorithm.

Rerun the Benchmark with less queries

```
#/usr/local/skysql/sysbench/bin/sysbench --test=oltp  --db-driver=mysql  --mysql-table-engine=innodb --
mysql-user=skysql --mysql-password=skyvodka --mysql-host=192.168.0.201 --mysql-port=5054 --mysql-
db=backend --mysql-engine-trx=yes --oltp-table-size=10000000 --num-threads=4 --max-requests=100 --oltp-
read-only=on run
```

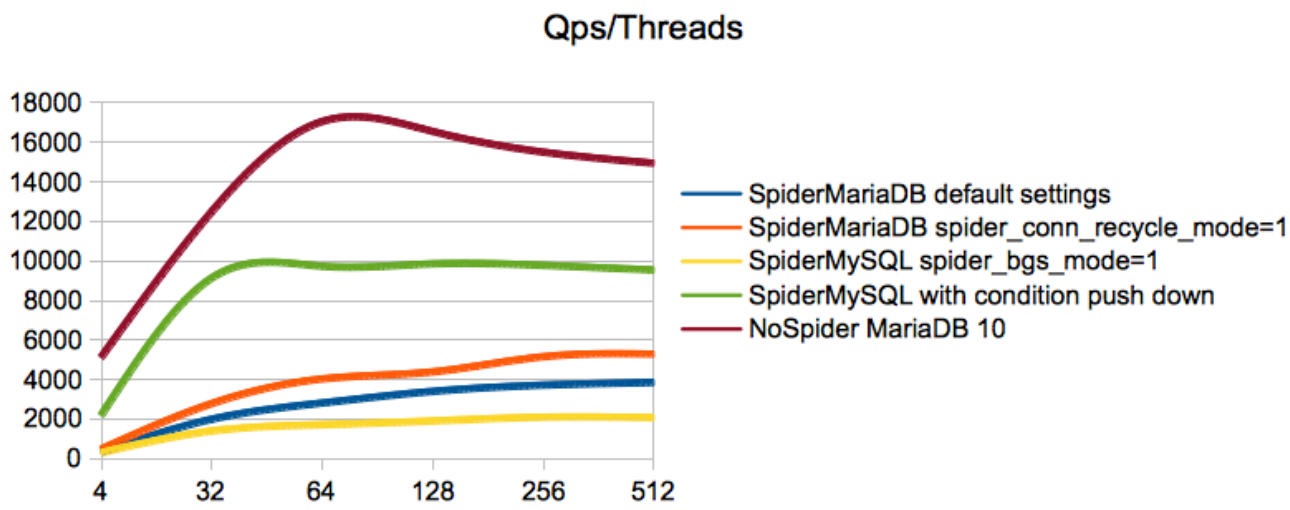
```
OLTP test statistics:
    queries performed:
        read:                            1414
        write:                           0
        other:                           202
        total:                           1616
    transactions:                        101    (22.95 per sec.)
    deadlocks:                           0      (0.00 per sec.)
    read/write requests:                 1414   (321.30 per sec.)
    other operations:                    202    (45.90 per sec.)

Test execution summary:
    total time:                          4.4009s
    total number of events:              101
    total time taken by event execution: 17.2960
    per-request statistics:
         min:                                  114.48ms
         avg:                                  171.25ms
         max:                                  200.98ms
         approx.  95 percentile:               195.12ms

Threads fairness:
    events (avg/stddev):           25.2500/0.43
    execution time (avg/stddev):   4.3240/0.04
```

The response time decreases to 0.04.
This is expected because the query latency is increased from multiple network round trips and condition push down is not implemented yet. Sysbench doing a lot of range queries. Just consider for now that this range query can be a badly optimized query.

We need to increase the concurrency to get better throughput.



spbench11

Background Setup

We have no background search available in MariaDB. It won't be available before MariaDB 10.2, but the next table definition mainly enables improving the performance of a single complex query plan with background search that can be found via the upstream spiral binaries MariaDB branch.

We have 4 cores per backend and 2 backends .

On `backend1`

```
#backend1 << EOF
CREATE DATABASE bsbackend1;
CREATE DATABASE bsbackend2;
CREATE DATABASE bsbackend3;
CREATE DATABASE bsbackend4;
CREATE TABLE bsbackend1.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
CREATE TABLE bsbackend2.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
CREATE TABLE bsbackend3.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
CREATE TABLE bsbackend4.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
EOF
```

On `backend2`

```
#backend2 << EOF
CREATE DATABASE bsbackend5;
CREATE DATABASE bsbackend6;
CREATE DATABASE bsbackend7;
CREATE DATABASE bsbackend8;
CREATE TABLE bsbackend5.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
CREATE TABLE bsbackend6.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
CREATE TABLE bsbackend7.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
CREATE TABLE bsbackend8.sbtest (
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=InnoDB;
EOF
```

On `Spider Node`

```
#spider2 << EOF
CREATE SERVER bsbackend1 FOREIGN DATA WRAPPER mysql OPTIONS( HOST '192.168.0.202', DATABASE
'bsbackend1',USER 'skysql', PASSWORD 'skyvodka',PORT 5054);
CREATE SERVER bsbackend2 FOREIGN DATA WRAPPER mysql OPTIONS( HOST '192.168.0.202', DATABASE
'bsbackend2',USER 'skysql', PASSWORD 'skyvodka',PORT 5054);
CREATE SERVER bsbackend3 FOREIGN DATA WRAPPER mysql OPTIONS( HOST '192.168.0.202', DATABASE
'bsbackend3',USER 'skysql', PASSWORD 'skyvodka',PORT 5054);
CREATE SERVER bsbackend4 FOREIGN DATA WRAPPER mysql OPTIONS( HOST '192.168.0.202', DATABASE
'bsbackend4',USER 'skysql', PASSWORD 'skyvodka',PORT 5054);
CREATE SERVER bsbackend5 FOREIGN DATA WRAPPER mysql OPTIONS( HOST '192.168.0.203', DATABASE
'bsbackend5',USER 'skysql', PASSWORD 'skyvodka',PORT 5054);
CREATE SERVER bsbackend6 FOREIGN DATA WRAPPER mysql OPTIONS( HOST '192.168.0.203', DATABASE
'bsbackend6',USER 'skysql', PASSWORD 'skyvodka',PORT 5054);
CREATE SERVER bsbackend7 FOREIGN DATA WRAPPER mysql OPTIONS( HOST '192.168.0.203', DATABASE
'bsbackend7',USER 'skysql', PASSWORD 'skyvodka',PORT 5054);
CREATE SERVER bsbackend8 FOREIGN DATA WRAPPER mysql OPTIONS( HOST '192.168.0.203', DATABASE
'bsbackend8',USER 'skysql', PASSWORD 'skyvodka',PORT 5054);

CREATE DATABASE IF NOT EXISTS bsbackend;
CREATE  TABLE bsbackend.sbtest
(
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=spider COMMENT='wrapper "mysql", TABLE "sbtest"'
 PARTITION BY KEY (id)
(
 PARTITION pt1 COMMENT = 'srv "bsbackend1"',
 PARTITION pt2 COMMENT = 'srv "bsbackend2"',
 PARTITION pt3 COMMENT = 'srv "bsbackend3"',
 PARTITION pt4 COMMENT = 'srv "bsbackend4"',
 PARTITION pt5 COMMENT = 'srv "bsbackend5"',
 PARTITION pt6 COMMENT = 'srv "bsbackend6"',
 PARTITION pt7 COMMENT = 'srv "bsbackend7"',
 PARTITION pt8 COMMENT = 'srv "bsbackend8"'
) ;
EOF
INSERT INTO  bsbackend.sbtest SELECT * FROM backend.sbtest;
```

Now test the following query :

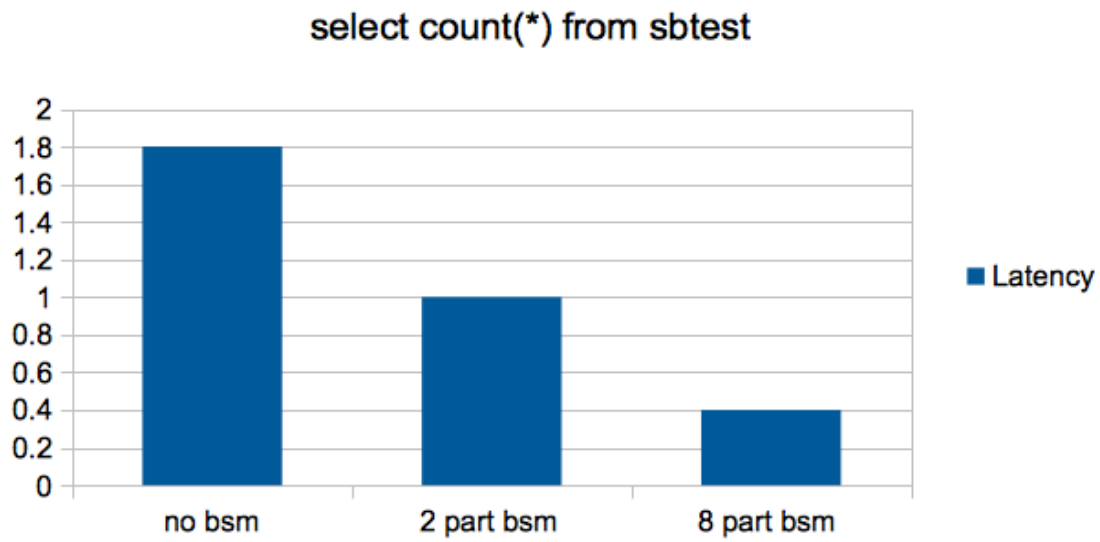
```
SELECT count(*) FROM sbtest;
+----------+
| count(*) |
+----------+
| 10000001 |
+----------+
1 row IN SET (8,38 sec)

SET spider_casual_read=1;
SET spider_bgs_mode=2;

SELECT count(*) FROM sbtest;
+----------+
| count(*) |
+----------+
| 10000001 |
+----------+
1 row IN SET (4,25 sec)

mysql> SELECT sum(k) FROM sbtest;
+--------+
| sum(k) |
+--------+
|      0 |
+--------+
1 row IN SET (5,67 sec)

mysql> SET spider_casual_read=0;
mysql> SELECT sum(k) FROM sbtest;
+--------+
| sum(k) |
+--------+
|      0 |
+--------+
1 row IN SET (12,56 sec)
```

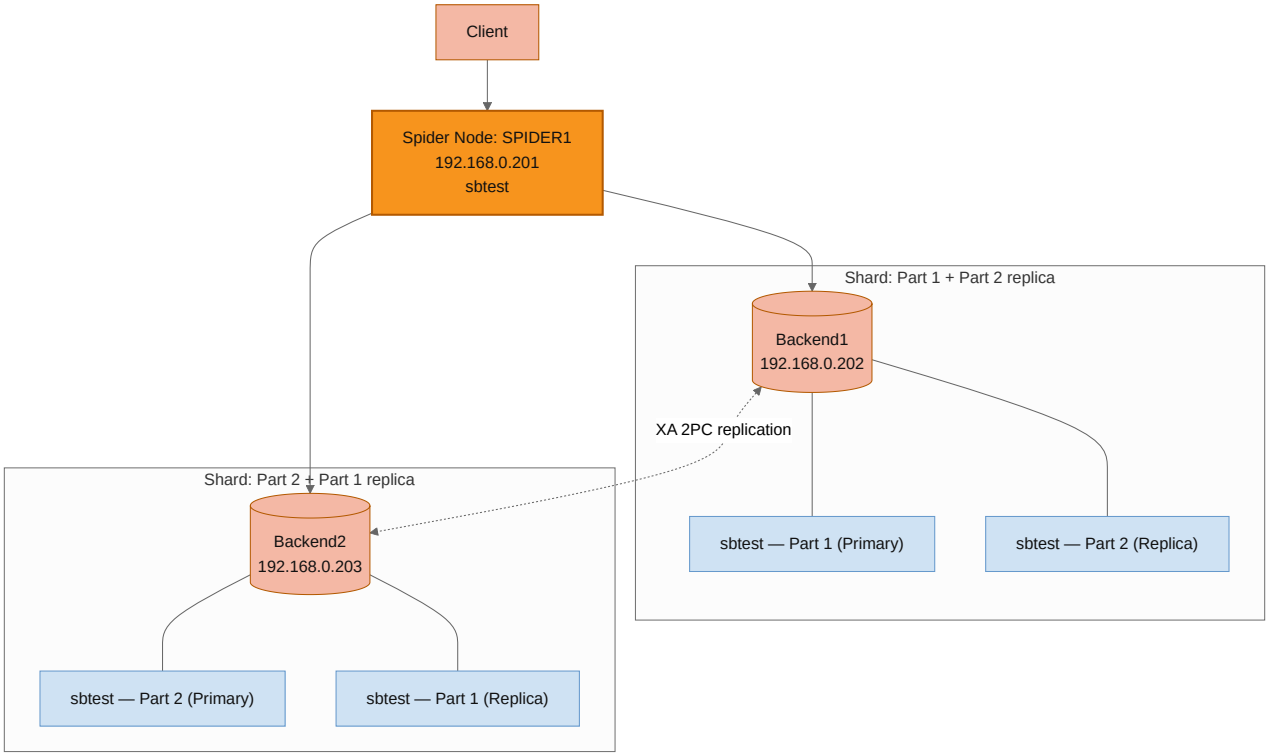


spbench8

High Availability Setup

MariaDB starting with 10.7.5

Spider's high availability feature has been deprecated (MDEV-28479), and are deleted. Please use other high availability solutions like replication or galera-cluster.



For high availability, Backend1 and Backend2 each hold their own primary shard plus a replica of the other's shard, kept in sync via XA 2PC.

```
#backend1 -e "CREATE DATABASE backend_rpl"
#backend2 -e "CREATE DATABASE backend_rpl"

#/usr/local/skysql/mariadb/bin/mysqldump  --user=skysql --password=skyvodka --host=192.168.0.202 --
port=5054  backend sbtest | backend1 backend_rpl
#/usr/local/skysql/mariadb/bin/mysqldump  --user=skysql --password=skyvodka --host=192.168.0.203 --
port=5054  backend sbtest | backend2 backend_rpl

#spider1 << EOF
DROP TABLE backend.sbtest;
CREATE SERVER backend1_rpl
  FOREIGN DATA WRAPPER mysql
OPTIONS(
  HOST '192.168.0.202',
  DATABASE 'backend_rpl',
  USER 'skysql',
  PASSWORD 'skyvodka',
  PORT 5054
);
CREATE SERVER backend2_rpl
  FOREIGN DATA WRAPPER mysql
OPTIONS(
  HOST '192.168.0.203',
  DATABASE 'backend_rpl',
  USER 'skysql',
  PASSWORD 'skyvodka',
  PORT 5054
);
CREATE  TABLE backend.sbtest
(
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=spider COMMENT='wrapper "mysql", TABLE "sbtest"'
 PARTITION BY KEY (id)
(
 PARTITION pt1 COMMENT = 'srv "backend1 backend2_rpl"',
 PARTITION pt2 COMMENT = 'srv "backend2 backend1_rpl"'
) ;
INSERT INTO backend.sbtest SELECT 10000001, 0, '' ,'replicas test';
EOF
#backend1 -e "SELECT * FROM backend.sbtest WHERE id=10000001";
+----------+---+---+---------------+
| id       | k | c | pad           |
+----------+---+---+---------------+
| 10000001 | 0 |   | replicas test |
+----------+---+---+---------------+
# backend2 -e "SELECT * FROM backend.sbtest WHERE id=10000001";
# backend2 -e "SELECT * FROM backend_rpl.sbtest WHERE id=10000001";
+----------+---+---+---------------+
| id       | k | c | pad           |
+----------+---+---+---------------+
| 10000001 | 0 |   | replicas test |
+----------+---+---+---------------+
```

What is happening if we stop one backend?

```
#spider1 -e "SELECT * FROM backend.sbtest WHERE id=10000001";
ERROR 1429 (HY000) at line 1: Unable to connect to foreign data source: backend1
```

Let’s fix this with spider monitoring. Note that msi is the list of spider nodes @@server_id variable participating in the quorum.

```
#spider1 << EOF
DROP  TABLE backend.sbtest;
CREATE  TABLE backend.sbtest
(
  id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=spider COMMENT='wrapper "mysql", TABLE "sbtest"'
 PARTITION BY KEY (id)
(
 PARTITION pt1 COMMENT = 'srv "backend1 backend2_rpl",  mbk "2", mkd "2", msi "5054", link_status "0 0"',
 PARTITION pt2 COMMENT = 'srv "backend2 backend1_rpl",  mbk "2", mkd "2", msi "5054", link_status "0 0" '
) ;

CREATE SERVER mon
  FOREIGN DATA WRAPPER mysql
OPTIONS(
  HOST '192.168.0.201’,
  DATABASE 'backend',
  USER 'skysql',
  PASSWORD 'skyvodka',
  PORT 5054
);
INSERT INTO `mysql`.`spider_link_mon_servers` VALUES
('%','%','%',5054,'mon',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,0,NULL,NULL);
SELECT spider_flush_table_mon_cache();
EOF
```

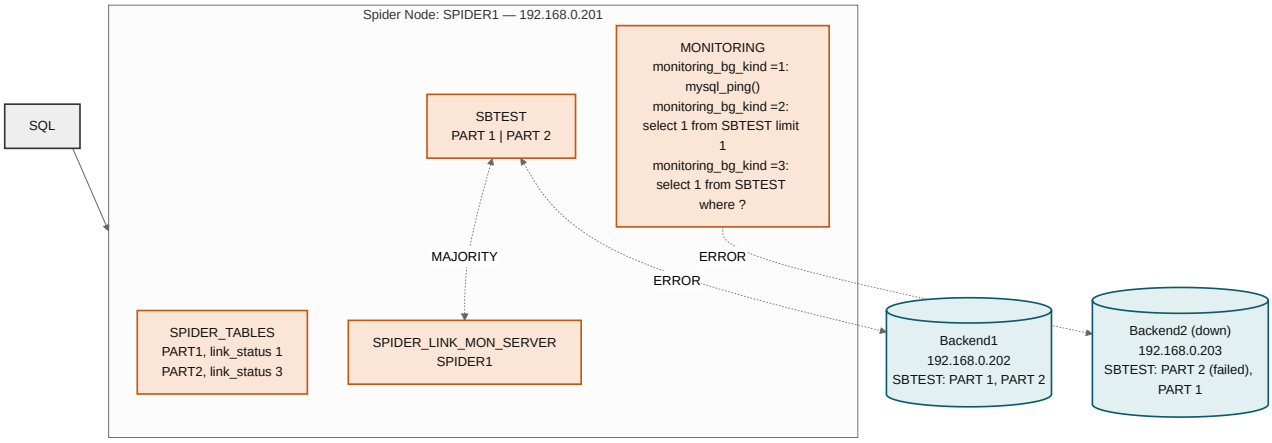
Monitoring should be setup between Spider nodes participating in the cluster. We only have one `Spider Node` and spider_link_mon_servers represent the inter-connection of all Spider nodes in our setup.

This simple setup does not bring HA in case the `Spider Node` is not available. In a production setup the number of `Spider Nodes` in the spider_link_mon_servers table should be at least 3 to get a majority consensus.

```
#spider1 -e "SELECT * FROM backend.sbtest WHERE id=10000001"
+----------+---+---+---------------+
| id       | k | c | pad           |
+----------+---+---+---------------+
| 10000001 | 0 |   | replicas test |
+----------+---+---+---------------+
```

Checking the state of the nodes:

```
#spider1 -e "SELECT db_name, table_name,server  FROM mysql.spider_tables WHERE link_status=3"
+---------+--------------+----------+
| db_name | table_name   | server   |
+---------+--------------+----------+
| backend | sbtest#P#pt1 | backend1 |
+---------+--------------+----------+
```



Spider's link monitoring: SPIDER1 tracks link_status for each partition, uses SPIDER_LINK_MON_SERVER and MAJORITY quorum voting to detect Backend1 vs. the failed Backend2, while a background MONITORING process pings and probes SBTEST on each backend.

No change has been made to cluster, so let's create a divergence:

```
# spider1 -e "INSERT INTO backend.sbtest SELECT 10000003, 0, '' ,'replicas test';"
# backend1 -e "SELECT * FROM backend.sbtest WHERE id=10000003"
# backend2 -e "SELECT * FROM backend_rpl.sbtest WHERE id=10000003"
+----------+---+---+---------------+
| id       | k | c | pad           |
+----------+---+---+---------------+
| 10000003 | 0 |   | replicas test |
+----------+---+---+---------------+
```

Reintroducing the failed backend1 in the cluster:

```
#spider1 << EOF
ALTER TABLE backend.sbtest
ENGINE=spider COMMENT='wrapper "mysql", TABLE "sbtest"'
 PARTITION BY KEY (id)
(
 PARTITION pt1 COMMENT = 'srv "backend1 backend2_rpl"  mbk "2", mkd "2", msi "5054", link_status "2 0"',
 PARTITION pt2 COMMENT = 'srv "backend2 backend1_rpl"  mbk "2", mkd "2", msi "5054", link_status "0 2" '
) ;
SELECT spider_copy_tables('backend.sbtest#P#pt1','0','1');
SELECT spider_copy_tables('backend.sbtest#P#pt2','1','0');
ALTER TABLE backend.sbtest
ENGINE=spider COMMENT='wrapper "mysql", TABLE "sbtest"'
 PARTITION BY KEY (id)
(
 PARTITION pt1 COMMENT = 'srv "backend1 backend2_rpl"  mbk "2", mkd "2", msi "5054", link_status "1 0"',
 PARTITION pt2 COMMENT = 'srv "backend2 backend1_rpl"  mbk "2", mkd "2", msi "5054", link_status "0 1" '
) ;
EOF
```

This page is licensed: CC BY-SA / Gnu FDL

Spider Storage Engine Introduction

See MariaDB Enterprise Spider Schema Design.

Learn about the schema design of the Spider storage engine.

See MariaDB Enterprise Spider Operations.

Learn about Spider operations in MariaDB Enterprise Server. This section covers advanced configurations, management, and best practices for leveraging Spider in enterprise environments.

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Spider Schema Design

Create Tables

Spider Tables can be created with MariaDB Enterprise Spider using the CREATE TABLE statement. The Enterprise Spider storage engine can be chosen for the table by specifying `ENGINE=Spider`.

For each Spider Table, MariaDB Enterprise Spider requires connection details for the Data Nodes. The connection details are provided by specifying Connection Options in the `COMMENT` option for the table or partition, depending on the topology.

Create Tables in a Federated Topology

In a Federated MariaDB Enterprise Spider topology, the Connection Options are specified in the `COMMENT` table option for the Spider Table:

```
CREATE SERVER hq_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.2",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "hq_sales"
);

CREATE DATABASE spider_hq_sales;

CREATE TABLE spider_hq_sales.invoices (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
COMMENT='server "hq_server", table "invoices"';
```

An alternative syntax is available. When you don't want to create a server object, the full connection details for the Data Node can be specified in the COMMENT table option:

```
CREATE TABLE spider_hq_sales.invoices_alternate (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
COMMENT='table "invoices", host "192.0.2.2", port "5801", user "spider_user", password "password",
database "hq_sales"';
```

Create Tables in a Sharded Topology

In a Sharded MariaDB Enterprise Spider topology, the Connection Options are specified in the `COMMENT` partition option for each partition of the Spider Table:

```
CREATE SERVER hq_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.2",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "hq_sales"
);

CREATE SERVER eastern_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.3",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "eastern_sales"
);

CREATE SERVER western_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.4",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "western_sales"
);

CREATE DATABASE spider_sharded_sales;

CREATE TABLE spider_sharded_sales.invoices (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
PARTITION BY LIST(branch_id) (
   PARTITION hq_partition VALUES IN (1) COMMENT = 'server "hq_server", table "invoices"',
   PARTITION eastern_partition VALUES IN (2) COMMENT = 'server "eastern_server", table "invoices"',
   PARTITION western_partition VALUES IN (3) COMMENT = 'server "western_server", table "invoices"'
);
```

An alternative syntax is available. When you don’t want to create a server object, the full connection details for the Data Nodes can be specified in the COMMENT partition option:

```
CREATE TABLE spider_sharded_sales.invoices (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
PARTITION BY LIST(branch_id) (
   PARTITION hq_partition VALUES IN (1) COMMENT = 'table "invoices", host "192.0.2.2", port 5801, user
"spider_user", password "password", database "hq_sales"',
   PARTITION eastern_partition VALUES IN (2) COMMENT = 'table "invoices", host "192.0.2.3", port 5801,
user "spider_user", password "password", database "eastern_sales"',
   PARTITION western_partition VALUES IN (3) COMMENT = 'table "invoices", host "192.0.2.4", port 5801,
user "spider_user", password "password", database "western_sales"'
);
```

Connection Options

The following connection options are supported in the COMMENT table option for Federated Spider Tables and in the COMMENT partition option for Sharded Spider Tables:

Option	Data Type	Definition
database	String	The database to select when connecting to the Data Node.
host	String	The hostname or IP address for the Data Node. This option is mutually exclusive with socket.
password	String	The password to use when connecting to the Data Node.
port	Integer	The port to use when connecting to the Data Node. This option is mutually exclusive with socket.
socket	String	The path to the Unix socket file to use when connecting to the Data Node. This option is mutually exclusive with port.
ssl_ca	String	The path to the file with the TLS certificate authority chain.
ssl_capath	String	The path to the directory with the TLS certificate authority chain files.
ssl_cert	String	The path to the TLS client certificate.
ssl_key	String	The path to the TLS private key.
ssl_cipher	String	The path to the TLS certificate authority chain.
table	String	The table to query.
user	String	The username to use when connecting to the Data Node.
wrapper	String	The foreign data wrapper should be mariadb for the Federated and Sharded topologies or odbc for the ODBC topology.

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Spider Operations

See Federated MariaDB Enterprise Spider Topology Operations.

See ODBC MariaDB Enterprise Spider Topology Operations.

See Sharded MariaDB Enterprise Spider Topology Operations.

See Also

See MariaDB Enterprise Spider Topologies.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Federated MariaDB Enterprise Spider Topology Operations

See Spider Federated Overview.

An overview of the federated topology for Spider, where a single Spider node aggregates data from multiple remote data nodes, acting as a unified access point.

See Federated MariaDB Enterprise Spider Topology Backup and Restore.

Procedures for performing consistent backups and restores in a federated Spider topology using MariaDB Backup and MariaDB Dump, ensuring data synchronization.

See Federated MariaDB Enterprise Spider Topology Migrate Tables.

A guide on how to migrate tables from a standard MariaDB deployment to a Federated Spider topology, distributing data across multiple backend nodes.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Spider Federated Overview

Overview

Choose an operation for the Sharded MariaDB Enterprise Spider topology:

Operation	Description
Backup and Restore	How to create a new backup or restore an existing backup.
Migrate Tables	How to update the character set or collation.
Update Character Set or Collation	How to update the character set or collation.
Update Connection Options	How to update the connection options for a Data Node.

Update Character Sets and Collations

The character set or collation for the Spider Table can be updated or modified using the ALTER TABLE statement.

On the Spider Node, alter the Spider Table's character set and collation:

```
ALTER TABLE spider_hq_sales.invoices
   DEFAULT CHARACTER SET 'utf8mb4'
   DEFAULT COLLATE 'utf8mb4_general_ci';
```

If the new character set and collation are not compatible with the character set and collation used by the Data Table, you must also alter the character set and collation for the Data Table on the Data Node.

Update Connection Options

In a Federated MariaDB Enterprise Spider topology, the connection options for a Data Node can be updated using the ALTER TABLE statement.

- On the Spider Node*, alter the table's connection details:

```
ALTER TABLE spider_hq_sales.invoices
   COMMENT = 'server "new_hq_server", table "invoices"'
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Federated MariaDB Enterprise Spider Topology Backup and Restore

Overview

When using Spider in the Federated MariaDB Enterprise Spider topology, backup and restore operations can be performed using MariaDB Backup or MariaDB Dump. These operations must be performed on the Spider Node and the Data Node.

MariaDB Backup

MariaDB Backup can be used to perform backup operations on Spider deployments.

Backup Spider with MariaDB Backup

MariaDB Backup can be used to create a physical backup of a Federated MariaDB Enterprise Spider topology. MariaDB Backup must be used to backup the Spider Node and the Data Node. The Spider Node and Data Node must be locked to guarantee that the backups of the Spider Node and Data Node are consistent.

The backup of the Spider Node contains:

- Physical backup for Spider Tables
- Physical backup for tables that use InnoDB, Aria, MyISAM, and MyRocks
- Physical backup for any other database objects on the Spider Node

The backup of the Data Node contains:

- Physical backup for Data Tables
- Physical backup for tables that use InnoDB, Aria, MyISAM, and MyRocks
- Physical backup for any other database objects on the Data Node

The following procedure shows how to take a consistent backup of a Spider Node and a Data Node deployed in a Spider Federated MariaDB Enterprise Spider topology.

1. On the Spider Node and on the Data Node, create a user account to perform the backup using the CREATE USER and GRANT statements:

```
CREATE USER 'mariadb-backup'@'localhost'
   IDENTIFIED BY 'mb_passwd';

GRANT RELOAD, PROCESS, LOCK TABLES, BINLOG MONITOR
   ON *.*
   TO 'mariadb-backup'@'localhost';
```

2. On the Data Node, grant the user account Spider uses to operate on the Data Node sufficient privileges to lock the Data Tables using the GRANT statement.

For example, on the hq_server Data Node:

```
GRANT LOCK TABLES ON hq_sales.* TO 'spider_user'@'192.0.2.2';
```

3. On the Spider Node, acquire a read lock on Spider Tables using the LOCK TABLES statement:

```
LOCK TABLES spider_federated_sales.invoices READ;
```

Keep this session open during the rest of the procedure.

The read lock will propagate to the Data Tables on the Data Node. The read locks will prevent the Data Tables from changing during the backup, so the backups of the Spider Node and the Data Node are consistent.

4. On the Data Node, perform the backup using MariaDB Backup:

```
$ sudo mariadb-backup --backup \
   --target-dir=/data/backups/full \
   --user=mariadb-backup \
   --password='mb_passwd'
```

5. When the Data Node backup is complete, perform the backup on the Spider Node using MariaDB Backup:

```
$ sudo mariadb-backup --backup \
   --target-dir=/data/backups/full \
   --user=mariadb-backup \
   --password='mb_passwd'
```

6. When the Spider Node backup is complete, release the table locks in your original session using the UNLOCK TABLES statement:

```
UNLOCK TABLES;
```

7. On the Spider Node and the Data Node, prepare each of the backups using MariaDB Backup:

```
$ sudo mariadb-backup --prepare \
   --target-dir=/data/backups/full
```

The Spider Node and the Data Node now have a complete backup of the data directory. Backups should be tested to confirm they are complete and consistent.

Restoring Spider with MariaDB Backup

MariaDB Backup can restore a Federated MariaDB Enterprise Spider topology from a backup taken with MariaDB Backup.

1. On the Spider Node and on the Data Node, stop the MariaDB Server process:

```
$ sudo systemctl stop mariadb
```

2. On the Spider Node and on the Data Node, empty the data directory:

```
$ sudo rm -fr /var/lib/mysql/*
```

3. On the Spider Node and on the Data Node, restore the backup for that server using MariaDB Backup:

```
$ sudo mariadb-backup --copy-back \
   --target-dir=/data/backups/full
```

4. On the Spider Node and on the Data Node, confirm that the restored files are owned by the user that owns the MariaDB Server process:

```
$ sudo chown -R mysql:mysql /var/lib/mysql
```

5. On the Spider Node and on the Data Node, start the MariaDB Server process:

```
$ sudo systemctl start mariadb
```

6. On the Spider Node, query a Spider Table to test it:

```
SELECT * FROM spider_hq_sales.invoices;
```

```
+-----------+------------+-------------+---------------------------+---------------+----------------+
| branch_id | invoice_id | customer_id | invoice_date              | invoice_total | payment_method |
+-----------+------------+-------------+---------------------------+---------------+----------------+
|         1 |          1 |           1 | 2020-05-10 12:35:10.000000 |       1087.23 | CREDIT_CARD    |
|         1 |          2 |           2 | 2020-05-10 14:17:32.000000 |       1508.57 | WIRE_TRANSFER  |
|         1 |          3 |           3 | 2020-05-10 14:25:16.000000 |        227.15 | CASH           |
+-----------+------------+-------------+---------------------------+---------------+----------------+
```

MariaDB Dump

MariaDB Dump generates backup files containing the SQL statements necessary to recreate the database. MariaDB Dump is included with MariaDB Enterprise Server and can be used to backup databases in Spider deployments. The MariaDB Client can then be used to restore databases from a MariaDB Dump backup.

Backing Up Spider with MariaDB Dump

MariaDB Dump can be used to create a logical backup of a Federated MariaDB Enterprise Spider topology. MariaDB Dump must be used to backup the Spider Node and the Data Node. The Spider Node and Data Node must be locked to guarantee that the backups of the Spider Node and Data Node are consistent.

The backup of the Spider Node contains:

- Table definitions for Spider Tables
- Table definitions for all tables
- Table data for tables that use InnoDB, Aria, MyISAM, and MyRocks
- Definitions for any other user accounts, privileges, views, and triggers on the Spider Node

The backup of the Data Node contains:

- Table definitions for Data Tables
- Table definitions for all tables
- Table data for tables that use InnoDB, Aria, MyISAM, and MyRocks
- Definitions for any other user accounts, privileges, views, and triggers on the Data Node

The following procedure shows how to take a consistent backup of a Spider Node and a Data Node deployed in a Federated MariaDB Enterprise Spider topology.

1. On the Spider Node and on the Data Node, create a user account to perform the backup using the CREATE USER and GRANT statements:

```
CREATE USER 'mariadb-dump'@'localhost'
   IDENTIFIED BY 'md_passwd';

GRANT SELECT, INSERT, SHOW VIEW, TRIGGER, CREATE, ALTER, EVENT, RELOAD, LOCK TABLES
   ON *.*
   TO 'mariadb-dump'@'localhost';
```

2. On the Data Node, grant the user account Spider uses to operate on the Data Node sufficient privileges to lock the Data Tables using the GRANT statement.

For example, on the hq_server Data Node:

```
GRANT LOCK TABLES ON hq_sales.* TO 'spider_user'@'192.0.2.2';
```

3. On the Spider Node, acquire a read lock on Spider Tables using the LOCK TABLES statement:

```
LOCK TABLES spider_federated_sales.invoices READ;
```

Keep this session open during the rest of the procedure.

The read lock will propagate to the Data Tables on the Data Node. The read locks will prevent the Data Tables from receiving any changes during the backup, which ensures the backups of the Spider Node and the Data Node are consistent.

4. On the Data Node, perform the backup using MariaDB Dump:

```
$ mariadb-dump \
   --user=mariadb-dump \
   --password='md_passwd' \
   --ssl-verify-server-cert \
   --ssl-ca ~/mariadb_chain.pem \
   --all-databases \
   --single-transaction \
   --default-character-set=utf8mb4 \
   > mariadb_dump.sql
```

5. When the Data Node has been backed up, perform the backup on the Spider Node using MariaDB Dump:

```
$ mariadb-dump \
   --user=mariadb-dump \
   --password='md_passwd' \
   --ssl-verify-server-cert \
   --ssl-ca ~/mariadb_chain.pem \
   --all-databases \
   --single-transaction \
   --default-character-set=utf8mb4 \
   > mariadb_dump.sql
```

6. When the Spider Node backup is complete, release the table locks in your original session using the UNLOCK TABLES statement:

```
UNLOCK TABLES;
```

The Spider Node and the Data Node now each have a mariadb_dump.sql backup file. This backup file contains the SQL statements needed to recreate the schema on the Spider Node and the schema and data on the Data Node. Backups should be tested to confirm they are complete and consistent.

Restoring Spider from MariaDB Dump

MariaDB Client can restore a Federated MariaDB Enterprise Spider topology from a backup taken with MariaDB Dump backup.

1. Stop all traffic to the Spider Node and the Data Node.
2. On the Spider Node, restore the backup for that server using MariaDB Client:

```
$ mariadb \
   --user=mariadb-dump \
   --password='md_passwd' \
   --ssl-verify-server-cert \
   --ssl-ca ~/mariadb_chain.pem \
   --default-character-set=utf8mb4 \
   < mariadb_dump.sql
```

3. On the Data Node, restore the backup for that server using MariaDB Client:

```
$ mariadb \
   --user=mariadb-dump \
   --password='md_passwd' \
   --ssl-verify-server-cert \
   --ssl-ca ~/mariadb_chain.pem \
   --default-character-set=utf8mb4 \
   < mariadb_dump.sql
```

On the Spider Node, query a Spider Table to test it:

```
SELECT * FROM spider_hq_sales.invoices;
```

```
+-----------+------------+-------------+----------------------------+--------------+----------------+
| branch_id | invoice_id | customer_id | invoice_date               | invoice_total | payment_method |
+-----------+------------+-------------+----------------------------+--------------+----------------+
|         1 |          1 |           1 | 2020-05-10 12:35:10.000000 |       1087.23 | CREDIT_CARD    |
|         1 |          2 |           2 | 2020-05-10 14:17:32.000000 |       1508.57 | WIRE_TRANSFER  |
|         1 |          3 |           3 | 2020-05-10 14:25:16.000000 |        227.15 | CASH           |
+-----------+------------+-------------+----------------------------+--------------+----------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Federated MariaDB Enterprise Spider Topology Migrate Tables

Overview

The Federated MariaDB Enterprise Spider topology can be used to migrate tables from one MariaDB Enterprise Server node to another MariaDB Enterprise Server node:

- The MariaDB Enterprise Server node with the source table is configured as a Data Node.
- The MariaDB Enterprise Server node with the destination table is configured as a Spider Node.
- The Data Table is the source table on the Data Node.
- A Spider Table is created on the Spider Node that references the Data Table on the Data Node.
- On the Spider node, the Data Table's data is migrated to the destination table by querying the Spider Table like the following:

```
INSERT INTO innodb_tab
   SELECT * FROM spider_tab;
```

Follow the steps below to migrate tables using the Federated MariaDB Enterprise Spider topology.

Deploy the Federated Topology

Before you can migrate tables, the Federated MariaDB Enterprise Spider topology must be deployed.

For additional information, see "Deploy MariaDB Enterprise Spider".

Create Local Tables

A local copy of the table must be created. This new table will contain the migrated data.

- On the Spider Node*, create a local copy of each table that is being migrated:

```
CREATE DATABASE hq_sales;

CREATE SEQUENCE hq_sales.invoice_seq;

CREATE TABLE hq_sales.invoices (
   branch_id INT NOT NULL DEFAULT (1) CHECK (branch_id=1),
   invoice_id INT NOT NULL DEFAULT (NEXT VALUE FOR migrated_hq_sales.invoice_seq),
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=InnoDB;
```

Migrate the Tables

The table data can be migrated to the local table using the Spider Tables.

- On the Spider Node*, migrate the table data to the local copy of the table using the INSERT SELECT statement:

```
INSERT INTO hq_sales.invoices
   SELECT * FROM spider_hq_sales.invoices;
```

Test Read Operations

On the Spider Node, read from the local copy of the table using a SELECT statement to confirm that the data has been migrated:

```
SELECT * FROM hq_sales.invoices;
```

```
+-----------+------------+-------------+----------------------------+---------------+----------------+
| branch_id | invoice_id | customer_id | invoice_date               | invoice_total | payment_method |
+-----------+------------+-------------+----------------------------+---------------+----------------+
|         1 |          1 |           1 | 2020-05-10 12:35:10.000000 |       1087.23 | CREDIT_CARD    |
|         1 |          2 |           2 | 2020-05-10 14:17:32.000000 |       1508.57 | WIRE_TRANSFER  |
|         1 |          3 |           3 | 2020-05-10 14:25:16.000000 |        227.15 | CASH           |
+-----------+------------+-------------+----------------------------+---------------+----------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

ODBC MariaDB Enterprise Spider Topology Operations

See Use Spider ODBC to Connect to Oracle.

This guide details how to configure Spider to connect to an Oracle database via ODBC, enabling data migration and federated access to Oracle tables.

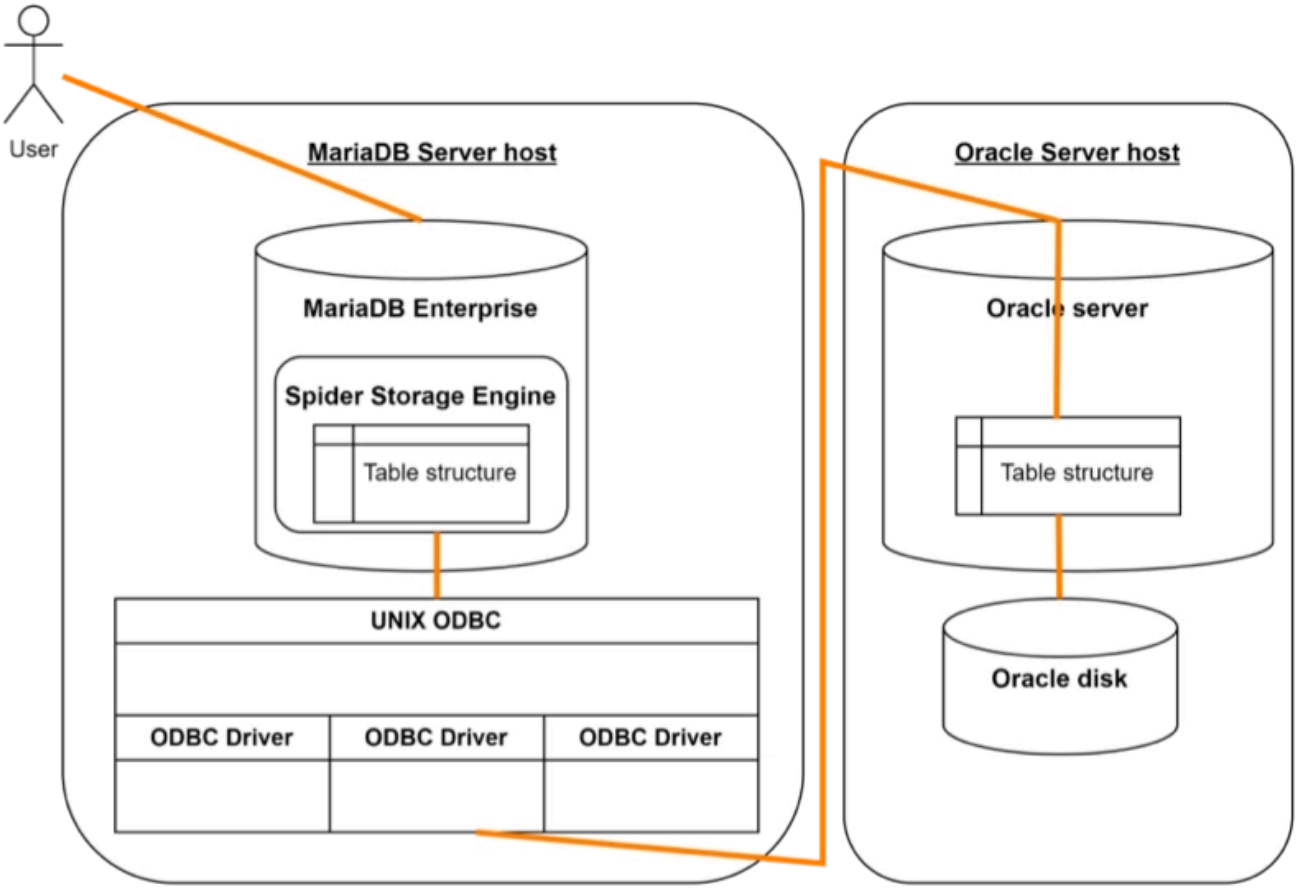
This page is: Copyright © 2026 MariaDB. All rights reserved.

Use Spider ODBC to Connect to Oracle

Overview

This article describes how to use Spider ODBC to connect to Oracle. This can make it easier to migrate, by not requiring a wholesale cutover, but instead enabling a piecemeal approach.

The setup looks as shown in this figure:



Prerequisites

In MariaDB,

- The Spider storage engine must be installed.
- The Spider storage engine plugin must be installed in MariaDB Enterprise Server. This is described in the instructions below.
- An ODBC driver manager, unixODBC, must be installed.

In Oracle Database,

- The ODBC driver must be installed.
- The Oracle Basic Client must be installed. This is a prerequisite for installing the ODBC driver.

> **Note:**
>
> The following procedure has been tested on *CentOS 7 AWS EC2 t3a.Medium* and a *t3.small RDS* Oracle 19 database. It assumes you are just doing a simple install of MariaDB. Also, the following schema is used to populate the Oracle database: Oracle Sample Database.

Instructions

> **Step 1**
>
> In your working directory, create a folder to hold the Oracle ODBC `RPMs mkdir oracle_odbc_rpms`, then change to that directory.

Step 2

Download the following files (make sure to use the appropriate driver for your scenario):

```
wget https://download.oracle.com/otn_software/linux/instantclient/1916000/oracle-instantclient19.16-odbc-19.16.0.0.0-1.x86_64.rpm
wget https://download.oracle.com/otn_software/linux/instantclient/1916000/oracle-instantclient19.16-sqlplus-19.16.0.0.0-1.x86_64.rpm
wget https://download.oracle.com/otn_software/linux/instantclient/1916000/oracle-instantclient19.16-basic-19.16.0.0.0-1.x86_64.rpm
```

Step 3

Install the RPMs:

```
yum localinstall *.rpm
```

Step 4

Add the following to `/etc/odbcinst.ini`:

```
[oracle]
Description = Oracle ODBC Connection
Driver = /usr/lib/oracle/19.16/client64/lib/libsqora.so.19.1
```

The driver path may be different if you downloaded a different version of the Oracle driver, be sure to update it with correct path for your scenario.

Step 5

Add the following to `/etc/odbc.ini`:

```
[ORARDS]
Description = Oracle
Driver = oracle
ServerName = REMOTE
UserID = {user_name}
Password = {password}
TNSNamesFile = /etc/tnsnames.ora
```

- `Driver` is the driver name in the `/etc/odbcinst.ini` .
- `Description` can be whatever you like, just make sure you use the same value later.
- `ServerName` is the TNSName given for our Oracle connection.
- Be sure to populate your user name and password.

Step 6

Add the following line to `/etc/tnsnames.ora`:

```
REMOTE=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST={host_name/ip address})(PORT=1521))(CONNECT_DATA=(SERVICE_NAME={service_name})))
```

- Name the connection string as you like — just be sure to use the same name in the `odbc.ini` file.
- Make sure to include your hostname and IP address, and the service name for your instance.

Step 7

Verify that the ODBC connection is working with:

```
isql -v ORARDS
```

- `ORARDS` is the section name we gave our entry in `odbc.ini`. If you are using something different, replace `ORARDS` with whatever you use.
- If any errors come up, double-check the values entered into the `.ini` files.

Step 8

Start the MariaDB database:

```
systemctl start mariadb or systemctl start mysqld
```

Step 9

Log in to MariaDB, and run the following:

```
INSTALL SONAME 'ha_spider';
CREATE DATABASE spider_test;
USE spider_test;
CREATE OR REPLACE TABLE spider_test.contacts
(
  CONTACT_ID BIGINT NOT NULL PRIMARY KEY,
  FIRST_NAME  VARCHAR( 255 ) NOT NULL,
  LAST_NAME   VARCHAR( 255 ) NOT NULL,
  EMAIL       VARCHAR( 255 ) NOT NULL,
  PHONE       VARCHAR( 20 )          ,
  CUSTOMER_ID BIGINT) ENGINE=SPIDER
  CONNECTION='WRAPPER "odbc", DSN "ORARDS", table "CONTACTS"';
```

- This creates a new database and a table that should be able to connect to an Oracle server via ODBC.
- `DSN` must be the same put in the `odbc.ini file` .
- It's best to always use upper case for both column names and the table value in the comment section, because these values are passed directly to Oracle Database, which prefers upper-case object names.

Step 10

You should now be able to select data from `spider_test.contacts` table. That data is coming from the remote Oracle database.

Notes

- Spider ODBC to Oracle tables does not support `INSERT SELECT` statements, where the Spider table is the source of the data. (MENT-1588).
- If the version of MariaDB does not have the spider_direct_aggregate variable, you aren't able to use a few aggregate functions on their own. (MENT-1558).
- If you get the following error message: `Error from ODBC 0 01004 [Oracle][ODBC]String data, right truncated.`, you need to set a larger value for the variable spider_buffer_size. (MENT-1557).

See Also

- Connecting to Oracle from MariaDB Enterprise Server using Spider
  Blog post by Anders Karlsson, January 2024 - 13 minute read

This page is: Copyright © 2026 MariaDB. All rights reserved.

Sharded MariaDB Enterprise Spider Topology Operations

See Spider Sharded Overview.

Provides an overview of using Spider for sharding in MariaDB Enterprise Server, allowing data distribution across multiple nodes for horizontal scalability.

See Sharded MariaDB Enterprise Spider Topology Add a Shard.

Instructions on how to expand a sharded Spider topology by adding new data nodes (shards) and rebalancing the data distribution.

See Sharded MariaDB Enterprise Spider Topology Backup and Restore.

Guidelines for backing up and restoring a sharded Spider topology, ensuring consistency across multiple shards using tools like MariaDB Backup.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Spider Sharded Overview

Overview

Choose an operation for the Sharded MariaDB Enterprise Spider topology:

Operation	Description
Add a Shard	How to add a new shard to a Spider Table.
Backup and Restore	How to create a new backup or restore an existing backup.
Update Character Set or Collation	How to update the character set or collation for a Spider Table.
Update Connection Options	How to update the connection options for a Data Node.

Update Character Sets and Collations

The character set or collation for the Spider Table can be updated or modified using the ALTER TABLE statement.

On the Spider Node, alter the Spider Table's character set and collation:

```
ALTER TABLE spider_sharded_sales.invoices
   DEFAULT CHARACTER SET 'utf8mb4'
   DEFAULT COLLATE 'utf8mb4_general_ci';
```

If the new character set and collation are not compatible with the character set and collation used by the Data Table, you must also alter the character set and collation for the Data Table on the Data Node.

Update Connection Options

In a Sharded MariaDB Enterprise Spider topology, the connection options for a Data Node can be updated using the ALTER TABLE statement.

On the Spider Node, alter the partition's connection details:

```
ALTER TABLE spider_sharded_sales.invoices
   REORGANIZE PARTITION hq_partition INTO (
      PARTITION hq_partition VALUES IN (1) COMMENT = 'server "new_hq_server", table "invoices"'
   );
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Sharded MariaDB Enterprise Spider Topology Add a Shard

Overview

In a Sharded MariaDB Enterprise Spider topology, new shards can be added using the following procedure.

Create Spider User

Each data node requires a user account that the Spider Node uses to connect.

On the Data Node hosting the new shard, create the Spider user account for the Spider Node using the CREATE USER statement:

```
CREATE USER spider_user@192.0.2.1 IDENTIFIED BY "password";
```

Privileges are granted to the user account in a later step.

Test Spider User

On the Spider Node, confirm that the Spider user account can connect to the Data Node using MariaDB Client:

```
$ mariadb --user spider_user --host 192.0.2.2 --password
```

Configure Connection Details

The Spider Node requires connection details for each Data Node.

On the Spider Node, create a server object to configure the connection details for the Data Node hosting the new shard using the CREATE SERVER statement:

```
CREATE SERVER southern_server
   FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST '192.0.2.6',
   PORT 5004,
   USER 'spider_user',
   PASSWORD 'password',
   DATABASE 'southern_sales'
);
```

The Data Node runs MariaDB Enterprise Server, so the `FOREIGN DATA WRAPPER` is set to mariadb.

Using a server object for connection details is optional. Alternatively, the connection details for the Data Node can be specified in the `COMMENT` table option of the CREATE TABLE statement when creating the Spider Table.

Create the Data Table

When queries read and write to a Spider Table, Spider reads and writes to the Data Tables for each partition on the Data Nodes. The Data Tables must be created on the Data Nodes with the same structure as the Spider Table.

If your Data Tables already exist, grant privileges on the tables to the Spider user.

On the Data Node hosting the new shard, create the Data Tables:

```
CREATE DATABASE southern_sales;

CREATE SEQUENCE southern_sales.invoice_seq;

CREATE TABLE southern_sales.invoices (
   branch_id INT NOT NULL DEFAULT (4) CHECK (branch_id=4),
   invoice_id INT NOT NULL DEFAULT (NEXT VALUE FOR hq_sales.invoice_seq),
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=InnoDB;

INSERT INTO southern_sales.invoices
   (customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, '2020-05-25 10:22:11', 2083.56, 'CREDIT_CARD'),
   (2, '2020-05-25 11:42:33', 515.22, 'WIRE_TRANSFER'),
   (3, '2020-05-25 13:15:16', 1213.80, 'CASH');
```

The Spider Node reads and writes to the Data Table using the server and user account configured previously. The user account must have privileges on the table.

Grant Privileges

The Spider Node connects to the Data Nodes with the user account configured previously.

On the Data Node hosting the new shard, grant the Spider user sufficient privileges to operate on the Data Table:

```
GRANT ALL PRIVILEGES ON southern_sales.invoices TO 'spider_user'@'192.0.2.1';
```

Privileges for Spider BKA Mode

By default, the Spider user also requires the CREATE TEMPORARY TABLES privilege on the database containing the Data Table. The CREATE TEMPORARY TABLES privilege is required, because Spider uses temporary tables to optimize read queries when Spider BKA Mode is 1.

Spider BKA Mode is configured using the following methods:

- The session value is configured by setting the Spider BKA Mode system variable on the Spider Node. The default value is -1. When the session value is -1, the value for each Spider Table is used.
- The value for each Spider Table is configured by setting the bka_mode option in the COMMENT table option. When the bka_mode option is not set, the implicit value is 1.

The default spider_bka_mode value is -1, and the implicit Spider Table value is 1, so the default Spider BKA Mode is 1.

On the Data Node hosting the new shard, grant the Spider user the CREATE TEMPORARY TABLES privilege on the database:

```
GRANT CREATE TEMPORARY TABLES ON southern_sales.* TO 'spider_user'@'192.0.2.1';
```

Alter the Spider Table

A partition for the new shard must be added to the Spider Table on the Spider Node.

On the Spider Node, alter the Spider Table to add the partition and reference the name of the Data Node hosting the new shard in the COMMENT partition option:

```
ALTER TABLE spider_sharded_sales.invoices
   ADD PARTITION (
      PARTITION southern_partition VALUES IN (4) COMMENT = 'server "southern_server", table "invoices"'
   );
```

Test Read Operations

On the Spider Node, read from the Spider Table using a SELECT statement:

```
SELECT * FROM spider_sharded_sales.invoices;
```

```
+-----------+------------+-------------+---------------------------+---------------+----------------+
| branch_id | invoice_id | customer_id | invoice_date              | invoice_total | payment_method |
+-----------+------------+-------------+---------------------------+---------------+----------------+
|         1 |          1 |           1 | 2020-05-10 12:35:10.000000 |       1087.23 | CREDIT_CARD    |
|         1 |          2 |           2 | 2020-05-10 14:17:32.000000 |       1508.57 | WIRE_TRANSFER  |
|         1 |          3 |           3 | 2020-05-10 14:25:16.000000 |        227.15 | CASH           |
|         2 |          1 |           2 | 2020-05-10 12:31:00.000000 |       1351.04 | CREDIT_CARD    |
|         2 |          2 |           2 | 2020-05-10 12:45:27.000000 |        162.11 | WIRE_TRANSFER  |
|         2 |          3 |           4 | 2020-05-10 13:11:23.000000 |        350.00 | CASH           |
|         3 |          1 |           5 | 2020-05-10 12:31:00.000000 |        111.50 | CREDIT_CARD    |
|         3 |          2 |           8 | 2020-05-10 12:45:27.000000 |       1509.23 | WIRE_TRANSFER  |
|         3 |          3 |           3 | 2020-05-10 13:11:23.000000 |       3301.66 | CASH           |
|         4 |          1 |           1 | 2020-05-25 10:22:11.000000 |       2083.56 | CREDIT_CARD    |
|         4 |          2 |           2 | 2020-05-25 11:42:33.000000 |        515.22 | WIRE_TRANSFER  |
|         4 |          3 |           3 | 2020-05-25 13:15:16.000000 |       1213.80 | CASH           |
+-----------+------------+-------------+---------------------------+---------------+----------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Sharded MariaDB Enterprise Spider Topology Backup and Restore

Overview

When using Spider in the Sharded MariaDB Enterprise Spider topology, backup and restore operations can be performed using MariaDB Backup or MariaDB Dump. These operations must be performed on the Spider Node as well as on each Data Node.

MariaDB Backup

MariaDB Backup can be used to perform backup operations on Spider deployments.

Backup Spider with MariaDB Backup

MariaDB Backup can be used to create a physical backup of a Sharded MariaDB Enterprise Spider topology. MariaDB Backup must be used to backup the Spider Node and all Data Nodes. The Spider Node and Data Nodes must be locked to guarantee that the backups of the Spider Node and Data Nodes are consistent.

The backup of the Spider Node contains:

- Physical backup for Spider Tables
- Physical backup for tables that use InnoDB, Aria, MyISAM, and MyRocks
- Physical backup for any other database objects on the Spider Node

The backups of the Data Nodes contain:

- Physical backup for Data Tables
- Physical backup for tables that use InnoDB, Aria, MyISAM, and MyRocks
- Physical backup for any other database objects on the Data Node

The following procedure shows how to take a consistent backup of a Spider Node and Data Nodes deployed in a Sharded MariaDB Enterprise Spider topology.

1. On the Spider Node and on each Data Node, create a user account to perform the backup using the CREATE USER and GRANT statements.

For MariaDB Enterprise Server 10.5 and later:

```
CREATE USER 'mariadb-backup'@'localhost'
   IDENTIFIED BY 'mb_passwd';

GRANT RELOAD, PROCESS, LOCK TABLES, BINLOG MONITOR
   ON *.*
   TO 'mariadb-backup'@'localhost';
```

2. On each Data Node, grant the user account Spider uses when operating on Data Nodes sufficient privileges to lock any Data Tables using the GRANT statement.

For example, on the hq_server Data Node:

```
GRANT LOCK TABLES ON hq_sales.* TO 'spider_user'@'192.0.2.2';
```

On the eastern_server Data Node:

```
GRANT LOCK TABLES ON eastern_sales.* TO 'spider_user'@'192.0.2.2';
```

On the western_server Data Node:

```
GRANT LOCK TABLES ON western_sales.* TO 'spider_user'@'192.0.2.2';
```

3. On the Spider Node, acquire a read lock on Spider Tables using the LOCK TABLES statement:

```
LOCK TABLES spider_sharded_sales.invoices READ;
```

Keep this session open during the rest of the procedure.

The read lock will propagate to the Data Tables on each Data Node as well. The read locks will prevent the Data Tables from changing during the backup, so the backups on the Spider Node and Data Nodes are consistent.

4. On each Data Node, perform the backup using MariaDB Backup .

With MariaDB Backup 10.5 and later, use the mariadb-backup command:

```
$ sudo mariadb-backup --backup \
   --target-dir=/data/backups/full \
   --user=mariadb-backup \
   --password='mb_passwd'
```

5. On the Spider Node, after backing up each Data Node, perform a backup with MariaDB Backup .

With MariaDB Backup 10.5 and later, use the mariadb-backup command:

```
$ sudo mariadb-backup --backup \
   --target-dir=/data/backups/full \
   --user=mariadb-backup \
   --password='mb_passwd'
```

6. On the Spider Node, after the backup is complete, in your original session, use the UNLOCK TABLES statement to release the table locks:

```
UNLOCK TABLES;
```

7. On the Spider Node and each of the Data Nodes, prepare each of the backups using MariaDB Backup.

With MariaDB Backup 10.5 and later, use the mariadb-backup command:

```
$ sudo mariadb-backup --prepare \
   --target-dir=/data/backups/full
```

The Spider Node and Data Nodes now each have a complete backup of the data directory. Backups should be tested to confirm they are complete and consistent.

Restore Spider with MariaDB Backup

MariaDB Backup can restore a Sharded MariaDB Enterprise Spider topology from a backup taken with MariaDB Backup.

1. On the Spider Node and on each Data Node, stop the MariaDB Server process:

```
$ sudo systemctl stop mariadb
```

2. On the Spider Node and on each Data Node, empty the data directory:

```
$ sudo rm -fr /var/lib/mysql/*
```

3. On the Spider Node and on each Data Node, restore the backup for that server using MariaDB Backup.

With MariaDB Backup 10.5 and later, use the mariadb-backup command:

```
$ sudo mariadb-backup --copy-back \
   --target-dir=/data/backups/full
```

4. On the Spider Node and on each Data Node, confirm that the restored files are owned by the user that owns the MariaDB Server process:

```
$ sudo chown -R mysql:mysql /var/lib/mysql
```

5. On the Spider Node and on each Data Node, start the MariaDB Server process:

```
$ sudo systemctl start mariadb
```

6. On the Spider Node, query a Spider Table to test it:

```
SELECT * FROM spider_sharded_sales.invoices;
```

```
+-----------+------------+-------------+----------------------------+---------------+----------------+
| branch_id | invoice_id | customer_id | invoice_date               | invoice_total | payment_method |
+-----------+------------+-------------+----------------------------+---------------+----------------+
|         1 |          1 |           1 | 2020-05-10 12:35:10.000000 |       1087.23 | CREDIT_CARD    |
|         1 |          2 |           2 | 2020-05-10 14:17:32.000000 |       1508.57 | WIRE_TRANSFER  |
|         1 |          3 |           3 | 2020-05-10 14:25:16.000000 |        227.15 | CASH           |
|         2 |          1 |           2 | 2020-05-10 12:31:00.000000 |       1351.04 | CREDIT_CARD    |
|         2 |          2 |           2 | 2020-05-10 12:45:27.000000 |        162.11 | WIRE_TRANSFER  |
|         2 |          3 |           4 | 2020-05-10 13:11:23.000000 |        350.00 | CASH           |
|         3 |          1 |           5 | 2020-05-10 12:31:00.000000 |        111.50 | CREDIT_CARD    |
|         3 |          2 |           8 | 2020-05-10 12:45:27.000000 |       1509.23 | WIRE_TRANSFER  |
|         3 |          3 |           3 | 2020-05-10 13:11:23.000000 |       3301.66 | CASH           |
+-----------+------------+-------------+----------------------------+---------------+----------------+
```

MariaDB Dump

MariaDB Dump generates backup files containing the SQL statements necessary to recreate the database. MariaDB Dump is included with MariaDB Server and can be used to backup databases in Spider deployments. The MariaDB Client can then be used to restore databases from a MariaDB Dump backup.

Backup Spider with MariaDB Dump

MariaDB Dump can be used to create a logical backup of a Sharded MariaDB Enterprise Spider topology. MariaDB Dump must be used to backup the Spider Node and all Data Nodes. The Spider Node and Data Nodes must be locked to guarantee that the backups of the Spider Node and Data Nodes are consistent.

The backup of the Spider Node contains:

- Table definitions for Spider Tables
- Table definitions for all tables
- Table data for tables that use InnoDB, Aria, MyISAM, and MyRocks
- Definitions for any other user accounts, privileges, views, and triggers on the Spider Node

The backups of the Data Nodes contain:

- Table definitions for Data Tables
- Table definitions for all tables
- Table data for tables that use InnoDB, Aria, MyISAM, and MyRocks
- Definitions for any other user accounts, privileges, views, and triggers on the Data Node

The following procedure shows how to take a consistent backup of a Spider Node and Data Nodes deployed in a Sharded MariaDB Enterprise Spider topology.

1. On the Spider Node and on each Data Node, create a user account to perform the backup using the CREATE USER and GRANT statements:

```
CREATE USER 'mariadb-dump'@'localhost'
   IDENTIFIED BY 'md_passwd';

GRANT SELECT, INSERT, SHOW VIEW, TRIGGER, CREATE, ALTER, EVENT, RELOAD, LOCK TABLES
   ON *.*
   TO 'mariadb-dump'@'localhost';
```

2. On each Data Node, grant the user account Spider uses to operate on the Data Nodes sufficient privileges to lock any Data Tables using the GRANT statement.

For example, on the hq_server Data Node:

```
GRANT LOCK TABLES ON hq_sales.* TO 'spider_user'@'192.0.2.2';
```

On the `eastern_server` Data Node:

```
GRANT LOCK TABLES ON eastern_sales.* TO 'spider_user'@'192.0.2.2';
```

On the `western_server` Data Node:

```
GRANT LOCK TABLES ON western_sales.* TO 'spider_user'@'192.0.2.2';
```

3. On the Spider Node, acquire a read lock on Spider Tables using the LOCK TABLES statement:

```
LOCK TABLES spider_sharded_sales.invoices READ;
```

Keep this session open during the rest of the procedure.

The read lock will propagate to the Data Tables on each Data Node as well. The read locks will prevent the Data Tables from changing during the backup, so the backups of the Spider Node and Data Nodes are consistent.

4. On each Data Node, perform the backup using MariaDB Dump.
   With MariaDB Dump 10.5 and later, use the mariadb-dump command:

```
$ mariadb-dump \
   --user=mariadb-dump \
   --password='md_passwd' \
   --ssl-verify-server-cert \
   --ssl-ca ~/mariadb_chain.pem \
   --all-databases \
   --single-transaction \
   --default-character-set=utf8mb4 \
   > mariadb_dump.sql
```

5. On the Spider Node, once the Data Nodes are backed up, perform a backup using MariaDB Dump.

With MariaDB Dump 10.5 and later, use the mariadb-dump command:

```
$ mariadb-dump \
   --user=mariadb-dump \
   --password='md_passwd' \
   --ssl-verify-server-cert \
   --ssl-ca ~/mariadb_chain.pem \
   --all-databases \
   --single-transaction \
   --default-character-set=utf8mb4 \
   > mariadb_dump.sql
```

6. On the Spider Node, after the backups are complete, in your original session use the UNLOCK TABLES statement to release the table locks:

```
UNLOCK TABLES;
```

The Spider Node and the Data Node now each have a mariadb_dump.sql backup file. This backup file contains the SQL statements needed to recreate the schema on the Spider Node and the schema and data on the Data Nodes. Backups should be tested to confirm they are complete and consistent.

Restore Spider from MariaDB Dump

MariaDB Client can restore a Sharded MariaDB Enterprise Spider topology from a backup taken with MariaDB Dump backup.

1. Stop all traffic to the Spider Node and each Data Node.
2. On the Spider Node, restore the backup for that server using MariaDB Client.

With MariaDB Client 10.5 and later, use the mariadb command:

```
$ mariadb \
   --user=mariadb-dump \
   --password='md_passwd' \
   --ssl-verify-server-cert \
   --ssl-ca ~/mariadb_chain.pem \
   --default-character-set=utf8mb4 \
   < mariadb_dump.sql
```

3. On the Data Node, restore the backup for that server using MariaDB Client.

With MariaDB Client 10.5 and later, use the mariadb command:

```
$ mariadb \
   --user=mariadb-dump \
   --password='md_passwd' \
   --ssl-verify-server-cert \
   --ssl-ca ~/mariadb_chain.pem \
   --default-character-set=utf8mb4 \
   < mariadb_dump.sql
```

4. On the Spider Node, query a Spider Table to test it:

```
SELECT * FROM spider_sharded_sales.invoices;
```

```
+-----------+------------+-------------+---------------------------+--------------+----------------+
| branch_id | invoice_id | customer_id | invoice_date              | invoice_total | payment_method |
+-----------+------------+-------------+---------------------------+--------------+----------------+
|         1 |          1 |           1 | 2020-05-10 12:35:10.000000 |       1087.23 | CREDIT_CARD    |
|         1 |          2 |           2 | 2020-05-10 14:17:32.000000 |       1508.57 | WIRE_TRANSFER  |
|         1 |          3 |           3 | 2020-05-10 14:25:16.000000 |        227.15 | CASH           |
|         2 |          1 |           2 | 2020-05-10 12:31:00.000000 |       1351.04 | CREDIT_CARD    |
|         2 |          2 |           2 | 2020-05-10 12:45:27.000000 |        162.11 | WIRE_TRANSFER  |
|         2 |          3 |           4 | 2020-05-10 13:11:23.000000 |        350.00 | CASH           |
|         3 |          1 |           5 | 2020-05-10 12:31:00.000000 |        111.50 | CREDIT_CARD    |
|         3 |          2 |           8 | 2020-05-10 12:45:27.000000 |       1509.23 | WIRE_TRANSFER  |
|         3 |          3 |           3 | 2020-05-10 13:11:23.000000 |       3301.66 | CASH           |
+-----------+------------+-------------+---------------------------+--------------+----------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Spider Differences Between SpiderForMySQL and MariaDB

SQL Syntax

- With `SpiderForMySQL`, the CREATE TABLE statement uses `CONNECTION` to define spider table variables whereas MariaDB uses `COMMENT`.

Features

- HANDLER can not be translated to SQL in MariaDB
- Concurrent background search is not yet implemented in MariaDB
- Vertical partitioning storage engine VP is not implemented in MariaDB
- `CREATE TABLE` can use table discovery in MariaDB
- JOIN performance improvement using join_cache_level>1 `and` join_buffer_size in MariaDB

This page is licensed: CC BY-SA / Gnu FDL

Spider Feature Matrix

Not complete yet - still being updated

F*()* *Federation only, P()*partitioning only.
Spider column is for SpiderForMySQL found on the Spider web sIte.

Feature	Spider	10.0
Clustering and High Availability		
Commit, Rollback transactions on multiple backend	Yes	Yes
Multiplexing to a number of replicas using xa protocol 2PC	Yes	Yes
Split brain resolution based on a majority decision, failed node is remove from the list of replicas	Yes	Yes

Feature	Spider	10.0
Enable a failed backend to re enter the cluster transparently	No	No
Synchronize DDL to backend, table modification, schema changes	No	No
Synchronize DDL to other Spider	No	No
GTID tracking per table on XA error	No	Yes
Transparent partitioning	No	No
Covered by generic SQL test case	Yes	Yes
Heterogeneous Backends		
MariaDB and MySQL database backend	Yes	Yes
Oracle database backend, if build from source against the client library ‘ORACLE_HOME’	Yes	Yes
Local table attachment	Yes	Yes
Performance		
Index Condition Pushdown	No	No
Engine Condition Pushdown	Yes	Yes
Concurrent backend scan	Yes	No
Concurrent partition scan	Yes	No
Batched key access	Yes	Yes
Block hash join	No	Yes
HANDLER backend propagation	Yes	Yes
HANDLER backend translation from SQL	Yes	Yes
HANDLER OPEN cache per connection	No	Yes
HANDLER use prepared statement	No	Yes
HANDLER_SOCKET protocol backend propagation	Yes	Yes
HANDLER_SOCKET backend translation from SQL	No	No
Map reduce for ORDER BY … LIMIT	Yes	Yes
Map reduce for MAX & MIN & SUM	Yes	Yes
Map reduce for some GROUP BY	Yes	Yes
Batch multiple WRITES in auto commit to reduce network round trip	Yes	Yes
Relaxing backend consistency	Yes	Yes
Execution Control		
Configuration at table and partition level, settings can change per data collection	Yes	Yes
Configurable empty result set on errors. For API that does not have transactions replay	Yes	Yes
Query Cache tuning per table of the on remote backend	Yes	Yes
Index Hint per table imposed on remote backend	Yes	Yes
SSL connections to remote backend connections	Yes	Yes
Table definition discovery from remote backend	Yes	F(*)
Direct SQL execution to backend via UDF	Yes	Yes
Table re synchronization between backends via UDF	Yes	Yes
Maintain Index and Table Statistics of remote backends	Yes	Yes
Can use Independent Index and Table Statistics	No	Yes

Feature	Spider	10.0
Maintain local or remote table increments	Yes	Yes
LOAD DATA INFILE translate to bulk inserting	Yes	Yes
Performance Schema Probes	Yes	Yes
Load Balance Reads to replicate weight control	Yes	Yes
Fine tuning tcp timeout, connections retry	Yes	Yes

This page is licensed: CC BY-SA / Gnu FDL

Spider Use Cases

Introduction

This article will cover simple working examples for some standard use cases for Spider. The example are illustrated using a sales opportunities table to be consistent throughout. In some cases the actual examples are contrived but are used to illustrate the varying syntax options.

Basic setup

Have 3 or more servers available and Install MariaDB on each of these servers:

- spider server which will act as the front end server hosting the spider storage engine.
- backend1 which will act as a backed server storing data
- backend2 which will act as a second backend server storing data

Follow the instructions here to enable the Spider storage engine on the spider server:

```
INSTALL SONAME 'ha_spider';
```

Setting the SUPER privilege for the Spider user on data nodes or alternatives to avoid privilege issues

When explicitly setting the spider_internal_sql_log_off system variable, please note that Spider will execute matching SET SQL_LOG_OFF statements on each of the data nodes. It will attempt to do this on the data nodes using the SUPER privilege, which thus requires one to grant this privilege to the Spider user on the data nodes.

If the Spider user on the data note is not configured with the SUPER privilege, you may encounter issues when working with Spider tables like ERROR 1227 (42000): Access denied for the missing SUPER privilege. To avoid this, don't explicitly set spider_internal_sql_log_off, or set it to -1, or grant the SUPER privilege to the Spider user on the data node.

Create accounts for spider to connect with on backend servers

Spider needs a remote connection to the backend server to actually perform the remote query. So this should be setup on each backend server. In this case 172.21.21.2 is the ip address of the spider node limiting access to just that server.

```
backend1> mysql
GRANT ALL ON test.* TO spider@'172.21.21.2' IDENTIFIED BY 'spider';
backend2> mysql
GRANT ALL ON test.* TO spider@'172.21.21.2' IDENTIFIED BY 'spider';
```

Now verify that these connections can be used from the spider node (here 172.21.21.3 = backend1 and 172.21.21.4 = backend2):

```
spider> mysql -u spider -p -h 172.21.21.3 test
spider> mysql -u spider -p -h 172.21.21.4 test
```

Create table on backend servers

The table definition should be created in the test database on both backend1 and backend2 servers:

```
CREATE TABLE opportunities (
id INT,
accountName VARCHAR(20),
name VARCHAR(128),
owner VARCHAR(7),
amount DECIMAL(10,2),
closeDate DATE,
stageName VARCHAR(11),
PRIMARY KEY (id),
KEY (accountName)
) ENGINE=InnoDB;
```

Create server entries on spider server

While the connection information can also be specified inline in the comment or (from MariaDB 10.8.1) as table options, it is cleaner to define a server object representing each remote backend server connection:

```
CREATE SERVER backend1 FOREIGN DATA WRAPPER MYSQL OPTIONS
(HOST '172.21.21.3', DATABASE 'test', USER 'spider', PASSWORD 'spider', PORT 3306);
CREATE SERVER backend2 FOREIGN DATA WRAPPER MYSQL OPTIONS
(HOST '172.21.21.4', DATABASE 'test', USER 'spider', PASSWORD 'spider', PORT 3306);
```

Unable to Connect Errors

Bear in mind, if you ever need to remove, recreate or otherwise modify the server definition for any reason, you need to also execute a FLUSH TABLES statement. Otherwise, Spider continues to use the old server definition, which can result in queries raising the error

```
Error 1429: Unable to connect to foreign data source
```

If you encounter this error when querying Spider tables, issue a FLUSH TABLES statement to update the server definitions.

```
FLUSH TABLES;
```

Use case 1: remote table

In this case, a spider table is created to allow remote access to the opportunities table hosted on backend1. This then allows for queries and remote dml into the backend1 server from the spider server:

```
CREATE TABLE opportunities (
id INT,
accountName VARCHAR(20),
name VARCHAR(128),
owner VARCHAR(7),
amount DECIMAL(10,2),
closeDate DATE,
stageName VARCHAR(11),
PRIMARY KEY (id),
KEY (accountName)
) ENGINE=spider COMMENT='wrapper "mysql", srv "backend1" , TABLE "opportunities"';
```

Use case 2: sharding by hash

See also hash-partitioning-type.

In this case a spider table is created to distribute data across backend1 and backend2 by hashing the id column. Since the id column is an incrementing numeric value the hashing will ensure even distribution across the 2 nodes.

```
CREATE TABLE opportunities (
id INT,
accountName VARCHAR(20),
name VARCHAR(128),
owner VARCHAR(7),
amount DECIMAL(10,2),
closeDate DATE,
stageName VARCHAR(11),
PRIMARY KEY (id),
KEY (accountName)
) ENGINE=spider COMMENT='wrapper "mysql", TABLE "opportunities"'
 PARTITION BY HASH (id)
(
 PARTITION pt1 COMMENT = 'srv "backend1"',
 PARTITION pt2 COMMENT = 'srv "backend2"'
) ;
```

Use case 3: sharding by range

See also range-partitioning-type.

In this case a spider table is created to distribute data across backend1 and backend2 based on the first letter of the accountName field. All accountNames that start with the letter L and prior are stored in backend1 and all other values stored in backend2. Note that the accountName column must be added to the primary key which is a requirement of MariaDB partitioning:

```
CREATE TABLE opportunities (
id INT,
accountName VARCHAR(20),
name VARCHAR(128),
owner VARCHAR(7),
amount DECIMAL(10,2),
closeDate DATE,
stageName VARCHAR(11),
PRIMARY KEY (id, accountName),
KEY(accountName)
) ENGINE=spider COMMENT='wrapper "mysql", TABLE "opportunities"'
 PARTITION BY RANGE COLUMNS (accountName)
(
 PARTITION pt1 VALUES LESS THAN ('M') COMMENT = 'srv "backend1"',
 PARTITION pt2 VALUES LESS THAN (MAXVALUE) COMMENT = 'srv "backend2"'
) ;
```

Use case 4: sharding by list

See also list-partitioning-type.

In this case a spider table is created to distribute data across backend1 and backend2 based on specific values in the owner field. Bill, Bob, and Chris are stored in backend1 and Maria and Olivier stored in backend2. Note that the owner column must be added to the primary key which is a requirement of MariaDB partitioning:

```
CREATE TABLE opportunities (
id INT,
accountName VARCHAR(20),
name VARCHAR(128),
owner VARCHAR(7),
amount DECIMAL(10,2),
closeDate DATE,
stageName VARCHAR(11),
PRIMARY KEY (id, owner),
KEY(accountName)
) ENGINE=spider COMMENT='wrapper "mysql", TABLE "opportunities"'
 PARTITION BY LIST COLUMNS (owner)
(
 PARTITION pt1 VALUES IN ('Bill', 'Bob', 'Chris') COMMENT = 'srv "backend1"',
 PARTITION pt2 VALUES IN ('Maria', 'Olivier') COMMENT = 'srv "backend2"'
) ;
```

With MariaDB 10.2 the following partition clause can be used to specify a default partition for all other values, however this must be a distinct partition / shard:

```
PARTITION partition_name DEFAULT
```

For a complete list of partition types, see partitioning-types.

This page is licensed: CC BY-SA / Gnu FDL

Spider Installation

The Spider storage engine supports partitioning and XA transactions, and allows tables of different database host instances to be handled as if they were on the same instance.

To make sensible use of Spider, you need two or more database host instances, typically running on separate hosts. Those database instances can be two or more MariaDB Server hosts, or a MariaDB Server host and an Oracle Database, etc. The Spider node is the MariaDB server that receives queries from your application. It then processes these queries, connecting to one or more data nodes. The data nodes are the MariaDB servers that actually store the table data.

In order for this to work, you need to configure the data nodes to accept queries from the Spider node and you need to configure the Spider node to use the data nodes as remote storage.

You don't need to install any additional packages to use it, but it does require some configuration.

Configuring Data Nodes

Spider deployments use data nodes to store the actual table data. In order for a MariaDB server to operate as a data node for Spider, you need to create a table or tables on which to store the data and configure the server to accept client connections from the Spider node.

For instance, first create the table:

```
CREATE TABLE test.spider_example (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(50)
) ENGINE=InnoDB;
```

Next, create a user for the Spider node and set a password for that user. For the sake of the example, assume the Spider node is at the IP address 192.168.1.1:

```
CREATE USER spider@192.168.1.1;

SET PASSWORD FOR spider@192.168.1.1 = PASSWORD('passwd');
```

Then grant the `spider` user privileges on the example table.

```
GRANT ALL ON test.spider_example TO spider@192.168.1.1;
```

The data node is now ready for use. You can test it by attempting to connect the MariaDB client to the data from the Spider node. For instance, assuming the data node is at the IP address 192.168.1.5, SSH into the Spider node then try to establish a client connection.

```
$ mysql -u spider -p -h 192.168.1.5 test -e "SHOW TABLES;"
+----------------+
| Tables_in_test |
+----------------+
| spider_example |
+----------------+
```

Install Spider on Spider Node

The Spider storage engine must be installed on the Spider node. The Spider node is the MariaDB server that receives queries for the table, (in this case `test.spider_example`). It then uses the Spider storage engine to connect to the tables on the data nodes to retrieve data and return the result-set.

To install the Spider storage engine, complete the installation process shown below.

Step 1: Install Spider Package

APT

APT (Debian, Ubuntu, etc.)

To install the Spider storage engine, execute the following command:

```
$ sudo apt install mariadb-plugin-spider
```

YUM

YUM (CentOS etc.)

To install the Spider storage engine, execute the following command:

```
sudo yum install MariaDB-spider-engine
```

OTHER LINUX DISTRIBUTIONS

On other Linux distributions, the Spider storage engine is installed with MariaDB Server.

Step 2: Load the Spider Plugin

The Spider storage engine can be loaded as a normal plugin, and Spider automatically creates its dependencies. There are two primary ways to load the plugin.

The plugin can be loaded dynamically without a server restart by executing `INSTALL SONAME` or `INSTALL PLUGIN`:

```
INSTALL SONAME "ha_spider";
```

Alternatively, the plugin can be loaded by adding `plugin_load_add=ha_spider` to a configuration file:

```
[mariadb]
...
plugin_load_add = "ha_spider"
```

If the plugin is loaded in a configuration file, then the server will load the plugin after the server has been restarted.

Loading the plugin also creates a series of new tables in the `mysql` database, including:

- spider_xa
- spider_xa_member
- spider_xa_failed_log
- spider_tables
- spider_link_mon_servers
- spider_link_failed_log
- spider_table_position_for_recovery
- spider_table_sts
- spider_table_crd

Step 3: Verify Loading of the Spider Plugin

You can verify that the Spider plugin has been loaded by querying the `information_schema.ENGINES` table:

```
SELECT ENGINE, SUPPORT
FROM information_schema.ENGINES
WHERE ENGINE = 'SPIDER';
+--------------------+---------+
| ENGINE             | SUPPORT |
+--------------------+---------+
| SPIDER             | YES     |
+--------------------+---------+
```

If the Spider plugin is not loaded, then the query does not return any results.

Configuring Spider Nodes

With the data node or data nodes configured, you can set up the Spider node to use them. The Spider node is the MariaDB server that receives queries for the table, (in this case `test.spider_example`). It then uses the Spider storage engine to connect to the tables on the data nodes to retrieve data and return the result-set.

Configure the Server

In order to connect the Spider node to the data nodes, you may issue a CREATE SERVER statement for each data node. You can then use the server definition in creating the Spider table.

```
CREATE SERVER dataNode1 FOREIGN DATA WRAPPER mysql
OPTIONS (
   HOST '192.168.1.5',
   DATABASE 'test',
   USER 'spider',
   PASSWORD 'passwd',
   PORT 3306);
```

In the event that you need to modify or replace this server after setting up the Spider table, remember to issue a FLUSH statement to update the server definition.

```
FLUSH TABLES;
```

Alternatively, you could also choose not to create a server, but specify the connection info in the spider table creation.

Create the Table

With the data nodes set up and the Spider node configured for use, you can create the Spider table. The Spider table must have the same column definitions as the tables on the data nodes. Spider can be configured through table parameters passed to the `COMMENT` or `CONNECTION` option.

```
CREATE TABLE test.spider_example (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(50)
) ENGINE=Spider
COMMENT='wrapper "mysql", srv "dataNode1", table "spider_example"';
```

This configures Spider to use the server `dataNode1`, (defined above), as a remote table. Any data you write to this table is actually stored on the MariaDB server at 192.168.1.5.

Alternatively, starting from MariaDB 10.8.1, one can specify spider table parameters using table options:

```
CREATE TABLE test.spider_example (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(50)
) ENGINE=Spider
REMOTE_SERVER=dataNode1 REMOTE_TABLE=spider_example;
```

This page is licensed: CC BY-SA / Gnu FDL

Spider Cluster Management

Direct SQL

Direct SQL is a way to map reduced execution on remote backends and store the results in a local table. This can either be sequential, using the UDF function spider_direct_sql, or concurrently, using spider_bg_direct_sql.

```
spider1 backend << EOF
CREATE TEMPORARY TABLE res
(
  id INT(10) UNSIGNED NOT NULL,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT ''
) ENGINE=MEMORY;

SELECT spider_direct_sql(
'SELECT * FROM sbtest s  WHERE s.id IN(10,12,13)',
  'res',
  concat('host "', host, '", port "', port, '", user "', username, '", password "', password, '",
database "', tgt_db_name, '"')
) a
FROM
  mysql.spider_tables
WHERE
  db_name = 'backend' AND table_name LIKE 'sbtest#P#pt%';

SELECT * FROM res;
EOF
```

Or if you are using a SERVER:

```
SELECT spider_direct_sql(
  'SELECT * FROM sbtest s  WHERE s.id IN(10,12,13)',
  'res',
  concat('server "', server, '"')
) a
  FROM mysql.spider_tables
  WHERE db_name = 'backend' AND table_name LIKE 'sbtest#P#pt%' ;
```

The default for spider_bg_direct_sql is to access concurrently all backends. If you have multiple partitions store inside a single backend, you still can increase parallelism affecting different channels to each partitions.

```
CREATE TEMPORARY TABLE res
(
  id INT(10) UNSIGNED NOT NULL ,
  col_microsec DATETIME(6) DEFAULT NOW(8),
  db VARCHAR(20)
) ENGINE=MEMORY;

SELECT spider_bg_direct_sql( 'SELECT count(*) ,min(NOW(6)),min(DATABASE())) FROM sbtest',   'res',
concat('srv "', server,'" cch ',@rn:=@rn+1  ) ) a  FROM    mysql.spider_tables,(SELECT @rn:=1) t2  WHERE
db_name = 'bsbackend' AND table_name LIKE 'sbtest#P#pt%';
```

Direct Handler Socket

MariaDB starting with 10.8.1

The Spider Handler Socket support has been removed, see MDEV-26858.

Check that Handler Socket is running on the backend nodes

```
:~# backend2 -e "show variables like 'handler%'"
+------------------------------+---------------+
| Variable_name                | Value         |
+------------------------------+---------------+
| handlersocket_accept_balance | 0             |
| handlersocket_address        | 192.168.0.201 |
| handlersocket_backlog        | 32768         |
| handlersocket_epoll          | 1             |
| handlersocket_plain_secret   |               |
| handlersocket_plain_secret_wr |               |
| handlersocket_port           | 20500         |
| handlersocket_port_wr        | 20501         |
| handlersocket_rcvbuf         | 0             |
| handlersocket_readsize       | 0             |
| handlersocket_sndbuf         | 0             |
| handlersocket_threads        | 4            |
| handlersocket_threads_wr     | 1             |
| handlersocket_timeout        | 300           |
| handlersocket_verbose        | 10            |
| handlersocket_wrlock_timeout | 12            |
+------------------------------+---------------+
```

```
spider1 backend << EOF
CREATE TEMPORARY TABLE res
(
  id INT(10) UNSIGNED NOT NULL,
  k INT(10) UNSIGNED NOT NULL DEFAULT '0',
  c CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT ''
) ENGINE=MEMORY;

SELECT spider_direct_sql('1\t=\t1\t2\t100000\t0','res', 'host "192.168.0.202", table "sbtest", database
"test", port "20500", access_mode "1"');
```

Inter Nodes Copy Table

MariaDB starting with 10.8.1

The UDF spider_copy_tables relies on Spider's high availability feature, which has been deprecated (MDEV-28479), and are deleted. Please use other high availability solutions like replication or galera-cluster.

The UDF function spider_copy_tables is available for copying table data from the source link ID to the destination link ID list without stopping your service for copying

```
spider_copy_tables(Spider table name, source link ID, destination link ID list[, parameters])
```

- `Returns 1` if copying data succeeded.
- `Returns 0` if copying data failed.

If the Spider table is partitioned, you must set "Spider table name" with a part name such as "table_name#P#part_name".

You can check the table name and the link ID with the part name using the following SQL:

```
SELECT table_name FROM mysql.spider_tables;
```

General Log

To capture all queries sent to remote backends on a `Spider Node` :

```
SET GLOBAL general_log=ON;
SET GLOBAL spider_general_log=ON;
SET GLOBAL spider_log_result_errors=1;
SET GLOBAL spider_log_result_error_with_sql=3;
```

Compiling in Debug Mode

See Compiling MariaDB for Debugging and Creating a Trace File.

Report the issue in MariaDB JIRA (see Reporting Bugs) or to the MariaDB Corporation support center.

Compiling in Static

Available since version 3.1.14

To activate spider as a static plugin change "MODULE_ONLY" to "MANDATORY" in storage/spider/CMakeList.txt before compiling

Note that Spider UDF functions will not work with such settings.

Status Variables

A number of new status variables have been introduced, see Spider Status Variables for a complete list.

Information Schema Tables

- A new Information Schema table is installed - SPIDER_ALLOC_MEM.

```
+-------------------+---------------------+------+-----+---------+-------+
| Field             | Type                | Null | Key | Default | Extra |
+-------------------+---------------------+------+-----+---------+-------+
| ID                | int(10) unsigned    | NO   |     | 0       |       |
| FUNC_NAME         | varchar(64)         | YES  |     | NULL    |       |
| FILE_NAME         | varchar(64)         | YES  |     | NULL    |       |
| LINE_NO           | int(10) unsigned    | YES  |     | NULL    |       |
| TOTAL_ALLOC_MEM   | bigint(20) unsigned | YES  |     | NULL    |       |
| CURRENT_ALLOC_MEM | bigint(20)          | YES  |     | NULL    |       |
| ALLOC_MEM_COUNT   | bigint(20) unsigned | YES  |     | NULL    |       |
| FREE_MEM_COUNT    | bigint(20) unsigned | YES  |     | NULL    |       |
+-------------------+---------------------+------+-----+---------+-------+
```

From MariaDB 10.5, Spider installs another Information Schema table, SPIDER_WRAPPER_PROTOCOLS.

Performance Schema

The Performance schema is commonly used to troubleshoot issues that consume time inside your workload. The Performance schema should not be activated for servers that are experimenting constant heavy load, but most of time it is acceptable to lose 5% to 20% additional CPU to keep track of server internals execution.

To activate the performance schema, use the performance_schema system variable and add the following to the server section of the MariaDB configuration file.

```
performance_schema=ON
```

Activate the Spider probes to be monitored.

```
UPDATE performance_schema.setup_instruments SET
  ENABLED='YES', TIMED='yes' WHERE NAME LIKE '%spider%';
```

Run your queries …

And check the performance metrics. Remove specific Spider metrics to have a more global view.

```
SELECT * FROM performance_schema.events_waits_summary_global_by_event_name
  WHERE COUNT_STAR<>0 AND EVENT_NAME LIKE '%spider%'
  ORDER BY SUM_TIMER_WAIT DESC LIMIT 10;
```

This page is licensed: CC BY-SA / Gnu FDL

Spider Case Studies

A list of users or clients that are using Spider and agree to be referenced:

- Tencent Games. They handle 100TB data on 396 Spider nodes and 2800 data nodes. They use this cluster for their online games.
- Kadokawa Corporation
- MicroAd, Inc.
- Sansan, Inc.
- teamLab Inc.
- CCM Benchmark ccm-escape-case-study-skysql-paris-meetup-17122013
- Softlink galaxy-big-data-with-mariadb
- Gavo Spider-MariaDB.pdf
- Blablacar
  Using for storing various logs

- Believe Digital
  Using for back office analytics queries to aggregate multi billions tables in real time

This page is licensed: CC BY-SA / Gnu FDL

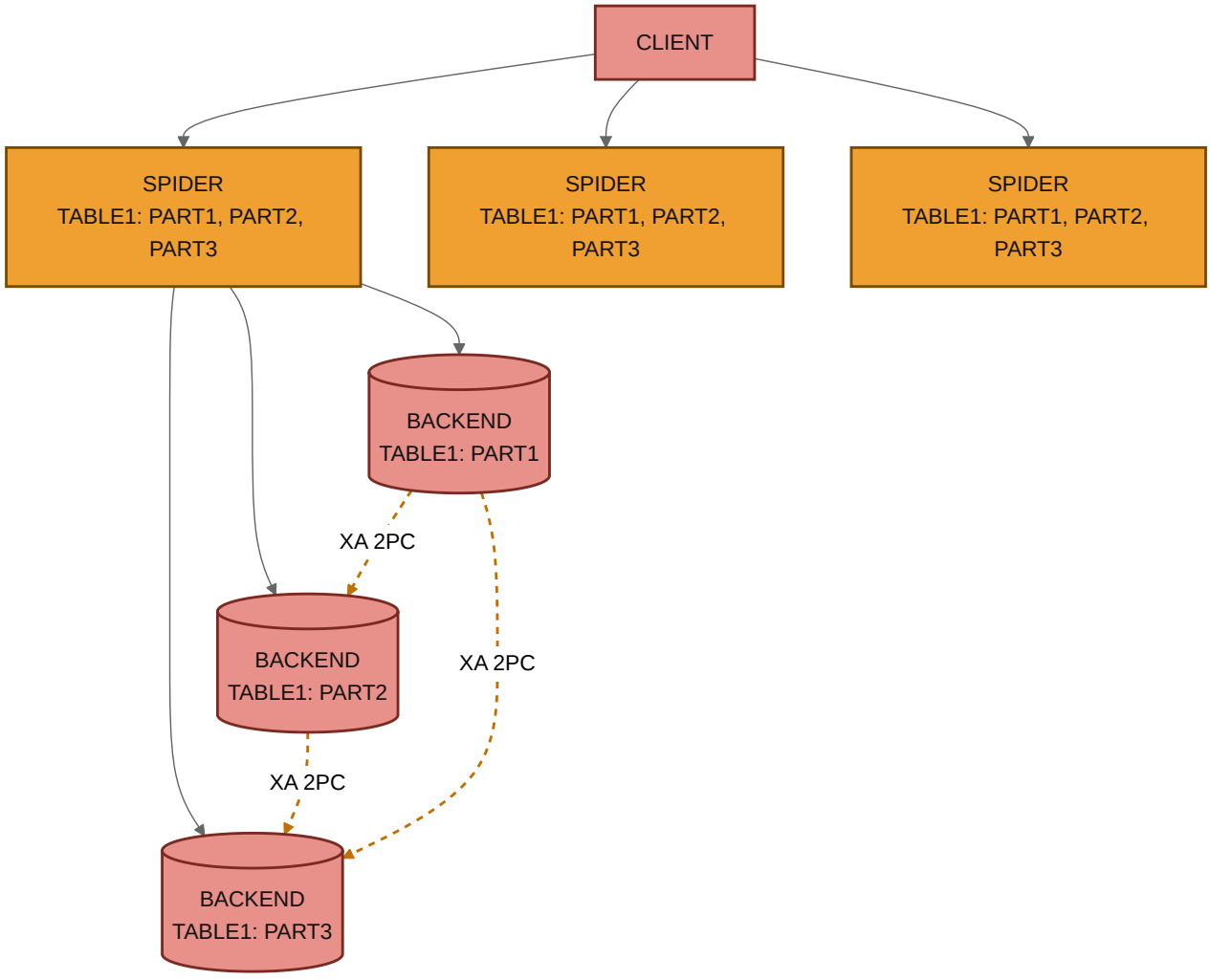
Spider Benchmarks

This is best run on a cluster of 3 nodes intel NUC servers 12 virtual cores model name : Intel® Core(TM) i3-3217U CPU @ 1.80GHz

All nodes have been running a mysqlslap client attached to the local spider node in the best run.

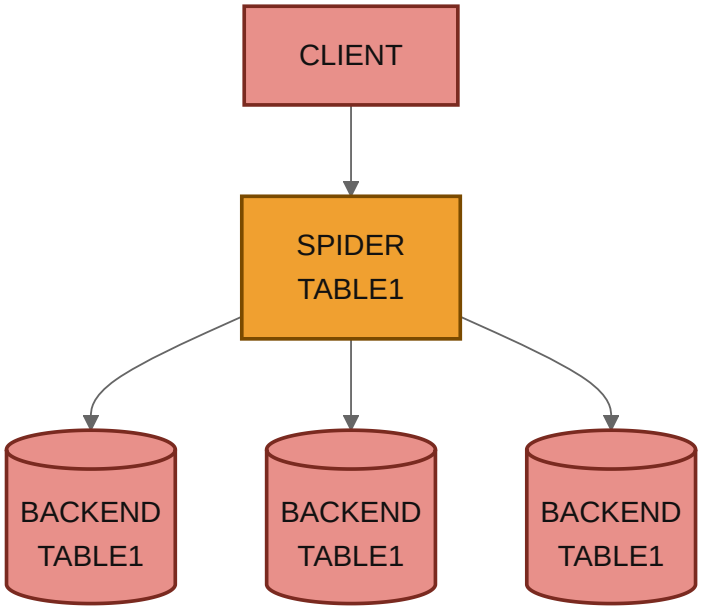
```
/usr/local/skysql/mysql-client/bin/mysqlslap --user=skysql --password=skyvodka --host=192.168.0.201 --
port=5012 -i1000000 -c32 -q "insert into test(c) values('0-31091-138522330')" --create-schema=test
```

`spider_conn_recycle_mode=1;`



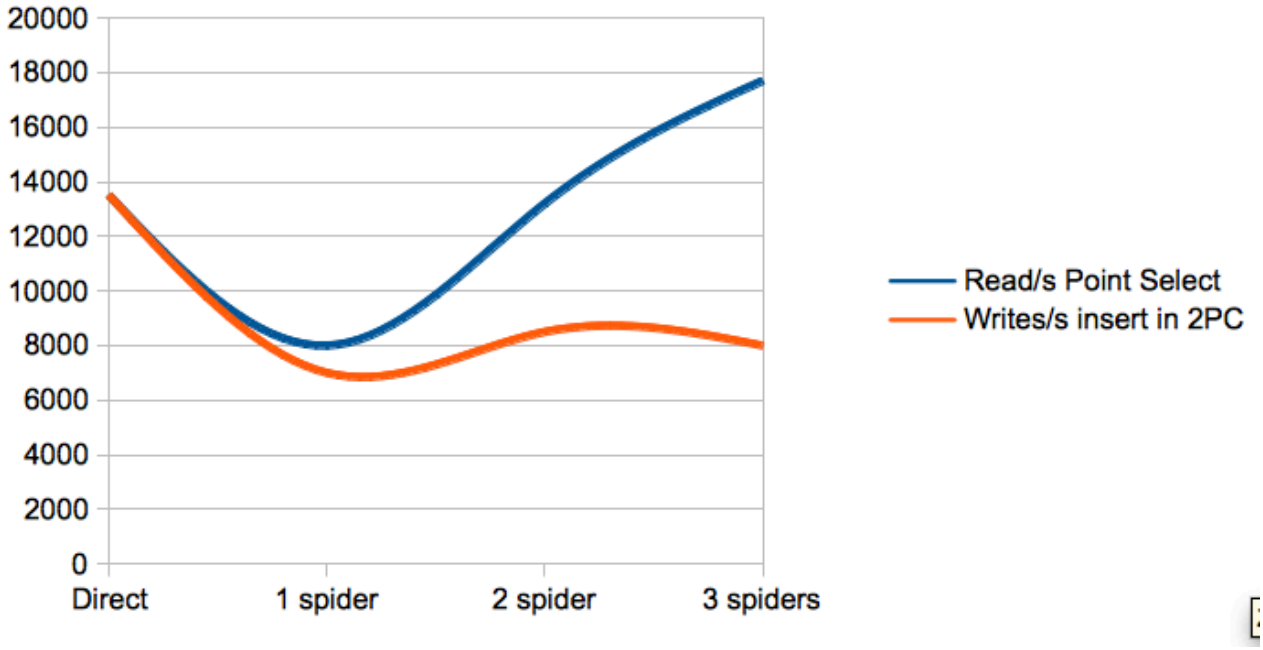
Spider sharding architecture: three Spider nodes route to three backend shards, each holding one partition of TABLE1, coordinated across backends by XA two-phase commit.

The read point select is produce with a 10M rows sysbench table



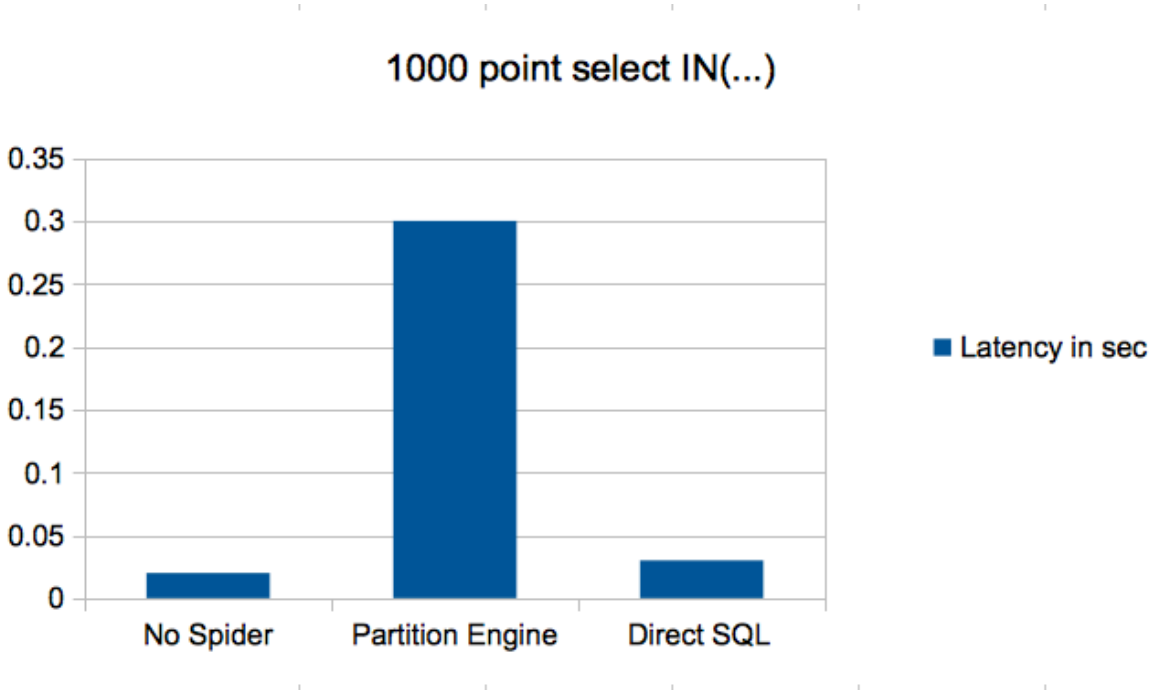
Single Spider node distributing client queries to three backend nodes, each holding a full copy of TABLE1.

The write insert a single string into a memory table



spbench6

Before Engine Condition Push Down patch .



benchspider7

Spider can benefit by 10% additional performance with Independent Storage Engine Statistics.

```
SET global use_stat_tables='preferably';
USE backend;
ANALYZE TABLE sbtest;
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema SPIDER_WRAPPER_PROTOCOLS Table

MariaDB starting with 10.5.4

The Information Schema `SPIDER_WRAPPER_PROTOCOLS` table is installed along with the Spider storage engine from MariaDB 10.5.4.

It contains the following columns:

Column	Type	Description
WRAPPER_NAME	varchar(64)	
WRAPPER_VERSION	varchar(20)	
WRAPPER_DESCRIPTION	longtext	
WRAPPER_MATURITY	varchar(12)	

This page is licensed: CC BY-SA / Gnu FDL

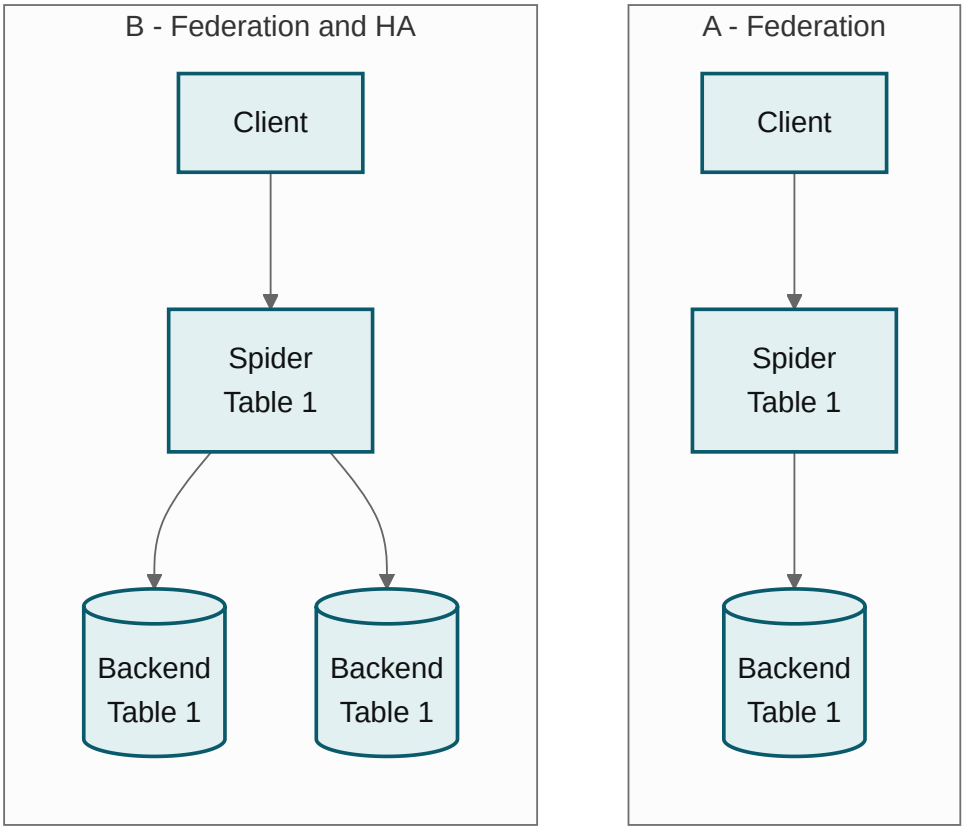
Spider Storage Engine Core Concepts

A typical Spider deployment has a shared-nothing clustered architecture. The system works with any inexpensive hardware, and with a minimum of specific requirements for hardware or software. It consists of a set of computers, with one or more MariaDB processes known as nodes.

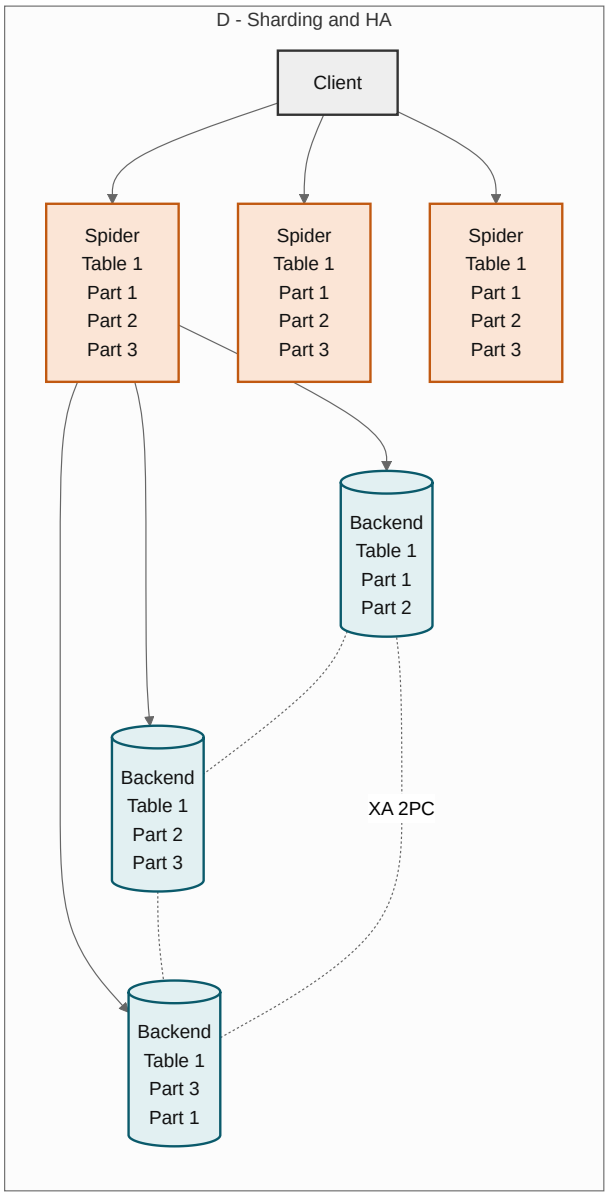
The nodes that store the data are designed as `Backend Nodes`, and can be any MariaDB, MySQL, Oracle server instances using any storage engine available inside the backend.

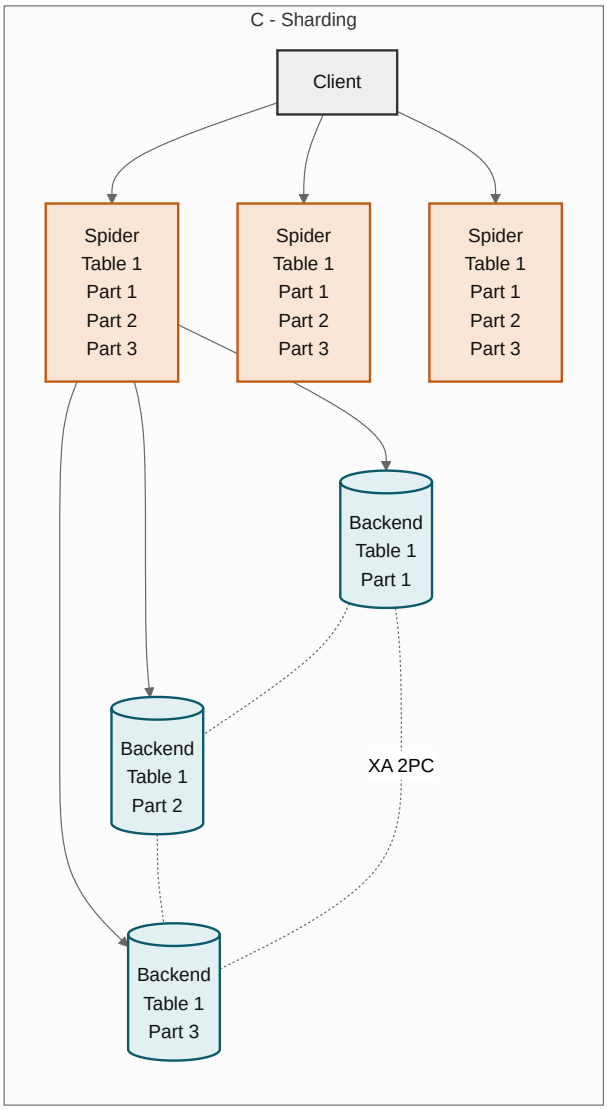
The `Spider Proxy Nodes` are instances running at least MariaDB 10. `Spider Proxy Nodes` are used to declare per table attachment to the backend nodes. In addition `Spider Proxy Nodes` can be setup to enable the tables to be split and mirrored to multiple `Backend Nodes`.

Spider Common Usage



Topology A (federation) routes a client through Spider to a single backend node; topology B (federation with HA) routes the same client through Spider to two redundant backend nodes.





Topology C (sharding) routes a client through three Spider nodes to three non-overlapping backend shards; topology D (sharding with HA) uses the same Spider layer but stores each partition redundantly across two backend shards, all coordinated via XA two-phase commit.

In the default high availability setup Spider Nodes produce SQL errors when a backend server is not responding. Per table monitoring can be setup to enable availability in case of unresponsive backends `monotoring_bg_kind=1` or `monotoring_bg_kind=2`. The Monitoring Spider Nodes are inter-connected with usage of the system table `mysql.link_mon_servers` to manage network partitioning. Better known as split brain, an even number of `Spider Monitor Nodes` should be setup to allow a consensus based on the majority. Rather a single separated shared `Monitoring Node` instance or a minimum set of 3 `Spider Nodes`. More information can be found here.

MariaDB starting with 10.7.5

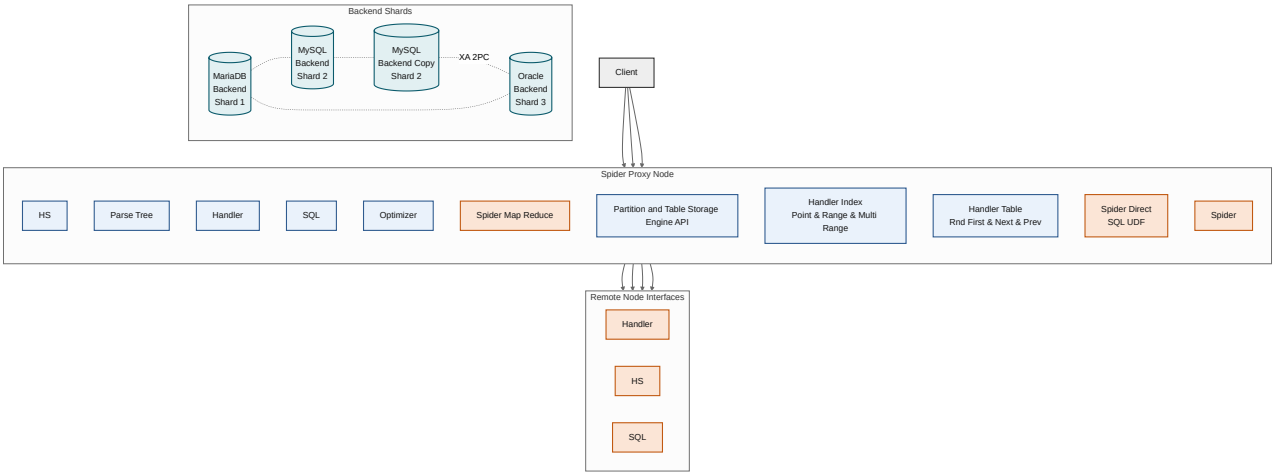
Spider's high availability feature has been deprecated (MDEV-28479), and are deleted. Please use other high availability solutions like replication or galera-cluster.

Spider Storage Engine Federation

Spider is a pluggable Storage Engine, acting as a proxy between the optimizer and the remote backends. When the optimizer requests multiple calls to the storage engine, Spider enforces consistency using the 2 phase commit

protocol to the backends and by creating transactions on the backends to preserve atomic operations for a single SQL execution.
Preserving atomic operation during execution is used at multiple levels in the architecture. For the regular optimizer plan, it refers to `multiple split reads` and for concurrent partition scans, it will refer to `semi transactions`.

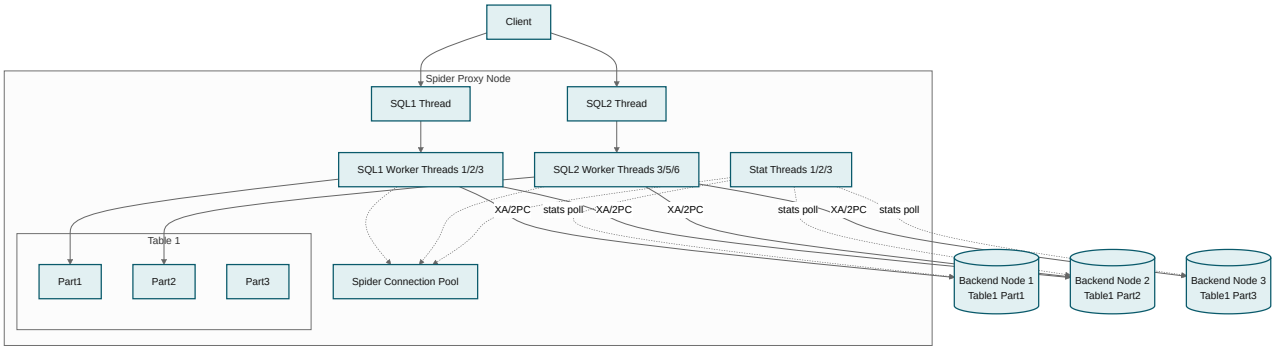
Costly queries can be more efficient when it is possible to fully push down part of the execution plan on each backend and reduce the result afterwards. Spider enables such execution with some direct execution shortcuts.



The Spider proxy node layers client-facing Handler, HS, and SQL interfaces (plus the optimizer, partition API, and Spider Map Reduce/Direct SQL UDF extensions) over the core Spider engine, which forwards to remote node interfaces fronting four backend shards kept consistent via XA two-phase commit.

Spider Threading Model

Spider uses the per partitions and per table model to concurrently access the remote backend nodes. For memory workload that property can be used to define multiple partitions on a single remote backend node to better adapt the concurrency to available CPUs in the hardware.



Spider's per-partition, per-table threading model: SQL1/SQL2 worker threads coordinate over a shared connection pool and commit via XA two-phase commit across three backend nodes, while stat threads poll statistics independently.

Spider maintains an internal dictionary of Table and Index statistics based on separated threads. The statistics are pulled per default on a time line basis and refer to `crd` for cardinality and `sts` for table status.

Spider Memory Model

Spider stores resultsets into memory, but spider_quick_mode=3 stores resultsets into internal temporary tables if the resultsets are larger than quick_table_size.

Specifying Connection Information for Spider Tables

Spider tables connect to remote servers by reading connection information from one or more sources. MariaDB supports several methods for defining this information, each introduced at different versions. When more than one method is present, a defined precedence order determines which takes effect.

Connection information for Spider tables can be provided in different ways:

- Foreign server options (using `CREATE SERVER`)
- `CONNECTION` or `COMMENT` strings
- Engine-defined options (such as `REMOTE_SERVER`, `REMOTE_DATABASE`, and `REMOTE_TABLE`)

Connection information for Spider tables can also be specified for different levels, where partition-level info overrides table-level info.

Spider has a predefined precedence order when using multiple methods.

Applies to both CS and ES

The following behavior applies to both CS and ES:

- Connection specification methods: (foreign server options, `COMMENT` string, `CONNECTION` string, and engine-defined options).
- Precedence rules (engine-defined options > `COMMENT` string > `CONNECTION` string > foreign server options; partition-level overrides table-level)
- Version-based behavior, such as the deprecation of `COMMENT` / `CONNECTION` in 11.4.1 and the introduction of engine-defined options `REMOTE_SERVER`, `REMOTE_DATABASE`, and `REMOTE_TABLE` in 10.8.1 and further options in 11.3.1

ES Additional Behavior (ODBC)

ODBC connection strings are used by Spider in Enterprise Server to create connections via ODBC drivers.

The source of the ODBC connection string could be:

- `COMMENT` clause
- `CONNECTION` clause
- engine-defined options
- Foreign server options

These values are combined and converted into a format that is compatible with ODBC.

Connection Methods

CREATE SERVER with COMMENT or CONNECTION

Since earlier versions, connection information can be specified by creating a foreign server, then referencing it in the Spider table definition using `COMMENT=` or `CONNECTION=`.

```
CREATE SERVER s
  FOREIGN DATA WRAPPER mysql
  OPTIONS (HOST 'remote_host', PORT 3306, DATABASE 'db_name',
           USER 'user', PASSWORD 'secret');

CREATE TABLE t (id INT) ENGINE=SPIDER COMMENT='srv "s"';
```

The same server can also be referenced at the partition level.

```
CREATE TABLE t (id INT) ENGINE=SPIDER COMMENT='srv "s"'
PARTITION BY RANGE (a) (
    PARTITION p1 VALUES LESS THAN (3),
    PARTITION p2 VALUES LESS THAN MAXVALUE COMMENT='srv "s_2_2"'
```

Engine-defined options: `REMOTE_SERVER`, `REMOTE_DATABASE`, `REMOTE_TABLE` (Since 10.8.1)

Since MariaDB 10.8.1 (MDEV-27106), engine-defined table options can be used to directly specify connection details:

```
CREATE TABLE t (id INT) ENGINE=SPIDER
  REMOTE_SERVER='s'
  REMOTE_DATABASE='db_name'
  REMOTE_TABLE='remote_t';
```

These engine-defined options override `COMMENT=` when both are present. These options are more structured and are the recommended method for new configurations.

Extended Engine-defined options (Since 11.3.1)

Since MariaDB 11.3.1 (MDEV-28856), more engine-defined options are available. `COMMENT=` and `CONNECTION=` are ignored if there are any engine-defined options and warnings may be generated. Set `spider_suppress_comment_ignored_warning = 1` to suppress the warning that is generated when they are ignored.

COMMENT and CONNECTION Deprecated (Since 11.4.1)

Since MariaDB 11.4.1 (MDEV-28861), the Spider connection information specified by `COMMENT=` and `CONNECTION=` has been deprecated. Instead, use the engine-defined parameters.

ODBC Connection String via CONNECTION (Since 11.4.3-1 ES)

In the following instances, `CONNECTION=` is used directly as the ODBC connection string since MariaDB Enterprise Server 11.4.3-1(MENT-2070):

- `COMMENT=` / `CONNECTION=` are explicitly ignored (i.e., engine-defined options are present, per MDEV-28856)
- The value of `CONNECTION=` follows the ODBC connection string format (`param1=val1;param2=val2;...`)

In previous ES versions, the chosen `COMMENT=`, `CONNECTION=`, and `OPTIONS` fields were used to create the ODBC connection string. See MENT-2076 comment for a complete list of fields used.

CREATE SERVER with FOREIGN DATA WRAPPER ODBC

Foreign server options are directly translated to ODBC connection key-value pairs when using `CREATE SERVER` with `FOREIGN DATA WRAPPER ODBC` (MENT-796). The above-described precedence order is maintained.

```
CREATE SERVER s
  FOREIGN DATA WRAPPER odbc
  OPTIONS (DSN 'MyDSN', UID 'user', PWD 'secret');
```

Supported FOREIGN DATA WRAPPER Values

`CREATE SERVER ... FOREIGN DATA WRAPPER` supports the following wrapper types:

Wrapper Value	Description	Availability
`mysql`	Connects to a remote MySQL server natively	CS + ES
`mariadb`	Connects to a remote MariaDB server natively	CS + ES
`odbc`	Connects to a remote server via ODBC	ES only

Note: The `odbc` wrapper value is only available in MariaDB Enterprise Server (ES). It is not supported in Community Server (CS).

Default Behavior When Connection Information is not Specified

If connection information is not explicitly specified using engine-defined options, Spider can depend on the configuration provided by other methods, such as `COMMENT`, `CONNECTION`, or `CREATE SERVER` statements.

The MariaDB version and configuration method determine the exact resolution behavior. The default behavior when no connection information is provided is currently being reviewed. To ensure predictable behavior, it is advised to explicitly specify connection details using engine-defined options (such as `REMOTE_SERVER`, `REMOTE_DATABASE`, and `REMOTE_TABLE`).

See Also

- Spider Table Parameters
- Spider System Variables
- CREATE SERVER Statement
- Spider Storage Engine

This page is licensed: CC BY-SA / Gnu FDL

Spider System Variables

The following variables are available when the Spider storage engine has been installed.

See Server System Variables for a complete list of system variables and instructions on setting them.

See also the Full list of MariaDB options, system and status variables.

> **Warning:**
>
> As of MariaDB 13.0, the following Spider table parameters can no longer be set through a table's `COMMENT` or `CONNECTION` string (in either their short or full form): `buffer_size` (`bfz`), `bka_table_name_type` (`btt`), `crd_mode` (`cmd`), `crd_type` (`ctp`), `crd_weight` (`cwg`), `init_sql_alloc_size` (`isa`), `internal_limit` (`ilm`), `internal_offset` (`ios`), `sts_mode` (`smd`), `semi_table_lock` (`stl`), and `semi_table_lock_connection` (`stc`). They were deprecated in earlier releases and have no corresponding table option; set the matching system variable below (for example, `spider_crd_mode`) instead.

MariaDB starting with 10.5.22

Starting from MariaDB 10.5.22, MariaDB 10.6.15, MariaDB 10.9.8, MariaDB 10.10.6, MariaDB 10.11.5, MariaDB 11.0.3, all spider system variables with the value -1 for deferring to table parameter values follow the correct overriding mechanism: table parameter (if set) overrides system variables (if not -1) overrides actual variable default. As a side effect, all such system variables in all versions have the same default value as the table param default value.
Before this change, a non-minus-one system variable value would override the table parameter value. That is, if both the system variable value and the table parameter value were set to be non-minus-one, the system variable value would prevail. For MariaDB 10.7+ where the system variable default values were the same as table param default instead of -1, this means that if the system variable were not set, but a table param is set to a non-default value, the default would override the non-default value.

`spider_auto_increment_mode`

- Description: The auto increment mode.
  - `-1` Falls back to the default value, if the table parameter is not set.

    - `0` Normal Mode. Uses a counter that Spider gets from the remote backend server with an exclusive lock for the auto-increment value. This mode is slow. Use Quick Mode (`2`), if you use Spider tables with the table partitioning feature and the auto-increment column is the first column of the index.
    - `1` Quick Mode. Uses an internal Spider counter for the auto-increment value. This mode is fast, but it is possible for duplicates to occur when updating the same table from multiple Spider proxies.
    - `2` Set Zero Mode. The auto-increment value is given by the remote backend. Sets the column to `0` , even if you set the value to the auto-increment column in your statement. If you use the table with the table partitioning feature, it sets to zero after choosing an inserted partition.
    - `3` When the auto-increment column is set to `NULL` , the value is given by the remote backend server. If you set the auto-increment column to `0` ,the value is given by the local server. Set spider_reset_auto_increment to `2` or `3` if you want to use an auto-increment column on the remote server.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `3`
- DSN Parameter Name: `aim`

`spider_bgs_first_read`

- Description: Number of first read records to use when performing a concurrent background search. To start a range scan on the remote backend, the storage engine first needs to send the first record. Fetching a second record in the same query can save a network round trip stopping the plan if the backend has a single record. The first and second reads are used to warm up for background search. When not using spider_split_read and spider_semi_split_read, the third read fetches the remaining data source in a single fetch.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` Records are usually retrieved.
    - `1` and greater: Number of records.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `2`
- Default Table Value: `2`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `bfr`

`spider_bgs_mode`

- Description: Background search mode. This enables the use of a thread per data server connection if the query is not shard-based and must be distributed across shards. The partitioning plugin scans partitions one after the other to optimize memory usage. Because the shards are external, reading all shards can be performed in parallel when the plan prunes multiple partitions.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` Disables background search.
    - `1` Uses background search when searching without locks
    - `2` Uses background search when searching without locks or with shared locks.
    - `3` Uses background search regardless of locks.

- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `3`
- DSN Parameter Name: `bmd`

`spider_bgs_second_read`

- Description: Number of second read records on the backend server when using background search. When the first records are found from spider_bgs_first_read, the engine continues scanning a range adding a `LIMIT` of spider_bgs_first_read and spider_bgs_second_read.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Records are usually retrieved.
  - `1` and greater: Number of records.
- Scope: Global, Session
- Dynamic: Yes
- Default Session Value: `100`
- Default Table Value: `100`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `bsr`

`spider_bka_engine`

- Description: Storage engine used with temporary tables when the spider_bka_mode system variable is set to `1`. Defaults to the value of the table parameter, which is MEMORY by default.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Session Value: `""`
- Default Table Value: `Memory`
- DSN Parameter Name: `bke`

`spider_bka_mode`

- Description: Internal action to perform when multi-split reads are disabled. If the spider_multi_split_read system variable is set to `0`, Spider uses this variable to determine how to handle statements when the optimizer resolves range retrieval to multiple conditions.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Uses “union all”.
  - `1` Uses a temporary table, if it is judged acceptable.
  - `2` Uses a temporary table, if it is judged acceptable and avoids replication delay.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Default Table Value: `1`

- Range: `-1` to `2`
- DSN Parameter Name: `bkm`

`spider_bka_table_name_type`

- Description: The type of temporary table name for bka.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Range: `-1` to `1`

`spider_block_size`

- Description: Size of memory block used in MariaDB. Can usually be left unchanged.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `16384`
- Range: `0` to `4294967295`
- DSN Parameter Name: `bsz`

`spider_buffer_size`

- Description: Buffer size. `-1`, the default, will use the table parameter.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `16000`
- Default Table Value: `16000`
- Range: `-1` to `2147483647`
- Introduced: MariaDB 10.5.4

`spider_bulk_size`

- Description: Size in bytes of the buffer when multiple grouping multiple `INSERT` statements in a batch, (that is, bulk inserts).
  - `-1` The table parameter is adopted.
  - `0` or greater: Size of the buffer.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `16000`
- Default Table Value: `16000`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `bsz`

`spider_bulk_update_mode`

- Description: Bulk update and delete mode. Note: If you use a non-default value for the spider_bgs_mode or spider_split_read system variables, Spider sets this variable to `2`.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Sends `UPDATE` and `DELETE` statements one by one.
  - `1` Collects multiple `UPDATE` and `DELETE` statements, then sends the collected statements one by one.
  - `2` Collects multiple `UPDATE` and `DELETE` statements and sends them together.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `2`
- DSN Parameter Name: `bum`

`spider_bulk_update_size`

- Description: Size in bytes for `UPDATE` and `DELETE` queries when generating bulk updates.
  - `-1` The table parameter is adopted.
  - `0` or greater: Size of buffer.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `16000`
- Default Table Value: `16000`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `bus`

`spider_casual_read`

- Description: Casual Reads enables all isolation levels, (such as repeatable reads) to work with transactions on multiple backends. With auto-commit queries, you can relax read consistency and run on multiple connections to the backends. This enables parallel queries across partitions, even if multiple shards are stored on the same physical server. Deprecated in MariaDB 11.5 due to the complexity of the code for little benefit.
  - `-1` Use table parameter.
  - `0` Use casual read.
  - `1` Choose connection channel automatically.
  - `2` to `63` Number of connection channels.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `63`
- DSN Parameter Name: `##`
- Deprecated: MariaDB 11.5

`spider_conn_recycle_mode`

- Description: Connection recycle mode.
  - `0` Disconnect.
  - `1` Recycle by all sessions.
  - `2` Recycle in the same session.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Range: `0` to `2`
- Default Session Value: `0`

`spider_conn_recycle_strict`

- Description: Whether to force the creation of new connections.
  - `1` Don't force.
  - `0` Force new connection
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Range: `0` to `1`

`spider_conn_wait_timeout`

- Description: Max waiting time in seconds for Spider to get a remote connection.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `10`
- Range: `0` to `1000`
- Introduced: MariaDB 10.3.3

`spider_connect_error_interval`

- Description: Return same error code until interval passes if connection is failed
- Scope: Global,
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `4294967295`

`spider_connect_mutex`

- Description: Whether to serialize remote servers connections (use mutex at connecting). Use this parameter if you get an error or slowdown due to too many connection processes.
  - `0` Not serialized.
  - `1` : Serialized.
- Scope: Global, Session

- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `0`

`spider_connect_retry_count`

- Description: Number of times to retry connections that fail due to too many connection processes.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `2` (>= MariaDB 11.8), `1000` (<= MariaDB 11.7)
- Range: `0` to `2147483647`

`spider_connect_retry_interval`

- Description: Interval in microseconds for connection failure due to too many connection processes.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1000`
- Range: `-1` to `9223372036854775807`

`spider_connect_timeout`

- Description: Timeout in seconds to declare remote backend unresponsive when opening a connection. Change for high-latency networks.
  - `-1` The table parameter is adopted.
  - `0` Less than 1.
  - `1` and greater: Number of seconds.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `6`
- Default Table Value: `0`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `cto`

`spider_crd_bg_mode`

- Description: Indexes cardinality statistics in the background. Disable when the spider_crd_mode system variable is set to `3` or when the spider_crd_interval variable is set to `0`.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Disables background confirmation.
  - `2` Enables background confirmation.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `2`
- Default Table Value: `1`

- Range: `-1` to `2`
- DSN Parameter Name: `cbm`

`spider_crd_interval`

- Description: Time interval in seconds of index cardinality statistics. Set to `0` to always get the latest information from remote servers.
  - `-1` The table parameter is adopted.
  - `1` or more: Interval in seconds table state confirmation.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `51`
- Default Table Value: `51`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `civ`

`spider_crd_mode`

- Description: Mode for index cardinality statistics. By default, uses `SHOW` at the table-level.
  - `-1,0` Uses the table parameter.
  - `1` Uses the `SHOW` command.
  - `2` Uses the Information Schema.
  - `3` Uses the `EXPLAIN` command.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Default Table Value: `1`
- Range: `-1` to `3`
- DSN Parameter Name: `cmd` (`COMMENT`/`CONNECTION`-string parameter removed in MariaDB 13.0)
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_crd_sync`

- Description: Synchronize index cardinality statistics in partitioned tables.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Disables synchronization.
  - `1` Uses table state synchronization when opening a table, but afterwards performs no synchronization.
  - `2` Enables synchronization.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `2`
- DSN Parameter Name: `csy`

`spider_crd_type`

- Description: Type of cardinality calculation. Only effective when the spider_crd_mode system variable is set to use `SHOW` (`1`) or to use the Information Schema (`2`).
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Uses the value of the spider_crd_weight system variable, as a fixed value.
  - `1` Uses the value of the spider_crd_weight system variable, as an addition value.
  - `2` Uses the value of the spider_crd_weight system variable, as a multiplication value.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `2`
- Default Table Value: `2`
- Range: `-1` to `2`
- DSN Parameter Name: `ctp` (`COMMENT` / `CONNECTION` -string parameter removed in MariaDB 13.0)
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_crd_weight`

- Description: Weight coefficient used to calculate effectiveness of index from the cardinality of column. For more information, see the description for the spider_crd_type system variable.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` or greater: Weight.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `2`
- Default Table Value: `2`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `cwg` (`COMMENT` / `CONNECTION` -string parameter removed in MariaDB 13.0)
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_delete_all_rows_type`

- Description: The type of delete_all_rows.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Range: `-1` to `1`

`spider_direct_dup_insert`

- Description: Manages duplicate key check for REPLACE, INSERT IGNORE and LOAD DATA LOCAL INFILE to remote servers. This can save on network roundtrips if the key always maps to a single partition. For bulk operations, records are checked for duplicate key errors one by one on the remote server, unless you set it to avoid duplicate checks on local servers (`1`).
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Performs duplicate checks on the local server.

    - `1` Avoids duplicate checks on the local server.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `1`
- DSN Parameter Name: `ddi`

`spider_direct_order_limit`

- Description: Pushes `ORDER BY` and `LIMIT` operations to the remote server.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` Uses local execution.
    - `1` Uses push down execution.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `2`
- Default Table Value: `9223372036854775807`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `dol`

`spider_disable_group_by_handler`

- Description: Whether to disable the spider group by handler, which if created takes over the query execution after query optimization is done.
    - `OFF` Does not disable the spider group by handler.
    - `ON` Disables the spider group by handler.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.10.7

`spider_dry_access`

- Description: Simulates an empty result-set. No queries are sent to the backend. Use for performance tuning.
    - `0` Normal access.
    - `1` All access from Spider to data node is suppressed.
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`spider_error_read_mode`

- Description: Sends an empty result-set when reading a backend server raises an error. Useful with applications that don't implement transaction replays.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Returns an error.
  - `1` Returns an empty result.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `1`
- DSN Parameter Name: `erm`

`spider_error_write_mode`

- Description: Sends an empty result-set when writing to a backend server raises an error. Useful with applications that don't implement transaction replays.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Returns an error.
  - `1` Returns an empty result-set on error.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `1`
- DSN Parameter Name: `ewm`

`spider_first_read`

- Description: Number of first read records to start a range scan on the backend server. Spider needs to send the first record. Fetching the second record saves network round-trips, stopping the plan if the backend has a single record. First read and second read are used to warm up for background searches, third reads without using the spider_split_read and spider_semi_split_read system variables fetches the remaining data source in a single last fetch.
  - `-1` Use the table parameter.
  - `0` Usually retrieves records.
  - `1` and greater: Sets the number of first read records.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `2`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `frd`

`spider_force_commit`

- Description: Behavior when error occurs on `XA PREPARE`, `XA COMMIT`, and `XA ROLLBACK` statements.
  - `0` Returns the error.
  - `1` Returns the error when the `xid` doesn't exist, otherwise it continues processing the XA transaction.
  - `2` Continues processing the XA transaction, disregarding all errors.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Range: `0` to `2`

`spider_general_log`

- Description: Whether Spider logs all commands to the General Log. Spider logs error codes according to the spider_log_result_errors system variable.
  - `OFF` Logs no commands.
  - `ON` Logs commands to the General Log.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `OFF`

`spider_ignore_comments`

- Description: Whether to unconditionally ignore COMMENT and CONNECTION strings without checking whether table options are specified.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 11.3.0

`spider_index_hint_pushdown`

- Description: Whether to use pushdown index hints, like `force_index`.
  - `0` Do not use pushdown hints
  - `1` Use pushdown hints
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `OFF`
- Introduced: MariaDB 10.3.3

`spider_init_sql_alloc_size`

- Description: Initial size of the local SQL buffer.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` or greater: Size of the buffer.

- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1024`
- Default Table Value: `1024`
- DSN Parameter Name: `isa` (`COMMENT` / `CONNECTION` -string parameter removed in MariaDB 13.0)
- Range: `-1` to `2147483647`
- Deprecated: MariaDB 10.7.5, MariaDB 10.8.4, MariaDB 10.9.2

`spider_internal_limit`

- Description: Limits the number of records when acquired from a remote server.
  - `-1` The table parameter is adopted.
  - `0` or greater: Records limit.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `9223372036854775807`
- Default Table Value: `9223372036854775807`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `ilm` (`COMMENT` / `CONNECTION` -string parameter removed in MariaDB 13.0)
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_internal_offset`

- Description: Skip records when acquired from the remote server.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` or more : Number of records to skip.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `ios` (`COMMENT` / `CONNECTION` -string parameter removed in MariaDB 13.0)
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_internal_optimize`

- Description: Whether to perform push down operations for OPTIMIZE TABLE statements.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Transmitted.
  - `1` Not transmitted.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`

- Default Table Value: `0`
- Range: `-1` to `1`
- DSN Parameter Name: `iom`

`spider_internal_optimize_local`

- Description: Whether to transmit to remote servers when OPTIMIZE TABLE statements are executed on the local server.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Not transmitted.
  - `1` Transmitted.
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `1`
- DSN Parameter Name: `iol`

`spider_internal_sql_log_off`

- Description: Whether to log SQL statements sent to the remote server in the General Query Log.
  - Explicitly setting this system variable to either `ON` or `OFF` causes the Spider node to send a `SET sql_log_off` statement to each of the data nodes using the `SUPER` privilege.
  - `-1` Don't know or does not matter; don't send 'SET SQL_LOG_OFF' statement
  - `0` Send 'SET SQL_LOG_OFF 0' statement to data nodes (logs SQL statements to the remote server)
  - `1` Send 'SET SQL_LOG_OFF 1' statement to data nodes (doesn't log SQL statements to the remote server)
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric` (previously `boolean`)
- Range: `-1` to `1`
- Default Session Value: `-1` (previously `ON`)

`spider_internal_unlock`

- Description: Whether to transmit unlock tables to the connection of the table used with `SELECT` statements.
  - `0` Not transmitted.
  - `1` Transmitted.
- Data Type: `boolean`
- Default Session Value: `0`

`spider_internal_xa`

- Description: Whether to implement XA at the server- or storage engine-level. When using the server-level, set different values for the server_id system variable on all server instances to generate different `xid` values.
  - `OFF` Uses the storage engine protocol.
  - `ON` Uses the server protocol.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `OFF`

`spider_internal_xa_id_type`

- Description: The type of internal_xa id.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Range: `-1` to `1`

`spider_internal_xa_snapshot`

- Description: Limitation for reading consistent data on all backend servers when using MariaDB's internal XA implementation and `START TRANSACTION WITH CONSISTENT SNAPSHOT`.
  - `0` Raise error when using a Spider table.
  - `1` Raise error when issued a `START TRANSACTION` statement.
  - `2` Takes a consistent snapshot on each backend, but loses global consistency.
  - `3` Starts transactions with XA, but removes `CONSISTENT SNAPSHOT`.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Range: `0` to `3`
- Default Session Value: `0`

`spider_load_crd_at_startup`

- Description: Whether to load CRD from the system table at startup.
  - `-1` Use table parameter
  - `0` Do not load
  - `1` Load
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Range: `-1` to `1`
- Introduced: MariaDB 10.3.3
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_load_sts_at_startup`

- Description: Whether to load STS from the system table at startup.
  - `-1` Use table parameter
  - `0` Do not load
  - `1` Load
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `1`
- Range: `-1` to `1`

- Introduced: MariaDB 10.3.3
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_local_lock_table`

- Description: Whether to push LOCK TABLES statements down to the remote server.
  - `0` Transmitted.
  - `1` Not transmitted.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`

`spider_lock_exchange`

- Description:
  Whether to convert SELECT… LOCK IN SHARE MODE and SELECT… FOR UPDATE statements into a LOCK TABLE statement.
  - `0` Not converted.
  - `1` Converted.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`

`spider_log_result_error_with_sql`

- Description: How to log SQL statements with result errors.
  - `0` No log
  - `1` Log error
  - `2` Log warning summary
  - `3` Log warning
  - `4` Log info (Added in MariaDB 10.5.4)
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4`

`spider_log_result_errors`

- Description: Log results from data nodes to the Spider node in the Error Log. Performs no logging by default.
  - `0` : Logs no errors from data nodes.
  - `1` : Logs errors from data nodes.
  - `2` : Logs errors from data nodes, as well as warning summaries.
  - `3` : Logs errors from data nodes, as well as warning summaries and details.
  - `4` : Logs errors from data nodes, as well as warning summaries and details, and result summaries.
- Scope: Global

- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4`

`spider_low_mem_read`

- Description: Whether to use low memory mode when executing queries issued internally to remote servers that return result-sets.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Doesn't use low memory mode.
  - `1` Uses low memory mode.
- Data Type: `numeric`
- Default Session Value: `1`
- Default Table Value: `1`
- Range: `-1` to `1`

`spider_max_connections`

- Description: Maximum number of connections from Spider to a remote MariaDB servers. Defaults to `0`, which is no limit.
- Command-line: `--spider-max-connections`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Range: `0` to `99999`
- Introduced: MariaDB 10.3.3

`spider_max_order`

- Description: Maximum number of columns for `ORDER BY` operations.
  - `-1` The table parameter is adopted.
  - `0` and greater: Maximum number of columns.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `32767`
- Default Table Value: `32767`
- Range: `-1` to `32767`
- DSN Parameter Name: `mod`

`spider_multi_split_read`

- Description:
  Whether to divide a statement into multiple SQL statements sent to the remote backend server when the optimizer resolves range retrievals to multiple conditions.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Doesn't divide statements.

    - `1` Divides statements.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `100`
- Default Table Value: `100`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `msr`

`spider_net_read_timeout`

- Description: TCP timeout in seconds to declare remote backend servers unresponsive when reading from a connection. Change for high latency networks.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` Less than 1 second timeout.
    - `1` and greater: Timeout in seconds.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `600`
- Default Table Value: `600`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `nrt`

`spider_net_write_timeout`

- Description: TCP timeout in seconds to declare remote backend servers unresponsive when writing to a connection. Change for high latency networks.
    - `-1` The table parameter is adopted.
    - `0` Less than 1 second timeout.
    - `1` and more: Timeout in seconds.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `600`
- Default Table Value: `600`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `nwt`

`spider_ping_interval_at_trx_start`

- Description: Resets the connection with keepalive timeout in seconds by sending a ping.
    - `0` At every transaction.
    - `1` and greater: Number of seconds.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`

- Default Value: `3600`
- Range: `0` to `2147483647`

`spider_quick_mode`

- Description: Sets the backend query buffering to cache on the remote backend server or in the local buffer.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Local buffering, it acquires records collectively with `store_result`.
  - `1` Remote buffering, it acquires records one by one. Interrupts don't wait and recovery on context switch back.
  - `2` Remote buffering, it acquires records one by one. Interrupts wait to the end of the acquisition.
  - `3` Local buffering, uses a temporary table on disk when the result-set is greater than the value of the spider_quick_page_size system variable.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `3`
- Default Table Value: `3`
- Range: `-1` to `3`
- DSN Parameter Name: `qmd`

`spider_quick_page_byte`

- Description: Memory limit by size in bytes in a page when acquired record by record.
  - `-1` The table parameter is used. When quick_mode is 1 or 2, Spider stores at least 1 record even if quick_page_byte is smaller than 1 record. When quick_mode is 3, quick_page_byte is used for judging using temporary tables. That is given priority when spider_quick_page_byte is set.
  - `0` or greater: Memory limit.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `10485760`
- Range: `-1` to `9223372036854775807`
- Introduced: MariaDB 10.4.3, MariaDB 10.3.13

`spider_quick_page_size`

- Description: Number of records in a page when acquired record by record.
  - `-1` The table parameter is adopted.
  - `0` or greater: Number of records.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1024` (>=MariaDB 10.7), `-1` (<= MariaDB 10.6)
- Default Table Value: `100`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `qps`

`spider_read_only_mode`

- Description: Whether to allow writes on Spider tables.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Allows writes to Spider tables.
  - `1` Makes tables read- only.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0` (>=MariaDB 10.7), `-1` (<= MariaDB 10.6)
- Default Table Value: `0`
- Range: `-1` to `1`
- DSN Parameter Name: `rom`

`spider_remote_access_charset`

- Description: Forces the specified session character set when connecting to the backend server. This can improve connection time performance.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Session Value: `null`

`spider_remote_autocommit`

- Description: Sets the auto-commit mode when connecting to backend servers. This can improve connection time performance.
  - `-1` Doesn't change the auto-commit mode.
  - `0` Sets the auto-commit mode to `0`.
  - `1` Sets the auto-commit mode to `1`.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `-1`
- Range: `-1` to `1`

`spider_remote_default_database`

- Description: Sets the local default database when connecting to backend servers. This can improve connection time performance.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Session Value: Empty string

`spider_remote_sql_log_off`

- Description: Sets the sql_log_off system variable to use when connecting to backend servers.
  - `-1` Doesn't set the value.
  - `0` Doesn't log Spider SQL statements to remote backend servers.

    - `1` Logs SQL statements on remote backend
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `-1`
- Range: `-1` to `1`

`spider_remote_time_zone`

- Description: Forces the specified time zone setting when connecting to backend servers. This can improve connection performance when you know the time zone.
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Session Value: `null`

`spider_remote_trx_isolation`

- Description: Sets the Transaction Isolation Level when connecting to the backend server.
    - `-1` Doesn't set the Isolation Level.
    - `0` Sets to the `READ UNCOMMITTED` level.
    - `1` Sets to the `READ COMMITTED` level.
    - `2` Sets to the `REPEATABLE READ` level.
    - `3` Sets to the `SERIALIZABLE` level.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `-1`
- Range: `-1` to `3`

`spider_remote_wait_timeout`

- Description: Wait timeout in seconds on remote server. `-1` means not set.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `2147483647`
- Introduced: MariaDB 10.4.5

`spider_reset_sql_alloc`

- Description: Resets the per connection SQL buffer after an SQL statement executes.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` Doesn't reset.
    - `1` Resets.
- Scope: Global, Session
- Dynamic: Yes

- Data Type: `numeric`
- Default Session Value: `1`
- Default Table Value: `1`
- Range: `-1` to `1`
- DSN Parameter Name: `rsa`

`spider_same_server_link`

- Description: Enables the linking of a table to the same local instance.
  - `0` Disables linking.
  - `1` Enables linking.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `OFF`

`spider_second_read`

- Description: Number of second read records on the backend server when the first records are found from the first read. Spider continues scanning a range, adding a `LIMIT` using the spider_first_read and spider_second_read variables.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Usually retrieves records.
  - `1` and greater: Number of records.
- Scope: Global, Session
- Dynamic: Yes
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `srd`

`spider_select_column_mode`

- Description: Mode for column retrieval from remote backend server.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Uses index columns when the `SELECT` statement can resolve with an index, otherwise uses all columns.
  - `1` Uses all columns judged necessary to resolve the query.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Default Table Value: `1`
- Range: `-1` to `1`
- DSN Parameter Name: `scm`

`spider_selupd_lock_mode`

- Description: Local lock mode on `INSERT SELECT`.
  - `-1` Falls back to the default value, if the table parameter is not set.

    - `0` Takes no locks.
    - `1` Takes shared locks.
    - `2` Takes exclusive locks.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Default Table Value: `1`
- Range: `-1` to `2`
- DSN Parameter Name: `slm#`

`spider_semi_split_read`

- Description:
  Whether to use chunk retrieval with offset and limit parameters on SQL statements sent to the remote backend server when using the spider_split_read system variable.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` Doesn't use chunk retrieval.
    - `1 or more` Uses chunk retrieval.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `2`
- Default Table Value: `2`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `ssr#`

`spider_semi_split_read_limit`

- Description: Sets the limit value for the spider_semi_split_read system variable.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` or more: The limit value.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `9223372036854775807`
- Default Table Value: `9223372036854775807`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `ssl#`

`spider_semi_table_lock`

- Description: Enables semi-table locking. This adds a LOCK TABLES statement to SQL executions sent to the remote backend server when using non-transactional storage engines to preserve consistency between roundtrips.
    - `0` Disables semi-table locking.
    - `1` Enables semi-table locking.

- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0` (>=MariaDB 10.8), `1` (<= MariaDB 10.7)
- Range: `0` to `1`
- DSN Parameter Name: `stl#`

`spider_semi_table_lock_connection`

- Description: Whether to use multiple connections with semi-table locking. To enable semi-table locking, use the spider_semi_table_lock system variable.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Uses the same connection.
  - `1` Uses different connections.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Default Table Value: `1`
- Range: `-1` to `1`
- DSN Parameter Name: `stc#`

`spider_semi_trx`

- Description: Enables semi-transactions. This controls transaction consistency when an SQL statement is split into multiple statements issued to the backend servers. You can preserve or relax consistency as need. Spider encapsulates auto-committed SQL statements within a transaction on the remote backend server. When using `READ COMMITTED` or `READ UNCOMMITTED` transaction isolation levels to force consistency, set the spider_semi_trx_isolation system variable to `2`.
  - `0` Disables semi-transaction consistency.
  - `1` Enables semi-transaction consistency.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `ON`

`spider_semi_trx_isolation`

- Description: Set consistency during range SQL execution when spider_sync_trx_isolation is 1
  - `-1` OFF
  - `0` READ UNCOMMITTED
  - `1` READ COMMITTED
  - `2` REPEATABLE READ
  - `3` SERIALIZABLE
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `-1`

- Range: `-1` to `3`

`spider_skip_default_condition`

- Description: Whether to compute condition push downs.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Computes condition push downs.
  - `1` Doesn't compute condition push downs.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `1`
- DSN Parameter Name: `sdc`

`spider_skip_parallel_search`

- Description: Whether to skip parallel search by specific conditions.
  - `-1` :use table parameter
  - `0` :not skip
  - `1` :skip parallel search if query is not SELECT statement
  - `2` :skip parallel search if query has SQL_NO_CACHE
  - `3` :1+2
- Command line: `--spider-skip-parallel-search=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Range: `-1` to `3`
- Introduced: MariaDB 10.3.3

`spider_slave_trx_isolation`

- Description: Transaction isolation level when Spider table is used by slave SQL thread.
  - `-1` off
  - `0` read uncommitted
  - `1` read committed
  - `2` repeatable read
  - `3` serializable
- Command line: `--spider-slave-trx-isolation=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `-1`
- Range: `-1` to `3`
- Introduced: MariaDB 10.4.3, MariaDB 10.3.13

`spider_split_read`

- Description: Number of records in chunk to retry the result when a range query is sent to remote backend servers.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` or more: Number of records.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `9223372036854775807`
- Default Table Value: `9223372036854775807`
- Range: `-1` to `9223372036854775807`
- DSN Parameter Name: `srd`

`spider_store_last_crd`

- Description: Whether to store last CRD result in the system table.
  - `-1` Use table parameter.
  - `0` Do not store last CRD result in the system table.
  - `1` Store last CRD result in the system table.
- Command line: `--spider-store-last-crd=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Range: `-1` to `1`
- Introduced: MariaDB 10.3.3
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_store_last_sts`

- Description: Whether to store last STS result in the system table.
  - `-1` Use table parameter.
  - `0` Do not store last STS result in the system table.
  - `1` Store last STS result in the system table.
- Command line: `--spider-store-last-sts=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Range: `-1` to `1`
- Introduced: MariaDB 10.3.3
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_strict_group_by`

- Description:
  Whether to use columns in select clause strictly for group by clause

    - `-1` Use the table parameter.
    - `0` Do not strictly use columns in select clause for group by clause
    - `1` Use columns in select clause strictly for group by clause
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Range: `-1` to `1`
- Introduced: MariaDB 10.5.4

`spider_sts_bg_mode`

- Description:
  Enables background confirmation for table statistics. When background confirmation is enabled, Spider uses one thread per partition to maintain table status. Disable when the spider_sts_interval system variable is set to `0`, which causes Spider to always retrieve the latest information as need. It is effective, when the spider_sts_interval system variable is set to `10`.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` Disables background confirmation.
    - `1` Enables background confirmation (create thread per table/partition).
    - `2` Enables background confirmation (use static threads). (from MariaDB 10.)
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `2`
- Range: `-1` to `2`
- DSN Parameter Name: `sbm`

`spider_sts_interval`

- Description: Time interval of table statistics from the remote backend servers.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` Retrieves the latest table statistics on request.
    - `1` or more: Interval in seconds for table state confirmation.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `10`
- Default Table Value: `10`
- Range: `-1` to `2147483647`
- DSN Parameter Name: `siv`

`spider_sts_mode`

- Description: Table statistics mode.
  Mode for table statistics. The SHOW command is used at the table level default.
    - `-1,0` Uses the table parameter.
    - `1` Uses the `SHOW` command.

    - `2` Uses the Information Schema.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `1`
- Default Table Value: `1`
- Range: `-1` to `2`
- DSN Parameter Name: `smd` (`COMMENT` / `CONNECTION` -string parameter removed in MariaDB 13.0)
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

`spider_sts_sync`

- Description: Synchronizes table statistics in partitioned tables.
    - `-1` Falls back to the default value, if the table parameter is not set.
    - `0` Doesn't synchronize table statistics in partitioned tables.
    - `1` Synchronizes table state when opening a table, doesn't synchronize after opening.
    - `2` Synchronizes table statistics.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Session Value: `0`
- Default Table Value: `0`
- Range: `-1` to `2`
- DSN Parameter Name: `ssy`

`spider_support_xa`

- Description: XA Protocol for mirroring and for multi-shard transactions.
    - `1` Enables XA Protocol for these Spider operations.
    - `0` Disables XA Protocol for these Spider operations.
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Table Value: `1`

`spider_suppress_comment_ignored_warning`

- Description: Whether to suppress warnings that table COMMENT or CONNECTION strings are ignored due to specified table options.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 11.3.0

`spider_sync_autocommit`

- Description: Whether to push down local auto-commits to remote backend servers.

    - `OFF` Pushes down local auto-commits.
    - `ON` Doesn't push down local auto-commits.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `ON`

`spider_sync_sql_mode`

- Description: Whether to sync sql_mode.
    - `OFF` No sync
    - `ON` Sync
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.4.7

`spider_sync_time_zone`

- Description: Whether to push the local time zone down to remote backend servers.
    - `OFF` Doesn't synchronize time zones.
    - `ON` Synchronize time zones.
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `OFF`
- Removed: MariaDB 10.3.9

`spider_sync_trx_isolation`

- Description: Pushes local transaction isolation levels down to remote backend servers.
    - `OFF` Doesn't push down local isolation levels.
    - `ON` Pushes down local isolation levels.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `ON`

`spider_table_crd_thread_count`

- Description: Static background thread count of table crd.
- Command line: `--spider-table-crd-thread-count=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value:

    - `1` (>= MariaDB 10.4.33, MariaDB 10.5.24, MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3)
    - `10` (<= MariaDB 10.4.32, MariaDB 10.5.23, MariaDB 10.6.16, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3, MariaDB 11.2.2)
- Range: `1` to `4294967295`
- Deprecated: MariaDB 11.7

`spider_table_init_error_interval`

- Description: Interval in seconds where the same error code is returned if table initialization fails. Use to protect against infinite loops in table links.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `4294967295`

`spider_table_sts_thread_count`

- Description: Static background thread count of table sts.
- Command line: `--spider-table-sts-thread-count=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
    - `1` (>= MariaDB 10.4.33, MariaDB 10.5.24, MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3)
    - `10` (<= MariaDB 10.4.32, MariaDB 10.5.23, MariaDB 10.6.16, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3, MariaDB 11.2.2)
- Range: `1` to `4294967295`
- Deprecated: MariaDB 11.7

`spider_udf_ct_bulk_insert_interval`

- Description: Interval in milliseconds between bulk inserts at copying. For use with the UDF spider_copy_tables, which copies table data linked to a Spider table from the source server to destination server using bulk insert. If this interval is 0, it may cause higher write traffic.
    - `-1` Uses the UDF parameter.
    - `0` and more: Time in milliseconds.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10`
- Default Table Value: `10`
- Range: `-1` to `2147483647`
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3
- Removed: MariaDB 10.10

`spider_udf_ct_bulk_insert_rows`

- Description: Number of rows to insert at a time when copying during bulk inserts.
  - `-1, 0`: Uses the table parameter.
  - `1` and more: Number of rows
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Default Table Value: `100`
- Range: `-1` to `9223372036854775807`
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3
- Removed: MariaDB 10.10

`spider_udf_ds_bulk_insert_rows`

- Description: Number of rows inserted at a time during bulk inserts when the result-set is stored in a temporary table on executing a UDF.
  - `-1, 0` Uses the UDF parameter.
  - `1` or more: Number of rows
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `3000`
- Default Table Value: `3000`
- Range: `-1` to `9223372036854775807`
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3
- Removed: MariaDB 10.10

`spider_udf_ds_table_loop_mode`

- Description: Whether to store the result-set in the same temporary table when the temporary table list count for UDF is less than the result-set count on UDF execution.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Drops records.
  - `1` Inserts the last table.
  - `2` Inserts the first table and loops again.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `-1` to `2`
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3
- Removed: MariaDB 10.10

`spider_udf_ds_use_real_table`

- Description: Whether to use real table for temporary table list.

    - `-1` Use UDF parameter.
    - `0` Do not use real table.
    - `1` Use real table.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `-1` to `1`
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3
- Removed: MariaDB 10.10

`spider_udf_table_lock_mutex_count`

- Description: Mutex count of table lock for Spider UDFs.
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `20`
- Range: `1` to `4294967295`
- Removed: MariaDB 10.10

`spider_udf_table_mon_mutex_count`

- Description: Mutex count of table mon for Spider UDFs.
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `20`
- Range: `1` to `4294967295`
- Removed: MariaDB 10.10

`spider_use_all_conns_snapshot`

- Description: Whether to pass `START TRANSACTION WITH SNAPSHOT` statements to all connections.
    - `OFF` Doesn't pass statement to all connections.
    - `ON` Passes statement to all connections.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Session Value: `OFF`

`spider_use_cond_other_than_pk_for_update`

- Description: Whether to use all conditions even if condition has a primary key.
    - `0` Don't use all conditions
    - `1` Use all conditions
- Scope: Global, Session
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `1`
- Introduced: MariaDB 10.3.13, MariaDB 10.4.3

`spider_use_consistent_snapshot`

- Description:
  Whether to push a local `START TRANSACTION WITH CONSISTENT` statement down to remote backend servers.
  - `OFF` Doesn't push the local statement down.
  - `ON` Pushes the local statement down.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`spider_use_default_database`

- Description: Whether to use the default database.
  - `OFF` Doesn't use the default database.
  - `ON` Uses the default database.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`spider_use_flash_logs`

- Description: Whether to push FLUSH LOGS statements down to remote backend servers.
  - `OFF` Doesn't push the statement down.
  - `ON` Pushes the statement down.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`spider_use_handler`

- Description: Converts HANDLER SQL statements.
  When the spider_sync_trx_isolation system variable is set to `0`, Spider disables HANDLER conversions to prevent use of the statement on the SERIALIZABLE isolation level.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Converts HANDLER statements into SELECT statements.
  - `1` Passes HANDLER to the remote backend server.
  - `2` Converts SQL statements to HANDLER statements.
  - `3` Converts SQL statements to HANDLER statements and HANDLER statements to SQL statements.
- Scope: Global, Session
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `0`
- Default Table Value: `0`
- Range: `-1` to `3`
- DSN Parameter Name: `uhd`
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3
- Removed: MariaDB 10.10

`spider_use_pushdown_udf`

- Description:
  When using a UDF function in a condition and the engine_condition_pushdown system variable is set to `1`, whether to execute the UDF function locally or push it down.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Doesn't transmit the UDF
  - `1` Transmits the UDF.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `-1`
- Default Table Value: `1`
- Range: `-1` to `1`
- DSN Parameter Name: `upu`

`spider_use_snapshot_with_flush_tables`

- Description:
  Whether to encapsulate FLUSH LOGS and UNLOCK TABLES statements when `START TRANSACTION WITH CONSISTENT` and `FLUSH TABLE WITH READ LOCK` statements are sent to the remote backend servers.
  - `0` : No encapsulation.
  - `1` : Encapsulates, only when the spider_use_all_conns_snapshot system variable i set to `1`.
  - `2` :
    Synchronizes the snapshot using a LOCK TABLES statement and [flush|FLUSH TABLES]] at the XA transaction level. This is only effective when the spider_use_all_cons_snapshot system variable is set to `1`.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `2`

`spider_use_table_charset`

- Description: Whether to use the local table character set for the remote backend server connections.
  - `-1` Falls back to the default value, if the table parameter is not set.
  - `0` Use `utf8`.
  - `1` Uses the table character set.
- Scope: Global, Session
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `1`
- Default Table Value: `1`
- Range: `-1` to `1`
- DSN Parameter Name: `utc`

`spider_version`

- Description: The current Spider version. Removed in MariaDB 10.9.2 when the Spider version number was matched with the server version.
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Removed: MariaDB 10.9.2

`spider_wait_timeout`

- Description: Wait timeout in seconds of setting to remote server. `-1` means not set.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `604800`
- Range: `-1` to `2147483647`
- Introduced: MariaDB 10.4.5

`spider_xa_register_mode`

- Description: Mode of XA transaction register into system table.
  - `0` Register all XA transactions
  - `1` Register only write XA transactions
- Command-line: `--spider-xa-register-mode=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `1`
- Introduced: MariaDB 10.3.3
- Deprecated: MariaDB 10.7.4, MariaDB 10.8.3

This page is licensed: CC BY-SA / Gnu FDL

Spider Table Parameters

When a table uses the Spider storage engine, the following Spider table parameters can be set in the `COMMENT` clause of the CREATE TABLE statement. Many Spider table parameters have corresponding system variables, so they can be set for all Spider tables on the node. For additional information, see the Spider System Variables page.

From MariaDB 11.3, many table parameters can be set using dedicated Spider table options, see the Table Option Name fields below. From MariaDB 11.4, using the `COMMENT` clause is deprecated, as well as table parameters that do not have corresponding table options.

`access_balances`

- Description: Connection load balancing integer weight.
- Default Table Value: `0`
- DSN Parameter Name: `abl`
- Deprecated: MariaDB 11.4.0

`active_link_count`

- Description: Number of active remote servers, for use in load balancing read connections
- Default Table Value: `all backends`
- DSN Parameter Name: `alc`
- Deprecated: MariaDB 11.4.0

`auto_increment_mode`

- Description: The table level value of spider_auto_increment_mode
- Table Option Name: `AUTO_INCREMENT_MODE`
- Table Option Introduced: MariaDB 11.3.0

`bgs_mode`

- Description: The table level value of spider_bgs_mode.
- Table Option Name: `BGS_MODE`
- Table Option Introduced: MariaDB 11.3.0

`bulk_size`

- Description: The table level value of spider_bulk_size.
- Table Option Name: `BULK_SIZE`
- Table Option Introduced: MariaDB 11.3.0

`bulk_update_size`

- Description: The table level value of spider_bulk_update_size.
- Table Option Name: `BULK_UPDATE_SIZE`
- Table Option Introduced: MariaDB 11.3.0

`casual_read`

- Description:
- Default Table Value:
- DSN Parameter Name:
- Introduced: Spider 3.2
- Deprecated: MariaDB 11.4.0

`connect_timeout`

- Description: The table level value of spider_connect_timeout.
- Table Option Name: `CONNECT_TIMEOUT`
- Table Option Introduced: MariaDB 11.3.0

`database`

- Description: Database name for reference table that exists on remote backend server.
- Default Table Value: `local table database`
- DSN Parameter Name: `database`
- Table Option Name: `REMOTE_DATABASE`
- Table Option Introduced: MariaDB 10.8.0

`default_file`

- Description: Configuration file used when connecting to remote servers. When the default_group table variable is set, this variable defaults to the values of the `--defaults-extra-file` or `--defaults-file` options. When the default_group table variable is not set, it defaults to `none`.
- Default Table Value: `none`
- DSN Parameter Name: `dff`
- Table Option Name: `DEFAULT_FILE`
- Table Option Introduced: MariaDB 11.3.0

`default_group`

- Description: Group name in configuration file used when connecting to remote servers.
- Default Table Value: `none`
- DSN Parameter Name: `dfg`
- Table Option Name: `DEFAULT_GROUP`
- Table Option Introduced: MariaDB 11.3.0

`delete_all_rows_type`

- Description: The table level value of spider_delete_all_rows_type.
- Introduced: Spider 3.2
- Table Option Name: `DELETE_ALL_ROWS_TYPE`
- Table Option Introduced: MariaDB 11.3.0

`force_bulk_delete`

- Description:
- Introduced: MariaDB 10.0.5
- Table Option Name: `FORCE_BULK_DELETE`
- Table Option Introduced: MariaDB 11.3.0

`force_bulk_update`

- Description:
- Introduced: MariaDB 10.0.5
- Table Option Name: `FORCE_BULK_UPDATE`
- Table Option Introduced: MariaDB 11.3.0

`host`

- Description: Host name of remote server.
- Default Table Value: `localhost`
- DSN Parameter Name: `host`

- Table Option Name: `REMOTE_HOST`
- Table Option Introduced: MariaDB 11.3.0

`idx000`

- Description: When using an index on Spider tables for searching, Spider uses this hint to search the remote table. The remote table index is related to the Spider table index by this hint. The number represented by `000` is the index ID, which is the number of the index shown by the SHOW CREATE TABLE statement. `000` is the Primary Key. For instance, `idx000 "force index(PRIMARY)"` (in abbreviated format `idx000 "f PRIMARY"`).
  - `f` force index
  - `u` use index
  - `ig` ignore index
- Default Table Value: `none`
- Table Option Name: `IDX`
- Table Option Introduced: MariaDB 11.3.0

`link_status`

- Description: Change status of the remote backend server link.
  - `0` Doesn't change status.
  - `1` Changes status to `OK`.
  - `2` Changes status to `RECOVERY`.
  - `3` Changes status to no more in group communication.
- Default Table Value: `0`
- DSN Parameter Name: `lst`
- Deprecated: MariaDB 11.4.0

`monitoring_bg_interval`

- Description: Interval of background monitoring in microseconds.
- Default Table Value: `10000000`
- DSN Parameter Name: `mbi`
- Deprecated: MariaDB 11.4.0

`monitoring_bg_kind`

- Description: Kind of background monitoring to use.
  - `0` Disables background monitoring.
  - `1` Monitors connection state.
  - `2` Monitors state of table without `WHERE` clause.
  - `3` Monitors state of table with `WHERE` clause (currently unsupported).
- Default Table Value: `0`
- DSN Parameter Name: `mbk`
- Deprecated: MariaDB 11.4.0

`monitoring_kind`

- Description: Kind of monitoring.
  - `0` Disables monitoring
  - `1` Monitors connection state.

    - `2` Monitors state of table without `WHERE` clause.
    - `3` Monitors state of table with `WHERE` clause (currently unsupported).
- Default Table Value: `0`
- DSN Parameter Name: `mkd`
- Deprecated: MariaDB 11.4.0

`monitoring_limit`

- Description: Limits the number of records in the monitoring table. This is only effective when Spider monitors the state of a table, which occurs when the monitoring_kind table variable is set to a value greater than `1`.
- Default Table Value: `1`
- Range: `0` upwards
- DSN Parameter Name: `mlt`
- Deprecated: MariaDB 11.4.0

`monitoring_server_id`

- Description: Preferred monitoring `@@server_id` for each backend failure. You can use this to geo-localize backend servers and set the first Spider monitoring node to contact for failover. In the event that this monitor fails, other monitoring nodes are contacted. For multiple copy backends, you can set a lazy configuration with a single MSI instead of one per backend.
- Default Table Value: `server_id`
- DSN Parameter Name: `msi`
- Deprecated: MariaDB 11.4.0

`multi_split_read`

- Description: The table level value of spider_multi_split_read.
- Table Option Name: `MULTI_SPLIT_READ`
- Table Option Introduced: MariaDB 11.3.0

`net_read_timeout`

- Description: The table level value of spider_net_read_timeout.
- Table Option Name: `NET_READ_TIMEOUT`
- Table Option Introduced: MariaDB 11.3.0

`net_write_timeout`

- Description: The table level value of spider_net_write_timeout.
- Table Option Name: `NET_WRITE_TIMEOUT`
- Table Option Introduced: MariaDB 11.3.0

`password`

- Description: Remote server password.
- Default Table Value: `none`
- DSN Parameter Name: `password`
- Table Option Name: `REMOTE_PASSWORD`
- Table Option Introduced: MariaDB 11.3.0

`port`

- Description: Remote server port.
- Default Table Value: `3306`
- DSN Parameter Name: `port`
- Table Option Name: `REMOTE_PORT`
- Table Option Introduced: MariaDB 11.3.0

`priority`

- Description: Priority. Used to define the order of execution. For instance, Spider uses priority when deciding the order in which to lock tables on a remote server.
- Default Table Value: `1000000`
- DSN Parameter Name: `prt`
- Table Option Name: `PRIORITY`
- Table Option Introduced: MariaDB 11.3.0

`query_cache`

- Description: Uses the option for the Query Cache when issuing SELECT statements to the remote server.
  - `0` No option used.
  - `1` Uses the SQL_CACHE option.
  - `2` Uses the SQL_NO_CACHE option.
- Default Table Value: `0`
- DSN Parameter Name: `qch`
- Table Option Name: `QUERY_CACHE`
- Table Option Introduced: MariaDB 11.3.0

`query_cache_sync`

- Description: A two-bit bitmap. Whether to pass the option for the Query Cache (if any) when issuing SELECT statements to the remote server.
  - `0` No option passed.
  - `1` Passes the SQL_CACHE option, if specified in the query to the spider table.
  - `2` Passes the SQL_NO_CACHE option, if specified in the query to the spider table.
  - `3` Passes both the SQL_CACHE option and the SQL_NO_CACHE option, if specified in the query to the spider table.
- Default Table Value: `3`
- Table Option Name: `QUERY_CACHE_SYNC`
- Table Option Introduced: MariaDB 11.3.0

`read_rate`

- Description: Rate used to calculate the amount of time Spider requires when executing index scans.
- Default Table Value: `0.0002`
- DSN Parameter Name: `rrt`
- Deprecated: MariaDB 11.4.0

`scan_rate`

- Description: Rate used to calculate the amount of time Spider requires when scanning tables.

- Default Table Value: `0.0001`
- DSN Parameter Name: `srt`
- Deprecated: MariaDB 11.4.0

`server`

- Description: Server name. Used when generating connection information with CREATE SERVER statements.
- Default Table Value: `none`
- DSN Parameter Name: `srv`
- Table Option Name: `REMOTE_SERVER`
- Table Option Introduced: MariaDB 10.8.0

`skip_parallel_search`

- Description: The table level value of spider_skip_parallel_search.
- Table Option Name: `SKIP_PARALLEL_SEARCH`
- Table Option Introduced: MariaDB 11.3.0

`socket`

- Description: Remote server socket.
- Default Table Value: `none`
- DSN Parameter Name: `socket`
- Table Option Name: `REMOTE_SOCKET`
- Table Option Introduced: MariaDB 11.3.0

`ssl_ca`

- Description: Path to the Certificate Authority file.
- Default Table Value: `none`
- DSN Parameter Name: `sca`
- Table Option Name: `SSL_CA`
- Table Option Introduced: MariaDB 11.3.0

`ssl_capath`

- Description: Path to directory containing trusted TLS CA certificates in PEM format.
- Default Table Value: `none`
- DSN Parameter Name: `scp`
- Table Option Name: `SSL_CAPATH`
- Table Option Introduced: MariaDB 11.3.0

`ssl_cert`

- Description: Path to the certificate file.
- Default Table Value: `none`
- DSN Parameter Name: `scr`
- Table Option Name: `SSL_CERT`
- Table Option Introduced: MariaDB 11.3.0

`ssl_cipher`

- Description: List of allowed ciphers to use with TLS encryption.
- Default Table Value: `none`
- DSN Parameter Name: `sch`
- Table Option Name: `SSL_CIPHER`
- Table Option Introduced: MariaDB 11.3.0

`ssl_key`

- Description: Path to the key file.
- Default Table Value: `none`
- DSN Parameter Name: `sky`
- Table Option Name: `SSL_KEY`
- Table Option Introduced: MariaDB 11.3.0

`ssl_verify_server_cert`

- Description: Enables verification of the server's Common Name value in the certificate against the host name used when connecting to the server.
  - `0` Disables verification.
  - `1` Enables verification.
- Default Table Value: `0`
- DSN Parameter Name: `svc`
- Table Option Name: `SSL_VSC`
- Table Option Introduced: MariaDB 11.3.0

`table`

- Description: Destination table name.
- Default Table Value: `Same table name`
- DSN Parameter Name: `tbl`
- Table Option Name: `REMOTE_TABLE`
- Table Option Introduced: MariaDB 10.8.0

`table_count_mode`

- Description: for setting table flags HA_STATS_RECORDS_IS_EXACT and HA_HAS_RECORDS.
- Default Table Value: `0`
- Table Option Name: `TABLE_COUNT_MODE`
- Table Option Introduced: MariaDB 11.3.0

`username`

- Description: user name for the data node.
- Default Table Value: `Same user name`
- Table Option Name: `REMOTE_USERNAME`
- Table Option Introduced: MariaDB 11.3.0

`use_pushdown_udf`

- Description: The table level value of spider_use_pushdown_udf.

- Table Option Name: `USE_PUSHDOWN_UDF`
- Table Option Introduced: MariaDB 11.3.0

`wrapper`

- Description: wrapper for the data node.
- Table Option Name: `WRAPPER`
- Table Option Introduced: MariaDB 11.3.0

Spider Table: Connection Parameters

The Spider table's connection to a remote server is managed via the following engine-defined settings. The `CREATE TABLE` statement specifies these parameters, which take precedence over `COMMENT=` and `CONNECTION=`. See Specifying Connection Information for Spider Tables for a detailed explanation of all connection methods and precedence rules.

Engine-defined Connection Options

Option	Description	Available since
`REMOTE_SERVER`	Name of the server object generated with `CREATE SERVER` that will be used for the remote connection.	10.8.1
`REMOTE_DATABASE`	Database name on the remote server. The value defined in the referenced server object is used if it is not provided.	10.8.1
`REMOTE_TABLE`	Name of the table on the remote server. The local table name is used if it is not provided.	10.8.1

> **Note:**
>
> Starting with version 11.3.1, `COMMENT=` and `CONNECTION=` are silently ignored whenever any engine-defined option is present.

> **Note:**
>
> To suppress the warning, `set spider_suppress_comment_ignored_warning = 1`. Since 11.4.1, the connection methods `COMMENT=` and `CONNECTION=` have been deprecated. Instead, make use of engine-defined options.

This page is licensed: CC BY-SA / Gnu FDL

Spider Functions

See SPIDER_BG_DIRECT_SQL.

This UDF allows you to execute SQL statements on remote data nodes in the background, enabling concurrent processing and non-blocking operations.

See SPIDER_COPY_TABLES.

Learn how to use this function to copy table data from one Spider link ID to another, useful for migrating data or rebalancing shards without stopping the service.

See SPIDER_DIRECT_SQL.

This function executes an SQL string directly on a specified remote backend server, allowing for maintenance tasks or queries that bypass local parsing.

See SPIDER_FLUSH_TABLE_MON_CACHE.

Use this UDF to refresh the cache used by Spider's monitoring threads, ensuring that the status of remote tables and connections is up to date.

This page is licensed: CC BY-SA / Gnu FDL

SPIDER_BG_DIRECT_SQL

Syntax

```
SPIDER_BG_DIRECT_SQL('sql', 'tmp_table_list', 'parameters')
```

Description

Executes the given SQL statement in the background on the remote server, as defined in the parameters listing. If the query returns a result-set, it sttores the results in the given temporary table. When the given SQL statement executes successfully, this function returns the number of called UDF's. It returns `0` when the given SQL statement fails.

This function is a UDF installed with the Spider storage engine.

Examples

```
SELECT SPIDER_BG_DIRECT_SQL('SELECT * FROM example_table',  '',
   'srv "node1", port "8607"') AS "Direct Query";
+--------------+
| Direct Query |
+--------------+
|            1 |
+--------------+
```

Parameters

`error_rw_mode`

- Description: Returns empty results on network error.
  - `0` : Return error on getting network error.
  - `1`: Return 0 records on getting network error.
- Default Table Value: `0`
- DSN Parameter Name: `erwm`

See also

- SPIDER_DIRECT_SQL

This page is licensed: CC BY-SA / Gnu FDL

SPIDER_COPY_TABLES

Syntax

```
SPIDER_COPY_TABLES(spider_table_name,
  source_link_id, destination_link_id_list [,parameters])
```

Description

A UDF installed with the Spider Storage Engine, this function copies table data from `source_link_id` to `destination_link_id_list`. The service does not need to be stopped in order to copy.

If the Spider table is partitioned, the name must be of the format `table_name#P#partition_name`. The partition name can be viewed in the `mysql.spider_tables` table, for example:

```
SELECT table_name FROM mysql.spider_tables;
+-------------+
| table_name  |
+-------------+
| spt_a#P#pt1 |
| spt_a#P#pt2 |
| spt_a#P#pt3 |
+-------------+
```

Returns `1` if the data was copied successfully, or `0` if copying the data failed.

This page is licensed: CC BY-SA / Gnu FDL

SPIDER_DIRECT_SQL

Syntax

```
SPIDER_DIRECT_SQL('sql', 'tmp_table_list', 'parameters')
```

Description

A UDF installed with the Spider Storage Engine, this function is used to execute the SQL string `sql` on the remote server, as defined in `parameters`. If any resultsets are returned, they are stored in the `tmp_table_list`.

The function returns `1` if the SQL executes successfully, or `0` if it fails.

Examples

```
SELECT SPIDER_DIRECT_SQL('SELECT * FROM s', '', 'srv "node1", port "8607"');
+---------------------------------------------------------------------------+
| SPIDER_DIRECT_SQL('SELECT * FROM s', '', 'srv "node1", port "8607"') |
+---------------------------------------------------------------------------+
|                                                                         1 |
+---------------------------------------------------------------------------+
```

See also

- SPIDER_BG_DIRECT_SQL

This page is licensed: CC BY-SA / Gnu FDL

SPIDER_FLUSH_TABLE_MON_CACHE

Syntax

```
SPIDER_FLUSH_TABLE_MON_CACHE()
```

Description

A UDF installed with the Spider Storage Engine, this function is used for refreshing monitoring server information. It returns a value of `1`.

Examples

```
SELECT SPIDER_FLUSH_TABLE_MON_CACHE();
+--------------------------------+
| SPIDER_FLUSH_TABLE_MON_CACHE() |
+--------------------------------+
|                              1 |
+--------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Spider FAQ

What does “[ERROR] mysqld: Can’t find record in ‘spider_tables’” mean?

This happens when you have a Spider table defined that does not point to an existing table on a data node.

Are there minimum Spider settings?

```
myisam-recover=FORCE,BACKUP
```

What does “select spider_ping_table()” in the general log mean?

This is used by Spider monitoring to ask other monitoring nodes the status of a table.

Do I need a primary key on physical tables?

Not having a primary key will generate errors for resynchronizing tables via spider_copy_table().

Can I use Spider on top of Galera shards?

Yes, XA transactions can be disabled from Spider. Until Galera 4.0 fully supports xa transactions, spider can point to a maxscale proxy that can manage transparent node election in case of failure inside a shard group. Note that disabling XA will break cross shard WRITES in case of transaction ROLLBACK.
This architecture need to be used with care if you have a highly transactional workload that can generate cross shard deadlocks.

What are the most used architectures for Spider HA?

- Delegation of shard node replication using asynchronous replication and slave election with GTID.
- Delegation of shard node replication via active passive HA solutions.
- Shard builds via replication into Spider tables is interesting when you can route READS to a pool of Spider nodes reattaching the shards.

What are the most used architectures for Spider Map Reduce?

- Map reduce in Spider is limited to a single table. Building spider on top of some views can eliminate the need to use joins.
- Replication to universal tables to every shard is commonly used to enable the views on each shard.

What about Grants on shards?

- When using MRR and BKA (and you do so with network storage), when Spider needs to create temporary tables on the backends, use the CREATE TEMPORARY TABLES privilege. Spider can still switch to a lower performance solution using spider_bka_mode=2, or Query push down or range predicate using spider_bka_mode=0

This page is licensed: CC BY-SA / Gnu FDL

VIDEX Storage Engine

> **Note:**
>
> VIDEX is a community contribution to MariaDB Server, developed and maintained by ByteDance. This storage engine is available from MariaDB 12.3.

This document explains how to install and use **VIDEX** with MariaDB, including:

- Installing/enabling the **VIDEX plugin** in MariaDB.
- Running the **VIDEX-Server** (statistics service) as a container.
- Running a one-shot **videx-sync** workflow to build a VIDEX database.
- Comparing `EXPLAIN` between your original schema and the VIDEX schema.

The VIDEX server repository, with more examples and extension points, can be accessed here. It contains an example dataset (TPC-H tiny).

Prerequisites

- A running MariaDB server you can connect to (the *target*).
- A MariaDB build that includes the `VIDEX` engine plugin (the *MariaDB-VIDEX*).
  - For a quick start, the *target* MariaDB and *MariaDB-VIDEX* can be the same instance.
- Docker (to run `VIDEX-Server` and `videx-sync`).

What is VIDEX

VIDEX is a **virtual/hypothetical index engine** for what-if analysis.

- **Goal**: Evaluate how potential indexes (and optimizer decisions such as join orders) would change query plans *without creating real indexes on production data*.
- **How it works**: VIDEX replays optimizer / handler calls using **statistics** (cardinality, NDV, histograms, etc.) instead of reading table data. Complex statistics computation is offloaded to an external service (**VIDEX-Server**) via HTTP.

In practice, you keep your existing MariaDB schema and data as the *target*, and create a second schema whose tables use `ENGINE=VIDEX`. You then run `EXPLAIN` on both schemas and compare plans.

> **Note:**
>
> Research note (AI4DB)
>
> Estimating **multi-column joint NDV (Number of Distinct Value)** and **cardinality** is a challenging research problem. You can extend VIDEX by implementing your own estimation models in **VIDEX-Server** to improve accuracy for your workload.

Components and Roles

VIDEX typically involves the following roles:

- **Target MariaDB**: your original database instance and schema (contains real data).
- **MariaDB with VIDEX plugin** (*MariaDB-VIDEX*): a MariaDB instance that has the `VIDEX` storage engine plugin enabled.
  - It can be the *same* instance as the *target MariaDB* (common for a quick start).
  - Alternatively, it can be a separate MariaDB instance used only for what-if analysis.
- **VIDEX-Server**: a standalone HTTP service that stores metadata/statistics and answers estimation requests from the VIDEX plugin.

This document focuses on a **docker-based workflow** for `VIDEX-Server` and `videx-sync`, so users can complete the VIDEX end-to-end flow with a few `docker` commands, while the MariaDB server itself is managed by users.

Prepare Sample Data

You can test VIDEX with the `TPC-H tiny` sample.

1. Download `tpch_tiny.sql.tar.gz` from here.
2. Create a database, and import the data (replace `<TARGET_HOST>`, `<TARGET_PORT>`, etc. with your settings:

```
mariadb -h<TARGET_HOST> -P<TARGET_PORT> -u<TARGET_USER> -p<TARGET_PASS> \
  -e "CREATE DATABASE tpch_tiny;"

tar -zxf tpch_tiny.sql.tar.gz
mariadb -h<TARGET_HOST> -P<TARGET_PORT> -u<TARGET_USER> -p<TARGET_PASS> \
  -Dtpch_tiny < tpch_tiny.sql
```

Install and Enable the VIDEX Plugin

Connect to your MariaDB instance and check whether the `VIDEX` engine is available:

```
SHOW ENGINES
```

If you see a row for `VIDEX` with `SUPPORT` as `YES` or `DEFAULT`, the engine is available.

Install and Run VIDEX-Server in Docker

`VIDEX-Server` is a separate service providing statistics/estimation over HTTP, and *MariaDB-VIDEX* calls it to get statistics information for generating query plans.

Images

Public images are on Docker Hub: `kangrongme/videx-server:0.2.0`

Start the Server

Expose container port `5001`:

```
docker run -d --name videx-server \
  -p 5001:5001 \
  kangrongme/videx-server:0.2.0
```

When done, the service is reachable under `http://<YOUR_HOST_IP>:5001`.

> **Note:**
>
> Reachability note
>
> Prefer using a **routable IP address** (your host/server IP) instead of `localhost/127.0.0.1`. This matters because both MariaDB-VIDEX (the plugin) and `videx-sync` need to reach `VIDEX-Server`. If any of them run inside a container, `localhost/127.0.0.1` refers to that container itself (not your host), so the service isn't reachable via `localhost`.

Build the VIDEX Schema

The `VIDEX-Server` image supports two entry point modes:

- `server` (default): start `VIDEX-Server`
- `sync`: run a one-shot workflow to collect metadata from `--target`, then:
  - add metadata to `VIDEX-Server`

- create virtual tables in `--videx`

Command Template

```
docker run --rm --name videx-sync \
  kangrongme/videx-server:0.2.0 sync \
  --target <TARGET_HOST>:<TARGET_PORT>:<TARGET_DB>:<TARGET_USER>:<TARGET_PASS> \
  [--videx <VIDEX_HOST>:<VIDEX_PORT>:<VIDEX_DB>:<VIDEX_USER>:<VIDEX_PASS>] \
  [--videx_server <VIDEX_SERVER_HOST>:<VIDEX_SERVER_PORT>]
```

Notes:

1. If `--videx` is not specified, a default database `videx_{TARGET_DB}` is created in `--target`.
2. If you run a separate MariaDB-VIDEX instance, pass that instance as `--videx`.
3. If your `VIDEX-Server` is not `<TARGET_HOST>:5001`, pass `--videx_server` explicitly.

Localhost Failures

Inside a container, `localhost/127.0.0.1` refers to the container itself.

On Linux Docker Engine, you can reach the host via `host.docker.internal` using `--add-host`:

```
docker run --rm --name videx-sync \
  --add-host=host.docker.internal:host-gateway \
  kangrongme/videx-server:0.2.0 sync \
  --target host.docker.internal:<PORT>:<DB>:<USER>:<PASS> \
  --videx host.docker.internal:<PORT>:<VIDEX_DB>:<VIDEX_USER>:<VIDEX_PASS> \
  --videx_server host.docker.internal:<VIDEX_SERVER_PORT>
```

However, if *MariaDB-VIDEX* itself is also running in a container, reachability can become tricky. Using a routable IP is the most robust approach.

Configure the Plugin

On MariaDB, the VIDEX plugin exposes **session** system variables.

```
SHOW VARIABLES LIKE '%videx_server_ip%';
```

Example output:

```
+-----------------+----------------+
| Variable_name   | Value          |
+-----------------+----------------+
| videx_server_ip | 127.0.0.1:5001 |
+-----------------+----------------+
```

- `videx_server_ip`: **critical**. This is the address (`host:port`) that *MariaDB-VIDEX* uses to call `VIDEX-Server`.

Configure them for your current session before running `EXPLAIN` on `ENGINE=VIDEX` tables (replace `<VIDEX_SERVER_HOST>:<VIDEX_SERVER_PORT>` with your settings):

```
SET SESSION videx_server_ip = '<VIDEX_SERVER_HOST>:<VIDEX_SERVER_PORT>';
```

Quickstart

This section assumes:

- Users already have **one MariaDB instance** running.
- Users want to create the VIDEX schema **in the same instance** (which means “MariaDB-VIDEX == Target MariaDB”).

- MariaDB is running and reachable.
- The routable IP is something like `203.0.113.42` (example only).
- MariaDB is reachable at, for instance, `203.0.113.42:15508` (example only).
- `VIDEX-Server` is reachable at, for instance, `203.0.113.42:5001` (example only).
- The default user/password credentials are: `videx` / `password`.

Step 1

Start VIDEX-Server

```
docker run -d --name videx-server \
  -p 5001:5001 \
  kangrongme/videx-server:0.2.0
```

Step 2

Build VIDEX schema via videx-sync

```
docker run --rm --name videx-sync \
  kangrongme/videx-server:0.2.0 sync \
  --target 203.0.113.42:15508:tpch_tiny:videx:password \
  --videx  203.0.113.42:15508:videx_tpch_tiny:videx:password \
  --videx_server 203.0.113.42:5001
```

Step 3

Run EXPLAIN on the original schema

Run `EXPLAIN` on your original tables:

```
USE tpch_tiny;
SET SESSION use_stat_tables = NEVER;
EXPLAIN SELECT s_name, count(*) AS numwait FROM supplier, lineitem l1, orders, nation WHERE s_suppkey =
l1.l_suppkey AND o_orderkey = l1.l_orderkey AND o_orderstatus = 'F' AND l1.l_receiptdate >
l1.l_commitdate AND EXISTS (SELECT * FROM lineitem l2 WHERE l2.l_orderkey = l1.l_orderkey AND
l2.l_suppkey <> l1.l_suppkey) AND NOT EXISTS (SELECT * FROM lineitem l3 WHERE l3.l_orderkey =
l1.l_orderkey AND l3.l_suppkey <> l1.l_suppkey AND l3.l_receiptdate > l3.l_commitdate) AND s_nationkey
= n_nationkey AND n_name = 'IRAQ' GROUP BY s_name ORDER BY numwait DESC, s_name;
```

> **Note:**
>
> Collecting histograms may change MariaDB's cardinality estimates (for instance, histogram-based estimates rather than InnoDB index statistics). Since VIDEX aims to simulate InnoDB index engine behavior, users can set `SET SESSION use_stat_tables = NEVER;` to make `EXPLAIN` results more similar between InnoDB and the VIDEX engine.

The `EXPLAIN` output for the original schema is:

```
+----+--------------------+----------+--------+-------------------------------------------------------
+--------------+---------+--------------------------------+-------+-----------------------------------
----------+
| id | select_type        | table    | type   | possible_keys                                         |
key          | key_len | ref                            | rows  | Extra
|
+----+--------------------+----------+--------+-------------------------------------------------------
+--------------+---------+--------------------------------+-------+-----------------------------------
----------+
| 1  | PRIMARY            | orders   | ALL    | PRIMARY                                               |
<null>       | <null>  | <null>                         | 14944 | Using where; Using temporary; Using
filesort |
| 1  | PRIMARY            | l1       | ref    | LINEITEM_UK1,LINEITEM_FK1                             |
LINEITEM_FK1 | 4       | tpch_tiny.orders.O_ORDERKEY    | 1     | Using where
|
| 1  | PRIMARY            | supplier | eq_ref | PRIMARY,SUPPLIER_FK1,idx_S_NATIONKEY_S_SUPPKEY_S_NAME |
PRIMARY      | 4       | tpch_tiny.l1.L_SUPPKEY         | 1     |
|
| 1  | PRIMARY            | nation   | eq_ref | PRIMARY                                               |
PRIMARY      | 4       | tpch_tiny.supplier.S_NATIONKEY | 1     | Using where
|
| 1  | PRIMARY            | l2       | ref    | LINEITEM_UK1,LINEITEM_FK1                             |
LINEITEM_FK1 | 4       | tpch_tiny.orders.O_ORDERKEY    | 1     | Using where; FirstMatch(nation)
|
| 3  | DEPENDENT SUBQUERY | l3       | ref    | LINEITEM_UK1,LINEITEM_FK1                             |
LINEITEM_UK1 | 4       | tpch_tiny.l1.L_ORDERKEY        | 1     | Using where
|
+----+--------------------+----------+--------+-------------------------------------------------------
+--------------+---------+--------------------------------+-------+-----------------------------------
----------+
```

Step 4

Run EXPLAIN on the VIDEX schema

Run `EXPLAIN` on the VIDEX schema (tables are `ENGINE=VIDEX`):

```
USE videx_tpch_tiny;
SET SESSION videx_server_ip = '203.0.113.42:5001';
EXPLAIN SELECT s_name, count(*) AS numwait FROM supplier, lineitem l1, orders, nation WHERE s_suppkey =
l1.l_suppkey AND o_orderkey = l1.l_orderkey AND o_orderstatus = 'F' AND l1.l_receiptdate >
l1.l_commitdate AND EXISTS (SELECT * FROM lineitem l2 WHERE l2.l_orderkey = l1.l_orderkey AND
l2.l_suppkey <> l1.l_suppkey) AND NOT EXISTS (SELECT * FROM lineitem l3 WHERE l3.l_orderkey =
l1.l_orderkey AND l3.l_suppkey <> l1.l_suppkey AND l3.l_receiptdate > l3.l_commitdate) AND s_nationkey
= n_nationkey AND n_name = 'IRAQ' GROUP BY s_name ORDER BY numwait DESC, s_name;
```

The `EXPLAIN` output for the VIDEX schema is:

```
+----+--------------------+----------+--------+-----------------------------------------------------
+--------------+---------+--------------------------------------+-------+-----------------------------
----------------+
| id | select_type        | table    | type   | possible_keys                                       |
key          | key_len | ref                                  | rows  | Extra
|
+----+--------------------+----------+--------+-----------------------------------------------------
+--------------+---------+--------------------------------------+-------+-----------------------------
----------------+
| 1  | PRIMARY            | orders   | ALL    | PRIMARY                                             |
<null>       | <null>  | <null>                               | 14883 | Using where; Using temporary;
Using filesort |
| 1  | PRIMARY            | l1       | ref    | LINEITEM_UK1,LINEITEM_FK1                           |
LINEITEM_FK1 | 4       | videx_tpch_tiny.orders.O_ORDERKEY    | 1     | Using where
|
| 1  | PRIMARY            | supplier | eq_ref | PRIMARY,SUPPLIER_FK1,idx_S_NATIONKEY_S_SUPPKEY_S_NAME |
PRIMARY      | 4       | videx_tpch_tiny.l1.L_SUPPKEY         | 1     |
|
| 1  | PRIMARY            | nation   | eq_ref | PRIMARY                                             |
PRIMARY      | 4       | videx_tpch_tiny.supplier.S_NATIONKEY | 1     | Using where
|
| 1  | PRIMARY            | l2       | ref    | LINEITEM_UK1,LINEITEM_FK1                           |
LINEITEM_FK1 | 4       | videx_tpch_tiny.orders.O_ORDERKEY    | 1     | Using where; FirstMatch(nation)
|
| 3  | DEPENDENT SUBQUERY | l3       | ref    | LINEITEM_UK1,LINEITEM_FK1                           |
LINEITEM_UK1 | 4       | videx_tpch_tiny.l1.L_ORDERKEY        | 1     | Using where
|
+----+--------------------+----------+--------+-----------------------------------------------------
+--------------+---------+--------------------------------------+-------+-----------------------------
----------------+
```

Compare the output between Step 3 and Step 4.

Notes and Best Practices

1. `videx-sync` can be time-consuming on large schemas, because it needs to collect metadata/statistics. The metadata collection method is extensible; the VIDEX source repository also discusses lighter-weight sampling approaches.
2. Networking matters. Since `VIDEX-Server` is often in a container, `localhost/127.0.0.1` may not refer to what you expect. The routable IP is recommended for reachability, as it ensures that both *MariaDB-VIDEX* and the container can reach it.

This page is licensed: CC BY-SA / Gnu FDL

Converting Tables from MyISAM to InnoDB

The task

You have decided to change one or more tables from MyISAM to InnoDB. That should be as simple as `ALTER TABLE foo ENGINE=InnoDB`. But you have heard that there might be some subtle issues.

This describes possible issues that may arise and what to do about them.

Recommendation. One way to assist in searching for issues in is to do (at least in *nix)

```
mysqldump --no-data --all-databases >schemas
egrep 'CREATE|PRIMARY' schemas   # Focusing on PRIMARY KEYs
egrep 'CREATE|FULLTEXT' schemas  # Looking for FULLTEXT indexes
egrep 'CREATE|KEY' schemas       # Looking for various combinations of indexes
```

Understanding how the indexes work will help you better understand what might run faster or slower in InnoDB.

INDEX Issues

(Most of these Recommendations and some of these Facts have exceptions.)

Fact. Every InnoDB table has a PRIMARY KEY. If you do not provide one, then the first non-NULL UNIQUE key is used. If that can't be done, then a 6-byte, hidden, integer is provided.

Recommendation. Look for tables without a PRIMARY KEY. Explicitly specify a PRIMARY KEY, even if it's an artificial AUTO_INCREMENT. This is not an absolute requirement, but it is a stronger admonishment for InnoDB than for MyISAM. Some day you may need to walk through the table; without an explicit PK, you can't do it.

Fact. The fields of the PRIMARY KEY are included in each Secondary key.

- Check for redundant indexes with this in mind.

```
PRIMARY KEY(id),
INDEX(b), -- effectively the same as INDEX(b, id)
INDEX(b, id) -- effectively the same as INDEX(b)
```

- (Keep one of the INDEXes, not both)
- Note subtle things like

```
PRIMARY KEY(id),
UNIQUE(b), -- keep for uniqueness constraint
INDEX(b, id) -- DROP this one
```

- Also, since the PK and the data coexist:

```
PRIMARY KEY(id),
INDEX(id, b) -- DROP this one; it adds almost nothing
```

Contrast. This feature of MyISAM is not available in InnoDB; the value of 'id' will start over at 1 for each different value of 'abc':

```
id INT UNSIGNED NOT NULL AUTO_INCREMENT,
PRIMARY KEY (abc, id)
```

A way to simulate the MyISAM 'feature' might be something like: What you want is this, but it won't work because it is referencing the table twice:

```
INSERT INTO foo
    (other, id, ...)
    VALUES
    (123, (SELECT MAX(id)+1 FROM foo WHERE other = 123), ...);
```

Instead, you need some variant on this. (You may already have a BEGIN…COMMIT.)

```
BEGIN;
SELECT @id := MAX(id)+1 FROM foo WHERE other = 123 FOR UPDATE;
INSERT INTO foo
    (other, id, ...)
    VALUES
    (123, @id, ...);
COMMIT;
```

Having a transaction is mandatory to prevent another thread from grabbing the same id.

Recommendation. Look for such PRIMARY KEYs. If you find such, ponder how to change the design. There is no straightforward workaround. However, the following may be ok. (Be sure that the datatype for id is big enough since it won't start over.):

```
id INT UNSIGNED NOT NULL AUTO_INCREMENT,
PRIMARY KEY (abc, id),
UNIQUE(id)
```

Recommendation. Keep the PRIMARY KEY short. If you have Secondary keys, remember that they include the fields of the PK. A long PK would make the
Secondary keys bulky. Well, maybe not — if the is a lot of overlap in fields. Example: `PRIMARY KEY(a,b,c), INDEX(c,b,a)`
— no extra bulk.

Recommendation. Check AUTO_INCREMENT sizes.

- BIGINT is almost never needed. It wastes at least 4 bytes per row (versus INT).
- Always use UNSIGNED and NOT NULL.
- MEDIUMINT UNSIGNED (16M max) might suffice instead of INT
- Be sure to be pessimistic — it is painful to ALTER.

Contrast. "Vertical Partitioning". This is where you artificially split a table to move bulky columns (eg, a BLOB) into another, parallel, table. It is beneficial in MyISAM to avoid stepping over the blob when you don't need to read it. InnoDB stores BLOB and TEXT differently — 767 bytes are in the record, the rest is in some other block. So, it may (or may not) be worth putting the tables back together. Caution: An InnoDB row is limited to 8KB, and the 767 counts against that.

Fact. FULLTEXT (prior to MariaDB 10.0.5) and SPATIAL indexes are not available in InnoDB. Note that MyISAM and InnoDB FULLTEXT indexes use different stopword lists and different system variables.

Recommendation. Search for such indexes. Keep such tables in MyISAM.
Better yet, do Vertical Partitioning (see above) to split out the minimum number of columns from InnoDB.

Fact. The maximum length of an INDEX is different between the Engines.
(This change is not likely to hit you, but watch out.) MyISAM allows 1000 bytes; InnoDB allows 767 bytes, just big enough for a

```
VARCHAR(255) CHARACTER SET utf8.

ERROR 1071 (42000): Specified key was too long; max key length is 767 bytes
```

Fact. The PRIMARY KEY is included in the data. Hence, SHOW TABLE STATUS will show and `Index_length` of 0 bytes (or 16KB) for a table with no secondary indexes. Otherwise, `Index_length` is the total size for the secondary keys.

Fact. The PRIMARY KEY is included in the data. Hence, exact match by PK may be a little faster with InnoDB. And, "range" scans by PK are likely to be faster.

Fact. A lookup by Secondary Key traverses the secondary key's BTree, grabs the PRIMARY KEY, then traverses the PK's BTree. Hence, secondary key lookups are a little more cumbersome in InnoDB.

Contrast. The fields of the PRIMARY KEY are included in each Secondary key. This may lead to "Using index" (in the EXPLAIN plan) for InnoDB for cases where it did not happen in MyISAM. (This is a slight performance boost, and counteracts the double-lookup otherwise needed.) However, when "Using index" would be useful on the PRIMARY KEY, MyISAM would do an "index scan", yet InnoDB effectively has to do a "table scan".

Same as MyISAM. Almost always

```
INDEX(a)   -- DROP this one because the other one handles it.
INDEX(a,b)
```

Contrast. The data is stored in PK order. This means that "recent" records are 'clustered' together at the end. This may give you better 'locality of reference' than in MyISAM.

Same as MyISAM. The optimizer almost never uses two indexes in a single SELECT. (5.1 will occasionally do "index merge".) SELECT in subqueries and UNIONs can independently pick indexes.

Subtle issue. When you DELETE a row, the AUTO_INCREMENT id are burned. Ditto for REPLACE, which is a DELETE plus an INSERT.

Very subtle issue. Replication occurs on COMMIT. If you have multiple threads using transactions, the AUTO_INCREMENTs can arrive at a slave out of order. One transaction BEGINs, grabs an id. Then another transaction grabs an id but COMMITs before the first finishes.

Same as MyISAM. "Prefix" indexing is usually bad in both InnoDB and MyISAM. Example: `INDEX(foo(30))`

Non-INDEX Issues

Disk space for InnoDB is likely to be 2-3 times as much as for MyISAM.

MyISAM and InnoDB use RAM radically differently. If you change all your tables, you should make significant adjustments:

- key_buffer_size — small but non-zero; say, 10M;
- innodb_buffer_pool_size — 70% of available RAM

InnoDB has essentially no need for CHECK, OPTIMIZE, or ANALYZE. Remove them from your maintenance scripts. (No real harm if you keep them.)

Backup scripts may need checking. A MyISAM table can be backed up by copying three files. With InnoDB this is only possible if innodb_file_per_table is set to 1. Before MariaDB 10.0, capturing a table or database for copying from production to a development environment was not possible. Change to mysqldump. Since MariaDB 10.0 a hot copy can be created - see Backup and restore overview.

Before MariaDB 5.5, the DATA DIRECTORY table option was not supported for InnoDB. Since MariaDB 5.5 it is supported, but only in CREATE TABLE. INDEX DIRECTORY has no effect, since InnoDB does not use separate files for indexes. To better balance the workload through several disks, the paths of some InnoDB log files can also be changed.

Understand autocommit and BEGIN/COMMIT.

- (default) autocommit = 1: In the absence of any BEGIN or COMMIT statements, every statement is a transaction by itself. This is close to the MyISAM behavior, but is not really the best.
- autocommit = 0: COMMIT will close a transaction and start another one. To me, this is kludgy.
- (recommended) BEGIN…COMMIT gives you control over what sequence of operation(s) are to be considered a transaction and "atomic". Include the
  ROLLBACK statement if you need to undo stuff back to the BEGIN.

Perl's DBIx::DWIW and Java's JDBC have API calls to do BEGIN and COMMIT. These are probably better than 'executing' BEGIN and COMMIT.

Test for errors everywhere! Because InnoDB uses row-level locking, it can stumble into deadlocks that you are not expecting. The engine will automatically ROLLBACK to the BEGIN. The normal recovery is to redo, beginning at the BEGIN. Note that this is a strong reason to have BEGINs.

LOCK/UNLOCK TABLES — remove them. Replace them (sort of) with BEGIN … COMMIT. (LOCK will work if innodb_table_locks is set to 1, but it is less efficient, and may have subtle issues.)

In 5.1, ALTER ONLINE TABLE can speed up some operations significantly.
(Normally ALTER TABLE copies the table over and rebuilds the indexes.)

The "limits" on virtually everything are different between MyISAM and InnoDB.
Unless you have huge tables, wide rows, lots of indexes, etc, you are unlikely to stumble into a different limit.

Mixture of MyISAM and InnoDB? This is OK. But there are caveats.

- RAM settings should be adjusted to accordingly.
- JOINing tables of different Engines works.
- A transaction that affects tables of both types can ROLLBACK InnoDB changes, but will leave MyISAM changes intact.
- Replication: MyISAM statements are replicated when finished; InnoDB statements are held until the COMMIT.

FIXED (vs DYNAMIC) is meaningless in InnoDB.

PARTITION — You can partition MyISAM and InnoDB tables. Remember the screwball rule: You must either

- have no UNIQUE (or PRIMARY) keys, or
- have the value you are "partitioning on" in every UNIQUE key.

The former is not advised for InnoDB. The latter is messy if you want an
AUTO_INCREMENT.

PRIMARY KEY in PARTITION — Since every key must include the field on which you are PARTITIONing, how can AUTO_INCREMENT work?
Well, there seems to be a convenient special case:

- This works: PRIMARY KEY(autoinc, partition_key)
- This does not work for InnoDB: PRIMARY KEY(partition_key, autoinc)

That is, an AUTO_INCREMENT will correctly increment, and be unique across all
PARTITINOs, when it is the first field of the PRIMARY KEY, but not otherwise.

See Also

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: myisam2innodb

This page is licensed: CC BY-SA / Gnu FDL

Machine Learning with MindsDB

> **Note:**
>
> MindsDB is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Overview

MindsDB is a third-party application that interfaces with MariaDB Server to provide Machine Learning capabilities through SQL. The interface is done via the Connect Storage Engine.

Installation

To get a functional MariaDB - MindsDB installation, one needs to install the following components:

- MindsDB: follow the instructions in the project's official documentation.
- Connect Storage Engine must be enabled for the integration to work. See installing the connect storage engine.

MindsDB connects to MariaDB Server via a regular user to setup a dedicated database called `mindsdb`. Which user are used is specified within MindsDB's configuration file.

For example, if MindsDB is installed locally, one can create a user called `mindsdb@localhost`. MindsDB only authenticates via the mysql_native_password plugin, hence one must set a password for the user:

```
CREATE USER mindsdb@localhost;
SET PASSWORD FOR mindsdb@localhost=PASSWORD("password");
```

The user must be granted the global FILE privilege and all privileges on the `mindsdb` database.

```
GRANT FILE ON *.* TO mindsdb@localhost;
GRANT ALL ON mindsdb.* TO mindsdb@localhost;
```

Assuming MindsDB is in the python path one can start up MindsDB with the following parameters:

```
python -m mindsdb --config=$CONFIG_PATH --api=http,mysql
```

Make sure `$CONFIG_PATH` points to the appropriate MindsDB configuration file.

Usage

Always consult the project's official documentation for up-to-date usage scenarios as MindsDB is an actively developed project.

For a step-by-step example, you can consult the following blog post.

If the connection between MindsDB and MariaDB is successful, you should see the `mindsdb` database present and two tables within it: `commands` and `predictors`.

MindsDB, as an AutoML framework does all the work when it comes to training the AI model. What is necessary is to pass it the initial data, which MindsDB retrieves via a SELECT statement. This can be done by inserting into the `predictors` table.

```
INSERT INTO `predictors`
       (`name`, `predict`, `select_data_query`)
VALUES ('bikes_model', 'count', 'SELECT * FROM test.bike_data');
```

The values inserted into predictors act as a command instructing MindsDB to:

1. Train a model called ‘bikes_model’
2. From the input data, learn to predict the ‘count’ column.
3. The input data is generated via the select statement ‘SELECT * FROM test.bike_data’.
   The `select_data_query` should be a valid select that MindsDB can run against MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

FEDERATED Storage Engine

The FEDERATED Storage Engine is a legacy storage engine no longer being supported. A fork, FederatedX is being actively maintained. Since MariaDB 10.0, the CONNECT storage engine also permits accessing a remote database via MySQL or ODBC connection (table types: MYSQL, ODBC).

The FEDERATED Storage Engine was originally designed to let one access data remotely without using clustering or replication, and perform local queries that automatically access the remote data.

This page is licensed: CC BY-SA / Gnu FDL

Triggers & Events

See Event Scheduler.

Utilize the Event Scheduler in MariaDB Server to automate tasks. Learn how to create, manage, and schedule events to execute SQL statements at specified intervals or times.

See Triggers.

Automate actions in MariaDB Server with triggers. Learn how to create and manage triggers that execute automatically before or after data modifications, ensuring data integrity and business logic enfo

This page is licensed: CC BY-SA / Gnu FDL

Event Scheduler

See Events Overview.

An introduction to creating and managing scheduled events, which are named database objects containing SQL statements to be executed by the Event Scheduler.

See ALTER EVENT.

Learn how to modify the characteristics of an existing event, such as its schedule, body, or enabled status, without dropping and recreating it.

See Event Limitations.

A list of restrictions associated with the Event Scheduler, including the inability to return result sets and specific date range limitations.

This page is licensed: CC BY-SA / Gnu FDL

Events Overview

Events are named database objects containing SQL statements that are to be executed at a later stage, either once off, or at regular intervals.

They function very similarly to the Windows Task Scheduler or Unix cron jobs.

Creating, modifying or deleting events requires the EVENT privilege.

Creating Events

Events are created with the CREATE EVENT statement.

Example

```
CREATE EVENT test_event
  ON SCHEDULE EVERY 1 MINUTE DO
   UPDATE test.t1 SET a = a + 1;
```

Executing Events

Events are only executed if the event scheduler is running. This is determined by the value of the event_scheduler system variable, which needs to be set to `On` for the event scheduler to be running.

You can check if the Event scheduler is running with:

```
SHOW PROCESSLIST;
+----+-----------------+-----------+------+---------+------+-----------------------------+---------------
---+----------+
| Id | User            | Host      | db   | Command | Time | State                       | Info
| Progress |
+----+-----------------+-----------+------+---------+------+-----------------------------+---------------
---+----------+
| 40 | root            | localhost | test | Sleep   | 4687 |                             | NULL
|    0.000 |
| 41 | root            | localhost | test | Query   |    0 | init                        | SHOW
PROCESSLIST |    0.000 |
| 42 | event_scheduler | localhost | NULL | Daemon  |   30 | Waiting for next activation | NULL
|    0.000 |
+----+-----------------+-----------+------+---------+------+-----------------------------+---------------
---+----------+
```

If the event scheduler is not running and `event_scheduler` has been set to `OFF`, use:

```
SET GLOBAL event_scheduler = ON;
```

to activate it. If `event_scheduler` has been set to `Disabled`, you cannot change the value at runtime. Changing the value of the `event_scheduler` variable requires the SUPER privilege.

Since MariaDB 10.0.22, setting the event_scheduler system variable will also try to reload the mysql.event table if it was not properly loaded at startup.

Viewing Current Events

A list of current events can be obtained with the SHOW EVENTS statement. This only shows the event name and interval - the full event details, including the SQL, can be seen by querying the Information Schema EVENTS table, or with SHOW CREATE EVENT.

If an event is currently being executed, it can be seen by querying the Information Schema PROCESSLIST table, or with the SHOW PROCESSLIST statement.

Example

```
SHOW EVENTS\G;
*************************** 1. row ***************************
                  Db: test
                Name: test_event
             Definer: root@localhost
           Time zone: SYSTEM
                Type: RECURRING
          Execute at: NULL
      Interval value: 1
      Interval field: MINUTE
              Starts: 2013-05-20 13:46:56
                Ends: NULL
              Status: ENABLED
          Originator: 1
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

```
SHOW CREATE EVENT test_event\G
*************************** 1. row ***************************
               Event: test_event
            sql_mode:
           time_zone: SYSTEM
        Create Event: CREATE DEFINER=`root`@`localhost` EVENT `test_event` ON SCHEDULE EVERY 1 MINUTE
STARTS '2013-05-20 13:46:56' ON COMPLETION NOT PRESERVE ENABLE DO UPDATE test.t1 SET a = a + 1
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

Altering Events

An event can be changed with the ALTER EVENT statement.

Example

```
ALTER EVENT test_event ON SCHEDULE EVERY '2:3' DAY_HOUR;
```

Dropping Events

Events are dropped with the DROP EVENT statement. Events are also automatically dropped once they have run for the final time according to their schedule, unless the ON COMPLETION PRESERVE clause has been specified.

Example

```
DROP EVENT test_event;
Query OK, 0 rows affected (0.00 sec)
```

See Also

- CREATE EVENT
- SHOW CREATE EVENT
- ALTER EVENT
- DROP EVENT
- Event Limitations
- Automating MariaDB Tasks with Events

This page is licensed: CC BY-SA / Gnu FDL

ALTER EVENT

Modifies one or more characteristics of an existing event.

Syntax

```
ALTER
    [DEFINER = { user | CURRENT_USER }]
    EVENT event_name
    [ON SCHEDULE schedule]
    [ON COMPLETION [NOT] PRESERVE]
    [RENAME TO new_event_name]
    [ENABLE | DISABLE | DISABLE ON SLAVE]
    [COMMENT 'comment']
    [DO sql_statement]
```

Description

The `ALTER EVENT` statement is used to change one or more of the characteristics of an existing event without the need to drop and recreate it. The syntax for each of the `DEFINER`, `ON SCHEDULE`, `ON COMPLETION`, `COMMENT`, `ENABLE` / `DISABLE`, and `DO` clauses is exactly the same as when used with CREATE EVENT.

This statement requires the EVENT privilege. When a user executes a successful `ALTER EVENT` statement, that user becomes the definer for the affected event.

(In MySQL 5.1.11 and earlier, an event could be altered only by its definer, or by a user having the SUPER privilege.)

`ALTER EVENT` works only with an existing event:

```
ALTER EVENT no_such_event ON SCHEDULE EVERY '2:3' DAY_HOUR;
ERROR 1539 (HY000): Unknown event 'no_such_event'
```

Examples

```
ALTER EVENT myevent
  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 2 HOUR
  DO
    UPDATE myschema.mytable SET mycol = mycol + 1;
```

See Also

- Events Overview
- CREATE EVENT
- SHOW CREATE EVENT
- DROP EVENT

This page is licensed: GPLv2, originally from fill_help_tables.sql

Event Limitations

The following restrictions apply to Events:

- All of the restrictions listed in Stored Routine Limitations.
- Events cannot return a result set.
- Event names are case insensitive, so it's not possible to define two events in the same database if their case insensitive names will match. This restriction has applied since MariaDB/MySQL 5.1.8. If you are upgrading from an older version of MySQL, and have events that could clash, these events need to be renamed before the upgrade.

- Events do not support dates beyond the maximum that can be represented in the Unix epoch (2038-01-19).
- Events cannot be created, dropped or altered by another stored program, trigger or event.
- Events cannot create, drop or alter stored programs or triggers
- Event timings cannot be strictly predicted. The intervals `MONTH`, `YEAR_MONTH`, `QUARTER`, and `YEAR` are all resolved in months. All others are resolved in seconds. A delay of up to two seconds is possible in extreme cases, and events scheduled to run at the same second cannot be executed in a given order. The `LAST_EXECUTED` column in the INFORMATION_SCHEMA.EVENTS table will however always be accurate to within a second.
- A new connection is used for each execution of statements within the body of an event, so the session counts for server status variables such as `Com_delete` and `Com_select` will not reflect these.
- Because the Event Scheduler depends on grant tables for its functionality, it is automatically disabled when the server is running with –skip-grant-tables.

This page is licensed: CC BY-SA / Gnu FDL

Triggers

See Trigger Overview.

Complete MariaDB triggers overview: BEFORE/AFTER INSERT/UPDATE/DELETE timing, CREATE TRIGGER syntax, FOLLOWS|PRECEDES ordering, and DROP/SHOW TRIGGERS.

See CREATE TRIGGER.

Complete CREATE TRIGGER reference: OR REPLACE, DEFINER, IF NOT EXISTS, FOLLOWS/PRECEDES options for BEFORE/AFTER INSERT, UPDATE, or DELETE triggers rules.

See Trigger Limitations.

Understand the constraints of triggers, such as the prohibition of statements that return result sets or explicitly start/commit transactions.

See Triggers and Implicit Locks.

Explains how triggers can cause implicit locks on referenced tables during the execution of a statement, potentially affecting concurrency.

This page is licensed: CC BY-SA / Gnu FDL

Trigger Overview

A trigger is a set of statements that run when, or are triggered by, an event that occurs on a table.

Events

The event can be an `INSERT`, an `UPDATE`, or a `DELETE`. The trigger can be executed `BEFORE` or `AFTER` the event. A table can have multiple triggers defined for the same event and timing combination.

The LOAD DATA INFILE and LOAD XML statements invoke `INSERT` triggers for each row that is being inserted.

The REPLACE statement is executed with the following workflow:

- `BEFORE INSERT`;
- `BEFORE DELETE` (only if a row is being deleted);
- `AFTER DELETE` (only if a row is being deleted);
- `AFTER INSERT`.

The INSERT … ON DUPLICATE KEY UPDATE statement, when a row already exists, follows the following workflow:

- `BEFORE INSERT`;
- `BEFORE UPDATE`;
- `AFTER UPDATE`.

Otherwise, it works like a normal `INSERT` statement.

Note that TRUNCATE TABLE does not activate any triggers.

MariaDB Enterprise Server 12.3 adds a separate class of *conflict* triggers (`CREATE TRIGGER ... FOR CONFLICT`) that fire on a replica when a row-based replication event conflicts with the replica's local data. See Conflict Detection and Resolution (CDR) Triggers.

Triggers and Errors

With non-transactional storage engines, if a `BEFORE` statement produces an error, the statement isn't executed. Statements that affect multiple rows fail before inserting the current row.

With transactional engines, triggers are executed in the same transaction as the statement that invoked them.

If a warning is issued with the `SIGNAL` or `RESIGNAL` statement (that is, an error with an `SQLSTATE` starting with `'01'`), it is treated like an error.

Creating a Trigger

Here's a simple example to demonstrate a trigger in action. Using these two tables as an example:

```
CREATE TABLE animals (id MEDIUMINT(9)
NOT NULL AUTO_INCREMENT,
name CHAR(30) NOT NULL,
PRIMARY KEY (`id`));

CREATE TABLE animal_count (animals INT);

INSERT INTO animal_count (animals) VALUES(0);
```

We want to increment a counter each time a new animal is added. Here's what the trigger looks like:

```
CREATE TRIGGER increment_animal
AFTER INSERT ON animals
FOR EACH ROW
UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
```

The trigger has:

- a name (in this case, `increment_animal`),
- a trigger time (in this case, after the specified trigger event),
- a trigger event (an `INSERT`),
- a table with which it is associated (`animals`),
- a set of statements to run (here, just the one `UPDATE` statement).

`AFTER INSERT` specifies that the trigger runs *after* an `INSERT`. The trigger could also be set to run *before*, and the statement causing the trigger could be a `DELETE` or an `UPDATE` as well. You can also have multiple triggers for an action. In this case, you can use `FOLLOWS | PRECEDES`` ``*other_trigger_name*` to specify the order of the triggers.

The set of statements to run are the statements on the table of the trigger; therefore, columns/values that change are always just a column name or an expression like `NEW.` *`column_name`*. Table references of other tables must come from explicit table references.

Now, if we insert a record into the `animals` table, the trigger runs, incrementing the *`animal_count`* table.

```
SELECT * FROM animal_count;
+---------+
| animals |
+---------+
|       0 |
+---------+

INSERT INTO animals (name) VALUES('aardvark');
INSERT INTO animals (name) VALUES('baboon');

SELECT * FROM animal_count;
+---------+
| animals |
+---------+
|       2 |
+---------+
```

For more details on the syntax, see CREATE TRIGGER.

Dropping Triggers

To drop a trigger, use the DROP TRIGGER statement. Triggers are also dropped if the table with which they are associated is also dropped.

```
DROP TRIGGER increment_animal;
```

Triggers Metadata

The Information Schema TRIGGERS Table stores information about triggers.

The SHOW TRIGGERS statement returns similar information.

The SHOW CREATE TRIGGER statement returns a `CREATE TRIGGER` statement that creates the given trigger.

More Complex Triggers

Triggers can consist of multiple statements enclosed by a BEGIN and END. If you're entering multiple statements on the command line, temporarily set a new delimiter so that you can use a semicolon to delimit the statements inside your trigger. See Delimiters in the mariadb client for more.

```
DROP TABLE animals;

UPDATE animal_count SET animals=0;

CREATE TABLE animals (id MEDIUMINT(9) NOT NULL AUTO_INCREMENT,
name CHAR(30) NOT NULL,
PRIMARY KEY (`id`))
ENGINE=InnoDB;

DELIMITER //
CREATE TRIGGER the_mooses_are_loose
AFTER INSERT ON animals
FOR EACH ROW
BEGIN
 IF NEW.name = 'Moose' THEN
  UPDATE animal_count SET animal_count.animals = animal_count.animals+100;
 ELSE
  UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
 END IF;
END; //

DELIMITER ;

INSERT INTO animals (name) VALUES('Aardvark');

SELECT * FROM animal_count;
+---------+
| animals |
+---------+
|       1 |
+---------+

INSERT INTO animals (name) VALUES('Moose');

SELECT * FROM animal_count;
+---------+
| animals |
+---------+
|     101 |
+---------+
```

Trigger Errors

If a trigger contains an error and the engine is transactional, or it is a `BEFORE` trigger, the trigger will not run, and will prevent the original statement from running as well. If the engine is non-transactional, and it is an `AFTER` trigger, the trigger will not run, but the original statement will.

Here, we'll drop the above examples and then recreate the trigger with an error, a field that doesn't exist, first using the default InnoDB, a transactional engine, and then again using MyISAM, a non-transactional engine.

```
DROP TABLE animals;

CREATE TABLE animals (id MEDIUMINT(9) NOT NULL AUTO_INCREMENT,
name CHAR(30) NOT NULL,
PRIMARY KEY (`id`))
ENGINE=InnoDB;

CREATE TRIGGER increment_animal
AFTER INSERT ON animals
FOR EACH ROW
UPDATE animal_count SET animal_count.id = animal_count_id+1;

INSERT INTO animals (name) VALUES('aardvark');
ERROR 1054 (42S22): Unknown column 'animal_count.id' in 'field list'

SELECT * FROM animals;
Empty set (0.00 sec)
```

And now the identical procedure, but with a MyISAM table.

```
DROP TABLE animals;

CREATE TABLE animals (id MEDIUMINT(9) NOT NULL AUTO_INCREMENT,
name CHAR(30) NOT NULL,
PRIMARY KEY (`id`))
ENGINE=MyISAM;

CREATE TRIGGER increment_animal
AFTER INSERT ON animals
FOR EACH ROW
UPDATE animal_count SET animal_count.id = animal_count_id+1;

INSERT INTO animals (name) VALUES('aardvark');
ERROR 1054 (42S22): Unknown column 'animal_count.id' in 'field list'

SELECT * FROM animals;
+----+----------+
| id | name     |
+----+----------+
|  1 | aardvark |
+----+----------+
```

The following example shows how to use a trigger to validate data. The SIGNAL statement is used to intentionally produce an error if the email field is not a valid email. As the example shows, in that case, the new row is not inserted (because it is a `BEFORE` trigger).

```
CREATE TABLE user (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    first_name CHAR(20),
    last_name CHAR(20),
    email CHAR(100)
)
    ENGINE = MyISAM;

DELIMITER //
CREATE TRIGGER bi_user
  BEFORE INSERT ON user
  FOR EACH ROW
BEGIN
  IF NEW.email NOT LIKE '_%@_%.__%' THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Email field IS NOT valid';
  END IF;
END; //
DELIMITER ;

INSERT INTO user (first_name, last_name, email) VALUES ('John', 'Doe', 'john_doe.example.net');
ERROR 1644 (45000): Email field is not valid

SELECT * FROM user;
Empty set (0.00 sec)
```

See Also

- CREATE TRIGGER
- DROP TRIGGER
- Information Schema TRIGGERS Table
- SHOW TRIGGERS
- SHOW CREATE TRIGGER
- Trigger Limitations
- Creative uses of triggers: Things you people wouldn't believe (video)

This page is licensed: CC BY-SA / Gnu FDL

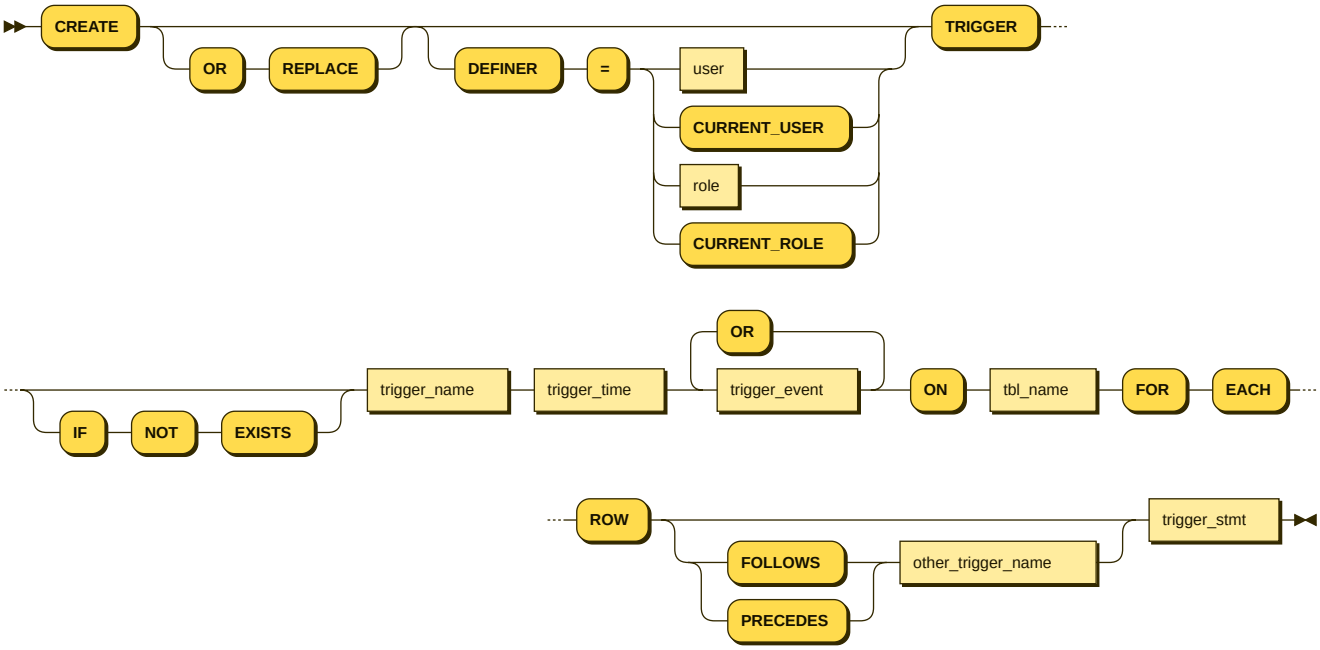
CREATE TRIGGER

Syntax

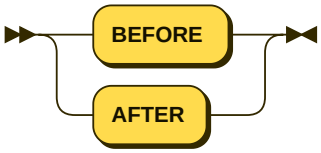
```
CREATE [OR REPLACE]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    TRIGGER [IF NOT EXISTS]
            trigger_name trigger_time {trigger_event [OR trigger_event] ...}
    ON tbl_name FOR EACH ROW
    [{ FOLLOWS | PRECEDES } other_trigger_name]
    trigger_stmt

trigger_time:
    BEFORE
  | AFTER

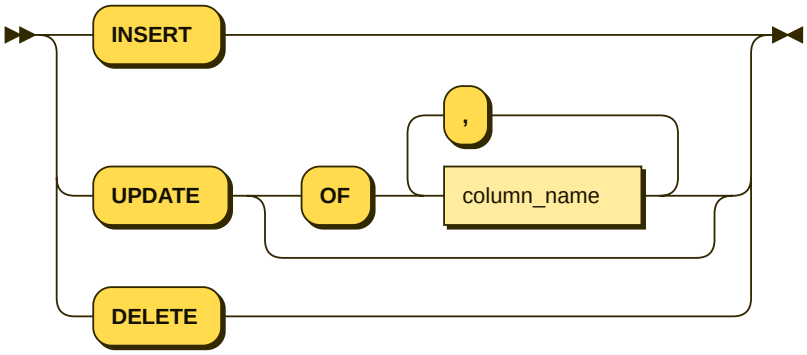
trigger_event:
    INSERT
  | UPDATE [OF column_name [, column_name] ...]
  | DELETE
```



Railroad diagram of CREATE TRIGGER — equivalent to the BNF above



Railroad diagram of trigger_time



Railroad diagram of trigger_event

> **Note:**
>
> MariaDB Enterprise Server 12.3 adds a third trigger form for replicas: `CREATE TRIGGER ... FOR CONFLICT`, which fires when a row-based replication event conflicts with the replica's local data. The `FOR CONFLICT` clause replaces the `BEFORE` / `AFTER` timing keyword and cannot be combined with the trigger events above. See Conflict Detection and Resolution (CDR) Triggers.

Description

This statement creates a new trigger. A trigger is a named database object that is associated with a table, and that activates when a particular event occurs for the table. The trigger becomes associated with the table named `tbl_name`, which must refer to a permanent table. You cannot associate a trigger with a `TEMPORARY` table or a view.

`CREATE TRIGGER` requires the TRIGGER privilege for the table associated with the trigger.

You can have multiple triggers for the same *`trigger_time`* and *`trigger_event`*.

For valid identifiers to use as trigger names, see Identifier Names.

`trigger_stmt` is the statement executed when the trigger activates. Here, you can refer to columns in the table associated with the trigger using the aliases `OLD` and `NEW`. `OLD.` *`col_name`* refers to a column of an existing row before it is updated or deleted. `NEW.` *`col_name`* refers to the column of a new row to be inserted or an existing row after it is updated.

`OLD` and `NEW` are MariaDB (and MySQL) extensions to triggers. They are not case-sensitive.

Triggers cannot use `NEW.` *`col_name`* or use `OLD.` *`col_name`* to refer to generated columns. For information about generated columns, see Generated Columns.

In an `INSERT` trigger, only `NEW.` *`col_name`* can be used, because there is no old row. In a `DELETE` trigger, only `OLD.` *`col_name`* can be used, because there is no new row. In an `UPDATE` trigger, use `OLD.` *`col_name`* to refer to the columns of a row before it is updated, and `NEW.` *`col_name`* to refer to the columns of the row after it is updated.

A column referenced with `OLD` is read-only. If you have the `SELECT` privilege, this means you can refer to it, but not modify it. You can refer to a column named with `NEW` if you have the `SELECT` privilege. In a `BEFORE` trigger, you can also change its value with `SET NEW.` *`col_name`* ``` `=` ``` *`value`* if you have the `UPDATE` privilege. This means you can use a trigger to modify the values to be inserted as a new row or to update a row. In an `AFTER` trigger, a `SET` statement has no effect, because the row change has already occurred.

OR REPLACE

If used and the trigger already exists, instead of an error being returned, the existing trigger is dropped and replaced by the newly defined trigger.

DEFINER

The `DEFINER` clause determines the security context to be used when checking access privileges at trigger activation time. Usage requires the SET USER privilege.

IF NOT EXISTS

If the `IF NOT EXISTS` clause is used, the trigger is created only if a trigger of the same name does not exist. If the trigger already exists, by default a warning is returned.

trigger_time

`trigger_time` is the trigger action time. It can be `BEFORE` or `AFTER` to indicate that the trigger activates before or after each row to be modified.

trigger_event

> **CURRENT**
>
> Multiple *`trigger_event`* events can be specified.
>
> **< 12.0**
>
> Only one *`trigger_event`* can be specified.

`trigger_event` indicates the kind of statement that activates the trigger. A `trigger_event` can be one of the following:

- `INSERT` : The trigger is activated whenever a new row is inserted into the table; for example, through INSERT, LOAD DATA, and REPLACE statements.
- `UPDATE` : The trigger is activated whenever a row is modified; for example, through UPDATE statements.

- `DELETE`: The trigger is activated whenever a row is deleted from the table; for example, through DELETE and REPLACE statements. However, `DROP TABLE` and `TRUNCATE` statements on the table do not activate this trigger, because they do not use `DELETE`. Dropping a partition does not activate `DELETE` triggers, either.

`INSERTING`, `UPDATING`, and `DELETING` Predicates

When a trigger is defined for multiple events, the `INSERTING`, `UPDATING`, and `DELETING` predicates can be used within the trigger body to determine which event fired the trigger:

- `INSERTING`: Evaluates to `TRUE` when the `trigger_event` is `INSERT`
- `UPDATING`: Evaluates to `TRUE` when the `trigger_event` is `UPDATE`
- `DELETING`: Evaluates to `TRUE` when the `trigger_event` is `DELETE`

Example

```
CREATE TRIGGER t1_b_any BEFORE INSERT OR UPDATE OR DELETE ON t1 FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO t2 VALUES (NEW.a, 'INSERTING');
  ELSEIF UPDATING THEN
    INSERT INTO t2 VALUES (NEW.a, 'UPDATING');
  ELSEIF DELETING THEN
    INSERT INTO t2 VALUES (OLD.a, 'DELETING');
  END IF;
END
```

FOLLOWS/PRECEDES *other_trigger_name*

The `FOLLOWS`` ``*other_trigger_name*` and `PRECEDES`` ``*other_trigger_name*` options support multiple triggers per action time.

`FOLLOWS` adds the new trigger after another trigger, while `PRECEDES` adds the new trigger before another trigger. If neither option is used, the new trigger is added last for the given action and time.

`FOLLOWS` and `PRECEDES` are not stored in the trigger definition. However, the trigger order is guaranteed to not change over time. mariadb-dump and other backup methods do not change trigger order.
You can verify the trigger order from the `ACTION_ORDER` column in INFORMATION_SCHEMA.TRIGGERS table.

```
SELECT trigger_name, action_order FROM information_schema.triggers
  WHERE event_object_table='t1';
```

Atomic DDL

> **CURRENT**
>
> MariaDB supports Atomic DDL, and `CREATE TRIGGER` is atomic.
>
> **< 10.6**
>
> MariaDB does **not** support Atomic DDL.

Examples

Creating a Trigger

```
CREATE DEFINER=`root`@`localhost` TRIGGER increment_animal
  AFTER INSERT ON animals FOR EACH ROW
   UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
```

`OR REPLACE` and `IF NOT EXISTS`

```
CREATE DEFINER=`root`@`localhost` TRIGGER increment_animal
  AFTER INSERT ON animals FOR EACH ROW
    UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
ERROR 1359 (HY000): Trigger already exists

CREATE OR REPLACE DEFINER=`root`@`localhost` TRIGGER increment_animal
  AFTER INSERT ON animals  FOR EACH ROW
    UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
Query OK, 0 rows affected (0.12 sec)

CREATE DEFINER=`root`@`localhost` TRIGGER IF NOT EXISTS increment_animal
  AFTER INSERT ON animals FOR EACH ROW
    UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-------+------+------------------------+
| Level | Code | Message                |
+-------+------+------------------------+
| Note  | 1359 | Trigger already exists |
+-------+------+------------------------+
1 row in set (0.00 sec)
```

Referencing NEW Column Values

```
DELIMITER //

CREATE TRIGGER trg_limit_population BEFORE UPDATE ON country_stats
FOR EACH ROW
BEGIN
    -- Ensure population is at least 1
    IF NEW.population < 1 THEN
        SET NEW.population = 1;
    -- Cap population at 2 billion for data integrity
    ELSEIF NEW.population > 2000000000 THEN
        SET NEW.population = 2000000000;
    END IF;
END;
//

DELIMITER ;
```

See Also

- Identifier Names
- Trigger Overview
- DROP TRIGGER
- Information Schema TRIGGERS Table
- SHOW TRIGGERS
- SHOW CREATE TRIGGER
- Trigger Limitations

This page is licensed: GPLv2, originally from fill_help_tables.sql

Trigger Limitations

The following restrictions apply to triggers:

- All of the restrictions listed in Stored Routine Limitations.
- All of the restrictions listed in Stored Function Limitations.

- Triggers are always executed for each row. MariaDB does not support the standard `FOR EACH STATEMENT` option.
- Triggers cannot operate on any tables in the mysql, `information_schema` or `performance_schema` database.
- Cannot return a result set.
- The RETURN statement is not permitted, since triggers don't return any values. Use LEAVE to immediately exit a trigger.
- Triggers are not activated by foreign key actions.
- If a trigger is loaded into cache, it is not automatically reloaded when the table metadata changes. In this case a trigger can operate using the outdated metadata.
- By default, with row-based replication, triggers run on the master, and the effects of their executions are replicated to the slaves. It is possible to run triggers on slaves. See Running triggers on the slave for Row-based events.

See Also

- Trigger Overview
- CREATE TRIGGER
- DROP TRIGGER
- Information Schema TRIGGERS Table
- SHOW TRIGGERS
- SHOW CREATE TRIGGER

This page is licensed: CC BY-SA / Gnu FDL

Triggers and Implicit Locks

A trigger may reference multiple tables, and if a LOCK TABLES statement is used on one of the tables, other tables may at the same time also implicitly be locked due to the trigger.

If the trigger only reads from the other table, that table are read locked. If the trigger writes to the other table, it is write-locked. If a table is read-locked for reading via `LOCK TABLES`, but needs to be write-locked because it could be modified by a trigger, a write lock is taken.

All locks are acquired together when the `LOCK TABLES` statement is issued, and released together on `UNLOCK TABLES`.

Example

```
LOCK TABLE table1 WRITE
```

Assume `table1` contains the following trigger:

```
CREATE TRIGGER trigger1 AFTER INSERT ON table1 FOR EACH ROW
BEGIN
  INSERT INTO table2 VALUES (1);
  UPDATE table3 SET writes = writes+1
    WHERE id = NEW.id AND EXISTS (SELECT id FROM table4);
END;
```

Not only is `table1` write locked, `table2` and `table3` are also write locked, due to the possible INSERT and UPDATE, while `table4` is read locked due to the SELECT.

This page is licensed: CC BY-SA / Gnu FDL

User-Defined Functions

See User-Defined Functions Overview.

An introduction to User-Defined Functions (UDFs) in MariaDB, explaining how they extend the server's functionality by adding new native-like functions.

See Creating User-Defined Functions.

A guide for developers on writing UDFs in C/C++, covering the required interface functions, memory allocation, and thread safety.

See CREATE FUNCTION UDF.

Install a user-defined function from a shared library. This command loads an external compiled function into the server for extended capabilities.

See DROP FUNCTION UDF.

Documentation for the DROP FUNCTION statement, which uninstalls a UDF and removes its entry from the system table.

See User-Defined Functions Calling Sequences.

Technical details on the execution flow of UDFs, explaining the sequence in which initialization, processing, and de-initialization functions are called.

See User-Defined Functions Security.

Overview of security measures for UDFs, including file location restrictions, required privileges, and system variable configurations.

This page is licensed: CC BY-SA / Gnu FDL

User-Defined Functions Overview

A user-defined function (UDF) is a historical way to extend MariaDB with a new function that works similar to a native (built-in) MariaDB function such as ABS( ) or CONCAT( ). It was introduced in 1998 and is generally limited to supporting features that existed at that time.

Statements making use of user-defined functions are not safe for replication.

For an example, see `sql/udf_example.cc` in the source tree. For a collection of existing UDFs go to the UDF Repository on GitHub.

There are alternative ways to add a new function: a native function, which requires modifying and compiling the server source code; a function plugin; or a stored function.

This page is licensed: CC BY-SA / Gnu FDL

Creating User-Defined Functions

User-defined functions allow MariaDB to be extended with a new function that works like a native (built-in) MariaDB function such as ABS() or CONCAT(). It was introduced in 1998 and is generally limited to supporting features that existed at that time. There are alternative ways to add a new function: a native function, which requires modifying and compiling the server source code; a function plugin; or a stored function.

Statements making use of user-defined functions are not safe for replication.

Functions are written in C or C++, and to make use of them, the operating system must support dynamic loading.

Each new SQL function requires corresponding functions written in C/C++. In the list below, at least the main function - x() - and one other, are required. *x* should be replaced by the name of the function you are creating.

All functions need to be thread-safe, so not global or static variables that change can be allocated. Memory is allocated in `x_init()` and freed in `x_deinit()`.

Simple Functions

x()

Required for all UDFs; this is where the results are calculated.

C/C++ type	SQL type
char *	STRING
long long	INTEGER
double	REAL

`DECIMAL` functions return string values, and so should be written accordingly. It is not possible to create `ROW` functions.

x_init()

Initialization function for `x()`. Can be used for the following:

- Check the number of arguments to `X()` (the SQL equivalent).
- Verify the argument types, or to force arguments to be of a particular type after the function is called.
- Specify whether the result can be `NULL`.
- Specify the maximum result length.
- For `REAL` functions, specify the maximum number of decimals for the result.
- Allocate any required memory.

x_deinit()

De-initialization function for `x()`. Used to de-allocate memory that was allocated in `x_init()`.

Description

Each time the SQL function `X()` is called:

- MariaDB will first call the C/C++ initialization function, `x_init()`, assuming it exists. All setup are performed, and if it returns an error, the SQL statement is aborted and no further functions are called.
- If there is no `x_init()` function, or it has been called and did not return an error, x() is then called once per row.
- After all rows have finished processing, `x_deinit()` is called, if present, to clean up by de-allocating any memory that was allocated in `x_init()`.
- See User-defined Functions Calling Sequences for more details on the functions.

Aggregate Functions

The following functions are required for aggregate functions, such as AVG() and SUM(). When using CREATE FUNCTION, the AGGREGATE keyword is required.

x_clear()

Used to reset the current aggregate, but without inserting the argument as the initial aggregate value for the new group.

x_add()

Used to add the argument to the current aggregate.

x_remove()

Improves the support of window functions (so it is not obligatory to add it) and should remove the argument from the current aggregate.

Description

Each time the aggregate SQL function `X()` is called:

- MariaDB will first call the C/C++ initialization function, `x_init()`, assuming it exists. All setup are performed, and if it returns an error, the SQL statement is aborted and no further functions are called.
- If there is no `x_init()` function, or it has been called and did not return an error, `x()` is then called once per row.
- After all rows have finished processing, `x_deinit()` is called, if present, to clean up by de-allocating any memory that was allocated in `x_init()`.
- MariaDB will first call the C/C++ initialization function, `x_init()`, assuming it exists. All setup are performed, and if it returns an error, the SQL statement is aborted and no further functions are called.
- The table is sorted according to the GROUP BY expression.
- `x_clear()` is called for the first row of each new group.
- `x_add()` is called once per row for each row in the same group.
- `x()` is called when the group changes, or after the last row, to get the aggregate result.
- The latter three steps are repeated until all rows have been processed.
- After all rows have finished processing, `x_deinit()` is called, if present, to clean up by de-allocating any memory that was allocated in `x_init()`.

Examples

For an example, see `sql/udf_example.cc` in the source tree. For a collection of existing UDFs see mysqludf.

See Also

- Stored Functions
- Stored Aggregate Functions
- User-defined Functions Calling Sequences
- allow-suspicious-udfs

This page is licensed: CC BY-SA / Gnu FDL

CREATE FUNCTION UDF

Syntax

```
CREATE [OR REPLACE] [AGGREGATE] FUNCTION [IF NOT EXISTS] function_name
    RETURNS {STRING|INTEGER|REAL|DECIMAL}
    SONAME shared_library_name
```

Description

A user-defined function (UDF) is a way to extend MariaDB with a new function that works like a native (built-in) MariaDB function such as ABS() or CONCAT().

`function_name` is the name that should be used in SQL statements to invoke the function.

To create a function, you must have the INSERT privilege for the mysql database. This is necessary because `CREATE FUNCTION` adds a row to the mysql.func system table that records the function's name, type, and shared library name. If you do not have this table, you should run the mariadb-upgrade command to create it.

UDFs need to be written in C, C++ or another language that uses C calling conventions, MariaDB needs to have been dynamically compiled, and your operating system must support dynamic loading.

For an example, see `sql/udf_example.cc` in the source tree. For a collection of existing UDFs see sql-udf.

Statements making use of user-defined functions are not safe for replication.

For creating a stored function as opposed to a user-defined function, see CREATE FUNCTION.

For valid identifiers to use as function names, see Identifier Names.

RETURNS

The `RETURNS` clause indicates the type of the function's return value, and can be one of string, INTEGER, REAL or DECIMAL. `DECIMAL` functions currently return string values and should be written like STRING functions.

shared_library_name

`shared_library_name` is the basename of the shared object file that contains the code that implements the function. The file must be located in the plugin directory. This directory is given by the value of the plugin_dir system variable. Note that before MariaDB/MySQL 5.1, the shared object could be located in any directory that was searched by your system's dynamic linker.

AGGREGATE

Aggregate functions are summary functions such as SUM() and AVG().
Aggregate UDF functions can be used as window functions.

OR REPLACE

If the optional `OR REPLACE` clause is used, it acts as a shortcut for:

```
DROP FUNCTION IF EXISTS function_name;
CREATE FUNCTION name ...;
```

IF NOT EXISTS

When the IF NOT EXISTS clause is used, MariaDB will return a warning instead of an error if the specified function already exists. Cannot be used together with OR REPLACE.

Upgrading a UDF

To upgrade the UDF's shared library, first run a DROP FUNCTION statement, then upgrade the shared library and finally run the CREATE FUNCTION statement. If you upgrade without following this process, you may crash the server.

Examples

```
CREATE FUNCTION jsoncontains_path RETURNS INTEGER SONAME 'ha_connect.so';
Query OK, 0 rows affected (0.00 sec)
```

OR REPLACE and IF NOT EXISTS:

```
CREATE FUNCTION jsoncontains_path RETURNS INTEGER SONAME 'ha_connect.so';
ERROR 1125 (HY000): Function 'jsoncontains_path' already exists

CREATE OR REPLACE FUNCTION jsoncontains_path RETURNS INTEGER SONAME 'ha_connect.so';
Query OK, 0 rows affected (0.00 sec)

CREATE FUNCTION IF NOT EXISTS jsoncontains_path RETURNS INTEGER SONAME 'ha_connect.so';
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------+
| Level | Code | Message                                 |
+-------+------+-----------------------------------------+
| Note  | 1125 | Function 'jsoncontains_path' already exists |
+-------+------+-----------------------------------------+
```

See Also

- Identifier Names
- DROP FUNCTION
- CREATE FUNCTION

This page is licensed: GPLv2, originally from fill_help_tables.sql

DROP FUNCTION UDF

Syntax

```
DROP FUNCTION [IF EXISTS] function_name
```

Description

This statement drops the user-defined function (UDF) named `function_name`.

To drop a function, you must have the DELETE privilege for the mysql database. This is because `DROP FUNCTION` removes the row from the mysql.func system table that records the function's name, type and shared library name.

For dropping a stored function, see DROP FUNCTION.

Upgrading a UDF

To upgrade the UDF's shared library, first run a DROP FUNCTION statement, then upgrade the shared library and finally run the CREATE FUNCTION statement. If you upgrade without following this process, you may crash the server.

Examples

```
DROP FUNCTION jsoncontains_path;
```

IF EXISTS:

```
DROP FUNCTION jsoncontains_path;
ERROR 1305 (42000): FUNCTION test.jsoncontains_path does not exist

DROP FUNCTION IF EXISTS jsoncontains_path;
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------------+
| Level | Code | Message                                       |
+-------+------+-----------------------------------------------+
| Note  | 1305 | FUNCTION test.jsoncontains_path does not exist |
+-------+------+-----------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

User-Defined Functions Calling Sequences

The functions described in Creating User-defined Functions are expanded on this page. They are declared as follows:

Simple Functions

x()

If `x()` returns an integer, it is declared as follows:

```
long long x(UDF_INIT *initid, UDF_ARGS *args,
            char *is_null, char *error);
```

If `x()` returns a string (`DECIMAL` functions also return string values), it is declared as follows:

```
char *x(UDF_INIT *initid, UDF_ARGS *args,
        char *result, unsigned long *length,
        char *is_null, char *error);
```

If `x()` returns a real, it is declared as follows:

```
double x(UDF_INIT *initid, UDF_ARGS *args,
            char *is_null, char *error);
```

x_init()

```
my_bool x_init(UDF_INIT *initid, UDF_ARGS *args, char *message);
```

x_deinit()

```
void x_deinit(UDF_INIT *initid);
```

Description

`initid` is a parameter passed to all three functions that points to a *UDF_INIT* structure, used for communicating information between the functions. Its structure members are:

- `my_bool maybe_null`
  - maybe_null should be set to 1 if x_init can return a NULL value, Defaults to 1 if any arguments are declared maybe_null.
- `unsigned int decimals`

    - Number of decimals after the decimal point. The default, if an explicit number of decimals is passed in the arguments to the main function, is the maximum number of decimals, so if 9.5, 9.55 and 9.555 are passed to the function, the default would be three (based on 9.555, the maximum). If there are no explicit number of decimals, the default is set to 31, or one more than the maximum for the `DOUBLE`, `FLOAT` and `DECIMAL` types. This default can be changed in the function to suit the actual calculation.

- `unsigned int max_length`
    - Maximum length of the result. For integers, the default is 21. For strings, the length of the longest argument. For reals, the default is 13 plus the number of decimals indicated by `initid->decimals`. The length includes any signs or decimal points. Can also be set to 65KB or 16MB in order to return a `BLOB`. The memory remains unallocated, but this is used to decide on the data type to use if the data needs to be temporarily stored.

- `char *ptr`
    - A pointer for use as required by the function. Commonly, `initid->ptr` is used to communicate allocated memory, with `x_init()` allocating the memory and assigning it to this pointer, `x()` using it, and `x_deinit()` de-allocating it.

- `my_bool const_item`
    - Should be set to 1 in `x_init()` if `x()` always returns the same value, otherwise `0`.

Aggregate Functions

x_clear()

x_clear() is a required function for aggregate functions, and is declared as follows:

```
void x_clear(UDF_INIT *initid, char *is_null, char *error);
```

It is called when the summary results need to be reset, that is at the beginning of each new group. but also to reset the values when there were no matching rows.

is_null is set to point to CHAR(0) before calling x_clear().

In the case of an error, you can store the value to which the error argument points (a single-byte variable, not a string buffer) in the variable.

x_reset()

x_reset() is declared as follows:

```
void x_reset(UDF_INIT *initid, UDF_ARGS *args,
             char *is_null, char *error);
```

It is called on finding the first row in a new group. Should reset the summary variables, and then use *UDF_ARGS* as the first value in the group's internal summary value. The function is not required if the UDF interface uses *x_clear()*.

x_add()

x_add() is declared as follows:

```
void x_add(UDF_INIT *initid, UDF_ARGS *args,
           char *is_null, char *error);
```

It is called for all rows belonging to the same group, and should be used to add the value in *UDF_ARGS* to the internal summary variable.

x_remove()

x_remove() was added in MariaDB 10.4 and is declared as follows (same as *x_add()*):

```
void x_remove(UDF_INIT* initid, UDF_ARGS* args,
              char* is_null, char *error );
```

It adds more efficient support of aggregate UDFs as window functions. *x_remove()* should "subtract" the row (reverse *x_add()*). In MariaDB 10.4 aggregate UDFs will work as WINDOW functions without *x_remove()* but it will not be so efficient.

If *x_remove()* supported (defined) detected automatically.

This page is licensed: CC BY-SA / Gnu FDL

User-Defined Functions Security

The MariaDB server imposes a number of limitations on user-defined functions for security purposes.

- The `INSERT` privilege for the mysql database is required to run CREATE FUNCTION, as a record are added to the mysql.func-table.
- The `DELETE` privilege for the mysql database is required to run DROP FUNCTION as the corresponding record are removed from the mysql.func-table.
- UDF object files can only be placed in the plugin directory, as specified by the value of the plugin_dir system variable.
- At least one symbol, beyond the required `x()` , corresponding to an SQL function `X()`, is required. These can be `x_init()`, `x_deinit()`, `xxx_reset()`, `x_clear()`, and `x_add()` functions (see Creating User-defined Functions). The allow-suspicious-udfs mariadbd option (by default unset) provides a workaround, permitting only one symbol to be used. This is not recommended, as it opens the possibility of loading shared objects that are not legitimate user-defined functions.

This page is licensed: CC BY-SA / Gnu FDL

Views

See ALTER VIEW.

Documentation for the ALTER VIEW statement, which is used to modify an existing view's definition without dropping and recreating it.

See CREATE VIEW.

Complete CREATE VIEW reference: OR REPLACE/IF NOT EXISTS, ALGORITHM=MERGE|TEMPTABLE|UNDEFINED, DEFINER/CURRENT_USER, and SQL SECURITY DEFINER|INVOKER.

See DROP VIEW.

Explains how to use the DROP VIEW statement to remove one or more views from the database, including required privileges.

See Information Schema VIEWS Table.

Describes the VIEWS table in the Information Schema, which provides metadata about all views in the database, such as definition and check options.

See Inserting and Updating with Views.

Covers the conditions under which a view is updatable, allowing INSERT, UPDATE, and DELETE operations to modify the underlying base tables.

See View Algorithms.

Learn about the different algorithms (MERGE, TEMPTABLE, UNDEFINED) MariaDB uses to process views and how they impact performance and updatability.

This page is licensed: CC BY-SA / Gnu FDL

ALTER VIEW

Syntax

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Description

This statement changes the definition of a view, which must exist. The syntax is similar to that for CREATE VIEW and the effect is the same as for `CREATE OR REPLACE VIEW` if the view exists. This statement requires the `CREATE VIEW` and `DROP` privileges for the view, and some privilege for each column referred to in the `SELECT` statement. `ALTER VIEW` is allowed only to the definer or users with the SUPER privilege.

Example

```
ALTER VIEW v AS SELECT a, a*3 AS a2 FROM t;
```

See Also

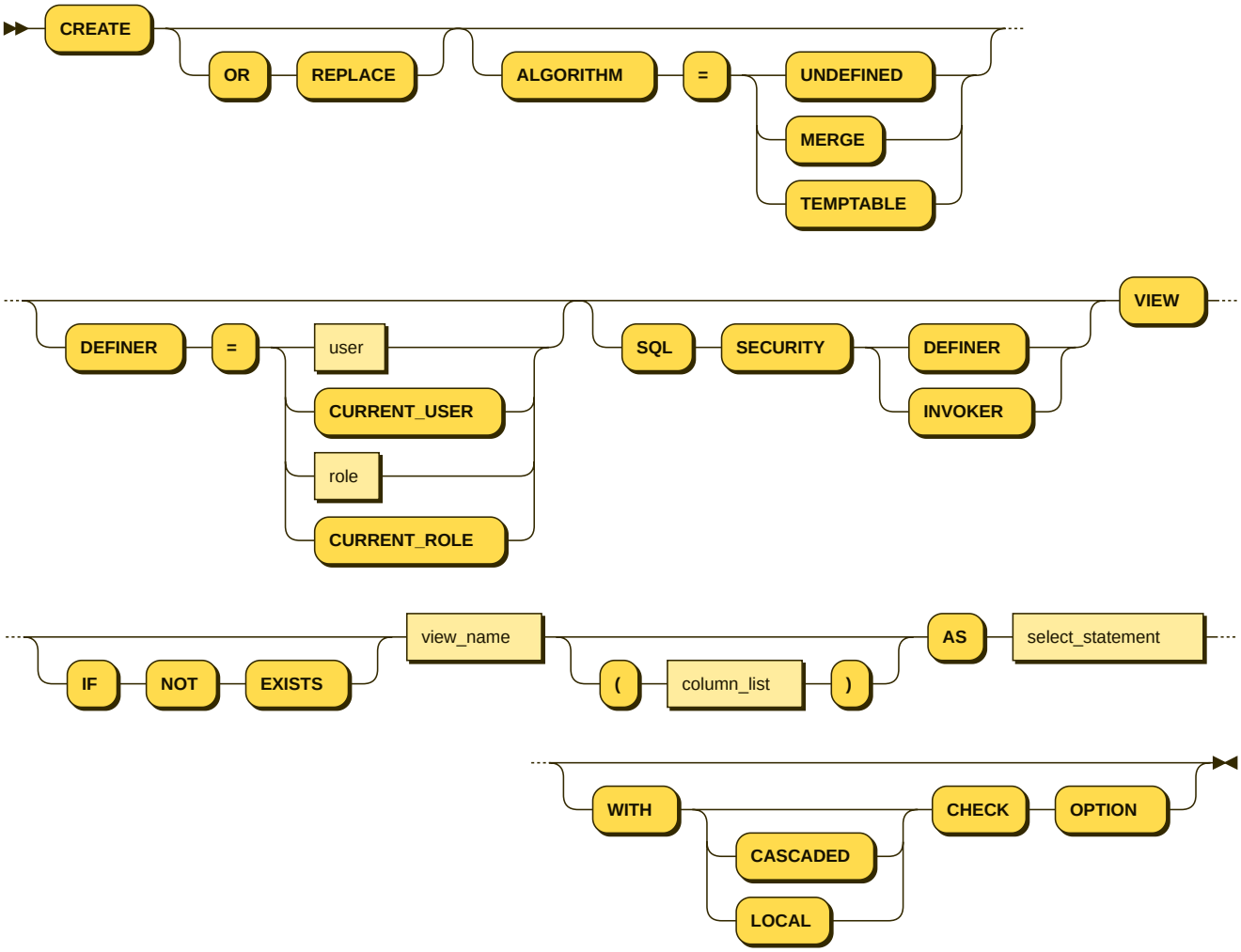
- CREATE VIEW
- DROP VIEW
- SHOW CREATE VIEW
- INFORMATION SCHEMA VIEWS Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

CREATE VIEW

Syntax

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW [IF NOT EXISTS] view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```



Railroad diagram of CREATE VIEW — equivalent to the BNF above

Description

The `CREATE VIEW` statement creates a new view, or replaces an existing one if the `OR REPLACE` clause is given. If the view does not exist, `CREATE OR REPLACE VIEW` is the same as `CREATE VIEW`. If the view does exist, `CREATE OR REPLACE VIEW` is the same as ALTER VIEW.

The `select_statement` is a SELECT statement that provides the definition of the view. (When you select from the view, you select in effect using the `SELECT` statement.) select_statement can select from base tables or other views.

The view definition is “frozen” at creation time, so changes to the underlying tables afterwards do not affect the view definition. For example, if a view is defined as `SELECT *` on a table, new columns added to the table later do not become part of the view. A SHOW CREATE VIEW shows that such queries are rewritten and column names are included in the view definition.

The view definition must be a query that does not return errors at view creation times. However, the base tables used by the views might be altered later and the query may not be valid anymore. In this case, querying the view will result in an error. CHECK TABLE helps in finding this kind of problems.

The ALGORITHM clause affects how MariaDB processes the view. The `DEFINER` and `SQL SECURITY` clauses specify the security context to be used when checking access privileges at view invocation time. The `WITH CHECK OPTION` clause can be given to constrain inserts or updates to rows in tables referenced by the view. These clauses are described later in this section.

The `CREATE VIEW` statement requires the `CREATE VIEW` privilege for the view, and some privilege for each column selected by the `SELECT` statement. For columns used elsewhere in the `SELECT` statement you must have the

`SELECT` privilege. If the `OR REPLACE` clause is present, you must also have the `DROP` privilege for the view.

A view belongs to a database. By default, a new view is created in the default database. To create the view explicitly in a given database, specify the name as db_name.view_name when you create it.

```
CREATE VIEW test.v AS SELECT * FROM t;
```

Base tables and views share the same namespace within a database, so a database cannot contain a base table and a view that have the same name.

Views must have unique column names with no duplicates, just like base tables. By default, the names of the columns retrieved by the `SELECT` statement are used for the view column names. To define explicit names for the view columns, the optional column_list clause can be given as a list of comma-separated identifiers. The number of names in column_list must be the same as the number of columns retrieved by the `SELECT` statement.

Columns retrieved by the `SELECT` statement can be simple references to table columns. They can also be expressions that use functions, constant values, operators, and so forth.

Unqualified table or view names in the `SELECT` statement are interpreted with respect to the default database. A view can refer to tables or views in other databases by qualifying the table or view name with the proper database name.

A view can be created from many kinds of `SELECT` statements. It can refer to base tables or other views. It can use joins, `UNION`, and subqueries. The `SELECT` need not even refer to any tables. The following example defines a view that selects two columns from another table, as well as an expression calculated from those columns:

```
CREATE TABLE t (qty INT, price INT);

INSERT INTO t VALUES(3, 50);

CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;

SELECT * FROM v;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    3 |    50 |   150 |
+------+-------+-------+
```

A view definition is subject to the following restrictions:

- The `SELECT` statement cannot contain a subquery in the `FROM` clause.
- The `SELECT` statement cannot refer to system or user variables.
- Within a stored program, the definition cannot refer to program parameters or local variables.
- The `SELECT` statement cannot refer to prepared statement parameters.
- Any table or view referred to in the definition must exist. However, after a view has been created, it is possible to drop a table or view that the definition refers to. In this case, use of the view results in an error. To check a view definition for problems of this kind, use the `CHECK TABLE` statement.
- The definition cannot refer to a `TEMPORARY` table, and you cannot create a `TEMPORARY` view.
- Any tables named in the view definition must exist at definition time.
- You cannot associate a trigger with a view.
- For valid identifiers to use as view names, see Identifier Names.

`ORDER BY` is allowed in a view definition, but it is ignored if you select from a view using a statement that has its own `ORDER BY`.

For other options or clauses in the definition, they are added to the options or clauses of the statement that references the view, but the effect is undefined. For example, if a view definition includes a `LIMIT` clause, and you select from the view using a statement that has its own `LIMIT` clause, it is undefined which limit applies. This same principle applies to options such as `ALL`, `DISTINCT`, or `SQL_SMALL_RESULT` that follow the `SELECT` keyword, and to clauses such as `INTO`, `FOR UPDATE`, and `LOCK IN SHARE MODE`.

The `PROCEDURE` clause cannot be used in a view definition, and it cannot be used if a view is referenced in the `FROM` clause.

If you create a view and then change the query processing environment by changing system variables, that may affect the results that you get from the view:

```
CREATE VIEW v (mycol) AS SELECT 'abc';

SET sql_mode = '';

SELECT "mycol" FROM v;
+-------+
| mycol |
+-------+
| mycol |
+-------+

SET sql_mode = 'ANSI_QUOTES';

SELECT "mycol" FROM v;
+-------+
| mycol |
+-------+
| abc   |
+-------+
```

The `DEFINER` and `SQL SECURITY` clauses determine which MariaDB account to use when checking access privileges for the view when a statement is executed that references the view. The legal `SQL SECURITY` characteristic values are `DEFINER` and `INVOKER`. These indicate that the required privileges must be held by the user who defined or invoked the view, respectively. The default `SQL SECURITY` value is `DEFINER`.

If a user value is given for the `DEFINER` clause, it should be a MariaDB account in `'user_name'@'host_name'` format (the same format used in the `GRANT` statement). The user_name and host_name values both are required. The definer can also be given as `CURRENT_USER` or `CURRENT_USER()`. The default `DEFINER` value is the user who executes the `CREATE VIEW` statement. This is the same as specifying `DEFINER = CURRENT_USER` explicitly.

If you specify the `DEFINER` clause, these rules determine the legal `DEFINER` user values:

- If you do not have the SET USER privilege, the only legal user value is your own account, either specified literally or by using `CURRENT_USER`. You cannot set the definer to some other account.
- If you have the SET USER privilege, you can specify any syntactically legal account name. If the account does not actually exist, a warning is generated.
- If the SQL `SECURITY` value is `DEFINER` but the definer account does not exist when the view is referenced, an error occurs.

Within a view definition, `CURRENT_USER` returns the view's `DEFINER` value by default. For views defined with the `SQL SECURITY INVOKER` characteristic, `CURRENT_USER` returns the account for the view's invoker. For information about user auditing within views, see account-activity-auditing.html.

Within a stored routine that is defined with the `SQL SECURITY DEFINER` characteristic, `CURRENT_USER` returns the routine's `DEFINER` value. This also affects a view defined within such a program, if the view definition contains a `DEFINER` value of `CURRENT_USER`.

View privileges are checked like this:

- At view definition time, the view creator must have the privileges needed to use the top-level objects accessed by the view. For example, if the view definition refers to table columns, the creator must have privileges for the columns, as described previously. If the definition refers to a stored function, only the privileges needed to invoke the function can be checked. The privileges required when the function runs can be checked only as it executes: For different invocations of the function, different execution paths within the function might be taken.
- When a view is referenced, privileges for objects accessed by the view are checked against the privileges held by the view creator or invoker, depending on whether the `SQL SECURITY` characteristic is `DEFINER` or `INVOKER`, respectively.
- If reference to a view causes execution of a stored function, privilege checking for statements executed within the function depend on whether the function is defined with a `SQL SECURITY` characteristic of `DEFINER` or `INVOKER`. If the security characteristic is `DEFINER`, the function runs with the privileges of its creator. If the characteristic is `INVOKER`, the function runs with the privileges determined by the view's `SQL SECURITY` characteristic.

Example: A view might depend on a stored function, and that function might invoke other stored routines. For example, the following view invokes a stored function `f()`:

```
CREATE VIEW v AS SELECT * FROM t WHERE t.id = f(t.name);

Suppose that f() contains a statement such AS this:

IF name IS NULL THEN
  CALL p1();
ELSE
  CALL p2();
END IF;
```

The privileges required for executing statements within `f()` need to be checked when `f()` executes. This might mean that privileges are needed for `p1()` or `p2()`, depending on the execution path within `f()`. Those privileges must be checked at runtime, and the user who must possess the privileges is determined by the `SQL SECURITY` values of the view `v` and the function `f()`.

The `DEFINER` and `SQL SECURITY` clauses for views are extensions to standard SQL. In standard SQL, views are handled using the rules for `SQL SECURITY INVOKER`.

If you invoke a view that was created before MySQL 5.1.2, it is treated as though it was created with a `SQL SECURITY DEFINER` clause and with a `DEFINER` value that is the same as your account. However, because the actual definer is unknown, MySQL issues a warning. To make the warning go away, it is sufficient to re-create the view so that the view definition includes a `DEFINER` clause.

The optional `ALGORITHM` clause is an extension to standard SQL. It affects how MariaDB processes the view. `ALGORITHM` takes three values: `MERGE`, `TEMPTABLE`, or `UNDEFINED`. The default algorithm is `UNDEFINED` if no `ALGORITHM` clause is present. See View Algorithms for more information.

Some views are updatable. That is, you can use them in statements such as `UPDATE`, `DELETE`, or `INSERT` to update the contents of the underlying table. For a view to be updatable, there must be a one-to-one relationship between the rows in the view and the rows in the underlying table. There are also certain other constructs that make a view non-updatable. See Inserting and Updating with Views.

WITH CHECK OPTION

The `WITH CHECK OPTION` clause can be given for an updatable view to prevent inserts or updates to rows except those for which the `WHERE` clause in the select_statement is true.

In a `WITH CHECK OPTION` clause for an updatable view, the `LOCAL` and `CASCADED` keywords determine the scope of check testing when the view is defined in terms of another view. The `LOCAL` keyword restricts the `CHECK OPTION` only to the view being defined. `CASCADED` causes the checks for underlying views to be evaluated as well. When neither keyword is given, the default is `CASCADED`.

For more information about updatable views and the `WITH CHECK OPTION` clause, see Inserting and Updating with Views.

IF NOT EXISTS

When the I`F NOT EXISTS` clause is used, MariaDB will return a warning instead of an error if the specified view already exists. Cannot be used together with the `OR REPLACE` clause.

Atomic DDL

MariaDB 10.6.1 supports Atomic DDL and `CREATE VIEW` is atomic.

Examples

```
CREATE TABLE t (a INT, b INT) ENGINE = InnoDB;

INSERT INTO t VALUES (1,1), (2,2), (3,3);

CREATE VIEW v AS SELECT a, a*2 AS a2 FROM t;

SELECT * FROM v;
+------+------+
| a    | a2   |
+------+------+
|    1 |    2 |
|    2 |    4 |
|    3 |    6 |
+------+------+
```

OR REPLACE and IF NOT EXISTS:

```
CREATE VIEW v AS SELECT a, a*2 AS a2 FROM t;
ERROR 1050 (42S01): Table 'v' already exists

CREATE OR REPLACE VIEW v AS SELECT a, a*2 AS a2 FROM t;
Query OK, 0 rows affected (0.04 sec)

CREATE VIEW IF NOT EXISTS v AS SELECT a, a*2 AS a2 FROM t;
Query OK, 0 rows affected, 1 warning (0.01 sec)

SHOW WARNINGS;
+-------+------+--------------------------+
| Level | Code | Message                  |
+-------+------+--------------------------+
| Note  | 1050 | Table 'v' already exists |
+-------+------+--------------------------+
```

See Also

- Identifier Names
- ALTER VIEW
- DROP VIEW

- SHOW CREATE VIEWS
- INFORMATION SCHEMA VIEWS Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

DROP VIEW

Syntax

```
DROP VIEW [IF EXISTS]
    view_name [, view_name] ...
    [RESTRICT | CASCADE]
```

Description

`DROP VIEW` removes one or more views. You must have the DROP privilege for each view. If any of the views named in the argument list do not exist, MariaDB returns an error indicating by name which non-existing views it was unable to drop, but it also drops all of the views in the list that do exist.

The `IF EXISTS` clause prevents an error from occurring for views that don't exist. When this clause is given, a `NOTE` is generated for each non-existent view. See SHOW WARNINGS.

`RESTRICT` and `CASCADE`, if given, are parsed and ignored.

It is possible to specify view names as `db_name.view_name`. This is useful to delete views from multiple databases with one statement. See Identifier Qualifiers for details.

The DROP privilege is required to use `DROP TABLE` on non-temporary tables. For temporary tables, no privilege is required, because such tables are only visible for the current session.

If a view references another view, it are possible to drop the referenced view. However, the other view will reference a view which does not exist any more. Thus, querying it will produce an error similar to the following:

```
ERROR 1356 (HY000): View 'db_name.view_name' references invalid table(s) or
column(s) or function(s) or definer/invoker of view lack rights to use them
```

This problem is reported in the output of CHECK TABLE.

Note that it is not necessary to use `DROP VIEW` to replace an existing view, because CREATE VIEW has an `OR REPLACE` clause.

Atomic DDL

MariaDB 10.6.1 supports Atomic DDL and `DROP VIEW` for a singular view is atomic. Dropping multiple views is crash-safe.

Examples

```
DROP VIEW v,v2;
```

Given views `v` and `v2`, but no view `v3`

```
DROP VIEW v,v2,v3;
ERROR 1051 (42S02): Unknown table 'v3'
```

```
DROP VIEW IF EXISTS v,v2,v3;
Query OK, 0 rows affected, 1 warning (0.01 sec)

SHOW WARNINGS;
+-------+------+-------------------------+
| Level | Code | Message                 |
+-------+------+-------------------------+
| Note  | 1051 | Unknown table 'test.v3' |
+-------+------+-------------------------+
```

See Also

- CREATE VIEW
- ALTER VIEW
- SHOW CREATE VIEWS
- INFORMATION SCHEMA VIEWS Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

Information Schema VIEWS Table

The Information Schema `VIEWS` table contains information about views. The `SHOW VIEW` privilege is required to view the table.

It has the following columns:

Column	Description
TABLE_CATALOG	Always def.
TABLE_SCHEMA	Database name containing the view.
TABLE_NAME	View table name.
VIEW_DEFINITION	Definition of the view.
CHECK_OPTION	YES if the WITH CHECK_OPTION clause has been specified, NO otherwise.
IS_UPDATABLE	Whether the view is updatable or not.
DEFINER	Account specified in the DEFINER clause (or the default when created).
SECURITY_TYPE	SQL SECURITY characteristic, either DEFINER or INVOKER.
CHARACTER_SET_CLIENT	The client character set when the view was created, from the session value of the character_set_client system variable.
COLLATION_CONNECTION	The client collation when the view was created, from the session value of the collation_connection system variable.
ALGORITHM	The algorithm used in the view. See View Algorithms.

Example

```
SELECT * FROM information_schema.VIEWS\G
*************************** 1. row ***************************
       TABLE_CATALOG: def
        TABLE_SCHEMA: test
          TABLE_NAME: v
     VIEW_DEFINITION: SELECT `test`.`t`.`qty` AS `qty`,`test`.`t`.`price` AS `price`,(`test`.`t`.`qty` *
`test`.`t`.`price`) AS `value` FROM `test`.`t`
        CHECK_OPTION: NONE
        IS_UPDATABLE: YES
             DEFINER: root@localhost
       SECURITY_TYPE: DEFINER
CHARACTER_SET_CLIENT: utf8
COLLATION_CONNECTION: utf8_general_ci
           ALGORITHM: UNDEFINED
```

See Also

- CREATE VIEW
- ALTER VIEW
- DROP VIEW
- SHOW CREATE VIEWS

This page is licensed: CC BY-SA / Gnu FDL

Inserting and Updating with Views

A view can be used for inserting or updating. However, there are certain limitations.

Updating with Views

A view cannot be used for updating if it uses any of the following:

- `ALGORITHM=TEMPTABLE` (see View Algorithms)
- HAVING
- GROUP BY
- DISTINCT
- UNION
- UNION ALL
- An aggregate function, such as MAX(), MIN(), SUM() or COUNT()
- Subquery in the `SELECT` list
- Subquery in the `WHERE` clause referring to a table in the `FROM` clause
- If it has no underlying table because it refers only to literal values
- The `FROM` clause contains a non-updatdable view.
- Multiple references to any base table column
- An outer join
- An inner join where more than one table in the view definition is being updated
- If there's a `LIMIT` clause, the view does not contain all primary or not null unique key columns from the underlying table and the updatable_views_with_limit system variable is set to `0`.

Inserting with Views

A view cannot be used for inserting if it fails any of the criteria for updating, and must also meet the following conditions:

- the view contains all base table columns that don't have default values
- no base table columns are present in view select list more than once
- the view columns are all simple columns, and not derived in any way. The following are examples of derived columns
  - column_name + 25
  - LOWER(column_name)
  - (subquery)
  - 9.5
  - column1 / column2

Checking Whether a View is Updatable

MariaDB stores an IS_UPDATABLE flag with each view, so it is always possible to see if MariaDB considers a view updatable (although not necessarily insertable) by querying the IS_UPDATABLE column in the INFORMATION_SCHEMA.VIEWS table.

WITH CHECK OPTION

The WITH CHECK OPTION clause is used to prevent updates or inserts to views unless the WHERE clause in the SELECT statement is true.

There are two keywords that can be applied. WITH LOCAL CHECK OPTION restricts the CHECK OPTION to only the view being defined, while WITH CASCADED CHECK OPTION checks all underlying views as well. CASCADED is treated as default if neither keyword is given.

If a row is rejected because of the CHECK OPTION, an error similar to the following is produced:

```
ERROR 1369 (HY000): CHECK OPTION failed 'db_name.view_name'
```

A view with a WHERE which is always false (like `WHERE 0`) and WITH CHECK OPTION is similar to a BLACKHOLE table: no row is ever inserted and no row is ever returned. An insertable view with a WHERE which is always false but no CHECK OPTION is a view that accepts data but does not show them.

Examples

```
CREATE TABLE table1 (x INT);

CREATE VIEW view1 AS SELECT x, 99 AS y FROM table1;
```

Checking whether the view is updateable:

```
SELECT TABLE_NAME,IS_UPDATABLE FROM INFORMATION_SCHEMA.VIEWS;
+------------+--------------+
| TABLE_NAME | IS_UPDATABLE |
+------------+--------------+
| view1      | YES          |
+------------+--------------+
```

This query works, as the view is updateable:

```
UPDATE view1 SET x = 5;
```

This query fails, since column `y` is a literal.

```
UPDATE view1 SET y = 5;
ERROR 1348 (HY000): Column 'y' is not updatable
```

Here are three views to demonstrate the WITH CHECK OPTION clause.

```
CREATE VIEW view_check1 AS SELECT * FROM table1 WHERE x < 100 WITH CHECK OPTION;

CREATE VIEW view_check2 AS SELECT * FROM view_check1 WHERE x > 10 WITH LOCAL CHECK OPTION;

CREATE VIEW view_check3 AS SELECT * FROM view_check1 WHERE x > 10 WITH CASCADED CHECK OPTION;
```

This insert succeeds, as `view_check2` only checks the insert against `view_check2`, and the WHERE clause evaluates to true (`150` is `>10`).

```
INSERT INTO view_check2 VALUES (150);
```

This insert fails, as `view_check3` checks the insert against both `view_check3` and the underlying views. The WHERE clause for `view_check1` evaluates as false (`150` is `>10`, but `150` is not `<100`), so the insert fails.

```
INSERT INTO view_check3 VALUES (150);
ERROR 1369 (HY000): CHECK OPTION failed 'test.view_check3'
```

This page is licensed: CC BY-SA / Gnu FDL

View Algorithms

Description

The CREATE VIEW statement accepts an optional `ALGORITHM` clause, an extension to standard SQL for Views.

It can contain one of three values: `MERGE`, `TEMPTABLE` or `UNDEFINED`, and affects how MariaDB will process the view.

With `MERGE`, the view definition and the related portion of the statement referring to the view are merged. If `TEMPTABLE` is selected, the view results are stored in a temporary table.

`MERGE` is usually more efficient, and a view can only be updated with this algorithm. `TEMPTABLE` can be useful in certain situations, as locks on the underlying tables can be released before the statement is finished processing.

If it's `UNDEFINED` (or the `ALGORITHM` clause is not used), MariaDB will choose what it thinks is the best algorithm. An algorithm can also be `UNDEFINED` if its defined as `MERGE`, but the view requires a temporary table.

Views with definition `ALGORITHM=MERGE` or `ALGORITHM=TEMPTABLE` got accidentally swapped between MariaDB and MySQL. When upgrading, you have to re-create views created with either of these definitions (see MDEV-6916).

MERGE Limitations

A view cannot be of type `ALGORITHM=MERGE` if it uses any of the following:

- HAVING
- LIMIT
- GROUP BY
- DISTINCT
- UNION
- UNION ALL
- An aggregate function, such as MAX(), MIN(), SUM() or COUNT()
- subquery in the `SELECT` list
- if it has no underlying table because it refers only to literal values

MERGE Examples

Example 1

Here's an example of how MariaDB handles a view with a `MERGE` algorithm. Take a view defined as follows:

```
CREATE ALGORITHM = MERGE VIEW view_name (view_field1, view_field2) AS
SELECT field1, field2 FROM table_name WHERE field3 > '2013-06-01';
```

Now, if we run a query on this view, as follows:

```
SELECT * FROM view_name;
```

to execute the view `view_name` becomes the underlying table, `table_name`, the `*` becomes the fields `view_field1` and `view_field2`, corresponding to `field1` and `field2` and the `WHERE` clause, `WHERE field3 > 100` is added, so the actual query executed is:

```
SELECT field1, field2 FROM table_name WHERE field3 > '2013-06-01'
```

Example 2

Given the same view as above, if we run the query:

```
SELECT * FROM view_name WHERE view_field < 8000;
```

Everything occurs as it does in the previous example, but `view_field < 8000` takes the corresponding field name and becomes `field1 < 8000`, connected with `AND` to the `field3 > '2013-06-01'` part of the query.

So the resulting query is:

```
SELECT field1, field2 FROM table_name WHERE (field3 > '2013-06-01') AND (field1 < 8000);
```

When connecting with `AND`, parentheses are added to make sure the correct precedence is used.

This page is licensed: CC BY-SA / Gnu FDL

Server Management

See Deployment.

Learn to install and upgrade MariaDB Server. This section provides detailed instructions and best practices for setting up new instances and seamlessly upgrading existing ones to newer versions.

See Starting & Stopping.

Start and stop MariaDB Server. This section covers service managers like systemd, the mariadbd and mariadbd-safe programs, startup options, and troubleshooting when the server won't start.

See Automated Deployment & Administration.

An overview of automation tools and strategies for deploying MariaDB, comparing systems like Ansible, Puppet, and Kubernetes, and discussing "Infrastructure as Code" principles.

See Server Monitoring & Logs.

Learn about MariaDB Server monitoring and logs. This section guides you through using various logs & monitoring tools to track database activity, troubleshoot issues, and ensure optimal performance.

See Variables and Modes.

Explore MariaDB Server variables and modes. This section explains how to configure global and session variables, and how different SQL modes influence database behavior and compatibility.

This page is licensed: CC BY-SA / Gnu FDL

Deployment

Deployment includes installing, configuring, upgrading, downgrading, migration from other DBMS[15]s, and automated deployment options. Here's an overview of what you can find in this section.

mariadb-package-repository-setup-and-usage

- MariaDB Repository Setup Scripts
- MariaDB Foundation Repository Configuration Tool
- Set up RedHat, Ubuntu, Debian, and Suse repositories

deployment-general-installing-and-upgrading-instructions

- Best practices
- Deployment methods

installing-enterprise-server

- Customer download token
- Single-node installation
- Multi-node installation (primary/replica)
- Installation with ColumnStore
- Upgrade paths

installing-mariadb

- MariaDB ID
- Installing binary packages (executables)
- Installing from source

- Installing system tables
- Troubleshooting installation issues

configuring-mariadb

- Option files
- Environment variables
- Advanced configuration options

upgrading

- Upgrade guides
- Upgrade paths

automated-mariadb-deployment-and-administration

- Automation systems overview
- Ansible
- Puppet
- Containers
- Kubernetes
- Vagrant

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Package Repository Setup and Usage

Overview

If you are looking to set up MariaDB Server, it is often easiest to use a repository. MariaDB Corporation provides two convenient shell scripts to configure access to their MariaDB Package Repositories:

- `mariadb_es_repo_setup` for MariaDB Enterprise Server, which can be downloaded from:
  - https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
- `mariadb_repo_setup` for MariaDB Community Server, which can be downloaded from:
  - https://r.mariadb.com/downloads/mariadb_repo_setup

> **Note:**
>
> **Using the MariaDB Foundation's Repository Configuration Tool**
>
> Another option for MariaDB Community Server is to use the MariaDB Foundation's repository configuration tool. Visit https://mariadb.org/download/?t=repo-config and follow the instructions from there. It will ask for your Linux distribution, desired MariaDB version, and the mirror to use, and will show what files to edit and what commands to run to configure a Community Server repository.

Using MariaDB Corporation's Repository Setup Scripts

Downloading and Verifying the Scripts

The scripts can be downloaded and verified in the following ways:

> **MARIADB_ES_REPO_SETUP**
>
> Download the script:

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

Verify the checksum of the script:

```
echo "${checksum} mariadb_es_repo_setup" | sha256sum -c -
```

MARIADB_REPO_SETUP

Download the script:

```
curl -LsSO https://r.mariadb.com/downloads/mariadb_repo_setup
```

Verify the checksum of the script:

```
echo "${checksum} mariadb_repo_setup" | sha256sum -c -
```

Note:

Checksums of the various releases of the MariaDB Corporation's repository setup scripts can be found in the Versions section at the bottom of this page. Substitute `${checksum}` in the example above with the checksum of the version of the script you are using.

The version number is a date and can be found by running the script with the `--version` option, or by opening the script in a text editor and looking at line 4, `version="..."`.

Script Prerequisites

For the scripts to work, the `curl` package needs to be installed on your system. Additionally, on Debian and Ubuntu, the `apt-transport-https` package needs to be installed. The script will check if these are installed and let you know before it attempts to create the repository configuration on your system.

They can be installed on your system as follows:

RHEL / ROCKY / ALMA

```
sudo dnf install curl
```

DEBIAN / UBUNTU

```
sudo apt update
sudo apt install curl apt-transport-https
```

SLES

```
sudo zypper install curl
```

Running the Scripts

After the script you are using is downloaded, you need to run it with `root` user permissions. This is normally accomplished using the `sudo` command:

MARIADB_ES_REPO_SETUP

Retrieve your customer downloads token:

1. Navigate to https://customers.mariadb.com/downloads/token/ and log in

2. Copy the Customer Download Token
3. Substitute your token for `${token}` when running the `mariadb_es_repo_setup` script, below
4. Set the script to be executable:

```
chmod +x mariadb_es_repo_setup
```

5. Run the script:

```
sudo ./mariadb_es_repo_setup --token="${token}" --apply
```

MARIADB_REPO_SETUP

1. Set the script to be executable:

```
chmod +x mariadb_repo_setup
```

2. Run the script:

```
sudo ./mariadb_repo_setup
```

Repositories

▼ MariaDB Enterprise Repositories

MariaDB Enterprise Repositories

The `mariadb_es_repo_setup` script will set up different repositories in a single repository configuration file. The default repositories set up by the script are:

- MariaDB Enterprise Server Repository
  - Debug packages via the `main/debug` component *(Ubuntu only)*
- MariaDB Enterprise MaxScale Repository
- MariaDB Enterprise Tools Repository

> **Note:**
>
> On Ubuntu, MariaDB Server debug packages are provided through an additional `main/debug` component in the MariaDB Server repository configuration, which the script enables automatically. Other Linux distributions include the debug packages in the main repository. Debug packages should normally only be installed for specific purposes under the direction of a qualified support engineer.

> **Note:**
>
> The **Tools** repository (not Enterprise Tools) is no longer configured by default. If you need it, run the script with the `--include-tools` option.

The `mariadb_es_repo_setup` script configures the MariaDB Enterprise repositories. These are hosted on the MariaDB customer download server (`dlm.mariadb.com`) and require a valid customer download token, which you supply with the `--token` option (see Run the Script for how to retrieve your token).

By default, `mariadb_es_repo_setup` configures three Enterprise repositories: MariaDB Enterprise Server, MariaDB Enterprise Tools, and MariaDB Enterprise MaxScale. It can also configure the Unsupported repository on request.

MariaDB Enterprise Server Repository

The **MariaDB Enterprise Server** repository software packages related to MariaDB Enterprise Server, including: the server itself, clients and utilities, client libraries, plugins, and MariaDB Enterprise Backup.The packages are served from `https://dlm.mariadb.com/repo/<token>/mariadb-enterprise-server/<version>/`.

To configure a different release series, or to pin a specific release, use the `--mariadb-server-version` option. To skip this repository, use the `--skip-server` option.

Output an example MariaDB Enterprise Server repository configuration

To output just the MariaDB Enterprise repository configuration to standard output (stdout), do the following:

```
./mariadb_es_repo_setup --token=$token --skip-check-installed --skip-enterprise-tools --skip-maxscale --
verbose
```

> **Note:**
>
> For more verbose output, use the `--verbose` option

You may also want to use the `--os-type`, `--os-version`, and `--arch` options if you are running the script on a machine running a different operating system and/or processor architecture than the machine the configuration will be used on. If you want to specify a specific MariaDB Enterprise version or series, then also use the `--mariadb-server-version` option.

Descriptions for all of these options are found in the Options section, below.

MariaDB Enterprise MaxScale Repository

> **Warning:**
>
> **Note**
>
> MaxScale Enterprise releases, as of 2025-12-09, are now signed with a new key. The `mariadb_repo_setup` and `mariadb_es_repo_setup` scripts have been updated to automatically install the new key, but for existing repositories, you'll need to do the following.
>
> On Debian and Ubuntu:
>
> ```
> curl -LsSO https://supplychain.mariadb.com/mariadb-keyring-2025.gpg
> sudo mv mariadb-keyring-2025.gpg /etc/apt/trusted.gpg.d/
> sudo apt update
> ```
>
> On RHEL & friends:
>
> ```
> sudo rpm --import https://supplychain.mariadb.com/MariaDB-Enterprise-GPG-KEY-2025
> ```

The **MariaDB Enterprise MaxScale** repository contains MariaDB MaxScale Enterprise packages. The packages are served from `https://dlm.mariadb.com/repo/<token>/maxscale-enterprise/<version>/`.

`mariadb_es_repo_setup` configures the latest stable MariaDB MaxScale Enterprise release series by default. To configure a different version, use the `--mariadb-maxscale-version` option. To skip this repository, use the `--skip-maxscale` option.

MariaDB Enterprise Tools Repository

The **MariaDB Enterprise Tools** repository contains additional tools for use with MariaDB Enterprise. The packages are served from `https://dlm.mariadb.com/repo/<token>/enterprise-tools/<version>/`.

By default, the script configures the latest version of the Enterprise Tools repository. In most cases, you should not change this. However, the option exists to configure a different version. To do so use the `--mariadb-enterprise-tools-version` option. To skip this repository, use the `--skip-enterprise-tools` option.

The MariaDB Enterprise Unsupported Repository

In addition to its default repositories, the `mariadb_es_repo_setup` script can configure a **MariaDB Enterprise Server Unsupported** repository. This repository contains unsupported storage engines and other supplemental packages, such as the CONNECT storage engine.

The Unsupported repository is *not* configured by default. To include it, add the `--include-unsupported` option when you run the script:

```
sudo ./mariadb_es_repo_setup --token="${token}" --include-unsupported --apply
```

Once the repository is configured, you can install its packages using your platform's package manager, as described in Installing Packages With the MariaDB Package Repository.

> **Warning:**
>
> Packages in the Unsupported repository are not covered by MariaDB Enterprise support. Only install them if you understand the implications.

▶ MariaDB Community Repositories

Supported Distributions

The MariaDB Package Repository setup scripts are designed for Linux distributions that meet MariaDB's current platform support policy. Supported platforms may vary over time and can differ across different MariaDB release series.

For a comprehensive and current list of supported platforms, refer to:

- MariDB Engineering Policy
- The MariaDB Downloads page for your specific version

If the setup script does not support your distribution, you can install MariaDB using the MariaDB Foundation's Repository Configuration Tool or check your distribution's native repositories for MariaDB packages.

Options

Using Options with Curl One-line usage

If you are running the script by piping the curl command to bash and you want to provide options to the script, you must tell it to expect them by executing bash with the options `-s --`, for example, to run the script with the `--help` option:

FOR MARIADB_ES_REPO_SETUP

```
curl -LsS https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup | bash -s -- --token=
<token> --help
```

FOR MARIADB_REPO_SETUP

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --help
```

Full List of Options

Option	Description
`--apply`	`mariadb_es_repo_setup` only. Write the repository configuration to the appropriate file, import the GPG public keys used by the repositories, and update the package cache. The default behavior of `mariadb_es_repo_setup` is to output the repository configuration to stdout. See –apply and –write-to-stdout for more information on this option
`--arch=`	Override detection of the CPU architecture. Acceptable values are `x86_64`, `aarch64`, `amd64`, and `arm64`. If you use this option, the script skips its own CPU architecture detection.
`--help`	Display a usage message and exit
`--include-tools`	Include the 'Tools' repository. This repository is no longer configured by default, so use this option to add it.
`--include-unsupported`	`mariadb_es_repo_setup` only. Include the 'Unsupported' repository, which contains unsupported storage engines (such as the CONNECT engine) and other supplemental packages. This repository is not configured by default. See The Unsupported Repository for more information on this option
`--list`	On Debian and Ubuntu, write the older one-line `/etc/apt/sources.list.d/mariadb.list` format instead of the default Deb822 `mariadb.sources` format
`--mariadb-server-version=`	Override the default MariaDB Server version. See –mariadb-server-version for more information on this option
`--mariadb-maxscale-version=`	Override the default MariaDB MaxScale version. See –mariadb-maxscale-version for more information on this option
`--mariadb-enterprise-tools-version=`	`mariadb_es_repo_setup` only. Override the default MariaDB Enterprise Tools version. By default, the script uses the latest version.
`--os-type=`	Override detection of OS type. Acceptable values include `debian`, `ubuntu`, `rhel`, and `sles`. See –ostype and –os-version for more information on this option
`--os-version=`	Override detection of OS version. Acceptable values depend on the OS type you specify, including: `8`, `9`, or `10` for RHEL; `bookworm` or `trixie` for Debian; and `jammy` or `noble` for Ubuntu. See –ostype and –os-version for more information on this option
`--skip-key-import`	Skip importing GPG signing keys, ***not recommended***
`--skip-maxscale`	Skip the 'MaxScale' repository
`--skip-server`	Skip the 'MariaDB Server' repository (either Community or Enterprise depending on the script you are using)
`--skip-tools`	Skip the 'Tools' repository
`--skip-enterprise-tools`	Skip the 'Enterprise Tools" repository (`mariadb_es_repo_setup` only)
`--skip-verify`	Skip verification of MariaDB Server versions. Use with caution, as this can lead to an invalid repository configuration file being created, sometimes useful for testing
`--skip-check-installed`	Skip tests for required prerequisites for this script, useful if generating the config on a different machine
`--skip-eol-check`	Skip tests for versions that are past their EOL date
`--skip-os-eol-check`	Skip tests for operating system versions past the EOL date

Option	Description
`--token`	`mariadb_es_repo_setup` only. Your MariaDB Enterprise Repository token. This is required for `mariadb_es_repo_setup` to work. The token is not required for `mariadb_repo_setup` or for MariaDB Community Server repositories. See Run the Script: mariadb_es_repo_setup for more information.
`--verbose`	`mariadb_es_repo_setup` only. Be more verbose when running by displaying [info] messages about what the script is doing. `mariadb_repo_setup` always runs in verbose mode, so it does not have this option.
`--version`	Output the script version and exit
`--write-to-stdout`	`mariadb_repo_setup` only. Write output to stdout instead of to the OS's repository configuration file. This will also skip importing GPG public keys and updating the package cache on platforms where that behavior exists. See –apply and –write-to-stdout for more information on this option

The `--mariadb-server-version` Option

By default, the script will configure your system to install from the repository of the latest GA version of MariaDB. If a new major GA release occurs and you would like to upgrade to it, then you will need to either manually edit the repository configuration file to point to the new version or run the script again.

The script can also configure your system to install from the repository of a different version of MariaDB by using the `--mariadb-server-version` option.

Usage Example

MARIADB ENTERPRISE SERVER

With `mariadb_es_repo_setup` for MariaDB Enterprise Server, to configure your system to install from the MariaDB Enterprise Server 11.8 repository you would do:

```
./mariadb_es_repo_setup --token=$token --mariadb-server-version="11.8"
```

To specify a specific MariaDB Enterprise Server release, you need to put in the full version number, e.g. for MariaDB Enterprise Server 11.8.5-2:

```
./mariadb_es_repo_setup --token=$token --mariadb-server-version="11.8.5-2"
```

For more details see Pinning the Repository to a Specific Minor Release

MARIADB COMMUNITY SERVER

With mariadb_repo_setup for MariaDB Community Server, the string `mariadb-` has to be prepended to the version number. For example, to configure your system to install from the repository of MariaDB 11.8, that would be:

```
./mariadb_repo_setup --mariadb-server-version="mariadb-11.8"
```

To specify a specific MariaDB Community Server release, you need to put in the full version number, e.g. for MariaDB Community Server 11.8.5:

```
./mariadb_repo_setup --mariadb-server-version="mariadb-11.8.5"
```

For more details see Pinning the Repository to a Specific Minor Release

Current Supported Major Versions

MARIADB ENTERPRISE SERVER

The following MariaDB Enterprise Server versions are currently supported:

- `10.6`
- `11.4`
- `11.8` (default)

MARIADB COMMUNITY SERVER

The following MariaDB Community Server versions are currently supported:

- `mariadb-10.6`
- `mariadb-10.11`
- `mariadb-11.4`
- `mariadb-11.8`
- `mariadb-11.rolling`
- `mariadb-11.rc`
- `mariadb-12.1`
- `mariadb-12.2`
- `mariadb-12.rolling` (default)
- `mariadb-12.rc`

Pinning the Repository to a Specific Minor Release

MARIADB ENTERPRISE SERVER

If you want to pin the repository of a specific minor release, such as MariaDB Enterprise Server 11.8.5-2, then you need to specify the full release version number, e.g. `11.8.5-2`. This may be helpful if you want to avoid automatic upgrades. However, avoiding upgrades is not recommended, since minor maintenance releases may contain important bug fixes and fixes for security vulnerabilities.

> **Note:**
>
> Note: MariaDB Enterprise Server version numbers include a '*release*' suffix that differs from MariaDB Community Server version numbers, e.g. `-2` for MariaDB Enterprise Server `11.8.5-2` whereas the equivalent MariaDB Community Server release is just `11.8.5`. This suffix is required when specifying the full Enterprise Server version number.
>
> To see a list of all MariaDB Enterprise Server releases, see the Enterprise Server - All Releases page. The Version column in the tables on that page list the version numbers of every MariaDB Enterprise Server release.

See the Usage Example - MariaDB Enterprise Server section above for an example of pinning the repository to a specific version.

MARIADB COMMUNITY SERVER

If you want to pin the repository of a specific minor release, such as MariaDB Community Server 11.8.5, then you need to specify the complete release version, e.g. `mariadb-11.8.5`. This may be helpful if you want to avoid upgrades. However, avoiding upgrades is not recommended, since minor maintenance releases may contain important bug fixes and fixes for security vulnerabilities.

> **Note:**
>
> To see a list of all MariaDB Community Server releases, see the Community Server - All Releases page. The Version column in the tables on that page list the version numbers of every MariaDB Community Server release.

> See the Usage Example - MariaDB Community Server section above for an example of pinning the repository to a specific version.

The `--mariadb-maxscale-version` Option

By default, the script will configure your system to install from the repository of the latest GA version of MariaDB MaxScale.

If you would like to pin the repository to a specific version of MariaDB MaxScale, then you will need to either manually edit the repository configuration file to point to the desired version or use the `--mariadb-maxscale-version` option.

For example, to configure your system to install from the repository of MariaDB MaxScale 6.1, that would be:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --mariadb-maxscale-version="6.1"
```

The following MariaDB MaxScale versions are currently supported:

- MaxScale 25.10
- MaxScale 25.01
- MaxScale 24.02
- MaxScale 23.08
- MaxScale 23.02
- MaxScale 22.08

The special identifiers `latest` (for the latest GA release) and `beta` (for the latest beta release) are also supported. By default, the `mariadb_repo_setup` script uses `latest` as the version.

The `--os-type` and `--os-version` Options

If you want to run this script on an unsupported OS that you believe to be package-compatible with an OS that is supported, then you can use the `--os-type` and `--os-version` options to override the script's OS detection. If you use either option, then you must use both options.

The supported values for `--os-type` are:

- `rhel`
- `debian`
- `ubuntu`
- `sles`

If you use a non-supported value, then the script will fail, just as it would fail if you ran the script on an unsupported OS.

The supported values for `--os-version` are entirely dependent on the OS type.

For Red Hat Enterprise Linux (RHEL): `8`, `9`, and `10` are valid options.

For Debian and Ubuntu, the version must be specified as the codename of the specific release. For example, Debian 13 must be specified as `trixie`, and Ubuntu 24.04 must be specified as `noble`.

These options can be useful if your distribution is a fork of another distribution. As an example, Pop!_OS 24.04 LTS is based on and is fully compatible with Ubuntu 24.04 LTS (noble). Therefore, if you are using Pop!_OS, then you can configure your system to install from the repository of Ubuntu 24.04 LTS (noble) by specifying `--os-type=ubuntu`` ``--os-version=noble` to the MariaDB Package Repository setup script.

For example, to manually set the `--os-type` and `--os-version` to RHEL 10, you could do:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --os-type=rhel --os-
version=10
```

The `--apply` and `--write-to-stdout` Options

The `mariadb_repo_setup` and `mariadb_es_repo_setup` scripts generally behave the same but there are some key differences. One of the major ones relates to the options `--apply` and `--write-to-stdout`.

The `--apply` option is only for `mariadb_es_repo_setup`. It directs the script to apply the changes to the system, including creating the repository configuration file, importing the repository GPG keys, and updating the package cache. `mariadb_repo_setup` does this by default, so it does not have the `--apply` option and instead has the `--write-to-stdout` option to flip the behavior.

The `--write-to-stdout` option is only for `mariadb_repo_setup`. It will prevent the script from modifying anything on the system. The repository configuration will not be written to the repository configuration file, GPG keys will not be imported, and the package cache will not be updated. Instead, the repository configuration will be printed to standard output (stdout). This allows the configuration to be reviewed, redirected elsewhere, consumed by another script, or used in some other way. `mariadb_es_repo_setup` does this by default, so it does not have the `--write-to-stdout` option, and instead has the `--apply` option to flip the behavior.

The `--write-to-stdout` option automatically enables `--skip-key-import`.

For example:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --write-to-stdout
```

Both scripts will ignore unknown options so it is safe to, for example, always pass `--apply` to both scripts so that their behavior is the same.

Platform-Specific Behavior

Locations of the repository configuration file the scripts will create on various Linux distributions and other tasks the scripts do on those distributions.

RHEL

Platform-Specific Behavior on RHEL and equivalents

On Red Hat Enterprise Linux (RHEL) and equivalents, the MariaDB Package Repository setup script performs the following tasks:

1. Creates a repository configuration file at `/etc/yum.repos.d/mariadb.repo`
2. Imports the GPG public key used to verify the signature of MariaDB software packages with `rpm --import` from `supplychain.mariadb.com`

DEBIAN / UBUNTU

Platform-Specific Behavior on Debian and Ubuntu

On Debian and Ubuntu, the MariaDB Package Repository setup script performs the following tasks:

1. Creates a repository configuration file at `/etc/apt/sources.list.d/mariadb.list`
2. Creates a package preferences file at `/etc/apt/preferences.d/mariadb-enterprise.pref`, which gives packages from MariaDB repositories a higher priority than packages from OS and other repositories, which can help avoid conflicts. It looks like the following:

```
Package: *
Pin: origin downloads.mariadb.com
Pin-Priority: 1000
```

1. Imports the GPG public key used to verify the signature of MariaDB software package
2. Updates the package cache with package definitions from the MariaDB Package Repository with `apt update`

SLES

Platform-Specific Behavior on SLES

On SUSE Linux Enterprise Server (SLES), the MariaDB Package Repository setup script performs the following tasks:

1. Creates a repository configuration file at `/etc/zypp/repos.d/mariadb.repo`
2. Imports the GPG public key used to verify the signature of MariaDB software packages with `rpm --import` from `supplychain.mariadb.com`

Installing Packages With the MariaDB Package Repository

After setting up the MariaDB Package Repository, you can install the software packages in the supported repositories.

RHEL

Installing Packages on RHEL and equivalents

To install MariaDB on Red Hat Enterprise Linux (RHEL) and equivalents, see the instructions in the Installing MariaDB Packages with YUM. For example:

```
sudo dnf install MariaDB-server MariaDB-client MariaDB-backup
```

To install MariaDB MaxScale on Red Hat Enterprise Linux (RHEL) and equivalents, see the instructions at MariaDB MaxScale Installation Guide. For example:

```
sudo dnf install maxscale
```

DEBIAN / UBUNTU

Installing Packages on Debian and Ubuntu

To install MariaDB on Debian and Ubuntu, see the instructions at Installing MariaDB Packages with APT. For example:

```
sudo apt install mariadb-server mariadb-client mariadb-backup
```

To install MariaDB MaxScale on Debian and Ubuntu, see the instructions at MariaDB MaxScale Installation & Configuration. For example:

```
sudo apt install maxscale
```

SLES

Installing Packages on SLES

To install MariaDB on SUSE Linux Enterprise Server (SLES), see the instructions at Installing MariaDB Packages with ZYpp. For example:

```
sudo zypper install MariaDB-server MariaDB-client MariaDB-backup
```

To install MariaDB MaxScale on SUSE Linux Enterprise Server (SLES), see the instructions at MariaDB MaxScale Installation Guide. For example:

```
sudo zypper install maxscale
```

Versions

The version number of each script is a date, which you can find by running the script with the `--version` option, or by opening the script in a text editor and looking at line 4, `version="..."`. The tables below list the released versions of each script and their `sha256sum` checksums.

Note:

For the list of changes made in each version, see the changelogs:

- mariadb_repo_setup Changelog
- mariadb_es_repo_setup Changelog

MARIADB_ES_REPO_SETUP

mariadb_es_repo_setup Versions

Version	sha256sum
2026-06-30	`f1baa2eb7af306bc17ede4659e207348f23851b47391415b6e4358a51ac45f28`
2026-04-30	`704beacaf236155a69eec63b06987007ba612c7f665dd76cfb384b68fccff8af`
2025-12-10	`62a28aa1f060b4055751d93a88bc11c5556c2b23103c6a6287a8fcb0a4b8a13f`
2025-10-22	`1f584ffd368d18c64b8820bf6cd9b1114dda11a0ecf9524be3c967a3a5be941b`
2025-09-08	`c33b022c2cc325fa50be62eae070ea0bdcaf85367f840accac7acaeea1e8a972`
2025-06-04	`4d483b4df193831a0101d3dfa7fb3e17411dda7fc06c31be4f9e089c325403c0`
2025-01-16	`99ea6c55dbf32bfc42cdcd05c892aebc5e51b06f4c72ec209031639d6e7db9fe`
2025-01-07	`b98c6436e01ff33d7e88513edd7b77a965c4500d6d52ee3f106a198a558927af`
2024-11-19	`97e5ef25b4c4a4bd70b30da46b1eae0b57db2f755ef820a28d254e902ab5a879`
2024-11-13	`0c181ada4e7a4cd1d7688435c478893502675b880be2b918af7d998e239eb325`
2024-09-20	`c12da6a9baa57eab7fa685aa24bf76e6929a8c67f4cd244835520c0181007753`
2024-09-09	`733f247c626d965304b678b62a4b86eb4bb8bf956f98a241b6578dedc6ca4020`
2024-06-12	`b96fcd684a84bbe1080b6276f424537fc9d9c11ebe243ad8b9a45dd459f6ee4f`
2023-07-27	`f8eb9c1b59ccfd979d27e39798d2f2a98447dd29e2149ce92bf606aab4493ad9`
2023-03-13	`8dfef0ec98eb03a4455df07b33107a6d4601425c9df0ab5749b8f10bf3abdcbb`
2022-10-26	`3f4a9d1c507a846a598e95d6223871aade69a9955276455324e7cc5f54a87021`
2022-09-12	`713a8f78ea7bab3eccfb46dc14e61cd54c5cf5a08acb5c320ef5370d375e48bd`
2022-06-14	`cfcd35671125d657a212d92b93be7b1f4ad2fda58dfa8b5ab4b601bf3afa4eae`
2022-03-11	`53efddb84ea12efa7d521499a7474065bd4a60c721492d0e72b4336192f4033f`
2021-12-13	`5feb2aac767c512cc9e2af674d1aef42df0b775ba2968fffa8700eb42702bd44`
2021-10-13	`4f266ff758fe15eeb9b8b448a003eb53e93f3064baf1acb789dd39de4f534b1d`

Version	sha256sum
2021-09-14	`b741361ea3a0a9fcaa30888a63ff3a8a4021882f126cf4ef26cf616493a29315`
2021-08-26	`a49347a4e36f99c5b248403ed9fb9b33a2f07f5e24605a694b1b1e24d7199f28`
2021-06-29	`99e768b24ae430b37dec7cb69cdd625396630dba18f5e1588ee24d3d8bb97064`
2021-06-14	`ec08f8ede524f568b3766795ad8ca1a0d0ac4db355a18c3d85681d7f9c0f8c09`
2021-05-04	`bf67a231c477fba0060996a83b197c29617b6193e1167f6f062216ae13c716c7`
2021-03-15	`99c7f4a3473a397d824d5f591274c2a4f5ebf6dc292eea154800bbaca04ddc7e`
2021-02-12	`c78db828709d94876406a0ea346f13fbc38e73996795903f40e3c21385857dd4`
2020-12-16	`c01fa97aed71ca0cd37cba7036ff80ab40efed4cc261c890aa2aa11cd8ab4e2f`
2020-12-15	`e42f1f16f2c78a3de0e73dcc2a9081e2f771b3161f4f4ceecb13ea788d84673b`
2020-12-14	`4aaf495606633a47c55ea602829e67e702aec0a5c6ff6b1af90709c19ee9f322`
2020-10-07	`93fa0df3d6491a791f5d699158dcfe3e6ce20c45ddc2f534ed2f5eac6468ff0a`
2020-09-08	`eeebe9e08dffb8a4e820cc0f673afe437621060129169ea3db0790eb649dbe9b`
2020-07-16	`957bc29576e8fd320fa18e35fa49b5733f3c8eeb4ca06792fb1f05e089c810ff`

MARIADB_REPO_SETUP

mariadb_repo_setup Versions

Version	sha256sum
2026-06-30	`7325ac7755809ca3312b446bd8325424216992 98f25b701f9a111bb42df0c7c1`
2026-04-23	`96d4ce68b93dc10afc6189bb01a90043bebbd2b8bdc1b62065e12c87a0757b25`
2025-12-10	`73f4ab14ccc3ceb8c03bb283dd131a3235cfc28086475f43e9291d2060d48c97`
2025-11-18	`7a3e1610fee91347e198214e3672a6d3932ccbbf67905d9e892e9255baaec292`
2025-08-07	`923eea378be2c129adb4d191f01162c1fe5473f1114d7586f096b5f6b9874efe`
2025-02-13	`c4a0f3dade02c51a6a28ca3609a13d7a0f8910cccbb90935a2f218454d3a914a`
2024-11-14	`ceaa5bd124c4d10a892c384e201bb6e0910d370ebce235306d2e4b860ed36560`
2024-08-14	`6083ef1974d11f49d42ae668fb9d513f7dc2c6276ffa47caed488c4b47268593`
2024-05-30	`26e5bf36846003c4fe455713777a4e4a613da0df3b7f74b6dad1cb901f324a84`
2024-02-16	`30d2a05509d1c129dd7dd8430507e6a7729a4854ea10c9dcf6be88964f3fdc25`
2023-11-21	`2d7291993f1b71b5dc84cc1d23a65a5e01e783aa765c2bf5ff4ab62814bb5da1`
2023-08-21	`935944a2ab2b2a48a47f68711b43ad2d698c97f1c3a7d074b34058060c2ad21b`
2023-08-14	`f5ba8677ad888cf1562df647d3ee843c8c1529ed63a896bede79d01b2ecc3c1d`
2023-06-09	`3a562a8861fc6362229314772c33c289d9096bafb0865ba4ea108847b78768d2`
2023-02-16	`ad125f01bada12a1ba2f9986a21c59d2cccbe8d584e7f55079ecbeb7f43a4da4`
2022-11-17	`367a80b01083c34899958cdd62525104a3de6069161d309039e84048d89ee98b`
2022-08-22	`733cf126b03f73050e242102592658913d10829a5bf056ab77e7f864b3f8de1f`
2022-08-15	`f99e1d560bd72a3a23f64eaede8982d5494407cafa8f995de45fb9a7274ebc5c`
2022-06-14	`d4e4635eeb79b0e96483bd70703209c63da55a236eadd7397f769ee434d92ca8`
2022-02-08	`b9e90cde27affc2a44f9fc60e302ccfcacf71f4ae02071f30d570e6048c28597`
2022-01-18	`c330d2755e18e48c3bba300a2898b0fc8ad2d3326d50b64e02fe65c67b454599`

This page is licensed: CC BY-SA / Gnu FDL

General Deployment Instructions

See Best Practices.

A guide outlining essential recommendations for stable MariaDB deployments, covering backup strategies, change management, security controls, and pre-production testing.

See Deployment Methods.

Overview of different deployment architectures (single-node, primary/replica) & methods, including manual installation via repositories or tarballs, and automated deployment with tools like Ansible.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Best Practices

These best practices warrant consideration, but are not expected to apply to every business or in every situation. Recommendations here are not mandatory.

General Recommendations

We recommend that you:

- Understand business requirements before deployment.
- Adapt deployment practices to align to business requirements.
- Consider to what extent deployed systems should integrate with existing business systems and practices.

Backups

Addition, removal, and change of database systems are types of production changes. Before undertaking any production changes:

- Perform a backup of existing data and database configurations.
- Establish a plan for data restoration for use in reverting a change.
- Perform a test to confirm your backup is complete and viable.

Irreversible Changes

Installation of new database servers, change of server configuration, migrations, upgrades, and downgrades can produce irreversible changes which may require you to restore from the last good backup.

Changes to data format on disk, including from upgrades and downgrades, are generally irreversible. A fault during the upgrade or downgrade process may corrupt data. Confirm you have a recent, full, complete, and tested backup before any upgrade or downgrade operation.

When System-Versioned temporal data tables are in use, downgrade operations will generally require migration of data between servers or restore from a backup made pre-upgrade.

Full and Complete Backup

Before proceeding with an upgrade, perform a full and complete backup. Confirm successful completion of the backup operation and test the backup.

Business requirements may require you to retain backups made to support upgrade and downgrade operations.

Additional information regarding backing up and restoring MariaDB database products can be found in “Recovery”.

Change Management

Change Management, Automation, Orchestration

Server configuration changes should be done through change management. Accurate records of the time of change and reason for change can enable faster issue diagnosis.

Automation can enable repeatability of change deployment, and can aid Pre-Production and Post-Production testing.

Automation or orchestration can enable repeatability of server deployment, including system provisioning.

Production Controls

Dedicated Servers

Database servers exist to run database server software. Database servers should not also run web server software, act as a file server, or run other workloads.

Understand workload and data isolation requirements before server deployment. Isolation requirements are often defined through business requirements, including:

- Data and application security requirements that trigger isolation of one workload from other workloads.
- Separation of Development and Testing environments from Production environments.

Production Controls

Understand control requirements before server deployment, including records of control implementation needed to support audits.

Control requirements typically follow data:

- Both production and non-production systems may require production-level controls based on the presence of data subject to control.
- Database servers, backup servers, and other systems may require production-level controls based on the presence of data subject to control.

Testing

Servers should be validated before exposure to production workloads and production data.

It may be appropriate to prevent access to an unconfigured server until configured and validated. Load balancer configuration, firewall rules, or database server configuration are often used to prevent unintended traffic to new servers.

Details assessed during Pre-Production validation can include:

- Server capacity, including performance of disk systems
- Server security configuration and hardening
- Tuning for initial data load vs production workloads
- Alignment to business requirements

Updates

MariaDB Product Notifications allow you to keep aware of new releases, including security fixes. Customers can manage MariaDB Product Notifications through the MariaDB Customer Portal.

Additionally, MariaDB Enterprise Server follows an enterprise lifecycle, that provides a predictable release schedule.

Obtaining Support

MariaDB Corporation provides commercial support and services for MariaDB database products.

New customers can contact MariaDB Corporation for more information.

Existing MariaDB Subscription customers can access technical support via the MariaDB Customer Portal as detailed in MariaDB Subscription Services Policy.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Deployment Methods

Overview of Deployment Methods

The most common way users acquire MariaDB is through Linux Distribution Repositories, as many prioritize the stability of native OS packaging. MariaDB's Official Repositories follow closely, favored by those needing the latest features. Containerized deployments (Docker) and Cloud/Managed Services represent a significant and growing portion of the user base, reflecting a shift toward automated and outsourced infrastructure. Finally, a smaller group uses Direct Downloads (Tarballs/MSI) or Source Builds for specialized environments.

Comparison of Installation Methods

Method	Pros	Cons
Linux Distro Repos (deb, rpm, macOS)	High stability; seamless OS integration; managed security updates.	Often contains older versions; slower feature rollout.
MariaDB Repos, Local Repository Mirror	Latest stable releases; direct support for new features.	Requires manual repo setup; potential dependency versioning issues.
Docker / Containers	Excellent isolation; high portability; ideal for microservices.	Networking/storage complexity; requires container expertise.
Cloud / Managed	Minimal maintenance; automated backups/scaling; high availability.	Higher long-term cost; less control over configuration; vendor lock-in.
Windows MSI	GUI-driven; handles service registration and basic security setup.	Limited to Windows; harder to automate at scale than CLI.
Binary Tarballs / Package Tarballs	OS independent; allows multiple versions on one host; no root required.	Manual dependency management; requires manual path configuration.
Source Code	Maximum optimization for hardware; custom feature selection.	Extremely time-consuming; requires compilation expertise.

See Also

Where Do Users Get MariaDB Server From? • Blog • 2026 • 4 minutes

This page is: Copyright © 2026 MariaDB. All rights reserved.

Installing Enterprise Server

See Customer Download Token.

Instructions on how to retrieve and use a Customer Download Token to access MariaDB Enterprise Server packages and binaries.

See Installing MariaDB Enterprise Server.

A guide to installing MariaDB Enterprise Server on various operating systems using package managers (YUM, APT, ZYpp) or binary tarballs.

See Installing Primary/Replica.

This page explains how to set up a standard Primary/Replica replication topology for MariaDB Enterprise Server.

See Enterprise Server with ColumnStore (Local Storage).

Deployment instructions for a single-node MariaDB Enterprise Server instance with the ColumnStore engine using local storage.

See Enterprise Server with ColumnStore (Object Storage).

Deployment instructions for a single-node MariaDB Enterprise Server instance with the ColumnStore engine using S3-compatible object storage.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Customer Download Token

MariaDB Corporation customers can use a Customer Download Token to download MariaDB database products using command-line tools or automation. This page provides instructions on how to retrieve and use the Customer Download Token.

Alternatively, customers can download MariaDB database products using a web browser from the MariaDB Enterprise Download page without a Customer Download Token if they are logged in with their MariaDB ID.

Retrieve Customer Download Token

MariaDB Corporation customers can retrieve their Customer Download Token through the MariaDB Customer Portal.

> **Warning:**
>
> MariaDB Corporation's Customer Download Tokens are customer-specific. Protect the token as you would any security credential.

To retrieve the Customer Download Token for your account:

1. Navigate to the Customer Download Token at the MariaDB Customer Portal.
2. Log in using your MariaDB ID.
3. Copy the Customer Download Token.

Use the Customer Download Token

MariaDB Corporation customers can use their Customer Download Token for multiple operations.

Choose the procedure for your desired operation and substitute your Customer Download Token for `CUSTOMER_DOWNLOAD_TOKEN`:

Operation	Use Case
Configure MariaDB Enterprise Repository	Install MariaDB database products using package managers on CentOS / RHEL (YUM), Debian / Ubuntu (APT), and SLES (ZYpp)
Download Binary Files with CLI or Automation	Download binary files using command-line tools or automation
Log In to MariaDB Enterprise Docker Registry	Deploy using Docker

Configure MariaDB Enterprise Repository

MariaDB Corporation provides the MariaDB Enterprise Repository to install MariaDB database products using package managers on CentOS / RHEL (YUM), Debian / Ubuntu (APT), and SLES (ZYpp).

The MariaDB Enterprise Repository is configured using the `mariadb_es_repo_setup` script, which requires the Customer Download Token to be provided via the `--token` option.

For additional information, see "MariaDB Package Repository Setup and Usage".

Download Binary Files

MariaDB Corporation provides an interface to download binary files using command-line tools or automation.

Binary files can be downloaded using command-line tools or automation from the MariaDB Download interface with the Customer Download Token. The URL is in the following format:

```
https://dlm.mariadb.com/CUSTOMER_DOWNLOAD_TOKEN/FILE
```

Download a binary file using the following procedure:

1. In your web browser, visit the MariaDB Download interface for the specific MariaDB database product:
   - MariaDB Enterprise Server
   - MariaDB MaxScale
2. In your web browser, navigate to the binary file that you would like to download and copy the URL. For example, to download a binary tarball of MariaDB Enterprise Server 10.6.20-16 for RHEL 8 on x86_64, the URL is:\

   ```
   https://dlm.mariadb.com/FILE_ID/mariadb-enterprise-server/10.6.20-16/bintar-rhel-8-x86_64/mariadb-enterprise-10.6.20-16-rhel-8-x86_64.tar.gz
   ```

   `FILE_ID` is an internal identifier that is different for each file.
3. Extract the `FILE` path from the copied URL. For example, to download the file mentioned above, the FILE path is:\

   ```
   FILE_ID/mariadb-enterprise-server/10.6.20-16/bintar-rhel-8-x86_64/mariadb-enterprise-10.6.20-16-rhel-8-x86_64.tar.gz
   ```
4. Use your Customer Download Token and the FILE path to construct your customer-specific URL to download the file using command-line tools or automation.For example, to download the file mentioned above, the customer-specific URL is:\

   ```
   https://dlm.mariadb.com/CUSTOMER_DOWNLOAD_TOKEN/FILE_ID/mariadb-enterprise-server/10.6.20-16/bintar-rhel-8-x86_64/mariadb-enterprise-10.6.20-16-rhel-8-x86_64.tar.gz
   ```
5. Use your customer-specific URL to download the file using command-line tools or automation:For example, to download the file mentioned above using `wget`:

```
wget https://dlm.mariadb.com/CUSTOMER_DOWNLOAD_TOKEN/FILE_ID/mariadb-enterprise-server/10.6.20-16/bintar-rhel-8-x86_64/mariadb-enterprise-10.6.20-16-rhel-8-x86_64.tar.gz
```

or using `curl`:

```
curl -LO https://dlm.mariadb.com/CUSTOMER_DOWNLOAD_TOKEN/FILE_ID/mariadb-enterprise-server/10.6.20-16/bintar-rhel-8-x86_64/mariadb-enterprise-10.6.20-16-rhel-8-x86_64.tar.gz
```

Log in to MariaDB Enterprise Docker Registry

Docker is an open platform for developing, shipping, and running applications that allows you to separate your applications from your infrastructure. MariaDB Corporation provides the MariaDB Enterprise Docker Registry.

The MariaDB Enterprise Docker Registry provides Docker images for MariaDB Enterprise Server. The Docker images for MariaDB Enterprise Server are currently beta maturity, so they are not currently recommended for production.

For additional information, see "Deploy MariaDB Enterprise Server with Docker".

This page is: Copyright © 2026 MariaDB. All rights reserved.

Installing MariaDB Enterprise Server

Installing Enterprise Server

Overview

These instructions detail the deployment of MariaDB Enterprise Server in a Single Standalone Server configuration on a range of supported Operating Systems.

These instructions detail how to deploy a single-node row database, which is suited for a transactional or OLTP workload that does not require high availability (HA). This deployment type is generally for non-production use cases, such as for development and testing.

MariaDB Database Products

These instructions detail the deployment of the following MariaDB database products:

Component	Description
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.

MariaDB Enterprise Server Components

These instructions detail the deployment of the following MariaDB Enterprise Server components:

Component	Description
InnoDB	• It is a general purpose storage engine • It is ACID-compliant • It is performant • It is the transactional component of MariaDB's single stack Hybrid Transactional/Analytical Processing (HTAP) solution

Term Definitions

Term	Definition
row database	• A database where all columns of each row are stored together • Best suited for transactional and OLTP workloads • Also known as a "row-oriented database"

Installation

MariaDB Corporation provides package repositories for YUM (RHEL, CentOS), APT (Debian, Ubuntu), and ZYpp (SLES).

Install via YUM (CentOS, RHEL, Rocky)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the YUM package repository.

Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.4.
To configure YUM package repositories:

```
$ sudo yum install curl

$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup

$ echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -

$ chmod +x mariadb_es_repo_setup

$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.4"
```

3. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*
4. Install MariaDB Enterprise Server and package dependencies:

```
$ sudo yum install MariaDB-server MariaDB-backup
```

4. Configure MariaDB.

Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use. See Configuration.

Install via APT (Debian, Ubuntu)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the APT package repository.

Pass the version to install using the –mariadb-server-version flag to mariadb_es_repo_setup. The following directions reference

To configure APT package repositories:

```
$ sudo apt install curl

$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup

$ echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -

$ chmod +x mariadb_es_repo_setup

$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.4"

$ sudo apt update
```

3. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*
4. Install MariaDB Enterprise Server and package dependencies:

```
$ sudo apt install mariadb-server mariadb-backup
```

4. Configure MariaDB.

Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use. See Configuration.

Install via ZYpp (SLES)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the ZYpp package repository.

Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.4.

To configure ZYpp package repositories:

```
$ sudo zypper install curl

$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup

$ echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -

$ chmod +x mariadb_es_repo_setup

$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.4"
```

3. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*
4. Install MariaDB Enterprise Server and package dependencies:

```
$ sudo zypper install MariaDB-server MariaDB-backup
```

4. Configure MariaDB.

Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use. See Configuration.

Configuration

MariaDB Enterprise Server can be configured in the following ways:

- System variables and options can be set in a configuration file (such as /etc/my.cnf). MariaDB Enterprise Server must be restarted to apply changes made to the configuration file.
- System variables and options can be set on the command-line.
- If a system variable supports dynamic changes, then it can be set on-the-fly using the SET statement.

Configuration Files

MariaDB's packages include several bundled configuration files. It is also possible to create custom configuration files.

On RHEL, CentOS, and SLES, MariaDB's packages bundle the following configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mariadb-enterprise.cnf`

- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

And on RHEL, CentOS, and SLES, custom configuration files from the following directories are read by default:

- `/etc/my.cnf.d/`

On Debian and Ubuntu, MariaDB's packages bundle the following configuration files:

- `/etc/mysql/my.cnf`
- `/etc/mysql/mariadb.cnf`
- `/etc/mysql/mariadb.conf.d/50-client.cnf`
- `/etc/mysql/mariadb.conf.d/50-mysql-clients.cnf`
- `/etc/mysql/mariadb.conf.d/50-mysqld_safe.cnf`
- `/etc/mysql/mariadb.conf.d/50-server.cnf`
- `/etc/mysql/mariadb.conf.d/60-galera.cnf`
- `/etc/mysql/mariadb.conf.d/mariadb-enterprise.cnf`

And on Debian and Ubuntu, custom configuration files from the following directories are read by default:

- `/etc/mysql/conf.d/`
- `/etc/mysql/mariadb.conf.d/`

Configure MariaDB

Determine which system variables and options you need to configure.

Useful system variables and options for MariaDB Enterprise Server include:

System Variable/Option	Description
datadir	Sets the path to the data directory. MariaDB Enterprise Server writes data files to this directory, including tablespaces, logs, and schemas. Change it to use a non-standard location or to start the Server on a different data directory for testing.
bind_address	Sets the local TCP/IP address on which MariaDB Enterprise Server listens for incoming connections. When testing on a local system, bind the address to the local host at 127.0.0.1 to prevent network access.
port	Sets the port MariaDB Enterprise Server listens on. Use this system variable to use a non-standard port or when running multiple Servers on the same host for testing.
max_connections	Sets the maximum number of simultaneous connections MariaDB Enterprise Server allows.
thread_handling	Sets how MariaDB Enterprise Server handles threads for client connections.
log_error	Sets the file name for the error log.
innodb_buffer_pool_size	Sets the amount of memory InnoDB reserves for the Buffer Pool.
innodb_log_file_size	Sets the size for each Redo Log file and innodb_log_files_in_group sets the number of Redo Log files used by InnoDB.
innodb_io_capacity	Sets the maximum number of I/O operations per second that InnoDB can use.

2. Choose a configuration file in which to configure your system variables and options.

It is not recommended to make custom changes to one of the bundled configuration files. Instead, it is recommended to create a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. If you want your custom configuration file to override the bundled configuration files, then it is a good idea to prefix the custom configuration file's name with a string that will be sorted last, such as `z-`.

- On RHEL, CentOS, and SLES, a good custom configuration file would be: `/etc/my.cnf.d/z-custom-my.cnf`
- On Debian and Ubuntu, a good custom configuration file would be: `/etc/mysql/mariadb.conf.d/z-custom-my.cnf`

3. Set your system variables and options in the configuration file.

They need to be set in a group that will be read by mariadbd, such as [mariadb] or [server].

For example:

```
[mariadb]
log_error                = mariadbd.err
innodb_buffer_pool_size   = 1G
```

Start the Server

MariaDB Enterprise Server includes configuration to start, stop, restart, enable/disable on boot, and check the status of the Server using the operating system default process management system.

For distributions that use systemd (most supported OSes), you can manage the Server process using the systemctl command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Test MariaDB Enterprise Server

When MariaDB Enterprise Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the server using mariadb Client using the `root@localhost` user account:

```
$ sudo mariadb

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Installing Primary/Replica

Overview

Software Version	Diagram	Features
• Enterprise Server 10.4 • Enterprise Server 10.5 • Enterprise Server 10.6 • Enterprise Server 11.4		**MariaDB Replication** • Highly available • Asynchronous or semi-synchronous replication • Automatic failover via MaxScale • Manual provisioning of new nodes from backup • Scales reads via MaxScale • Enterprise Server 10.3+, MaxScale 2.5+

This procedure describes the deployment of the **Primary/Replica topology** with MariaDB Enterprise Server and MariaDB MaxScale.

Primary/Replica topology provides read scalability and fault tolerance through asynchronous or semi-synchronous single-primary replication.

This procedure has 7 steps, which are executed in sequence.

MariaDB products can be deployed in many different topologies to suit specific use cases. The Primary/Replica topology can be deployed on its own or integrated with MariaDB Enterprise Cluster.

This procedure represents basic product capability with 3 Enterprise Server nodes and 1 MaxScale node.

This page provides an overview of the topology, requirements, and deployment procedure.

Please read and understand this procedure before executing.

Procedure Steps

Step	Description
Step 1	Install MariaDB Enterprise Server
Step 2	Start and Configure MariaDB Enterprise Server on Primary Server
Step 3	Start and Configure MariaDB Enterprise Server on Replica Servers
Step 4	Test MariaDB Enterprise Server
Step 5	Install MariaDB MaxScale
Step 6	Start and Configure MariaDB MaxScale
Step 7	Test MariaDB MaxScale

Components

The following components are deployed during this procedure:

Component	Function
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.
MariaDB MaxScale	Database proxy that extends the availability, scalability, and security of MariaDB Enterprise Servers

MariaDB Enterprise Server Components

Component	Description
InnoDB	

MariaDB MaxScale Components

Component	Description
Listener	Listens for client connections to MaxScale, then passes them to the router service associated with the listener
MariaDB Monitor	Tracks changes in the state of MariaDB Enterprise Servers.
Read Connection Router	Routes connections from the listener to any available Enterprise Server node
Read/Write Split Router	Routes read operations from the listener to any available Enterprise Server node, and routes write operations from the listener to a specific server operating as the primary server
Server Module	Connection configuration in MaxScale to an Enterprise Server node

Topology

Primary/Replica topology provides read scalability and fault tolerance through asynchronous or semi-synchronous single-primary replication of MariaDB Enterprise Server 11.4

The Primary/Replica topology consists of:

- 1 or more MaxScale nodes
- 1 Enterprise Server node operating as the primary server
- 2 or more Enterprise Server nodes operating as replica servers.

The MaxScale nodes:

- Monitor the health and availability of the Enterprise Server nodes
- Route queries to Enterprise Server nodes using Read/Write Split (`readwritesplit`) and Read Connection (`readconnroute`) routers.
- Promote replica servers in the event that the primary server fails.

The Enterprise Server node is operating as the primary server:

- Receives write queries from MaxScale, logging them to the Binary Log
- Provides Binary Logs to replica servers for replication

The Enterprise Server nodes are operating as replica servers:

- Receive read queries from MaxScale
- Replicate writes asynchronously or semi-synchronously from the primary server

Requirements

These requirements are for the Primary/Replica topology when deployed with MariaDB Enterprise Server 11.4 and MariaDB MaxScale 25.01.

- Operating System
- System User Accounts

Operating System

In alignment to the enterprise lifecycle, the Primary/Replica topology with MariaDB Enterprise Server 11.4 and MariaDB MaxScale 25.01 is provided for:

- AlmaLinux 8 (x86_64, ARM64)
- AlmaLinux 9 (x86_64, ARM64)
- Debian 11 (x86_64, ARM64)
- Debian 12 (x86_64, ARM64)

- Microsoft Windows (x86_64)
- Red Hat Enterprise Linux 8 (x86_64, ARM64)
- Red Hat Enterprise Linux 9 (x86_64, PPC64LE, ARM64)
- Red Hat UBI 8 (x86_64, ARM64)
- Rocky Linux 8 (x86_64, ARM64)
- Rocky Linux 9 (x86_64, ARM64)
- SUSE Linux Enterprise Server 15 (x86_64, ARM64)
- Ubuntu 20.04 LTS (x86_64, ARM64)
- Ubuntu 22.04 LTS (x86_64, ARM64)
- Ubuntu 24.04 LTS (x86_64, ARM64)

System User Accounts

User Account	Purpose
`maxscale`	MaxScale process owner
`mysql`	Enterprise Server process owner

Quick Reference

- MariaDB Enterprise Server Configuration Management
- MariaDB Enterprise Server Service Management
- MariaDB Enterprise Server Logs
- MaxScale Configuration Management
- MaxScale Service Management

MariaDB Enterprise Server Configuration Management

Method	Description
Configuration File	Configuration files (such as `/etc/my.cnf`) can be used to set system-variables and options. The server must be restarted to apply changes made to configuration files.
Command-line	The server can be started with command-line options that set system-variables and options.
SQL	Users can set system-variables that support dynamic changes on-the-fly using the SET statement.

MariaDB Enterprise Server packages are configured to read configuration files from different paths, depending on the operating system. Making custom changes to Enterprise Server default configuration files is not recommended because custom changes may be overwritten by other default configuration files that are loaded later.

To ensure that your custom changes will be read last, create a custom configuration file with the `z-` prefix in one of the include directories.

Distribution	Example Configuration File Path
• CentOS • Red Hat Enterprise Linux (RHEL) • SUSE Linux Enterprise Server (SLES)	`/etc/my.cnf.d/z-custom-mariadb.cnf`
• Debian • Ubuntu	`/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf`

MariaDB Enterprise Server Service Management

The `systemctl` command is used to start and stop the MariaDB Enterprise Server service.

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

For additional information, see "Starting and Stopping MariaDB".

MariaDB Enterprise Server Logs

MariaDB Enterprise Server produces log data that can be helpful in problem diagnosis.

Log filenames and locations may be overridden in the server configuration. The default location of logs is the data directory. The data directory is specified by the datadir system variable.

Log	System Variable/Option	Default Filename
MariaDB Error Log	log_error	`<hostname>.err`
MariaDB Enterprise Audit Log	server_audit_file_path	`server_audit.log`
Slow Query Log	slow_query_log_file	`<hostname>-slow.log`
General Query Log	general_log_file	`<hostname>.log`
Binary Log	log_bin	`<hostname>-bin`

MaxScale Configuration Management

MaxScale can be configured using several methods. These methods make use of MaxScale's REST API.

Method	Benefits
MaxCtrl	Command-line utility to perform administrative tasks through the REST API. See MaxCtrl Commands.
MaxGUI	MaxGUI is a graphical utility that can perform administrative tasks through the REST API.
REST API	The REST API can be used directly. For example, the `curl` utility could be used to make REST API calls from the command-line. Many programming languages also have libraries to interact with REST APIs.

The procedure on these pages configures MaxScale using MaxCtrl.

MaxScale Service Management

The `systemctl` command is used to start and stop the MaxScale service.

Operation	Command
Start	`sudo systemctl start maxscale`
Stop	`sudo systemctl stop maxscale`
Restart	`sudo systemctl restart maxscale`
Enable during startup	`sudo systemctl enable maxscale`
Disable during startup	`sudo systemctl disable maxscale`

Operation	Command
Status	`sudo systemctl status maxscale`

For additional information, see “Starting and Stopping MariaDB”.

Next Step

Navigation in the procedure “Deploy Primary/Replica Topology”:

- Next: Step 1: Install MariaDB Enterprise Server

This page is: Copyright © 2026 MariaDB. All rights reserved.

Enterprise Server with ColumnStore (Local Storage)

This procedure describes the deployment of the **Single-Node Enterprise ColumnStore** topology with Local storage.

MariaDB Enterprise ColumnStore 23.10 is a columnar storage engine for MariaDB Enterprise Server 10.6. Enterprise ColumnStore is best suited for Online Analytical Processing (OLAP) workloads.

This procedure has 5 steps, which are executed in sequence.

This page provides an overview of the topology, requirements, and deployment procedures.

Please read and understand this procedure before executing.

Procedure Steps

Step	Description
Step 1	Prepare System for Enterprise ColumnStore
Step 2	Install Enterprise ColumnStore
Step 3	Start and Configure Enterprise ColumnStore 23.10
Step 4	Test Enterprise ColumnStore
Step 5	Bulk Import Data to Enterprise ColumnStore

Support

Customers can obtain support by submitting a support case.

Components

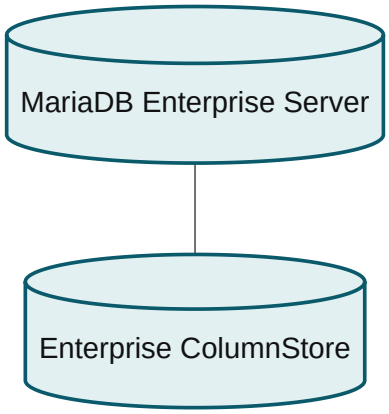
The following components are deployed during this procedure:

Component	Function
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.

MariaDB Enterprise Server Components

Component	Description
MariaDB Enterprise ColumnStore	• Columnar Storage Engine • Optimized for Online Analytical Processing (OLAP) workloads

Topology



A single Enterprise Server node runs ColumnStore on local storage.

The Single-Node Enterprise ColumnStore topology provides support for Online Analytical Processing (OLAP) workloads to MariaDB Enterprise Server.

The Enterprise ColumnStore node:

- Receives queries from the application
- Executes queries
- Uses the local disk for storage.

High Availability

Single-Node Enterprise ColumnStore does not provide high availability (HA) for Online Analytical Processing (OLAP). If you would like to deploy Enterprise ColumnStore with high availability, see Enterprise ColumnStore with Shared Local storage.

Requirements

These requirements are for the Single-Node Enterprise ColumnStore, when deployed with MariaDB Enterprise Server 10.6 and MariaDB Enterprise ColumnStore 23.10.

Operating System

- Debian 11 (x86_64, ARM64)
- Debian 12 (x86_64, ARM64)
- Red Hat Enterprise Linux 8 (x86_64, ARM64)
- Red Hat Enterprise Linux 9 (x86_64, ARM64)
- Rocky Linux 8 (x86_64, ARM64)
- Rocky Linux 9 (x86_64, ARM64)
- Ubuntu 20.04 LTS (x86_64, ARM64)
- Ubuntu 22.04 LTS (x86_64, ARM64)
- Ubuntu 24.04 LTS (x86_64, ARM64)

Minimum Hardware Requirements

MariaDB Enterprise ColumnStore's minimum hardware requirements are not intended for production environments, but the minimum hardware requirements can be appropriate for development and test environments. For production environments, see the recommended hardware requirements instead.

The minimum hardware requirements are:

Component	CPU	Memory
Enterprise ColumnStore node	4+ cores	16+ GB

MariaDB Enterprise ColumnStore will refuse to start if the system has less than 3 GB of memory.

If Enterprise ColumnStore is started on a system with less memory, the following error message will be written to the ColumnStore system log called `crit.log`:

```
Apr 30 21:54:35 a1ebc96a2519 PrimProc[1004]: 35.668435 |0|0|0| C 28 CAL0000: Error total memory available
is less than 3GB.
```

And the following error message will be raised to the client:

```
ERROR 1815 (HY000): Internal error: System is not ready yet. Please try again.
```

Recommended Hardware Requirements

MariaDB Enterprise ColumnStore's recommended hardware requirements are intended for production analytics.

The recommended hardware requirements are:

Component	CPU	Memory
Enterprise ColumnStore node	64+ cores	128+ GB

Quick Reference

MariaDB Enterprise Server Configuration Management

Method	Description
Configuration File	Configuration files (such as `/etc/my.cnf`) can be used to set system variables and options. The server must be restarted to apply changes made to configuration files.
Command-line	The server can be started with command-line options that set system variables and options.
SQL	Users can set system variables that support dynamic changes on-the-fly using the SET statement.

MariaDB Enterprise Server packages are configured to read configuration files from different paths, depending on the operating system. Making custom changes to Enterprise Server default configuration files is not recommended because custom changes may be overwritten by other default configuration files that are loaded later.

To ensure that your custom changes will be read last, create a custom configuration file with the `z-` prefix in one of the include directories.

Distribution	Example Configuration File Path
• CentOS • Red Hat Enterprise Linux (RHEL)	`/etc/my.cnf.d/z-custom-mariadb.cnf`
• Debian • Ubuntu	`/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf`

MariaDB Enterprise Server Service Management

The `systemctl` command is used to start and stop the MariaDB Enterprise Server service.

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Local storage deployment procedure:

- Next: Step 1: Install MariaDB Enterprise ColumnStore 23.10.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Enterprise Server with ColumnStore (Object Storage)

Overview

This procedure describes the deployment of the Single-Node Enterprise ColumnStore topology with Object storage.

MariaDB Enterprise ColumnStore 23.10 is a columnar storage engine for MariaDB Enterprise Server and Enterprise ColumnStore is best suited for Online Analytical Processing (OLAP) workloads.

This procedure has 5 steps, which are executed in sequence.

This page provides an overview of the topology, requirements, and deployment procedures.

Please read and understand this procedure before executing.

Procedure Steps

Step	Description
Step 1	Prepare System for Enterprise ColumnStore
Step 2	Install Enterprise ColumnStore
Step 3	Start and Configure Enterprise ColumnStore
Step 4	Test Enterprise ColumnStore
Step 5	Bulk Import Data to Enterprise ColumnStore

Support

Customers can obtain support by submitting a support case.

Components

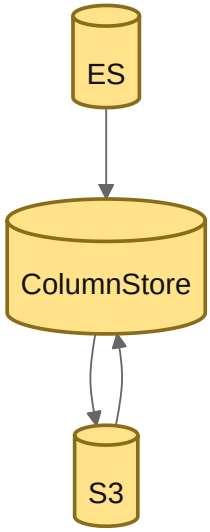
The following components are deployed during this procedure:

Component	Function
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.

MariaDB Enterprise Server Components

Component	Description
MariaDB Enterprise ColumnStore	• Columnar Storage Engine • Optimized for Online Analytical Processing (OLAP) workloads • S3-compatible object storage

Topology



A single MariaDB Enterprise Server node running the Enterprise ColumnStore engine, which reads and writes data to and from S3-compatible object storage.

The Single-Node Enterprise ColumnStore topology provides support for Online Analytical Processing (OLAP) workloads to MariaDB Enterprise Server.

The Enterprise ColumnStore node:

- Receives queries from the application
- Executes queries
- Use S3-compatible object storage for data

High Availability

Single-Node Enterprise ColumnStore does not provide high availability (HA) for Online Analytical Processing (OLAP). If you would like to deploy Enterprise ColumnStore with high availability, see Enterprise ColumnStore with Object storage.

Requirements

These requirements are for the Single-Node Enterprise ColumnStore, when deployed with MariaDB Enterprise Server and MariaDB Enterprise ColumnStore.

Operating System

- Debian 11 (x86_64, ARM64)
- Debian 12 (x86_64, ARM64)
- Red Hat Enterprise Linux 8 (x86_64, ARM64)
- Red Hat Enterprise Linux 9 (x86_64, PPC64LE, ARM64)

- Red Hat UBI 8 (x86_64, ARM64)
- Rocky Linux 8 (x86_64, ARM64)
- Rocky Linux 9 (x86_64, ARM64)
- Ubuntu 20.04 LTS (x86_64, ARM64)
- Ubuntu 22.04 LTS (x86_64, ARM64)
- Ubuntu 24.04 LTS (x86_64, ARM64)

Minimum Hardware Requirements

MariaDB Enterprise ColumnStore's minimum hardware requirements are not intended for production environments, but the minimum hardware requirements can be appropriate for development and test environments. For production environments, see the recommended hardware requirements instead.

The minimum hardware requirements are:

Component	CPU	Memory
Enterprise ColumnStore node	4+ cores	16+ GB

MariaDB Enterprise ColumnStore will refuse to start if the system has less than 3 GB of memory.

If Enterprise ColumnStore is started on a system with less memory, the following error message will be written to the ColumnStore system log called `crit.log`:

```
Apr 30 21:54:35 a1ebc96a2519 PrimProc[1004]: 35.668435 |0|0|0| C 28 CAL0000: Error total memory available
is less than 3GB.
```

And the following error message will be raised to the client:

```
ERROR 1815 (HY000): Internal error: System is not ready yet. Please try again.
```

Recommended Hardware Requirements

MariaDB Enterprise ColumnStore's recommended hardware requirements are intended for production analytics.

The recommended hardware requirements are:

Component	CPU	Memory
Enterprise ColumnStore node	64+ cores	128+ GB

Storage Requirements

Single-node Enterprise ColumnStore with Object Storage requires the following storage type:

Storage Type	Description
S3-Compatible Object Storage	Single-node Enterprise ColumnStore with Object Storage uses S3-compatible object storage to store data.

S3-Compatible Object Storage Requirements

Single-node Enterprise ColumnStore with Object Storage uses S3-compatible object storage to store data.

Many S3-compatible object storage services exist. MariaDB Corporation cannot make guarantees about all S3-compatible object storage services, because different services provide different functionality.

For the preferred S3-compatible object storage providers that provide cloud and hardware solutions, see the following sections:

- Cloud
- Hardware

The use of non-cloud and non-hardware providers is at your own risk.

If you have any questions about using specific S3-compatible object storage with MariaDB Enterprise ColumnStore, contact us.

Preferred Object Storage Providers: Cloud

- Amazon Web Services (AWS) S3
- Google Cloud Storage
- Azure Storage
- Alibaba Cloud Object Storage Service

Preferred Object Storage Providers: Hardware

- Cloudian HyperStore
- Dell EMC
- Seagate Lyve Rack
- Quantum ActiveScale
- IBM Cloud Object Storage

Quick Reference

MariaDB Enterprise Server Configuration Management

Method	Description
Configuration File	Configuration files (such as `/etc/my.cnf`) can be used to set system variables and options. The server must be restarted to apply changes made to configuration files.
Command-line	The server can be started with command-line options that set system variables and options.
SQL	Users can set system variables that support dynamic changes on-the-fly using the SET statement.

MariaDB Enterprise Server packages are configured to read configuration files from different paths, depending on the operating system. Making custom changes to Enterprise Server default configuration files is not recommended because custom changes may be overwritten by other default configuration files that are loaded later.

To ensure that your custom changes will be read last, create a custom configuration file with the `z-` prefix in one of the include directories.

Distribution	Example Configuration File Path
• CentOS • Red Hat Enterprise Linux (RHEL)	`/etc/my.cnf.d/z-custom-mariadb.cnf`
• Debian • Ubuntu	`/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf`

MariaDB Enterprise Server Service Management

The `systemctl` command is used to start and stop the MariaDB Enterprise Server service.

Operation	Command
Start	`sudo systemctl start mariadb`

Operation	Command
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Object storage deployment procedure:

Next: Step 1: Install MariaDB Enterprise ColumnStore.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Installing Community Server

See MariaDB ID Sign Up.

Information on creating a free MariaDB ID account to access software downloads, including binary packages and repositories.

See MariaDB Binary Packages.

Provides details on using MariaDB binary packages (tarballs, RPMs, DEBs) for installation, including repository configuration scripts.

See Installing System Tables.

Explains the necessity of initializing system tables (using mariadb-install-db) after installation and troubleshooting startup issues related to missing tables.

See Troubleshooting Installation Issues.

A guide to diagnosing and resolving common installation and connection problems, such as socket errors, permission denied messages, and configuration conflicts.

See Installing MariaDB on IBM Cloud.

Step-by-step instructions for deploying MariaDB on IBM Cloud Kubernetes Service, including provisioning clusters and configuring storage.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB ID Sign Up

A MariaDB ID is a free account that provides:

- Customer access to MariaDB database product downloads

Create A MariaDB ID

You can register for a MariaDB ID with a social login or a traditional email/password.

Register for a MariaDB ID account.

Create a MariaDB ID using one of the following Single sign-on methods:

- Sign up using your Google Account
- Sign up using your GitHub Account

- Sign up using your LinkedIn Account

Or use your email address:

- Enter your email address and press next.
- Click the reset button. A password link will appear in your email.
- Once you set a new password, the MariaDB account page will appear.

Obtaining Support

MariaDB Corporation provides commercial support and services for MariaDB database products.

To contact us with questions, or if you need assistance:

- If you are able to login to your MariaDB ID, open a support case
- If you do not have a MariaDB ID or are unable to login, contact us
- MariaDB Customer Portal

\

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Binary Packages

See GPG.

Information about the GPG keys used to sign MariaDB packages and repositories, including how to import them for verification.

See Installing MariaDB .deb Files.

Complete .deb installation guide: add repo via mariadb_repo_setup, import GPG keys, apt install mariadb-server galera-4, and APT configuration.

See Installing MariaDB RPM Files.

Install and manage MariaDB Server using RPM packages. This section provides detailed instructions for deploying and upgrading MariaDB on RPM-based Linux distributions. width: default title: visible: true description: visible: true tableOfContents: visible: true outline: visible: true pagination: visible: false metadata: visible: true tags: visible: true

See Installing MariaDB Alongside MySQL.

Instructions for installing MariaDB on the same server as an existing MySQL installation, useful for migration testing or running multiple versions.

See Installing MariaDB Binary Tarballs.

A guide to installing MariaDB from pre-compiled binary tarballs on Linux, allowing for flexible installation paths and multiple versions.

See Installing MariaDB MSI Packages on Windows.

Complete MariaDB installation guide. Complete setup instructions for Linux, Windows, and macOS with configuration and verification for production use.

See Installing MariaDB on macOS.

How to install MariaDB Server on macOS using the Homebrew package manager, including starting the service and securing the installation.

See Installing MariaDB on macOS (PKG).

How to install MariaDB Server on macOS. This is possible using Homebrew.

See Installing MariaDB Windows ZIP Packages.

Instructions for manually installing MariaDB on Windows from a ZIP archive, useful for portable installations or advanced configuration needs.

See Deploy with Package Tarballs.

Explains the benefits and use cases for deploying MariaDB using package tarballs (containing RPMs or DEBs) for offline or custom installations.

See Deploy with Local Package Repository Mirrors.

Learn how to create and maintain local mirrors of MariaDB package repositories for secure or air-gapped deployments.

This page is licensed: CC BY-SA / Gnu FDL

GPG

The MariaDB project signs their MariaDB packages for Debian, Ubuntu, Fedora, and Red Hat. This page documents information about the keys that we use and have used.

MariaDB Community Server Debian / Ubuntu key

Our MariaDB Community Server repositories for Debian “Sid” and the Ubuntu 16.04 and beyond “Xenial” use the following GPG signing key. As detailed in MDEV-9781, APT 1.2.7 (and later) prefers SHA2 GPG keys and now prints warnings when a repository is signed using a SHA1 key like our previous GPG key. We have created a SHA2 key for use with these.

Information about this key:

- The short Key ID is: `0xC74CD1D8`
- The long Key ID is: `0xF1656F24C74CD1D8`
- The full fingerprint of the key is: `177F 4010 FE56 CA33 3630 0305 F165 6F24 C74C D1D8`
- The key can be added on Debian-based systems using the following command:

```
sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xF1656F24C74CD1D8
```

- Usage of the `apt-key` command is deprecated in the latest versions of Debian and Ubuntu, and the replacement method is to download the keyring file to the `/etc/apt/trusted.gpg.d/` directory. This can be done with the following:

```
sudo curl -LsSo /etc/apt/trusted.gpg.d/mariadb-keyring-2019.gpg https://supplychain.mariadb.com/mariadb-keyring-2019.gpg
```

Prior to Debian 9 (Stretch), and Debian Unstable (Sid), and Ubuntu 16.04 LTS (Xenial), the id of our GPG public key was `0xcbcb082a1bb943db`. The full key fingerprint is:

```
1993 69E5 404B D5FC 7D2F E43B CBCB 082A 1BB9 43DB
```

MariaDB Community Server RPM / Source Keys

> **2023+**
>
> Beginning in 2023 we migrated the key used to sign our `yum/dnf/zypper` repositories and to sign our source code and binary tarballs to the same key we use for Debian and Ubuntu.
>
> Key ID and Fingerprint information is in the MariaDB Community Server Debian/Ubuntu Key section, above.

The key can be imported on RPM-based systems using the following command:

```
sudo rpm --import https://supplychain.mariadb.com/MariaDB-Server-GPG-KEY
```

or

```
sudo rpmkeys --import https://supplychain.mariadb.com/MariaDB-Server-GPG-KEY
```

PRE-2023

The GPG Key ID of the MariaDB signing key we used for `yum/dnf/zypper` repositories and to sign our source code tarballs until the end of 2022 was `0xCBCB082A1BB943DB`. The short form of the id is `0x1BB943DB` and the full key fingerprint is:

```
1993 69E5 404B D5FC 7D2F E43B CBCB 082A 1BB9 43DB
```

This key was used by the `yum/dnf/zypper` repositories for RedHat, CentOS, Fedora, openSUSE, and SLES.

If you configure the mariadb.org rpm repositories using the repository configuration tool then your package manager will prompt you to import the key the first time you install a package from the repository.

You can also import the key directly using the following command:

```
sudo rpmkeys --import https://supplychain.mariadb.com/MariaDB-Server-GPG-KEY-2010
```

MariaDB Enterprise GPG Keys

RHEL 7/8/9, DEBIAN 11/12, UBUNTU

Information about the key we use on most platforms for MariaDB Enterprise Server releases:

- The short Key ID is: `0xE3C94F49`
- The long Key ID is: `0xCE1A3DD5E3C94F49`
- The full fingerprint of the key is: `4C47 0FFF EFC4 D3DC 5977 8655 CE1A 3DD5 E3C9 4F49`

On Debian / Ubuntu systems, you can install the keyring containing this key with:

```
sudo curl -LsSo /etc/apt/trusted.gpg.d/mariadb-keyring-2019.gpg https://supplychain.mariadb.com/mariadb-keyring-2019.gpg
```

The key can be imported on RPM-based systems using the following command:

```
sudo rpm --import https://supplychain.mariadb.com/MariaDB-Enterprise-GPG-KEY
```

or

```
sudo rpmkeys --import https://supplychain.mariadb.com/MariaDB-Enterprise-GPG-KEY
```

RHEL 10+ , DEBIAN 13+

For RHEL, AlmaLinux, Rocky, and Oracle Linux 10 and above, and for Debian 13 Trixie and above there are new requirements around GPG keys used to sign packages. For these distributions we have created a stronger GPG key for our Enterprise Server releases.

Information on this key:

- The short Key ID is: `0x8C27D14E`

- The long Key ID is: `0x5D87FACA8C27D14E`
- The full fingerprint of the key is: `BB2A 36F3 6C3B 4D37 3BAC 328A 5D87 FACA 8C27 D14E`

On Debian / Ubuntu systems, you can install the keyring containing this key with:

```
sudo curl -LsSo /etc/apt/trusted.gpg.d/mariadb-keyring-2025.gpg https://supplychain.mariadb.com/mariadb-keyring-2025.gpg
```

The key can be imported on RPM-based systems using the following command:

```
sudo rpm --import https://supplychain.mariadb.com/MariaDB-Enterprise-GPG-KEY-2025
```

or

```
sudo rpmkeys --import https://supplychain.mariadb.com/MariaDB-Enterprise-GPG-KEY-2025
```

MariaDB MaxScale GPG Keys

DEC 2025+

In December 2025, MariaDB MaxScale moved to using the same GPG key used by MariaDB Enterprise Server for RHEL 10 and Debian 13 Trixie.

Information on this key:

- The short Key ID is: `0x8C27D14E`
- The long Key ID is: `0x5D87FACA8C27D14E`
- The full fingerprint of the key is: `BB2A 36F3 6C3B 4D37 3BAC 328A 5D87 FACA 8C27 D14E`

On Debian / Ubuntu systems, you can install the keyring containing this key with:

```
sudo curl -LsSo /etc/apt/trusted.gpg.d/mariadb-keyring-2025.gpg https://supplychain.mariadb.com/mariadb-keyring-2025.gpg
```

The key can be imported on RPM-based systems using the following command:

```
sudo rpm --import https://supplychain.mariadb.com/MariaDB-Enterprise-GPG-KEY-2025
```

or

```
sudo rpmkeys --import https://supplychain.mariadb.com/MariaDB-Enterprise-GPG-KEY-2025
```

PRE DEC 2025

Information on the key used by MaxScale repositories prior to December 2025:

- The short Key ID is: `0xE3C94F49`
- The long Key ID is: `0xCE1A3DD5E3C94F49`
- The full fingerprint of the key is: `4C47 0FFF EFC4 D3DC 5977 8655 CE1A 3DD5 E3C9 4F49`

On Debian / Ubuntu systems, you can install the keyring containing this key with:

```
sudo curl -LsSo /etc/apt/trusted.gpg.d/mariadb-keyring-2025.gpg https://supplychain.mariadb.com/mariadb-keyring-2025.gpg
```

The key can be imported on RPM-based systems using the following command:

```
sudo rpm --import https://supplychain.mariadb.com/MariaDB-MaxScale-GPG-KEY
```

or

```
sudo rpmkeys --import https://supplychain.mariadb.com/MariaDB-MaxScale-GPG-KEY
```

Configuring Repositories

See the MariaDB Package Repository Setup and Usage page for details on using the `mariadb_repo_setup` and `mariadb_es_repo_setup` scripts to configure repositories that use these keys.

See the details on configuring MariaDB Foundation repositories that use these keys.

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB .deb Files

Installing MariaDB with APT

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant `.deb` packages from MariaDB's repository using apt, aptitude, Ubuntu Software Center, Synaptic Package Manager, or another package manager.

This page walks you through the simple installation steps using `apt`.

Adding the MariaDB APT repository

We currently have APT repositories for the following Linux distributions:

- Debian 11 (Bullseye)
- Debian 12 (Bookworm)
- Debian 13 (Trixie)
- Debian Unstable (Sid)
- Ubuntu 22.04 (Jammy)
- Ubuntu 24.04 (Noble)

Using the MariaDB Package Repository Setup Script

If you want to install MariaDB with `apt`, then you can configure `apt` to install from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script.

MariaDB Corporation provides a MariaDB Package Repository for several Linux distributions that use `apt` to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, clients and utilities, client libraries, plugins, and mariadb-backup. The MariaDB Package Repository setup script automatically configures your system to install packages from the MariaDB Package Repository.

To use the script, execute the following command:

```
curl -sS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
```

Note that this script also configures a repository for MariaDB MaxScale and a repository for MariaDB Tools, which currently only contains Percona XtraBackup and its dependencies.

See MariaDB Package Repository Setup and Usage for more information.

Using the MariaDB Repository Configuration Tool

If you want to install MariaDB with `apt`, then you can configure `apt` to install from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool.

The MariaDB Foundation provides a MariaDB repository for several Linux distributions that use `apt-get` to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, clients and utilities, client libraries, plugins, and mariadb-backup. The MariaDB Repository Configuration Tool can easily generate the appropriate commands to add the repository for your distribution.

There are several ways to add the repository.

Executing add-apt-repository

One way to add an `apt` repository is by using the add-apt-repository command. This command will add the repository configuration to `/etc/apt/sources.list`.

For example, if you wanted to use the repository to install MariaDB 10.6 on Ubuntu 18.04 LTS (Bionic), then you could use the following commands to add the MariaDB `apt` repository:

```
sudo apt-get install software-properties-common
sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el]
http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.6/ubuntu bionic main'
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

Creating a Source List File

Another way to add an `apt` repository is by creating a source list file in `/etc/apt/sources.list.d/`.

For example, if you wanted to use the repository to install MariaDB 10.6 on Ubuntu 18.04 LTS (Bionic), then you could create the `MariaDB.list` file in `/etc/apt/sources.list.d/` with the following contents to add the MariaDB `apt` repository:

```
# MariaDB 10.6 repository list - created 2019-01-27 09:50 UTC
# http://downloads.mariadb.org/mariadb/repositories/
deb [arch=amd64,arm64,ppc64el] http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.6/ubuntu bionic main
deb-src http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.6/ubuntu bionic main
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

Using Ubuntu Software Center

Another way to add an `apt` repository is by using Ubuntu Software Center.

You can do this by going to the **Software Sources** window. This window can be opened either by navigating to **Edit** > **Software Sources** or by navigating to **System** > **Administration** > **Software Sources**.

Once the **Software Sources** window is open, go to the **Other Software** tab, and click the **Add** button. At that point, you can input the repository information provided by the MariaDB Repository Configuration Tool.

See here for more information.

Using Synaptic Package Manager

Another way to add an `apt` repository is by using Synaptic Package Manager.

You can do this by going to the **Software Sources** window. This window can be opened either by navigating to **System** > **Administrator** > **Software Sources** or by navigating to **Settings** > **Repositories**.

Once the **Software Sources** window is open, go to the **Other Software** tab, and click the **Add** button. At that point, you can input the repository information provided by the MariaDB Repository Configuration Tool.

See here for more information.

Pinning the MariaDB Repository to the repository of a Specific Minor Release

If you wish to pin the `apt` repository to a specific minor release, or if you would like to downgrade to a specific minor release, then you can create an `apt` repository with the URL hard-coded to that specific minor release.

MARIADB CORPORATION REPO SETUP SCRIPTS

If you used MariaDB Corporation's `mariadb_repo_setup` or `mariadb_es_repo_setup` scripts to generate your repository configuration, simply re-run the script and specify the full version number to use with the `--mariadb-server-version` option.

See Pinning the Repository to a Specific Minor Release on the MariaDB Package Repository Setup and Usage page for details.

The full list of MariaDB Enterprise Server releases can be found on the Enterprise Server - All Releases page.

MARIADB FOUNDATION REPO CONFIG TOOL

If you used the MariaDB Foundation's Repository Configuration tool, then you need to update the repository file you created to include the full version number. By default the Foundation's tool configures repositories with just the main series of MariaDB, e.g. `mariadb-11.8`, and to pin to a specific version you need to specify the full version, for example `mariadb-11.8.6`. The full list of MariaDB Community Server releases can be found on the Community Server - All Releases page.

Archives are only of the distros and architectures supported at the time of release. For example, MariaDB Community Server 10.6.21 exists for Ubuntu `bionic`, `focal`, `jammy`, and `kinetic`, and the list of what distributions are available is obtained by looking in the `dists` folder of the 10.6.21 Debian or Ubuntu repositories.

For example, if you wanted to pin your repository to MariaDB Community Server 10.5.9 on Ubuntu 20.04 LTS (Focal), then you would have to first remove any existing MariaDB repository source list file from `/etc/apt/sources.list.d/`. And then you could use the following commands to add the MariaDB `apt-get` repository:

```
sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el,s390x] http://archive.mariadb.org/mariadb-10.5.9/repo/ubuntu/ focal main main/debug'
```

Ensure you have the signing key installed.

Ubuntu Xenial and older will need:

```
sudo apt-get install -y apt-transport-https
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

Updating the MariaDB APT repository to a New Major Release

MariaDB's `apt` repository can be updated to a new major release. How this is done depends on how you originally configured the repository.

Updating the Major Release with the MariaDB Package Repository Setup Script

If you configured `apt` to install from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script, then you can update the major release that the repository uses by running the script again.

Updating the Major Release with the MariaDB Foundation's Repository Configuration Tool

If you configured `apt` to install from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool, then you can update the major release in various ways, depending on how you originally added the repository.

Updating a Repository with add-apt-repository

If you added the `apt` repository by using the add-apt-repository command, then you can update the major release that the repository uses by using the add-apt-repository command again.

First, look for the repository string for the old version in `/etc/apt/sources.list`.

And then, you can remove the repository for the old version by executing the add-apt-repository command and providing the `--remove` option. For example, if you wanted to remove a MariaDB 10.6 repository, then you could do so by executing something like the following:

```
sudo add-apt-repository --remove 'deb [arch=amd64,arm64,ppc64el]
http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.6/ubuntu bionic main'
```

After that, you can add the repository for the new version with the add-apt-repository command. For example, if you wanted to use the repository to install MariaDB 10.6 on Ubuntu 18.04 LTS (Bionic), then you could use the following commands to add the MariaDB `apt` repository:

```
sudo apt-get install software-properties-common
sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el]
http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.6/ubuntu bionic main'
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

After that, the repository should refer to MariaDB 10.6 and you can upgrade to the new version of MariaDB.

Updating a Source List File

If you added the `apt` repository by creating a source list file in `/etc/apt/sources.list.d/`, then you can update the major release that the repository uses by updating the source list file in-place. For example, if you wanted to change the repository from MariaDB 10.5 to MariaDB 10.6, and if the source list file was at `/etc/apt/sources.list.d/MariaDB.list`, then you could execute the following:

```
sudo sed -i 's/10.5/10.6/' /etc/apt/sources.list.d/MariaDB.list
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

After that, the repository should refer to MariaDB 10.6 and you can upgrade to the new version of MariaDB.

Importing the MariaDB GPG Public Key

Before MariaDB can be installed, you also have to import the GPG public key that is used to verify the digital signatures of the packages in our repositories. This allows the `apt` utility to verify the integrity of the packages that

it installs.

For **MariaDB Community Server**, see the MariaDB Community Server Debian / Ubuntu key section of the GPG page for details on how to import the key used by those repositories on your Debian or Ubuntu system.

For **MariaDB Enterprise Server**, see the MariaDB Enterprise GPG Keys section of the GPG page for details on how to import the key used by those repositories on your Debian or Ubuntu system.

> **Note:**
>
> Starting with Debian 9 (Stretch), the dirmngr package needs to be installed before the GPG public key can be imported. To install it, execute: `sudo apt install dirmngr`

Once the GPG public key is imported, you are ready to install packages from the repository.

Installing MariaDB Packages with APT

After the `apt` repository is configured, you can install MariaDB by executing the apt-get command. The specific command that you would use would depend on which specific packages that you want to install.

Installing the Most Common Packages with APT

To Install the most common packages, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

To Install the most common packages, execute the following command:

```
sudo apt-get install mariadb-server mariadb-server-galera galera-4 mariadb-client libmariadb3 mariadb-backup mariadb-common
```

Installing MariaDB Server with APT

To Install MariaDB Server, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

Then, execute the following command:

```
sudo apt-get install mariadb-server
```

Installing MariaDB Galera Cluster with APT

The process to install MariaDB Galera Cluster with the MariaDB `apt-get` repository is practically the same as installing standard MariaDB Server.

Galera Cluster support is included in the standard MariaDB Server packages, so you will need to install the `mariadb-server` package, as you normally would.

You also need to install the `galera-4` package to obtain the Galera 4 wsrep provider library.

To install MariaDB Galera Cluster, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

To install MariaDB Galera Cluster, you could execute the following command:

```
sudo apt-get install mariadb-server mariadb-client galera-4
```

MariaDB Galera Cluster also has a separate package that can be installed on arbitrator nodes. The package is called `galera-arbitrator-4`. This package should be installed on whatever node you want to serve as the arbitrator. It can either run on a separate server that is not acting as a cluster node, which is the recommended configuration, or it can run on a server that is also acting as an existing cluster node.

To install the arbitrator package, you could execute the following command:

```
sudo apt install galera-arbitrator-4
```

See Galera Cluster for more information on MariaDB Galera Cluster.

Installing MariaDB Clients and Client Libraries with APT

MariaDB Connector/C is included as the client library.

To Install the clients and client libraries, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

Then, execute the following command:

```
sudo apt-get install mariadb-client libmariadb3
```

Installing mariadb-backup with APT

To install mariadb-backup, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

Then, execute the following command:

```
sudo apt-get install mariadb-backup
```

Installing Plugins with APT

Some plugins may also need to be installed.

For example, to install the cracklib_password_check password validation plugin, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

Then, execute the following command:

```
sudo apt-get install mariadb-cracklib-password-check
```

Installing Older Versions from the Repository

The MariaDB `apt` repository contains the last few versions of MariaDB. To show what versions are available, use the apt-cache command:

```
sudo apt-cache showpkg mariadb-server
```

There will be a lot of output, but in the “Provides” section at the end of the output you will see the available versions. For example:

```
Package: mariadb-server
Versions:
1:12.1.2+maria~ubu2404 (/var/lib/apt/lists/dlm.mariadb.com_repo_mariadb-
server_12.rolling_repo_ubuntu_dists_noble_main_binary-amd64_Packages)
 Description Language:
 ...<extra-output-snipped>...
 Provides:
1:12.1.2+maria~ubu2404 - virtual-mysql-server (= ) default-mysql-server (= )
1:12.0.2+maria~ubu2404 - virtual-mysql-server (= ) default-mysql-server (= )
1:10.11.13-0ubuntu0.24.04.1 - virtual-mysql-server (= )
1:10.11.7-2ubuntu2 - virtual-mysql-server (= )
```

In the above example there are four versions available, two from the MariaDB repositories, and two from the Ubuntu repositories. The versions from MariaDB have “maria” in the version number, and the versions from Ubuntu have “ubuntu” in the version number.

To install an older version of a package instead of the latest version we just need to specify the package name, an equal sign, and then the complete version number. From the example above, the complete version number for MariaDB 12.0.2 is: `1:12.0.2+maria~ubu2404`

However, when installing an older version of a package, `apt` will automatically choose to install the latest versions of any dependencies, which doesn’t work for dependencies of the `mariadb-server` package like `mariadb-client` and `mariadb-server-core`.

```
sudo apt install mariadb-server=1:12.0.2+maria~ubu2404
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 mariadb-client : Breaks: mariadb-server (< 1:12.1.2+maria~ubu2404) but 1:12.0.2+maria~ubu2404 is to be
installed
 mariadb-server-core : Breaks: mariadb-server (< 1:12.1.2+maria~ubu2404) but 1:12.0.2+maria~ubu2404 is to
be installed
E: Unable to correct problems, you have held broken packages.
```

To ensure that all MariaDB packages are on the same version in this scenario, it is necessary to specify them all. Therefore, to install the 12.0.2 version of the `mariadb-server` package from this `apt` repository, we would do the following (using a variable to hold the version number, and putting each package on its own line so things are cleaner):

```
ver="1:12.0.2+maria~ubu2404"
sudo apt install \
  mariadb-server=${ver} \
  mariadb-client=${ver} \
  mariadb-server-core=${ver} \
  mariadb-client-core=${ver}
```

For MariaDB Enterprise, version numbers are similar, but have an extra point. For example, MariaDB Enterprise Server 11.8.5-2 for Ubuntu 24.04 Noble has the version number: `1:11.8.5.2+maria~ubu2404` .

The rest of the install and setup process is as normal.

Installing MariaDB with dpkg

While it is not recommended, it is possible to download and install the `.deb` packages manually. However, it is generally recommended to use a package manager like `apt-get`.

A tarball that contains the `.deb` packages can be downloaded from the following URL:

- downloads.mariadb.com

For example, to install the MariaDB 10.6.21 `.deb` packages on Ubuntu 18.04 LTS (Bionic), you could execute the following:

```
sudo apt-get update
sudo apt-get install libdbi-perl libdbd-mysql-perl psmisc libaio1 socat
wget https://downloads.mariadb.com/MariaDB/mariadb-10.6.21/repo/ubuntu/mariadb-10.6.21-ubuntu-bionic-
amd64-debs.tar
tar -xvf mariadb-10.6.21-ubuntu-bionic-amd64-debs.tar
cd mariadb-10.6.21-ubuntu-bionic-amd64-debs/
sudo dpkg --install ./mariadb-common*.deb \
   ./mysql-common*.deb \
   ./mariadb-client*.deb \
   ./libmariadb3*.deb \
   ./libmysqlclient18*.deb
sudo dpkg --install ./mariadb-server*.deb \
   ./mariadb-backup*.deb \
   ./galera-4*.deb
```

After Installation

After the installation is complete, you can start MariaDB.

If you are using MariaDB Galera Cluster, then keep in mind that the first node will have to be bootstrapped.

Upgrading to a new version of MariaDB

After updating your repository configuration to move from the repository of one version of MariaDB to a newer version, as per the previous instructions, you will need to upgrade the MariaDB packages, this is done with:

```
sudo apt upgrade mariadb-server
```

Alternatively you can also run the following to installing the new version:

```
sudo apt install mariadb-server galera-4
```

The reason for specifying the `galera-4` package is to ensure the correct updated version of Galera is installed along with the new server version, replacing the versions that were there before.

Available DEB Packages

The available DEB packages depend on the specific MariaDB release series.

Available DEB Packages

For MariaDB, the following DEBs are available:

Package Name	Description
`galera-4`	The WSREP provider for Galera 4.
`libmariadb3`	Dynamic client libraries.
`libmariadb-dev`	Development headers and static libraries.
`libmariadbclient18`	Virtual package to satisfy external depends

Package Name	Description
`libmysqlclient18`	Virtual package to satisfy external depends
`mariadb-backup`	mariadb-backup
`mariadb-client`	Client tools like mariadb CLI, mariadb-dump, and others.
`mariadb-client-core`	Core client tools
`mariadb-common`	Character set files and /etc/my.cnf
`mariadb-plugin-connect`	The CONNECT storage engine.
`mariadb-plugin-cracklib-password-check`	The cracklib_password_check password validation plugin.
`mariadb-plugin-gssapi-client`	The client-side component of the gssapi authentication plugin.
`mariadb-plugin-gssapi-server`	The server-side component of the gssapi authentication plugin.
`mariadb-plugin-rocksdb`	The MyRocks storage engine.
`mariadb-plugin-spider`	The SPIDER storage engine.
`mariadb-server`	The server and server tools, like myisamchk and mariadb-hotcopy are here.
`mariadb-server-galera`	Required for MariaDB Cluster (Galera) functionality. Includes cluster-specific scripts, systemd bootstrap logic, and the `wsrep_info` plugin.
`mariadb-server-core`	The core server.
`mariadb-test`	mysql-client-test executable, and mysql-test framework with the tests.
`mariadb-test-data`	MariaDB database regression test suite - data files

Actions Performed by DEB Packages

Users and Groups Created

When the `mariadb-server` DEB package is installed, it will create a user and group named `mysql`, if they do not already exist.

See Also

- Differences in MariaDB in Debian
- Installing MariaDB .deb Files with Ansible

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB RPM Files

See About the MariaDB RPM Files.

Provides an overview of the RPM packages available for MariaDB, listing the various RPMs such as server, client, backup, and shared libraries, and explaining their contents and dependencies.

See Installing MariaDB with yum/dnf.

How to install MariaDB on systems that use the yum or dnf package managers

See Installing MariaDB with zypper.

Detailed steps for installing MariaDB on SLES and OpenSUSE using the `zypper` package manager, including repository configuration and package installation.

See Checking MariaDB RPM Package Signatures.

Instructions on how to verify the integrity of MariaDB RPM packages using GPG signatures, including importing the public key and running `rpm --checksig`.

See Installing MariaDB With the rpm Tool.

A guide to installing MariaDB using the low-level `rpm` command, suitable for situations where package managers like `yum` or `dnf` are not available or preferred.

See MariaDB for DirectAdmin Using RPMs.

Specific instructions for installing MariaDB RPMs on servers running the DirectAdmin control panel, including necessary configuration edits to prevent conflicts.

See MariaDB Installation via RPMs on CentOS 7.

A detailed walkthrough for installing a specific legacy version of MariaDB (10.1.21) on CentOS 7 using individual RPM packages, including dependency resolution.

See Troubleshooting MariaDB Installs on RHEL / CentOS.

Solutions for common installation issues on RHEL and CentOS, such as conflicts with existing MySQL installations and handling configuration file backups (.rpmsave).

See Why Source RPMs (SRPMs) Aren't Packaged For Some Platforms.

Explains the limitations in providing Source RPMs (SRPMs) for certain platforms due to CMake version requirements and build system dependencies.

This page is licensed: CC BY-SA / Gnu FDL

About the MariaDB RPM Files

Available RPM Packages

The available RPM packages depend on the specific MariaDB release series.

List of RPM Packages in MariaDB

ENTERPRISE SERVER

The following RPMs are in MariaDB Enterprise Server 11.8:

Package Name	Description
`galera-4`	The WSREP provider for Galera 4
`galera-4-debuginfo`	Debuginfo for `galera-4`
`MariaDB-backup`	mariadb-backup is a command-line utility for creating consistent and reliable backups of MariaDB databases, supporting full and incremental backup options
`MariaDB-backup-debuginfo`	Debuginfo for `mariadb-backup`
`MariaDB-client`	Client tools like mariadb CLI, `mariadb-dump`, and others
`MariaDB-client-compat`	Symbolic links from old MySQL tool names to MariaDB, like `mysqladmin` → `mariadb-admin` or `mysql` → `mariadb`. Good to have if you are using MySQL tool names in your scripts
`MariaDB-client-debuginfo`	Debuginfo for client tools like mariadb CLI, `mariadb-dump`, and others
`MariaDB-columnstore-cmapi`	MariaDB ColumnStore cluster management API (CMAPI) and command line tool

Package Name	Description
`MariaDB-columnstore-engine`	The MariaDB ColumnStore storage engine
`MariaDB-columnstore-engine-debuginfo`	Debuginfo for `MariaDB-columnstore-engine`
`MariaDB-common`	Character set files and `/etc/my.cnf`
`MariaDB-common-debuginfo`	Debuginfo for character set files and `/etc/my.cnf`
`MariaDB-cracklib-password-check`	The `cracklib_password_check` password validation plugin
`MariaDB-cracklib-password-check-debuginfo`	Debuginfo for the `cracklib_password_check` password validation plugin
`MariaDB-devel`	Development headers and static libraries
`MariaDB-devel-debuginfo`	Debuginfo for development headers and static libraries
`MariaDB-gssapi-server`	The gssapi authentication plugin
`MariaDB-gssapi-server-debuginfo`	Debuginfo for the gssapi authentication plugin
`MariaDB-hashicorp-key-management`	Encryption plugin that uses Hashicorp Vault for storing encryption keys for MariaDB Data-at-Rest encryption
`MariaDB-hashicorp-key-management-debuginfo`	Debuginfo for `MariaDB-hashicorp-key-management`
`MariaDB-provider-bzip2`	BZip2 compression support in the server and storage engines
`MariaDB-provider-bzip2-debuginfo`	Debuginfo for `MariaDB-provider-bzip2`
`MariaDB-provider-lz4`	LZ4 compression support in the server and storage engines
`MariaDB-provider-lz4-debuginfo`	Debuginfo for `MariaDB-provider-lz4`
`MariaDB-provider-lzma`	LZMA compression support in the server and storage engines
`MariaDB-provider-lzma-debuginfo`	Debuginfo for `MariaDB-provider-lzma`
`MariaDB-provider-lzo`	LZO compression support in the server and storage engines
`MariaDB-provider-lzo-debuginfo`	Debuginfo for `MariaDB-provider-lzo`
`MariaDB-provider-snappy`	Snappy compression support in the server and storage engines
`MariaDB-provider-snappy-debuginfo`	Debuginfo for `MariaDB-provider-snappy`
`MariaDB-rocksdb-engine`	The MyRocks storage engine
`MariaDB-rocksdb-engine-debuginfo`	Debuginfo for the MyRocks storage engine
`MariaDB-s3-engine`	The S3 storage engine, which allows one to archive MariaDB tables in Amazon S3 (or any third-party public or private cloud that implements S3 API), but still have them accessible in MariaDB in read-only mode

Package Name	Description
`MariaDB-s3-engine-debuginfo`	Debuginfo for the S3 storage engine
`MariaDB-server`	The server and server tools, like `myisamchk` and `mariadb-hotcopy` are here
`MariaDB-server-compat`	Symbolic links from old MySQL server executable names to MariaDB, like `mysqld` -> `mariadbd` or `mysql_install_db` -> `mariadb-install-db`. Good to have if you are using MySQL tool names in your scripts
`MariaDB-server-debuginfo`	Debuginfo for the server and server tools, like `myisamchk` and `mariadb-hotcopy` are here
`MariaDB-shared`	Dynamic client libraries
`MariaDB-shared-debuginfo`	Debuginfo for dynamic client libraries
`MariaDB-spider-engine`	The Spider storage engine
`MariaDB-spider-engine-debuginfo`	Debuginfo for the Spider storage engine

Additionally, the following RPMs are available in the MariaDB Enterprise Unsupported repository:

Package Name	Description
`MariaDB-connect-engine`	The CONNECT storage engine
`MariaDB-connect-engine-debuginfo`	Debuginfo for the CONNECT storage engine
`MariaDB-connect-engine-jdbc`	Connect storage engine JDBC interface used to interact with external DBMS via Java connector
`MariaDB-test`	`mysql-client-test` executable, and mysql-test framework with the tests
`MariaDB-test-debuginfo`	Debuginfo for `MariaDB-test`

COMMUNITY SERVER

The following RPMs are in MariaDB Community Server 12.3:

Package Name	Description
`galera-4`	The WSREP provider for Galera 4
`galera-4-debuginfo`	Debuginfo for `galera-4`
`galera-4-debugsource`	Debug sources for package `galera-4`
`MariaDB-backup`	mariadb-backup is a command-line utility for creating consistent and reliable backups of MariaDB databases, supporting full and incremental backup options
`MariaDB-backup-debuginfo`	Debuginfo for `mariadb-backup`
`MariaDB-client`	Client tools like mariadb CLI, `mariadb-dump`, and others
`MariaDB-client-compat`	Symbolic links from old MySQL tool names to MariaDB, like `mysqladmin` -> `mariadb-admin` or `mysql` -> `mariadb`. Good to have if you are using MySQL tool names in your scripts
`MariaDB-client-debuginfo`	Debuginfo for client tools like mariadb CLI, `mariadb-dump`, and others
`MariaDB-columnstore-engine`	The MariaDB ColumnStore storage engine
`MariaDB-columnstore-engine-debuginfo`	Debuginfo for `MariaDB-columnstore-engine`
`MariaDB-common`	Character set files and `/etc/my.cnf`

Package Name	Description
`MariaDB-common-debuginfo`	Debuginfo for character set files and `/etc/my.cnf`
`MariaDB-connect-engine`	The CONNECT storage engine
`MariaDB-connect-engine-debuginfo`	Debuginfo for the CONNECT storage engine
`MariaDB-connect-engine-jdbc`	Connect storage engine JDBC interface used to interact with external DBMS via Java connector
`MariaDB-cracklib-password-check`	The `cracklib_password_check` password validation plugin
`MariaDB-cracklib-password-check-debuginfo`	Debuginfo for the `cracklib_password_check` password validation plugin
`MariaDB-devel`	Development headers and static libraries
`MariaDB-devel-debuginfo`	Debuginfo for development headers and static libraries
`MariaDB-gssapi-server`	The gssapi authentication plugin
`MariaDB-gssapi-server-debuginfo`	Debuginfo for the gssapi authentication plugin
`MariaDB-hashicorp-key-management`	Encryption plugin that uses Hashicorp Vault for storing encryption keys for MariaDB Data-at-Rest encryption
`MariaDB-hashicorp-key-management-debuginfo`	Debuginfo for `MariaDB-hashicorp-key-management`
`MariaDB-oqgraph-engine`	The Open Query GRAPH computation engine, or OQGRAPH, storage engine
`MariaDB-oqgraph-engine-debuginfo`	Debuginfo for the OQGRAPH storage engine
`MariaDB-provider-bzip2`	BZip2 compression support in the server and storage engines
`MariaDB-provider-bzip2-debuginfo`	Debuginfo for `MariaDB-provider-bzip2`
`MariaDB-provider-lz4`	LZ4 compression support in the server and storage engines
`MariaDB-provider-lz4-debuginfo`	Debuginfo for `MariaDB-provider-lz4`
`MariaDB-provider-lzma`	LZMA compression support in the server and storage engines
`MariaDB-provider-lzma-debuginfo`	Debuginfo for `MariaDB-provider-lzma`
`MariaDB-provider-lzo`	LZO compression support in the server and storage engines
`MariaDB-provider-lzo-debuginfo`	Debuginfo for `MariaDB-provider-lzo`
`MariaDB-provider-snappy`	Snappy compression support in the server and storage engines
`MariaDB-provider-snappy-debuginfo`	Debuginfo for `MariaDB-provider-snappy`

Package Name	Description
`MariaDB-rocksdb-engine`	The MyRocks storage engine
`MariaDB-rocksdb-engine-debuginfo`	Debuginfo for the MyRocks storage engine
`MariaDB-s3-engine`	The S3 storage engine, which allows one to archive MariaDB tables in Amazon S3 (or any third-party public or private cloud that implements S3 API), but still have them accessible in MariaDB in read-only mode
`MariaDB-s3-engine-debuginfo`	Debuginfo for the S3 storage engine
`MariaDB-server_galera`	Provides scripts and plugins required for Galera Cluster operation. Decoupled from the base server package starting in 12.3
`MariaDB-server`	The server and server tools, like `myisamchk` and `mariadb-hotcopy` are here
`MariaDB-server-compat`	Symbolic links from old MySQL server executable names to MariaDB, like `mysqld` -> `mariadbd` or `mysql_install_db` -> `mariadb-install-db`. Good to have if you are using MySQL tool names in your scripts
`MariaDB-server-debuginfo`	Debuginfo for the server and server tools, like `myisamchk` and `mariadb-hotcopy` are here
`MariaDB-shared`	Dynamic client libraries
`MariaDB-shared-debuginfo`	Debuginfo for dynamic client libraries
`MariaDB-test`	`mysql-client-test` executable, and mysql-test framework with the tests
`MariaDB-test-debuginfo`	Debuginfo for `MariaDB-test`

Removed RPM Packages

Occasionally, RPM packages are removed from MariaDB. This could be because they are no longer needed, or because they were for features or storage engines that have been removed from MariaDB.

Some examples of removed RPMs are:

Package Name	Description
MariaDB-compat	Old shared client libraries, removed in MariaDB 11.4 as they were no longer needed
MariaDB-tokudb-engine	The TokuDB storage engine. It was disabled in MariaDB 10.5 and removed in MariaDB 10.6

Installing RPM Packages

Preferably, you should install MariaDB RPM packages using the package manager of your Linux distribution, for example `yum` or `zypper`. But you can also use the lower-level `rpm` tool.

Actions Performed by RPM Packages

Users and Groups Created

When the `aria-server` RPM package is installed, it will create a user and group named `mysql`, if it does not already exist.

See Also

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux RPM based distributions, these provide MariaDB packages. These are supported by those distributions. If you have a particular need for a later version than what is in the distribution, then MariaDB provides repositories for them.

Using repositories rather than installing RPM allows for an ease of update when a new release is made. It is highly recommended to install the relevant RPM packages from MariaDB's repository using yum or dnf. Centos 7 still uses `yum`, most others use `dnf`, and SUSE/openSUSE use `zypper`.

This page walks you through the simple installation steps using `dnf` and `yum`.

Adding the MariaDB YUM repository

We currently have YUM/DNF repositories for the following Linux distributions, and for the versions that are in standard (not extended) support:

- Red Hat Enterprise Linux (RHEL)
- CentOS
- Fedora
- openSUSE
- SUSE

Using the MariaDB Package Repository Setup Script

MariaDB provides two helpful scripts for setting up repositories, one for MariaDB Community Server named `mariadb_repo_setup`, and one for MariaDB Enterprise Server named `mariadb_es_repo_setup`.

See the Using MariaDB Corporation's Repository Setup Scripts section on the MariaDB Package Repository Setup and Usage page for information on using these scripts.

Using the MariaDB Foundation Repository Configuration Tool

Visit https://mariadb.org/download/?t=repo-config and follow the instructions from there. It will ask for your Linux distribution, desired MariaDB version, and the mirror to use, and will show what files to edit and what commands to run to configure a repository.

Pinning the MariaDB Repository to a Specific Minor Release

If you wish to pin your `yum` or `dnf` repository to a specific minor release, or if you would like to downgrade to a specific minor release, then you can configure a repository with the URL hard-coded to that specific minor release.

> #### MARIADB CORPORATION REPO SETUP SCRIPTS
>
> If you used MariaDB Corporation's `mariadb_repo_setup` or `mariadb_es_repo_setup` scripts to generate your repository configuration, simply re-run the script and specify the full version number to use with the `--mariadb-server-version` option.
>
> See Pinning the Repository to a Specific Minor Release on the MariaDB Package Repository Setup and Usage page for details.
>
> The full list of MariaDB Enterprise Server releases can be found on the Enterprise Server - All Releases page.
>
> #### MARIADB FOUNDATION REPO CONFIG TOOL
>
> If you used the MariaDB Foundation's Repository Configuration tool, then you need to update the repository file you created to include the full version number to use on the `baseurl` line.

By default the Foundation's tool configures repositories with just the main series of MariaDB, e.g. `mariadb-11.8`, and to pin to a specific version you need to specify the full version, for example `mariadb-11.8.6`.

The full list of MariaDB Community Server releases can be found on the Community Server - All Releases page.

For example, to pin your repository to MariaDB 11.8.6 on RHEL/Alma/Rocky 8, 9, or 10, then you could use the following repository configuration in `/etc/yum.repos.d/MariaDB.repo`:

```
[mariadb]
name = MariaDB-11.8.6
baseurl= http://archive.mariadb.org/mariadb-11.8.6/yum/rhel/$releasever/$basearch
gpgkey= https://archive.mariadb.org/PublicKey
gpgcheck=1
```

After updating the repository configuration, it is a good idea to clean the repository metadata with:

```
sudo dnf clean all
```

Updating the MariaDB YUM repository to a New Major Release

MariaDB's `yum` repository can be updated to a new major release. How this is done depends on how you originally configured the repository.

Updating the Major Release with the MariaDB Package Repository Setup Script

If you configured `yum` to install from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script, then you can update the major release that the repository uses by running the script again.

Updating the Major Release with the MariaDB Repository Configuration Tool

If you configured `yum` to install from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool, then you can update the major release that the repository uses by updating the `yum` repository configuration file in-place. For example, if you wanted to change the repository from MariaDB 10.6 to MariaDB 10.11, and if the repository configuration file was at `/etc/yum.repos.d/MariaDB.repo`, then you could execute the following:

```
sudo sed -i 's/10.6/10.11/' /etc/yum.repos.d/MariaDB.repo
```

After that, the repository should refer to MariaDB 10.11.

If the `yum` repository is pinned to a specific minor release, then the above `sed` command can result in an invalid repository configuration. In that case, the recommended options are:

- Edit the `MariaDB.repo` repository file manually.
- Or delete the `MariaDB.repo` repository file, and then install the repository of the new version with the more robust MariaDB Package Repository setup script.

The MariaDB GPG Key

See the GPG page for information on the various keys used by MariaDB.

Installing MariaDB Packages with YUM/DNF

After the `dnf` / `yum` repository is configured, you can install MariaDB by executing the dnf or yum command. The specific command that you would use would depend on which specific packages that you want to install.

Installing the Most Common Packages

To Install the most common packages, execute the following command:

```
sudo dnf install MariaDB-server MariaDB-server-galera galera-4 MariaDB-client MariaDB-shared MariaDB-backup MariaDB-common
```

Installing MariaDB Server

To Install MariaDB Server, execute the following command:

```
sudo dnf install MariaDB-server
```

Installing MariaDB Galera Cluster with YUM

> **Note:**
>
> Note for MariaDB 12.3 and later: Galera Cluster support is no longer included in the base `MariaDB-server` package. To enable cluster functionality, you must explicitly install the `MariaDB-server-galera` package. This package contains cluster-specific scripts, systemd bootstrap capability, and the `wsrep_info` plugin.

You need to install the `galera-4` package to obtain the Galera 4 wsrep provider library.

To install MariaDB Galera Cluster, you could execute the following command:

```
sudo dnf install MariaDB-server MariaDB-server-galera MariaDB-client galera-4
```

If you haven't yet imported the MariaDB GPG public key, then `yum` will prompt you to import it after it downloads the packages, but before it prompts you to install them.

See MariaDB Galera Cluster for more information on MariaDB Galera Cluster.

Installing MariaDB Clients and Client Libraries with YUM

MariaDB Connector/C has been included as the client library (statically linked). However, the package name for the client library has not been changed.

To Install the clients and client libraries, execute the following command:

```
sudo dnf install MariaDB-client MariaDB-shared
```

If you want compile your own programs against MariaDB Connector/C, execute the following command:

```
sudo dnf install MariaDB-devel
```

Installing mariadb-backup with YUM

To install mariadb-backup, execute the following command:

```
sudo yum install MariaDB-backup
```

Installing Plugins with YUM

Some plugins may also need to be installed.

For example, to install the cracklib_password_check password validation plugin, execute the following command:

```
sudo dnf install MariaDB-cracklib-password-check
```

Installing Debug Info Packages with YUM

The MariaDB `yum` repository also contains debuginfo packages. These package may be needed when debugging a problem.

Installing Debug Info for the Most Common Packages with YUM

To install debuginfo for the most common packages, execute the following command:

```
sudo dnf install MariaDB-server-debuginfo MariaDB-client-debuginfo MariaDB-shared-debuginfo MariaDB-
backup-debuginfo MariaDB-common-debuginfo
```

All packages have their debuginfo by appending `-debuginfo` to the package name.

Installing Debug Info for MariaDB Server with YUM

To install debuginfo for MariaDB Server, execute the following command:

```
sudo dnf install MariaDB-server-debuginfo
```

Installing Older Versions from the Repository

The MariaDB `yum` repository contains the last few versions of MariaDB. To show what versions are available, use the following command:

```
sudo dnf list --showduplicates MariaDB-server
```

The output shows the available versions. For example:

```
sudo dnf list --showduplicates MariaDB-server
Last metadata expiration check: 0:01:42 ago on Fri 12 Dec 2025 03:47:20 PM UTC.
Available Packages
MariaDB-server.x86_64  11.8.2-1.el8                              mariadb-main
MariaDB-server.x86_64  12.0.2-1.el8                              mariadb-main
MariaDB-server.x86_64  12.1.2-1.el8                              mariadb-main
mariadb-server.x86_64  3:10.3.39-1.module+el8.8.0+1452+2a7eab68   appstream
```

The MariaDB repository in this example contains MariaDB 12.1.2, 12.0.2, and 11.8.2; and the appstream repository contains MariaDB 10.3.39.

To install an older version of a package instead of the latest version we just need to specify the package name, a dash, and then the version number. And we only need to specify enough of the version number for it to be unique from the other available versions.

However, when installing an older version of a package, if dependencies need to be installed, then it will automatically choose to install the latest versions of those packages, which can sometimes break those dependencies. To ensure that all MariaDB packages are on the same version in this scenario, it is necessary to specify them all.

The MariaDB packages that the `MariaDB-server` package depend on are: `MariaDB-client`, `MariaDB-shared`, and `MariaDB-common`. Therefore, to install MariaDB 12.0.2 from this `yum` repository, we could do the following (putting the version in a variable and each package on its own line so things are cleaner):

```
ver=12.0.2
sudo dnf install \
  MariaDB-server-${ver} \
  MariaDB-client-${ver} \
  MariaDB-shared-${ver} \
  MariaDB-common-${ver}
```

For MariaDB Enterprise it is necessary to specify the release part of the version number as well, but with an underscore (_) instead of a dash (-), as that is how dnf/yum see the version number. For example, for MariaDB Enterprise Server 11.8.5-2 you would specify the version as `11.8.5_2`. For example:

```
ver=11.8.5_2
sudo dnf install \
  MariaDB-server-${ver} \
  MariaDB-client-${ver} \
  MariaDB-shared-${ver} \
  MariaDB-common-${ver}
```

The rest of the install and setup process is as normal.

After Installation

After the installation is complete, you can start MariaDB with:

```
sudo systemctl start mariadb
```

If you are using MariaDB Galera Cluster, then keep in mind that the first node will have to be bootstrapped.

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant RPM packages from MariaDB's repository using zypper.

This page walks you through the simple installation steps using `zypper`.

Adding the MariaDB ZYpp repository

We currently have ZYpp repositories for the following Linux distributions:

- SUSE Linux Enterprise Server (SLES) 12
- SUSE Linux Enterprise Server (SLES) 15
- OpenSUSE 15
- OpenSUSE 42

Using the MariaDB Package Repository Setup Script

If you want to install MariaDB with `zypper`, then you can configure `zypper` to install from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script.

MariaDB Corporation provides a MariaDB Package Repository for several Linux distributions that use `zypper` to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, clients and utilities, client libraries, plugins, and mariadb-backup. The MariaDB Package Repository setup script automatically configures your system to install packages from the MariaDB Package Repository.

To use the script, execute the following command:

```
curl -sS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
```

Note that this script also configures a repository for MariaDB MaxScale and a repository for MariaDB Tools, which currently only contains Percona XtraBackup and its dependencies.

See MariaDB Package Repository Setup and Usage for more information.

Using the MariaDB Repository Configuration Tool

If you want to install MariaDB with `zypper`, then you can configure `zypper` to install from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool.

The MariaDB Foundation provides a MariaDB repository for several Linux distributions that use `zypper` to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, clients and utilities, client libraries, plugins, and mariadb-backup. The MariaDB Repository Configuration Tool can easily generate the appropriate commands to add the repository for your distribution.

For example, if you wanted to use the repository to install MariaDB 10.6 on SLES 15, then you could use the following commands to add the MariaDB `zypper` repository:

```
sudo zypper addrepo --gpgcheck --refresh https://yum.mariadb.org/10.6/sles/15/x86_64 mariadb
sudo zypper --gpg-auto-import-keys refresh
```

Pinning the MariaDB Repository to a Specific Minor Release

If you wish to pin the `zypper` repository to a specific minor release, or if you would like to downgrade to a specific minor release, then you can create a `zypper` repository with the URL hard-coded to that specific minor release.

MARIADB CORPORATION REPO SETUP SCRIPTS

If you used MariaDB Corporation's `mariadb_repo_setup` or `mariadb_es_repo_setup` scripts to generate your repository configuration, simply re-run the script and specify the full version number to use with the `--mariadb-server-version` option.

See Pinning the Repository to a Specific Minor Release on the MariaDB Package Repository Setup and Usage page for details.

The full list of MariaDB Enterprise Server releases can be found on the Enterprise Server - All Releases page.

MARIADB FOUNDATION REPO CONFIG TOOL

If you used the MariaDB Foundation's Repository Configuration tool, then you need to update the repository file you created to include the full version number to use on the `baseurl` line.

By default the Foundation's tool configures repositories with just the main series of MariaDB, e.g. `mariadb-11.8`, and to pin to a specific version you need to specify the full version, for example `mariadb-11.8.6`.

The full list of MariaDB Community Server releases can be found on the Community Server - All Releases page.

For example, to pin your repository to MariaDB 11.8.6 on SLES 15, then you could use the following commands to add the MariaDB `zypper` repository:

```
sudo zypper removerepo mariadb
sudo zypper addrepo --gpgcheck --refresh https://yum.mariadb.org/11.8.6/sles/15/x86_64 mariadb
```

Updating the MariaDB ZYpp repository to a New Major Release

MariaDB's `zypper` repository can be updated to a new major release. How this is done depends on how you originally configured the repository.

Updating the Major Release with the MariaDB Package Repository Setup Script

If you configured `zypper` to install from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script, then you can update the major release that the repository uses by running the script again.

Updating the Major Release with the MariaDB Repository Configuration Tool

If you configured `zypper` to install from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool, then you can update the major release that the repository uses by removing the repository for the old version and adding the repository for the new version.

First, you can remove the repository for the old version by executing the following command:

```
sudo zypper removerepo mariadb
```

After that, you can add the repository for the new version. For example, if you wanted to use the repository to install MariaDB 10.6 on SLES 15, then you could use the following commands to add the MariaDB `zypper` repository:

```
sudo zypper addrepo --gpgcheck --refresh https://yum.mariadb.org/10.6/sles/15/x86_64 mariadb
sudo zypper --gpg-auto-import-keys refresh
```

After that, the repository should refer to MariaDB 10.6.

Importing the MariaDB GPG Public Key

Before MariaDB can be installed, you also have to import the GPG public key that is used to verify the digital signatures of the packages in our repositories. This allows the `zypper` and `rpm` utilities to verify the integrity of the packages that they install.

See the GPG page for information on the various keys used by MariaDB.

The rpm utility can be used to import this key. For example:

```
sudo rpm --import https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
```

Once the GPG public key is imported, you are ready to install packages from the repository.

Installing MariaDB Packages with ZYpp

After the `zypper` repository is configured, you can install MariaDB by executing the zypper command. The specific command that you would use would depend on which specific packages that you want to install.

Installing the Most Common Packages with ZYpp

To Install the most common packages, execute the following command:

```
sudo zypper install MariaDB-server MariaDB-server-galera galera-4 MariaDB-client MariaDB-shared MariaDB-
backup MariaDB-common
```

Installing MariaDB Server with ZYpp

To Install MariaDB Server, execute the following command:

```
sudo zypper install MariaDB-server
```

Installing MariaDB Galera Cluster with ZYpp

> **Note:**
>
> Note for MariaDB 12.3 and later: Galera Cluster support is no longer included in the base `MariaDB-server` package. To enable cluster functionality, you must explicitly install the `MariaDB-server-galera` package. This package contains cluster-specific scripts, systemd bootstrap capability, and the `wsrep_info` plugin.

You also need to install the `galera-4` package to obtain the Galera 4 wsrep provider library.

To install MariaDB Galera Cluster, you could execute the following command:

```
sudo zypper install MariaDB-server MariaDB-server-galera MariaDB-client galera-4
```

If you haven't yet imported the MariaDB GPG public key, then `zypper` will prompt you to import it after it downloads the packages, but before it prompts you to install them.

See MariaDB Galera Cluster for more information on MariaDB Galera Cluster.

Installing MariaDB Clients and Client Libraries with ZYpp

MariaDB Connector/C has been included as the client library. However, the package name for the client library has not been changed.

To Install the clients and client libraries, execute the following command:

```
sudo zypper install MariaDB-client MariaDB-shared
```

Installing mariadb-backup with ZYpp

To install mariadb-backup, execute the following command:

```
sudo zypper install MariaDB-backup
```

Installing Plugins with ZYpp

Some plugins may also need to be installed.

For example, to install the cracklib_password_check password validation plugin, execute the following command:

```
sudo zypper install MariaDB-cracklib-password-check
```

Installing Debug Info Packages with ZYpp

The MariaDB `zypper` repository also contains debuginfo packages. These package may be needed when debugging a problem.

Installing Debug Info for the Most Common Packages with ZYpp

To install debuginfo for the most common packages, execute the following command:

```
sudo zypper install MariaDB-server-debuginfo MariaDB-client-debuginfo MariaDB-shared-debuginfo MariaDB-
backup-debuginfo MariaDB-common-debuginfo
```

Installing Debug Info for MariaDB Server with ZYpp

To install debuginfo for MariaDB Server, execute the following command:

```
sudo zypper install MariaDB-server-debuginfo
```

Installing Debug Info for MariaDB Clients and Client Libraries with ZYpp

MariaDB Connector/C has been included as the client library. However, the package name for the client library has not been changed.

To install debuginfo for the clients and client libraries, execute the following command:

```
sudo zypper install MariaDB-client-debuginfo MariaDB-shared-debuginfo
```

Installing Debug Info for mariadb-backup with ZYpp

To install debuginfo for mariadb-backup, execute the following command:

```
sudo zypper install MariaDB-backup-debuginfo
```

Installing Debug Info for Plugins with ZYpp

For some plugins, debuginfo may also need to be installed.

For example, to install debuginfo for the cracklib_password_check password validation plugin, execute the following command:

```
sudo zypper install MariaDB-cracklib-password-check-debuginfo
```

Installing Older Versions from the Repository

The MariaDB `zypper` repository contains the last few versions of MariaDB. To show what versions are available, use the following command:

```
zypper search --details MariaDB-server
```

In the output you will see the available versions.

To install an older version of a package instead of the latest version we just need to specify the package name, a dash, and then the version number. And we only need to specify enough of the version number for it to be unique from the other available versions.

However, when installing an older version of a package, if `zypper` has to install dependencies, then it will automatically choose to install the latest versions of those packages. To ensure that all MariaDB packages are on the same version in this scenario, it is necessary to specify them all.

The core packages required for a server installation include `MariaDB-server`, `MariaDB-client`, `MariaDB-shared`, `MariaDB-backup`, and `MariaDB-common`. For MariaDB 12.3 and later, you must also explicitly include the `MariaDB-server-galera` package if cluster functionality is required.

Therefore, to install MariaDB 10.6.21 from this `zypper` repository, we would do the following:

```
sudo zypper install MariaDB-server-10.6.21 MariaDB-client-10.6.21 MariaDB-shared-10.6.21 MariaDB-backup-10.6.21 MariaDB-common-10.6.21
```

To install MariaDB 12.3 (including Galera support):"

```
sudo zypper install MariaDB-server-12.3.2 MariaDB-server-galera-12.3.2 MariaDB-client-12.3.2 MariaDB-shared-12.3.2 MariaDB-backup-12.3.2 MariaDB-common-12.3.2
```

The rest of the install and setup process is as normal.

After Installation

After the installation is complete, you can start MariaDB.

If you are using MariaDB Galera Cluster, then keep in mind that the first node will have to be bootstrapped.

This page is licensed: CC BY-SA / Gnu FDL

Checking MariaDB RPM Package Signatures

MariaDB RPM packages since MariaDB 5.1.55 are signed.

For **MariaDB Community Server**, see the MariaDB Community Server RPM / Source Keys section of the GPG page for details on how to import the key used by those repositories.

For **MariaDB Enterprise Server**, see the MariaDB Enterprise GPG Keys section of the GPG page for details on how to import the key used by those repositories.

To check the signature you first need to import the public part of the key like so:

```
gpg --keyserver hkp://pgp.mit.edu --recv-keys 1BB943DB
```

Next you need to let pgp know about the key like so:

```
gpg --export --armour 1BB943DB > mariadb-signing-key.asc
sudo rpm --import mariadb-signing-key.asc
```

You can check to see if the key was imported with:

```
rpm -qa gpg-pubkey*
```

Once the key is imported, you can check the signature of the MariaDB RPM files by running the something like the following in your download directory:

```
rpm --checksig $(find . -name '*.rpm')
```

The output of the above will look something like this (make sure gpg shows up on each OK line):

```
me@desktop:~$ rpm --checksig $(find . -name '*.rpm')
./kvm-rpm-centos5-amd64/rpms/MariaDB-test-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-server-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-client-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-shared-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-devel-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-debuginfo-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/srpms/MariaDB-5.1.55-98.el5.src.rpm: (sha1) dsa sha1 md5 gpg OK
```

See Also

- Installing MariaDB RPM Files
- Troubleshooting MariaDB Installs on RedHat/CentOS

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB With the rpm Tool

This article describes how to download the RPM files and install them using the `rpm` command.

It is highly recommended to Install MariaDB with yum where possible.

Navigate toand choose the desired database version and then select the RPMs that match your Linux distribution and architecture.

Clicking those links takes you to a local mirror. Choose the rpms link and download the desired packages. The packages will be similar to the following:

```
MariaDB-client-5.2.5-99.el5.x86_64.rpm
MariaDB-debuginfo-5.2.5-99.el5.x86_64.rpm
MariaDB-devel-5.2.5-99.el5.x86_64.rpm
MariaDB-server-5.2.5-99.el5.x86_64.rpm
MariaDB-shared-5.2.5-99.el5.x86_64.rpm
MariaDB-test-5.2.5-99.el5.x86_64.rpm
```

For a standard server installation you will need to download at least the *client*, *shared*, and *server* RPM files. See About the MariaDB RPM Files for more information about what is included in each RPM package.

After downloading the MariaDB RPM files, you might want to check their signatures. See Checking MariaDB RPM Package Signatures for more information about checking signatures.

```
rpm --checksig $(find . -name '*.rpm')
```

Prior to installing MariaDB, be aware that it will conflict with an existing installation of MySQL. To check whether MySQL is already installed, issue the command:

```
rpm -qa 'mysql*'
```

If necessary, you can remove found MySQL packages before installing MariaDB.

To install MariaDB, use the command:

```
rpm -ivh MariaDB-*
```

You should see output such as the following:

```
Preparing...                ########################################### [100%]
   1:MariaDB-shared         ########################################### [ 14%]
   2:MariaDB-client         ########################################### [ 29%]
   3:MariaDB-client         ########################################### [ 43%]
   4:MariaDB-debuginfo      ########################################### [ 57%]
   5:MariaDB-devel          ########################################### [ 71%]
   6:MariaDB-server         ########################################### [ 86%]

PLEASE REMEMBER TO SET A PASSWORD FOR THE MariaDB root USER !
To do so, start the server, then issue the following commands:

/usr/bin/mariadb-admin -u root password 'new-password'
/usr/bin/mariadb-admin -u root -h hostname password 'new-password'

Alternatively you can run:
/usr/bin/mysql_secure_installation

which will also give you the option of removing the test
databases and anonymous user created by default.  This is
strongly recommended for production servers.

See the MySQL manual for more instructions.

Please report any problems with the /usr/bin/mysqlbug script!

The latest information about MariaDB is available at http://www.askmonty.org/.
You can find additional information about the MySQL part at:
http://dev.mysql.com
Support MariaDB development by buying support/new features from
Monty Program Ab. You can contact us about this at sales@askmonty.org.
Alternatively consider joining our community based development effort:
http://askmonty.org/wiki/index.php/MariaDB#How_can_I_participate_in_the_development_of_MariaDB

Starting MySQL....[  OK  ]
Giving mysqld 2 seconds to start
   7:MariaDB-test           ########################################### [100%]
```

Be sure to follow the instructions given in the preceding output and create a password for the root user either by using mariadb-admin or by running the `/usr/bin/mysql_secure_installation` script.

Installing the MariaDB RPM files installs the MySQL tools in the `/usr/bin` directory. You can confirm that MariaDB has been installed by using the mariadb client program. Issuing the command `mariadb` should give you the MariaDB cursor.

See Also

- Installing MariaDB with yum
- Troubleshooting MariaDB Installs on RedHat/CentOS
- Checking MariaDB RPM Package Signatures

This page is licensed: CC BY-SA / Gnu FDL

MariaDB for DirectAdmin Using RPMs

If you are using DirectAdmin and you encounter any issues with Installing MariaDB with YUM, then the directions below may help. The process is very straightforward.

> **Note:**
>
> Installing with YUM is preferable to installing the MariaDB RPM packages manually, so only do this if you are having issues such as:
> Starting httpd:
>
> ```
> httpd:
>   Syntax error on line 18 of /etc/httpd/conf/httpd.conf:
>   Syntax error on line 1 of /etc/httpd/conf/extra/httpd-phpmodules.conf:
>     Cannot load /usr/lib/apache/libphp5.so into server:
>       libmysqlclient.so.18: cannot open shared object file: No such file or directory
> ```

Or:

```
Starting httpd:
  httpd:
    Syntax error on line 18 of /etc/httpd/conf/httpd.conf:
    Syntax error on line 1 of /etc/httpd/conf/extra/httpd-phpmodules.conf:
      Cannot load /usr/lib/apache/libphp5.so into server:
        /usr/lib/apache/libphp5.so: undefined symbol: client_errors
```

RPM Installation

To install the RPMs, there is a quick and easy guide to Installing MariaDB with the RPM Tool. Follow the instructions there.

Necessary Edits

We do not want DirectAdmin's custombuild to remove/overwrite our MariaDB installation whenever an update is performed. To rectify this, disable automatic MySQL installation.

Edit `/usr/local/directadmin/custombuild/options.conf` and replace `mysql_inst=yes` with `mysql_inst=no`

```
sudo sed -i 's/^mysql_inst=yes/mysql_inst=no/' /usr/local/directadmin/custombuild/options.conf
```

> **Warning:**
>
> When MariaDB is installed manually (i.e. not using YUM), updates are not automatic. You will need to update the RPMs yourself.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Installation via RPMs on CentOS 7

This guide provides the detailed steps for installing MariaDB 10.1.21 via individual RPM packages on CentOS 7. The process involves installing dependencies, then the main packages, and resolving potential conflicts as they appear.

The RPMs needed for the installation are available from the MariaDB website. The required packages for this guide are:

```
jemalloc-3.6.0-1.el7.x86_64.rpm
MariaDB-10.1.21-centos7-x86_64-client.rpm
MariaDB-10.1.21-centos7-x86_64-compat.rpm
galera-25.3.19-1.rhel7.el7.centos.x86_64.rpm
jemalloc-devel-3.6.0-1.el7.x86_64.rpm
MariaDB-10.1.21-centos7-x86_64-common.rpm
MariaDB-10.1.21-centos7-x86_64-server.rpm
```

Step-by-Step Installation

1. Install Basic Dependencies

First, use yum to install some general system packages that may be required.

```
yum install rsync nmap lsof perl-DBI nmap-ncat
```

2. Install MariaDB RPM Packages

Next, install the downloaded RPMs in sequence. Make sure to run these commands as a user with sufficient privileges (e.g., using sudo).

```
rpm -ivh jemalloc-3.6.0-1.el7.x86_64.rpm
rpm -ivh jemalloc-devel-3.6.0-1.el7.x86_64.rpm
rpm -ivh MariaDB-10.1.21-centos7-x86_64-common.rpm
rpm -ivh MariaDB-10.1.21-centos7-x86_64-compat.rpm
rpm -ivh MariaDB-10.1.21-centos7-x86_64-client.rpm
rpm -ivh galera-25.3.19-1.rhel7.el7.centos.x86_64.rpm
rpm -ivh MariaDB-10.1.21-centos7-x86_64-server.rpm
```

> **Warning:**
>
> During this process, you may encounter errors related to dependencies or conflicts. The sections below describe these common issues and their solutions.

Troubleshooting Common Installation Errors and Warnings

Error 1: Package Conflict with `mariadb-libs`

> **ERROR MESSAGE**
>
> While installing `mariadb-common`, you may encounter a conflict with an existing package.
>
> ```
> # rpm -ivh MariaDB-10.1.21-centos7-x86_64-common.rpm
>
> warning: MariaDB-10.1.21-centos7-x86_64-common.rpm: Header V4 DSA/SHA1 Signature, key ID 1bb943db: NOKEY
> error: Failed dependencies:
>     mariadb-libs < 1:10.1.21-1.el7.centos conflicts with MariaDB-common-10.1.21-1.el7.centos.x86_64
> ```
>
> **SOLUTION**
>
> You must find and remove the conflicting `mariadb-libs` package.

```
# Search for the installed package
rpm -qa | grep mariadb-libs
# Expected output: mariadb-libs-5.5.52-1.el7.x86_64

# Remove the conflicting package (use the exact name from the command above)
rpm -ev --nodeps mariadb-libs-5.5.52-1.el7.x86_64
```

After removing it, you can run the `rpm -ivh MariaDB-10.1.21-centos7-x86_64-common.rpm` command again.

Error 2: Failed Dependency for Galera

ERROR MESSAGE

While installing the Galera package, the installation may fail due to a missing library.

```
# rpm -ivh galera-25.3.19-1.rhel7.el7.centos.x86_64.rpm

error: Failed dependencies:
    libboost_program_options.so.1.53.0()(64bit) is needed by galera-25.3.19-1.rhel7.el7.centos.x86_64
```

SOLUTION

The required dependency `libboost_program_options` can be installed using yum.

```
yum install boost-devel.x86_64
```

After installing `boost-devel`, you can run the `rpm -ivh galera...` command again.

Warning: GPG Key `NOKEY`

WARNING MESSAGE

You may also see a warning about a missing GPG key during the installation.

```
warning: galera-25.3.19-1.rhel7.el7.centos.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID 1bb943db:
NOKEY
```

SOLUTION

This warning can be resolved by importing the official MariaDB GPG key.

```
rpm --import http://yum.mariadb.org/RPM-GPG-KEY-MariaDB
```

Post-Installation

After all RPMs are successfully installed, the final step is to secure the server. This involves setting the root password, removing test databases, and disallowing remote root login.

```
# First, start the newly installed MariaDB service
systemctl start mariadb

# Now, run the security script and follow the prompts
mysql_secure_installation
```

This page is licensed: CC BY-SA / Gnu FDL

Troubleshooting MariaDB Installs on RHEL / CentOS

The following article is about different issues people have encountered when installing MariaDB on RHEL / CentOS.

It is highly recommended to install with yum where possible.

In RHEL/ CentOS it is also possible to install a RPM or a tar ball. The RPM is the preferred version, except if you want to install many versions of MariaDB or install MariaDB in a non standard location.

Replacing MySQL

If you removed an MySQL RPM to install MariaDB, note that the MySQL RPM on uninstall renames `/etc/my.cnf` to `/etc/my.cnf.rpmsave.`

After installing MariaDB you should do the following to restore your configuration options:

```
mv /etc/my.cnf.rpmsave /etc/my.cnf
```

Unsupported configuration options

If you are using any of the following options in your `/etc/my.cnf` or other `my.cnf` file you should remove them. This is also true for MySQL 5.1 or newer:

```
skip-bdb
```

See also

- Installing with yum (recommended)
- Installing MariaDB RPM Files
- Checking MariaDB RPM Package Signatures

This page is licensed: CC BY-SA / Gnu FDL

Why Source RPMs (SRPMs) Aren't Packaged For Some Platforms

MariaDB source RPMs (SRPMs) are not packaged on all platforms for which MariaDB RPMs are packaged.

The reason is that MariaDB's build process relies heavily on cmake for a lot of things. In this specific case, MariaDB's build process relies on CMake CPack Package Generators to build RPMs. The specific package generator that it uses to build RPMs is called CPackRPM.

Support for source RPMs in CPackRPM became usable with MariaDB's build system starting from around cmake 3.10. This means that we do not produce source RPMs on platforms where the installed cmake version is older than that.

See also Building MariaDB from a Source RPM.

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB Alongside MySQL

MariaDB was originally designed as a drop-in replacement of MySQL, with more features, new storage engines, fewer bugs, and better performance, but you can also install it alongside MySQL. (This can be useful, for example, if you want to migrate databases/applications one by one.)

Here are the steps to install MariaDB near an existing MySQL installation.

- Download the compiled binary tar.gz file that contains the latest version.
- Extract the files in a directory of your choice. In the following, we assume you chose the `/opt` directory.

```
[root@mariadb-near-mysql ~]# cat /etc/issue
CentOS release 6.2 (Final)

[root@mariadb-near-mysql ~]# rpm -qa mysql*
mysql-5.1.61-1.el6_2.1.x86_64
mysql-libs-5.1.61-1.el6_2.1.x86_64
mysql-server-5.1.61-1.el6_2.1.x86_64

[root@mariadb-near-mysql ~]# ps axf | grep mysqld
 2072 pts/0    S+     0:00          \_ grep mysqld
 1867 ?        S      0:01 /bin/sh /usr/bin/mysqld_safe --datadir=/var/lib/mysql --
socket=/var/lib/mysql/mysql.sock ...
 1974 ?        Sl     0:06  \_ /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=mysql
...
```

- Create data directory and symlinks as below:

```
[root@mariadb-near-mysql opt]# mkdir mariadb-data
[root@mariadb-near-mysql opt]# ln -s mariadb-5.5.24-linux-x86_64 mariadb
[root@mariadb-near-mysql opt]# ls -al
total 20
drwxr-xr-x.  5 root root 4096 2012-06-06 07:27 .
dr-xr-xr-x. 23 root root 4096 2012-06-06 06:38 ..
lrwxrwxrwx.  1 root root   27 2012-06-06 07:27 mariadb -> mariadb-5.5.24-linux-x86_64
drwxr-xr-x. 13 root root 4096 2012-06-06 07:07 mariadb-5.5.24-linux-x86_64
drwxr-xr-x.  2 root root 4096 2012-06-06 07:26 mariadb-data
```

- Create group mariadb and user mariadb and set correct ownerships:

```
[root@mariadb-near-mysql opt]# groupadd --system mariadb
[root@mariadb-near-mysql opt]# useradd -c "MariaDB Server" -d /opt/mariadb -g mariadb --system mariadb
[root@mariadb-near-mysql opt]# chown -R mariadb:mariadb mariadb-5.5.24-linux-x86_64/
[root@mariadb-near-mysql opt]# chown -R mariadb:mariadb mariadb-data/
```

- Create a new my.cnf in /opt/mariadb from support files:

```
[root@mariadb-near-mysql opt]# cp mariadb/support-files/my-medium.cnf mariadb-data/my.cnf
[root@mariadb-near-mysql opt]# chown mariadb:mariadb mariadb-data/my.cnf
```

- Edit the file /opt/mariadb-data/my.cnf and add custom paths, socket, port, user and the most important of all: data directory and base directory. Finally the file should have at least the following:

```
[client]
port        = 3307
socket      = /opt/mariadb-data/mariadb.sock

[mysqld]
datadir         = /opt/mariadb-data
basedir         = /opt/mariadb
port        = 3307
socket      = /opt/mariadb-data/mariadb.sock
user            = mariadb
```

- Copy the init.d script from support files in the right location:

```
[root@mariadb-near-mysql opt]# cp mariadb/support-files/mysql.server /etc/init.d/mariadb
[root@mariadb-near-mysql opt]# chmod +x /etc/init.d/mariadb
```

- Edit /etc/init.d/mariadb replacing mysql with mariadb as below:

```
- # Provides: mysql
+ # Provides: mariadb
- basedir=
+ basedir=/opt/mariadb
- datadir=
+ datadir=/opt/mariadb-data
- lock_file_path="$lockdir/mysql"
+ lock_file_path="$lockdir/mariadb"
```

The trickiest part will be the last changes to this file. You need to tell mariadb to use only one configuration file. In the **start** section after**$bindir/mysqld_safe** add **–defaults-file=/opt/mariadb-data/my.cnf**. Finally the lines should look like:

```
# Give extra arguments to mysqld with the my.cnf file. This script
# may be overwritten at next upgrade.
$bindir/mysqld_safe --defaults-file=/opt/mariadb-data/my.cnf --datadir="$datadir" --pid-
file="$mysqld_pid_file_path" $other_args >/dev/null 2>&1 &
```

The same change needs to be made to the mariadb-admin command below in the wait_for_ready() function so that the mariadb start command can properly listen for the server start. In the **wait_for_ready()** function, after **$bindir/mariadb-admin** add **–defaults-file=/opt/mariadb-data/my.cnf**. The lines should look like:

```
wait_for_ready () {
[...]
    if $bindir/mariadb-admin --defaults-file=/opt/mariadb-data/my.cnf ping >/dev/null 2>&1; then
```

- Run mariadb-install-db by explicitly giving it the my.cnf file as argument:

```
[root@mariadb-near-mysql opt]# cd mariadb
[root@mariadb-near-mysql mariadb]# scripts/mariadb-install-db --defaults-file=/opt/mariadb-data/my.cnf
```

- Now you can start MariaDB by

```
[root@mariadb-near-mysql opt]# /etc/init.d/mariadb start
Starting MySQL...                                          [  OK  ]
```

- Make MariaDB start at system start:

```
[root@mariadb-near-mysql opt]# cd /etc/init.d
[root@mariadb-near-mysql init.d]# chkconfig --add mariadb
[root@mariadb-near-mysql init.d]# chkconfig --levels 3 mariadb on
```

- Finally test that you have both instances running:

```
[root@mariadb-near-mysql ~]# mysql -e "SELECT VERSION();"
+-----------+
| VERSION() |
+-----------+
| 5.1.61    |
+-----------+
[root@mariadb-near-mysql ~]# mysql -e "SELECT VERSION();" --socket=/opt/mariadb-data/mariadb.sock
+----------------+
| VERSION()      |
+----------------+
| 5.5.24-MariaDB |
+----------------+
```

What about MariaDB Upgrades ?

By having the **mariadb.socket**, **my.cnf** file and **databases** in **/opt/mariadb-data** means that if you want to upgrade the MariaDB version you will only need to:

- Extract the new version from the archive in `/opt` near the current version,
- Stop MariaDB.
- Change the symlink mariadb to point to the new directory.
- Start MariaDB.
- Run upgrade script… but remember you will need to provide the socket option – `socket=/opt/mariadb-data/mariadb.sock` .

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB Binary Tarballs

> **Note:**
>
> Starting with MariaDB 12.3, the generic binary tarballs (bintars) no longer include the Galera Cluster library or management scripts by default. Users requiring cluster functionality should obtain these libraries from the official MariaDB repositories or via the `mariadb-server-galera` package.

Binary tarballs (bintars) are compressed tar archives that contain pre-compiled executables, libraries, and other deployment dependencies. They can usually be installed on any modern Linux distribution.

MariaDB Binary tarballs are named following the pattern: `mariadb-VERSION-OS.tar.gz` . Be sure to download the correct version for your machine.

> **Warning:**
>
> Some older binary tarballs are marked *'(GLIBC_2.14)'* or *'(requires GLIBC_2.14+)'*. These binaries are built the same as the others, but on a newer build host, and they require GLIBC 2.14 or higher. Use the other binaries for machines with older versions of GLIBC installed. Run `ldd --version` to see which version is running on your distribution.
> Others are marked `systemd` , which are for systems with `systemd` and GLIBC 2.19 or higher.

Benefits of Binary Tarballs

Binary tarballs provide multiple benefits:

- They are highly OS independent. As long as you get the `bintar` for the architecture, GLIBC version, and if you are using `systemd` or not, the `bintar` should work almost anywhere.
- You do not need to be root to use them.
  - They can be installed by anyone to any path, including one's home directory.
- You can have any number of different MariaDB installations on the same machine. This is often desired during upgrades when one wants to have the old installation running until switching to the new one.
- You can use them on systems for which MariaDB does not support native packages.

Installing binary tarballs

To install the binaries, unpack the distribution into the directory of your choice and run the mariadb-install-db script.

In the example below, we install MariaDB in the `/usr/local/mysql` directory (this is the default location for MariaDB for many platforms). However, any other directory should work too.

We install the binary with a symlink to the original name. This is done so that you can easily change MariaDB versions just by moving the symlink to point to another directory.

Ensure You Use the Correct my.cnf Files

MariaDB searches for the configuration files '`/etc/my.cnf`' (on some systems '`/etc/mysql/my.cnf`') and '`~/.my.cnf`'. If you have an old `my.cnf` file (maybe from a system installation of MariaDB or MySQL), you need to take care that you don't accidentally use the old one with your new binary .tar installation.

The normal solution for this is to ignore the `my.cnf` file in `/etc` when you use the programs in the tar file.

This is done by creating your own .my.cnf file in your home directory and telling mariadb-install-db,mysqld_safe, and possibly mariadb (the command-line client utility) to **only** use this one with the option '`--defaults-file=~/.my.cnf`'. Note that this has to be the first option for the above commands!

Installing MariaDB as root in /usr/local/mysql

If you have root access to the system, you probably want to install MariaDB under the user and group 'mysql' (to keep compatibility with MySQL installations):

```
groupadd mysql
useradd -g mysql mysql
cd /usr/local
tar -zxvpf /path-to/mariadb-VERSION-OS.tar.gz
ln -s mariadb-VERSION-OS mysql
cd mysql
./scripts/mariadb-install-db --user=mysql
chown -R root .
chown -R mysql data
```

The symlinking with `ln -s` is recommended as it makes it easy to install many MariaDB versions at the same time (for easy testing, upgrading, downgrading, etc).

If you are installing MariaDB to replace MySQL, then you can leave out the call to `mariadb-install-db`. Instead, shut down MySQL. MariaDB should find the path to the data directory from your old `/etc/my.cnf` file (path may vary depending on your system).

To start mariadbd, you should now do:

```
./bin/mariadbd-safe --user=mysql &
or
./bin/mariadbd-safe --defaults-file=~/.my.cnf --user=mysql &
```

To test the connection, modify your $PATH so you can invoke clients such as mariadb, mariadb-dump, etc.

```
export PATH=$PATH:/usr/local/mysql/bin/
```

You may want to modify your .bashrc or .bash_profile to make it permanent.

Installing MariaDB as Not root in Any Directory

Below, change /usr/local to the directory of your choice.

```
cd /usr/local
gunzip < /path-to/mariadb-VERSION-OS.tar.gz | tar xf -
ln -s mariadb-VERSION-OS mysql
cd mysql
./scripts/mariadb-install-db --defaults-file=~/.my.cnf
```

If you have problems with the above gunzip command line, you can instead, if you have gnu tar, do:

```
tar xfz /path-to/mariadb-VERSION-OS.tar.gz
```

To start mariadbd, you should now do:

```
./bin/mariadbd-safe --defaults-file=~/.my.cnf &
```

Auto Start of mariadbd

You can get mariadbd (the MariaDB server) to autostart by copying the file `mysql.server` file to the right place.

```
cp support-files/mysql.server /etc/init.d/mysql.server
```

The exact place depends on your system. The `mysql.server` file contains instructions on how to use and fine tune it.

For systemd installation, the mariadb.service file will need to be copied from the `support-files/systemd` folder to the `/usr/lib/systemd/system/` folder.

```
cp support-files/systemd/mariadb.service /usr/lib/systemd/system/mariadb.service
```

After this, you can start the service using this command:

```
systemctl start mariadb.service
```

Stop the service with this command:

```
systemctl stop mariadb.service
```

Please refer to the systemd page for further information.

Post Installation

After this, remember to set proper passwords for all accounts accessible from untrusted sources, to avoid exposing the host to security risks!

Also consider using the mysql.server to start MariaDB automatically when your system boots.

On systems using systemd, instead, enable automatic startup during system boot with this command:

```
systemctl enable mariadb.service
```

For details on the exact steps used to build the binaries, see the compiling MariaDB section.

This page is licensed: CC BY-SA / Gnu FDL

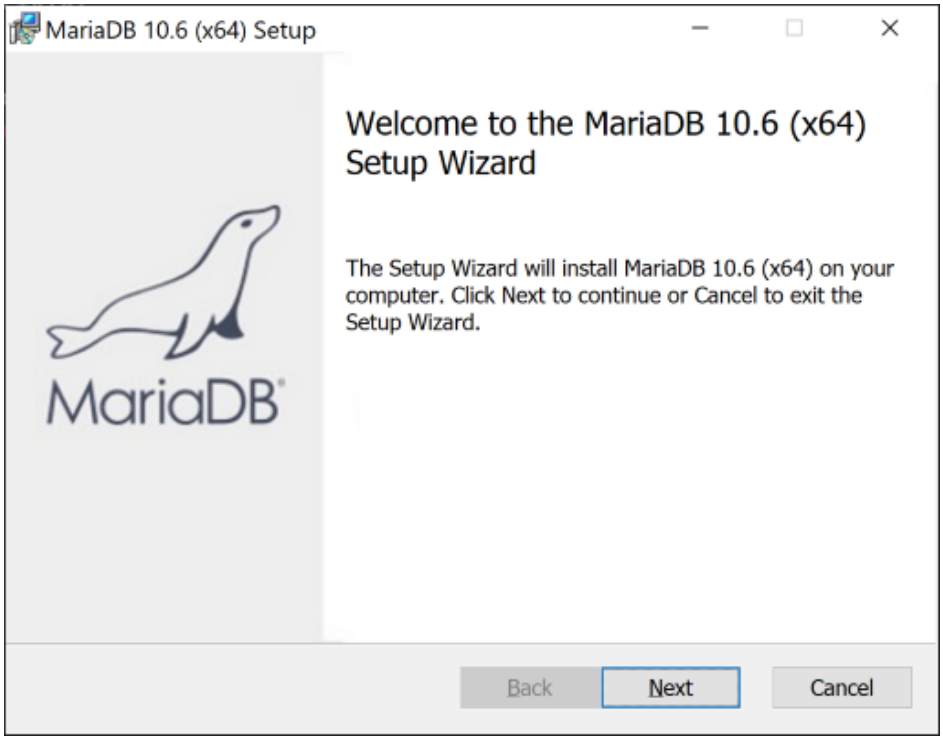
Installing MariaDB MSI Packages on Windows

MSI packages is available for x64 (64 bit) processor architectures and, in some older releases only, for x86 (32 bit). We use screenshots from an x64 installation in the following.

Installation UI

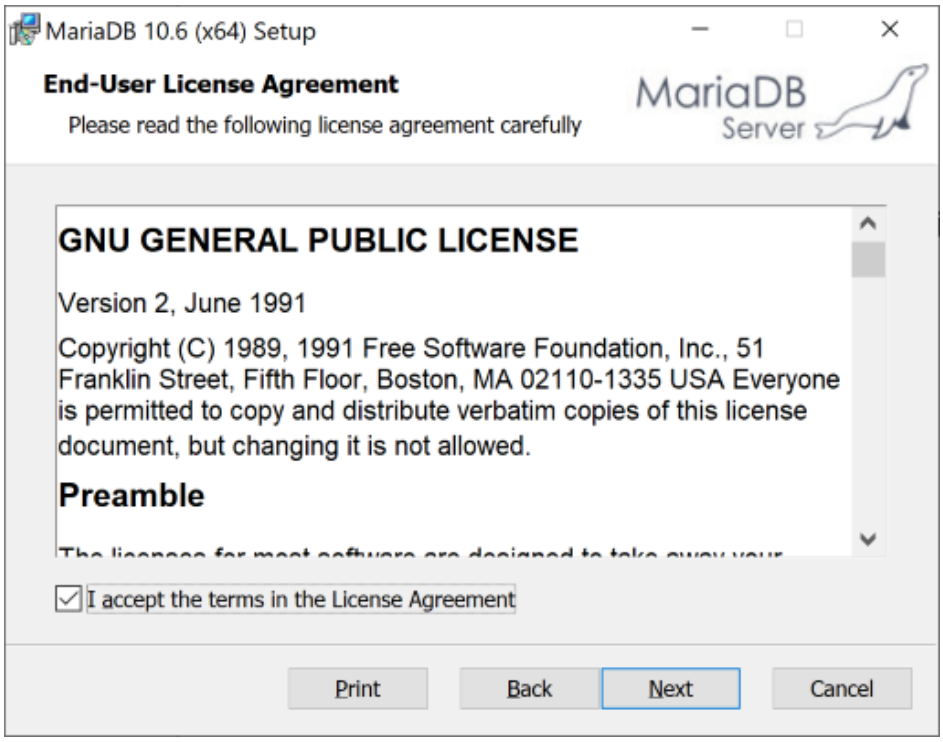
This is the typical mode of installation. To start the installer, click on the mariadb-…msi.

Welcome



Welcome dialog

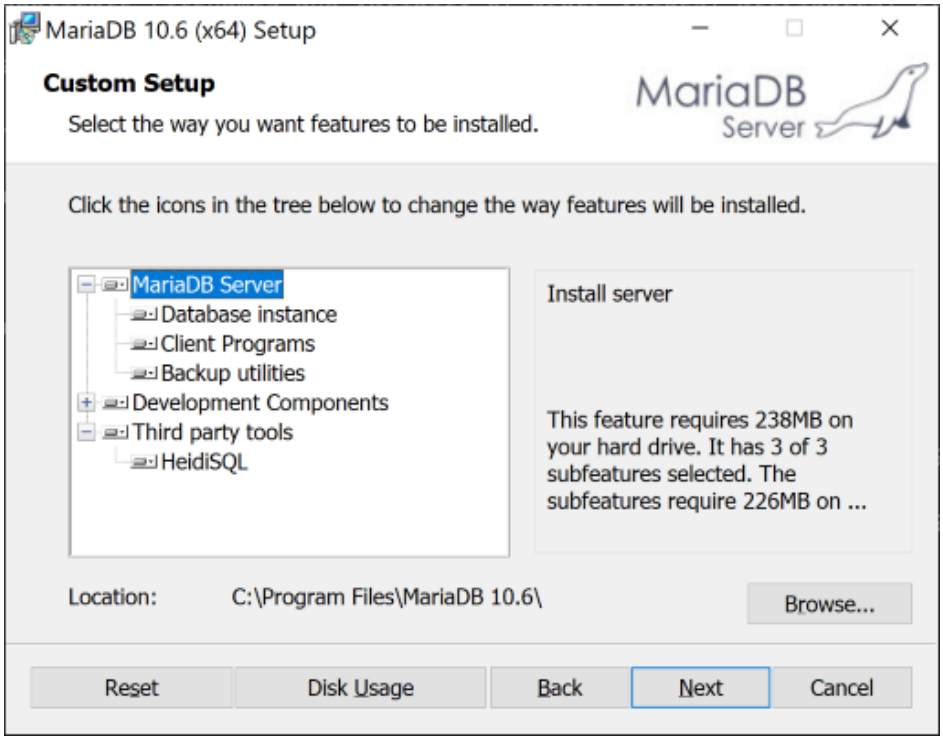
License Agreement



License Agreement

You must accept the terms in the license agreement. Proceed to the next dialog.

Custom Setup

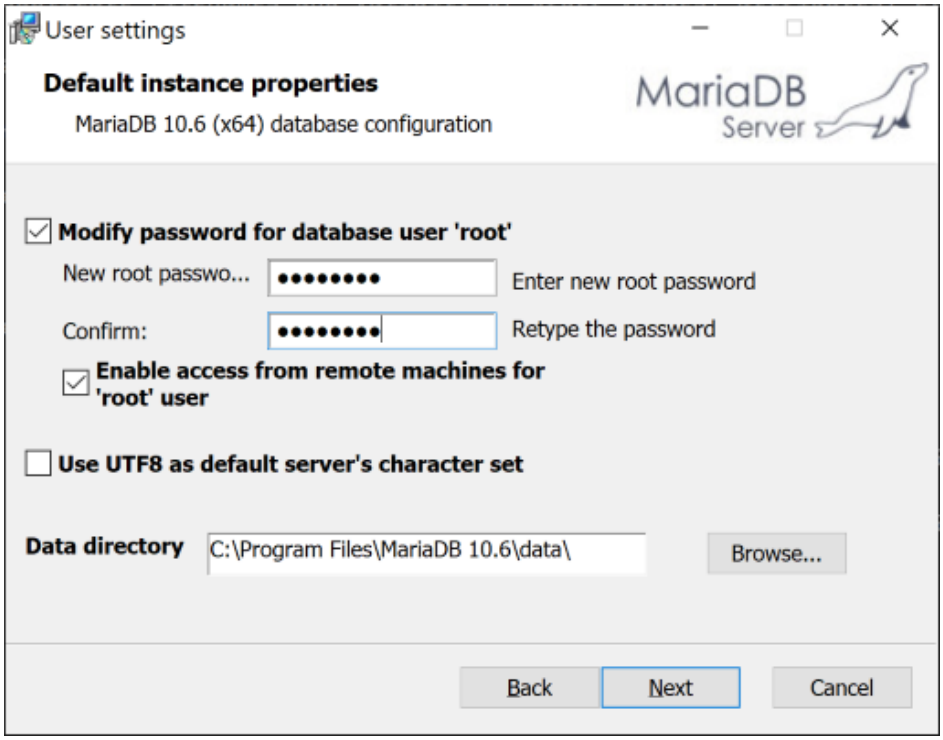


Custom Setup

Choose what features to install. By default, all features are installed with the exception of the debug symbols. If the "Database instance" feature is selected, the installer will create a database instance, by default running as a service. In this case the installer will present additional dialogs to control various database properties. Note that you do not necessarily have to create an instance at this stage. For example, if you already have MySQL or MariaDB databases running as services, you can just upgrade them during the installation. Also, you can create additional database instances after the installation, with the mysql_install_db.exe utility.

> **Note:**
>
> By default, if you install a database instance, the data directory are in the `data` folder under the installation root. To change the data directory location, select `Database instance` in the feature tree, and use the `Browse` button to select another location.

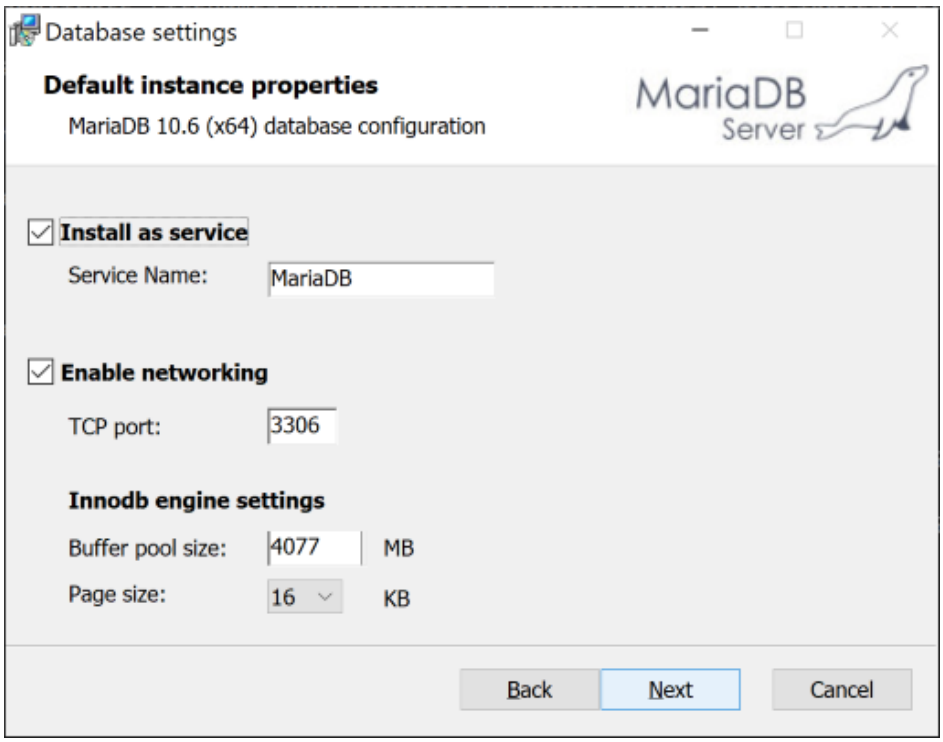
Database Authentication/Security Related Properties



Database security properties

This dialog is shown if you selected the *Database instance* feature. Here, you can set the password for the “root” database user and specify whether root can access databases from remote machines. The *Create anonymous account* setting allows for anonymous (non-authenticated) users. It is off by default; it is not recommended to change this setting.

Other Database Properties

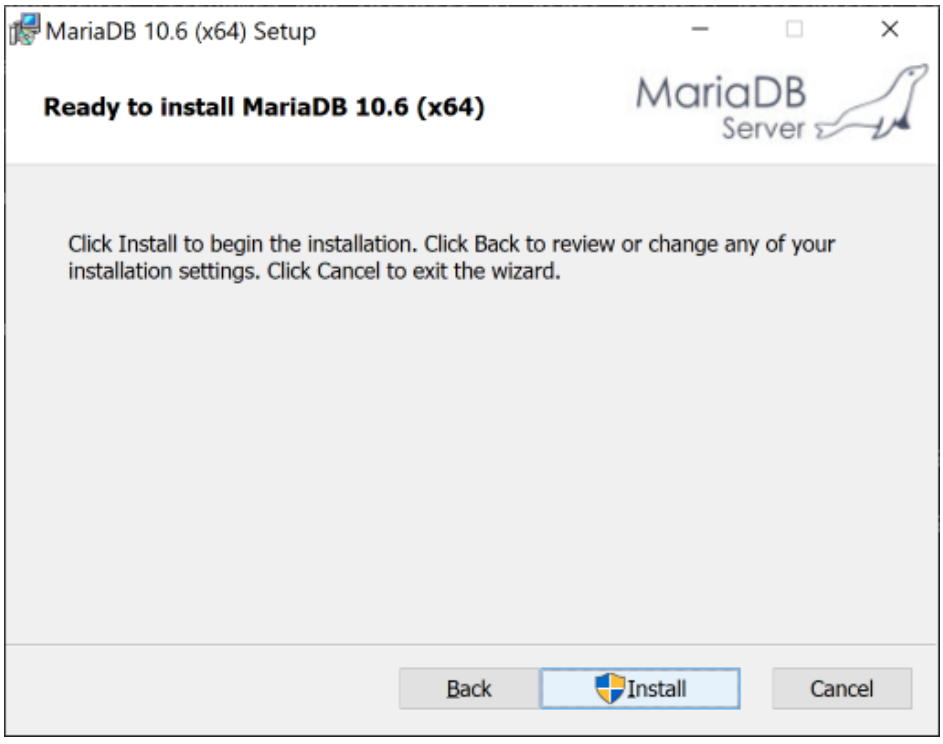


Other database properties

- Install as service.

- Defines whether the database should be run as a service. If it should be run as a service, then it also defines the service name. It is recommended to run your database instance as a service as it greatly simplifies database management. In MariaDB 10.4 and later, the default service name used by the MSI installer is “MariaDB”. In 10.3 and before, the default service name used by the MSI installer is “MySQL”. Note that the default service name for the –install and –install-manual options for `mysqld.exe` is “MySQL” in all versions of MariaDB.
- Enable Networking.
- Whether to enable TCP/IP (recommended) and which port MariaDB should listen to. If security is a concern, you can change the bind-address parameter post-installation to bind to only local addresses. If the “Enable networking” checkbox is deselected, the database will use named pipes for communication.
- InnoDB engine settings.
- Defines the InnoDB buffer pool size, and the InnoDB page size. The default buffer pool size is 12.5% of RAM, and depending on your requirements you can give InnoDB more (up to 70-80% RAM). 32 bit versions of MariaDB have restrictions on maximum buffer pool size, which is approximately 1GB, due to virtual address space limitations for 32bit processes. A 16k page size is suitable for most situations. See the innodb_page_size system variable for details on other settings.

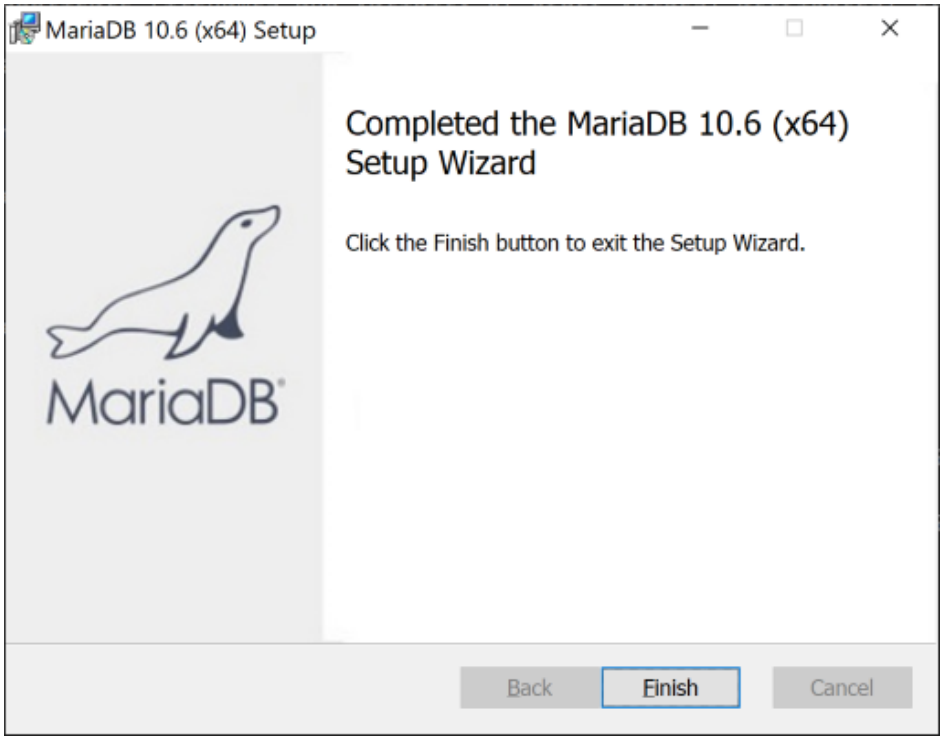
Ready to Install



Ready Dialog

At this point, all installation settings are collected. Click on the “Install” button.

End



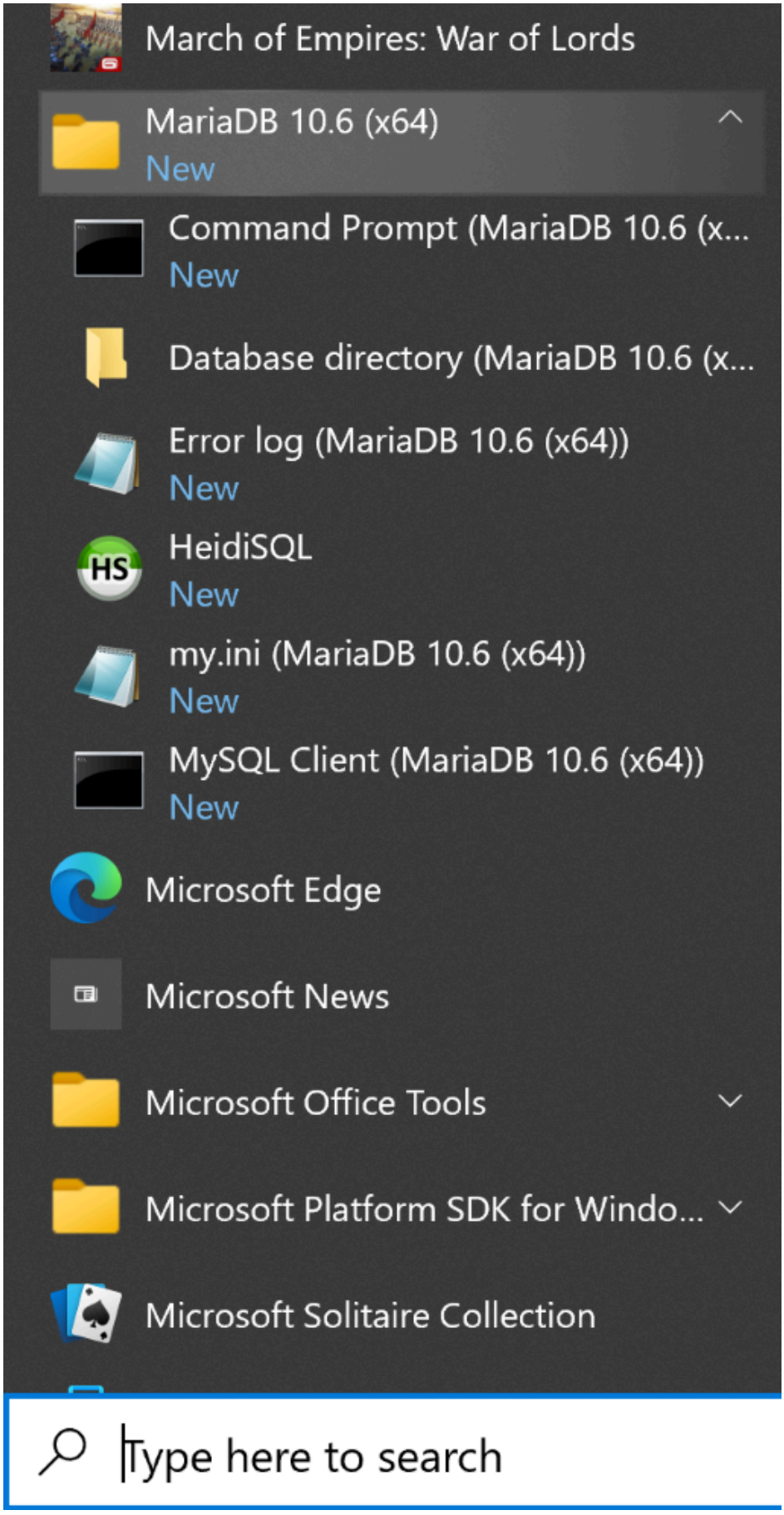
Finish

Installation is finished now. If you have upgradable instances of MariaDB/MySQL, running as services, this dialog will present a “Do you want to upgrade existing instances” checkbox (if selected, it launches the Upgrade Wizard post-installation).

If you installed a database instance as service, the service will be running already.

New Entries in Start Menu

Installation will add some entries in the Start Menu:



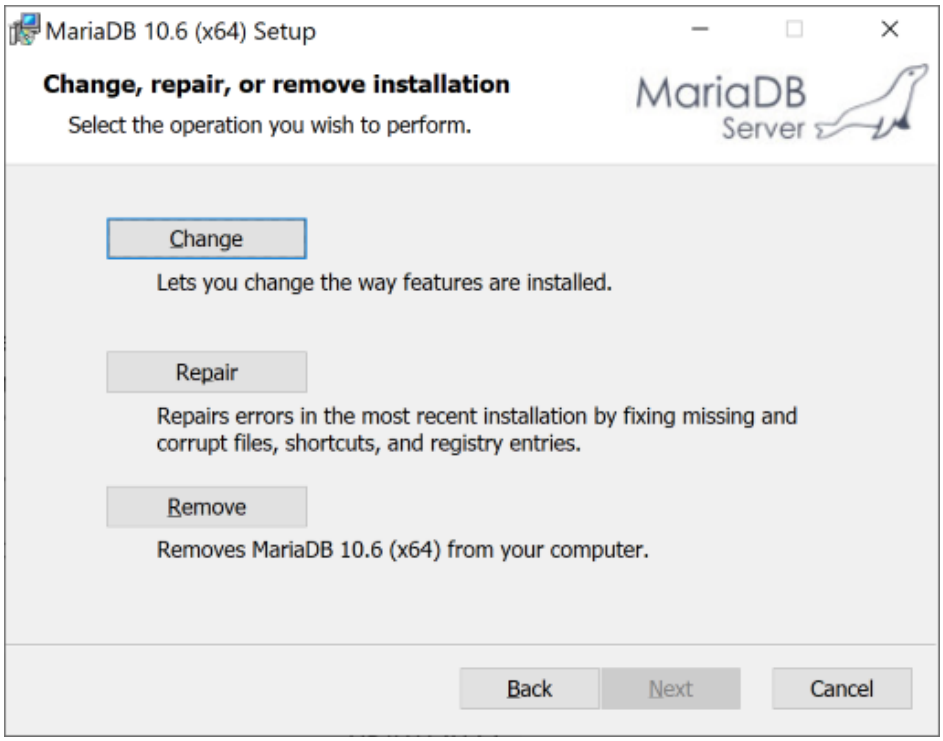
Start Menu

- **MariaDB Client** - Starts command line client mysql.exe.

- **Command Prompt** - Starts a command prompt. Environment is set such that “bin” directory of the installation is included into PATH environment variable, for instance, use this command prompt to issue MariaDB commands (for example, mysqldadmin or mysql).
- **Database directory** - Opens the data directory in Explorer.
- **Error log** - Opens the database error log in Notepad.
- **my.ini** - Opens the database configuration file my.ini in Notepad.
- **Upgrade Wizard** - Starts the Wizard to upgrade an existing MariaDB/MySQL database instance to this MariaDB version.

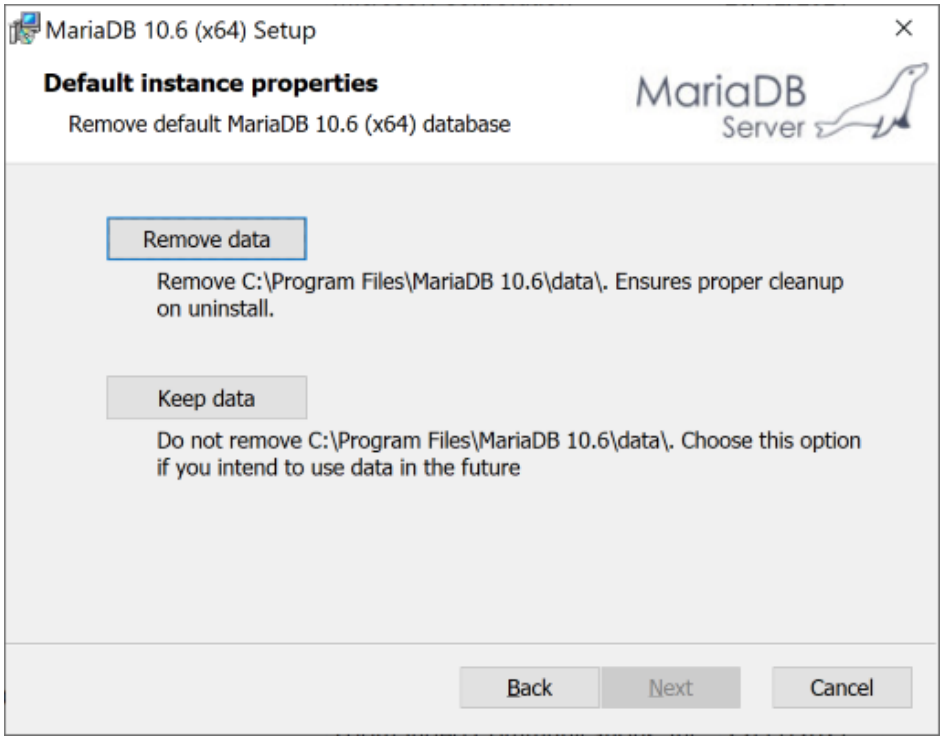
Uninstall UI

In the Explorer applet *Programs and Features*, find the entry for MariaDB, choose *Uninstall/Change* and click on the `Remove` button in the dialog:



UninstallChangeDialog_New

If you installed a database instance, you will need to decide if you want to remove or keep the data in the database directory.



KeepOrRemoveDataDialog_New

Silent Installation

The MSI installer supports silent installations as well. In its simplest form silent installation with all defaults can be performed from an elevated command prompt like this:

```
msiexec /i path-to-package.msi /qn
```

> **Note:**
>
> The installation is silent due to msiexe.exe's /qn switch (no user interface). If you remove the switch, the installation has the full UI.

Properties

Silent installations also support installation properties (a property would correspond for example to checked/unchecked state of a checkbox in the UI, user password, etc). With properties the command line to install the MSI package would look like this:

```
msiexec /i path-to-package.msi [PROPERTY_1=VALUE_1 ... PROPERTY_N=VALUE_N] /qn
```

The MSI installer package requires property names to be all capitals and contain only English letters. By convention, for a boolean property, an empty value means "false" and a non-empty is "true".

MariaDB installation supports the following properties:

Property name	Default value	Description
INSTALLDIR	%ProgramFiles%\MariaDB \	Installation root
PORT	3306	–port parameter for the server
ALLOWREMOTEROOTACCESS		Allow remote access for root user
BUFFERPOOLSIZE	RAM/8	Bufferpoolsize for innodb

Property name	Default value	Description
CLEANUPDATA	1	Remove the data directory (uninstall only)
DATADIR	INSTALLDIR\data	Location of the data directory
DEFAULTUSER		Allow anonymous users
PASSWORD		Password of the root user
SERVICENAME		Name of the Windows service. A service is not created if this value is empty.
SKIPNETWORKING		Skip networking
STDCONFIG	1	Corresponds to “optimize for transactions” in the GUI, default engine innodb, strict sql mode
UTF8		if set, adds character-set-server=utf8 to my.ini file
PAGESIZE	16K	page size for innodb

Features

Feature is a Windows installer term for a unit of installation. Features can be selected and deselected in the UI in the feature tree in the “Custom Setup” dialog.

Silent installation supports adding features with the special property `ADDLOCAL=Feature_1,..,Feature_N` and removing features with `REMOVE=Feature_1,..., Feature_N` .

Features in the MariaDB installer:

Feature id	Installed by default?	Description
DBInstance	yes	Install database instance
Client	yes	Command line client programs
MYSQLSERVER	yes	Install server
SharedLibraries	yes	Install client shared library
DEVEL	yes	install C/C++ header files and client libraries
HeidiSQL	yes	Installs HeidiSQL

Silent Installation Examples

All examples here require running as administrator with elevated command line privileges.

- Install default features, database instance as service, non-default datadir and port:

```
msiexec /i path-to-package.msi SERVICENAME=MySQL DATADIR=C:\mariadb5.2\data PORT=3307 /qn
```

- Install service, add debug symbols, do not add development components (client libraries and headers).

```
msiexec /i path-to-package.msi SERVICENAME=MySQL ADDLOCAL=DEBUGSYMBOLS REMOVE=DEVEL /qn
```

Silent Uninstall

To uninstall silently, use the `REMOVE=ALL` property with msiexec:

```
msiexec /i path-to-package.msi REMOVE=ALL /qn
```

To keep the data directory during an uninstall, you will need to pass an additional parameter:

```
msiexec /i path-to-package.msi REMOVE=ALL CLEANUPDATA="" /qn
```

Installation Logs

If you encounter a bug in the installer, use the installer logs for diagnosis. Please attach verbose logs to the bug reports you create. To create a verbose installer log, start the installer from the command line with the `/l*v` switch, like so:

```
msiexec.exe /i path-to-package.msi  /l*v path-to-logfile.txt
```

Running 32 and 64 Bit Distributions on the Same Machine

It is possible to install 32 and 64 bit packages on the same Windows x64.

Apart from testing, an example where this feature can be useful is a development scenario, where users want to run a 64 bit server and develop both 32 and 64 bit client components. In this case the full 64 bit package can be installed, including a database instance plus development-related features (headers and libraries) from the 32 bit package.

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB on macOS

MariaDB Server is available for installation on macOS via the Homebrew package manager. MariaDB Server (together with many client programs and helper tools) is available as a Homebrew “bottle”, a precompiled package. If you haven’t yet installed Homebrew, see this section.

Installing & Starting MariaDB

Install MariaDB Server:

```
brew install mariadb
```

> **Note:**
>
> This can take several minutes to complete, depending on hardware, network, and other factors.

Start MariaDB Server:

```
mysql.server start
```

Alternatively, and strongly recommended, automatically start MariaDB Server:

```
brew services start mariadb
```

> **Note:**
>
> Automatically starting MariaDB server installs a background service on macOS. Make sure to allow adding that background service. See this section for more information.

Connecting to MariaDB Server

After MariaDB Server has started, you can connect to the server using the shell user name (see this section for information on the user):

```
mariadb
```

Alternatively, connect as root:

```
sudo mysql -u root
```

For graphical clients you can use instead of the `mariadb` command-line client, see this section.

Upgrading MariaDB

Update Homebrew packages:

```
brew update
```

Then upgrade MariaDB Server:

```
brew upgrade mariadb
```

Notes & Further Information

Homebrew Installation

Install Homebrew like this:

1. Open a Terminal (`⌘ + Space` to open *Spotlight*, type *Terminal*).
2. Issue this command:
   `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`
   1. Alternatively, use the package installer (`.pkg` ; at the time of writing this, https://github.com/Homebrew/brew/releases/download/5.0.4/Homebrew-5.0.4.pkg)
3. Refer to the Homebrew website for more information, particularly to the Homebrew documentation.

MariaDB Configuration

In Homebrew, the configuration file for MariaDB is located at:

- `/usr/local/etc/my.cnf` for Intel-based Macs.
- `/opt/homebrew/etc/my.cnf` for Apple Silicon Macs (ARM architecture).

MariaDB Information

Find information about the MariaDB version, analytics, and more, using the `brew info` command:

```
~> brew info mariadb
==> mariadb: stable 12.1.2 (bottled)
Drop-in replacement for MySQL
https://mariadb.org/
...
To restart mariadb after an upgrade:
  brew services restart mariadb
Or, if you don't want/need a background service you can just run:
  /opt/homebrew/opt/mariadb/bin/mariadbd-safe --datadir\=/opt/homebrew/var/mysql
==> Analytics
install: 6,319 (30 days), 12,735 (90 days), 62,444 (365 days)
install-on-request: 6,291 (30 days), 12,670 (90 days), 62,137 (365 days)
build-error: 8 (30 days)
```

MariaDB Programs

MariaDB Server (`mariadbd`), the MariaDB command-line client (`mariadb`), and many more clients and tools are installed in `/opt/homebrew/Cellar/mariadb` (for Apple Silicon Macs). Find the location for your machine, as well as the MariaDB programs installed, with these commands:

```
~> which mariadb
/opt/homebrew/bin/mariadb # in the next command, use this location to cd to
~> cd /opt/homebrew/bin/; ls -1 maria*
mariabackup
mariadb
mariadb-access
...
mariadbd
mariadbd-multi
mariadbd-safe
mariadbd-safe-helper
```

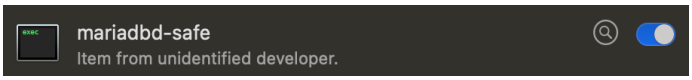
Terminal User & MariaDB User

To find out which user is used, issue these commands in a shell like *Terminal*:

```
~> users
myuser
~> mariadb -e "SELECT USER()"
+------------------+
| USER()           |
+------------------+
| myuser@localhost |
+------------------+
```

macOS Background Service

If you start MariaDB automatically, a macOS background service is added. You can find the MariaDB background service in `System Settings > General > Login Items & Extensions`. It's named `mariadbd-safe`.



The toggle switch allows you to turn off the automatic start of MariaDB. This prevents MariaDB Server from automatically starting once you reboot macOS.

To review the resource usage of MariaDB Server, use this command (type `q` to exit `top` when done):

```
top -pid $(pgrep mariadbd)
```

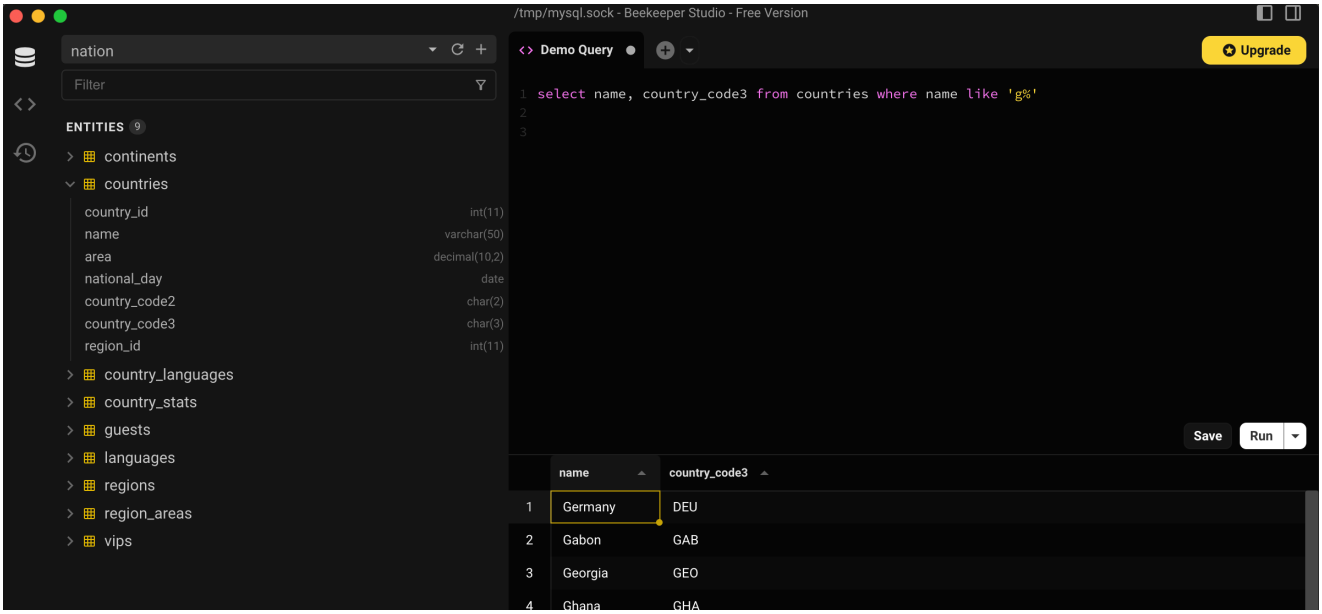
Graphical Clients

MariaDB doesn't offer graphical clients for working with MariaDB Server, but there are many third-party graphical clients, some of which run on macOS. One of those is Beekeeper Studio, a subscription-based client that has a (not too) limited free version, though.

Assuming a standard Homebrew installation of MariaDB, and assuming you connect to MariaDB Server using the standard shell user, configure Beekeeper Studio like this:

- Connection type: MariaDB
- Authentication method: Username/Password
- Connection mode: Socket
- Socket path: `/tmp/mysql.sock`
- Username: (name of the shell user, without the `@localhost` suffix)

Once connected to MariaDB Server, you can run queries in Beehive Studio:



The query shown in this screenshot uses a MariaDB sample database called *nation* which you can use to get familiar with MariaDB. See this section for more information.

MariaDB Sample Database

MariaDB offers a sample database you can use to get familiar with using MariaDB. You can download it here: https://mariadbtutorial.com/wp-content/uploads/2019/10/nation.zip

Unzip `nation.zip`, then import the database into MariaDB Server, using this command (assuming you downloaded and unzipped the sample database in the `Downloads` folder):

```
mariadb < Downloads/nation.sql
```

When done, use that database in the `mariadb` command-line client, like this:

```
~> mariadb nation
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 14
Server version: 12.1.2-MariaDB Homebrew

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [nation]> SHOW TABLES;
+------------------+
| Tables_in_nation  |
+------------------+
| continents        |
| countries         |
| country_languages |
| country_stats     |
| guests            |
| languages         |
| region_areas      |
| regions           |
| vips              |
+------------------+
9 rows in set (0.001 sec)
```

Alternatively, open the database in a graphical client.

Building MariaDB Server from source

In addition to the “bottled” MariaDB Server package available from Homebrew, you can use Homebrew to build MariaDB from source. This is useful if you want to use a different version of the server or enable some different capabilities that are not included in the bottle package.

Two components not included in the bottle package are the CONNECT and OQGRAPH engines, because they have non-standard dependencies. To build MariaDB Server with these engines, you must first install `boost` and `judy`. Follow these steps to install the dependencies and build the server:

```
brew install boost judy
brew install mariadb --build-from-source
```

You can also use Homebrew to build and install a pre-release version of MariaDB Server. Use this command to build and install a “development” version of MariaDB Server:

```
brew install mariadb --devel
```

Other resources

- mariadb.rb on github
- MariaDB Server on macOS: Does it even make sense to try? (video)

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB on macOS (PKG)

MariaDB Server does not currently provide a `.pkg` installer for macOS. For information about how to install MariaDB Server on macOS using Homebrew, see Installing MariaDB Server on macOS Using Homebrew.

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB Windows ZIP Packages

Users need to run mariadb-install-db.exe, without parameters to create a data directory, e.g

```
C:\zip_unpack\directory> bin\mariadb-install-db.exe
```

Then you can start server like this

```
C:\zip_unpack\directory> bin\mariadbd.exe --console
```

If you like to customize the server instance (data directory, install as service etc), please refer to mariadb-install-db.exe documentation

This page is licensed: CC BY-SA / Gnu FDL

Deploy with Package Tarballs

MariaDB Corporation provides package tarballs for some MariaDB database products.

Package tarballs provide multiple benefits:

- Package tarballs are compressed tar archives that contain software packages.
- Software packages can be installed using the operating system's package manager without relying on a remote repository.
- RPM (.rpm) files are distributed for CentOS, Red Hat Enterprise Linux (RHEL), and SUSE Linux Enterprise Server (SLES).
- DEB (.deb) files are distributed for Debian and Ubuntu.

If you want to deploy MariaDB database products without using a package tarball, alternative deployment methods are available. Available deployment methods are component-specific.

Use Cases

MariaDB database products can be deployed with package tarballs to support use cases, such as:

- Transfer the package tarball to an air-gapped network for installation without an internet connection
- Install software using a package manager without configuring a package repository
- Automatically install missing dependencies using a package manager

Compatibility

The following MariaDB database products can be deployed using package tarballs:

- MariaDB Community Server 10.5
- MariaDB Community Server 10.6
- MariaDB Enterprise Server 10.5
- MariaDB Enterprise Server 10.6
- MariaDB Enterprise Server 11.4
- MariaDB MaxScale 22.08

Download a Package Tarball

MariaDB Corporation provides package tarballs (.debs.tar, .rpms.tar) to support customers who leverage in-house package repositories to distribute software to their servers. Secure any such repository to prevent outside access.

MariaDB Corporation provides multiple interfaces to download package tarballs.

> **WEB BROWSER**

1. Visit the MariaDB Downloads page.
2. Complete customer login.
3. Select the desired version and operating system.
4. Click the Download button.

COMMAND LINE OR AUTOMATION

Package tarballs can be downloaded using command-line tools or automation from the MariaDB Download interface with the Customer Download Token.

For additional information, see "Download Binary Files".

Installation From Package Tarball

Once downloaded and extracted, you can:

- Install .rpm packages (RHEL, CentOS, and SLES): `rpm -i`
- Install .deb packages (Debian, Ubuntu): `dpkg -i`
- Install from the simple package repositories included in the tarball. Missing dependencies will be resolved when using the `apt`, `yum`, or `zypper` package manager.
  See the `README file` enclosed in the package tarball for more information.
- Test packages before placement in an internal package repository for distribution to your servers. Secure this repository from outside access.

Installation loads software to the system. This software requires configuration before the database server is ready for use.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Deploy with Local Package Repository Mirrors

MariaDB Corporation provides package repositories, including the MariaDB Enterprise Repository and the MariaDB Community Repository, that can be used to install MariaDB products using the operating system's package manager. Local mirrors of the package repositories can be used for local deployments.

Local package repository mirrors provide multiple benefits:

- MariaDB Corporation's official package repositories are the source for the local mirror.
- Tools provided by the operating system are used to create and maintain the local mirror.
- After a local mirror is created, the mirror can be used just like the MariaDB repositories to install MariaDB products using the operating system's package manager.

If you want to deploy MariaDB database products without using a local package repository mirror, alternative deployment methods are available. Available deployment methods are component-specific.

Use Cases

MariaDB database products can be deployed with local package repository mirrors to support use cases, such as:

- Install from the mirror on an air-gapped network that is not connected to the internet
- Remove packages from mirror for versions that are not used in the local environment
- Add packages to mirror for tools and clients that are used in the local environment
- Automatically install missing dependencies using a package manager

Compatibility

The following MariaDB database products can be deployed using package repositories:

- MariaDB Community Server 10.6
- MariaDB Community Server 10.11
- MariaDB Community Server 11.4
- MariaDB Community Server 11.8
- MariaDB Community Server 12.3
- MariaDB Enterprise Server 10.6
- MariaDB Enterprise Server 11.4
- MariaDB Enterprise Server 11.8
- MariaDB MaxScale 23.02
- MariaDB MaxScale 23.08
- MariaDB MaxScale 24.02
- MariaDB MaxScale 25.01

Operating System Package Managers

The package manager depends on the operating system:

Operating System	Package Manager
Debian 11	APT
Debian 12	APT
Debian 13	APT
Red Hat Enterprise Linux 8 (RHEL 8)	DNF
Red Hat Enterprise Linux 9 (RHEL 9)	DNF
Red Hat Enterprise Linux 10 (RHEL 10)	DNF
Rocky Linux 8	DNF
Rocky Linux 9	DNF
Rocky Linux 10	DNF
SUSE Linux Enterprise Server 12 (SLES 12)	ZYpp
SUSE Linux Enterprise Server 15 (SLES 15)	ZYpp
Ubuntu 22.04 LTS (Jammy)	APT
Ubuntu 24.04 LTS (Noble)	APT

Create a Local Repository Mirror

Creating a local mirror of the MariaDB Enterprise Repository or the MariaDB Community Repository enables you to distribute MariaDB products to your servers from a local repository you support. Secure any such repository mirror to prevent outside access.

1. Configure a MariaDB repository
2. Set up a repository mirroring tool, for example:
   - YUM: `reposync`, available at: 23016
   - APT: `debmirror`, available at: Setup#Debian_Repository_Mirroring_Tools
3. Secure the repository mirror to prevent outside access.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Installing System Tables

See Installing System Tables on Unix.

Instructions for running the `mariadb-install-db` script on Unix-like systems to initialize the MariaDB data directory and system tables.

See Installing System Tables on Windows.

Details the use of `mariadb-install-db.exe` on Windows to create new database instances, set the root password, and register Windows services.

This page is licensed: CC BY-SA / Gnu FDL

Installing System Tables on Unix

`mariadb-install-db` initializes the MariaDB data directory and creates the system tables in the mysql database, if they do not exist. MariaDB uses these tables to manage privileges, roles, and plugins. It also uses them to provide the data for the help command in the mariadb client.

mariadb-install-db works by starting MariaDB Server's `mysqld` process in –bootstrap mode and sending commands to create the system tables and their content.

There is a version specifically for Windows, mysql_install_db.exe.

To invoke `mariadb-install-db`, use the following syntax:

```
mariadb-install-db --user=mysql
```

For the options supported by mariadb-install-db, see mariadb-install-db: Options.

For the option groups read by mariadb-install-db, see mariadb-install-db: Option Groups.

See mariadb-install-db: Installing System Tables for information on the installation process.

See mariadb-install-db: Troubleshooting Issues for information on how to troubleshoot the installation process.

See Also

- mariadb-install-db
- The Windows version of `mariadb-install-db`: mysql_install_db.exe

This page is licensed: CC BY-SA / Gnu FDL

Installing System Tables on Windows

The `mariadb-install-db.exe` utility is the Windows equivalent of mariadb-install-db.

Functionality

The functionality of `mariadb-install-db.exe` is comparable with the shell script `mariadb-install-db` used on Unix, however it has been extended with both Windows specific functionality (creating a Windows service) and to generally useful functionality. For example, it can set the 'root' user password during database creation. It also creates the `my.ini` configuration file in the data directory and adds most important parameters to it (e.g port).

`mariadb-install-db.exe` is used by the MariaDB installer for Windows if the "Database instance" feature is selected. It obsoletes similar utilities and scripts that were used in the past such as `mysqld.exe --install`, `mysql_install_db.pl`, and `mysql_secure_installation.pl`.

Parameter(s)	Description
`-?`, `--help`	Display help message and exit
`-d`, `--datadir=name`	Data directory of the new database
`-S`, `--service=name`	Name of the `Windows` service

Parameter(s)	Description
`-p`, `--password=name`	Password of the `root` user
`-P`, `--port=#`	`mysqld` port
`-W`, `--socket=name`	`named pipe` name
`-D`, `--default-user`	Create default user
`-R`, `--allow-remote-root-access`	Allow remote access from network for user `root`
`-N`, `--skip-networking`	Do not use `TCP` connections, use `pipe` instead
`-i`, `--innodb-page-size`	`Innodb` page size, since `MariaDB 10.2.5`
`-s`, `--silent`	Print less information
`-o`, `--verbose-bootstrap`	Include `mysqld` bootstrap output
`-l`, `--large-pages`	Use large pages, since `MariaDB 10.6.1`
`-c`, `--config`	`my.ini` config template file, since `MariaDB 10.6.1`

> **Note:**
>
> `--config` takes a **template** option file: `mariadb-install-db.exe` copies it into the data directory as `my.ini` and then appends the server settings it generates (such as `datadir` and `port`). The template must be a valid option file. Note that `config` is a command-line option of this tool, not a server variable, so it must not appear as a setting inside the template.

> **Note:**
>
> **T**o create a Windows service, `mariadb-install-db.exe` should be run by a user with full administrator privileges (which means elevated command prompt on systems with UAC).

Example

```
mariadb-install-db.exe --datadir=C:\db --service=MyDB --password=secret
```

will create the database in the directory `C:\db`, register the auto-start Windows service “MyDB”, and set the root password to ‘secret’.

To start the service from the command line, execute

```
sc start MyDB
```

Removing Database Instances

If you run your database instance as service, to remove it completely from the command line, use

```
sc stop <servicename>
sc delete <servicename>
rmdir /s /q <path-to-datadir>
```

This page is licensed: CC BY-SA / Gnu FDL

Troubleshooting Installation Issues

See Error: symbol mysql_get_server_name, version libmysqlclient_16 not defined.

Troubleshooting guide for a specific linker error involving `mysql_get_server_name` and `libmysqlclient_16`, typically occurring due to library version mismatches.

See Installation issues on Windows.

Common installation problems on Windows, such as issues with User Account Control (UAC) or unsupported Windows versions, and their solutions.

See Installation Issues with PHP5.

Addresses compatibility issues between MariaDB and older PHP5 client libraries, specifically regarding header and library version mismatches.

See Installing on an Old Linux Version.

Guidance for dealing with glibc or shared library errors when attempting to install modern MariaDB binaries on older Linux distributions.

See Installation Issues on Debian and Ubuntu.

A collection of troubleshooting articles specific to Debian and Ubuntu deployments, covering upgrade failures, repository conflicts, and migration issues.

This page is licensed: CC BY-SA / Gnu FDL

Error: symbol mysql_get_server_name, version libmysqlclient_16 not defined

If you see the error message:

```
symbol mysql_get_server_name, version libmysqlclient_16 not defined in file libmysqlclient.so.16 with
link time reference
```

…then you are probably trying to use the mysql command-line client from MariaDB with `libmysqlclient.so` from MySQL.

The symbol `mysql_get_server_name()` is something present in the MariaDB source tree and not in the MySQL tree.

If you have both the MariaDB client package and the MySQL client packages installed this error will happen if your system finds the MySQL version of `libmysqlclient.so` first.

To figure out which library is being linked in dynamically (for instance, the wrong one) use the 'ldd' tool.

```
ldd $(which mysql) | grep mysql
```

or

```
ldd /path/to/the/binary | grep mysql
```

For example:

```
me@mybox:~$ ldd $(which mysql)|grep mysql
        libmysqlclient.so.16 => /usr/lib/libmysqlclient.so.16 (0xb74df000)
```

You can then use your package manager's tools to find out which package the library belongs to.

On CentOS the command to find out which package installed a specific file is:

```
rpm -qf /path/to/file
```

On Debian-based systems, the command is:

```
dpkg -S /path/to/file
```

Here's an example of locating the library and finding out which package it belongs to on an Ubuntu system:

```
me@mybox:~$ ldd $(which mysql)|grep mysql
    libmysqlclient.so.16 => /usr/lib/libmysqlclient.so.16 (0xb75f8000)
me@mybox:~$ dpkg -S /usr/lib/libmysqlclient.so.16
libmariadbclient16: /usr/lib/libmysqlclient.so.16
```

The above shows that the mysql command-line client is using the library `/usr/lib/libmysqlclient.so.16` and that library is part of the `libmariadbclient16` Ubuntu package. Unsurprisingly, the mysql command-line client works perfectly on this system.

If the answer that came back had been something other than a MariaDB package, then it is likely there would have been issues with running the MariaDB mysql client application.

If the library that the system tries to use is not from a MariaDB package, the remedy is to remove the offending package (and possibly install or re-install the correct package) so that the correct library can be used.

This page is licensed: CC BY-SA / Gnu FDL

Installation issues on Windows

Unsupported Versions of Windows

Recent versions of MariaDB may not install on unsupported Windows versions. See Deprecated Package Platforms to find the final supported versions.

Systems with User Account Control

Running `mysql_install_db.exe` from a standard command prompt might cause the error:

```
FATAL ERROR: OpenSCManager failed
```

To get rid of it, use the elevated command prompt, for example on Windows 7 start it via 'Run as administrator' option.

This page is licensed: CC BY-SA / Gnu FDL

Installation Issues with PHP5

PHP5 may give an error if used with the old connect method:

```
'mysql_connect(): Headers and client library minor version mismatch. Headers:50156 Library:50206'
```

This is because the library wrongly checks and expects that the client library must be the exact same version as PHP was compiled with. You would get the same error if you tried to upgrade just the MySQL client library without upgrading PHP at the same time.

Ways to fix this issue:

1. Switch to using the mysqlnd driver in PHP (Recommended solution).
2. Run with a lower error reporting level:

```
$err_level = error_reporting(0);
$conn = mysql_connect('params');
error_reporting($err_level);
```

1. Recompile PHP with the MariaDB client libraries.
2. Use your original MySQL client library with the MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

Installing on an Old Linux Version

This article lists some typical errors that may happen when you try to use an incompatible MariaDB binary on a linux system:

The following example errors are from trying to install MariaDB built for SuSE 11.x on a SuSE 9.0 server:

```
> scripts/mysql_install_db
./bin/my_print_defaults: /lib/i686/libc.so.6:
  version `GLIBC_2.4' not found (required by ./bin/my_print_defaults)
```

and

```
> ./bin/mysqld --skip-grant &
./bin/mysqld: error while loading shared libraries: libwrap.so.0:
cannot open shared object file: No such file or directory
```

If you see either of the above errors, the binary MariaDB package you installed is not compatible with your system.

The options you have are:

- Find another MariaDB package or tar from the download page that matches your system.
- Alternatively, download the source and build it.

This page is licensed: CC BY-SA / Gnu FDL

Installation Issues on Debian and Ubuntu

See apt-upgrade Fails, But the Database is Running.

Solutions for when `apt-get upgrade` hangs or fails because the MariaDB service takes too long to start, triggering a timeout in the init script.

See Differences in MariaDB in Debian (and Ubuntu).

Explains the differences between official Debian/Ubuntu repository packages and those from MariaDB.org, particularly regarding library linking and configuration defaults.

This page is licensed: CC BY-SA / Gnu FDL

apt-upgrade Fails, But the Database is Running

After running `apt-upgrade mariadb`, it's possible that apt shows a fail in trying to start the server, but in fact the database is up and running, which then provokes apt to remain in a non finished state.

For example:

```
# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
2 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n]
Setting up mariadb-server-10.1 (10.1.10+maria-1~trusty) ...
 * Stopping MariaDB database server mysqld
   ...done.
 * Starting MariaDB database server mysqld
   ...fail!
invoke-rc.d: initscript mysql, action "start" failed.
dpkg: error processing package mariadb-server-10.1 (--configure):
 subprocess installed post-installation script returned error exit status 1
dpkg: dependency problems prevent configuration of mariadb-server:
 mariadb-server depends on mariadb-server-10.1 (= 10.1.10+maria-1~trusty); however:
  Package mariadb-server-10.1 is not configured yet.

dpkg: error processing package mariadb-server (--configure):
 dependency problems - leaving unconfigured
No apport report written because the error message indicates its a followup error from a previous
failure.
Errors were encountered while processing:
 mariadb-server-10.1
 mariadb-server
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

This situation could occur if the timeout for the init script was too short. For example, see MDEV-9382, a situation where the timeout was 30 seconds, but the server was taking 48 seconds to start.

To overcome this, the timeout needs to be increased. This can be achieved as follows:

- On systems where `systemd` is not `enabled/supported`: The timeout can be increased by setting `MYSQLD_STARTUP_TIMEOUT` either directly in the script or via the command line. The init script also sources `/etc/default/mariadb`, so it can also be used to set `MYSQLD_STARTUP_TIMEOUT` to persistently change the startup timeout.
- On systems that support `systemd`: The startup timeout can be increased by setting TimeoutStartSec systemd option.

This page is licensed: CC BY-SA / Gnu FDL

Differences in MariaDB in Debian (and Ubuntu)

The .deb packages provided by MariaDB Foundation's and MariaDB Corporation's repositories are not identical to the official .deb packages provided by Debian's and Ubuntu's default repositories.

The packages provided by MariaDB Foundation's and MariaDB Corporation's repositories are generated using the Debian packaging in MariaDB's official source code. The Debian packaging scripts are specifically in the `debian/` directory.

The packages provided by Debian's and Ubuntu's default repositories are generated using the Debian packaging in Debian's mirror of MariaDB's source code, which contains some custom changes. The source tree can be found here:

- mariadb-server

As a consequence, MariaDB behaves a bit differently if it is installed from Debian's and Ubuntu's default repositories.

Option File Locations

- The option file located at `/etc/mysql/my.cnf` is handled by the update-alternatives mechanism when the `mysql-common` package is installed. It is a symbolic link that references either `mysql.cnf` or `mariadb.cnf` depending on whether MySQL or MariaDB is installed. Most of the MariaDB option files are therefore actually located in `/etc/mysql/mariadb.d/`.

System Variables

Since MariaDB 11.6 there is no system variable difference from the Standard MariaDB.

Variable	MariaDB in Debian	Standard MariaDB	Notes
character_set_server	utf8mb4	latin1 (for <= MariaDB 11.5)	MariaDB 11.6 also defaults to utf8mb4 hence no difference for this version and greater
collation_server	utf8mb4_general_ci (for < MariaDB 11.5.2)	latin1_swedish_ci	character_set_collations has dominant effect hence removal from Debian default settings

Options

Option	MariaDB in Debian	Standard MariaDB	Notes
plugin-load-add	auth_socket.so	-	Before MariaDB 10.4.3, MariaDB did not enable the unix_socket authentication plugin by default.This is default in Debian, allowing passwordless login.

TLS

- MariaDB binaries from .deb packages provided by Debian's and Ubuntu's default repositories are linked with a different TLS library than MariaDB binaries from .deb packages provided by MariaDB Foundation's and MariaDB Corporation's repositories.
- MariaDB Server binaries:
  - MariaDB Server is statically linked with the bundled wolfSSL libraries in .deb packages provided by Debian's and Ubuntu's default repositories.
  - In contrast, MariaDB Server is dynamically linked with the system's OpenSSL libraries in .deb packages provided by MariaDB Foundation and MariaDB Corporation.
- MariaDB client and utility binaries:
  - MariaDB's clients and utilities and MariaDB Connector/C are dynamically linked with the system's GnuTLS libraries in .deb packages provided by Debian's and Ubuntu's default repositories. libmysqlclient is still statically linked with the bundled wolfSSL libraries.
  - MariaDB's clients and utilities and MariaDB Connector/C are dynamically linked with the system's GnuTLS libraries in .deb packages provided by Debian's and Ubuntu's default repositories. libmysqlclient is still statically linked with the bundled yaSSL libraries.
  - In contrast, MariaDB's clients and utilities, libmysqlclient, and MariaDB Connector/C are dynamically linked with the system's OpenSSL libraries in .deb packages provided by MariaDB Foundation's and MariaDB Corporation's repositories.
- See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

Authentication

- The unix_socket authentication plugin is installed by default in new installations that use the .deb packages provided by Debian's default repositories in Debian 9 and later and Ubuntu's default repositories in Ubuntu 15.10 and later.

- The `root@localhost` created by mariadb-install-db will also be created to authenticate via the unix_socket authentication plugin in these builds.

See Also

- Moving from MySQL

More Information

For details, check out the Debian and Ubuntu official repositories:

- search?keywords=mariadb-server&searchon=names&suite=all§ion=all
- search?keywords=mariadb-server&searchon=names&suite=all§ion=all

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB on IBM Cloud

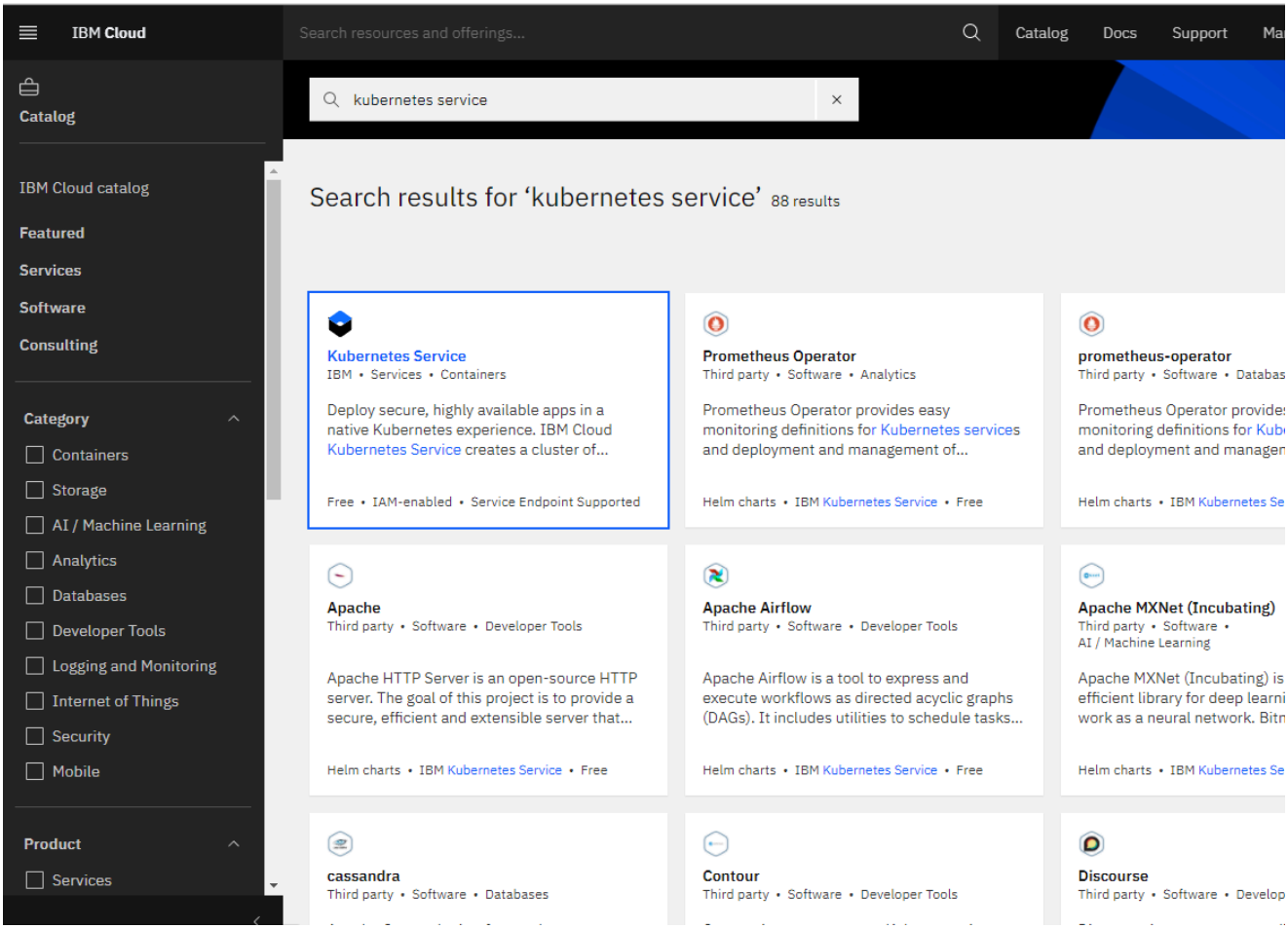
Get MariaDB on IBM Cloud.

You should have an IBM Cloud account, otherwise you can register here.
At the end of the tutorial you will have a cluster with MariaDB up and running. IBM Cloud uses Bitnami charts to deploy MariaDB on with helm

1. We will provision a new Kubernetes Cluster for you if, you already have one skip to step **2**
2. We will deploy the IBM Cloud Block Storage plug-in, if already have it skip to step **3**
3. MariaDB deployment

Step 1 provision Kubernetes Cluster

1. Click the Catalog button on the top.
2. Select Service from the catalog.
3. Search for Kubernetes Service and click on it.

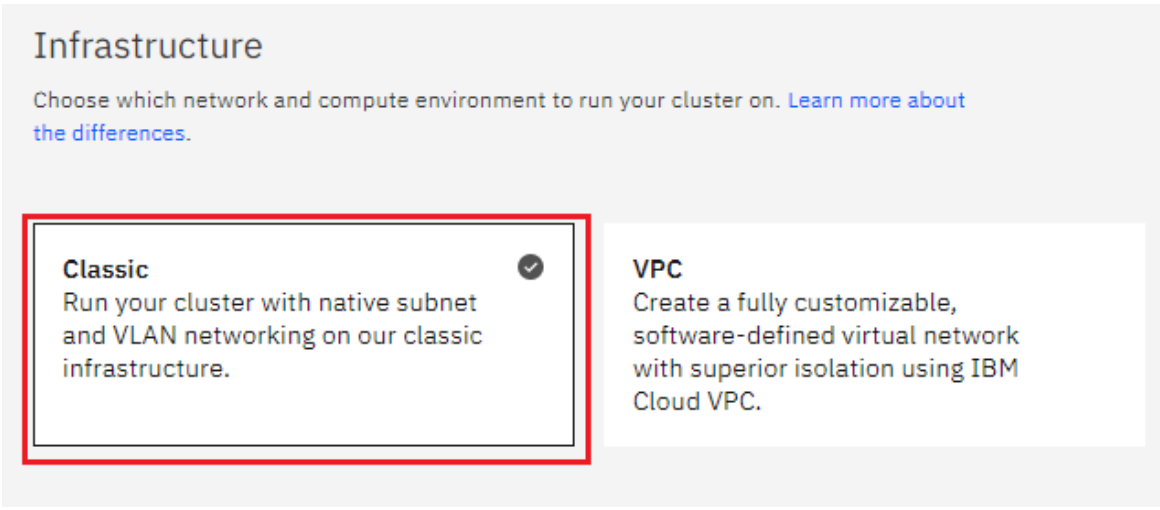


kubernetes-select

4. Choose a plan: standard or free.

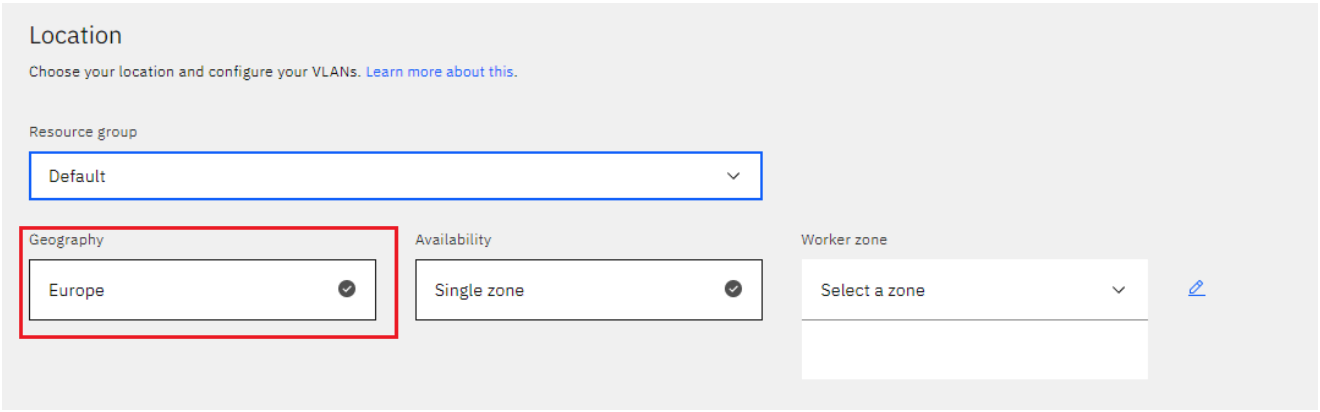
> **Note:**
>
> The free plan includes one worker node and no subnet.

5. To provision a standard cluster, upgrade your account to Pay-As-You-Go.
6. To upgrade to a Pay-As-You-Go account, follow these steps:
    1. In the console, go to **Manage > Account**.
    2. Select **Account settings**, and click **Add credit card**.
    3. Enter your payment information, click **Next**, and submit your information.
7. Choose between classic or VPC after reviewing the documentation to determine the most suitable option.



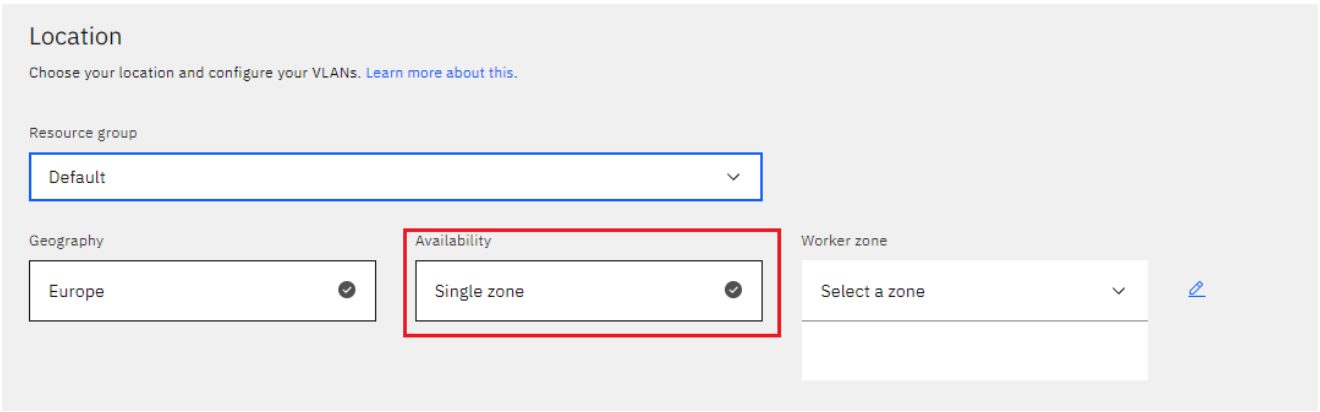
infra-select

8. Now choose your location settings, for more information please visit Locations
9. Choose **Geography** (continent)



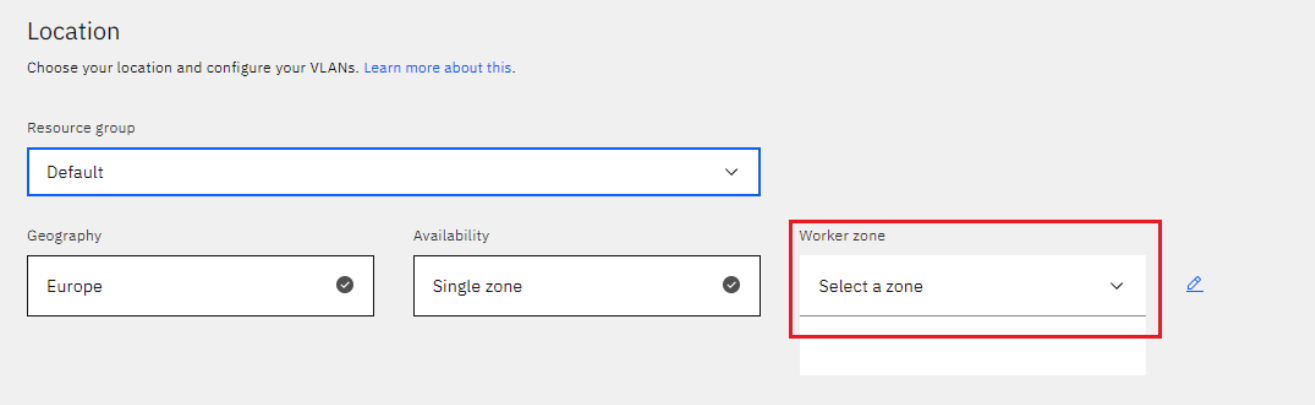
location-geo

10. Select either **Single-Zone** or **Multi-Zone** storage. Single Zone stores your data in one data center, while Multi-Zone distributes your data across multiple zones, offering greater safety in the event of unexpected zone failures.



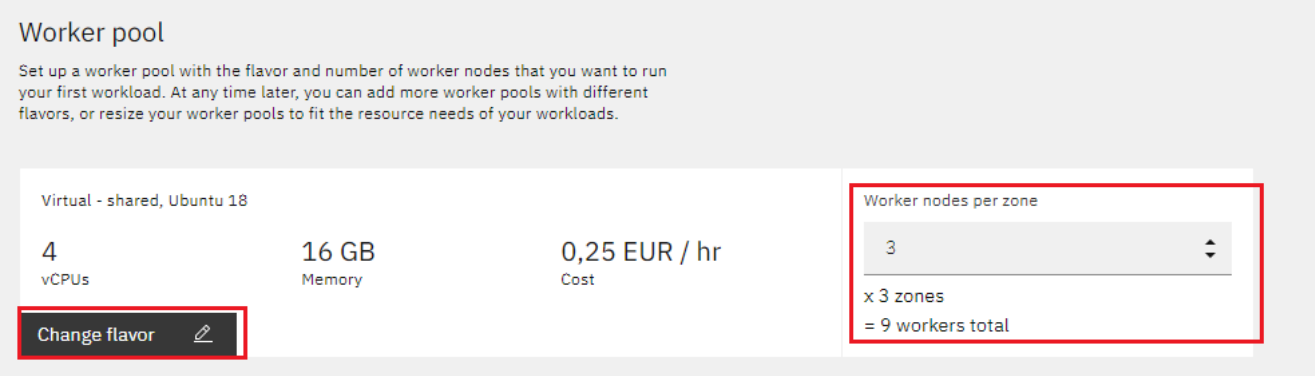
location-avail

11. Choose a **Worker Zone** if using Single zones or Metro if Multizone



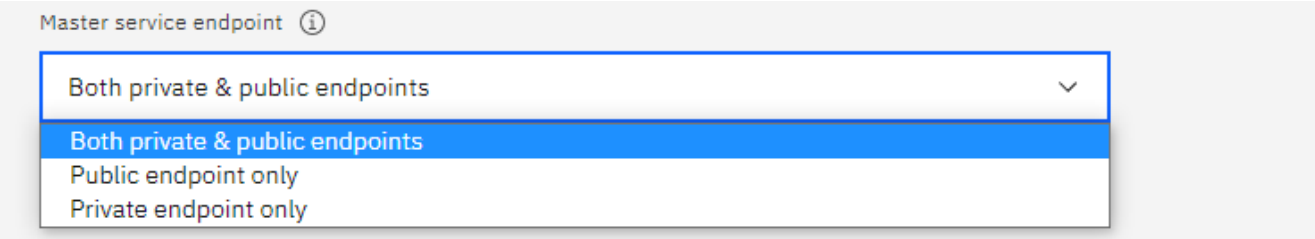
location-worker

12. Set up your account with VRF or enable VLAN spanning to use Multizone.
13. If no Virtual LAN is available at your selected location, a new VLAN will be created for you.
14. Choose a Worker node setup or use the preselected one.
15. Set the **Worker node per zone**.



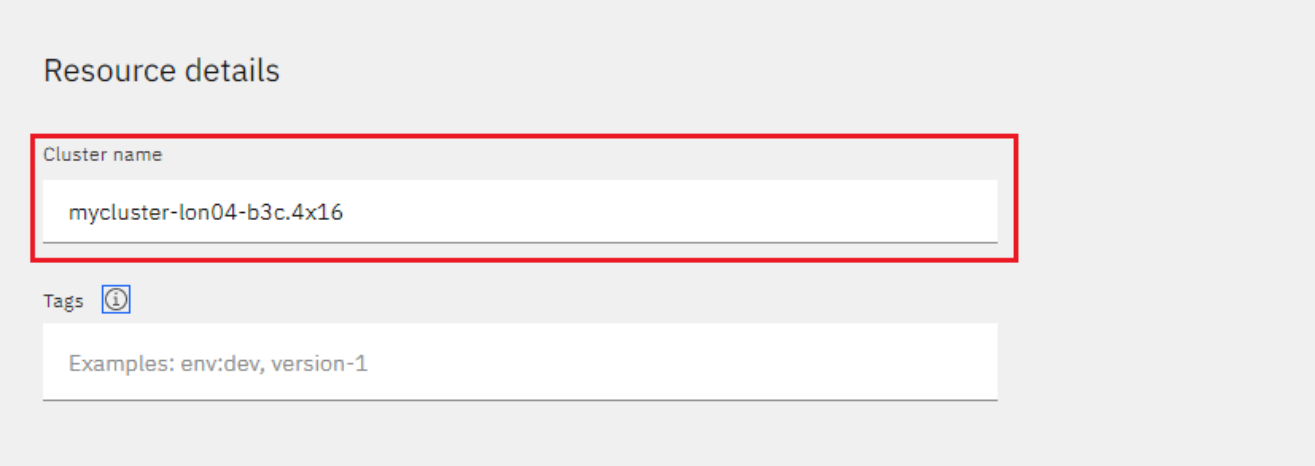
worker-pool

16. Choose Master Service Endpoint, In VRF-enabled accounts, you can choose **Private endpoint only** to make your master accessible on the private network or via VPN tunnel. Choose **Public endpoint only** to make your master publicly accessible. When you have a VRF-enabled account, your cluster is set up by default to use **Both private and public endpoints**. For more information visit endpoints.



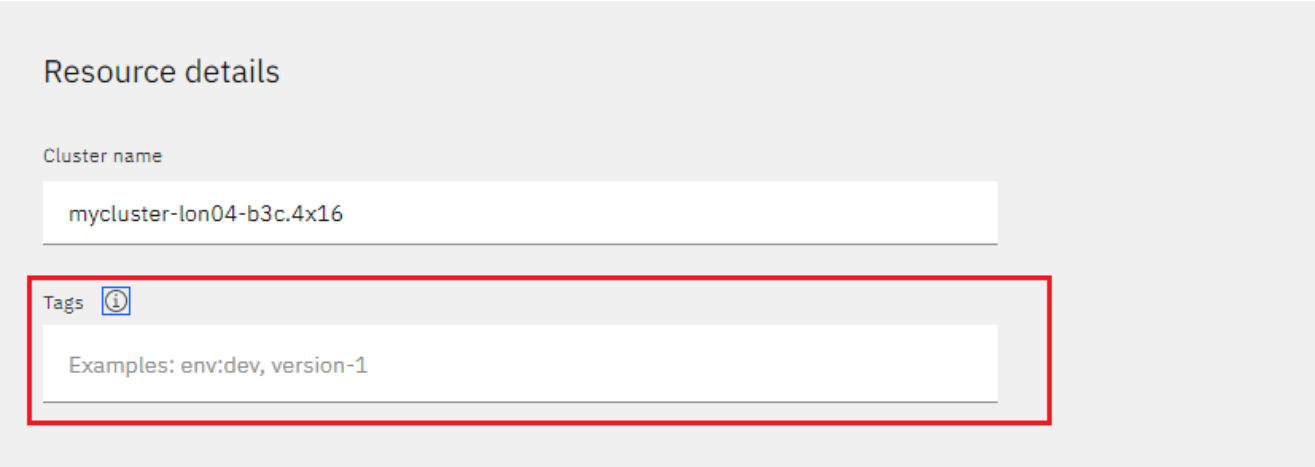
endpoints

17. Give **Cluster name**



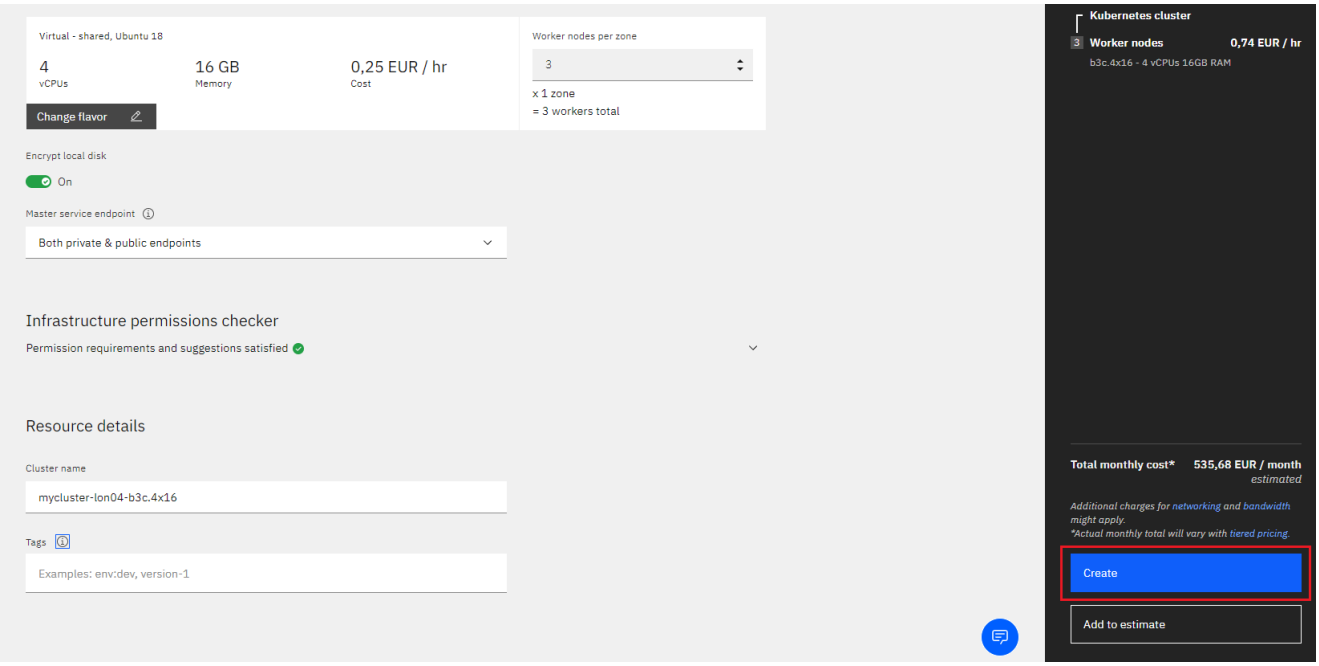
name-new

18. Give desired **Tags** to your cluster, for more information visit tags



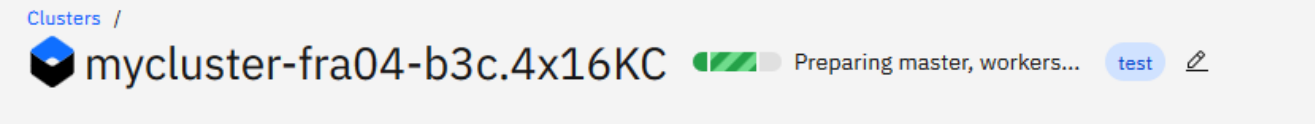
tasg-new

19. Click **Create**



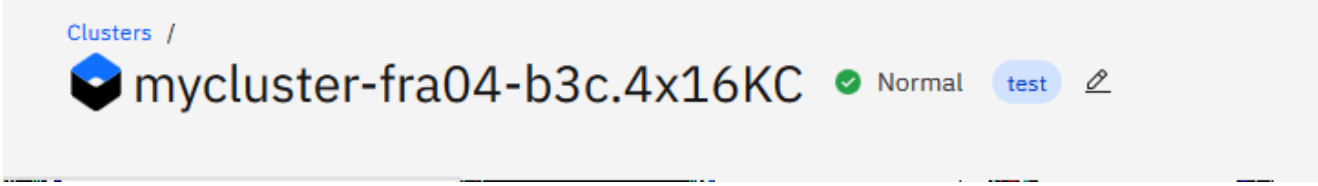
create-new

20. Wait for you cluster to be provisioned



cluster-prepare

21. Your cluster is ready for usage

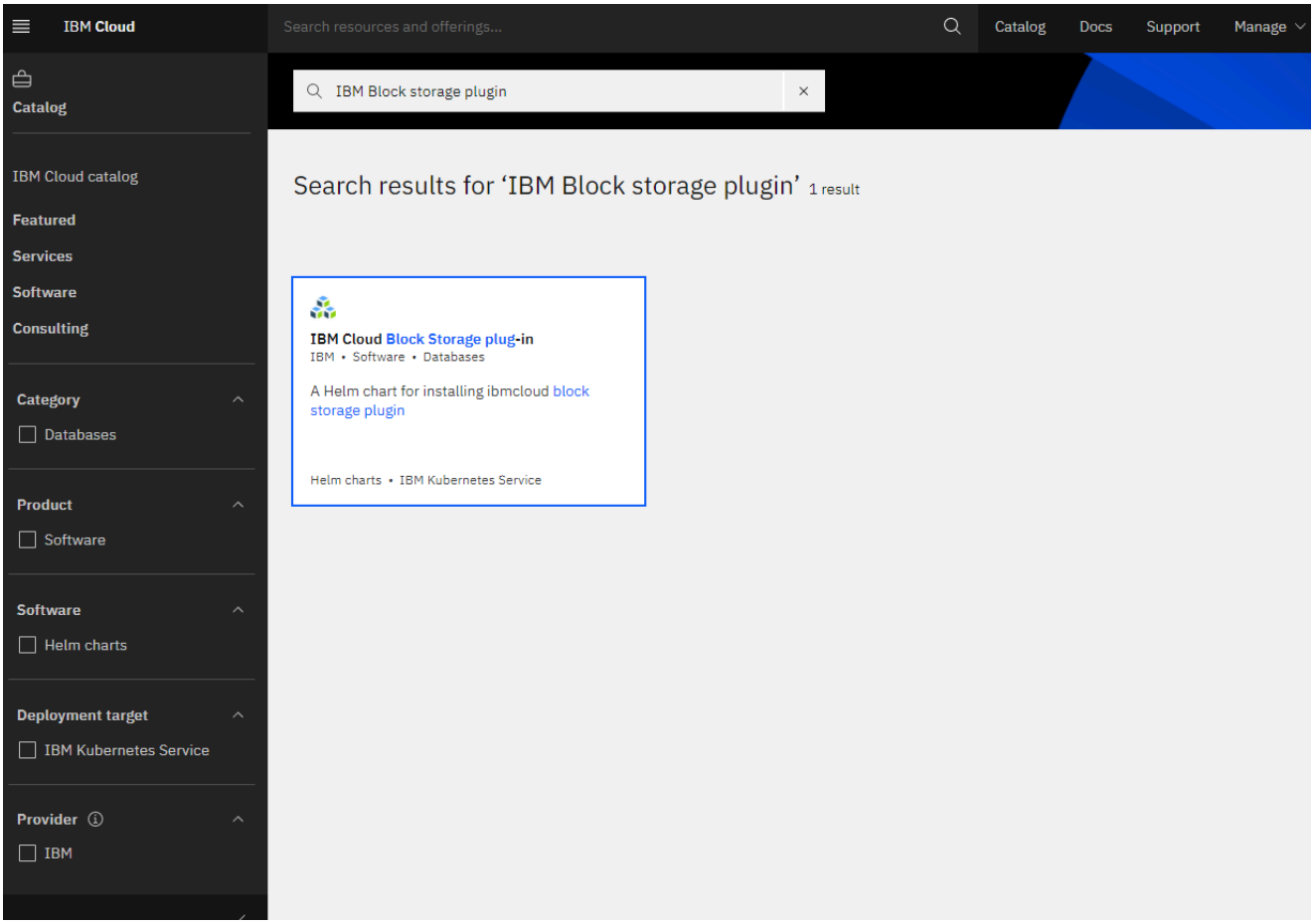


cluster-done

Step 2 deploy IBM Cloud Block Storage plug-in

The Block Storage plug-in is a persistent, high-performance iSCSI storage that you can add to your apps by using Kubernetes Persistent Volumes (PVs).

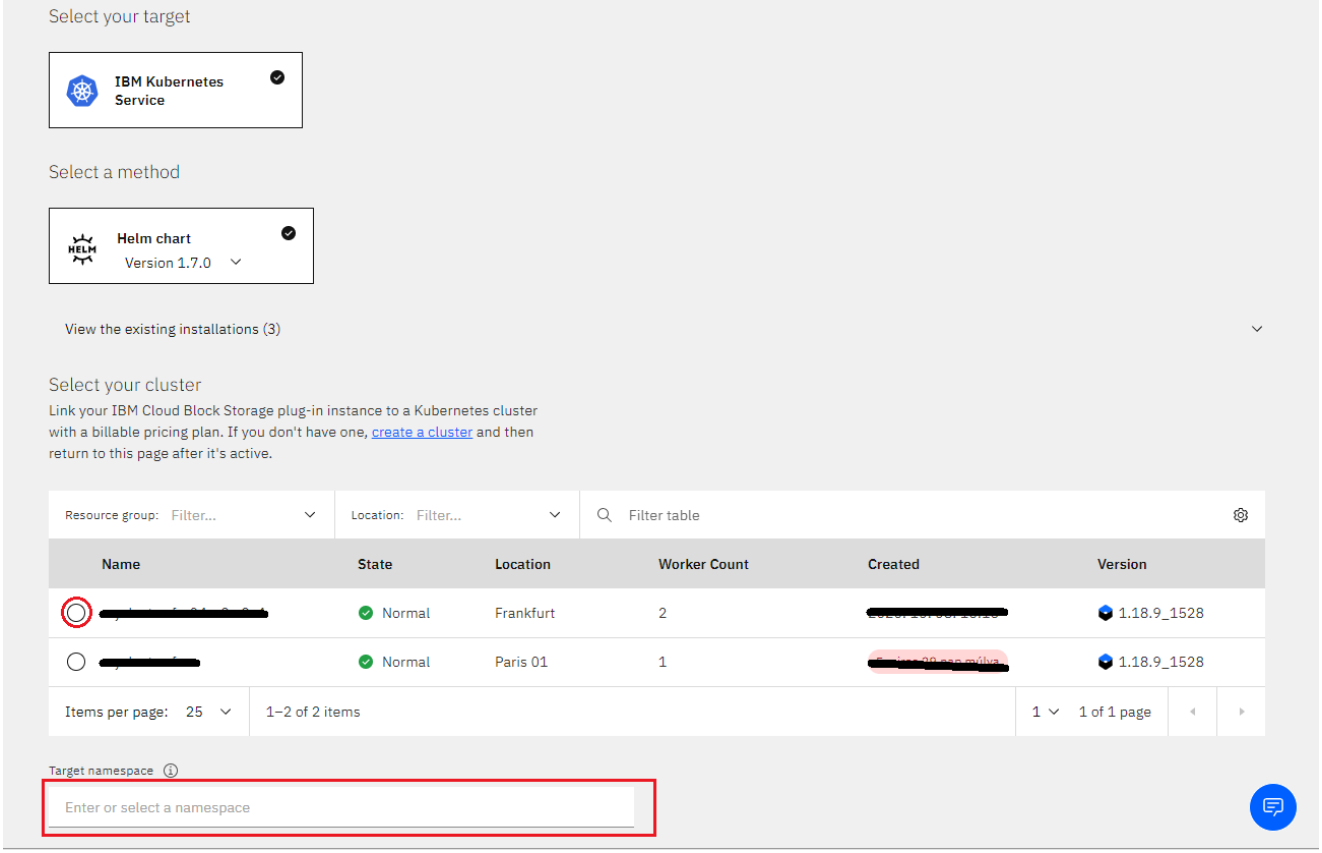
1. Click the **Catalog** button on the top
2. Select **Software** from the catalog
3. Search for *IBM Cloud Block Storage plug-in* and click on it



block-search

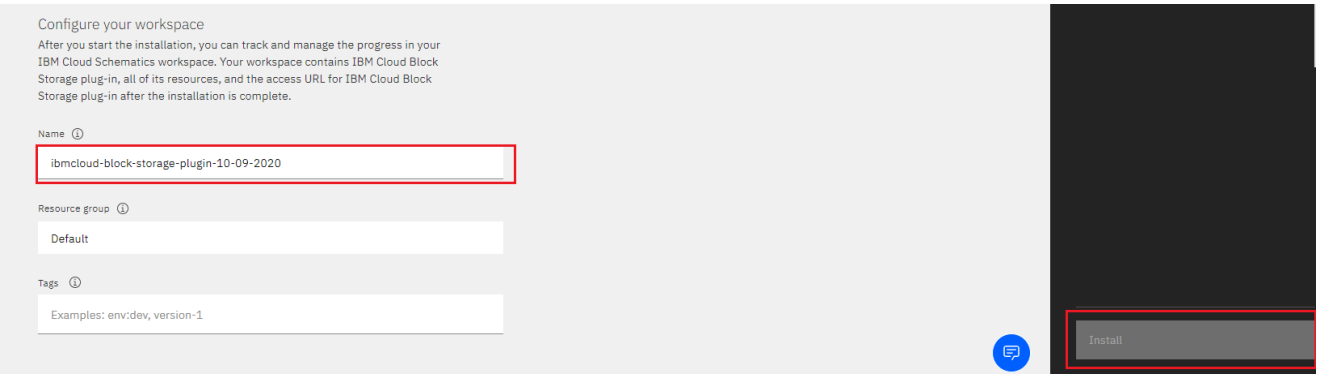
4. On the application page, select the cluster you wish to use

5. Click on **Enter or select namespace** and choose the default Namespace or use a custom one (if you get error please wait 30 minutes for the cluster to finalize)



block-cluster

6. Give a **Name** to this workspace
7. Click **Install** and wait for the deployment

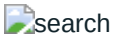


block-storage-create

Step 3 deploy MariaDB

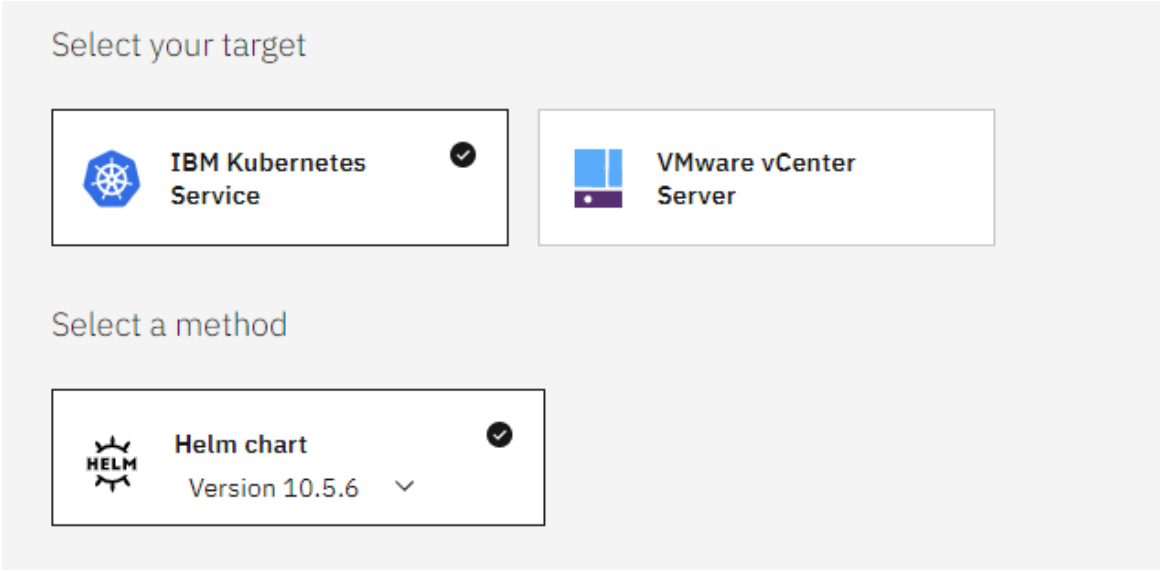
We will deploy MariaDB on our cluster

1. Click the **Catalog** button on the top
2. Select **Software** from the catalog
3. Search for *MariaDB* and click on it



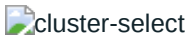
search

4. Please select **IBM Kubernetes Service**



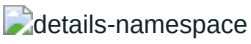
target-select

5. On the application page, select the cluster you wish to use



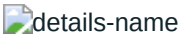
cluster-select

6. Click on **Enter or select namespace** and choose the default Namespace or use a custom one



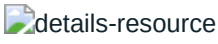
details-namespace

7. Give a unique name to workspace, which you can easily recognize



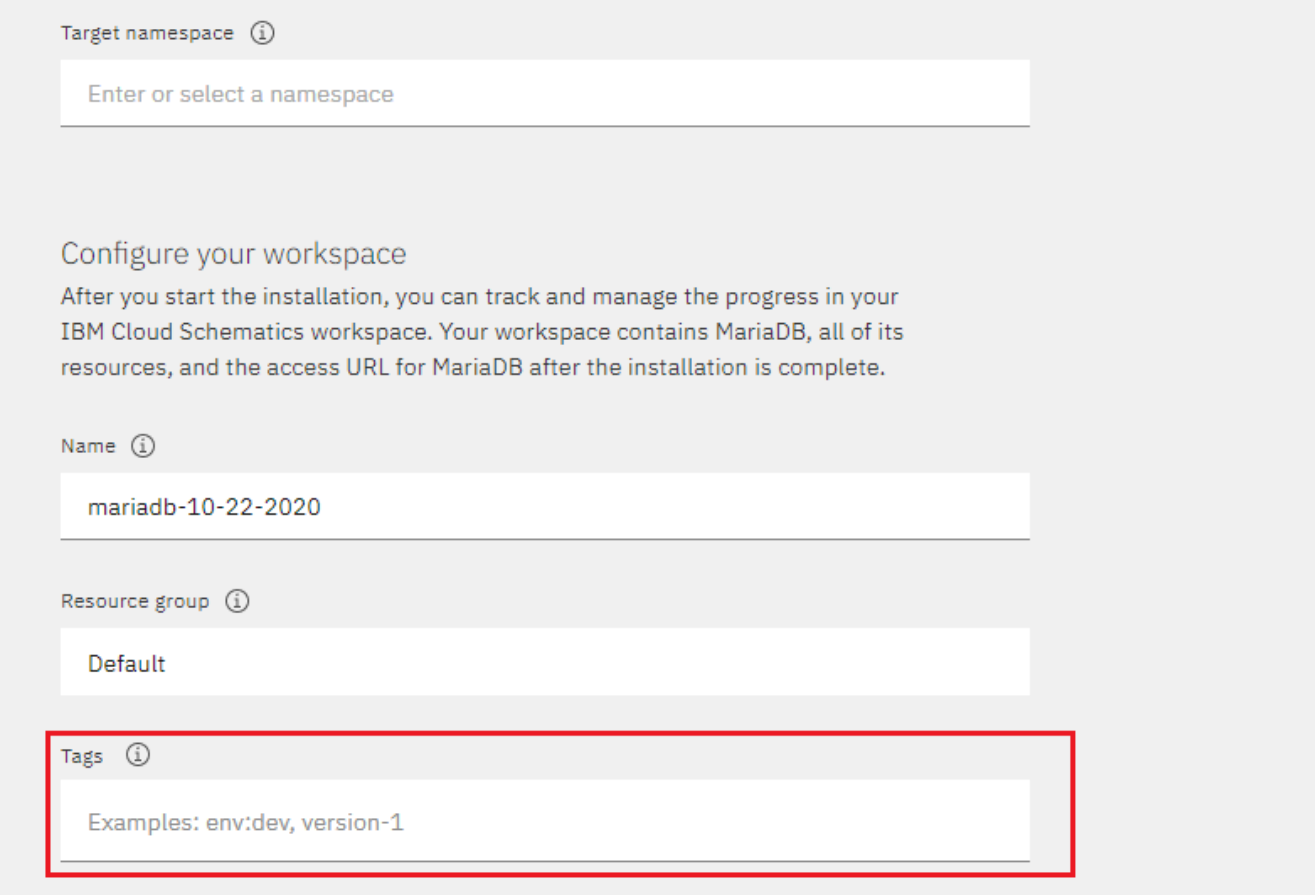
details-name

8. Select which **Resource Group** you want to use, it's for access control and billing purposes. For more information please visit resource groups



details-resource

9. Give **Tags** to your MariaDB, for more information visit tags

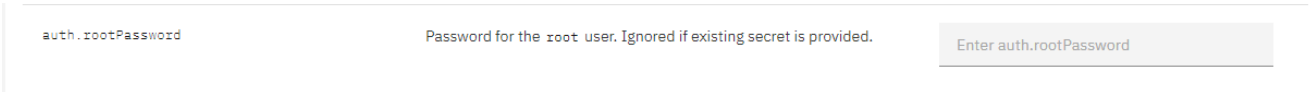


details-tags

10. Click on **Parameters with default values**, You can set deployment values or use the default ones

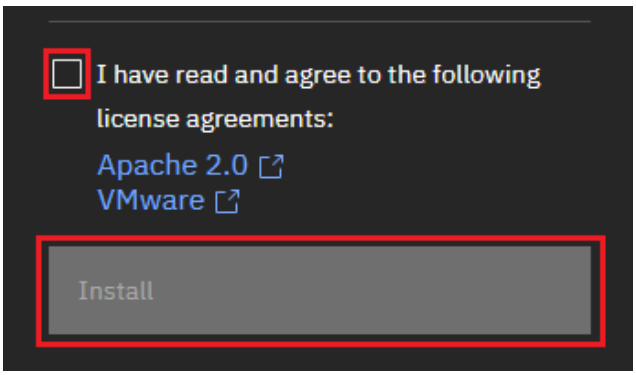
parameters

11. Please set the MariaDB root password in the parameters



root-password

12. After finishing everything, tick the box next to the agreements and click **Install**



aggreement-create

13. The MariaDB workspace will start installing, wait a couple of minutes

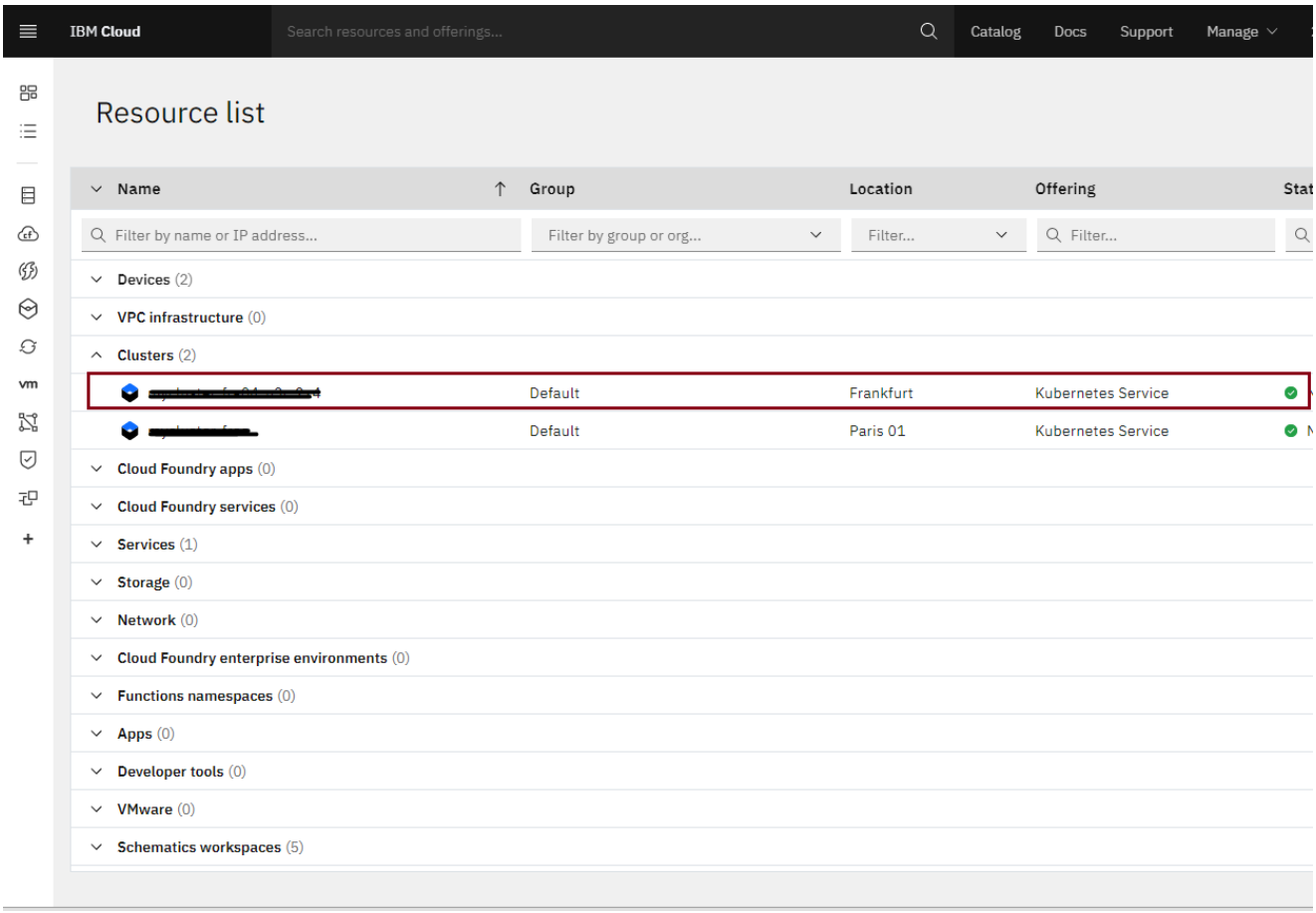
in-progress

14. Your MariaDB workspace has been successfully deployed

done

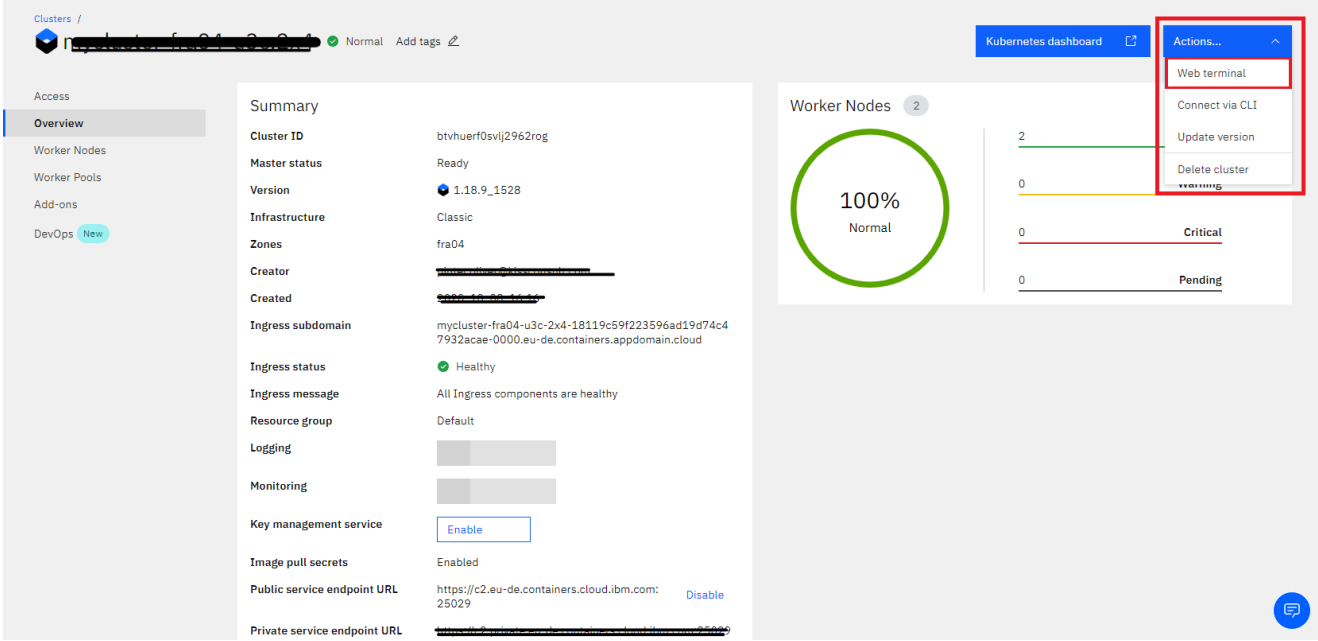
Verify MariaDB installation

1. Go to Resources in your browser
2. Expand **Clusters.** Click on your cluster



resource-select

3. On the **Overview** page, click **Actions**, then select **Web Terminal** from the dropdown menu.



cluster-main

4. Click Install and wait for sometime
5. Click on Actions
6. Click Web terminal. A terminal window will open up
7. On the terminal window, please change NAMESPACE to the namespace you choose at the deployment setup:

```
$ kubectl get ns
```

get-ns

```
$ kubectl get pod -n NAMESPACE -o wide
```

```
pinter.oliver@k8s-terminal ~ (⎈ mycluster-fra04-u3c.2x4/btvhuerf0svlj2962rog:default)$ kubectl get pods -n mariadb -o wide
NAME                 READY   STATUS    RESTARTS   AGE   IP              NODE            NOMINATED NODE   READINESS GATES
mariadb2f-c27f-41-0  1/1     Running   0          21m   172.30.40.152   10.240.129.19   <none>           <none>
pinter.oliver@k8s-terminal ~ (⎈ mycluster-fra04-u3c.2x4/btvhuerf0svlj2962rog:default)$
```

get-pods

```
$ kubectl get service -n NAMESPACE
```

```
pinter.oliver@k8s-terminal ~ (⎈ mycluster-fra04-u3c.2x4/btvhuerf0svlj2962rog:default)$ kubectl get service -n mariadb
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
mariadb2f-c27f-41  ClusterIP   172.21.247.125   <none>        3306/TCP   22m
pinter.oliver@k8s-terminal ~ (⎈ mycluster-fra04-u3c.2x4/btvhuerf0svlj2962rog:default)$
```

get-service

8. Enter your pod with bash, please replace PODNAME with your mariadb pod's name

```
$ kubectl exec --stdin --tty PODNAME -n NAMESPACE -- /bin/bash
```

bash

9. After you are in your pod please enter Mariadb and enter your root password after the prompt

```
$ mysql -u root -p
```

welcome

You have successfully deployed MariaDB IBM Cloud!

This page is licensed: CC BY-SA / Gnu FDL

Compiling MariaDB From Source

See Compiling MariaDB From Source: The Master Guide.

Complete step-by-step guide for compiling and building MariaDB from source on various flavors of Linux and on macOS.

See Building MariaDB on Windows.

Guide to building MariaDB on Windows using Visual Studio, CMake, and Git, including creating ZIP and MSI packages.

See Cross-compiling MariaDB.

Instructions for cross-compiling MariaDB for different architectures, including using Buildroot and CMake toolchain files.

See Compiling MariaDB with Extra Modules/Options.

Compile MariaDB Server with extra modules and options. This section details how to customize your build from source, enabling specific features or optimizations for your deployment. width: default title: visible: true description: visible: true tableOfContents: visible: true outline: visible: true pagination: visible: false metadata: visible: true tags: visible: true

This page is licensed: CC BY-SA / Gnu FDL

Compiling MariaDB From Source: The Master Guide

> **Note:**
>
> This guide covers compiling MariaDB on Unix-like systems, including Linux and macOS. If you are building on Windows, please refer to the dedicated Building MariaDB on Windows guide.

This guide provides the universal workflow for building MariaDB Server from source code. While specific dependencies may vary by operating system, the core build process remains consistent across all modern platforms.

Major Steps

> **Step 1**
>
> **Prepare Your Environment**
>
> Before you begin, your system must have the necessary compilers, build tools, and library headers.
>
> > For details, see Install Build Dependencies.

Step 2

Obtain the Source Code

Decide whether you need the latest development branch or a specific stable release.

- Option A: Git clone (best for developers)

```
git clone --branch 11.8 https://github.com/MariaDB/server.git
cd server
```

- Option B: Source tarball (best for stability)

  Download the `.tar.gz` from the official MariaDB downloads and extract it:

```
tar -xf mariadb-11.8.x.tar.gz
cd mariadb-11.8.x
```

> For details on both options, see Obtaining the Source Code.

Step 3

Configure the Build (CMake)

MariaDB uses out-of-source builds to keep the source tree clean. This is where you define installation paths and features.

1. Create a build directory: `mkdir build && cd build`
2. Run CMake:

```
cmake .. -DCMAKE_BUILD_TYPE=RelWithDebInfo
```

   - *Common flags like `-DCMAKE_INSTALL_PREFIX` or debug options go here.*

> For common CMake configuration flags, see Common CMake Configuration Flags.
>
> For a configuration alternative to configuring CMake, see The Quick Way: Using BUILD Scripts.

Step 4

Compile

Once configured, use the CMake build tool to compile the binaries. Using the `--parallel` flag speeds this up by using multiple CPU cores.

LINUX

```
cmake --build . --parallel $(nproc)
```

MACOS

```
cmake --build . --parallel $(sysctl -n hw.logicalcpu)
```

UNIVERSAL

For CMake 3.13+, run this:

```
cmake --build . --parallel
```

Without a number, modern CMake automatically picks an appropriate number of jobs.

> If you encounter build errors, see Troubleshooting Common Build Errors.

Step 5

Install and Initialize

After a successful build, you must prepare the data directory and system tables before the server can start.

1. Install: `sudo cmake --install .` (or run directly from the build directory for testing).
2. Create Data Directory: Ensure the `mysql` user exists and has permissions.
3. Initialize System Tables.
    1. If running from the build directory (use this if you did *not* run the `cmake` install command – in that case, you must run the script from within your build folder):

        ```
        ./scripts/mariadb-install-db --user=mysql --datadir=[path]
        ```

    2. If you installed to the system (`sudo cmake --install ...`, in which case the `PATH` to `mariadb-install-db` is set):

        ```
        mariadb-install-db --user=mysql --datadir=[path]
        ```

> See this section for conventional locations of the data directory (`--datadir`).

Step 6

Start and Verify

Launch the server and check that it is responsive.

```
./bin/mariadbd-safe --user=mysql &
./bin/mariadb -u root -p
```

> **Step 7**
>
> **Test the Build**
>
> Before putting your fresh build into production, verify it against the official test suite. MariaDB includes MTR (MariaDB Test Run) for this purpose.
>
> 1. Run Unit Tests: Quickly check the core logic.
>
>    ```
>    cmake --build . --target test
>    ```
>
> 2. Run Integration Tests (MTR): This launches actual server instances and runs thousands of SQL tests.
>
>    ```
>    cd mysql-test
>    ./mysql-test-run.pl --parallel=$(nproc)
>    ```
>
>   - Pro-Tip: If a specific test fails, you can run it individually: `./mysql-test-run.pl alias`.
>   - Note: You do not need to install MariaDB to the system to run these tests; they run entirely within the build directory.

> **Step 8**
>
> **After Building**
>
> After successfully building MariaDB from source, additional tasks can be found under Post-Build.

Summary of the Build Workflow

Step	Action	Primary Tool
1	Install Build Tools	`apt` / `dnf` / `brew`
2	Download Source	`git` / `wget`
3	Configure Features	`cmake`
4	Compile Binaries	`cmake --build`
5	Initialize Database	`mariadb-install-db`

Preparing Your Environment: Install Build Dependencies

> This section covers the details of step 1 (Prepare Your Environment).

To compile MariaDB, you need a set of core build tools and development headers for various libraries.

Setting up the MariaDB Source Repository

> **Note:**
>
> Setting up the MariaDB source repository is optional but recommended. If you want to build the MariaDB version that comes with your operating system, you can skip this subsection, though.

Adding the MariaDB Repository Tool into the workflow ensures that you are getting the dependencies specifically tailored for the version of MariaDB you want to build (for instance, 11.8), rather than whatever outdated version happens to be in your operating system's default upstream repository.

Using the MariaDB Repository Tool

While your operating system's default repositories contain many build tools, they may lack the specific library versions required for the latest MariaDB releases. Using the MariaDB Repository Tool ensures your environment is perfectly synced with the version you intend to build.

1. Configure the Repository

Use the MariaDB Repository Configuration Tool to generate the setup commands for your specific operating system and desired MariaDB version.

Example for Ubuntu 24.04 and MariaDB 11.8 (**don't copy this blindly** – the example uses the 11.8 release and a specific mirror; adjust these strings based on the output of the Repository Configuration Tool):

```
sudo apt-get install software-properties-common dirmngr
sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xF1656F24C74CD1D8
sudo add-apt-repository --update --yes --enable-source \
'deb [arch=amd64] https://mirrors.xtom.com/mariadb/repo/11.8/ubuntu noble main'
```

2. Install Tailored Dependencies

Once the repository is active and the `source` lines are enabled, you can use the package manager to pull exactly what that specific MariaDB version requires:

> **UBUNTU / DEBIAN**
>
> ```
> sudo apt-get build-dep mariadb-11.8  # Replace with your target version
> ```
>
> **RHEL / CENTOS / FEDORA**
>
> ```
> sudo dnf builddep mariadb-server
> ```

Advantages of Using the Repository Tool

1. Version Accuracy: If MariaDB 11.x requires a newer `libssl` or `zstd` than what your operating system provides by default, the MariaDB repo often provides the correct headers.
2. Completeness: It automatically handles the "Source Repositories" issue because the `add-apt-repository` command includes the `--enable-source` flag by default.
3. Automation: It reduces the "Manual Package Table" from a primary task to a fallback for users who cannot or will not add external repositories.

The "Shortcut" Command

Most Linux distributions allow you to install all necessary dependencies for the "official" build with a single command.

This requires that "source repositories" are enabled in your package manager. Enable them like this:

> **UBUNTU**
>
> Run this command:
>
> ```
> sudo add-apt-repository --update --yes --enable-source
> ```
>
> **DEBIAN**
>
> Edit `/etc/apt/sources.list` and uncomment lines starting with `deb-src`.
>
> **RHEL / ALMA**
>
> Run `dnf config-manager --set-enabled baseos-source appstream-source`.
>
> **FEDORA**
>
> Source repos are usually enabled by default, but check `dnf repolist` to verify.

The shortcut command itself varies by operating system:

> **UBUNTU / DEBIAN**
>
> `sudo apt build-dep mariadb-server`
>
> **RHEL / CENTOS / FEDORA**
>
> `sudo dnf builddep mariadb` (or `yum-builddep mariadb-server`)
>
> **SLES / OPENSUSE**
>
> `sudo zypper source-install -d mariadb`
>
> **MACOS**
>
> Homebrew does not have a direct `build-dep` command. You can install the standard MariaDB build stack using:
>
> ```
> brew install cmake git bison flex openssl zlib boost
> ```

Manual Package Table

If you are building a specific version or don't want to use the shortcut, use the list below to install the mandatory packages.

> **UBUNTU / DEBIAN**
>
> Use `apt` to install these tools:
>
> - Build tools: `build-essential git cmake`
> - Parser tools: `bison flex`
> - Terminal/UI: `libncurses-dev`
> - Security/SSL: `libssl-dev` or `gnutls-dev`
> - Compression: `zlib1g-dev`
> - Miscellaneous headers: `libaio-dev libboost-all-dev`
>
> **RHEL / CENTOS / FEDORA**
>
> Use `dnf` (or `yum`) to install these tools:
>
> - Build tools: `gcc-c++ git cmake`
> - Parser tools: `bison flex`
> - Terminal/UI: `ncurses-devel`
> - Security/SSL: `openssl-devel`
> - Compression: `zlib-devel`
> - Miscellaneous headers: `libaio-devel boost-devel`
>
> **SLES / OPENSUSE**
>
> Use `zypper` to install these tools:
>
> - Build tools: `gcc-c++ git cmake`
> - Parser tools: `bison flex`
> - Terminal/UI: `ncurses-devel`
> - Security/SSL: `libopenssl-devel`
> - Compression: `libzlib-devel`
> - Miscellaneous headers: `libaio-devel boost-devel`

> **MACOS**
>
> Use `brew` to install these tools:
>
> - Build tools: `cmake git`
> - Parser tools: `bison flex`
> - Terminal/UI: `ncurses`
> - Security/SSL: `openssl`
> - Compression: `zlib`
> - Miscellaneous headers: `boost`
>
> The system-provided version of `bison` on macOS is often too old. After running `brew install bison`, you may need to add it to your `PATH`:
>
> ```
> export PATH="/opt/homebrew/opt/bison/bin:$PATH"
> ```

Optional Dependencies (Plugins & Features)

If you plan to enable specific storage engines or features, install these additional packages:

Feature	Dependency	Package Name (Generic)
S3 Storage Engine	`libcurl`	`libcurl-devel` / `libcurl4-openssl-dev`
GSSAPI (Kerberos)	`krb5`	`krb5-devel` / `libkrb5-dev`
RocksDB / MyRocks	`snappy, lz4, zstd`	
Systemd Support	`systemd`	`systemd-devel` / `libsystemd-dev`
Authentication	`pam`	`pam-devel` / `libpam0g-dev`

Obtaining the Source Code

> This section covers the details of step 2 (Obtain the Source Code).

Depending on your goal, you can either clone the repository using Git or download a stable source tarball.

Option A: Git Clone (Recommended for Developers)

Using Git allows you to easily switch between versions and contribute patches. MariaDB uses submodules for certain components; you must initialize these for the build to succeed.

1. Clone the specific branch:

   Choose a branch that matches the major version you need (for instance, `11.8`).

   ```
   git clone --branch 11.8 https://github.com/MariaDB/server.git
   cd server
   ```

2. Initialize Submodules:

   This command downloads the source code for external storage engines and connectors.

   ```
   git submodule update --init --recursive
   ```

Option B: Source Tarball (Best for Stability)

If you do not need version control, downloading a pre-packaged source tarball is simpler. Submodules are already included in the tarball, so no extra steps are required.

1. Download: Visit the MariaDB Downloads page and select the “Source” tab.
2. Extract:

```
tar -xf mariadb-11.8.x.tar.gz
cd mariadb-11.8.x
```

Key Takeaway: Git vs. Tarball

Feature	Git Clone	Source Tarball
Updates	Easy (`git pull`)	Manual (download new version)
Submodules	Requires manual initialization	Pre-included
History	Full commit history available	No history included
Size	Larger (includes `.git` data)	Smaller

Common CMake Configuration Flags

> This section covers the details of step 3 (Configure the Build).

When running `cmake ..`, you can pass these options (using the `-D` prefix) to customize how MariaDB is built.

Category	Flag	Description
Install Path	`-DCMAKE_INSTALL_PREFIX=/opt/mariadb`	Defines where the server is installed (default: `/usr/local`).
Build Type	`-DCMAKE_BUILD_TYPE=RelWithDebInfo`	Options: `Release` , `Debug` , `RelWithDebInfo` (recommended).
Features	`-DWITH_EMBEDDED_SERVER=ON`	Builds the `libmariadbd` embedded library.
Storage Engines	`-DPLUGIN_ROCKSDB=NO`	Explicitly disables a specific storage engine (use `YES` to force enable).
Security	`-DWITH_SSL=system`	Use the system’s OpenSSL (default). Options: `system` , `openssl` , `gnutls` .
Debugging	`-DWITH_ASAN=ON`	Enables AddressSanitizer to find memory leaks.
Standardization	`-DBUILD_CONFIG=mysql_release`	Configures the build to match the official MariaDB release binaries.

Pro-Tip: Managing Plugins

Most plugins are auto-detected based on the dependencies you installed in step 1. To see a full list of available options and their current status, run:

```
cmake .. -LH
```

The Quick Way: Using BUILD Scripts

> This section covers alternative details of step 3 (Configure the Build).

If you don’t want to manually toggle CMake flags, MariaDB includes a `BUILD/` directory containing pre-configured scripts for common development environments. These are especially useful for developers who need a specific setup (like a Debug build with Valgrind support) quickly.

How to Use a BUILD Script

These scripts must be run from the root of the source directory (unlike the manual “out-of-source” CMake method).

```
# Example: Build an optimized 64-bit version with debugging symbols
./BUILD/compile-pentium64-debug
```

Common Script Variants

Most scripts follow a naming convention: `compile-[architecture]-[type]`.

Script	Best For…
`compile-pentium64-debug`	Standard 64-bit development with debug symbols.
`compile-pentium64-max`	Enables almost all features and plugins (highest compatibility).
`compile-pentium64-valgrind-max`	Optimized for memory testing with Valgrind.
`compile-amd64-debian-build`	Mimics the configuration used for official Debian/Ubuntu packages.

Adding Extra Flags

You can still pass custom CMake flags to these scripts using environment variables or arguments, though most users find the defaults sufficient for testing.

```
./BUILD/compile-pentium64-debug --extra-configs="-DCMAKE_INSTALL_PREFIX=/tmp/mariadb"
```

Troubleshooting Common Build Errors

> This section covers details of step 4 (Compile).

Compiling from source often hits roadblocks. Here are the most common issues and how to fix them.

Error Message	Likely Cause	Solution
`CMAKE_CXX_COMPILER not found`	Missing C++ compiler.	Install `g++` (Linux) or `clang` (macOS).
`Bison version is too old`	macOS default `bison` is outdated.	`brew install bison` and update your `PATH`.
`Could not find GSSAPI`	Missing Kerberos headers.	Install `libkrb5-dev` or `krb5-devel`.
`Could not find Curses`	Missing terminal headers.	Install `libncurses-dev` or `ncurses-devel`.
`No space left on device`	`/tmp` or build dir is full.	MariaDB requires ~5GB to 10GB of disk space for a full build.

The “Clean Slate” Command

If you changed your CMake flags and things are getting weird, the best fix is to wipe the cache and start over.

```
# Inside your build directory
rm -rf *
cmake .. [your flags]
```

> **Note:**
>
> Do not delete the source directory, only the contents of the `build` folder.

Notes

Conventional Data Directory Locations

This table lists typical locations of `--datadir`.

Platform	Conventional Data Directory
Linux (General)	`/var/lib/mysql`
macOS (Homebrew)	`/opt/homebrew/var/mysql`
macOS (Intel/Legacy)	`/usr/local/var/mysql`
FreeBSD	`/var/db/mysql`
Custom/Dev Build	`~/mariadb-data` (or any folder owned by your user)

Post-Build

> This section covers optional additional things to do after building MariaDB successfully (step 8 (After Building)).

Packaging

If you need to deploy MariaDB to multiple servers, you don't have to compile it on every machine. Instead, use MariaDB's built-in packaging support to create a distributable package.

MariaDB uses CPack (part of the CMake suite) to generate these packages.

1. Configure for Packaging: To ensure the package follows standard filesystem hierarchies, use the `mysql_release` configuration.

   ```
   cmake .. -DBUILD_CONFIG=mysql_release
   ```

2. Generate the Package: After the build is complete, run `cpack` from your build directory.

   ```
   # To create the default package for your OS (DEB on Ubuntu/Debian, RPM on RHEL)
   cpack -G DEB   # For Debian-based
   cpack -G RPM   # For RedHat-based
   cpack -G TGZ   # For a generic binary tarball
   ```

3. Result: You will find a file like `mariadb-11.x.x-linux-x86_64.deb` in your build directory, which can be installed via `dpkg -i` or `rpm -ivh`.

See Also

A compiling guide aimed at developers & contributors is available here. It was last updated in 2025. The reading time is 8-10 minutes. Quick summary:

> This guide provides a comprehensive technical walkthrough for developers looking to contribute to MariaDB Server. It covers hardware prerequisites (2GB+ RAM, 4 cores), source code acquisition via GitHub forks, and the build process using CMake and Ninja. Key sections explain how to configure debug builds, execute parallel testing using the mysql-test-run (MTR) framework, and initialize the database from the build directory for local development. It emphasizes using specific branches for features (main) versus bug fixes (earliest maintained branch).

This page is licensed: CC BY-SA / Gnu FDL

Building MariaDB on Windows

Build Requirements

To build MariaDB you need the following:

- Visual C++: We currently support Visual Studio 2019 and 2022. Generally we try to support the two most recent VS versions, but build ourselves using the last one. Community editions will work fine; we only use them in our builds.
  While installing Visual Studio, make sure to add “Desktop Development with C++”.
  Also, make sure to use recent enough Windows SDK - latest Windows 11 SDK is recommended.
- CMake: We recommend the latest release. Older releases might not support your version of Visual Studio. Visual Studio 2019 requires cmake 3.14 at least.
- Git: Required to build newer versions from the source tree.
  - NOTE: run

```
git config --global core.autocrlf input
```

after the installation, otherwise some mtr tests will fail

In the “Adjusting your PATH” dialog, choose “Use Git from Windows command prompt”, otherwise wrong (mingw64) git and perl will be in your PATH

- Bison from GnuWin32:
  Bison creates parts of the SQL parser. Choose “Complete package except sources” when downloading.
  - NOTE: Do not install this into your default path with spaces (e.g. under `C:\Program Files\GnuWin32`); the build will break due to this bison bug.
    Instead, install into `C:\GnuWin32`.
  - Add `C:\GnuWin32\bin` to your system `PATH` after installation.
- Strawberry perl: Used to run the test suite.ActiveState Perl is another Win32 Perl distribution and should work as well (but it is not as well tested). NOTE: `Cygwin` or `mingw` Perl versions will not work for testing. Use Windows native Perl, please.
- Optional: If you intend to build the MSI packages, install Windows Installer XML . If you build MSI with 10.4, also modify your Visual Studio installation, add “Redistributable MSMs” (see MDEV-22555)
- Gnu Diff, needed if you run mysql-test-run.pl tests.

Verify that bison.exe, or git.exe, cmake.exe and perl.exe can be found in the PATH environment variable with “`where bison`”, “`where git`”, “`where perl`” etc. from the command line prompt.

Building Windows Binaries

The above instructions assume MariaDB 10.2 or higher.

Branch the MariaDB repository, or unpack the source archive. On the command prompt, switch to your source directory, then execute:

```
mkdir bld
cd bld
cmake ..
cmake --build . --config RelWithDebInfo
```

The above example builds a release configured for 64 bit systems in a subdirectory named `bld`. “`cmake ..`” is the configuration step,
“`cmake --build . --config Relwithdebinfo`” is the build step.

Build Variations

Debug Builds

Building Debug version is done with:

```
cmake --build . --config Debug
```

32bit and 64 bit Builds

Build 64 bit binary

Visual Studio 2019-2022 `cmake` generator will use host architecture by default, that is, with the steps above, `cmake` will build x64 binaries on x64 machine.

Build 32 bit binary

pass `-A Win32` parameter for `cMake`, like this

```
cmake .. -A Win32
```

Historical note:
With Visual Studio 2017 and earlier, one had to pass the name of 32bit generator ,e.g `cmake .. -G "Visual Studio 15 2017`

For a complete list of available generators, call `cmake` without any parameters.

IDE Builds

Instead of calling "`cmake --build`" as above, open solution file `MariaDB.sln` (in older versions, prior to 11.0, `MySQL.sln`). When Visual Studio starts, choose Build/Compile.

Building the ZIP Package

```
cmake --build . --config relwithdebinfo --target package
```

This is how it is "done by the book", standard `cmake` target.

MariaDB however uses non-standard target `win_package` for the packaging for its releases, it generates 2 ZIPs, a slim one with executables, and another one with `debuginfo` (.PDB files). The `debuginfo` is important to be able to debug released binaries, and to analyze crashes.

```
cmake --build . --config relwithdebinfo --target win_package
```

Building the MSI Package

```
cmake --build . --config relwithdebinfo
cmake --build . --config relwithdebinfo --target MSI
```

Including HeidiSQL in the MSI Installer

Starting with MariaDB 5.2.7, it is possible to build an installer which includes 3rd party products, as described in MWL#200. Currently only HeidiSQL support is implemented; it is also included in the official builds. Use the `CMake` parameter `-DWITH_THIRD_PARTY=HeidiSQL` to include it in the installer.

Code Signing

MariaDB builds optionally support authenticode code signing with an optional parameter `SIGNCODE`. Use `cmake -DSIGNCODE=1` during the configuration step to sign the binaries in the `ZIP` and `MSI` packages.

Important: for `SIGNCODE=1` to work, the user that runs the build needs to install a valid authenticode digital certificate into their certificate store, otherwise the packaging step will fail.

Building Packages for MariaDB Releases

The full script to create the release in an out-of-source build with Visual Studio with signed binaries might look like:

```
mkdir bld
cd bld
cmake .. -DSIGNCODE=1 -DWITH_THIRD_PARTY=HeidiSQL
cmake --build . --config relwithdebinfo --target win_package
cmake --build . --config relwithdebinfo  --target MSI
```

This command sequence will produce a ZIP package (e.g mariadb-5.2.6-win32.zip) and MSI package (e.g mariadb-5.2.6-win32.msi) in the `bld` directory.

Running Tests

- Important: Do not use Cygwin bash, MinGW bash, Git bash, WSL bash, or any other bash when running the test suite. You will then very likely use the wrong version of Perl too (a “Unix-flavoured” one on Windows), and spend a lot of time trying to figure out why this version of Perl does not work for the test suite. Use native perl, in cmd.exe , or powershell instead,
- Switch `mysql-test` subdirectory of the build directory

```
cd C:\server\bld\mysql-test
```

- Run the test suite

```
perl mysql-test-run.pl --suite=main --parallel=auto
```

Running a Test Under Debugger

Assuming VS is installed on the machine

```
perl mysql-test-run.pl  <test_name> --vsjitdebugger
```

If `vsjitdebugger` does not start, you can edit `AeDebug` registry key as mentioned in

debug-using-the-just-in-time-debugger?view=vs-2019#jit_errors

Alternatively:

```
perl mysql-test-run.pl  <test_name> --devenv
```

(devenv.exe needs to be in PATH)

or, if you prefer `WinDBG`

```
perl mysql-test-run.pl  <test_name> --windbg
```

This page is licensed: CC BY-SA / Gnu FDL

Cross-compiling MariaDB

Buildroot

Buildroot is a way to cross compile MariaDB and other packages into a root filesystem. In the menuconfig you need to enable “Enable C++ Support” first under “Toolchain”. After C++ is enabled MariaDB is an option under “Target Packages” ->“Libraries” -> “Databases” -> “mysql support” -> “mysql variant” -> “mariadb”. Also enable the “mariadb server” below the “mysql support” option.

Details

To cross-compile with `cmake` you will need a `toolchain` file. See, for example, here. Besides `cmake` specific variables it should have, at least

```
SET(STACK_DIRECTION -1)
  SET(HAVE_IB_GCC_ATOMIC_BUILTINS 1)
```

with appropriate values for your target architecture. Normally these MariaDB configuration settings are detected by running a small test program, and it cannot be done when cross-compiling.

Note that during the build few helper tools are compiled and then immediately used to generate more source files for this very build. When cross-compiling these tools naturally should be built for the host architecture, not for the target architecture. Do it like this (assuming you’re in the mariadb source tree):

```
$ mkdir host
$ cd host
$ cmake ..
$ make import_executables
$ cd ..
```

Now the helpers are built and you can cross-compile:

```
$ mkdir target
$ cd target
$ cmake .. -DCMAKE_TOOLCHAIN_FILE=/path/to/toolchain/file.cmake -
DIMPORT_EXECUTABLES=../host/import_executables.cmake
$ make
```

Here you invoke `cmake` , specifying the path to your toolchain file and the path to the `import_executables.cmake` that you have just built on the previous step. Of course, you can also specify any other cmake parameters that could be necessary for this build, for example, enable or disable specific storage engines.

See also msg02911.html

This page is licensed: CC BY-SA / Gnu FDL

Compiling MariaDB with Extra Modules/Options

See Specifying Which Plugins to Build.

Explains how to use CMake options like `PLUGIN_xxx` to control which plugins are built statically, dynamically, or not at all during compilation.

See Using MariaDB with TCMalloc or jemalloc.

Instructions on building and configuring MariaDB to use alternative memory allocators like TCMalloc or jemalloc for improved performance and profiling.

This page is licensed: CC BY-SA / Gnu FDL

Specifying Which Plugins to Build

By default all plugins are enabled and built as dynamic `.so` (or `.dll`) modules. If a plugin does not support dynamic builds, it is not built at all.

Use `PLUGIN_xxx` cmake variables. They can be set on the command line with `-DPLUGIN_xxx=value` or in the cmake gui. Supported values are

Value	Effect
`NO`	the plugin will be not compiled at all
`STATIC`	the plugin will be compiled statically, if supported. Otherwise it will be not compiled.
`DYNAMIC`	the plugin will be compiled dynamically, if supported. Otherwise it will be not compiled. This is the default behavior.
`AUTO`	the plugin will be compiled statically, if supported. Otherwise it will be compiled dynamically.
`YES`	same as AUTO, but if plugin prerequisites (for example, specific libraries) are missing, it will not be skipped, it will abort cmake with an error.

Note that unlike autotools, cmake tries to configure and build incrementally. You can modify one configuration option and cmake will only rebuild the part of the tree affected by it. For example, when you do `cmake -DWITH_EMBEDDED_SERVER=1` in the already-built tree, it will make libmysqld to be built, but no other configuration options will be changed or reset to their default values.

In particular this means that if you have run, for example `cmake -DPLUGIN_OQGRAPH=NO` and later you want to restore the default behavior (with OQGraph being built) in the same build tree, you would need to run `cmake -DPLUGIN_OQGRAPH=DYNAMIC`

Alternatively, you might simply delete the `CMakeCache.txt` file — this is the file where cmake stores current build configuration — and rebuild everything from scratch.

This page is licensed: CC BY-SA / Gnu FDL

Using MariaDB with TCMalloc or jemalloc

> **Note:**
>
> Read the Profiling Memory Usage page for more information on how to debug high memory consumption.

Using tcmalloc or jemalloc

TCMalloc is a malloc replacement library optimized for multi-threaded usage. It features a built-in heap debugger and profiler. Another malloc replacement that can speed up MariaDB is jemalloc.

The procedures to use one of these libraries with MariaDB are the same. Many other malloc replacement libraries (as well as heap debuggers and profilers) can be used with MariaDB in a similar fashion.

Prerequisites

`jemalloc` is a powerhouse for reducing memory fragmentation and improving concurrency, but it isn't a "one-size-fits-all" drop-in for every environment. Its availability and stability depend heavily on how the operating system handles memory pages and how MariaDB was compiled.

Operating System Availability

- Linux: Jemalloc is most at home here. Most major distributions (Ubuntu, Debian, CentOS/RHEL, Arch) include it in their official repositories (`libjemalloc2` or `jemalloc`). It is the standard recommendation for high-traffic MariaDB setups on Linux.

- Windows: Jemalloc does not have the same “native” presence on Windows. While the `jemalloc` source code can technically be compiled for Windows using MSVC or MinGW, MariaDB for Windows is almost exclusively used with the default Windows system allocator. Most DBAs do not use `jemalloc` on Windows production servers.
- FreeBSD: This is the “birthplace” of `jemalloc` – it replaced the system malloc in FreeBSD back in 2005. If you are running MariaDB on FreeBSD, you are likely using `jemalloc` by default at the OS level.

Hardware and Architecture Hurdles

One of the most common “hidden” failures occurs on non-x86 architectures:

- Raspberry Pi (ARM): On newer devices like the Raspberry Pi 5, `jemalloc` often fails to start with the error: `<jemalloc>: Unsupported system page size`. This happens because `jemalloc` is frequently compiled assuming a 4KB page size, while many ARM kernels now use 16KB or 64KB pages for better performance.
- Cloud Instances: Small instances (like AWS `t2.micro`) can sometimes behave unpredictably with `jemalloc` if they lack sufficient swap space or if the memory overhead of the `jemalloc` metadata conflicts with the tiny RAM footprint.

Compatibility Matrix Summary

Platform	Jemalloc Support	Typical Usage
Linux (x86_64)	Excellent	Highly recommended; installed via `apt` or `yum`.
FreeBSD	Native	It is the system default.
Windows	Poor/Partial	Rarely used; usually requires custom builds.
ARM (RPi/M1/M2)	Conditional	Depends on the kernel page size (4KB vs 16KB).
Solaris/Illumos	Experimental	Possible but lacks the robust testing seen on Linux.

Check if it’s Ready to Use

Before trying to force it via `LD_PRELOAD`, check if the library even exists on your system paths:

```
# On Debian/Ubuntu
find /usr/lib -name "libjemalloc.so.*"

# On RHEL/CentOS
find /usr/lib64 -name "libjemalloc.so.*"
```

If these commands return nothing, you need to install the package first (`sudo apt install libjemalloc2` or `sudo dnf install jemalloc`).

Why Use it

The main reason to switch isn’t just speed, but fragmentation. Standard `malloc` can leave small gaps of unused memory that the OS can’t reclaim, leading to “memory bloat” where MariaDB appears to use 10GB of RAM even if the data only needs 6GB. Jemalloc is much better at packing these allocations tightly.

Checking the malloc Implementation in Use

To check which `malloc` implementation is used, run this query:

```
SHOW GLOBAL VARIABLES LIKE 'version_malloc_library';
+-------------------------+--------+
| Variable_name           | Value  |
+-------------------------+--------+
| version_malloc_library | system |
+-------------------------+--------+
```

A value of `system` indicates the system default, which is normally `malloc`. If another library is used, the name and version of the library is shown.

Building MariaDB With an Alternative to malloc

To build MariaDB with `TCMalloc`, you need to use the following command:

```
cmake -DCMAKE_EXE_LINKER_FLAGS='-ltcmalloc'  -DWITH_SAFEMALLOC=OFF
```

To use `jemalloc`, specify `-ljemalloc` instead of `-ltcmalloc`.

Starting mariadbd-safe With an Alternative to malloc

Start a standard MariaDB server with `TCmalloc` like this:

```
/usr/sbin/mariadbd-safe --malloc-lib=tcmalloc
```

To configure mariadbd-safe to use `tcmalloc` or `jemalloc`, add this to your configuration file:

```
[mariadbd-safe]
malloc-lib=tcmalloc
```

Starting mariadbd With an Alternative to malloc

Locate the library file that needs to be used:

```
# jemalloc
find /usr/lib -name "libjemalloc.so.*"
# tcmalloc
find /usr/lib -name "libtcmalloc.so.*"
```

Pass that library file to `mariadbd` using the `LD_PRELOAD` variable:

```
LD_PRELOAD=/path/to/library mariadbd
```

Configuring systemd

If you use `systemd` to run MariaDB, locate the library as explained above, then locate the service configuration file:

```
systemctl status mariadb | grep Loaded
```

Edit the `mariadb.service` file by adding a line to the `[Service]` group:

```
[Service]
Environment=LD_PRELOAD=<path-to-library>
```

For example, add this:

```
[Service]
Environment=LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libjemalloc.so.2
```

Reload the configuration for the news setting to take effect, and restart MariaDB:

```
systemctl daemon-reload
systemctl restart mariadb
```

Dockerfile

If you run MariaDB on Docker and use an image from a Dockerfile that is publicly available, most probably you have an entry point that is a bash script, which starts `mariadbd` directly. Edit that bash script, or set the `LD_PRELOAD` variable from the Dockerfile:

```
ENV LD_PRELOAD=<path-to-library>
```

To find the library file, run one of these commands while the container is running:

```
# jemalloc
docker exec -ti <container-name> find /usr/lib -name "libjemalloc.so.*"
# tcmalloc
docker exec -ti <container-name> find /usr/lib -name "libtcmalloc.so.*"
```

Example:

```
docker run -P -d --name mariadb --env LD_PRELOAD="/usr/lib/x86_64-linux-gnu/libjemalloc.so.2" --env
MARIADB_ROOT_PASSWORD=Password123! mariadb:latest
```

Vagrantfile

Usually Vagrant is used to start a complete system in a virtual machine. In that case, you can use one of the methods above, for example you can modify your Vagrantfile to copy a modified version of the `mariadb.service` file to the guest system to configure systemd.

If you use Vagrant with the Docker provider, you can follow the instructions above to modify the Dockerfile.

Making jemalloc Persistent

To make MariaDB use `jemalloc` persistently, set it up like this. These instructions work on Linux using `systemd` and running on an x86 architecture.

Steps

Step 1

Identify the jemalloc library path.

Identify the `jemalloc` library path, to find the exact location of the `jemalloc` shared library on your system:

```
find /usr -name "libjemalloc.so.*"
```

Take a note of the output path. Typically, it is something like `/usr/lib64/libjemalloc.so.2`.

Step 2

Create or edit the MariaDB Systemd override file.

This is the crucial step for persistence. Use `systemctl edit` to manage the override file:

```
sudo systemctl edit mariadb
```

If an override file already exists for MariaDB, it opens in the editor. If not, the editor creates a new, empty file, typically at `/etc/systemd/system/mariadb.service.d/override.conf`. Add the following content to the file:

```
[Service]
Environment="LD_PRELOAD=/usr/lib64/libjemalloc.so.2"
```

Replace `/usr/lib64/libjemalloc.so.2` with the actual path you noted in step 1. Save the file and exit the editor.

Step 3

Restart services and MariaDB.

```
sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl restart mariadb
```

Step 4

Reboot the computer, and verify that jemalloc is used.

You can verify this on various levels.

MariaDB:

```
mariadb -e "SHOW GLOBAL VARIABLES LIKE 'version_malloc_library';"
+------------------------+----------------------------------------------------------+
| Variable_name          | Value                                                    |
+------------------------+----------------------------------------------------------+
| version_malloc_library | jemalloc 5.2.1-0-gea6b3e973b477b8061e0076bb257dbd7f3faa756 |
+------------------------+----------------------------------------------------------+
```

Operating system:

```
sudo cat /proc/$MARIADB_PID/environ | tr '\0' '\n' | grep LD_PRELOAD
LD_PRELOAD=/usr/lib64/libjemalloc.so.2
```

Operating system processes:

```
sudo cat /proc/$MARIADB_PID/maps | grep jemalloc
7f6981c00000-7f6981c06000 r--p 00000000 fd:00 50981369 /usr/lib64/libjemalloc.so.2
7f6981c06000-7f6981c76000 r-xp 00006000 fd:00 50981369 /usr/lib64/libjemalloc.so.2
7f6981c76000-7f6981c81000 r--p 00076000 fd:00 50981369 /usr/lib64/libjemalloc.so.2
7f6981c81000-7f6981c87000 r--p 00080000 fd:00 50981369 /usr/lib64/libjemalloc.so.2
7f6981c87000-7f6981c88000 rw-p 00086000 fd:00 50981369 /usr/lib64/libjemalloc.so.2
```

Notes

`systemctl edit` creates a separate configuration file (`/etc/systemd/system/mariadb.service.d/override.conf`). `systemd` always prioritizes settings in override files over the main service file (`/usr/lib/systemd/system/mariadb.service`).

When upgrading MariaDB, the package manager only replaces or updates the main service file. Your custom override file in `/etc/systemd/system/mariadb.service.d/` remains untouched, thus preserving your `jemalloc`

configuration.

`systemd` reads all its service configurations, including override files, on system start. This means the `LD_PRELOAD` environment variable is set for MariaDB automatically each time the service starts, making the change persistent across reboots.

Finding memory leaks with jemalloc

`jemalloc` provides a report of memory leaks at program exit:

```
MALLOC_CONF=prof_leak:true,lg_prof_sample:0,prof_final:true \
LD_PRELOAD=${JEMALLOC_PATH}/lib/libjemalloc.so.2  path-to-mariadbd
```

This produces output like this:

```
<jemalloc>: Leak summary: 267184 bytes, 473 objects, 20 contexts
<jemalloc>: Run jeprof on "jeprof.19678.0.f.heap" for leak detail
```

You can learn more about the memory leaks with `jeprof`, which is shipped with jemalloc:

```
jeprof --show_bytes path-to-mariadbd jeprof.19678.0.f.heap
```

You can also generate a PDF call graph of the leak:

```
jeprof --show_bytes --pdf path-to-mariadbd  jeprof.19678.0.f.heap > /tmp/mariadbd.pdf
```

See Also

- Profiling memory usage
- Jemalloc leak checking

This page is licensed: CC BY-SA / Gnu FDL

Configuring MariaDB

See Configuring MariaDB with Option Files.

Complete guide to MariaDB option files. Complete my.cnf reference with configuration groups, parameter syntax, and file hierarchy for production use.

See MariaDB Environment Variables.

Complete MariaDB environment variables: MYSQL_HOME, LD_PRELOAD, my.cnf search path precedence, and interaction with option files/command-line options.

See Advanced Configuration.

Dive into advanced configurations for MariaDB Server performance. This section covers in-depth tuning parameters, optimization strategies, and best practices to maximize speed and efficiency.

This page is licensed: CC BY-SA / Gnu FDL

Configuring MariaDB with Option Files

You can configure MariaDB to run the way you want by configuring the server with MariaDB's option files. The default MariaDB option file is called `my.cnf` on Unix-like operating systems and `my.ini` on Windows. Depending on how you've installed MariaDB, the default option file may be in a number of places, or it may not exist at all.

MariaDB Enterprise Server users should also refer to the Enterprise Server-Specific section for additional configuration files like `mariadb-enterprise.cnf`.

Global Options Related to Option Files

The following options relate to how MariaDB handles option files. These options can be used with most of MariaDB's command-line tools, not just mariadbd. They must be given as the first argument on the command-line:

Option	Description
–print-defaults	Read options from option files, print all option values, and then exit the program.
–no-defaults	Don't read options from any option file.
–defaults-file =path	Only read options from the given option file.
–defaults-extra-file =path	Read this extra option file after all other option files are read.
–defaults-group-suffix =suffix	In addition to the default option groups, also read option groups with the given suffix.

Default Option File Locations

MariaDB reads option files from many different directories by default. See the sections below to find out which directories are checked for which system.

For an exact list of option files read on your system by a specific program, you can execute a command like this (`$program` stands for any MariaDB program, including the Server):

```
$program --help --verbose
```

For example:

```
$ mariadbd --help --verbose
mariadbd  Ver 10.11.2-MariaDB for linux-systemd on x86_64 (MariaDB Server)
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Starts the MariaDB database server.

Usage: mariadbd [OPTIONS]

Default options are read from the following files in the given order:
/etc/my.cnf ~/.my.cnf
The following groups are read: mysqld server mysqld-10.11 mariadb mariadb-10.11 mariadbd mariadbd-10.11
client-server galera
....
```

The option files are each scanned once, in the order given by `--help --verbose`. The effect of the configuration options is the same as if they would have been given as command-line options in the order they are found.

Default Option File Locations on Linux and Unix

On Linux and Unix operating systems, the default option file is called `my.cnf`. MariaDB looks for the MariaDB option file in the locations and orders listed below.

The locations are dependent on whether the `DEFAULT_SYSCONFDIR` cmake option was defined when MariaDB was built. This option is usually defined as `/etc` when building RPM packages, but it is usually not defined when building DEB packages or binary tarballs.

- If the `DEFAULT_SYSCONFDIR` `cmake` option is **undefined**, MariaDB looks for the MariaDB option file in the following locations, and in the following order:

Location	Scope
`/etc/my.cnf`	Global
`/etc/mysql/my.cnf`	Global
`$MARIADB_HOME/my.cnf`	Server (from MariaDB 10.6)

Location	Scope
`$MYSQL_HOME/my.cnf`	Server (before MariaDB 10.6)
`defaults-extra-file`	File specified with –defaults-extra-file, if any
`~/.my.cnf`	User

- If the `DEFAULT_SYSCONFDIR` `cmake` option is **defined**, MariaDB looks for the MariaDB option file in the following locations in the following order:

Location	Scope
`DEFAULT_SYSCONFDIR/my.cnf`	Global
`$MARIADB_HOME/my.cnf`	Server (from MariaDB 10.6)
`$MYSQL_HOME/my.cnf`	Server (before MariaDB 10.6)
`defaults-extra-file`	File specified with –defaults-extra-file, if any
`~/.my.cnf`	User

- `MARIADB_HOME` or `MYSQL_HOME` is the environment variable containing the path to the directory holding the server-specific `my.cnf` file. If `MYSQL_HOME` is not set, and the server is started with mysqld_safe, `MYSQL_HOME` is set as follows:
  - If there is a `my.cnf` file in the MariaDB data directory, but not in the MariaDB base directory, `MYSQL_HOME` is set to the MariaDB data directory.
  - Else, `MYSQL_HOME` is set to the MariaDB base directory.

> **Warning:**
>
> Note that if `MARIADB_HOME` is set, `MYSQL_HOME` is not used, even if set.

Default Option File Locations on Windows

On Windows, the option file can be called either `my.ini` or `my.cnf`. Depending on the distribution, MariaDB Enterprise Server for Windows may include additional configuration files. These files comply with the load order rules described in the following section.

MariaDB looks for option files in the following locations, and in the following order:

Location	Scope
System Windows Directory\my.ini	Global
System Windows Directory\my.cnf	Global
Windows Directory\my.ini	Global
Windows Directory\my.cnf	Global
`c:\my.ini`	Global
`c:\my.cnf`	Global
`installdir\my.ini`	Server
`installdir\my.cnf`	Server
`installdir\data\my.ini`	Server
`installdir\data\my.cnf`	Server
`%mariadb_home%\my.ini`	Server (from MariaDB 10.6)
`%mariadb_home%\my.cnf`	Server (from MariaDB 10.6)

Location	Scope
`%mysql_home%\my.ini`	Server
`%mysql_home%\my.cnf`	Server
`defaults-extra-file`	File specified with –defaults-extra-file, if any

- The `System Windows Directory` is the directory returned by the GetSystemWindowsDirectory function. The value is usually `C:\Windows`. To find its specific value on your system, open cmd.exe and execute:

```
echo %WINDIR%
```

- The `Windows Directory` is the directory returned by the `GetWindowsDirectory` function. The value may be a private `Windows Directory` for the application, or it may be the same as the `System Windows Directory` returned by the `GetSystemWindowsDirectory` function.
- `INSTALLDIR` is the parent directory of the directory where `mysqld.exe` is located. For example, if `mysqld.exe` is in `C:\Program Files\` , `INSTALLDIR` is `C:\Program Files\`.
- `MARIADB_HOME` or `MYSQL_HOME` is the environment variable containing the path to the directory holding the server-specific `my.cnf` file.

> **Warning:**
>
> Note that if `MARIADB_HOME` is set, `MYSQL_HOME` is not used, even if set.

MariaDB Enterprise Server Option File Locations

MariaDB Enterprise Server includes additional configuration files that are installed with the Enterprise Server packages. Specifically, the file `mariadb-enterprise.cnf` is typically placed in a configuration include directory and enables Enterprise-specific features by default (for example, loading the Enterprise Audit plugins with `plugin-load-add`, `server_audit` and other enterprise plugins):

```
/etc/my.cnf.d/mariadb-enterprise.cnf
```

The configuration values defined in these files can be validated at runtime. See Verifying Option File Sources in Enterprise Server.

Default Option File Hierarchy

MariaDB looks in all of the above locations, in order, even if it has already found an option file, and it's possible for more than one option file to exist. For example, you could have an option file in `/etc/my.cnf` with global settings for all servers, and another option file in `~/.my.cnf` (that is, your user account's home directory) which specifies additional settings (or override previously specified settings) that are specific only to that user.

Option files are usually optional. However, if the –defaults-file option is set, but the file does not exist, MariaDB raises an error. If the `--defaults-file` option is set, MariaDB *only* reads the option file referred to by this option.

If an option or system variable is not explicitly set, then it will be set to its default value. See Server System Variables for a full list of all server system variables and their default values.

> **Warning:**
>
> If an option is set multiple times, the later setting will override the earlier setting.

If –log-basename is set, there are various other log file naming options that, if set, should be placed **later** in the configuration file hierarchy. Later settings override earlier settings, so `log-basename` overrides any earlier log file name settings. See –log-basename for details.

Custom Option File Locations

MariaDB can be configured to read options from custom options files with the following command-line arguments. These command-line arguments can be used with most of MariaDB's command-line tools, not just mariadbd. They must be given as the first argument on the command line:

Option	Description
–defaults-file =path	Only read options from the given option file.
–defaults-extra-file =path	Read this extra option file after all other option files are read.

Option File Syntax

The syntax of MariaDB option files is as follows:

- Lines starting with `#` are comments.
- Empty lines are ignored.
- Option groups use the syntax `[group-name]`. See the Option Groups section below for more information on available option groups.
- The same option group can appear multiple times.
- The `!include` directive can be used to include other option files. See the Including Option Files section below for more information on this syntax.
- The `!includedir` directive can be used to include all `.cnf` files (and potentially `.ini` files) in a given directory. The option files within the directory are read in alphabetical order. See the Including Option File Directories section below for more information on this syntax.
- The `?includedir` (introduced in MariaDB 11.4.10 and 11.8.6) directive works like `!includedir` but skips unreadable files, instead of failing with an error.
- Dashes (`-`) and underscores (`_`) in options are interchangeable.
- Double quotes can be used to quote values
- `\n`, `\r`, `\t`, `\b`, `\s`, `\"`, `\'`, and `\\` are recognized as character escapes for new line, carriage return, tab, backspace, space, double quote, single quote, and backslash respectively.
- Certain option prefixes are supported. See the Option Prefixes section below for information about available option prefixes.
- See the Options section below for information about available options.

Option Groups

A MariaDB program can read options from one or many option groups. For an exact list of option groups read on your system by a specific program, you can execute (`$program` is a placeholder for any MariaDB program):

```
$program --help --verbose
```

For example:

```
$ mariadbd --help --verbose
mariadbd  Ver 10.11.2-MariaDB for linux-systemd on x86_64 (MariaDB Server)
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Starts the MariaDB database server.

Usage: mariadbd [OPTIONS]

Default options are read from the following files in the given order:
/etc/my.cnf ~/.my.cnf
The following groups are read: mysqld server mysqld-10.11 mariadb mariadb-10.11 mariadbd mariadbd-10.11
client-server galera
....
```

Server Option Groups

MariaDB programs reads server options from the following server option groups:

Group	Description
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[server]	Options read by MariaDB Server.
[mysqld]	Options read by mysqld, including both MariaDB Server and MySQL Server.
[mysqld-X.Y]	Options read by a specific version of mysqld, including both MariaDB Server and MySQL Server. For example, `[mariadb-11.4]`.
[mariadb]	Options read by MariaDB Server.
[mariadb-X.Y]	Options read by a specific version of MariaDB Server. For example, `[mariadb-11.4]`.
[mariadbd]	Options read by MariaDB Server.
[mariadbd-X.Y]	Options read by a specific version of MariaDB Server. For example, `[mariadbd-11.4]`.
[galera]	Options read by MariaDB Server, but only if it is compiled with Galera Cluster support. All builds on Linux are compiled with Galera Cluster support. When using one of these builds, options from this option group are read even if the Galera Cluster functionality is not enabled.

X.Y in the examples above refer to the base (major.minor) version of the server. For example, MariaDB 11.4 would read from `[mariadb-11.4]`. By using the `mariadb-X.Y` syntax, you can create option files that have MariaDB-only options in the MariaDB-specific option groups. That would allow the option file to work for both MariaDB and MySQL.

Client Option Groups

MariaDB programs reads client options from the following option groups:

Group	Description
[client]	Options read by all MariaDB and MySQL client programs, including both MariaDB and MySQL clients. For example, mariadb-dump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

Tool-Specific Option Groups

Many MariaDB tools reads options from their own option groups as well:

Group	Description
[mysqld_safe]	Options read by mysqld_safe, including both MariaDB Server and MySQL Server.
[safe_mysqld]	Options read by mysqld_safe, including both MariaDB Server and MySQL Server.
[mariadbd-safe]	Options read by mysqld_safe from MariaDB Server.
[mariadb_safe]	Options read by mysqld_safe from MariaDB Server. Deprecated, please avoid using this.
[mariadb-backup]	Options read by mariadb-backup.
[xtrabackup]	Options read by mariadb-backup and Percona XtraBackup.
[mysql_upgrade]	Options read by mysql_upgrade, including both MariaDB Server and MySQL Server.
[mariadb-upgrade]	Options read by mariadb-upgrade.
[sst]	Specific options read by the mariadb-backup SST method and the xtrabackup-v2 SST method.
[mysql]	Options read by mysql, including both MariaDB Server and MySQL Server.
[mariadb-client]	Options read by mariadb.
[mysqldump]	Options read by mysqldump, including both MariaDB Server and MySQL Server.
[mariadb-dump]	Options read by mariadb-dump.
[mysqlimport]	Options read by mysqlimport, including both MariaDB Server and MySQL Server.
[mariadb-import]	Options read by mariadb-import.
[mysqlbinlog]	Options read by mysqlbinlog, including both MariaDB Server and MySQL Server.
[mariadb-binlog]	Options read by mariadb-binlog.
[mysqladmin]	Options read by mysqladmin, including both MariaDB Server and MySQL Server.
[mariadb-admin]	Options read by mariadb-admin.
[mysqlshow]	Options read by mysqlshow, including both MariaDB Server and MySQL Server.
[mariadb-show]	Options read by mariadb-show. Note that this program reads `[client]` group options, too.
[mysqlcheck]	Options read by mariadb-check, including both MariaDB Server and MySQL Server.
[mariadb-check]	Options read by mariadb-check.
[mysqlslap]	Options read by mysqlslap, including both MariaDB Server and MySQL Server.
[mariadb-slap]	Options read by mariadb-slap.
[odbc]	Options read by MariaDB Connector/ODBC, but only if the USE_MYCNF parameter is set.

Custom Option Group Suffixes

MariaDB can be configured to read options from option groups with a custom suffix by providing the following command-line argument. This command-line argument can be used with most of MariaDB's command-line tools, not just mariadbd. It must be given as the first argument on the command line:

Option	Description
–defaults-group-suffix =suffix	In addition to the default option groups, also read option groups with the given suffix.

The default group suffix can also be specified via the `MYSQL_GROUP_SUFFIX` environment variable.

Example: To create a custom prompt for the `mariadb` command-line client via a custom group, add this to the option file (for instance, `/etc/my.cnf`):

```
[client_custom_group]
prompt='(custom prompt for mariadb client)'
```

Then, launch `mariadb` like this:

```
mariadb --defaults-group-suffix=_custom_group
```

Including Option Files

It is possible to include additional option files from another option file. For example, to include `/etc/mysql/dbserver1.cnf`, an option file could contain:

```
[mariadb]
...
!include /etc/mysql/dbserver1.cnf
```

Including Option File Directories

It is also possible to include all option files in a directory from another option file. For example, to include all option files in `/etc/my.cnf.d/`, an option file could contain:

```
[mariadb]
...
!includedir /etc/my.cnf.d/
```

The option files within the directory are read in alphabetical order.

All option file names must end in `.cnf` on Unix-like operating systems. On Windows, all option file names must end in `.cnf` or `.ini`.

> **CURRENT**
>
> If a `.cnf` file cannot be read an executable (a MariaDB server or a client tool) will exit with an error. One can use `?includedir` to skip unreadable files without failing out.
>
> **< 11.4.10, < 11.8.6**
>
> If a `.cnf` file cannot be read an executable (a MariaDB server or a client tool) will exit with an error.

Checking Program Options

You can check which options a given program is going to use by using the –print-defaults command-line argument:

Option	Description
–print-defaults	Read options from option files, print all option values, and then exit the program.

This command-line argument can be used with most of MariaDB's command-line tools, not just mariadbd. It must be given as the first argument on the command-line:

```
$ mariadb-dump --print-defaults
mariadb-dump would have been started with the following arguments:
--ssl_cert=/etc/my.cnf.d/certificates/client-cert.pem --ssl_key=/etc/my.cnf.d/certificates/client-key.pem
--ssl_ca=/etc/my.cnf.d/certificates/ca.pem --ssl-verify-server-cert --max_allowed_packet=1G
```

You can also check which options a given program is going to use by using the my_print_defaults utility, and providing the names of the option groups that the program reads:

```
$ my_print_defaults mariadb-dump client client-server client-mariadb
--ssl_cert=/etc/my.cnf.d/certificates/client-cert.pem
--ssl_key=/etc/my.cnf.d/certificates/client-key.pem
--ssl_ca=/etc/my.cnf.d/certificates/ca.pem
--ssl-verify-server-cert
--max_allowed_packet=1GB
```

The `my_print_defaults` utility's `--mariadbd` command-line option provides a shortcut to refer to all of the server option groups:

```
$ my_print_defaults --mysqld
--log_bin=mariadb-bin
--log_slave_updates=ON
--ssl_cert=/etc/my.cnf.d/certificates/server-cert.pem
--ssl_key=/etc/my.cnf.d/certificates/server-key.pem
--ssl_ca=/etc/my.cnf.d/certificates/ca.pem
```

Verifying Option File Sources in Enterprise Server

After modifying a configuration file in Enterprise Server, you can determine the source of a system variable by querying the `INFORMATION_SCHEMA.SYSTEM_VARIABLES` table.

Check a specific variable

```
SELECT VARIABLE_NAME, GLOBAL_VALUE, GLOBAL_VALUE_PATH
FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES
WHERE VARIABLE_NAME = 'innodb_buffer_pool_size';   -- replace with your variable
```

Explanation of columns:

- `GLOBAL_VALUE_PATH`: Full path to the configuration file that set this value (e.g. `/etc/my.cnf.d/mariadb-enterprise.cnf`). Returns `NULL` if not set in any config file.
- `GLOBAL_VALUE_ORIGIN`: Shows the origin (config file, a compile-time default, or a command line, etc.).

The `global_value_path` column, introduced in MariaDB 10.5.0, shows the full path to the configuration file from which the variable's value was loaded. This is especially helpful in environments with multiple option files as it enables you to determine which configuration file is currently in effect for a given system setting.

Note: The `global_value_path` column requires MariaDB ES Server 10.5.0 or later. On previous versions, this column doesn't exist.

View all variable set from configuration files

To view all system variables that were set from configuration files, you can run the following:

```
SELECT VARIABLE_NAME, GLOBAL_VALUE, GLOBAL_VALUE_PATH
FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES
WHERE GLOBAL_VALUE_PATH IS NOT NULL;
```

This query returns all variables with values loaded from option files, along with their file paths.

Obfuscated Authentication Credential Option File

MySQL supports an obfuscated authentication credential option file called `.mylogin.cnf` that is created with mysql_config_editor.

MariaDB and MariaDB Enterprise Server do not support this feature. The passwords in MySQL's `.mylogin.cnf` are only obfuscated, rather than encrypted, so the feature does not really add much from a security perspective. It is more likely to give users a false sense of security, rather than to seriously protect them.

Option Prefixes

MariaDB supports certain prefixes that can be used with options. The supported option prefixes are:

Option Prefix	Description
autoset	Sets the option value automatically. Only supported for certain options.
disable	For all boolean options, disables the setting (equivalent to setting it to 0). Same as skip.
enable	For all boolean options, enables the setting (equivalent to setting it to 1).
loose	Don't produce an error if the option doesn't exist.
maximum	Sets the maximum value for the option.
skip	For all boolean options, disables the setting (equivalent to setting it to 0). Same as disable.

For example:

```
[mariadb]
...
# determine a good value for open_files_limit automatically
autoset_open_files_limit

# disable the unix socket plugin
disable_unix_socket

# enable the slow query log
enable_slow_query_log

# don't produce an error if these options don't exist
loose_file_key_management_filename = /etc/mysql/encryption/keyfile.enc
loose_file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
loose_file_key_management_encryption_algorithm = AES_CTR

# set max_allowed_packet to maximum value
maximum_max_allowed_packet

# disable external locking for MyISAM
skip_external_locking
```

Dashes and Underscores

Dashes (`-`) and underscores (`_`) **in option names** are interchangeable. Both variants can be used, for example:

```
[mariadbd]
debug-no-sync
debug_no_sync
```

The same is true for specifying **options on the command line**, for example:

```
mariadbd --debug-no-sync
mariadbd --debug_no_sync
```

> **Note:**
>
> In option files, however, you must omit the leading double-dash (`--`).

This is only different when referring to system variables at runtime. Here, only underscores can be used, for example:

```
MariaDB [(none)]> SELECT @debug_no_sync;
+----------------+
| @debug_no_sync |
+----------------+
| NULL           |
+----------------+
1 row in set (0.002 sec)

MariaDB [(none)]> SELECT @debug-no-sync;
ERROR 1054 (42S22): Unknown column 'no' in 'SELECT'
```

If an option is not explicitly set, the server or client uses the default value for that option.

MariaDB Server Options

MariaDB Server options can be set in server option groups.

For a list of options that can be set for MariaDB Server, see the list of options available for mariadbd.

Most of the server system variables can also be set in MariaDB's option file.

MariaDB Client Options

MariaDB client options can be set in client option groups.

See the specific page for each client program to determine what options are available for that program.

Example Option Files

Most MariaDB installations include a sample MariaDB option file called `my-default.cnf`.

In source distributions, the sample option files are usually found in the `support-files` directory, and in other distributions, in the `share/mysql` directory that is relative to the MariaDB base installation directory.

You can copy one of these sample MariaDB option files and use it as the basis for building your server's primary MariaDB option file.

Example Minimal Option File

Minimal `my.cnf` file that you can use to test MariaDB:

```
[client-server]
# Uncomment these if you want to use a nonstandard connection to MariaDB
#socket=/tmp/mysql.sock
#port=3306

# This will be passed to all MariaDB clients
[client]
#password=my_password

# The MariaDB server
[mysqld]
# Directory where you want to put your data
data=/usr/local/mysql/var
# Directory for the errmsg.sys file in the language you want to use
language=/usr/local/share/mysql/english

# This is the prefix name to be used for all log, error and replication files
log-basename=mysqld

# Enable logging by default to help find problems
general-log
log-slow-queries
```

Example Hybrid Option File

The following is an extract of an option file that one can use if one wants to work with both MySQL and MariaDB.

```
# Example mysql config file.

[client-server]
socket=/tmp/mysql-dbug.sock
port=3307

# This will be passed to all mariadb clients
[client]
password=my_password

# Here are entries for some specific programs
# The following values assume you have at least 32M ram

# The MariaDB server
[mysqld]
temp-pool
key_buffer_size=16M
datadir=/my/mysqldata
loose-innodb_file_per_table

[mariadb]
datadir=/my/data
default-storage-engine=aria
loose-mutex-deadlock-detector
max-connections=20

[mariadb-5.5]
language=/my/maria-5.5/sql/share/english/
socket=/tmp/mysql-dbug.sock
port=3307

[mariadb-10.1]
language=/my/maria-10.1/sql/share/english/
socket=/tmp/mysql2-dbug.sock

[mysqldump]
quick
max_allowed_packet=16M

[mysql]
no-auto-rehash
loose-abort-source-on-error
```

See Also

- Configuring MariaDB Connector/C with Option Files
- Troubleshooting Connection Issues
- MariaDB Enterprise Audit
- Information_schema.SYSTEM_VARIABLES Table
- Configuring MariaDB for Remote Client Access

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Environment Variables

MariaDB makes use of numerous environment variables that may be set on your system. Environment variables have the lowest precedence, so any options set on the command line or in an option file will take precedence.

It's usually better not to rely on environment variables, and to rather set the options you need directly, as this makes the system a little more robust and easier to administer.

Here is a list of environment variables used by MariaDB:

Environment Variable	Description
`HOME`	Default directory for the mysql_history file.
`MYSQL_DEBUG`	Debug trace options used when debugging.
`MYSQL_GROUP_SUFFIX`	In addition to the given option groups, also read groups with this suffix.
`MYSQL_HISTFILE`	Path to the mysql_history file, overriding the `$HOME/.mysql_history` setting.
`MYSQL_HOME`	Path to the directory containing the my.cnf file used by the server.
`MYSQL_HOST`	Default host name used by the mariadb command line client.
`MYSQL_PS1`	Command prompt for use by the mariadb command line client.
`MYSQL_PWD`	Default password when connecting to `mysqld`. It is strongly recommended to use a more secure method of sending the password to the server.
`MYSQL_TCP_PORT`	Default TCP/IP port number.
`MYSQL_UNIX_PORT`	On Unix, default socket file used for localhost connections.
`PATH`	Path to directories that hold executable programs (such as the mariadb client, mariadb-admin), so that these can be run from any location.
`TMPDIR`	Directory where temporary files are created.
`TZ`	Local time zone.
`UMASK`	Creation mode when creating files. See Specifying Permissions for Schema (Data) Directories and Tables.
`UMASK_DIR`	Creation mode when creating directories. See Specifying Permissions for Schema (Data) Directories and Tables.
`USER`	On Windows, up to MariaDB 5.5, the default user name when connecting to the mysqld server. API GetUserName() is used in later versions.

This page is licensed: CC BY-SA / Gnu FDL

Advanced Configuration

Articles of how to setup your MariaDB optimally on different systems

See Atomic Write Support.

Explains the concept of atomic writes in MariaDB, which improve performance and data integrity on SSDs by bypassing the InnoDB doublewrite buffer, supported on devices like Fusion-io and Shannon SSDs.

See Configuring Linux for MariaDB.

Guidance on tuning Linux kernel settings for MariaDB performance, including I/O schedulers (using `none` or `mq-deadline`), open file limits, and core file sizes.

See Configuring MariaDB for Optimal Performance.

Complete MariaDB performance tuning: innodb_buffer_pool_size, aria_pagecache_buffer_size, thread_handling configuration, and SHOW GLOBAL STATUS monitoring.

See Configuring Swappiness.

Recommendations for setting the Linux `vm.swappiness` kernel parameter (ideally to 1) to prevent the OS from swapping out MariaDB memory pages, which degrades performance.

See Fusion-io.

Introduction to using Fusion-io flash memory cards with MariaDB to significantly boost I/O throughput and reduce latency, including benefits like atomic write support.

This page is licensed: CC BY-SA / Gnu FDL

Atomic Write Support

> **Note:**
>
> For the OS-level meaning of `O_DIRECT` and the other `innodb_flush_method` values mentioned on this page, see Storage I/O: Buffering and Persistence.

Partial Write Operations

When Innodb writes to the filesystem, there is generally no guarantee that a given write operation will be complete (not partial) in cases of a poweroff event, or if the operating system crashes at the exact moment a write is being done.

Without detection or prevention of partial writes, the integrity of the database can be compromised after recovery.

`innodb_doublewrite` –an Imperfect Solution

Since its inception, Innodb has had a mechanism to detect and ignore partial writes via the InnoDB Doublewrite Buffer (also `innodb_checksum` can be used to detect a partial write).

Doublewrites, controlled by the innodb_doublewrite system variable, comes with its own set of problems. Especially on SSD, writing each page twice can have detrimental effects (write leveling).

Atomic Write - a Faster Alternative to `innodb_doublewrite`

A better solution is to directly ask the filesystem to provide an atomic (all or nothing) write guarantee. Currently this is only available on a few SSD cards.

Enabling Atomic Writes from MariaDB 10.2

When starting, MariaDB 10.2 and beyond automatically detects if any of the supported SSD cards are used.

When opening an InnoDB table, there is a check if the tablespace for the table is on a device that supports atomic writes and if yes, it will automatically enable atomic writes for the table. If atomic writes support is not detected, the doublewrite buffer will be used.

One can disable atomic write support for all cards by setting the variable innodb-use-atomic-writes to `OFF` in your my.cnf file. It's `ON` by default.

Enabling Atomic Writes in MariaDB 5.5 to MariaDB 10.1

To use atomic writes instead of the doublewrite buffer, add:

```
innodb_use_atomic_writes = 1
```

to the `my.cnf` config file.

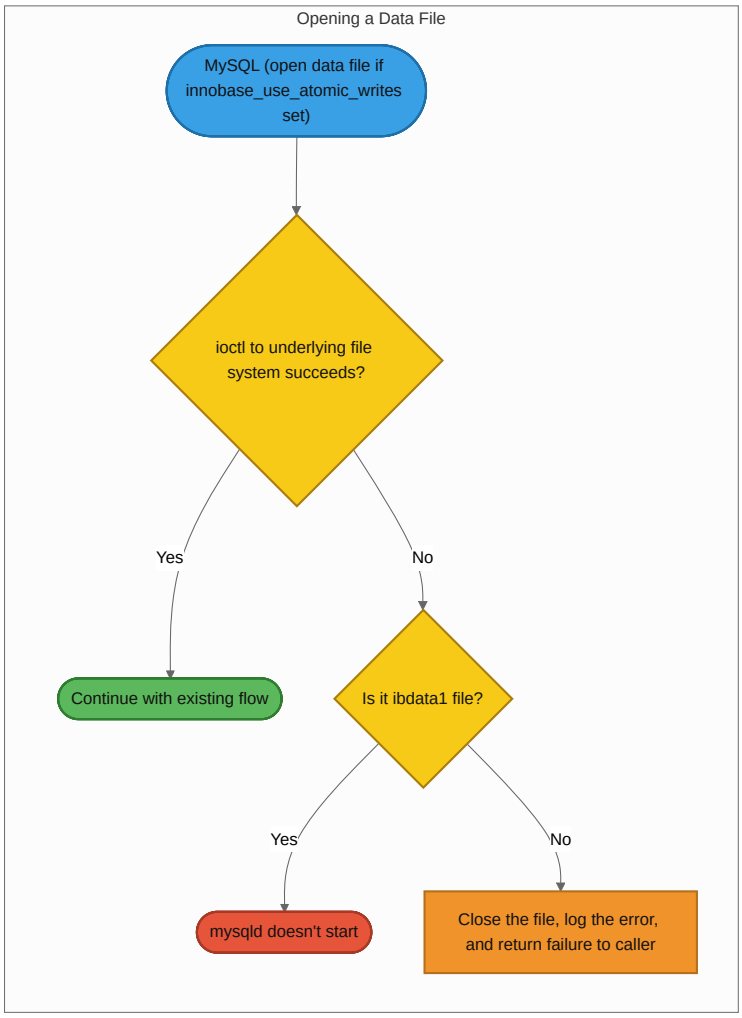
Note that atomic writes are only supported on Fusion-io devices that use the NVMFS file system in these versions of MariaDB.

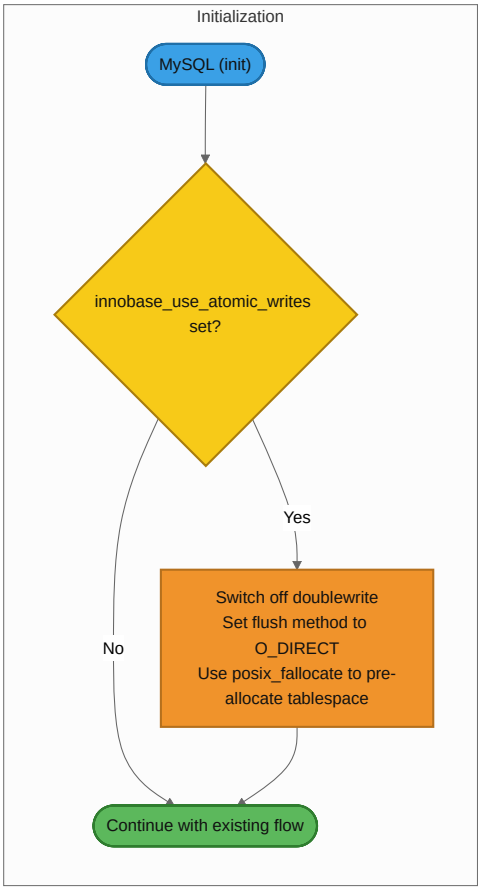
About innodb_use_atomic_writes (in MariaDB 5.5 to MariaDB 10.1)

The following happens when atomic writes are enabled

- if innodb_flush_method is neither `O_DIRECT`, `ALL_O_DIRECT`, or `O_DIRECT_NO_FSYNC`, it is switched to `O_DIRECT`
- innodb_use_fallocate is switched `ON` (files are extended using `posix_fallocate` rather than writing zeros behind the end of file)
- Whenever an Innodb datafile is opened, a special `ioctl()` is issued to switch on atomic writes. If the call fails, an error is logged and returned to the caller. This means that if the system tablespace is not located on an atomic write capable device or filesystem, InnoDB/XtraDB will refuse to start.
- if innodb_doublewrite is set to `ON`, `innodb_doublewrite` will be switched `OFF` and a message written to the error log.

Here is a flowchart showing how atomic writes work inside InnoDB:





Initialization checks the atomic-writes setting before continuing, while opening a data file checks the ioctl result, treating a failed ibdata1 open as fatal and any other file as a closable error.

Devices that Support Atomic Writes with MariaDB

MariaDB currently supports atomic writes on the following devices:

- Fusion-io devices with the NVMFS file system . MariaDB 5.5 and above.
- Shannon SSD. MariaDB 10.2 and above.

This page is licensed: CC BY-SA / Gnu FDL

Configuring Linux for MariaDB

Linux kernel settings

IO scheduler

For optimal IO performance running a database on modern hardware we recommend using the *none* (previously called *noop*) scheduler.

Recommended schedulers are *none*, for SSDs and NVMes, and *mq-deadline* (previously called *deadline*) for hard disks.

You can check your scheduler setting with:

```
cat /sys/block/${DEVICE}/queue/scheduler
```

For instance, it should look like this output:

```
cat /sys/block/vdb/queue/scheduler
[none] mq-deadline kyber bfq
```

Older kernels may look like:

```
cat /sys/block/sda/queue/scheduler
[noop] deadline cfq
```

Writing the new scheduler name to the same `/sys` node will change the scheduler:

```
echo noop >  /sys/block/vdb/queue/scheduler
```

The impact of schedulers depend significantly on workload and hardware. You can measure the IO-latency using the biolatency bcc-tools script with an aim to keep the mean as low as possible.

Resource Limits

Configuring the Open Files Limit

By default, the system limits how many open file descriptors a process can have open at one time. It has both a soft and hard limit. On many systems, both the soft and hard limit default to 1024. On an active database server, it is very easy to exceed 1024 open file descriptors. Therefore, you may need to increase the soft and hard limits. There are a few ways to do so.

If you are using mysqld_safe or mariadbd-safe to start `mysqld`, then see the instructions at mariadbd-safe: Configuring the Open Files Limit.

If you are using systemd to start `mysqld`, then see the instructions at systemd: Configuring the Open Files Limit.

Otherwise, you can set the soft and hard limits for the `mysql` user account by adding the following lines to /etc/security/limits.conf:

```
mysql soft nofile 65535
mysql hard nofile 65535
```

After the system is rebooted, the `mysql` user should use the new limits, and the user's `ulimit` output should look like the following:

```
$ ulimit -Sn
65535
$ ulimit -Hn
65535
```

Configuring the Core File Size

By default, the system limits the size of core files that could be created. It has both a soft and hard limit. On many systems, the soft limit defaults to 0. If you want to enable core dumps, then you may need to increase this. Therefore, you may need to increase the soft and hard limits. There are a few ways to do so.

If you are using mysqld_safe or mariadbd-safe to start `mysqld`, then see the instructions at mariadb-safe: Configuring the Core File Size.

If you are using systemd to start `mysqld`, then see the instructions at systemd: Configuring the Core File Size.

Otherwise, you can set the soft and hard limits for the `mysql` user account by adding the following lines to /etc/security/limits.conf:

```
mysql soft core unlimited
mysql hard core unlimited
```

After the system is rebooted, the `mysql` user should use the new limits, and the user's `ulimit` output should look like the following:

```
$ ulimit -Sc
unlimited
$ ulimit -Hc
unlimited
```

Swappiness

See configuring swappiness.

This page is licensed: CC BY-SA / Gnu FDL

Configuring MariaDB for Optimal Performance

This article will help you configure MariaDB for optimal performance.

By default, MariaDB is configured to work on a desktop system and therefore use relatively few resources. To optimize installation for a dedicated server, you have to do a few minutes of work.

For this article we assume that you are going to run MariaDB on a dedicated server.

Feel free to update this article if you have more ideas.

my.cnf Files

MariaDB is normally configured by editing the my.cnf file. In the next section you have a list of variables that you may want to configure for dedicated MariaDB servers.

InnoDB Storage Engine

InnoDB is normally the default storage engine with MariaDB.

- You should set innodb_buffer_pool_size to about 80% of your memory. The goal is to ensure that 80 % of your working set is in memory.

The other most important InnoDB variables are:

- innodb_log_file_size
- innodb_flush_method
- innodb_thread_sleep_delay

Some other important InnoDB variables:

- innodb_max_dirty_pages_pct_lwm
- innodb_read_ahead_threshold
- innodb_buffer_pool_instances. Deprecated and ignored from MariaDB 10.5.1.
- innodb_adaptive_max_sleep_delay. Deprecated and ignored from MariaDB 10.5.5.
- innodb_thread_concurrency. Deprecated and ignored from MariaDB 10.5.5.

Aria Storage Engine

- MariaDB uses by default the Aria storage engine for internal temporary files. If you have many temporary files, you should set aria_pagecache_buffer_size to a reasonably large value so that temporary overflow data is not flushed to disk. The default is 128M.

You can check if Aria is configured properly by executing:

```
MariaDB [test]> show global status like "Aria%";
```

```
+----------------------------------+-------+
| Variable_name                    | Value |
+----------------------------------+-------+
| Aria_pagecache_blocks_not_flushed | 0     |
| Aria_pagecache_blocks_unused      | 964   |
| Aria_pagecache_blocks_used        | 232   |
| Aria_pagecache_read_requests      | 9598  |
| Aria_pagecache_reads              | 0     |
| Aria_pagecache_write_requests     | 222   |
| Aria_pagecache_writes             | 0     |
| Aria_transaction_log_syncs        | 0     |
+----------------------------------+-------+
```

If `Aria_pagecache_reads` is much smaller than `Aria_pagecache_read_request` and `Aria_pagecache_writes` is much smaller than Aria_pagecache_write_request#, then your setup is good. If the aria_pagecache_buffer_size is big enough, the two variables should be 0, like above.

MyISAM

- If you don't use MyISAM tables explicitly (true for most MariaDB 10.4+ users), you can set key_buffer_size to a very low value, like 64K.

Using in memory temporary tables

Using memory tables for internal temporary results can speed up execution.
However, if the memory table gets full, then the memory table will be moved to disk, which can hurt performance.

You can check how the internal memory tables are performing by executing:

```
MariaDB [test]> show global status like "Created%tables%";
```

```
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 1     |
| Created_tmp_tables      | 2     |
+-------------------------+-------+
```

`Created_tmp_tables` is the total number of internal temporary tables created as part of executing queries like SELECT. `Created_tmp_disk_tables` shows how many of these did hit the storage.

You can increase the storage for internal temporary tables by setting max_heap_table_size and tmp_memory_table_size high enough. These values are per connection.

Lots of Connections

A Lot of Fast Connections + Small Set of Queries + Disconnects

- If you are doing a lot of fast connections / disconnects, you should increase back_log and if you are running MariaDB 10.1 or below thread_cache_size.
- If you have a lot (> 128) of simultaneous running fast queries, you should consider setting thread_handling to `pool_of_threads`.

Connecting From a Lot of Different Machines

- If you are connecting from a lot of different machines you should increase host_cache_size to the max number of machines (default 128) to cache the resolving of hostnames. If you don't connect from a lot of machines, you can set this to a very low value!

See Also

- MariaDB Memory Allocation
- Full List of MariaDB Options, System and Status Variables
- Server system variables
- mysqld options
- Performance schema helps you understand what is taking time and resources.
- Slow query log is used to find queries that are running slow.
- OPTIMIZE TABLE helps you defragment tables.

External Links

- what-to-tune-in-mysql-56-after-installation.html
- optimizing-mysql-configuration-percona-mysql-university-montevideo

This page is licensed: CC BY-SA / Gnu FDL

Configuring Swappiness

Why to Avoid Swapping

Obviously, accessing swap memory from disk is far slower than accessing RAM directly. This is particularly bad on a database server because:

- MariaDB's internal algorithms assume that memory is not swap, and are highly inefficient if it is. Some algorithms are intended to avoid or delay disk IO, and use memory where possible - performing this with swap can be worse than just doing it on disk in the first place.
- Swap increases IO over just using disk in the first place as pages are actively swapped in and out of swap. Even something like removing a dirty page that is no longer going to be stored in memory, while designed to improve efficiency, will under a swap situation cost more IO.
- Database locks are particularly inefficient in swap. They are designed to be obtained and released often and quickly, and pausing to perform disk IO will have a serious impact on their usability.

The main way to avoid swapping is to make sure you have enough RAM for all processes that need to run on the machine. Setting the system variables too high can mean that under load the server runs short of memory, and needs to use swap. So understanding what settings to use and how these impact your server's memory usage is critical.

Setting Swappiness on Linux

Linux has a swappiness setting which determines the balance between swapping out pages (chunks of memory) from RAM to a preconfigured swap space on the hard drive.

The setting is from `0` to `100`, with lower values meaning a lower likelihood of swapping. The default is usually `60` –you can check this by running:

```
sysctl vm.swappiness
```

The default setting encourages the server to use swap. Since there probably won't be much else on the database server besides MariaDB processes to put into swap, you'll probably want to reduce this to zero to avoid swapping as much as possible. You can change the default by adding a line to the `sysctl.conf` file (usually found in `/etc/sysctl.conf`).

To set the swappiness to zero, add the line:

```
vm.swappiness = 0
```

This normally takes effect after a reboot, but you can change the value without rebooting as follows:

```
sysctl -w vm.swappiness=0
```

Since RHEL 6.4, setting swappiness=0 more aggressively avoids swapping out, which increases the risk of OOM killing under strong memory and I/O pressure.

A low swappiness setting is recommended for database workloads. For MariaDB databases, it is recommended to set swappiness to a value of 1.

```
vm.swappiness = 1
```

Disabling Swap Altogether

While some disable swap altogether, and you certainly want to avoid any database processes from using it, it can be prudent to leave some swap space to at least allow the kernel to fall over gracefully should a spike occur. Having emergency swap available at least allows you some scope to kill any runaway processes.

This page is licensed: CC BY-SA / Gnu FDL

Fusion-io

See Fusion-io Introduction.

Fusion-io develops PCIe based NAND flash memory cards and related software that can be used to speed up MariaDB databases.

This page is licensed: CC BY-SA / Gnu FDL

Fusion-io Introduction

The ioDrive branded products can be used as block devices (super-fast disks) or to extend basic DRAM memory. ioDrive is deployed by installing it on an x86 server and then installing the card driver under the operating system.

All main line 64-bit operating systems and hypervisors are supported: RHEL, CentOS, SuSe, Debian, OEL etc. and VMware, Microsoft Windows/Server etc. Drivers and their features are constantly developed further.

ioDrive cards support software RAID and you can combine two or more physical cards into one logical drive. Through ioMemory SDK and its APIs, one can integrate and enable more thorough interworking between your own software and the cards - and cut latency.

The key differentiator between a Fusion-io and a legacy SSD/HDD is the following: **A Fusion-io card is connected directly on the system bus (PCIe)**, this enables high data transfer throughput (1.5 GB/s, 3.0 GB/s or 6GB/s) and the fast direct memory access (DMA) method can be used to transfer data. The ATA/SATA protocol stack is omitted and therefore latency is cut short. Fusion-io performance is dependent on server speed: the faster processors and the newer PCIe-bus version you have, the better is the ioDrive performance. Fusion-io memory is non-volatile, in other words, data remains on the card even when the server is powered off.

Use Cases

1. You can start by using ioDrive for database files that need heavy random access.
2. Whole database on ioDrive.
3. In some cases, Fusion-io devices allow for atomic writes, which allows the server to safely disable the doublewrite buffer.
4. Use ioDrive as a write-through read cache. This is possible on server level with Fusion-io directCache software or in VMware environments using ioTurbine software or the ioCache bundle product. Reads happen from ioDrive and all writes go directly to your SAN or disk.
5. Highly Available shared storage with ION. Have two different hosts, Fusion-io cards in them and share/replicate data with Fusion-io's ION software.
6. The luxurious Platinum setup: MariaDB Galera Cluster running on Fusion-io SLC cards on several hosts.

Atomic Writes

Starting with MariaDB 5.5.31, MariaDB Server supports atomic writes on Fusion-io devices that use the NVMFS (formerly called DirectFS) file system. Unfortunately, NVMFS was never offered under 'General Availability', and SanDisk declared that NVMFS would reach end-of-life in December 2015. Therefore, NVMFS support is no longer offered by SanDisk.

MariaDB Server does not currently support atomic writes on Fusion-io devices with any other file systems.

See atomic write support for more information about MariaDB Server's atomic write support.

Future Suggested Development

- Extend InnoDB disk cache to be stored on Fusion-io acting as extended memory.

Settings For Best Performance

Fusion-io memory can be formatted with different sector size of either 512 or 4096 bytes. Bigger sectors are expected to be faster, but only if I/O is done in blocks of 4KB or multiples of that. Speaking of MariaDB: if only InnoDB data files are stored in Fusion-io memory, all I/O is done in blocks of 16K and thus 4K sector size can be used. If the InnoDB redo log (I/O block size: 512 bytes) goes to the same Fusion-io memory, then short sectors should be used.

Note: XtraDB has the experimental feature of an increased InnoDB log block size of 4K. If this is enabled, then both redo log I/O and page I/O in InnoDB will match a sector size of 4K.

As of file systems: currently XFS is expected to yield the best performance with MariaDB. However depending on the exact kernel version and version of XFS code in use, one might be affected by a bug that severely limits XFS

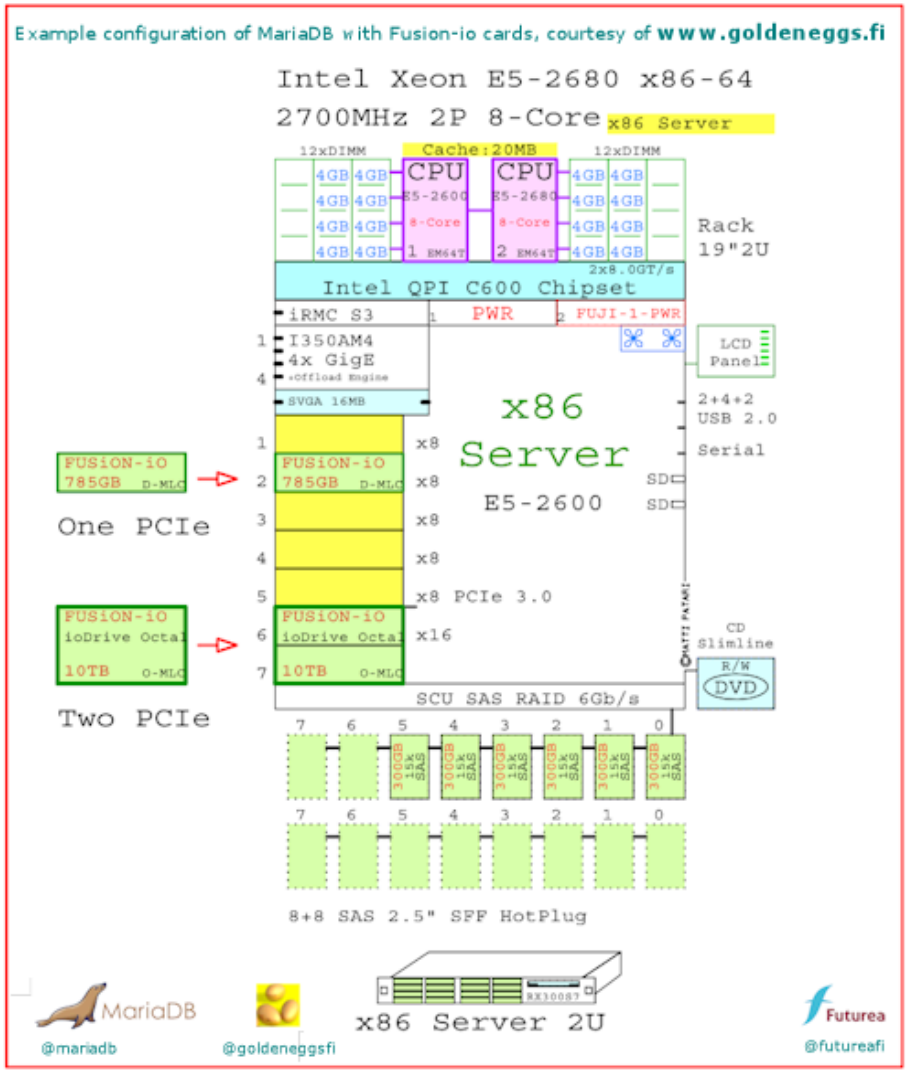
performance in concurrent environments. This has been fixed in kernel versions above 3.5 or RHEL6 kernels kernel-2.6.32-358 or later (because of bug 807503 being fixed).

For the pitbull machine where I have run such tests, ext4 was faster than xfs for 32 or more threads:

- up to 8 threads xfs was few percent faster (10% on average).
- at 16 threads it was a draw (2036 tps vs. 2070 tps).
- at 32 threads ext4 was 28% faster (2345 tps vs. 1829 tps).
- at 64 threads ext4 was even 47% faster (2362 tps vs. 1601 tps).
- at higher concurrency ext4 lost its bite, but was still constantly better than xfs.

Those numbers are for spinning disks. I guess for Fusion-io memory the XFS numbers will be even worse.

Example Configuration



GE-Fusionio-MariaDB

Card Models

There are several card models. ioDrive is older generation, ioDrive2 is newer. SLC sustains more writes. MLC is good enough for normal use.

1. ioDrive2, capacities per card 365GB, 785GB, 1.2TB with MLC. 400GB and 600GB with SLC, performance up to 535000 IOPS & 1.5GB/s bandwidth

2. ioDrive2 Duo, capacities per card 2.4TB MLC and 1.2TB SLC, performance up to 935000 IOPS & 3.0GB/s bandwidth
3. ioDrive, capacities per card 320GB, 640GB MLC and 160GB, 320GB SLC, performance up to 145000 IOPS & 790MB/s bandwidth
4. ioDrive Duo, capacities per card 640GB, 1.28TB MLC and 320GB, 640GB SLC, performance up to 285000 IOPS & 1.5GB/s bandwidth
5. ioDrive Octal, capacities per card 5TB and 10TB MLC, performance up to 1350000 IOPS & 6.7GB/s bandwidth
6. ioFX, a 420GB QDP MLC workstation product, 1.4GB/s bandwidth
7. ioCache, a 600GB MLC card with ioTurbine software bundle that can be used to speed up VMware based virtual hosts.
8. ioScale, 3.2TB card, building block to enable all-flash data center build out in hyperscale web and cloud environments. Product has been developed in co-operation with Facebook.

Additional Software

- directCache - transforms ioDrive to work as a read cache in your server. Writes go directly to your SAN
- ioTurbine - read cache software for VMware
- ION - transforms ioDrive into a shareable storage
- ioSphere - software to manage and monitor several ioDrives

See Also

- FusionIO atomic-series devices

This page is licensed: CC BY-SA / Gnu FDL

Upgrading MariaDB

See Platform Specific Upgrade Guides.

Provides tailored instructions for upgrading MariaDB on different operating systems, including Linux and Windows, and within specific environments like Galera Cluster.

See MariaDB Community Server Upgrade Paths.

An overview of supported upgrade paths for MariaDB Community Server, linking to specific guides for upgrading between major versions like 10.x and 11.x.

See MariaDB Enterprise Server Upgrade Paths.

Provides an overview of supported upgrade paths for MariaDB Enterprise Server, linking to specific guides for upgrading between major versions.

This page is licensed: CC BY-SA / Gnu FDL

Platform Specific Upgrade Guides

See Upgrading on Linux.

Guide to upgrading MariaDB on Linux.

See Upgrading MariaDB on Windows.

Guide to upgrading MariaDB on Windows using the MSI installer, supporting both minor upgrades and major version migrations using the upgrade wizard.

This page is licensed: CC BY-SA / Gnu FDL

Upgrading on Linux

See Upgrading Between Major MariaDB Versions.

Upgrade between major MariaDB versions, which is normally straightforward thanks to backward-compatible data files, connection protocol, and replication.

See Upgrading Between Minor MariaDB Versions.

Step-by-step minor version upgrade (e.g., 11.4.4 to 11.4.5) for MariaDB Community Server on Linux using YUM, APT, or ZYpp — backup, stop, upgrade packages, restart, and run mariadb-upgrade.

This page is licensed: CC BY-SA / Gnu FDL

Upgrading Between Major MariaDB Versions

MariaDB is designed to allow easy upgrades. You should be able to trivially upgrade from ANY earlier MariaDB version to the latest one (for example MariaDB 10.3.x to MariaDB 10.11.x), usually in a few seconds. This is also mainly true for any MySQL version < 8.0 to MariaDB 10.4 and up.

Upgrades are normally easy because:

- All MariaDB table data files are backward compatible
- The MariaDB connection protocol is backward compatible. You don't normally need to upgrade any of your old clients to be able to connect to a newer MariaDB version.
- The MariaDB replica can be of any newer version than the primary.

MariaDB Corporation regularly runs tests to check that one can upgrade from MariaDB 5.5 to the latest MariaDB version without any trouble. All older versions should work too (as long as the storage engines you were using are still around).

Note that if you are using MariaDB Galera Cluster, you have to follow the Galera upgrading instructions!

Requirements for Doing an Upgrade Between Major Versions

- Galera Cluster Users (Upgrading to 12.3+): You must manually install the `mariadb-server-galera` (or `MariaDB-server-galera`) package during the upgrade. Because the automatic dependency has been removed, a standard upgrade will not pull in the necessary cluster management components, which may prevent your cluster from bootstrapping.
- Go through the individual version upgrade notes (listed below) to look for any major changes or configuration options that have changed.
- Ensure that the target MariaDB version supports the storage engines you are using. For example, in 10.5 TokuDB is not supported.
- Back up the database (just in case). At least, take a copy of the `mysql` system database directory under the data directory with mariadb-dump –add-drop-table mysql (called `mysqldump` in MariaDB 10.3 and earlier) as most of the upgrade changes are done there (adding new fields and new system tables etc).
- Cleanly shutdown the server. This is necessary because even if data files are compatible between versions, recovery logs may not be.
  - Ensure that the innodb_fast_shutdown variable is not 2 (fast crash shutdown). The default of this variable is `1`.
  - innodb_force_recovery must be less than `3`.

Note that rpms don't support upgrading between major versions, only minor like 10.4.1 to 10.4.2. If you are using rpms, you should de-install the old MariaDB rpms and install the new MariaDB rpms before running mariadb-upgrade. Note that when installing the new rpms, mariadb-upgrade may be run automatically. There is no problem with running mariadb-upgrade many times.

Recommended Steps

- If you have a primary-replica setup, first upgrade one replica and when you have verified that the replica works well, upgrade the rest of the replicas (if any). Then upgrade one replica to primary, upgrade the primary, and change the replica to a primary.
- If you don't have a primary-replica setup, then take a backup, shutdown MariaDB and do the upgrade.

Step by Step Instructions for Upgrades

- Upgrade MariaDB binaries and libraries, preferably without starting MariaDB.
- If the MariaDB server process, mariadbd was not started as part of the upgrade, start it by executing `mariadbd --skip-grant-tables`. This may produce some warnings about some system tables not being up to date, but you can ignore these for now as mariadb-upgrade will fix that.
- Run mariadb-upgrade
- Restart MariaDB server.

Work Done by mariadb-upgrade

The main work done when upgrading is done by running mariadb-upgrade. The main things it does are:

- Updating the system tables in the `mysql` database to the newest version. This is very quick.
- mariadb-upgrade also runs mariadb-check –check-upgrade to check if there have been any collation changes between the major versions. This recreates indexes in old tables that are using any of the changed collations. This can take a bit of time if there are a lot of tables or there are many tables which used the changed collation. The last time a collation changed was in MariaDB/MySQL 5.1.23.

Post Upgrade Work

Check the MariaDB error log for any problems during upgrade. If there are any warnings in the log files, do your best to get rid of them!

The common warnings/errors are:

- Using obsolete options. If this is the case, remove them from your my.cnf files.
- Check the manual for new features that have been added since your last MariaDB version.
- Test that your application works as before. The main difference from before is that because of optimizer improvements your application should work better than before, but in some rare cases the optimizer may get something wrong. In this case, you can try to use explain, optimizer trace or optimizer_switch to fix the queries.

If Something Goes Wrong

- First, check the MariaDB error log to see if you are using configure options that are not supported anymore.
- Check the upgrade notices for the MariaDB release that you are upgrading to.
- File an issue in the MariaDB bug tracker so that we know about the issue and can provide a fix to make upgrades even better.
- Add a comment to this manual entry for how we can improve it.

Disaster Recovery

In the unlikely event something goes wrong, you can try the following:

- Remove the InnoDB tables from the `mysql` data directory. They are:
  - `gtid_slave_pos`
  - `innodb_table_stats`
  - `innodb_index_stats`

    - `transaction_registry`

- Move the `mysql` data directory to `mysql-old` and run mariadb-install-db to generate a new one.
- After the above, you have to add back your old users.
- When done, delete the `mysql-old` data directory.

Downgrading

MariaDB server is not designed for downgrading. That said, in most cases, as long as you haven't run any ALTER TABLE or CREATE TABLE statements and you have a mariadb-dump of your old `mysql` database , you should be able to downgrade to your previous version by doing the following:

- Do a clean shutdown. For this special case you have to set innodb_fast_shutdown to 0,before taking down the new MariaDB server, to ensure there are no redo or undo logs that need to be applied on the downgraded server.
- Delete the tables in the `mysql` database (if you didn't use the option `--add-drop-table` to mariadb-dump)
- Delete the new MariaDB installation
- Install the old MariaDB version
- Start the server with mariadbd –skip-grant-tables
- Install the old `mysql` database
- Execute in the mariadb client FLUSH PRIVILEGES

See Also

- Upgrading from MySQL to MariaDB
- Upgrading from MariaDB 10.11 to MariaDB 11.4
- Upgrading from MariaDB 10.6 to MariaDB 10.11
- Upgrading from MariaDB 10.6 to MariaDB 10.7
- Upgrading from MariaDB 10.5 to MariaDB 10.6
- Upgrading from MariaDB 10.4 to MariaDB 10.5
- Galera upgrading instructions
- innodb_fast_shutdown

This page is licensed: CC BY-SA / Gnu FDL

Upgrading Between Minor MariaDB Versions

These instructions detail a **minor release upgrade** of **MariaDB Community Server** on Linux. A minor release upgrade refers to upgrading within the same release series. For example, upgrading from MariaDB 11.4.4 to MariaDB 11.4.5 without changing the major version. Minor releases typically deliver bug fixes, security patches, and stability improvements.

For Windows, see Upgrading MariaDB on Windows. For major version upgrades (for example, MariaDB 10.11 to MariaDB 11.4), see Upgrading Between Major MariaDB Versions and the per-version guides listed in See Also.

> **Note:**
>
> The examples in this page use MariaDB Community Server 11.4.4 → 11.4.5 as the worked scenario. The same procedure applies to any supported minor-version upgrade within a release series.

Data Backup

Occasionally, issues can be encountered during an upgrade. These issues can even potentially corrupt the database's data files, making it difficult to revert to the previous installation. It is strongly recommended to create a

backup before upgrading. * If the upgrade fails, the backup can be used to restore the database to the previous version of MariaDB Server. * If the upgrade finishes successfully, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, see the Recovery Guide.

1. Take a full backup:

   ```
   sudo mariadb-backup --backup \
         --user=mariadb-backup_user \
         --password=mariadb-backup_passwd \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Replace `mariadb-backup_user` and `mariadb-backup_passwd` with your actual backup user credentials. Confirm successful completion of the backup operation.

2. Prepare the backup so it is ready for immediate restoration if required:

   ```
   sudo mariadb-backup --prepare \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Confirm successful completion of the prepare operation.

3. Backups should be tested before they are trusted. Restore the backup to a non-production instance and verify it before proceeding with the upgrade.

Stop the MariaDB Server Process

Before the new packages are installed, stop the running server.

1. Set the innodb_fast_shutdown system variable to `1` so InnoDB flushes all dirty pages and closes cleanly:

   ```
   SET GLOBAL innodb_fast_shutdown = 1;
   ```

2. Use XA RECOVER to confirm that there are no external XA transactions in a prepared state:

   ```
   XA RECOVER;
   ```

   Commit or roll back any open XA transactions before stopping the server.

3. Stop the server process. On distributions that use `systemd` (most supported OSes), manage the server with `systemctl`:

   ```
   sudo systemctl stop mariadb
   ```

Install the New Version

Unlike a major-version upgrade, a minor-version upgrade does **not** require uninstalling the old packages first. The distribution's package manager upgrades the packages in place, while keeping the data directory and configuration files unchanged.

MariaDB Community Server provides a setup script, `mariadb_repo_setup`, that configures the package repository for your distribution. To pin to a specific minor release, pass the full version number to `--mariadb-server-version`, prefixed with `mariadb-`. For example, to install MariaDB 11.4.5 instead of the latest 11.4.x release:

```
--mariadb-server-version="mariadb-11.4.5"
```

To track the latest release in a series rather than pinning to a specific minor, pass only the series:

```
--mariadb-server-version="mariadb-11.4"
```

UPGRADE VIA YUM

Upgrade via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Configure the YUM package repository for the target release. The following example uses MariaDB 11.4.5; adjust the version number for your target release:

   ```
   sudo yum install curl
   ```

   ```
   curl -LsSO https://downloads.mariadb.com/MariaDB/mariadb_repo_setup
   ```

   ```
   chmod +x mariadb_repo_setup
   ```

   ```
   sudo ./mariadb_repo_setup --mariadb-server-version="mariadb-11.4.5"
   ```

   See Pinning the Repository to a Specific Minor Release for more on minor-release pinning.

2. Update MariaDB Community Server and package dependencies:

   ```
   sudo yum update "MariaDB-*" "galera*"
   ```

   Check that the wildcards do not unintentionally match any custom applications.

UPGRADE VIA APT

Upgrade via APT (Debian, Ubuntu)

1. Configure the APT package repository for the target release. The following example uses MariaDB 11.4.5; adjust the version number for your target release:

   ```
   sudo apt install curl
   ```

   ```
   curl -LsSO https://downloads.mariadb.com/MariaDB/mariadb_repo_setup
   ```

   ```
   chmod +x mariadb_repo_setup
   ```

   ```
   sudo ./mariadb_repo_setup --mariadb-server-version="mariadb-11.4.5"
   ```

   ```
   sudo apt update
   ```

2. Upgrade MariaDB Community Server and package dependencies. The `--only-upgrade` flag ensures `apt` does not install new packages, only updates existing ones:

   ```
   sudo apt install --only-upgrade "mariadb-*" "galera*"
   ```

   Check that the wildcards do not unintentionally match any custom applications.

UPGRADE VIA ZYPP

Upgrade via ZYpp (SLES, openSUSE)

1. Configure the ZYpp package repository for the target release. The following example uses MariaDB 11.4.5; adjust the version number for your target release:

```
sudo zypper install curl
```

```
curl -LsSO https://downloads.mariadb.com/MariaDB/mariadb_repo_setup
```

```
chmod +x mariadb_repo_setup
```

```
sudo ./mariadb_repo_setup --mariadb-server-version="mariadb-11.4.5"
```

2. Update MariaDB Community Server and package dependencies:

   ```
   sudo zypper update "MariaDB-*" "galera*"
   ```

   Check that the wildcards do not unintentionally match any custom applications.

Configuration

For platforms that use YUM or ZYpp as a package manager, MariaDB Community Server's packages bundle several configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

If your version of any of these configuration files contained custom edits, then the package manager may save your edited version with the `.rpmsave` extension during the upgrade. To continue using your customized version, restore it before starting the server. For example, to restore `server.cnf`:

```
sudo mv /etc/my.cnf.d/server.cnf /etc/my.cnf.d/server.cnf.original
```

```
sudo mv /etc/my.cnf.d/server.cnf.rpmsave /etc/my.cnf.d/server.cnf
```

On APT-based systems (Debian, Ubuntu), preserved configuration files use the `.dpkg-old` extension instead. The principle is the same: rename and restore as needed.

Now is also the time to make any desired changes to option files such as `my.cnf`, before starting the new server.

Starting the Server

MariaDB Community Server includes configuration to start, stop, restart, enable/disable on boot, and check the status of the server using the operating system's default process management system.

For distributions that use `systemd`, manage the server process using `systemctl`:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Upgrading the Data Directory

MariaDB ships with a utility that checks for and corrects any compatibility issues introduced by the upgrade. After the server is upgraded and running, run mariadb-upgrade to upgrade the data directory:

```
sudo mariadb-upgrade
```

Note:

Minor-version upgrades do not change the data dictionary. On mariadb-upgrade 2.0 and later, `mariadb-upgrade` detects this automatically and exits immediately with a message confirming no upgrade is needed and this is expected behaviour. Running `mariadb-upgrade` after a minor upgrade is still recommended as a safety check, and is harmless when no work is needed. Use `--force` to run the full check regardless.

Testing

When the upgraded server is up and running, verify it is working and there were no issues during startup.

1. Connect to the server using the mariadb client as the `root@localhost` user:

   ```
   sudo mariadb
   ```

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MariaDB connection id is 9
   Server version: 11.4.5-MariaDB MariaDB Server

   Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

   MariaDB [(none)]>
   ```

2. Verify the server version by checking the version system variable:

   ```
   SHOW GLOBAL VARIABLES LIKE 'version';
   ```

   ```
   +---------------+-----------------+
   | Variable_name | Value           |
   +---------------+-----------------+
   | version       | 11.4.5-MariaDB  |
   +---------------+-----------------+
   ```

3. Verify by calling the VERSION() function:

   ```
   SELECT VERSION();
   ```

   ```
   +-----------------+
   | VERSION()       |
   +-----------------+
   | 11.4.5-MariaDB  |
   +-----------------+
   ```

See Also

For major version upgrades (between release series), see:

- Upgrading Between Major MariaDB Versions
- Upgrading from MariaDB 11.4 to MariaDB 11.8
- Upgrading from MariaDB 10.11 to MariaDB 11.4

- Upgrading from MariaDB 10.6 to MariaDB 10.11

For the equivalent Enterprise Server minor-upgrade guides, see:

- Upgrade MariaDB Enterprise Server from 11.4.X to 11.4.Y
- Upgrade MariaDB Enterprise Server from 10.6.X to 10.6.Y

Related references:

- MariaDB Package Repository Setup and Usage — full reference for `mariadb_repo_setup` options, including `--mariadb-server-version` and minor-release pinning.
- Installing MariaDB Packages with YUM
- Installing MariaDB .deb Files (APT)
- Installing MariaDB with Zypper
- mariadb-upgrade — the post-upgrade data-directory utility.
- mariadb-backup — the recommended pre-upgrade backup tool.

This page is licensed: CC BY-SA / Gnu FDL

Upgrading MariaDB on Windows

For incompatibilities such as removed features, and changes to variables, see the pages describing changes by version on Upgrading MariaDB.

Minor Upgrades

To install a minor upgrade with MSI, just download the MSI and install it. It will do everything that needs to be done for minor upgrade automatically - shutdown MariaDB service(s), replace executables and DLLs, and start service(s) again.

The rest of the article is dedicated to *major* upgrades, e.g 10.1.x to 10.2.y.

General Information on Upgrade and Version Coexistence

This section assumes MSI installations.

First, check everything listed in the Incompatibilities section of the article relating to the version you are upgrading, for example, Upgrading from MariaDB 10.1 to MariaDB 10.2, to make sure you are prepared for the upgrade.

MariaDB (and also MySQL) allows different versions of the product to co-exist on the same machine, as long as these versions are different either in major or minor version numbers. For example, it is possible to have say MariaDB 5.1.51 and 5.2.6 to be installed on the same machine.

However only a single instance of 5.2 can exist. If for example 5.2.7 is installed on a machine where 5.2.6 is already installed, the installer will just replace 5.2.6 executables with 5.2.7 ones.

Now imagine, that both 5.1 and 5.2 are installed on the same machine and we want to upgrade the database instance running on 5.1 to the new version. In this case special tools are required. Traditionally, mysql_upgrade is used to accomplish this. On Windows, theMySQL upgrade is a complicated multiple-step manual process.

Since MariaDB 5.2.6, the Windows distribution includes tools that simplify migration between different versions and also allow migration between MySQL and
MariaDB.

Note. Automatic upgrades are only possible for DB instances that run as a
Windows service.

General Recommendations

Important: Ignore any statement that tells you to *"just uninstall MySQL and install MariaDB"*. This does not work on Windows, never has, and never will. Keep your MySQL installed until after the database had been converted.

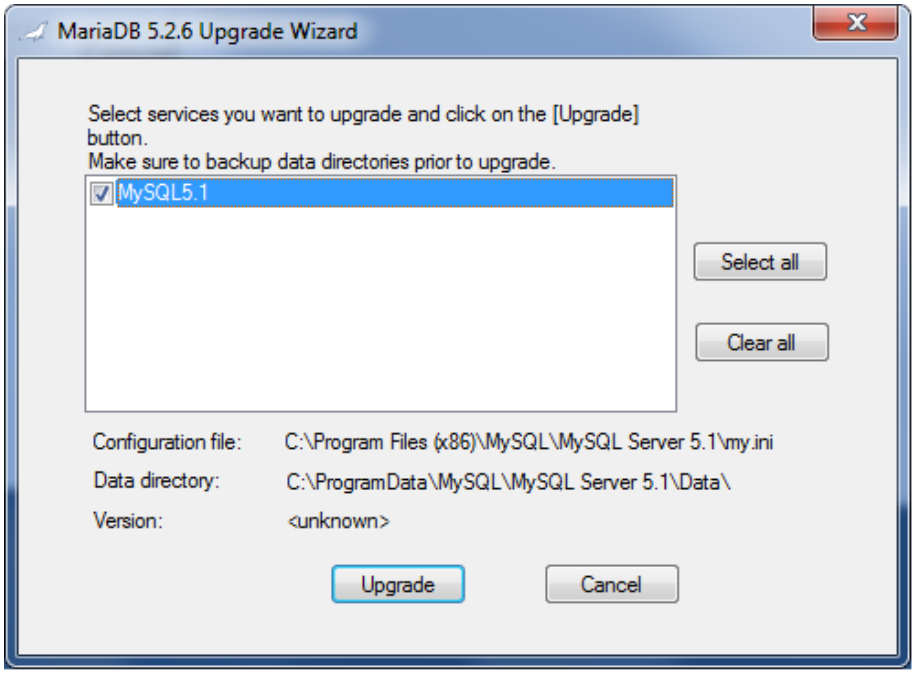
The following install/upgrade sequence is recommended in case of "major" upgrades, like going from 5.3 to 5.5

- Install new version, while still retaining the old one
- Upgrade services one by one, like described later in the document (e.g with mysql_upgrade_service). It is recommended to have services cleanly shut down before the upgrade.
- Uninstall old version when previous step is done.

Note. This recommendation differs from the procedure on Unixes, where the upgrade sequence is "uninstall old version, install new version"

Upgrade Wizard

This is a GUI tool that is typically invoked at the end of a MariaDB installation if upgradable services are found. The UI allows you to select instances you want to upgrade.



UpgradeWizard

`mysql_upgrade_service`

This is a command line tool that performs upgrades. The tool requires full administrative privileges (it has to start and stop services).

Example usage:

```
mysql_upgrade_service --service=MySQL
```

`mysql_upgrade_service` accepts a single parameter — the name of the MySQL or MariaDB service. It performs all the steps to convert a MariaDB/MySQL instance running as the service to the current version.

Migration to 64 bit MariaDB from 32 bit

Earlier we said that only single instance of "MariaDB ." version can be installed on the same machine. This was almost correct, because MariaDB

MSI installations allow 32 and 64-bit versions to be installed on the same machine, and in this case it is possible to have two instances of say 5.2 installed at the same time, an x86 one and an x64 one. One can use the x64 Upgrade wizard to upgrade an instance running as a 32-bit process to run as 64-bit.

Upgrading ZIP-based Installations.

Both UpgradeWizard and mysql_upgrade_service can also be used to upgrade database instances that were installed with the ZIP installation.

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MySQL to MariaDB

See Migrating to MariaDB from MySQL.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Community Server Upgrade Paths

See Upgrading from MariaDB 11.8 to MariaDB 12.3.

Upgrade guide for moving from MariaDB 11.8 to 12.3, covering removed options and replication caveats.

See Upgrading from MariaDB 11.4 to MariaDB 11.8.

Upgrade guide for moving from MariaDB 11.4 to 11.8, covering new features like vector search, optimizer improvements, and data type enhancements.

See Upgrading from MariaDB 11.3 to MariaDB 11.4.

Instructions for upgrading from the rolling release 11.3 to the long-term support release 11.4, detailing package updates and system table upgrades.

See Upgrading from MariaDB 10.11 to MariaDB 11.4.

Guide for upgrading from the previous LTS version 10.11 to 11.4, highlighting major optimizer changes, replication improvements, and SSL defaults.

See Upgrading from MariaDB 10.6 to MariaDB 10.11.

Complete MariaDB 10.6 to 10.11 upgrade: repository config, apt-get/yum/zypper remove+install, my.cnf option changes, and run mariadb-upgrade.

See Upgrading from MariaDB 10.5 to MariaDB 10.6.

Instructions for upgrading to MariaDB 10.6, noting significant changes like the default character set switch to `utf8mb3` and atomic DDL support.

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 11.8 to MariaDB 12.3

This page includes details for upgrading from MariaDB 11.8, the previous long-term maintenance version, to MariaDB 12.3.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 12.3.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.
2. Stop MariaDB.
3. Uninstall the old version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
    3. On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`
4. Install the new version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.
5. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
6. Start MariaDB.
7. Run mariadb-upgrade, to:
    1. Ensure that the system tables in the mysql database are fully compatible with the new version.
    2. Perform a very quick check of all tables and marks them as compatible with the new version of MariaDB.

Incompatible Changes Between 11.8 and 12.3

On most servers upgrading from 11.8 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
big_tables	Deprecated in MariaDB 10.5.0 and removed in MariaDB 12.0. No longer needed, as the server handles large result sets automatically.
storage_engine	Deprecated alias for `default_storage_engine` since MariaDB 5.5, and removed in MariaDB 12.0. Use default_storage_engine instead.
large_page_size	Read-only variable deprecated in MariaDB 10.5.3 and removed in MariaDB 12.0.

New Reserved Words

`CONVERSION`, `ST_COLLECT`, and `TO_DATE` are now reserved words. This only affects you if you use one of them as an unquoted identifier (for example, a table, column, or variable name) — quote it or rename it before upgrading, otherwise the statement will fail to parse. For the full list of 12.3 changes, see Features in MariaDB 12.3.

Changes in Replication Behavior

> **Warning:**
>
> When upgrading a replica from a pre-12.3 release to MariaDB 12.3.2, the CHANGE MASTER TO … master_use_gtid setting is not carried over and is reset to `DEFAULT`. This is fixed in 12.3.3 via MDEV-39788. After upgrading, check the replication configuration and re-apply `master_use_gtid` if you rely on it. Downgrading is not affected.

Deprecated Options

The `MYSQLD_OPTS` environment variable for the systemd service is deprecated. Place configuration options directly into option files, such as `my.cnf`, instead.

See Also

- Features in MariaDB 12.3
- Features in MariaDB 11.8
- Upgrading from MariaDB 11.8 to MariaDB 12.3 with Galera Cluster
- Upgrading from MariaDB 11.4 to MariaDB 11.8
- Upgrading from MariaDB 10.11 to MariaDB 11.4

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 11.4 to MariaDB 11.8

This page includes details for upgrading from MariaDB 11.4 to the subsequent long-term maintenance version, MariaDB 11.8.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 11.8.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.
2. Stop MariaDB.
3. Uninstall the old version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`

2. On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
3. On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

4. Install the new version of MariaDB.
   1. On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
   2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
   3. On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.
5. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
6. Start MariaDB.
7. Run mariadb-upgrade, to:
   1. Ensure that the system tables in the mysql database are fully compatible with the new version.
   2. Perform a very quick check of all tables and marks them as compatible with the new version of MariaDB.

Incompatible Changes Between 11.4 and 11.8

On most servers upgrading from 11.4 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
wsrep_load_data_splitting	Deprecated in MariaDB 10.4, defaults to `OFF`.

Options That Have Changed Default Values

N/A

Changes in Transaction Behavior

In MariaDB 11.8, InnoDB transactions under the Repeatable Read isolation level behave differently due to changes to the innodb_snapshot_isolation handling.

Therefore, in cases where no problem was previously provided, statements that modify data (such as DELETE or UPDATE) may now generate the following error:

```
ERROR 1020 (HY000): Record has changed since last read in table 't'
```

Description

A transaction under the Repeatable Read isolation level runs on a consistent snapshot of the data taken at the time of its initial read. A conflict may be identified when the initial transaction tries to change the same data after the snapshot has been created.

Such conflicts are more rigorously recognized in MariaDB 11.8. For example:

- If another transaction has added, altered, or removed rows in the meantime, a transaction that reads a table and then tries to delete or update data may fail.

- Even if the change impacts rows that weren't visible in the initial snapshot, this may still occur.

Example

```
-- Session 1
BEGIN;
SELECT * FROM t;

-- Session 2
BEGIN;
INSERT INTO t VALUES (2);
COMMIT;

-- Session 1
DELETE FROM t;
-- ERROR 1020 (HY000): Record has changed since last read in table 't'
```

Differences in Behavior by Isolation Level

- **Repeatable Read**
  When a transaction tries to modify data based on an outdated snapshot, the server detects conflicts, rolls back the transactions, and returns `ERROR 1020`.
- **Serializable**
  Conflicting transactions may be blocked earlier due to stricter locking. Depending on the execution sequence, the error may occur in another transaction or be avoided entirely.

Impact

After updating to MariaDB 11.8, applications that depend on Repeatable Read transactions may experience additional `ERROR 1020` failures. This may have an impact on:

- `DELETE` or `UPDATE` operations in bulk
- Workloads with simultaneous changes

For more details, see innodb_snapshot_isolation.

Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

N/A

See Also

- Features in MariaDB 11.8
- Features in MariaDB 11.4
- Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster
- Upgrading from MariaDB 10.11 to MariaDB 11.4
- Upgrading from MariaDB 10.6 to MariaDB 10.11
- Upgrading from MariaDB 10.5 to MariaDB 10.6

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 11.3 to MariaDB 11.4

This page includes details for upgrading from MariaDB 11.3 to MariaDB 11.4. Note that MariaDB 11.3 is a rolling release, and MariaDB 11.4 is a long-term maintenance release. After MariaDB 11.4.2, one can continue to the next rolling release, 11.5.2, 11.6.2 and so on, or remain on the long-term series, MariaDB 11.4.3. MariaDB 11.4.4 etc.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

For MariaDB Galera Cluster, see Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster instead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 11.4.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.
2. Stop MariaDB.
3. Uninstall the old version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
    3. On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`
4. Install the new version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.
5. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
6. Start MariaDB.
7. Run mariadb-upgrade, to:
    1. Ensure that the system tables in the mysql database are fully compatible with the new version.
    2. Perform a very quick check of all tables and marks them as compatible with the new version of MariaDB.

Incompatible Changes Between 11.3 and 11.4

Options That Have Been Removed

The following options should be removed if you use them in your option files:

Option	Reason
debug_no_thread_alarm	Unused code.

See Also

- Features in MariaDB 11.4

- Features in MariaDB 11.3
- Upgrading from MariaDB 11.2 to MariaDB 11.3
- Upgrading from MariaDB 11.1 to MariaDB 11.2
- Upgrading from MariaDB 10.11 to MariaDB 11.4
- Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.11 to MariaDB 11.4

This page includes details for upgrading from MariaDB 10.11 to the subsequent long-term maintenance version, MariaDB 11.4.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

For MariaDB Galera Cluster, see Upgrading Galera Cluster section.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaD11.4.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.
2. Stop MariaDB.
3. Uninstall the old version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
    3. On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`
4. Install the new version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.
5. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
6. Start MariaDB.
7. Run mariadb-upgrade, to:
    1. Ensure that the system tables in the mysql database are fully compatible with the new version.

2. Perform a very quick check of all tables and marks them as compatible with the new version of MariaDB.

Incompatible Changes Between 10.11 and 11.4

On most servers upgrading from 10.11 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
debug_no_thread_alarm	Unused code.
innodb_defragment	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_fill_factor	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_fill_factor_n_recs	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_frequency	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_n_pages	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_stats_accuracy	Defragmenting InnoDB Tablespaces in this manner no longer supported.

Options That Have Changed Default Values

Option	Old default	New default
optimizer_switch		See optimizer-switch.
innodb_purge_batch_size	300	1000

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
old_alter_table	Superseded by alter_algorithm.

Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

Option	Reason
innodb_purge_rseg_truncate_frequency	The motivation for introducing this in MySQL seems to have been to avoid stalls due to freeing undo log pages or truncating undo log tablespaces. In MariaDB, innodb_undo_log_truncate=ON should be a much lighter operation because it will not involve any log checkpoint, hence this is deprecated and ignored
tx_isolation	Replaced with transaction_isolation to align the option and system variable.
tx_read_only	Replaced with transaction_read_only to align the option and system variable.

See Also

- Features in MariaDB 11.4
- Features in MariaDB 11.3
- Features in MariaDB 11.2
- Features in MariaDB 11.1
- Features in MariaDB 11.0

- Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster
- Upgrading from MariaDB 10.6 to MariaDB 10.11
- Upgrading from MariaDB 10.5 to MariaDB 10.6

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.6 to MariaDB 10.11

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

For MariaDB Galera Cluster, see Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.11. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
    3. On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`
3. Install the new version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.
4. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
5. Start MariaDB.
6. Run mariadb-upgrade, to:
    1. Ensure that the system tables in the mysql database are fully compatible with the new version.
    2. Perform a very quick check of all tables and marks them as compatible with the new version of MariaDB.

Incompatible Changes Between 10.6 and 10.11

On most servers upgrading from 10.6 should be painless. However, there are some things that have changed which could affect an upgrade:

Compression

If a non-zlib compression algorithm was used in InnoDB or Mroonga before upgrading to 10.11, those tables will be unreadable until the appropriate compression library is installed. See Compression Plugins#Upgrading.

Options That Have Changed Default Values

Option	Old default	New default
innodb_buffer_pool_chunk_size	134217728	Autosized
spider_auto_increment_mode	-1	0
spider_bgs_first_read	-1	2
spider_bgs_mode	-1	0
spider_bgs_second_read	-1	100
spider_bka_mode	-1	1
spider_bka_table_name_type	-1	1
spider_buffer_size	-1	16000
spider_bulk_size	-1	16000
spider_bulk_update_mode	-1	0
spider_bulk_update_size	-1	16000
spider_casual_read	-1	0
spider_connect_timeout	-1	6
spider_crd_bg_mode	-1	2
spider_crd_interval	-1	51
spider_crd_mode	-1	1
spider_crd_sync	-1	0
spider_crd_type	-1	2
spider_crd_weight	-1	2
spider_delete_all_rows_type	-1	1
spider_direct_dup_insert	-1	0
spider_direct_order_limit	-1	9223372036854775807
spider_error_read_mode	-1	0
spider_error_write_mode	-1	0
spider_first_read	-1	0
spider_init_sql_alloc_size	-1	1024
spider_internal_limit	-1	9223372036854775807
spider_internal_offset	-1	0
spider_internal_optimize	-1	0
spider_internal_optimize_local	-1	0
spider_load_crd_at_startup	-1	1
spider_load_sts_at_startup	-1	1

Option	Old default	New default
spider_low_mem_read	-1	1
spider_max_order	-1	32767
spider_multi_split_read	-1	100
spider_net_read_timeout	-1	600
spider_net_write_timeout	-1	600
spider_quick_mode	-1	3
spider_quick_page_byte	-1	10485760
spider_quick_page_size	-1	1024
spider_read_only_mode	-1	0
spider_reset_sql_alloc	-1	1
spider_second_read	-1	0
spider_selupd_lock_mode	-1	1
spider_semi_split_read	-1	2
spider_semi_split_read_limit	-1	1
spider_semi_table_lock_connection	-1	1
spider_semi_table_lock	1	0

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
innodb_log_write_ahead_size	On Linux and Windows, the physical block size of the underlying storage is instead detected and used.
innodb_version	Redundant
wsrep_replicate_myisam	Use wsrep_mode instead.
wsrep_strict_ddl	Use wsrep_mode instead.

Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

Option	Reason
keep_files_on_create	MariaDB now deletes orphan files, so this setting should never be necessary.

See Also

- Features in MariaDB 10.11
- Features in MariaDB 10.10
- Features in MariaDB 10.9
- Features in MariaDB 10.8
- Features in MariaDB 10.7
- Upgrading from MariaDB 10.6 to MariaDB 10.7 with Galera Cluster
- Upgrading from MariaDB 10.7 to MariaDB 10.8
- Upgrading from MariaDB 10.6 to MariaDB 10.7
- Upgrading from MariaDB 10.5 to MariaDB 10.6

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.5 to MariaDB 10.6

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

For MariaDB Galera Cluster, see Upgrading from MariaDB 10.5 to MariaDB 10.6 with Galera Cluster.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.6.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.
2. Stop MariaDB.
3. Uninstall the old version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
    3. On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`
4. Install the new version of MariaDB.
    1. On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
    2. On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
    3. On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.
5. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
6. Start MariaDB.
7. Run mariadb-upgrade, to:
    1. Ensure that the system tables in the mysql database are fully compatible with the new version.
    2. Perform a very quick check of all tables and marks them as compatible with the new version of MariaDB.

Incompatible Changes Between 10.5 and 10.6

On most servers upgrading from 10.5 should be painless. However, there are some things that have changed which could affect an upgrade:

The behaviour of sorting non-deterministic variables in a Select query can be changed , see (MDEV-27745)

Reserved Word

- New reserved word: `OFFSET`. This can no longer be used as an identifier without being quoted.

InnoDB COMPRESSED Row Format

From MariaDB 10.6.0 until MariaDB 10.6.5, tables that are of the `COMPRESSED` row format are read-only by default. This was intended to be the first step towards removing write support and deprecating the feature.

This plan has been scrapped, and from MariaDB 10.6.6, `COMPRESSED` tables are no longer read-only by default.

From MariaDB 10.6.0 to MariaDB 10.6.5, set the innodb_read_only_compressed variable to `OFF` to make the tables writable.

Character Sets

From MariaDB 10.6, the `utf8` character set (and related collations) is by default an alias for `utf8mb3` rather than the other way around. It can be set to imply `utf8mb4` by changing the value of the old_mode system variable.

Options That Have Changed Default Values

Option	Old default value	New default value
character_set_client	utf8	utf8mb3
character_set_connection	utf8	utf8mb3
character_set_results	utf8	utf8mb3
character_set_system	utf8	utf8mb3
innodb_flush_method	fsync	O_DIRECT
old_mode	Empty	UTF8_IS_UTF8MB3

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
innodb_adaptive_max_sleep_delay	
innodb_background_scrub_data_check_interval	
innodb_background_scrub_data_compressed	
innodb_background_scrub_data_interval	
innodb_background_scrub_data_uncompressed	
innodb_buffer_pool_instances	
innodb_checksum_algorithm	The variable is still present, but the *innodb and *none options have been removed as the crc32 algorithm only is supported from MariaDB 10.6.
innodb_commit_concurrency	
innodb_concurrency_tickets	
innodb_file_format	
innodb_large_prefix	
innodb_lock_schedule_algorithm	

Option	Reason
innodb_log_checksums	
innodb_log_compressed_pages	
innodb_log_files_in_group	
innodb_log_optimize_ddl	
innodb_page_cleaners	
innodb_replication_delay	
innodb_scrub_log	
innodb_scrub_log_speed	
innodb_sync_array_size	
innodb_thread_concurrency	
innodb_thread_sleep_delay	
innodb_undo_logs	

Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

Option	Reason
wsrep_replicate_myisam	Use wsrep_mode instead.
wsrep_strict_ddl	Use wsrep_mode instead.

Major New Features To Consider

- See also System Variables Added in MariaDB 10.6.

See Also

- Features in MariaDB 10.6
- Upgrading from MariaDB 10.5 to MariaDB 10.6 with Galera Cluster
- Upgrading from MariaDB 10.4 to MariaDB 10.5
- Upgrading from MariaDB 10.3 to MariaDB 10.4
- Upgrading from MariaDB 10.2 to MariaDB 10.3

This page is licensed: CC BY-SA / Gnu FDL

Archived Guides (Unmaintained CS Versions)

See Upgrading from MariaDB 11.2 to MariaDB 11.3.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 11.1 to MariaDB 11.2.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 11.0 to MariaDB 11.1.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 10.11 to MariaDB 11.0.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 10.7 to MariaDB 10.8.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 10.6 to MariaDB 10.7.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 10.4 to MariaDB 10.5.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 10.3 to MariaDB 10.4.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 10.2 to MariaDB 10.3.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 10.1 to MariaDB 10.2.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 10.0 to MariaDB 10.1.

An upgrading guide for unmaintained versions of MariaDB Community Server.

See Upgrading from MariaDB 5.3 to MariaDB 5.5.

An upgrading guide for unmaintained versions of MariaDB Community Server.

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 11.2 to MariaDB 11.3

This page includes details for upgrading from MariaDB 11.2 to MariaDB 11.3. Note that MariaDB 11.2 is a short-term release, only maintained for one year. MariaDB 11.3 is a rolling release, after 11.3.2 one should upgrade to 11.4.2.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 11.3. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`

- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mariadb-upgrade.

- `mariadb-upgrade` does two things:
  1. Ensures that the system tables in the mysql database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 11.2 and 11.3

On most servers upgrading from 11.2 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Changed Default Values

Option	Old default	New default
optimizer_switch		See optimizer-switch.
session_track_system_variables	autocommit, character_set_client, character_set_connection, character_set_results, time_zone	autocommit, character_set_client, character_set_connection, character_set_results, redirect_url, time_zone

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
date_format	Unused.
datetime_format	Unused.
max_tmp_tables	Unused.
time_format	Unused.
wsrep_causal_reads	Deprecated by wsrep_sync_wait=1.

See Also

- Features in MariaDB 11.3
- Features in MariaDB 11.2
- Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster
- Upgrading from MariaDB 11.1 to MariaDB 11.2

- Upgrading from MariaDB 11.0 to MariaDB 11.1

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 11.1 to MariaDB 11.2

This page includes details for upgrading from MariaDB 11.1 to MariaDB 11.2. Note that MariaDB 11.1 and MariaDB 11.2 are both short-term releases, only maintained for one year.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

For MariaDB Galera Cluster, see Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster instead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 11.2. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mariadb-upgrade.

- `mariadb-upgrade` does two things:
  1. Ensures that the system tables in the mysql database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 11.1 and 11.2

On most servers upgrading from 11.1 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Changed Default Values

Option	Old default	New default
optimizer_switch		See optimizer-switch.
innodb_purge_batch_size	300	1000

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
old_alter_table	Superseded by alter_algorithm.

Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

Option	Reason
innodb_purge_rseg_truncate_frequency	The motivation for introducing this in MySQL seems to have been to avoid stalls due to freeing undo log pages or truncating undo log tablespaces. In MariaDB, innodb_undo_log_truncate=ON should be a much lighter operation because it will not involve any log checkpoint, hence this is deprecated and ignored

See Also

- Features in MariaDB 11.2
- Features in MariaDB 11.1
- Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster
- Upgrading from MariaDB 11.0 to MariaDB 11.1
- Upgrading from MariaDB 10.11 to MariaDB 11.0

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 11.0 to MariaDB 11.1

This page includes details for upgrading from MariaDB 11.0 to MariaDB 11.1. Note that MariaDB 11.0 and MariaDB 11.1 are both short-term releases, only maintained for one year.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

For MariaDB Galera Cluster, see Upgrading from MariaDB 10.6 to MariaDB 10.7 with Galera Cluster instead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 11.1. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.

- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mariadb-upgrade.

- `mariadb-upgrade` does two things:
  1. Ensures that the system tables in the mysql database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 11.0 and 11.1

On most servers upgrading from 10.11 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
innodb_defragment	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_fill_factor	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_fill_factor_n_recs	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_frequency	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_n_pages	Defragmenting InnoDB Tablespaces in this manner no longer supported.
innodb_defragment_stats_accuracy	Defragmenting InnoDB Tablespaces in this manner no longer supported.

Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

Option	Reason
innodb_purge_rseg_truncate_frequency	The motivation for introducing this in MySQL seems to have been to avoid stalls due to freeing undo log pages or truncating undo log tablespaces. In MariaDB, innodb_undo_log_truncate=ON should be a much lighter operation because it will not involve any log checkpoint, hence this is deprecated and ignored
tx_isolation	Replaced with transaction_isolation to align the option and system variable.
tx_read_only	Replaced with transaction_read_only to align the option and system variable.

See Also

- Features in MariaDB 11.1
- Features in MariaDB 11.0
- Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster
- Upgrading from MariaDB 10.11 to MariaDB 11.0
- Upgrading from MariaDB 10.6 to MariaDB 10.11

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.11 to MariaDB 11.0

This page includes details for upgrading from MariaDB 10.11 to MariaDB 11.0. It is currently incomplete. Note that MariaDB 10.11 is maintained for five years, while MariaDB 11.0 is a short-term maintenance release, only maintained for one year.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

For MariaDB Galera Cluster, see Upgrading from MariaDB 10.6 to MariaDB 10.7 with Galera Cluster instead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 11.0. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`

- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mariadb-upgrade.

- `mariadb-upgrade` does two things:
  1. Ensures that the system tables in the mysql database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 10.11 and 11.0

On most servers upgrading from 10.11 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Changed Default Values

Option	Old default	New default
innodb_undo_tablespaces	0	3
histogram_type	DOUBLE_PREC_HB	JSON_HB

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
innodb_change_buffer_max_size	InnoDB Change Buffer removed
innodb_change_buffering	InnoDB Change Buffer removed

Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

Option	Reason
innodb_defragment	InnoDB Defragmentation is not particularly useful and causes a maintenance burden.
innodb_defragment_n_pages	
innodb_defragment_stats_accuracy	
innodb_defragment_fill_factor_n_recs	
innodb_defragment_fill_factor	
innodb_defragment_frequency	

Option	Reason
innodb_file_per_table	
innodb_flush_method	
innodb_file_per_table	Has been set for many releases. Unsetting (the original InnoDB default) is no longer useful
innodb_flush_method	Mapped it to 4 new boolean parameters that can be changed while the server is running
log_slow_admin_statements	Use log_slow_filter without admin

See Also

- Features in MariaDB 11.0
- Features in MariaDB 10.11
- Upgrading from MariaDB 10.6 to MariaDB 10.7 with Galera Cluster
- Upgrading from MariaDB 10.6 to MariaDB 10.11
- Upgrading from MariaDB 10.7 to MariaDB 10.8
- Upgrading from MariaDB 10.6 to MariaDB 10.7

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.7 to MariaDB 10.8

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.8. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.

- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mariadb-upgrade.

- `mariadb-upgrade` does two things:
  1. Ensures that the system tables in the mysql database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 10.7 and 10.8

On most servers upgrading from 10.7 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Changed Default Values

Option	Old default value	New default value
innodb_buffer_pool_chunk_size	134217728	Autosized
spider_semi_table_lock	1	0

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
innodb_log_write_ahead_size	On Linux and Windows, the physical block size of the underlying storage is instead detected and used.

Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

Option	Reason
keep_files_on_create	MariaDB now deletes orphan files, so this setting should never be necessary.

Major New Features To Consider

You might consider using the following major new features in MariaDB 10.8:

- Stored procedures already have support for the IN, OUT and INOUT parameter qualifiers. Added as well for stored functions and (IN only) cursors (MDEV-10654).
- Individual columns in the index can now be explicitly sorted in the ascending or descending order. This can be useful for optimizing certain ORDER BY cases (MDEV-13756, MDEV-26938, MDEV-26939, MDEV-26996).
- See also System Variables Added in MariaDB 10.8.

See Also

- The features in MariaDB 10.8
- Upgrading from MariaDB 10.7 to MariaDB 10.7 with Galera Cluster
- Upgrading from MariaDB 10.6 to MariaDB 10.7
- Upgrading from MariaDB 10.5 to MariaDB 10.6
- Upgrading from MariaDB 10.4 to MariaDB 10.5

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.6 to MariaDB 10.7

Note that MariaDB 10.7 is only maintained for one year. MariaDB 10.6 is currently the latest long-term maintenance release.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows.

For MariaDB Galera Cluster, see Upgrading from MariaDB 10.6 to MariaDB 10.7 with Galera Cluster instead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.7. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mariadb-upgrade.

- `mariadb-upgrade` does two things:
  1. Ensures that the system tables in the mysql database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 10.6 and 10.7

On most servers upgrading from 10.6 should be painless. However, there are some things that have changed which could affect an upgrade:

Reserved Words

- ROW_NUMBER is now a reserved word.

Compression

If a non-zlib compression algorithm was used in InnoDB or Mroonga before upgrading to 10.7, those tables will be unreadable until the appropriate compression library is installed. See Compression Plugins#Upgrading.

Options That Have Changed Default Values

Option	Old default	New default
spider_auto_increment_mode	-1	0
spider_bgs_first_read	-1	2
spider_bgs_mode	-1	0
spider_bgs_second_read	-1	100
spider_bka_mode	-1	1
spider_bka_table_name_type	-1	1
spider_buffer_size	-1	16000
spider_bulk_size	-1	16000
spider_bulk_update_mode	-1	0
spider_bulk_update_size	-1	16000
spider_casual_read	-1	0
spider_connect_timeout	-1	6
spider_crd_bg_mode	-1	2
spider_crd_interval	-1	51
spider_crd_mode	-1	1
spider_crd_sync	-1	0
spider_crd_type	-1	2
spider_crd_weight	-1	2
spider_delete_all_rows_type	-1	1
spider_direct_dup_insert	-1	0
spider_direct_order_limit	-1	9223372036854775807
spider_error_read_mode	-1	0
spider_error_write_mode	-1	0
spider_first_read	-1	0
spider_init_sql_alloc_size	-1	1024
spider_internal_limit	-1	9223372036854775807
spider_internal_offset	-1	0
spider_internal_optimize	-1	0
spider_internal_optimize_local	-1	0

Option	Old default	New default
spider_load_crd_at_startup	-1	1
spider_load_sts_at_startup	-1	1
spider_low_mem_read	-1	1
spider_max_order	-1	32767
spider_multi_split_read	-1	100
spider_net_read_timeout	-1	600
spider_net_write_timeout	-1	600
spider_quick_mode	-1	3
spider_quick_page_byte	-1	10485760
spider_quick_page_size	-1	1024
spider_read_only_mode	-1	0
spider_reset_sql_alloc	-1	1
spider_second_read	-1	0
spider_selupd_lock_mode	-1	1
spider_semi_split_read	-1	2
spider_semi_split_read_limit	-1	1
spider_semi_table_lock_connection	-1	1
spider_reset_sql_alloc	-1	1

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
wsrep_replicate_myisam	Use wsrep_mode instead.
wsrep_strict_ddl	Use wsrep_mode instead.

See Also

- The features in MariaDB 10.7
- Upgrading from MariaDB 10.6 to MariaDB 10.11 with Galera Cluster
- Upgrading from MariaDB 10.5 to MariaDB 10.6
- Upgrading from MariaDB 10.4 to MariaDB 10.5
- Upgrading from MariaDB 10.3 to MariaDB 10.4

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.4 to MariaDB 10.5

How to Upgrade

For Windows, see Upgrading MariaDB on Windows instead.

For MariaDB Galera Cluster, see Upgrading from MariaDB 10.4 to MariaDB 10.5 with Galera Cluster.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.5. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mysql_upgrade.

- `mysql_upgrade` does two things:
  1. Ensures that the system tables in the mysql database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 10.4 and 10.5

On most servers upgrading from 10.4 should be painless. However, there are some things that have changed which could affect an upgrade:

Binary name changes

All binaries previously beginning with mysql now begin with mariadb, with symlinks for the corresponding mysql command.

Usually that shouldn't cause any changed behavior, but when starting the MariaDB server via systemd, or via the mysqld_safe script symlink, the server process will now always be started as `mariadbd`, not `mysqld`.

So anything looking for the `mysqld` name in the system process list, like e.g. monitoring solutions, now needs for `mariadbd` instead when the server / service is not started directly, but via `mysqld_safe` or as a system service.

GRANT PRIVILEGE changes

A number of statements changed the privileges that they require. The old privileges were historically inappropriately chosen in the upstream. 10.5.2 fixes this problem. Note, these changes are incompatible to previous versions. A number of GRANT commands might be needed after upgrade.

- `SHOW BINLOG EVENTS` now requires the `BINLOG MONITOR` privilege (required `REPLICATION SLAVE` prior to 10.5.2).
- `SHOW SLAVE HOSTS` now requires the `REPLICATION MASTER ADMIN` privilege (required `REPLICATION SLAVE` prior to 10.5.2).
- `SHOW SLAVE STATUS` now requires the `REPLICATION SLAVE ADMIN` or the `SUPER` privilege (required `REPLICATION CLIENT` or `SUPER` prior to 10.5.2).
- `SHOW RELAYLOG EVENTS` now requires the `REPLICATION SLAVE ADMIN` privilege (required `REPLICATION SLAVE` prior to 10.5.2).

Options That Have Changed Default Values

Option	Old default value	New default value
innodb_adaptive_hash_index	ON	OFF
innodb_checksum_algorithm	crc32	full_crc32
innodb_log_optimize_ddl	ON	OFF
slave_parallel_mode	conservative	optimistic
performance_schema_max_cond_classes	80	90
performance_schema_max_file_classes	50	80
performance_schema_max_mutex_classes	200	210
performance_schema_max_rwlock_classes	40	50
performance_schema_setup_actors_size	100	-1
performance_schema_setup_objects_size	100	-1

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
innodb_checksums	Deprecated and functionality replaced by innodb_checksum_algorithms in MariaDB 10.0.
innodb_idle_flush_pct	Has had no effect since merging InnoDB 5.7 from mysql-5.7.9 (MariaDB 10.2.2).
innodb_locks_unsafe_for_binlog	Deprecated in MariaDB 10.0. Use READ COMMITTED transaction isolation level instead.
innodb_rollback_segments	Deprecated and replaced by innodb_undo_logs in MariaDB 10.0.
innodb_stats_sample_pages	Deprecated in MariaDB 10.0. Use innodb_stats_transient_sample_pages instead.
max_long_data_size	Deprecated and replaced by max_allowed_packet in MariaDB 5.5.
multi_range_count	Deprecated and has had no effect since MariaDB 5.3.
thread_concurrency	Deprecated and has had no effect since MariaDB 5.5.
timed_mutexes	Deprecated and has had no effect since MariaDB 5.5.

Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

Option	Reason
innodb_adaptive_max_sleep_delay	No need for thread throttling any more.
innodb_background_scrub_data_check_interval	Problematic 'background scrubbing' code removed.
innodb_background_scrub_data_interval	Problematic 'background scrubbing' code removed.
innodb_background_scrub_data_compressed	Problematic 'background scrubbing' code removed.
innodb_background_scrub_data_uncompressed	Problematic 'background scrubbing' code removed.
innodb_buffer_pool_instances	Having more than one buffer pool is no longer necessary.
innodb_commit_concurrency	No need for thread throttling any more.
innodb_concurrency_tickets	No need for thread throttling any more.
innodb_log_files_in_group	Redo log was unnecessarily split into multiple files. Limited to 1 from MariaDB 10.5.
innodb_log_optimize_ddl	Prohibited optimizations.
innodb_page_cleaners	Having more than one page cleaner task no longer necessary.
innodb_replication_delay	No need for thread throttling any more.
innodb_scrub_log	Never really worked as intended, redo log format is being redone.
innodb_scrub_log_speed	Never really worked as intended, redo log format is being redone.
innodb_thread_concurrency	No need for thread throttling any more.
innodb_thread_sleep_delay	No need for thread throttling any more.
innodb_undo_logs	It always makes sense to use the maximum number of rollback segments.
large_page_size	Unused since multiple page size support was added.

Major New Features To Consider

You might consider using the following major new features in MariaDB 10.5:

- The S3 storage engine allows one to archive MariaDB tables in Amazon S3, or any third-party public or private cloud that implements S3 API.
- ColumnStore columnar storage engine.
- See also System Variables Added in MariaDB 10.5.

See Also

- The features in MariaDB 10.5
- Upgrading from MariaDB 10.4 to MariaDB 10.5 with Galera Cluster
- Upgrading from MariaDB 10.3 to MariaDB 10.4
- Upgrading from MariaDB 10.2 to MariaDB 10.3
- Upgrading from MariaDB 10.1 to MariaDB 10.2

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.3 to MariaDB 10.4

How to Upgrade

For Windows, see Upgrading MariaDB on Windows instead.

For MariaDB Galera Cluster, see Upgrading from MariaDB 10.3 to MariaDB 10.4 with Galera Cluster instead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.4. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mysql_upgrade.

- `mysql_upgrade` does two things:
  1. Ensures that the system tables in the mysql database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 10.3 and 10.4

On most servers upgrading from 10.3 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Changed Default Values

Option	Old default value	New default value
slave_transaction_retry_errors	1213,1205	1158,1159,1160,1161,1205,1213,1429,2013,12701
wsrep_debug	OFF	NONE
wsrep_load_data_splitting	ON	OFF

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason

Authentication and TLS

- See Authentication from MariaDB 10.4 for an overview of the changes.
- The unix_socket authentication plugin is now default on Unix-like systems.
- TLSv1.0 is disabled by default in MariaDB 10.4. See tls_version and TLS Protocol Versions.

Major New Features To Consider

You might consider using the following major new features in MariaDB 10.4:

- Galera has been upgraded from Galera 3 to Galera 4.
- System-versioning extended with support for application-time periods.
- User password expiry
- Account Locking
- See also System Variables Added in MariaDB 10.4.

See Also

- The features in MariaDB 10.4
- Upgrading from MariaDB 10.3 to MariaDB 10.4 with Galera Cluster
- Upgrading from MariaDB 10.2 to MariaDB 10.3
- Upgrading from MariaDB 10.1 to MariaDB 10.2
- Upgrading from MariaDB 10.0 to MariaDB 10.1

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.2 to MariaDB 10.3

How to Upgrade

For Windows, see Upgrading MariaDB on Windows instead.

For MariaDB Galera Cluster, see Upgrading with Galera Cluster instead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.3. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Stop MariaDB. The server should be cleanly shut down, with no incomplete transactions remaining. innodb_fast_shutdown must be set to `0` or `1` and innodb_force_recovery must be less than `3`.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`

- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mysql_upgrade.

- `mysql_upgrade` does two things:
  1. Ensures that the system tables in the `[mysq](#pg-reference-system-tables-the-mysql-database-tables-readme-35e9dbf3) l` database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 10.2 and 10.3

On most servers upgrading from 10.2 should be painless. However, there are some things that have changed which could affect an upgrade:

Options That Have Changed Default Values

Option	Old default value	New default value
innodb_flush_method	(empty)	fsync
innodb_spin_wait_delay	6	4
performance_schema_max_stage_classes	150	160
plugin_maturity	unknown	One less than the server maturity

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
innodb_buffer_pool_populate	Used in XtraDB-only
innodb_cleaner_lsn_age_factor	Used in XtraDB-only
innodb_corrupt_table_action	Used in XtraDB-only
innodb_empty_free_list_algorithm	Used in XtraDB-only
innodb_fake_changes	Used in XtraDB-only
innodb_file_format	The InnoDB file format is now Barracuda, and the old Antelope file format is no longer supported.
innodb_file_format_check	No longer necessary as the Antelope InnoDB file format is no longer supported.
innodb_file_format_max	No longer necessary as the Antelope InnoDB file format is no longer supported.

Option	Reason
innodb_foreground_preflush	Used in XtraDB-only
innodb_instrument_semaphores	
innodb_kill_idle_transaction	Used in XtraDB-only
innodb_large_prefix	Large index key prefixes were made default from MariaDB 10.2, and limiting tables to small prefixes is no longer permitted in MariaDB 10.3.
innodb_locking_fake_changes	Used in XtraDB-only
innodb_log_arch_dir	Used in XtraDB-only
innodb_log_arch_expire_sec	Used in XtraDB-only
innodb_log_archive	Used in XtraDB-only
innodb_log_block_size	Used in XtraDB-only
innodb_log_checksum_algorithm	Translated to innodb_log_checksums (NONE to OFF, everything else to ON); only existed to allow easier upgrade from earlier XtraDB versions.
innodb_mtflush_threads	Replaced by the innodb_page_cleaners system variable.
innodb_sched_priority_cleaner	Used in XtraDB-only
innodb_show_locks_held	Used in XtraDB-only
innodb_show_verbose_locks	Used in XtraDB-only
innodb_support_xa	XA transactions are always supported.
innodb_use_fallocate	
innodb_use_global_flush_log_at_trx_commit	Used in XtraDB-only
innodb_use_mtflush	Replaced by the innodb_page_cleaners system variable.
innodb_use_stacktrace	Used in XtraDB-only
innodb_use_trim	

Reserved Words

- New reserved words: EXCEPT and INTERSECT. These can no longer be used as identifiers without being quoted.

SQL_MODE=ORACLE

- MariaDB 10.3 has introduced major new Oracle compatibility features. If you upgrade and are using this setting, please check the changes carefully.

Functions

- As a result of implementing Table Value Constructors, the VALUES function has been renamed to VALUE().
- Functions that used to only return 64-bit now can return 32-bit results (MDEV-12619). This could cause incompatibilities with strongly-typed clients.

mysqldump

- mysqldump in MariaDB 10.3 includes logic to cater for the mysql.transaction_registry table. `mysqldump` from an earlier MariaDB release cannot be used on MariaDB 10.3 and beyond.

MariaDB Backup and Percona XtraBackup

- Percona XtraBackup is not compatible with MariaDB 10.3. Installations currently using XtraBackup should upgrade to MariaDB Backup before upgrading to MariaDB 10.3.

Privileges

- If a user has the SUPER privilege but not the `DELETE HISTORY` privilege, running mysql_upgrade will grant `DELETE HISTORY` as well.

Major New Features To Consider

You might consider using the following major new features in MariaDB 10.3:

- System-versioned tables
- Sequences
- See also System Variables Added in MariaDB 10.3.

See Also

- The features in MariaDB 10.3
- Upgrading from MariaDB 10.2 to MariaDB 10.3 with Galera Cluster
- Upgrading from MariaDB 10.1 to MariaDB 10.2
- Upgrading from MariaDB 10.0 to MariaDB 10.1

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.1 to MariaDB 10.2

How to Upgrade

For Windows, see Upgrading MariaDB on Windows instead.

For MariaDB Galera Cluster, see Upgrading with Galera Cluster instead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend mariadb-backup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.2. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Set innodb_fast_shutdown to `0`. It can be changed dynamically with SET GLOBAL. For example:`SET GLOBAL innodb_fast_shutdown=0;`

- This step is not necessary when upgrading to MariaDB 10.2.5 or later. Omitting it can make the upgrade process far faster. See MDEV-12289 for more information.

1. Stop MariaDB.
2. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mysql_upgrade.

- `mysql_upgrade` does two things:
  1. Ensures that the system tables in the `[mysq](#pg-reference-system-tables-the-mysql-database-tables-readme-35e9dbf3) l` database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 10.1 and 10.2

On most servers upgrading from 10.1 should be painless. However, there are some things that have changed which could affect an upgrade:

InnoDB Instead of XtraDB

MariaDB 10.2 uses InnoDB as the default storage engine, rather than XtraDB, used in MariaDB 10.1 and before. See Why does MariaDB 10.2 use InnoDB instead of XtraDB? In most cases this should have minimal effect as the latest InnoDB has incorporated most of the improvements made in earlier versions of XtraDB. Note that certain XtraDB system variables are now ignored (although they still exist so as to permit easy upgrading).

Options That Have Changed Default Values

In particular, take note of the changes to innodb_strict_mode, sql_mode, binlog_format, binlog_checksum and innodb_checksum_algorithm.

Option	Old default value	New default value
aria_recover(_options)	NORMAL	BACKUP, QUICK
binlog_annotate_row_events	OFF	ON
binlog_checksum	NONE	CRC32
binlog_format	STATEMENT	MIXED
group_concat_max_len	1024	1048576
innodb_autoinc_lock_mode	1	2

Option	Old default value	New default value
innodb_buffer_pool_dump_at_shutdown	OFF	ON
innodb_buffer_pool_dump_pct	100	25
innodb_buffer_pool_instances	8	Varies
innodb_buffer_pool_load_at_startup	OFF	ON
innodb_checksum_algorithm	innodb	crc32
innodb_file_format	Antelope	Barracuda
innodb_large_prefix	OFF	ON
innodb_lock_schedule_algorithm	VATS	FCFS
innodb_log_compressed_pages	OFF	ON
innodb_max_dirty_pages_pct_lwm	0.001000	0
innodb_max_undo_log_size	1073741824	10485760
innodb_purge_threads	1	4
innodb_strict_mode	OFF	ON
innodb_undo_directory	.	NULL
innodb_use_atomic_writes	OFF	ON
innodb_use_trim	OFF	ON
lock_wait_timeout	31536000	86400
log_slow_admin_statements	OFF	ON
log_slow_slave_statements	OFF	ON
log_warnings	1	2
max_allowed_packet	4M	16M
max_long_data_size	4M	16M
myisam_recover_options	NORMAL	BACKUP, QUICK
optimizer_switch	See Optimizer Switch for details.	
replicate_annotate_row_events	OFF	ON
server_id	0	1
slave_net_timeout	3600	60
sql_mode	NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION	STRICT_TRANS_TABLES, ERROR_FOR_DIVISION_BY_ZERO, NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION
thread_cache_size	0	Auto

Option	Old default value	New default value
thread_pool_max_threads	1000	65536
thread_stack	295936	299008

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files:

Option	Reason
aria_recover	Renamed to aria_recover_options to match myisam_recover_options.
innodb_additional_mem_pool_size	Deprecated in MariaDB 10.0.
innodb_api_bk_commit_interval	Memcache never implemented in MariaDB.
innodb_api_disable_rowlock	Memcache never implemented in MariaDB.
innodb_api_enable_binlog	Memcache never implemented in MariaDB.
innodb_api_enable_mdl	Memcache never implemented in MariaDB.
[	innodb_api_trx_level](../../../../reference/storage-engines/innodb/innodb-system-variables.md)
innodb_use_sys_malloc	Deprecated in MariaDB 10.0.

Reserved Words

New reserved words: OVER, RECURSIVE and ROWS. These can no longer be used as identifiers without being quoted.

TokuDB

TokuDB has been split into a separate package, mariadb-plugin-tokudb.

Replication

Replication from legacy MySQL servers may require setting binlog_checksum to NONE.

SQL Mode

SQL_MODE has been changed; in particular, NOT NULL fields with no default will no longer fall back to a dummy value for inserts which do not specify a value for that field.

Auto_increment

Auto_increment columns are no longer permitted in CHECK constraints, DEFAULT value expressions and virtual columns. They were permitted in earlier versions, but did not work correctly.

TLS

Starting with MariaDB 10.2, when the user specifies the `--ssl` option with a client or utility, the client or utility will not verify the server certificate by default. In order to verify the server certificate, the user must specify the `--ssl-verify-server-cert` option to the client or utility. For more information, see the list of options for the mysql client.

Major New Features To Consider

You might consider using the following major new features in MariaDB 10.2:

- Window Functions
- mysqlbinlog now supports continuous binary log backups

- Recursive Common Table Expressions
- JSON functions
- See also System Variables Added in MariaDB 10.2.

See Also

- The features in MariaDB 10.2
- Upgrading from MariaDB 10.1 to MariaDB 10.2 with Galera Cluster
- Upgrading from MariaDB 10.0 to MariaDB 10.1
- Upgrading from MariaDB 5.5 to MariaDB 10.0

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 10.0 to MariaDB 10.1

What You Need to Know

There are no changes in table or index formats between MariaDB 10.0 and MariaDB 10.1, so on most servers the upgrade should be painless.

How to Upgrade

For Windows, see Upgrading MariaDB on Windows instead.

For MariaDB Galera Cluster, see Upgrading from MariaDB Galera Clusterinstead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend Percona XtraBackup.

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.1. For example,

- On Debian, Ubuntu, and other similar Linux distributions, see Updating the MariaDB APT repository to a New Major Release for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Updating the MariaDB YUM repository to a New Major Release for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Updating the MariaDB ZYpp repository to a New Major Release for more information.

1. Set innodb_fast_shutdown to `0`. It can be changed dynamically with SET GLOBAL. For example:`SET GLOBAL innodb_fast_shutdown=0;`
2. Stop MariaDB.
3. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following: `sudo apt-get remove mariadb-server`
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following: `sudo yum remove MariaDB-server`
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following: `sudo zypper remove MariaDB-server`

1. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see Installing MariaDB Packages with APT for more information.

- On RHEL, CentOS, Fedora, and other similar Linux distributions, see Installing MariaDB Packages with YUM for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see Installing MariaDB Packages with ZYpp for more information.

1. Make any desired changes to configuration options in option files, such as `my.cnf`. This includes removing any options that are no longer supported.
2. Start MariaDB.
3. Run mysql_upgrade.

- `mysql_upgrade` does two things:
  1. Ensures that the system tables in the `[mysq](#pg-reference-system-tables-the-mysql-database-tables-readme-35e9dbf3) l` database are fully compatible with the new version.
  2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

Incompatible Changes Between 10.0 and 10.1

As mentioned previously, on most servers upgrading from 10.0 should be painless.
However, there are some things that have changed which could affect an upgrade:

Storage Engines

- The ARCHIVE storage engine is no longer enabled by default, and the plugin needs to be specifically enabled.
- The BLACKHOLE storage engine is no longer enabled by default, and the plugin needs to be specifically enabled.

Replication

- MariaDB 10.1 introduces new, standards-compliant behavior for dealing with primary keys over nullable columns. In certain edge cases this could cause replication issues when replicating from a MariaDB 10.0 master to a MariaDB 10.1 slave using statement-based replication. See MDEV-12248.

Options That Have Changed Default Values

Most of the following options have increased in value to give better performance.

Option	Old default value	New default value
innodb_log_compressed_pages	ON	OFF
join_buffer_size	128K	256K
max_allowed_packet	1M	4M
query_alloc_block_size	8192	16384
query_cache_size	0	1M
query_cache_type	ON	OFF
sync_master_info	0	10000
sync_relay_log	0	10000
sync_relay_log_info	0	10000
query_prealloc_size	8192	24576
secure_auth	OFF	ON
sql_log_bin		No longer affects replication of events in a Galera cluster.
sql_mode	empty	NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION
table_open_cache	400	2000

Option	Old default value	New default value
thread_pool_max_threads	500	1000

Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your config files:

Option	Reason
rpl_recovery_rank	Unused in 10.0

Other Issues

Note that explicit or implicit casts from MAX(string) to INT, DOUBLE or DECIMAL now produce warnings (MDEV-8852).

Major New Features To Consider

You might consider using the following major new features in MariaDB 10.1:

- Galera Cluster is now included by default.
- Encryption
- InnoDB/XtraDB Page Compression

Notes

See Also

- The features in MariaDB 10.1
- Upgrading from MariaDB 10.1 to MariaDB 10.2
- Upgrading from MariaDB 5.5 to MariaDB 10.0

This page is licensed: CC BY-SA / Gnu FDL

Upgrading from MariaDB 5.3 to MariaDB 5.5

What you need to know

There are no changes in table or index formats between MariaDB 5.3 and MariaDB 5.5, so on most servers the upgrade should be painless.

How to upgrade

The suggested upgrade procedure is:

1. For Windows, see Upgrading MariaDB on Windows instead.
2. Shutdown MariaDB 5.3
3. Take a backup (this is the perfect time to take a backup of your databases)
4. Uninstall MariaDB 5.3
5. Install MariaDB 5.5 [1]
6. Run mysql_upgrade

- Ubuntu and Debian packages do this automatically when they are installed; Red Hat, CentOS, and Fedora packages do not
- `mysql_upgrade` does two things:
  1. Upgrades the permission tables in the `mysql` database with some new fields

2. Does a very quick check of all tables and marks them as compatible with MariaDB 5.5

- In most cases this should be a fast operation (depending of course on the number of tables)

1. Add new options to my.cnf to enable features

- If you change `my.cnf` then you need to restart `mysqld`

Incompatible changes between 5.3 and 5.5

As mentioned previously, on most servers upgrading from 5.5 should be painless.
However, there are some things that have changed which could affect an upgrade:

XtraDB options that have changed default values

Option	Old value	New value
innodb_change_buffering	inserts	all
innodb_flush_neighbor_pages	1	area

Options that have been removed or renamed

Percona, the provider of XtraDB, does not provide all earlier XtraDB features in the 5.5 code base. Because of that, MariaDB 5.5 can't provide them either. The following options are not supported by XtraDB 5.5. If you are using them in any of your my.cnf files, you should remove them before upgrading to 5.5.

- innodb_adaptive_checkpoint; Use innodb_adaptive_flushing_method instead.
- innodb_auto_lru_dump; Use innodb_buffer_pool_restore_at_startup instead (and innodb_buffer_pool_load_at_startup in MariaDB 10.0).
- innodb_blocking_lru_restore; Use innodb_blocking_buffer_pool_restore instead.
- innodb_enable_unsafe_group_commit
- innodb_expand_import; Use innodb_import_table_from_xtrabackup instead.
- innodb_extra_rsegments; Use innodb_rollback_segments instead.
- innodb_extra_undoslots
- innodb_fast_recovery
- innodb_flush_log_at_trx_commit_session
- innodb_overwrite_relay_log_info; Use innodb_recovery_update_relay_log instead.
- innodb_pass_corrupt_table; Use innodb_corrupt_table_action instead.
- innodb_use_purge_thread

Notes

1. ↑ If using a MariaDB `apt` or `yum` repository, it is often enough to replace instances of '5.3' with '5.5' and then run an update/upgrade. For example, in Ubuntu/Debian update the MariaDB `sources.list` entry from something that looks similar to this:

```
deb http://ftp.osuosl.org/pub/mariadb/repo/5.3/ubuntu trusty main
```

To something like this:

```
deb http://ftp.osuosl.org/pub/mariadb/repo/5.5/ubuntu trusty main
```

And then run

```
apt-get update && apt-get upgrade
```

And in Red Hat, CentOS, and Fedora, change the `baseurl` line from something that looks like this:

```
baseurl = http://yum.mariadb.org/5.3/centos6-amd64
```

To something that looks like this:

```
baseurl = http://yum.mariadb.org/5.5/centos6-amd64
```

And then run

```
yum update
```

See also

- The features in MariaDB 5.5
- Perconas guide of how to upgrade to 5.5

This page is licensed: CC BY-SA / Gnu FDL

General Upgrade Information

While specific guides exist for upgrading between individual versions, it is important to understand the general policies regarding skipping versions and upgrading clusters.

Direct Upgrade Policy

Historically, upgrading MariaDB Community Server required stepping through each major version individually. However, for modern versions, **you can upgrade directly from your current version to the latest target version** in most scenarios.

- **Skip Intermediate Versions:** Skipping intermediate major or LTS versions is fully supported and tested for standalone servers.
- **Check Incompatible Changes:** Even when skipping versions, you **must** review the “Incompatible Changes” section for *every* major version between your current version and your target version. This ensures your application does not rely on functionality, configuration, or reserved keywords that have been deprecated or modified in the interim.
- **Troubleshooting:** If you are unsure about your application’s compatibility, upgrading one major/LTS version at a time is recommended. This incremental approach makes it easier to isolate the specific version that introduces a breaking change or application issue.

Galera Cluster Upgrades

The upgrade strategy for Galera Clusters differs depending on whether you require zero downtime.

Rolling Upgrades

If you are performing a rolling upgrade to keep the cluster online:

- You **must upgrade one major/LTS version at a time**.
- Due to the complexity of intra-node communication, skipping versions during a rolling upgrade is not supported.
- This process requires multiple upgrade cycles, but the cluster remains available to applications throughout.

Full Cluster Shutdown

If you can schedule downtime for the entire cluster:

- You may shut down all nodes, upgrade them all to the target version, and then bring the cluster back up.

- In this scenario, a **direct upgrade** (skipping intermediate versions) is supported, similar to a standalone server.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Enterprise Server Upgrade Paths

See MariaDB Enterprise Server 11.8.

Upgrade guide for MariaDB Enterprise Server 11.8, highlighting significant performance improvements in transactional throughput, new vector search optimizations, and enhanced observability features.

See MariaDB Enterprise Server 11.4.

Instructions for upgrading to MariaDB Enterprise Server 11.4, which introduces new data types like UUID and INET4, advanced JSON functions, and non-blocking online schema changes.

See MariaDB Enterprise Server 10.6.

Upgrade documentation for MariaDB Enterprise Server 10.6, featuring Atomic DDL support, JSON_TABLE function, improved Oracle compatibility modes, and the removal of older storage engines.

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Server 11.8

See Upgrade to MariaDB Enterprise Server 11.8.

Upgrade guide for MariaDB Enterprise Server 11.8, highlighting significant performance improvements in transactional throughput, new vector search optimizations, and enhanced observability features.

See Upgrade MariaDB Enterprise Server from 11.8.X to 11.8.Y.

Instructions for performing a minor release upgrade of MariaDB Enterprise Server within the 11.8 series, from an earlier 11.8.x release to a later 11.8.y release.

See Upgrading from MariaDB Enterprise Server 10.6 to 11.8.

Perform a major-version upgrade of MariaDB Enterprise Server directly from 10.6 to 11.8 on systemd-based Linux, including prerequisites, backup, and the upgrade procedure.

See Upgrade from MariaDB Community Server 10.6 to MariaDB Enterprise Server 11.8.

Upgrade directly from MariaDB Community Server 10.6 to MariaDB Enterprise Server 11.8 — combining a product switch (Community to Enterprise) and a major version jump (10.6 to 11.8) in a single upgrade

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade to MariaDB Enterprise Server 11.8

Overview

These instructions detail the **upgrade** from a previous version of **MariaDB Enterprise Server** to **MariaDB Enterprise Server 11.8** on a range of supported Operating Systems.

When MariaDB Enterprise Server is upgraded, the old version needs to be uninstalled, and the new version needs to be installed.

See What's New in MariaDB Enterprise Server 11.8.

Data Backup

Occasionally, issues can be encountered during upgrades. These issues can even potentially corrupt the database's data files, preventing you from easily reverting to the old installation. Therefore, it is generally best to perform a backup before upgrading. If an issue is encountered during the upgrade, you can use the backup to restore your MariaDB Server database to the old version. If the upgrade finishes without issue, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, please see the Recovery Guide.

1. Take a full backup. On MariaDB Enterprise Server 11.8 and later:

```
sudo mariadb-backup --backup \
      --user=mariadb-backup_user \
      --password=mariadb-backup_passwd \
      --target-dir=/data/backup/preupgrade_backup
```

Confirm successful completion of the backup operation.

2. The backup must be prepared. On the MariaDB Enterprise Server 10.4 and later:

```
sudo mariadb-backup --prepare \
      --target-dir=/data/backup/preupgrade_backup
```

Confirm successful completion of the prepare operation.

3. Backups should be tested before they are trusted.

Audit Plugin Considerations

If you have the MariaDB Audit Plugin installed and are upgrading from MariaDB Enterprise Server 10.2 or 10.3, the audit plugin should be removed prior to the upgrade to prevent conflicts with the MariaDB Enterprise Audit Plugin, which MariaDB Enterprise Audit Plugin that is present in MariaDB Enterprise Server 10.4 or later. It can be removed by using the UNINSTALL SONAME statement:

```
UNINSTALL SONAME 'server_audit';
```

And if you load the plugin in a configuration file using the `plugin\_load\_add` option, then the option should also be removed. The MariaDB Enterprise Audit Plugin will automatically be installed after installing MariaDB Enterprise Server 10.4 or later.

Uninstall the Old Version

When upgrading to a new major release of MariaDB Enterprise Server, it is necessary to remove the existing installation of MariaDB Enterprise Server before installing the new version of MariaDB Enterprise Server. Otherwise, the package manager will refuse to install the new version of MariaDB Enterprise Server.

Stop the MariaDB Server Process

Before the old version can be uninstalled, we first need to stop the current MariaDB Server process.

1. Set the innodb_fast_shutdown system variable to 1:

```
SET GLOBAL innodb_fast_shutdown = 1;
```

2. Use XA RECOVER to confirm that there are no external XA transactions in a prepared state:

```
XA RECOVER;
```

Commit or roll back any open XA transactions before stopping the node for upgrade.

3. Stop the server process: For distributions that use `systemd` (most supported OSes), you can manage the server process using the `systemctl` command:

```
sudo systemctl stop mariadb
```

Uninstall via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Uninstall all of the MariaDB Enterprise Server packages. Note that a wildcard character is used to ensure that all MariaDB Enterprise Server packages are uninstalled:

```
sudo yum remove "MariaDB-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications:

2. Uninstall the Galera package as well. The name of the package depends on the specific version of MariaDB Enterprise Server. When upgrading from MariaDB Enterprise Server 10.4 or later, the package is called `galera-enterprise-4`:

```
sudo yum remove galera-enterprise-4
```

3. Before proceeding, verify that all MariaDB Enterprise Server packages are uninstalled. The following command should not return any results:

```
rpm --query --all | grep -i -E "mariadb|galera"
```

Uninstall via APT (Debian, Ubuntu)

1. Uninstall all of the MariaDB Enterprise Server packages. Note that a wildcard character is used to ensure that all MariaDB Enterprise Server packages are uninstalled:

```
sudo apt-get remove "mariadb-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well. The name of the package depends on the specific version of MariaDB Enterprise Server. When upgrading from MariaDB Enterprise Server 10.4 or later, the package is called galera-enterprise-4:

```
sudo apt remove galera-enterprise-4
```

3. Before proceeding, verify that all MariaDB Enterprise Server packages are uninstalled. The following command should not return any results:

```
apt list --installed | grep -i -E "mariadb|galera"
```

Uninstall via ZYpp (SLES)

1. Uninstall all of the MariaDB Enterprise Server packages. Note that a wildcard character is used to ensure that all MariaDB Enterprise Server packages are uninstalled:

```
sudo zypper remove "MariaDB-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well. The name of the package depends on the specific version of MariaDB Enterprise Server. When upgrading from MariaDB Enterprise Server 10.4 or later, the package is called galera-enterprise-4:

```
sudo zypper remove galera-enterprise-4
```

3. Before proceeding, verify that all MariaDB Enterprise Server packages are uninstalled. The following command should not return any results:

```
rpm --query --all | grep -i -E "mariadb|galera"
```

Install the New Version

MariaDB Corporation provides package repositories for YUM (RHEL, AlmaLinux, CentOS, Rocky Linux), APT (Debian, Ubuntu), and ZYpp (SLES).

Install via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Retrieve your Customer Download Token atand substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the YUM package repository. Installable versions of MariaDB Enterprise Server are `11.8, 11.4, 10.6, 10.5, 10.4, and 10.3.` Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

To configure YUM package repositories:

```
sudo yum install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

> **Note:**
>
> *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.8"
```

3. Install MariaDB Enterprise Server and package dependencies:

```
sudo yum install MariaDB-server MariaDB-backup
```

4. Installation of additional packages may be required for some plugins.
5. Configure MariaDB. Installation only loads MariaDB Enterprise Server onto the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Install via APT (Debian, Ubuntu)

1. Retrieve your Customer Download Token atand substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the APT package repository. Installable versions of MariaDB Enterprise Server are `11.8 11.4, 10.6, 10.5, 10.4, and 10.3.` Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

To configure APT package repositories:

```
sudo apt install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

> **Note:**
>
> *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.8"
```

```
sudo apt update
```

3. Install MariaDB Enterprise Server and package dependencies:

```
$ sudo apt install mariadb-server mariadb-backup galera-enterprise-4
```

4. Installation of additional packages may be required for some plugins. Configure MariaDB. Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Install via ZYpp (SLES)

1. Retrieve your Customer Download Token atand substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the ZYpp package repository. Installable versions of MariaDB Enterprise Server are `11.8, 11.4, 10.6, 10.5, 10.4, and 10.3.` Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

To configure ZYpp package repositories:

```
sudo zypper install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

> **Note:**
>
> *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.8"
```

3. Install MariaDB Enterprise Server and package dependencies:

```
sudo zypper install MariaDB-server MariaDB-backup
```

4. Installation of additional packages may be required for some plugins.
5. Configure MariaDB. Installation only loads MariaDB Enterprise Server onto the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Configuration

For platforms that use YUM or ZYpp as a package manager: MariaDB Enterprise Server's packages bundle several configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mariadb-enterprise.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

If your version of any of these configuration files contained any custom edits, then the package manager may save your edited version with the `.rpmsave` extension during the upgrade process. If you want to continue using your version with the custom edits, then you may need to move it back.

For example, to move `server.cnf` back in place:

```
sudo mv /etc/my.cnf.d/server.cnf /etc/my.cnf.d/server.cnf.original
```

```
sudo mv /etc/my.cnf.d/server.cnf.rpmsave /etc/my.cnf.d/server.cnf
```

Starting the Server

MariaDB Enterprise Server includes configuration to `start, stop, restart, enable/disable` on boot, and check the status of the Server using the operating system's default process management system.

For distributions that use `systemd`, you can manage the Server process using the `systemctl` command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`

Operation	Command
Status	`sudo systemctl status mariadb`

Upgrading the Data Directory

MariaDB Enterprise Server ships with a utility that can be used to identify and correct compatibility issues in the new version. After you upgrade your Server and start the server process, run this utility to upgrade the data directory.

The utility is called `mariadb-upgrade` in MariaDB Enterprise Server 10.4 and later:

```
sudo mariadb-upgrade
```

Testing

When MariaDB Enterprise Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the server using MariaDB Client using the `root@localhost` user account. MariaDB Client is called `mariadb` (ES 10.4 and later):

```
sudo mariadb
```

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 11.8.2-0-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

2. You can also verify the server version by checking the value of the version system variable with the SHOW GLOBAL STATUS statement:

```
SHOW GLOBAL VARIABLES LIKE 'version';
```

```
+---------------+----------------------------+
| Variable_name | Value                      |
+---------------+----------------------------+
| version       | 11.8.3-1-MariaDB-Enterprise |
+---------------+----------------------------+
```

3. You can also verify the server version by calling the VERSION() function:

```
SELECT VERSION();
```

```
+----------------------------+
| VERSION()                  |
+----------------------------+
| 11.8.3-1-MariaDB-Enterprise |
+----------------------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade MariaDB Enterprise Server from 11.8.X to 11.8.Y

These instructions detail a **minor release upgrade** with **MariaDB Enterprise Server 11.8** on a range of supported Operating Systems.

A minor release upgrade is a change from an earlier release of MariaDB Enterprise Server 11.8 to a later release in the same release series.

For example, it would be a minor release upgrade to upgrade from MariaDB Enterprise Server 11.8.2-0 to MariaDB Enterprise Server 11.8.3-1.

Data Backup

Occasionally, issues can be encountered during upgrades. These issues can even potentially corrupt the database's data files, preventing you from easily reverting to the old installation. Therefore, it is generally best to perform a backup before upgrading. If an issue is encountered during the upgrade, you can use the backup to restore your MariaDB Server database to the old version. If the upgrade finishes without issue, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, please see the Recovery Guide.

1. Take a full backup:

```
sudo mariadb-backup --backup \
      --user=mariadb-backup_user \
      --password=mariadb-backup_passwd \
      --target-dir=/data/backup/preupgrade_backup
```

Confirm successful completion of the backup operation.

2. The backup must be prepared:

```
sudo mariadb-backup --prepare \
      --target-dir=/data/backup/preupgrade_backup
```

Confirm successful completion of the prepare operation.

3. Backups should be tested before they are trusted.

Stop the MariaDB Server Process

Before the new version can be installed, we first need to stop the current MariaDB Server process.

1. Set the innodb_fast_shutdown system variable to 1:

```
SET GLOBAL innodb_fast_shutdown = 1;
```

2. Use XA RECOVER to confirm that there are no external XA transactions in a prepared state:

```
XA RECOVER;
```

Commit or roll back any open XA transactions before stopping the node for upgrade.

3. Stop the server process. For distributions that use `systemd` (most supported OSes), you can manage the server process using the `systemctl` command:

```
sudo systemctl stop mariadb
```

Install the New Version

MariaDB Corporation provides package repositories for YUM (RHEL, AlmaLinux, CentOS, Rocky Linux), APT (Debian, Ubuntu), and ZYpp (SLES).

For a minor release upgrade, the existing MariaDB Enterprise Server packages do not need to be uninstalled. The packages are updated in place.

Install via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the YUM package repository. Installable versions of MariaDB Enterprise Server are `11.8, 11.4, 10.6, 10.5, 10.4, and 10.3`. Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

To configure YUM package repositories:

```
sudo yum install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

> **Note:**
>
> *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.8"
```

3. Update MariaDB Enterprise Server and package dependencies:

```
sudo yum update "MariaDB-*" "galera*"
```

Install via APT (Debian, Ubuntu)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the APT package repository. Installable versions of MariaDB Enterprise Server are `11.8, 11.4, 10.6, 10.5, 10.4, and 10.3`. Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

To configure APT package repositories:

```
sudo apt install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

Note:

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.8"
```

```
sudo apt update
```

3. Update MariaDB Enterprise Server and package dependencies:

```
sudo apt install --only-upgrade "mariadb-*" "galera*"
```

Install via ZYpp (SLES)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the ZYpp package repository. Installable versions of MariaDB Enterprise Server are `11.8, 11.4, 10.6, 10.5, 10.4, and 10.3`. Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

To configure ZYpp package repositories:

```
sudo zypper install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

Note:

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.8"
```

3. Update MariaDB Enterprise Server and package dependencies:

```
sudo zypper update "MariaDB-*" "galera*"
```

Configuration

For platforms that use YUM or ZYpp as a package manager: MariaDB Enterprise Server's packages bundle several configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mariadb-enterprise.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

If your version of any of these configuration files contained any custom edits, then the package manager may save your edited version with the `.rpmsave` extension during the upgrade process. If you want to continue using your version with the custom edits, then you may need to move it back.

For example, to move `server.cnf` back in place:

```
sudo mv /etc/my.cnf.d/server.cnf /etc/my.cnf.d/server.cnf.original
```

```
sudo mv /etc/my.cnf.d/server.cnf.rpmsave /etc/my.cnf.d/server.cnf
```

Starting the Server

MariaDB Enterprise Server includes configuration to start, stop, restart, enable/disable on boot, and check the status of the Server using the operating system's default process management system.

For distributions that use `systemd`, you can manage the Server process using the `systemctl` command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Upgrading the Data Directory

MariaDB Enterprise Server ships with a utility that can be used to identify and correct compatibility issues in the new version. After you upgrade your Server and start the server process, run this utility to upgrade the data directory.

The utility is called `mariadb-upgrade`:

```
sudo mariadb-upgrade
```

Testing

When MariaDB Enterprise Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the server using MariaDB Client using the `root@localhost` user account. MariaDB Client is called `mariadb`:

```
sudo mariadb
```

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 11.8.3-1-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

2. You can also verify the server version by checking the value of the version system variable with the SHOW GLOBAL STATUS statement:

```
SHOW GLOBAL VARIABLES LIKE 'version';
```

```
+---------------+----------------------------+
| Variable_name | Value                      |
+---------------+----------------------------+
| version       | 11.8.3-1-MariaDB-Enterprise |
+---------------+----------------------------+
```

3. You can also verify the server version by calling the VERSION() function:

```
SELECT VERSION();
```

```
+----------------------------+
| VERSION()                  |
+----------------------------+
| 11.8.3-1-MariaDB-Enterprise |
+----------------------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrading from MariaDB Enterprise Server 10.6 to 11.8

This guide outlines the process for performing a major version upgrade from MariaDB Enterprise Server (ES) 10.6 directly to MariaDB Enterprise Server 11.8.

> **Note:**
>
> This guide assumes you are running on a variant of Linux that uses `systemd` to manage services (such as RHEL, CentOS, AlmaLinux, Rocky Linux, Debian, Ubuntu, or SLES).

> **Note:**
>
> Running MariaDB Enterprise Server in a **Galera Cluster**? See Upgrading from MariaDB Enterprise Server 10.6 to 11.8 with Galera Cluster for the cluster-specific procedure (packages, shutdown order, and bootstrapping). The configuration changes below apply to every node.

Prerequisites

Before beginning the upgrade, ensure these precautionary measures and environment checks are completed to protect your data.

Data Backup and Integrity

- Perform a Full Backup: Use `mariadb-backup` to create a complete copy of your current data.

```
sudo mariadb-backup --backup \
      --user=mariadb-backup_user \
      --password=mariadb-backup_passwd \
      --target-dir=/data/backup/preupgrade_10.6_to_11.8
```

- Prepare the Backup: Consolidate the backup files so they are ready for immediate restoration if required.

```
sudo mariadb-backup --prepare --target-dir=/data/backup/preupgrade_10.6_to_11.8
```

- Verify Recoverability: Test your backup by restoring it to a non-production environment before proceeding.

Service and Plugin Preparation

- Audit Plugin Transition: If you currently use the MariaDB 10.6 Audit Plugin (`server_audit.so`), it is recommended to transition to the MariaDB Enterprise Audit Plugin during this upgrade. If you maintain the Community version, ensure your configuration explicitly loads it to avoid conflicts.
- Commit or Roll Back XA Transactions: Run XA RECOVER; to identify any external XA transactions in a prepared state; these must be finalized before the service is stopped.

Compatibility & Legacy Support

This ensures the team can maintain 10.6 behavior for applications that aren't ready for the new defaults.

- Handling Non-Default Character Sets: If your 10.6 instance uses `latin1` or `utf8mb3` , do not switch to `utf8mb4` immediately. You must explicitly define your existing character set in the new `my.cnf` to override the 11.8 defaults.
- The `OLD_MODE` Tip: Set `old_mode = UTF8_IS_UTF8MB3` in your configuration; this ensures that `utf8` remains an alias for the legacy 3-byte character set instead of the new 4-byte standard.
- Maintaining SSL Compatibility: Version 11.4+ enables "Zero-configuration TLS" by default. If your application does not support SSL, set `require_secure_transport = OFF` in the `[mariadb]` section of your `my.cnf` to prevent connection refusals.

Environment Compatibility

- Engineering Policy: Verify your operating system version is still supported for the 11.8 series by checking the MariaDB Engineering Policies.
- Customer Token: Have your Customer Download Token ready for the repository configuration step.

The Upgrade Procedure

> **Step 1**
>
> **Perform a Controlled Shutdown of 10.6**
>
> 1. Initiate Fast Shutdown to ensure the InnoDB engine closes cleanly.
>
> ```
> SET GLOBAL innodb_fast_shutdown = 1;
> ```
>
> 2. Stop the Service `mariadb` .
>
> ```
> sudo systemctl stop mariadb
> ```

Step 2

Purge Legacy 10.6 Packages

Remove the old version to prevent package manager conflicts before installing 11.8.

- YUM (RHEL/CentOS/Alma/Rocky): `sudo yum remove "MariaDB-*" galera-enterprise-4`
- APT (Debian/Ubuntu): `sudo apt-get remove "mariadb-*" galera-enterprise-4`
- ZYpp (SLES): `sudo zypper remove "MariaDB-*" galera-enterprise-4`

Step 3

Switch to 11.8 Enterprise Repositories

Download and run the setup script, specifying version `11.8`.

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
chmod +x mariadb_es_repo_setup
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply --mariadb-server-version="11.8"
```

Step 4

Install the 11.8 Release Series

The repository setup only configures the source; you must explicitly install the new binaries.

- YUM: `sudo yum install MariaDB-server MariaDB-backup`
- APT: `sudo apt update && sudo apt install mariadb-server mariadb-backup`
- ZYpp: `sudo zypper install MariaDB-server MariaDB-backup`

Step 5

Implement Version-Specific Configuration Changes

> **Note:**
>
> Do not apply 11.8-specific variables while the 10.6 service is active. During the package swap, update `my.cnf` to adopt the 11.8 defaults for the Optimizer Cost Model. These variables replace legacy hardcoded logic and are essential for the new engine's performance.

- Scrub: Remove legacy 10.6 variables (`old_alter_table`, etc.).
- Adopt: Add the new Optimizer Cost Model variables using their 11.8 defaults.
- Stabilize: Set `new_mode = ""` to keep query planning closer to 10.6 and prevent unpredictable execution plan changes on Day 1. `new_mode` is a `SET` variable and does not accept a literal `OFF` value — setting `OFF` aborts startup (including during Galera/wsrep recovery). Note: `""` is not the 11.8 default (the default is `FIX_DISK_TMPTABLE_COSTS`), so this is a deliberate opt-out.

> **Tip:**
>
> **Recommended `my.cnf` for Version 11.8 section**

```
[mariadb]
# --- CHARACTER SETS & COLLATIONS ---
# utf8mb4 is the modern default; UCA 14.0.0 is ID #2304
character-set-server  = utf8mb4
collation-server      = utf8mb4_uca1400_ai_ci
# Use old_mode if your app requires 3-byte utf8 aliases
old_mode              = UTF8_IS_UTF8MB3

# --- OPTIMIZER COST MODEL (CRITICAL #) ---
# Adopting 11.8 defaults replaces legacy hardcoded logic.
# Set new_mode empty to keep 10.6-like query planning ("" is not the default; the default is
FIX_DISK_TMPTABLE_COSTS).
new_mode                      = ""
optimizer_disk_read_cost      = 10.24
optimizer_disk_read_ratio     = 0.02
optimizer_extra_pruning_depth = 8
optimizer_index_block_copy_cost = 0.0356
optimizer_key_compare_cost    = 0.011361
optimizer_key_copy_cost       = 0.015685
optimizer_key_lookup_cost     = 0.435777
optimizer_key_next_find_cost  = 0.082347
optimizer_rowid_compare_cost  = 0.002653
optimizer_rowid_copy_cost     = 0.002653
optimizer_row_copy_cost       = 0.060866
optimizer_row_lookup_cost     = 0.130839
optimizer_row_next_find_cost  = 0.045916
optimizer_scan_setup_cost     = 10
optimizer_where_cost          = 0.032

# --- INNODB STORAGE ENGINE ---
innodb_flush_method      = O_DIRECT
innodb_undo_log_truncate = ON
innodb_purge_batch_size  = 1000

# --- REPLICATION SAFETY NET ---
# REQUIRED for failing back to an existing 10.6 node.
# Fixes "Character set #2304" errors on legacy replicas.
character_set_collations = ''
binlog_alter_two_phase   = 1

# --- COMPLIANCE & CLEANUP ---
transaction_isolation    = REPEATABLE-READ
transaction_read_only    = OFF
```

> **Step 6**
>
> **Bring the Service Online and Finalize Data**
>
> 1. Start the New Service: `sudo systemctl start mariadb`.
> 2. Execute the Data Upgrade Utility: This corrects system table structures and marks data files as compatible with version 11.8.
>
>    ```
>    sudo mariadb-upgrade
>    ```

Incompatible and Significant Changes

The following variables from version 10.6 have been removed, renamed, or deprecated in the 11.8 release series.

New Variables Added in 11.8

> **Note:**
>
> Once the 11.8 binaries are installed, update your `my.cnf` to define these new variables.

Optimizer Cost Model Variables

> **Warning:**
>
> **Handle with care**
>
> Because the 11.8 engine uses a completely new “weighting” system designed for modern storage (SSDs), altering these variables without extensive benchmarks can lead to severe performance degradation or inefficient query execution plans. It is strongly recommended to keep these at their defaults upon upgradation unless you are performing expert-level performance tuning.

These variables define the weights of the new optimizer. If query execution plans change after the upgrade, these parameters are the primary audit points and represent the optimizer cost model.

Variable Name	11.8 Default	Note
`NEW_MODE`	`FIX_DISK_TMPTABLE_COSTS`	Recommended: set to `""` (empty) for 10.6-like planning. `new_mode` is a `SET` variable and does not accept a literal `OFF` value (`OFF` aborts startup).
`OPTIMIZER_DISK_READ_COST`	`10.24`	**Optimizer Plan Consistency:** Row-based replication (RBR) remains unaffected by these changes. However, because the 11.8 Primary uses the new cost model while legacy 10.6 nodes use hardcoded logic, query execution plans may differ significantly between the Master and Slave.
`OPTIMIZER_DISK_READ_RATIO`	`0.02`	
`OPTIMIZER_EXTRA_PRUNING_DEPTH`	`8`	

Variable Name	11.8 Default	Note
OPTIMIZER_INDEX_BLOCK_COPY_COST	0.0356	
OPTIMIZER_KEY_COMPARE_COST	0.011361	
OPTIMIZER_KEY_COPY_COST	0.015685	
OPTIMIZER_KEY_LOOKUP_COST	0.435777	
OPTIMIZER_KEY_NEXT_FIND_COST	0.082347	
OPTIMIZER_ROWID_COMPARE_COST	0.002653	
OPTIMIZER_ROWID_COPY_COST	0.002653	
OPTIMIZER_ROW_COPY_COST	0.060866	
OPTIMIZER_ROW_LOOKUP_COST	0.130839	
OPTIMIZER_ROW_NEXT_FIND_COST	0.045916	
OPTIMIZER_SCAN_SETUP_COST	10	
OPTIMIZER_WHERE_COST	0.032	

InnoDB Variables

InnoDB used complex buffering (like the “Change Buffer”) to delay writes because hard drives were slow at random I/O. In version 11.8, these legacy layers are being stripped back. The following InnoDB System Variables allow MariaDB to communicate more directly with modern SSD/NVMe storage, reducing the “middleman” overhead of the Operating System’s cache.

Variable Name	11.8 Default
INNODB_ADAPTIVE_HASH_INDEX_CELLS ^	34679
INNODB_ALTER_COPY_BULK #	ON
INNODB_BUFFER_POOL_SIZE_AUTO_MIN	134217728
INNODB_BUFFER_POOL_SIZE_MAX	134217728
INNODB_DATA_FILE_BUFFERING	OFF
INNODB_DATA_FILE_WRITE_THROUGH	OFF
INNODB_LINUX_AIO	auto
INNODB_LOG_CHECKPOINT_NOW	OFF
INNODB_LOG_FILE_BUFFERING	OFF
INNODB_LOG_FILE_MMAP	OFF
INNODB_LOG_FILE_WRITE_THROUGH	OFF
INNODB_LOG_SPIN_WAIT_DELAY	0
INNODB_TRUNCATE_TEMPORARY_TABLESPACE_NOW	OFF

Replication & Binlog Variables

As clusters grow, managing the Global Transaction ID (GTID) list becomes a performance bottleneck. 11.8 introduces “GTID Indexing,” which treats the replication and log more like a database table, allowing for near-instant lookups when a replica reconnects. See Replication and Binary Logging Variables to learn more about the variables.

Variable Name	11.8 Default	Note
`BINLOG_ALTER_TWO_PHASE` #	`OFF`	
`BINLOG_DO_DB`	`(None)`	
`BINLOG_GTID_INDEX` #	`ON`	
`BINLOG_GTID_INDEX_PAGE_SIZE` #	`4096`	
`BINLOG_GTID_INDEX_SPAN_MIN` #	`65536`	
`BINLOG_IGNORE_DB`	`(None)`	
`BINLOG_LARGE_COMMIT_THRESHOLD`	`134217728`	
`BINLOG_LEGACY_EVENT_POS` #	`OFF`	
`BINLOG_ROW_EVENT_MAX_SIZE` #	`8192`	Packet Safety: Ensure this value on the 11.8 Primary is strictly lower than the `MAX_ALLOWED_PACKET` setting on your 10.6 Replicas. If an 11.8 row event exceeds the legacy replica's packet limit, replication will fail with a "Packet too large on slave" error.
`BINLOG_SPACE_LIMIT`	`0`	
`REPLICATE_REWRITE_DB` #	`(None)`	
`SLAVE_ABORT_BLOCKING_TIMEOUT`	`31536000`	
`SLAVE_CONNECTIONS_NEEDED_FOR_PURGE` #	`1`	**Binary Log Purge Safety:** The default value is `1`. If your topology requires that binary logs be retained until they are acknowledged by multiple replicas, increase this value to prevent the primary from purging logs before all critical slaves have synchronized the data.
`SLAVE_MAX_STATEMENT_TIME`	`0`	

Advanced Logging Variables

Slow logging was often a “blunt instrument” that could either miss critical spikes or flood the disk with useless data. Version 11.8 introduces granular filters. You can now tell the engine to ignore queries that touch many rows but are still fast, or to ensure that queries exceeding a specific “emergency” duration are always captured, regardless of other filters. See Server System Variables to learn more about the following variables

Variable Name	11.8 Default
`LOG_SLOW_ALWAYS_QUERY_TIME`	`31536000`
`LOG_SLOW_MIN_EXAMINED_ROW_LIMIT`	`0`
`LOG_SLOW_QUERY`	`OFF`
`LOG_SLOW_QUERY_FILE`	`(Host Specific)`
`LOG_SLOW_QUERY_TIME`	`10`

Security & Authentication Variables

MariaDB 11.8 shifts toward modern cryptographic standards. This includes better handling of User Defined Functions (UDFs) and a transition toward the caching_sha2_password plugin, which provides significantly stronger protection against “man-in-the-middle” and brute-force attacks compared to legacy authentication.

Variable Name	11.8 Default
`ALLOW_SUSPICIOUS_UDFS`	`OFF`
`SKIP_GRANT_TABLES` #	`OFF`

Resource Limits Variables

A single unoptimized query could theoretically consume all available disk space by creating a massive temporary table, crashing the entire operating system. Version 11.8 introduces “Safety Valves” that allow administrators to set hard limits on how much temporary space a single session or the entire server can use.

Variable Name	11.8 Default
`MAX_BINLOG_TOTAL_SIZE`	`0`
`MAX_TMP_SESSION_SPACE_USAGE`	`1099511627776` (1TB)
`MAX_TMP_TOTAL_SPACE_USAGE`	`1099511627776` (1TB)

Vector Search / MHNSW Variables

MariaDB 11.8 introduces a native Vector Search engine using the Metadata-HNSW (Hierarchical Navigable Small Worlds) algorithm. This allows the database to store and query “embeddings” (mathematical representations of text/images), enabling AI-powered semantic search directly within the SQL layer.

> **Note:**
>
> A feature entirely absent in MariaDB 10.6, MariaDB 11.8 introduced native Vector Search (MHNSW).
>
> - Primary Config: Use `MHNSW_DEFAULT_DISTANCE` to define semantic search logic (default: `euclidean`).
> - Compatibility Warning: Legacy 10.6 replicas cannot process vector data types or SQL functions; using them will break the replication link.

Variable Name	11.8 Default
`MHNSW_DEFAULT_DISTANCE` #	`euclidean`
`MHNSW_DEFAULT_M`	`6`

Variable Name	11.8 Default
`MHNSW_EF_SEARCH`	`20`
`MHNSW_MAX_CACHE_SIZE`	`16777216`

General Architecture Changes Variables

The final group of changes focuses on renaming legacy variables for industry compliance (SQL Standard) and adding features that improve how the database communicates with external tools, such as proxies or system-versioning audit logs.

Variable Name	11.8 Default	Note
`REDIRECT_URL` #	`(None)`	**Replication Redirection Safety:** The `REDIRECT_URL` variable allows the server to send redirection hints to proxies or clients. **Crucial**: Configure this carefully on replicas to ensure that internal slave-to-primary connections are not redirected, as accidental redirection of replication traffic will break the sync link.
`SYSTEM_VERSIONING_INSERT_HISTORY` #	`OFF`	**System Versioning & Binlog Replay** The default value for this variable is acceptable for standard operations. However, because MariaDB 11.8's `mariadb-binlog` utility explicitly prints this value in its output, replaying binary logs using the 11.8 version of the tool is not recommended during the transition period.
`BLOCK_ENCRYPTION_MODE` ^	`aes-128-ecb`	High; Block_encryption_mode default on 11.8 is fine, but we should be careful when calling AES_ENCRYPT and AES DECRYPT_FUNCTION as syntax is different in 10.6
`CHARACTER_SET_COLLATIONS` #	`utf8mb4=…`	Character_set_collations should be empty for 11.8 -> 10.6
`TRANSACTION_ISOLATION` #	`REPEATABLE-READ`	
`TRANSACTION_READ_ONLY` #	`OFF`	
`WSREP_ALLOWLIST`	`(None)`	
`WSREP_STATUS_FILE`	`(None)`	

Important:

#Handle them with extra care

These variables have the highest impact on system stability and performance; please review during your configuration updates.

Warning:

^Caution

These variables may impact system stability and performance; please review during your configuration updates.

Options to Remove, Rename, or Update in 11.8

Removed, Superseded or Renamed Options

During the maintenance window (after stopping 10.6 and before starting 11.8), you must scrub your `my.cnf` of all removed, superseded and renamed options.

Variable Name	10.6 Default	Technical Action / Replacement
`DATETIME_FORMAT` [2]	`%Y-%m-%d %H:%i:%s`	Scrub. 11.8 enforces standard internal format strings.
`DATE_FORMAT` [2]	`%Y-%m-%d`	Scrub. Enforced standard format.
`DEBUG_NO_THREAD_ALARM` [1]	`OFF`	Remove. Retired legacy debug code.
`INNODB_CHANGE_BUFFERING` [2]	`none`	Scrub. Logic replaced by SSD-optimized write paths.
`INNODB_CHANGE_BUFFER_MAX_SIZE` [1]	`25`	Remove. Buffer size is now managed internally by the engine.
`INNODB_DEFRAGMENT` [2]	`OFF`	Scrub. Manual defragmentation is no longer supported.
`INNODB_DEFRAGMENT_FILL_FACTOR` [2]	`0.9`	Scrub. Feature retired; logic is now internal.
`INNODB_DEFRAGMENT_FILL_FACTOR_N_RECS` [2]	`20`	Scrub. Feature retired; logic is now internal.
`INNODB_DEFRAGMENT_FREQUENCY` [2]	`40`	Scrub. Feature retired; logic is now internal.
`INNODB_DEFRAGMENT_N_PAGES` [2]	`7`	Scrub. Feature retired; logic is now internal.
`INNODB_DEFRAGMENT_STATS_ACCURACY` [2]	`0`	Scrub. Feature retired; logic is now internal.
`INNODB_VERSION` [1]	`10.6.26`	Remove. Versioning is consolidated in global server metadata.
`MAX_TMP_TABLES` [1]	`32`	Remove. Internal temp table management is now automated.
`OLD_ALTER_TABLE` [1]	`DEFAULT`	Remove. Superseded by `ALTER_ALGORITHM`.
`TIME_FORMAT` [2]	`%H:%i:%s`	Scrub. Enforced standard format.
`WSREP_CAUSAL_READS` [2]	`OFF`	Scrub. Superseded by `wsrep_sync_wait`.
`WSREP_LOAD_DATA_SPLITTING` [1]	`OFF`	Remove. Legacy Galera splitting logic retired.
`WSREP_REPLICATE_MYISAM` [1]	`OFF`	Remove. Galera no longer supports MyISAM replication.
`WSREP_STRICT_DDL` [1]	`OFF`	Remove. Replaced by `wsrep_mode=STRICT_REPLICATION`.

Important:

[1]Fatal Error

MariaDB 11.8 will abort startup if these legacy parameters are detected in the configuration file. These parameters MUST be removed prior to restart to prevent upgrade failure.

Example: `[ERROR] /usr/sbin/mariadbd: unknown variable 'MAX_TMP_TABLES=32'`

Warning:

[2]Warning:

MariaDB 11.8 will start but log a warning if these legacy parameters are detected in the configuration file. Scrub the file of these parameter during the upgrade to maintain configuration hygiene and ensure settings reflect active 11.8 features.

Warning Example: `[Warning] 'innodb-change-buffering' was removed. It does nothing now and exists only for compatibility with old my.cnf files.`

Options That Have Changed Default Values

> **Note:**
>
> For variables that have existed in both versions but have different defaults (e.g., `innodb_purge_batch_size`), the 11.8 engine will automatically apply the new value. If you require identical behavior to your 10.6 environment during the initial cutover, you must explicitly hardcode the 10.6 values into your new configuration file.

Options	10.6 Default	11.8 Default	Impact / Note
`character_set_server`	`latin1`	`utf8mb4`	Certified Change:Global default encoding shift.
`CHARACTER_SET_CLIENT`	`latin1`	`utf8mb4`	Modern standard for client connections.
`CHARACTER_SET_CONNECTION`	`latin1`	`utf8mb4`	Modern standard for session connections.
`CHARACTER_SET_DATABASE`	`latin1`	`utf8mb4`	Modern standard for database storage.
`CHARACTER_SET_RESULTS`	`latin1`	`utf8mb4`	Modern standard for query results.

Options	10.6 Default	11.8 Default	Impact / Note
`collation_server`	`latin1_swedish_ci`	`utf8mb4_uca1400_ai_ci`	Transition to the modern Unicode collation standard.
`COLLATION_CONNECTION`	`latin1_swedish_ci`	`utf8mb4_uca1400_ai_ci`	Update to modern Unicode Collation Algorithm (UCA).
`COLLATION_DATABASE`	`latin1_swedish_ci`	`utf8mb4_uca1400_ai_ci`	Update to modern Unicode Collation Algorithm (UCA).

Options	10.6 Default	11.8 Default	Impact / Note
`EXPLICIT_DEFAULTS_FOR_TIMESTAMP`	`OFF`	`ON`	Impacts `NULL` handling in `TIMESTAMP` columns. Modify for reverse replication (11.8->10.6) to maintain same behavior on master and slave
`HAVE_SSL`	`DISABLED`	`YES`	SSL/TLS is now natively available and enabled by default.

Options	10.6 Default	11.8 Default	Impact / Note
`HISTOGRAM_TYPE`	(Empty)	`JSON_HB`	Optimizer now stores histogram stats in JSON format. Modify for reverse replication (11.8->10.6) to maintain same behavior on master and slave
`IGNORE_DB_DIRS`		#binlog_cache_files	
`INNODB_BUFFER_POOL_CHUNK_SIZE`	`134217728`	`0`	Set to `0` to enable automatic calculation.
`INNODB_LOG_WRITE_AHEAD_SIZE`	`8192`	`4096`	Optimized for modern storage block alignment.

Options	10.6 Default	11.8 Default	Impact / Note
`innodb_snapshot_isolation`	`OFF`	`ON`	Enabled by default for improved consistency.
`innodb_undo_tablespaces`	`0`	`3`	Enables online truncation of undo logs.
`optimizer_prune_level`	`1`	`2`	Red Flag: Primary audit point for plan changes.

Options	10.6 Default	11.8 Default	Impact / Note
OPTIMIZER_SWITCH	index_merge=on index_merge_union=on index_merge_sort_union=on index_merge_intersection=on index_merge_sort_intersection=off engine_condition_pushdown=off index_condition_pushdown=on derived_merge=on derived_with_keys=on firstmatch=on loosescan=onmaterialization=on in_to_exists=on semijoin=on partial_match_rowid_merge=on partial_match_table_scan=on subquery_cache=on mrr=offmrr_cost_based=off mrr_sort_keys=off outer_join_with_cache=on semijoin_with_cache=on join_cache_incremental=on join_cache_hashed=onjoin_cache_bka=on optimize_join_buffer_size=on table_elimination=on extended_keys=on exists_to_in=on orderby_uses_equalities=onondition_pushdown_for_derived=on split_materialized=on condition_pushdown_for_subquery=on rowid_filter=oncondition_pushdown_from_having=on not_null_range_scan=off hash_join_cardinality=off cset_narrowing=off sargable_casefold=off	index_merge=on index_merge_union=on index_merge_sort_union=on index_merge_intersection=on index_merge_sort_intersection=offindex_condition_pushdown=on derived_merge=on derived_with_keys=on firstmatch=on loosescan=on materialization=on in_to_exists=onsemijoin=on partial_match_rowid_merge=on partial_match_table_scan=on subquery_cache=on mrr=off mrr_cost_based=off mrr_sort_keys=offouter_join_with_cache=on semijoin_with_cache=on join_cache_incremental=on join_cache_hashed=on join_cache_bka=onoptimize_join_buffer_size=on table_elimination=on extended_keys=on exists_to_in=on orderby_uses_equalities=onondition_pushdown_for_derived=on split_materialized=on condition_pushdown_for_subquery=on rowid_filter=oncondition_pushdown_from_having=on not_null_range_scan=off hash_join_cardinality=on cset_narrowing=on sargable_casefold=on	Added hash_join_cardinality=on, sargable_casefold=on.
OPTIMIZER_ADJUST_SECONDARY_KEY_COSTS	fix_reuse_range_for_ref fix_card_multiplier	0	Weights for secondary key lookups simplified.
SESSION_TRACK_SYSTEM_VARIABLES	autocommit character_set_client character_set_connection character_set_results time_zone	autocommit character_set_client character_set_connection character_set_results time_zone	Added redirect_url to the tracking list.

Options	10.6 Default	11.8 Default	Impact / Note
`VERSION`	`10.6.26-MariaDB-log`	`11.8.7-MariaDB-log`	Metadata only

Deprecated Options

Option	Reason / Recommendation
`tx_isolation`	Replaced by `transaction_isolation`.
`tx_read_only`	Replaced by `transaction_read_only`.
`innodb_purge_rseg_truncate_frequency`	Obsolete due to lighter truncation operations.

Reverse Replication (11.8 to 10.6)

If a critical regression is discovered, you can use an existing 10.6 machine in your setup as a failback safety net.

> **Warning:**
>
> Replicating from a MariaDB 11.8 Primary to a MariaDB 10.6 Replica is NOT officially supported by MariaDB Engineering. This configuration should only be used as a temporary emergency safety net during the upgrade window.

Required 11.8 Primary Configuration

To prevent the 10.6 replica from crashing due to modern metadata (such as the `#2304` character set ID), the 11.8 Primary must be configured to “downgrade” its binary log output using a compatibility file.

The `rollback_compat.cnf` Setup

Create a standalone configuration file to house these temporary reversion settings at `/etc/my.cnf.d/rollback_compat.cnf`

```
[mariadb]
# --- Core 10.6 Behavioral Reversion ---
character_set_server            = latin1
collation_server                = latin1_swedish_ci
explicit_defaults_for_timestamp = OFF
innodb_snapshot_isolation       = OFF

# --- Mandatory Metadata Compatibility (SME "Red" List) ---
# Prevents "Character set #2304" (utf8mb4_uca1400_ai_ci) errors on 10.6
character_set_collations        = ''

# Forces session metadata to legacy-compatible versions
collation_connection            = utf8mb3_general_ci
collation_database              = latin1_swedish_ci

# Standardizes log checksums for the 10.6 parser
binlog_checksum                 = CRC32
```

To activate these settings, append the following line to the end of the `[mariadb]` section in your primary `/etc/my.cnf` file: `!include /etc/my.cnf.d/rollback_compat.cnf`

> **Important:**
>
> **Critical Compatibility Step**
>
> MariaDB 11.8 uses a new default collation ID (#2304) that version 10.6 does not recognize. To prevent the 10.6 replica from crashing, you must set `character_set_collations = ''` on the 11.8 Primary. This forces the Primary to use legacy IDs that the 10.6 machine can process

Known “Breaking” Factors

Certain 11.8 features will immediately break the 10.6 replication link if used:

- Vector Data Types: Any `INSERT` or `UPDATE` involving a `VECTOR(N)` column.
- New Functions: Use of `VEC_Distance` or other 11.8-specific SQL functions.
- Large Row Events: If `binlog_row_event_max_size` is tuned significantly higher than 10.6 defaults.

> **Note:**
>
> **Note on Existing Nodes**
>
> You do not need a new machine for reverse replication. You can use an existing 10.6 node already in your setup. Simply stop replication on that node before the upgrade, and resume it once the 11.8 Primary is configured for compatibility.

Operational Steps for the Safety Net

> **Step 1**
>
> **Isolate a 10.6 Node**
>
> Before upgrading your entire environment, identify one existing replica to remain on version 10.6. Stop the replication on this node just before you upgrade the Primary to 11.8.

> **Step 2**
>
> **Configure 11.8 for Compatibility**
>
> Immediately after installing version 11.8 on your Primary, apply the `rollback_compat.cnf` settings (such as `character_set_collations = ''` and `binlog_checksum = CRC32`).

> **Step 3**
>
> **Start 11.8 and Rotate Logs**
>
> Start the 11.8 service and run `FLUSH LOGS;`. This ensures the Primary begins writing its binary logs in a format the 10.6 replica can understand.

> **Step 4**
>
> **Connect the 10.6 Node**
>
> Point your existing 10.6 machine to the new 11.8 Primary. Because the data on the 10.6 node is already consistent with the pre-upgrade state, it can simply “pick up” the new changes from the 11.8 Primary.

Post-Upgrade Verification

After the data upgrade is complete, verify the functionality of 11.8 features:

- Confirm Version: `SELECT VERSION();` should reflect the 11.8 GA series.
- Confirm Vector Search: Verify the new `VECTOR(N)` data type and conversion functions.

```
CREATE TABLE test_vector (v VECTOR(3));
SELECT VEC_ToText(VEC_FromText('[1,2,3]'));
```

- Verify Optimizer Performance: Prefix any SQL query with ANALYZE FORMAT=JSON on upgraded 11.8 instances to audit the new SSD-aware cost model:

  ```
  ANALYZE FORMAT=JSON SELECT * FROM orders WHERE total > 1000 AND status = 'shipped';
  ```

  This captures real-time execution metrics like `engine_cost` (measured in ms) and `pages_accessed`, verifying that the optimizer is correctly prioritizing high-speed storage over legacy 10.6 logic:

  ```
  "table": {
    "table_name": "orders",
    "access_type": "index_scan",
    "engine_cost": 4.12, // Actual cost in MILLISECONDS (ms)
    "r_engine_stats": {
      "pages_accessed": 142, // Real-world I/O footprint
      "pages_read_time_ms": 1.05 // Precise SSD latency
    }
  }
  ```

- Check Replication Lag Fields: On a replica server, run `SHOW REPLICA STATUS\G` and look for the new `Master_Slave_time_diff` field.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade from MariaDB Community Server 10.6 to MariaDB Enterprise Server 11.8

This guide outlines a direct upgrade from **MariaDB Community Server 10.6** to **MariaDB Enterprise Server 11.8**. It combines two transitions in a single upgrade window: a product switch (Community to Enterprise) and a major version jump (10.6 to 11.8). With MariaDB Community Server 10.6 approaching end-of-life, this path lets you move to a maintained Enterprise release without an intermediate stop on ES 10.6.

> **Note:**
>
> This guide assumes you are running on a variant of Linux that uses `systemd` to manage services (such as RHEL, CentOS, AlmaLinux, Rocky Linux, Debian, Ubuntu, or SLES).

> **Warning:**
>
> This is a substantial change: a product switch and a major version jump performed back-to-back. Read this guide end-to-end before you start, and rehearse the procedure in a non-production environment first. Review the MariaDB Engineering Policies to confirm your operating system is supported on the 11.8 release series.

Prerequisites

Complete these checks and preparations before stopping your 10.6 service.

Data Backup and Integrity

Occasionally, issues can be encountered during upgrades. These issues can even potentially corrupt the database's data files, preventing you from easily reverting to the old installation. It is generally best to perform a backup prior to upgrading. If an issue is encountered during the upgrade, you can use the backup to restore your MariaDB Server database to the old version. If the upgrade finishes without issue, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, please see the Recovery Guide.

1. Take a full backup.

```
sudo mariadb-backup --backup \
      --user=mariadb-backup_user \
      --password=mariadb-backup_passwd \
      --target-dir=/data/backup/preupgrade_cs10.6_to_es11.8
```

Confirm successful completion of the backup operation.

2. Prepare the backup so it is ready for immediate restoration if required.

```
sudo mariadb-backup --prepare \
      --target-dir=/data/backup/preupgrade_cs10.6_to_es11.8
```

Confirm successful completion of the prepare operation.

3. Verify recoverability: test your backup by restoring it to a non-production environment before proceeding.

Convert InnoDB Row Format

MariaDB Enterprise Server 10.6 and later changed the `COMPRESSED` InnoDB row format to read-only. Before upgrading to MariaDB Enterprise Server 11.8, modify any compressed InnoDB tables on your 10.6 Community Server to use the `DYNAMIC` row format with `PAGE_COMPRESSED=1`.

1. Identify any InnoDB tables that use the `COMPRESSED` row format:

```
SELECT NAME, ROW_FORMAT
FROM information_schema.INNODB_SYS_TABLES
WHERE NAME NOT LIKE 'SYS_%'
   AND ROW_FORMAT = 'COMPRESSED';
```

2. Convert each table to `DYNAMIC` row format with page compression:

```
ALTER TABLE accounts.hq_sales
ROW_FORMAT = DYNAMIC
PAGE_COMPRESSED = 1;
```

Audit Plugin Considerations

If you have the MariaDB Audit Plugin (`server_audit.so`) installed on your Community Server, remove it before upgrading. Otherwise it will conflict with the MariaDB Enterprise Audit Plugin that ships with MariaDB Enterprise Server 10.4 and later.

Remove the plugin by using the UNINSTALL SONAME statement:

```
UNINSTALL SONAME 'server_audit';
```

If you load the plugin in a configuration file using the `plugin_load_add` option, remove that option as well. The MariaDB Enterprise Audit Plugin is automatically installed when you install MariaDB Enterprise Server.

Compatibility and Legacy Support

These defaults ensure 11.8 keeps behaving like your 10.6 instance for applications that aren't yet ready for the new defaults.

- **Non-default character sets:** If your 10.6 instance uses `latin1` or `utf8mb3`, do not switch to `utf8mb4` immediately. Explicitly define your existing character set in the new `my.cnf` to override the 11.8 defaults.
- **`OLD_MODE` tip:** Set `old_mode = UTF8_IS_UTF8MB3` in your configuration so that `utf8` remains an alias for the legacy 3-byte character set instead of the new 4-byte standard.
- **SSL compatibility:** MariaDB 11.4 and later enable "Zero-configuration TLS" by default. If your application does not support SSL, set `require_secure_transport = OFF` in the `[mariadb]` section of `my.cnf` to prevent

connection refusals.

- **XA transactions:** Run `XA RECOVER;` to identify any external XA transactions in a prepared state. These must be committed or rolled back before the service is stopped.

Environment Compatibility

- **Engineering policy:** Verify your operating system version is supported for the 11.8 series in the MariaDB Engineering Policies.
- **Customer Download Token:** Retrieve your token from https://customers.mariadb.com/downloads/token/. You'll need it to configure the Enterprise repository in the steps below.

The Upgrade Procedure

> **Step 1**
>
> **Perform a Controlled Shutdown of MariaDB Community Server 10.6**
>
> 1. Initiate a fast shutdown so the InnoDB engine closes cleanly:
>
>    ```
>    SET GLOBAL innodb_fast_shutdown = 1;
>    ```
>
> 2. Stop the service:
>
>    ```
>    sudo systemctl stop mariadb
>    ```

Step 2

Purge the MariaDB Community Server 10.6 Packages

When switching from MariaDB Community Server to MariaDB Enterprise Server, it is necessary to remove the existing Community Server installation before installing Enterprise Server. Otherwise, the package manager will refuse to install the Enterprise Server packages.

For Community Server 10.6, the Galera package is `galera-4` (named `galera-3` on APT for some legacy 10.3 installations, but 10.6 always uses `galera-4`).

UNINSTALL VIA YUM

On RHEL, AlmaLinux, CentOS, and Rocky Linux:

```
sudo yum remove "MariaDB-*" galera-4
```

Verify that all MariaDB Community Server packages are uninstalled. The following command should return no results:

```
rpm --query --all | grep -i -E "mariadb|galera"
```

UNINSTALL VIA APT

On Debian and Ubuntu:

```
sudo apt-get remove "mariadb-*" galera-4
```

Verify that all MariaDB Community Server packages are uninstalled. The following command should return no results:

```
apt list --installed | grep -i -E "mariadb|galera"
```

UNINSTALL VIA ZYPP

On SLES:

```
sudo zypper remove "MariaDB-*" galera-4
```

Verify that all MariaDB Community Server packages are uninstalled. The following command should return no results:

```
rpm --query --all | grep -i -E "mariadb|galera"
```

Be sure to check that the wildcard does not unintentionally refer to any of your custom applications.

Step 3

Configure the MariaDB Enterprise Server 11.8 Repository

Download the repository setup script, verify it, and run it with your Customer Download Token to register the 11.8 Enterprise repository. Replace `CUSTOMER_DOWNLOAD_TOKEN` with the token retrieved during Prerequisites, and substitute `${checksum}` with the latest published checksum from the Versions section of the MariaDB Package Repository Setup and Usage page.

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
echo "${checksum}  mariadb_es_repo_setup" | sha256sum -c -
chmod +x mariadb_es_repo_setup
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.8"
```

On APT systems, refresh the package index after running the setup script:

```
sudo apt update
```

Step 4

Install MariaDB Enterprise Server 11.8

The repository setup only configures the source; install the new binaries explicitly.

INSTALL VIA YUM

```
sudo yum install MariaDB-server MariaDB-backup
```

INSTALL VIA APT

```
sudo apt install mariadb-server mariadb-backup
```

INSTALL VIA ZYPP

```
sudo zypper install MariaDB-server MariaDB-backup
```

Installation of additional packages may be required for some plugins. Installation only loads MariaDB Enterprise Server onto the system — configuration is the next step.

Step 5

Implement Version-Specific Configuration Changes

> **Note:**
>
> Do not apply 11.8-specific variables while the 10.6 service is active. During the package swap, update `my.cnf` to adopt the 11.8 defaults for the Optimizer Cost Model. These variables replace legacy hardcoded logic and are essential to the new engine's performance.

- **Scrub** legacy 10.6 variables (`old_alter_table`, etc.) — see Removed, Superseded, or Renamed Options below.
- **Adopt** the new Optimizer Cost Model variables using their 11.8 defaults.
- **Stabilize** with `NEW_MODE = OFF` to prevent unpredictable execution-plan changes on day one.

> **Tip:**
>
> **Recommended `my.cnf` for Version 11.8**

```
[mariadb]
# --- CHARACTER SETS & COLLATIONS ---
# utf8mb4 is the modern default; UCA 14.0.0 is ID #2304
character-set-server  = utf8mb4
collation-server      = utf8mb4_uca1400_ai_ci
# Use old_mode if your app requires 3-byte utf8 aliases
old_mode              = UTF8_IS_UTF8MB3

# --- OPTIMIZER COST MODEL (CRITICAL) ---
# Adopting 11.8 defaults replaces legacy hardcoded logic.
# Set NEW_MODE = OFF to stabilize query plans during initial cutover.
new_mode                        = OFF
optimizer_disk_read_cost        = 10.24
optimizer_disk_read_ratio       = 0.02
optimizer_extra_pruning_depth   = 8
optimizer_index_block_copy_cost = 0.0356
optimizer_key_compare_cost      = 0.011361
optimizer_key_copy_cost         = 0.015685
optimizer_key_lookup_cost       = 0.435777
optimizer_key_next_find_cost    = 0.082347
optimizer_rowid_compare_cost    = 0.002653
optimizer_rowid_copy_cost       = 0.002653
optimizer_row_copy_cost         = 0.060866
optimizer_row_lookup_cost       = 0.130839
optimizer_row_next_find_cost    = 0.045916
optimizer_scan_setup_cost       = 10
optimizer_where_cost            = 0.032

# --- INNODB STORAGE ENGINE ---
innodb_flush_method      = O_DIRECT
innodb_undo_log_truncate = ON
innodb_purge_batch_size  = 1000

# --- REPLICATION SAFETY NET ---
# REQUIRED if you intend to fail back to an existing 10.6 node.
# Fixes "Character set #2304" errors on legacy replicas.
character_set_collations = ''
binlog_alter_two_phase   = 1

# --- COMPLIANCE & CLEANUP ---
transaction_isolation    = REPEATABLE-READ
transaction_read_only    = OFF
```

> If your platform uses YUM or ZYpp, note that MariaDB Enterprise Server's packages bundle the following configuration files: `/etc/my.cnf`, `/etc/my.cnf.d/client.cnf`, `/etc/my.cnf.d/mariadb-enterprise.cnf`, `/etc/my.cnf.d/mysql-clients.cnf`, and `/etc/my.cnf.d/server.cnf`. If your previous Community Server configuration files contained custom edits, the package manager may save your edited version with the `.rpmsave` extension during installation. Restore your edits before starting the new server.

> **Step 6**
>
> **Bring the Service Online and Run mariadb-upgrade**
>
> 1. Start the new service:
>
>    ```
>    sudo systemctl start mariadb
>    ```
>
> 2. Run the data-directory upgrade utility. It corrects system table structures and marks data files as compatible with version 11.8:
>
>    ```
>    sudo mariadb-upgrade
>    ```

Incompatible and Significant Changes

The following variables from version 10.6 have been removed, renamed, or deprecated in the 11.8 release series.

New Variables Added in 11.8

> **Note:**
>
> Once the 11.8 binaries are installed, update your `my.cnf` to define these new variables.

Optimizer Cost Model Variables

> **Warning:**
>
> **Handle with care.** The 11.8 engine uses a new weighting system designed for modern storage (SSDs). Altering these variables without extensive benchmarks can lead to severe performance degradation or inefficient query execution plans. Keep these at their defaults upon upgrade unless you are performing expert-level performance tuning.

These variables define the weights of the new optimizer. If query execution plans change after the upgrade, these parameters are the primary audit points and represent the optimizer cost model.

Variable Name	11.8 Default	Note
`NEW_MODE`	`OFF`	
`OPTIMIZER_DISK_READ_COST`	`10.24`	**Optimizer Plan Consistency:** Row-based replication (RBR) remains unaffected by these changes. However, because the 11.8 Primary uses the new cost model while legacy 10.6 nodes use hardcoded logic, query execution plans may differ significantly between Master and Slave.
`OPTIMIZER_DISK_READ_RATIO`	`0.02`	
`OPTIMIZER_EXTRA_PRUNING_DEPTH`	`8`	

Variable Name	11.8 Default	Note
OPTIMIZER_INDEX_BLOCK_COPY_COST	0.0356	
OPTIMIZER_KEY_COMPARE_COST	0.011361	
OPTIMIZER_KEY_COPY_COST	0.015685	
OPTIMIZER_KEY_LOOKUP_COST	0.435777	
OPTIMIZER_KEY_NEXT_FIND_COST	0.082347	
OPTIMIZER_ROWID_COMPARE_COST	0.002653	
OPTIMIZER_ROWID_COPY_COST	0.002653	
OPTIMIZER_ROW_COPY_COST	0.060866	
OPTIMIZER_ROW_LOOKUP_COST	0.130839	
OPTIMIZER_ROW_NEXT_FIND_COST	0.045916	
OPTIMIZER_SCAN_SETUP_COST	10	
OPTIMIZER_WHERE_COST	0.032	

InnoDB Variables

InnoDB used complex buffering (such as the Change Buffer) to delay writes because hard drives were slow at random I/O. In version 11.8 these legacy layers are stripped back. The following InnoDB system variables let MariaDB communicate more directly with modern SSD/NVMe storage, reducing the “middleman” overhead of the operating system’s cache.

Variable Name	11.8 Default
INNODB_ADAPTIVE_HASH_INDEX_CELLS ^	34679
INNODB_ALTER_COPY_BULK #	ON
INNODB_BUFFER_POOL_SIZE_AUTO_MIN	134217728
INNODB_BUFFER_POOL_SIZE_MAX	134217728
INNODB_DATA_FILE_BUFFERING	OFF
INNODB_DATA_FILE_WRITE_THROUGH	OFF
INNODB_LINUX_AIO	auto
INNODB_LOG_CHECKPOINT_NOW	OFF
INNODB_LOG_FILE_BUFFERING	OFF
INNODB_LOG_FILE_MMAP	OFF
INNODB_LOG_FILE_WRITE_THROUGH	OFF
INNODB_LOG_SPIN_WAIT_DELAY	0
INNODB_TRUNCATE_TEMPORARY_TABLESPACE_NOW	OFF

Replication and Binlog Variables

As clusters grow, managing the Global Transaction ID (GTID) list becomes a performance bottleneck. 11.8 introduces “GTID Indexing,” which treats the replication log more like a database table, allowing near-instant lookups when a replica reconnects. See Replication and Binary Logging Variables for details.

Variable Name	11.8 Default	Note
`BINLOG_ALTER_TWO_PHASE` #	`OFF`	
`BINLOG_DO_DB`	`(None)`	
`BINLOG_GTID_INDEX` #	`ON`	
`BINLOG_GTID_INDEX_PAGE_SIZE` #	`4096`	
`BINLOG_GTID_INDEX_SPAN_MIN` #	`65536`	
`BINLOG_IGNORE_DB`	`(None)`	
`BINLOG_LARGE_COMMIT_THRESHOLD`	`134217728`	
`BINLOG_LEGACY_EVENT_POS` #	`OFF`	
`BINLOG_ROW_EVENT_MAX_SIZE` #	`8192`	Packet safety: ensure this value on the 11.8 Primary is strictly lower than the `MAX_ALLOWED_PACKET` setting on your 10.6 Replicas. If an 11.8 row event exceeds the legacy replica's packet limit, replication will fail with a "Packet too large on slave" error.
`BINLOG_SPACE_LIMIT`	`0`	
`REPLICATE_REWRITE_DB` #	`(None)`	
`SLAVE_ABORT_BLOCKING_TIMEOUT`	`31536000`	
`SLAVE_CONNECTIONS_NEEDED_FOR_PURGE` #	`1`	**Binary log purge safety:** The default value is `1`. If your topology requires that binary logs be retained until acknowledged by multiple replicas, increase this value to prevent the primary from purging logs before all critical slaves have synchronized the data.
`SLAVE_MAX_STATEMENT_TIME`	`0`	

Advanced Logging Variables

Slow logging was often a blunt instrument that could either miss critical spikes or flood the disk with useless data. Version 11.8 introduces granular filters: you can ignore queries that touch many rows but are still fast, or ensure that queries exceeding a specific “emergency” duration are always captured regardless of other filters. See Server System Variables for details.

Variable Name	11.8 Default
`LOG_SLOW_ALWAYS_QUERY_TIME`	`31536000`
`LOG_SLOW_MIN_EXAMINED_ROW_LIMIT`	`0`
`LOG_SLOW_QUERY`	`OFF`
`LOG_SLOW_QUERY_FILE`	`(Host Specific)`
`LOG_SLOW_QUERY_TIME`	`10`

Security and Authentication Variables

MariaDB 11.8 shifts toward modern cryptographic standards. This includes better handling of User-Defined Functions (UDFs) and a transition toward the caching_sha2_password plugin, which provides significantly stronger protection against man-in-the-middle and brute-force attacks compared to legacy authentication.

Variable Name	11.8 Default
`ALLOW_SUSPICIOUS_UDFS`	`OFF`
`SKIP_GRANT_TABLES` #	`OFF`

Resource Limits Variables

A single unoptimized query could theoretically consume all available disk space by creating a massive temporary table, crashing the operating system. Version 11.8 introduces safety valves that let administrators set hard limits on how much temporary space a single session or the entire server can use.

Variable Name	11.8 Default
`MAX_BINLOG_TOTAL_SIZE`	`0`
`MAX_TMP_SESSION_SPACE_USAGE`	`1099511627776` (1TB)
`MAX_TMP_TOTAL_SPACE_USAGE`	`1099511627776` (1TB)

Vector Search / MHNSW Variables

MariaDB 11.8 introduces a native Vector Search engine using the Metadata-HNSW (Hierarchical Navigable Small Worlds) algorithm. This allows the database to store and query embeddings (mathematical representations of text/images), enabling AI-powered semantic search directly within the SQL layer.

> **Note:**
>
> Vector Search is entirely absent from MariaDB 10.6.
>
> - Primary config: use `MHNSW_DEFAULT_DISTANCE` to define semantic search logic (default: `euclidean`).
> - Compatibility warning: legacy 10.6 replicas cannot process vector data types or SQL functions; using them will break the replication link.

Variable Name	11.8 Default
`MHNSW_DEFAULT_DISTANCE` #	`euclidean`
`MHNSW_DEFAULT_M`	`6`

Variable Name	11.8 Default
`MHNSW_EF_SEARCH`	`20`
`MHNSW_MAX_CACHE_SIZE`	`16777216`

General Architecture Changes Variables

The final group of changes focuses on renaming legacy variables for industry compliance (SQL Standard) and adding features that improve how the database communicates with external tools such as proxies or system-versioning audit logs.

Variable Name	11.8 Default	Note
`REDIRECT_URL` #	`(None)`	**Replication redirection safety:** `REDIRECT_URL` lets the server send redirection hints to proxies or clients. **Crucial:** configure this carefully on replicas to ensure internal slave-to-primary connections are not redirected, as accidental redirection of replication traffic will break the sync link.
`SYSTEM_VERSIONING_INSERT_HISTORY` #	`OFF`	**System versioning & binlog replay:** The default is acceptable for standard operations. Because MariaDB 11.8's `mariadb-binlog` utility explicitly prints this value in its output, replaying binary logs with the 11.8 version of the tool is not recommended during the transition period.
`BLOCK_ENCRYPTION_MODE` ^	`aes-128-ecb`	The 11.8 default is fine, but exercise care when calling `AES_ENCRYPT` and `AES_DECRYPT`; the syntax differs from 10.6.
`CHARACTER_SET_COLLATIONS` #	`utf8mb4=…`	`character_set_collations` must be empty for reverse replication (11.8 → 10.6).
`TRANSACTION_ISOLATION` #	`REPEATABLE-READ`	
`TRANSACTION_READ_ONLY` #	`OFF`	
`WSREP_ALLOWLIST`	`(None)`	
`WSREP_STATUS_FILE`	`(None)`	

> **Important:**
>
> #**Handle with extra care.** These variables have the highest impact on system stability and performance; please review during your configuration updates.

> **Warning:**
>
> ^**Caution.** These variables may impact system stability and performance; please review during your configuration updates.

Options to Remove, Rename, or Update in 11.8

Removed, Superseded, or Renamed Options

During the maintenance window (after stopping 10.6 and before starting 11.8), you must scrub your `my.cnf` of all removed, superseded, and renamed options.

Variable Name	10.6 Default	Technical Action / Replacement
`DATETIME_FORMAT` [2]	`%Y-%m-%d %H:%i:%s`	Scrub. 11.8 enforces standard internal format strings.
`DATE_FORMAT` [2]	`%Y-%m-%d`	Scrub. Enforced standard format.
`DEBUG_NO_THREAD_ALARM` [1]	`OFF`	Remove. Retired legacy debug code.
`INNODB_CHANGE_BUFFERING` [2]	`none`	Scrub. Logic replaced by SSD-optimized write paths.
`INNODB_CHANGE_BUFFER_MAX_SIZE` [1]	`25`	Remove. Buffer size is now managed internally by the engine.
`INNODB_DEFRAGMENT` [2]	`OFF`	Scrub. Manual defragmentation is no longer supported.
`INNODB_DEFRAGMENT_FILL_FACTOR` [2]	`0.9`	Scrub. Feature retired; logic is now internal.
`INNODB_DEFRAGMENT_FILL_FACTOR_N_RECS` [2]	`20`	Scrub. Feature retired; logic is now internal.
`INNODB_DEFRAGMENT_FREQUENCY` [2]	`40`	Scrub. Feature retired; logic is now internal.
`INNODB_DEFRAGMENT_N_PAGES` [2]	`7`	Scrub. Feature retired; logic is now internal.
`INNODB_DEFRAGMENT_STATS_ACCURACY` [2]	`0`	Scrub. Feature retired; logic is now internal.
`INNODB_VERSION` [1]	`10.6.26`	Remove. Versioning is consolidated in global server metadata.
`MAX_TMP_TABLES` [1]	`32`	Remove. Internal temp table management is now automated.
`OLD_ALTER_TABLE` [1]	`DEFAULT`	Remove. Superseded by `ALTER_ALGORITHM`.
`TIME_FORMAT` [2]	`%H:%i:%s`	Scrub. Enforced standard format.
`WSREP_CAUSAL_READS` [2]	`OFF`	Scrub. Superseded by `wsrep_sync_wait`.
`WSREP_LOAD_DATA_SPLITTING` [1]	`OFF`	Remove. Legacy Galera splitting logic retired.
`WSREP_REPLICATE_MYISAM` [1]	`OFF`	Remove. Galera no longer supports MyISAM replication.
`WSREP_STRICT_DDL` [1]	`OFF`	Remove. Replaced by `wsrep_mode=STRICT_REPLICATION`.

Important:

[1]**Fatal error.** MariaDB 11.8 will abort startup if these legacy parameters are detected in the configuration file. They MUST be removed before restart to prevent upgrade failure.

Example: `[ERROR] /usr/sbin/mariadbd: unknown variable 'MAX_TMP_TABLES=32'`

Warning:

[2]**Warning.** MariaDB 11.8 will start but log a warning if these legacy parameters are detected in the configuration file. Scrub them during the upgrade to maintain configuration hygiene and ensure settings reflect active 11.8 features.

Warning example: `[Warning] 'innodb-change-buffering' was removed. It does nothing now and exists only for compatibility with old my.cnf files.`

Options That Have Changed Default Values

Note:

For variables that exist in both versions but have different defaults (such as `innodb_purge_batch_size`), the 11.8 engine automatically applies the new value. If you require identical behavior to your 10.6 environment during the initial cutover, hardcode the 10.6 values into your new configuration file.

Option	10.6 Default	11.8 Default	Impact / Note
`character_set_server`	`latin1`	`utf8mb4`	Global default encoding shift.
`CHARACTER_SET_CLIENT`	`latin1`	`utf8mb4`	Modern standard for client connections.

Option	10.6 Default	11.8 Default	Impact / Note
`CHARACTER_SET_CONNECTION`	`latin1`	`utf8mb4`	Modern standard for session connections.
`CHARACTER_SET_DATABASE`	`latin1`	`utf8mb4`	Modern standard for database storage.
`CHARACTER_SET_RESULTS`	`latin1`	`utf8mb4`	Modern standard for query results.
`collation_server`	`latin1_swedish_ci`	`utf8mb4_uca1400_ai_ci`	Transition to the modern Unicode collation standard.
`COLLATION_CONNECTION`	`latin1_swedish_ci`	`utf8mb4_uca1400_ai_ci`	Update to modern Unicode Collation Algorithm (UCA).
`COLLATION_DATABASE`	`latin1_swedish_ci`	`utf8mb4_uca1400_ai_ci`	Update to modern Unicode Collation Algorithm (UCA).
`EXPLICIT_DEFAULTS_FOR_TIMESTAMP`	`OFF`	`ON`	Impacts `NULL` handling in `TIMESTAMP` columns. Modify for reverse replication (11.8 → 10.6) to maintain the same behavior on master and slave.
`HAVE_SSL`	`DISABLED`	`YES`	SSL/TLS is now natively available and enabled by default.
`HISTOGRAM_TYPE`	(Empty)	`JSON_HB`	Optimizer now stores histogram stats in JSON format. Modify for reverse replication (11.8 → 10.6).
`INNODB_BUFFER_POOL_CHUNK_SIZE`	`134217728`	`0`	Set to `0` to enable automatic calculation.
`INNODB_LOG_WRITE_AHEAD_SIZE`	`8192`	`4096`	Optimized for modern storage block alignment.
`innodb_snapshot_isolation`	`OFF`	`ON`	Enabled by default for improved consistency.
`innodb_undo_tablespaces`	`0`	`3`	Enables online truncation of undo logs.
`optimizer_prune_level`	`1`	`2`	Primary audit point for plan changes.
`VERSION`	`10.6.26-MariaDB-log`	`11.8.7-MariaDB-log`	Metadata only.

Deprecated Options

Option	Reason / Recommendation
`tx_isolation`	Replaced by `transaction_isolation`.
`tx_read_only`	Replaced by `transaction_read_only`.
`innodb_purge_rseg_truncate_frequency`	Obsolete due to lighter truncation operations.

Reverse Replication (11.8 to 10.6)

If a critical regression is discovered, you can use an existing 10.6 machine in your setup as a failback safety net.

> **Warning:**
>
> Replicating from a MariaDB 11.8 Primary to a MariaDB 10.6 Replica is NOT officially supported by MariaDB Engineering. This configuration should only be used as a temporary emergency safety net during the upgrade window.

Required 11.8 Primary Configuration

To prevent the 10.6 replica from crashing due to modern metadata (such as the `#2304` character set ID), the 11.8 Primary must be configured to “downgrade” its binary log output using a compatibility file.

The `rollback_compat.cnf` Setup

Create a standalone configuration file to house these temporary reversion settings at `/etc/my.cnf.d/rollback_compat.cnf`:

```
[mariadb]
# --- Core 10.6 Behavioral Reversion ---
character_set_server            = latin1
collation_server                = latin1_swedish_ci
explicit_defaults_for_timestamp = OFF
innodb_snapshot_isolation       = OFF

# --- Mandatory Metadata Compatibility ---
# Prevents "Character set #2304" (utf8mb4_uca1400_ai_ci) errors on 10.6
character_set_collations        = ''

# Force session metadata to legacy-compatible versions
collation_connection            = utf8mb3_general_ci
collation_database              = latin1_swedish_ci

# Standardize log checksums for the 10.6 parser
binlog_checksum                 = CRC32
```

To activate these settings, append the following line to the end of the `[mariadb]` section in your primary `/etc/my.cnf` file: `!include /etc/my.cnf.d/rollback_compat.cnf`

> **Important:**
>
> **Critical Compatibility Step**
>
> MariaDB 11.8 uses a new default collation ID (#2304) that version 10.6 does not recognize. To prevent the 10.6 replica from crashing, you must set `character_set_collations = ''` on the 11.8 Primary. This forces the Primary to use legacy IDs that the 10.6 machine can process.

Known “Breaking” Factors

Certain 11.8 features will immediately break the 10.6 replication link if used:

- **Vector data types:** any `INSERT` or `UPDATE` involving a `VECTOR(N)` column.
- **New functions:** use of `VEC_Distance` or other 11.8-specific SQL functions.
- **Large row events:** if `binlog_row_event_max_size` is tuned significantly higher than 10.6 defaults.

> **Note:**
>
> **Note on Existing Nodes**
>
> You do not need a new machine for reverse replication. You can use an existing 10.6 node already in your setup — stop replication on that node before the upgrade and resume it once the 11.8 Primary is configured for compatibility.

Operational Steps for the Safety Net

> **Step 1**
>
> **Isolate a 10.6 Node**
>
> Before upgrading your entire environment, identify one existing replica to remain on version 10.6. Stop replication on this node just before you upgrade the Primary to 11.8.

> **Step 2**
>
> **Configure 11.8 for Compatibility**
>
> Immediately after installing version 11.8 on your Primary, apply the `rollback_compat.cnf` settings (such as `character_set_collations = ''` and `binlog_checksum = CRC32`).

> **Step 3**
>
> **Start 11.8 and Rotate Logs**
>
> Start the 11.8 service and run `FLUSH LOGS;`. This ensures the Primary begins writing its binary logs in a format the 10.6 replica can understand.

> **Step 4**
>
> **Connect the 10.6 Node**
>
> Point your existing 10.6 machine to the new 11.8 Primary. Because the data on the 10.6 node is already consistent with the pre-upgrade state, it can simply pick up the new changes from the 11.8 Primary.

Post-Upgrade Verification

After the data upgrade is complete, verify the functionality of 11.8 features.

1. Confirm the server version:

   ```
   SELECT VERSION();
   ```

   The result should reflect the 11.8 GA series.

2. Confirm Vector Search by verifying the new `VECTOR(N)` data type and conversion functions:

   ```
   CREATE TABLE test_vector (v VECTOR(3));
   SELECT VEC_ToText(VEC_FromText('[1,2,3]'));
   ```

3. Verify optimizer performance by prefixing any SQL query with `ANALYZE FORMAT=JSON` to audit the new SSD-aware cost model:

   ```
   ANALYZE FORMAT=JSON SELECT * FROM orders WHERE total > 1000 AND status = 'shipped';
   ```

   The output includes real-time execution metrics such as `engine_cost` (measured in milliseconds) and `pages_accessed`, verifying that the optimizer is correctly prioritizing high-speed storage over legacy 10.6 logic:

   ```
   "table": {
     "table_name": "orders",
     "access_type": "index_scan",
     "engine_cost": 4.12,
     "r_engine_stats": {
       "pages_accessed": 142,
       "pages_read_time_ms": 1.05
     }
   }
   ```

4. On a replica server, run `SHOW REPLICA STATUS\G` and look for the new `Master_Slave_time_diff` field.

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Server 11.4

See Upgrade to MariaDB Enterprise Server 11.4.

Instructions for upgrading to MariaDB Enterprise Server 11.4, which introduces new data types like UUID and INET4, advanced JSON functions, and non-blocking online schema changes.

See Upgrade MariaDB Enterprise Server from 11.4.X to 11.4.Y.

Instructions for upgrading to MariaDB Enterprise Server 11.4, which introduces new data types like UUID and INET4, advanced JSON functions, and non-blocking online schema changes.

See Upgrade MariaDB Enterprise Server from 10.6 to 11.4.

Upgrade MariaDB Enterprise Server from 10.6 to 11.4, covering the query-optimizer changes introduced in 11.0, with both a standard in-place upgrade and a staged, replication-based rollout.

See Upgrade from MariaDB Community Server to MariaDB Enterprise Server 11.4.

Instructions for upgrading to MariaDB Enterprise Server 11.4, which introduces new data types like UUID and INET4, advanced JSON functions, and non-blocking online schema changes.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade to MariaDB Enterprise Server 11.4

Overview

These instructions detail the **upgrade** from a previous version of **MariaDB Enterprise Server** to **MariaDB Enterprise Server 11.4** on a range of supported Operating Systems.

When MariaDB Enterprise Server is upgraded, the old version needs to be uninstalled, and the new version needs to be installed.

See What's New in MariaDB Enterprise Server 11.4.

Incompatible Changes

MariaDB Enterprise Server 11.4 includes changes from previous release series (10.11, 11.0, 11.1, etc.). If you are upgrading from **MariaDB Enterprise Server 10.6** or older, you are skipping several major versions. Review the following breaking changes to ensure application stability.

Removed Variables (Action Required)

The following variables have been removed. You **must remove** them from your configuration files (`my.cnf`, `.ini`) before starting the upgrade, or the server will fail to start:

- **`innodb_defragment`**: (and all associated variables like `innodb_defragment_fill_factor`, `innodb_defragment_n_pages`).
- **`des_encrypt`** / **`des_decrypt`**: These functions have been removed.
- **`innodb_version`**: This variable is removed as it was redundant.

Behavior Changes & Deprecations

- **Binary Names**: Scripts using `mysql`, `mysqldump`, or `mysqladmin` may fail or emit deprecation warnings. Update automation to use `mariadb`, `mariadb-dump`, and `mariadb-admin`.
- **Compression Plugins**: If you used non-zlib compression (like `lz4` or `lzo`) in 10.6, ensure you have the necessary provider plugins installed in 11.4. Tables may be unreadable until the correct `mariadb-plugin-provider-*` package is installed.
- **Spider Engine**: Default values for Spider buffers (e.g., `spider_buffer_size`) have changed significantly (often autosized). If you rely on specific Spider tuning from 10.6, verify your settings.

- **Replication (GTID)**: ES 11.4 defaults to stricter GTID consistency. If upgrading a cluster, ensure Replicas are compatible before upgrading the Primary.
- **InnoDB Redo Log**: The `innodb_log_file_size` variable is now dynamic and largely ignored. The physical format of the redo log changed in 10.9; ensure you have a valid logical backup (mariadb-dump) before upgrading.
- **Optimizer (Cost Model)**: The Query Optimizer was rewritten in version 11.0. Execution plans may change. It is recommended to run `ANALYZE TABLE` on major tables after the upgrade.

Data Backup

Occasionally, issues can be encountered during upgrades. These issues can even potentially corrupt the database's data files, preventing you from easily reverting to the old installation. Therefore, it is generally best to perform a backup before upgrading. If an issue is encountered during the upgrade, you can use the backup to restore your MariaDB Server database to the old version. If the upgrade finishes without issue, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, please see the Recovery Guide.

1. Take a full backup. On MariaDB Enterprise Server 11.4 and later:

```
sudo mariadb-backup --backup \
      --user=mariadb-backup_user \
      --password=mariadb-backup_passwd \
      --target-dir=/data/backup/preupgrade_backup
```

Confirm successful completion of the backup operation.

2. The backup must be prepared. On the MariaDB Enterprise Server 10.4 and later:

```
sudo mariadb-backup --prepare \
      --target-dir=/data/backup/preupgrade_backup
```

Confirm successful completion of the prepare operation.

3. Backups should be tested before they are trusted.

Audit Plugin Considerations

If you have the MariaDB Audit Plugin installed and are upgrading from MariaDB Enterprise Server 10.2 or 10.3, the audit plugin should be removed prior to the upgrade to prevent conflicts with the MariaDB Enterprise Audit Plugin, which MariaDB Enterprise Audit Plugin that is present in MariaDB Enterprise Server 10.4 or later. It can be removed by using the UNINSTALL SONAME statement:

```
UNINSTALL SONAME 'server_audit';
```

And if you load the plugin in a configuration file using the `plugin\_load\_add` option, then the option should also be removed. The MariaDB Enterprise Audit Plugin will automatically be installed after installing MariaDB Enterprise Server 10.4 or later.

Uninstall the Old Version

When upgrading to a new major release of MariaDB Enterprise Server, it is necessary to remove the existing installation of MariaDB Enterprise Server before installing the new version of MariaDB Enterprise Server. Otherwise, the package manager will refuse to install the new version of MariaDB Enterprise Server.

Stop the MariaDB Server Process

Before the old version can be uninstalled, we first need to stop the current MariaDB Server process.

1. Set the innodb_fast_shutdown system variable to 1:

```
SET GLOBAL innodb_fast_shutdown = 1;
```

2. Use XA RECOVER to confirm that there are no external XA transactions in a prepared state:

```
XA RECOVER;
```

Commit or roll back any open XA transactions before stopping the node for upgrade.

3. Stop the server process: For distributions that use `systemd` (most supported OSes), you can manage the server process using the `systemctl` command:

```
sudo systemctl stop mariadb
```

Uninstall via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Uninstall all of the MariaDB Enterprise Server packages. Note that a wildcard character is used to ensure that all MariaDB Enterprise Server packages are uninstalled:

```
sudo yum remove "MariaDB-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications:

2. Uninstall the Galera package as well. The name of the package depends on the specific version of MariaDB Enterprise Server. When upgrading from MariaDB Enterprise Server 10.4 or later, the package is called `galera-enterprise-4`:

```
sudo yum remove galera-enterprise-4
```

3. Before proceeding, verify that all MariaDB Enterprise Server packages are uninstalled. The following command should not return any results:

```
rpm --query --all | grep -i -E "mariadb|galera"
```

Uninstall via APT (Debian, Ubuntu)

1. Uninstall all of the MariaDB Enterprise Server packages. Note that a wildcard character is used to ensure that all MariaDB Enterprise Server packages are uninstalled:

```
sudo apt-get remove "mariadb-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well. The name of the package depends on the specific version of MariaDB Enterprise Server. When upgrading from MariaDB Enterprise Server 10.4 or later, the package is called galera-enterprise-4:

```
sudo apt remove galera-enterprise-4
```

3. Before proceeding, verify that all MariaDB Enterprise Server packages are uninstalled. The following command should not return any results:

```
apt list --installed | grep -i -E "mariadb|galera"
```

Uninstall via ZYpp (SLES)

1. Uninstall all of the MariaDB Enterprise Server packages. Note that a wildcard character is used to ensure that all MariaDB Enterprise Server packages are uninstalled:

```
sudo zypper remove "MariaDB-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well. The name of the package depends on the specific version of MariaDB Enterprise Server. When upgrading from MariaDB Enterprise Server 10.4 or later, the package is called galera-enterprise-4:

```
sudo zypper remove galera-enterprise-4
```

3. Before proceeding, verify that all MariaDB Enterprise Server packages are uninstalled. The following command should not return any results:

```
rpm --query --all | grep -i -E "mariadb|galera"
```

Install the New Version

MariaDB Corporation provides package repositories for YUM (RHEL, AlmaLinux, CentOS, Rocky Linux), APT (Debian, Ubuntu), and ZYpp (SLES).

Install via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Retrieve your Customer Download Token atand substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the YUM package repository. Installable versions of MariaDB Enterprise Server are `11.8, 11.4, 10.6, 10.5, 10.4, and 10.3.` Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

   To configure YUM package repositories:

```
sudo yum install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

> **Note:**
>
> *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.4"
```

3. Install MariaDB Enterprise Server and package dependencies:

```
sudo yum install MariaDB-server MariaDB-backup
```

4. Installation of additional packages may be required for some plugins.
5. Configure MariaDB. Installation only loads MariaDB Enterprise Server onto the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Install via APT (Debian, Ubuntu)

1. Retrieve your Customer Download Token atand substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the APT package repository. Installable versions of MariaDB Enterprise Server are `11.8 11.4, 10.6, 10.5, 10.4, and 10.3.` Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

   To configure APT package repositories:

```
sudo apt install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

> **Note:**
>
> *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.4"
```

```
sudo apt update
```

3. Install MariaDB Enterprise Server and package dependencies:

```
$ sudo apt install mariadb-server mariadb-backup galera-enterprise-4
```

4. Installation of additional packages may be required for some plugins. Configure MariaDB. Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Install via ZYpp (SLES)

1. Retrieve your Customer Download Token atand substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the ZYpp package repository. Installable versions of MariaDB Enterprise Server are `11.8, 11.4, 10.6, 10.5, 10.4, and 10.3.` Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

To configure ZYpp package repositories:

```
sudo zypper install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

Note:

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.4"
```

3. Install MariaDB Enterprise Server and package dependencies:

```
sudo zypper install MariaDB-server MariaDB-backup
```

4. Installation of additional packages may be required for some plugins.
5. Configure MariaDB. Installation only loads MariaDB Enterprise Server onto the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Configuration

For platforms that use YUM or ZYpp as a package manager: MariaDB Enterprise Server's packages bundle several configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mariadb-enterprise.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

If your version of any of these configuration files contained any custom edits, then the package manager may save your edited version with the `.rpmsave` extension during the upgrade process. If you want to continue using your version with the custom edits, then you may need to move it back.

For example, to move `server.cnf` back in place:

```
sudo mv /etc/my.cnf.d/server.cnf /etc/my.cnf.d/server.cnf.original
```

```
sudo mv /etc/my.cnf.d/server.cnf.rpmsave /etc/my.cnf.d/server.cnf
```

Starting the Server

MariaDB Enterprise Server includes configuration to `start, stop, restart, enable/disable` on boot, and check the status of the Server using the operating system's default process management system.

For distributions that use `systemd`, you can manage the Server process using the `systemctl` command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Upgrading the Data Directory

MariaDB Enterprise Server ships with a utility that can be used to identify and correct compatibility issues in the new version. After you upgrade your Server and start the server process, run this utility to upgrade the data directory.

The utility is called `mariadb-upgrade` in MariaDB Enterprise Server 10.4 and later:

```
sudo mariadb-upgrade
```

Testing

When MariaDB Enterprise Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the server using MariaDB Client using the `root@localhost` user account. MariaDB Client is called `mariadb` (ES 10.4 and later):

```
sudo mariadb
```

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

2. You can also verify the server version by checking the value of the version system variable with the SHOW GLOBAL STATUS statement:

```
SHOW GLOBAL VARIABLES LIKE 'version';
```

```
+---------------+----------------------------+
| Variable_name | Value                      |
+---------------+----------------------------+
| version       | 11.4.5-3-MariaDB-Enterprise |
+---------------+----------------------------+
```

3. You can also verify the server version by calling the VERSION() function:

```
SELECT VERSION();
```

```
+----------------------------+
| VERSION()                  |
+----------------------------+
| 11.4.5-3-MariaDB-Enterprise |
+----------------------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade MariaDB Enterprise Server from 11.4.X to 11.4.Y

These instructions detail a **minor release upgrade** with **MariaDB Enterprise Server 11.4** on a range of supported Operating Systems.

A minor release upgrade is a change from an earlier release of MariaDB Enterprise Server 11.4 to a later release in the same release series.

For example, it would be a minor release upgrade to upgrade from MariaDB Enterprise Server 11.4.4-2 to MariaDB Enterprise Server 11.4.5-3.

Data Backup

Occasionally, issues can be encountered during upgrades. These issues can even potentially corrupt the database's data files, preventing you from easily reverting to the old installation. Therefore, it is generally best to perform a backup prior to upgrading. If an issue is encountered during the upgrade, you can use the backup to restore your MariaDB Server database to the old version. If the upgrade finishes without issue, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, please see the Recovery Guide.

1. Take a full backup.

   On MariaDB Enterprise Server 10.4 and later:

   ```
   $ sudo mariadb-backup --backup \
         --user=mariadb-backup_user \
         --password=mariadb-backup_passwd \
         --target-dir=/data/backup/preupgrade_backup
   ```

   On MariaDB Enterprise Server 10.3 and earlier:

   ```
   $ sudo mariadb-backup --backup \
         --user=mariadb-backup_user \
         --password=mariadb-backup_passwd \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Confirm successful completion of the backup operation.

2. The backup must be prepared.

   On MariaDB Enterprise Server 10.4 and later:

   ```
   $ sudo mariadb-backup --prepare \
         --target-dir=/data/backup/preupgrade_backup
   ```

   On MariaDB Enterprise Server 10.3 and earlier:

   ```
   $ sudo mariadb-backup --prepare \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Confirm successful completion of the prepare operation.

3. Backups should be tested before they are trusted.

Stop the MariaDB Server Process

Before the new version can be installed, we first need to stop the current MariaDB Server process.

1. Set the innodb_fast_shutdown system variable to `1`:

   ```
   SET GLOBAL innodb_fast_shutdown = 1;
   ```

2. Use XA RECOVER to confirm that there are no external XA transactions in a prepared state:

   ```
   XA RECOVER;
   ```

   Commit or rollback any open XA transactions before stopping the node for upgrade.

3. Stop the server process:

   For distributions that use systemd (most supported OSes), you can manage the Server process using the `systemctl` command:

   ```
   $ sudo systemctl stop mariadb
   ```

Install the New Version

MariaDB Corporation provides package repositories for YUM (RHEL, AlmaLinux, CentOS, Rocky Linux), APT (Debian, Ubuntu), and ZYpp (SLES).

INSTALL VIA YUM

Install via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the YUM package repository. Installable versions of MariaDB Enterprise Server are `11.4`, `10.6`, `10.5`, `10.4`, and `10.3`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `11.4`.

   To configure YUM package repositories:

   ```
   $ sudo yum install curl
   ```

   ```
   $ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
   ```

   ```
   $ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
   ```

   ```
   $ chmod +x mariadb_es_repo_setup
   ```

   ```
   $ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --mariadb-server-version="11.4"
   ```

   1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Update MariaDB Enterprise Server and package dependencies:

```
$ sudo yum update "MariaDB-*" "galera*"
```

INSTALL VIA APT

Install via APT (Debian, Ubuntu)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the APT package repository.

   Installable versions of MariaDB Enterprise Server are `11.4`, `10.6`, `10.5`, `10.4`, and `10.3`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `11.4`.

   To configure APT package repositories:

   ```
   $ sudo apt install curl
   ```

   ```
   $ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
   ```

   ```
   $ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
   ```

   ```
   $ chmod +x mariadb_es_repo_setup
   ```

   ```
   $ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --mariadb-server-version="11.4"
   ```

   ```
   $ sudo apt update
   ```

   1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Update MariaDB Enterprise Server and package dependencies:

   ```
   $ sudo apt install --only-upgrade "mariadb-*" "galera*"
   ```

INSTALL VIA ZYPP

Install via ZYpp (SLES)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the ZYpp package repository.

   Installable versions of MariaDB Enterprise Server are `11.4`, `10.6`, `10.5`, `10.4`, and `10.3`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `11.4`.

   To configure ZYpp package repositories:

   ```
   $ sudo zypper install curl
   ```

   ```
   $ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
   ```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.4"
```

1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Update MariaDB Enterprise Server and package dependencies:

```
$ sudo zypper update "MariaDB-*" "galera*"
```

Configuration

For platforms that use YUM or ZYpp as a package manager:

MariaDB Enterprise Server's packages bundle several configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mariadb-enterprise.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

If your version of any of these configuration files contained any custom edits, then the package manager may save your edited version with the `.rpmsave` extension during the upgrade process. If you want to continue using your version with the custom edits, then you may need to move it back. For example, to move `server.cnf` back in place:

```
$ sudo mv /etc/my.cnf.d/server.cnf /etc/my.cnf.d/server.cnf.original
```

```
$ sudo mv /etc/my.cnf.d/server.cnf.rpmsave /etc/my.cnf.d/server.cnf
```

Starting the Server

MariaDB Enterprise Server includes configuration to start, stop, restart, enable/disable on boot, and check the status of the Server using the operating system default process management system.

For distributions that use systemd, you can manage the Server process using the `systemctl` command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Upgrading the Data Directory

MariaDB Enterprise Server ships with a utility that can be used to identify and correct compatibility issues in the new version. After you upgrade your Server and start the server process, run this utility to upgrade the data directory.

The utility is called mariadb-upgrade in MariaDB Enterprise Server 10.4 and later:

```
$ sudo mariadb-upgrade
```

And the utility is called mysql_upgrade in MariaDB Enterprise Server 10.3 and 10.2:

```
$ sudo mysql_upgrade
```

Testing

When MariaDB Enterprise Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the server using MariaDB Client using the `root@localhost` user account.

   MariaDB Client is called mariadb (ES10.4 and later) or `mysql` (ES10.3, ES10.2):

   ```
   $ sudo mariadb
   ```

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MariaDB connection id is 9
   Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

   Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

   MariaDB [(none)]>
   ```

2. You can also verify the server version by checking the value of the version system variable with the SHOW GLOBAL STATUS statement:

   ```
   SHOW GLOBAL VARIABLES LIKE 'version';
   ```

   ```
   +---------------+----------------------------+
   | Variable_name | Value                      |
   +---------------+----------------------------+
   | version       | 11.4.5-3-MariaDB-Enterprise |
   +---------------+----------------------------+
   ```

3. You can also verify the server version by calling the VERSION() function:

   ```
   SELECT VERSION();
   ```

   ```
   +----------------------------+
   | VERSION()                  |
   +----------------------------+
   | 11.4.5-3-MariaDB-Enterprise |
   +----------------------------+
   ```

© 2025 MariaDB. All rights reserved.

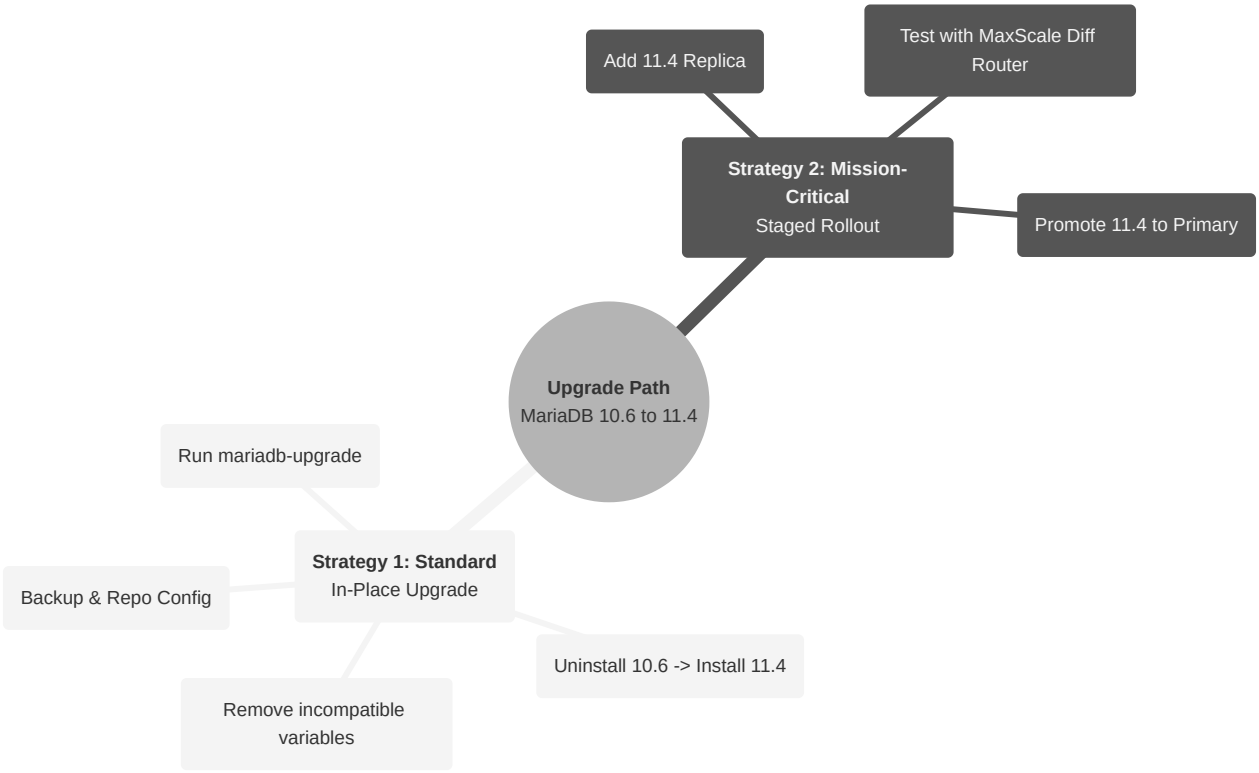
This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade MariaDB Enterprise Server from 10.6 to 11.4

The upgrade guide details the standard upgrade path from MariaDB Enterprise Server 10.6 to MariaDB Enterprise Server 11.4. The upgrade process involves significant changes, particularly to the query optimizer, which was rewritten in version 11.0. While performance is generally improved, query execution plans may change.

This guide provides two strategies:

1. Standard In-Place Upgrade: The direct procedure for upgrading an existing node.
2. Staged Rollout (Recommended for Mission-Critical Systems): A low-risk approach using replication and validation.



Prerequisite: Review What's New in MariaDB Enterprise Server 11.4.

Part 1: Standard In-Place Upgrade Process

This section details the standard procedure for performing an in-place upgrade. This involves replacing the server binaries while keeping the data directory intact.

> **Step 1**
>
> **Backup Database**
>
> Before performing any upgrade, it is critical to perform a full backup of your database. MariaDB Backup (`mariadb-backup`) is recommended for this purpose.

```
# Example backup command
sudo mariadb-backup --backup \
      --user=mariadb-backup_user \
      --password=mariadb-backup_passwd \
      --target-dir=/data/backup/preupgrade_backup
```

Step 2

Modify Repository Configuration

Update your system's package manager repository to point to MariaDB Enterprise Server 11.4. You will need to regenerate your repository configuration command using the MariaDB Customer Download Token.

APT (DEBIAN/UBUNTU)

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply --mariadb-server-version="11.4"
```

YUM (RHEL/CENTOS/ROCKY)

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply --mariadb-server-version="11.4"
```

ZYPP (SLES)

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply --mariadb-server-version="11.4"
```

Step 3

Stop MariaDB Service

Stop the running `mariadbd` process to release locks on the data files.

```
sudo systemctl stop mariadb
```

> **Warning:**
>
> Ensure `innodb_fast_shutdown` is set to `1` and check for open XA transactions before stopping.

Step 4

Uninstall Old Version

Remove the existing MariaDB 10.6 packages. This removes the binaries but preserves the configuration and data directory.

DEBIAN/UBUNTU

```
sudo apt-get remove "mariadb-*" galera-enterprise-4
```

RHEL/CENTOS/ROCKY

```
sudo yum remove "MariaDB-*" galera-enterprise-4
```

Step 5

Install New Version

Install the new MariaDB 11.4 packages.

DEBIAN/UBUNTU

```
sudo apt-get install mariadb-server mariadb-backup galera-enterprise-4
```

RHEL/CENTOS/ROCKY

```
sudo yum install MariaDB-server MariaDB-backup
```

Step 6

Update Configuration (Critical)

Before starting the server, you must update your option files (e.g., `my.cnf`) to remove unsupported parameters.

- Remove `innodb_defragment`: This variable and its associated options (e.g., `innodb_defragment_fill_factor`) have been removed.
- Remove `optimizer_adjust_secondary_key_costs`: This variable has no effect in 11.4 and must be removed.
- Check `innodb_flush_method`: This variable is deprecated in 11.4.

Step 7

Start MariaDB Service

Start the new `mariadbd` process.

```
sudo systemctl start mariadb
```

Step 8

Run mariadb-upgrade (Critical)

Execute `mariadb-upgrade` to check and update system tables to be compatible with the new version.

```
sudo mariadb-upgrade
```

Part 2: Alternative Strategy: Staged Rollout for Mission-Critical Systems

For environments sensitive to performance changes or requiring zero downtime, the following staged rollout strategy is recommended.

Step 1

Introduce Replica

Provision a new server with MariaDB Enterprise Server 11.4 and configure it as a Replica of your existing 10.6 Primary server.

1. **Perform Backup on Primary (10.6)**: Use `mariadb-backup` to create a consistent, non-blocking physical backup.

```
# On the 10.6 Primary
sudo mariadb-backup --backup \
    --user=backup_user \
    --password=backup_password \
    --target-dir=/data/backup/preupgrade_backup
```

2. **Prepare the Backup**: Prepare the backup to ensure consistency (this can be done on the Primary or the Replica).

```
sudo mariadb-backup --prepare \
    --target-dir=/data/backup/preupgrade_backup
```

3. **Restore on Replica (11.4)**: Transfer the backup to the new 11.4 server, stop the service, and restore.

```
# On the 11.4 Replica
sudo systemctl stop mariadb

# Ensure datadir is empty
sudo rm -rf /var/lib/mysql/*

# Restore backup
sudo mariadb-backup --copy-back \
    --target-dir=/data/backup/preupgrade_backup

# Fix permissions
sudo chown -R mysql:mysql /var/lib/mysql
```

4. **Start and Configure Replication:** Start the 11.4 server and configure it to replicate from the 10.6 Primary.

```
sudo systemctl start mariadb
```

Log in to the 11.4 Replica and enable replication. (Note: Ensure `server_id` is unique in `my.cnf`).

```
-- On the 11.4 Replica
-- Set the GTID position (found in the xtrabackup_binlog_info file from the backup)
SET GLOBAL gtid_slave_pos = '0-1-12345'; -- Replace with your actual GTID

CHANGE MASTER TO
    MASTER_HOST='10.6._primary_ip',
    MASTER_USER='replication_user',
    MASTER_PASSWORD='replication_password',
    MASTER_USE_GTID=slave_pos;

START SLAVE;
```

Step 2

Validate and Compare (Recommended)

To proactively catch performance regressions caused by the optimizer rewrite, use MaxScale with the Diff Router(Enterprise feature).

1. Configure MaxScale Ensure your MaxScale `maxscale.cnf` defines both the existing service (routing to 10.6) and the new 11.4 server definition.
2. Create the Diff Service Use `maxctrl` to create a “Diff” service that compares the output of your existing production server against the new 11.4 replica.

```
# Syntax: maxctrl call command diff create <NewDiffServiceName> <ExistingService> <ReferenceServer>
<TargetServer>

maxctrl call command diff create DiffService ReadWriteService Primary10.6 Replica11.4
```

3. Analyze Results MaxScale will generate JSON reports in its log directory (e.g., `/var/lib/maxscale/diff/`). Analyze these files to identify queries that return different results or have significantly higher latency on the 11.4 server.

Step 3

Shift Read Traffic

Gradually migrate read-only traffic to the 11.4 replica.

Update MaxScale Routing If using MaxScale, you can adjust the server weights or maintenance modes to direct read traffic.

```
# Put the 10.6 replicas into maintenance mode or lower their weight
maxctrl set server Primary10.6 maintenance
```

> **Note:**
>
> Alternatively, update your application connection strings to point to the 11.4 Replica’s IP for reporting modules.

Step 4

Promote to Primary

Once the 11.4 replica has been validated under load, promote it to be the Primary node.

1. Stop Replication on 11.4 Replica

```
-- On the 11.4 Replica
STOP SLAVE;
RESET SLAVE ALL;
```

2. Enable Writes on 11.4 New Primary

```
SET GLOBAL read_only=0;
```

3. Direct Traffic Update MaxScale or your application configuration to point to the new 11.4 Primary.

Step 5

Shift Application Traffic

Direct your application traffic to the new 11.4 Primary. Ensure you perform a `mariadb-upgrade` on the new Primary if it hasn't been run yet (though `mariadb-backup` restore usually preserves the old system tables, running upgrade is a safety step to ensure system tables are fully compatible with 11.4).

```
sudo mariadb-upgrade
```

Part 3: Critical Upgrade Information

Step 1

Optimizer Changes and Application Testing

The Query Optimizer was rewritten in version 11.0. While performance is generally better, query plans can change. It is vital to perform validation (as described in the Staged Rollout section) to catch regressions before production deployment. You should also run `ANALYZE TABLE` on major tables after upgrading to update statistics.

Step 2

Required Configuration Changes

The following variables must be addressed in your configuration files (`my.cnf`) to prevent startup failures or warnings:

- `optimizer_adjust_secondary_key_costs`: MUST REMOVE. This variable has no effect in 11.4 as features are integrated into the new cost model.
- `innodb_defragment`: MUST REMOVE. This variable has been removed.
- `des_encrypt` / `des_decrypt`: Functions have been removed.

Step 3

Backward Replication Compatibility

You can replicate from a MariaDB 11.4 Primary to a MariaDB 10.6 Replica (Backward Replication) under specific conditions:

- No new features specific to 11.4 (e.g., new JSON functions, `UUID` data type) are used in DML or DDL.
- The variable `binlog_alter_two_phase` must be set to `0` (default) on the 11.4 Primary.

Step 4

System-Versioned Tables Conversion

If using System-Versioned Tables, the `row_end` column format must be updated to support the extended `TIMESTAMP` range (up to year 2106).

MARIADB-DUMP

Use the new `--update-history` option with `mariadb-dump` to convert `row_end` values during export.

IN-PLACE

Use `ALTER TABLE` to convert the tables after upgrade.

```
ALTER TABLE my_table FORCE;
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade from MariaDB Community Server to MariaDB Enterprise Server 11.4

These instructions detail the **upgrade** from **MariaDB Community Server** to **MariaDB Enterprise Server 11.4** on a range of supported Operating Systems.

When MariaDB Enterprise Server is upgraded, the old version needs to be uninstalled, and the new version needs to be installed.

Data Backup

Occasionally, issues can be encountered during upgrades. These issues can even potentially corrupt the database's data files, preventing you from easily reverting to the old installation. Therefore, it is generally best to perform a backup prior to upgrading. If an issue is encountered during the upgrade, you can use the backup to restore your MariaDB Server database to the old version. If the upgrade finishes without issue, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, please see the Recovery Guide.

1. Take a full backup.

   On MariaDB Community Server 10.4 and later:

   ```
   $ sudo mariadb-backup --backup \
         --user=mariadb-backup_user \
         --password=mariadb-backup_passwd \
         --target-dir=/data/backup/preupgrade_backup
   ```

   On MariaDB Community Server 10.3 and earlier:

   ```
   $ sudo mariadb-backup --backup \
         --user=mariadb-backup_user \
         --password=mariadb-backup_passwd \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Confirm successful completion of the backup operation.

2. The backup must be prepared.

   On MariaDB Community Server 10.4 and later:

   ```
   $ sudo mariadb-backup --prepare \
         --target-dir=/data/backup/preupgrade_backup
   ```

   On MariaDB Community Server 10.3 and earlier:

   ```
   $ sudo mariadb-backup --prepare \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Confirm successful completion of the prepare operation.

3. Backups should be tested before they are trusted.

Audit Plugin Considerations

If you have the MariaDB Audit Plugin installed and if you are upgrading to MariaDB Enterprise Server 10.4 or later, then the audit plugin should be removed prior to the upgrade to prevent conflict with the MariaDB Enterprise Audit Plugin that is present in MariaDB Enterprise Server 10.4 or later.

It can be removed by using the UNINSTALL SONAME statement:

```
UNINSTALL SONAME 'server_audit';
```

And if you load the plugin in a configuration file using the `plugin_load_add` option, then the option should also be removed.

The MariaDB Enterprise Audit Plugin will automatically be installed after installing MariaDB Enterprise Server 10.4 or later.

Convert InnoDB Row Format

MariaDB Enterprise Server 11.4 changes the `COMPRESSED` row format to read-only. Before upgrading, modify any compressed InnoDB tables to use the `DYNAMIC` row format.

1. Use the `information\_schema.INNODB\_SYS\_TABLES` to identify any InnoDB tables that use the `COMPRESSED` row format:

```
SELECT NAME, ROW_FORMAT
FROM information_schema.INNODB_SYS_TABLES
WHERE NAME NOT LIKE 'SYS_%'
   AND ROW_FORMAT = 'COMPRESSED';
```

2. Execute an ALTER TABLE statement for each table, changing its row format from `COMPRESSED` to `DYNAMIC`:

```
ALTER TABLE accounts.hq_sales
ROW_FORMAT = DYNAMIC
PAGE_COMPRESSED = 1;
```

Uninstall the Old Version

When upgrading to MariaDB Enterprise Server, it is necessary to remove the existing installation of MariaDB Community Server, before installing MariaDB Enterprise Server. Otherwise, the package manager will refuse to install MariaDB Enterprise Server.

Stop the MariaDB Server Process

Before the old version can be uninstalled, we first need to stop the current MariaDB Server process.

1. Set the innodb_fast_shutdown system variable to `1`:

```
SET GLOBAL innodb_fast_shutdown = 1;
```

2. Use XA RECOVER to confirm that there are no external XA transactions in a prepared state:

```
XA RECOVER;
```

   Commit or rollback any open XA transactions before stopping the node for upgrade.

3. Stop the server process:

   For distributions that use systemd (most supported OSes), you can manage the Server process using the `systemctl` command:

```
$ sudo systemctl stop mariadb
```

UNINSTALL VIA YUM

Uninstall via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Uninstall all of the MariaDB Community Server packages. Note that a wildcard character is used to ensure that all MariaDB Community Server packages are uninstalled:

```
$ sudo yum remove "MariaDB-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications:

2. Uninstall the Galera package as well.

   The name of the package depends on the specific version of MariaDB Community Server.

   When upgrading from MariaDB Community Server 10.4 or later, the package is called `galera-4`:

   ```
   $ sudo yum remove galera-4
   ```

   When upgrading from MariaDB Community Server 10.3 or earlier, the package is called `galera`:

   ```
   $ sudo yum remove galera
   ```

3. Before proceeding, verify that all MariaDB Community Server packages are uninstalled. The following command should not return any results:

   ```
   $ rpm --query --all | grep -i -E "mariadb|galera"
   ```

UNINSTALL VIA APT

Uninstall via APT (Debian, Ubuntu)

1. Uninstall all of the MariaDB Community Server packages. Note that a wildcard character is used to ensure that all MariaDB Community Server packages are uninstalled:

   ```
   $ sudo apt-get remove "mariadb-*"
   ```

   Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well.

   The name of the package depends on the specific version of MariaDB Community Server.

   When upgrading from MariaDB Community Server 10.4 or later, the package is called `galera-4`:

   ```
   $ sudo apt remove galera-4
   ```

   When upgrading from MariaDB Community Server 10.3 or earlier, the package is called `galera-3`:

   ```
   $ sudo apt remove galera-3
   ```

3. Before proceeding, verify that all MariaDB Community Server packages are uninstalled. The following command should not return any results:

   ```
   $ apt list --installed | grep -i -E "mariadb|galera"
   ```

UNINSTALL VIA ZYPP

Uninstall via ZYpp (SLES)

1. Uninstall all of the MariaDB Community Server packages. Note that a wildcard character is used to ensure that all MariaDB Community Server packages are uninstalled:

   ```
   $ sudo zypper remove "MariaDB-*"
   ```

   Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well.

   The name of the package depends on the specific version of MariaDB Community Server.

   When upgrading from MariaDB Community Server 10.4 or later, the package is called `galera-4`:

   ```
   $ sudo zypper remove galera-4
   ```

   When upgrading from MariaDB Community Server 10.3 or earlier, the package is called `galera`:

   ```
   $ sudo zypper remove galera
   ```

3. Before proceeding, verify that all MariaDB Community Server packages are uninstalled. The following command should not return any results:

   ```
   $ rpm --query --all | grep -i -E "mariadb|galera"
   ```

Install the New Version

MariaDB Corporation provides package repositories for YUM (RHEL, AlmaLinux, CentOS, Rocky Linux), APT (Debian, Ubuntu), and ZYpp (SLES).

INSTALL VIA YUM

Install via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the YUM package repository. Installable versions of MariaDB Enterprise Server are `11.4`, `10.6`, `10.5`, `10.4`, and `10.3`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `11.4`.

   To configure YUM package repositories:

   ```
   $ sudo yum install curl
   ```

   ```
   $ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
   ```

   ```
   $ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
   ```

   ```
   $ chmod +x mariadb_es_repo_setup
   ```

   ```
   $ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --mariadb-server-version="11.4"
   ```

   1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Install MariaDB Enterprise Server and package dependencies:

   ```
   $ sudo yum install MariaDB-server MariaDB-backup
   ```

   Installation of additional packages may be required for some plugins.

4. Configure MariaDB.

   Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

INSTALL VIA APT

Install via APT (Debian, Ubuntu)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the APT package repository.

   Installable versions of MariaDB Enterprise Server are `11.4`, `10.6`, `10.5`, `10.4`, and `10.3`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `11.4`.

   To configure APT package repositories:

   ```
   $ sudo apt install curl
   ```

   ```
   $ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
   ```

   ```
   $ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
   ```

   ```
   $ chmod +x mariadb_es_repo_setup
   ```

   ```
   $ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --mariadb-server-version="11.4"
   ```

   ```
   $ sudo apt update
   ```

   1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Install MariaDB Enterprise Server and package dependencies:

   ```
   $ sudo apt install mariadb-server mariadb-backup
   ```

   Installation of additional packages may be required for some plugins.

4. Configure MariaDB.

   Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

INSTALL VIA ZYPP

Install via ZYpp (SLES)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the ZYpp package repository.

   Installable versions of MariaDB Enterprise Server are `11.4`, `10.6`, `10.5`, `10.4`, and `10.3`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions

reference `11.4`.

To configure ZYpp package repositories:

```
$ sudo zypper install curl
```

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="11.4"
```

1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Install MariaDB Enterprise Server and package dependencies:

   ```
   $ sudo zypper install MariaDB-server MariaDB-backup
   ```

   Installation of additional packages may be required for some plugins.

4. Configure MariaDB.

   Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Configuration

For platforms that use YUM or ZYpp as a package manager:

MariaDB Community Server's packages bundle several configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

If your version of any of these configuration files contained any custom edits, then the package manager may save your edited version with the `.rpmsave` extension during the upgrade process. If you want to continue using your version with the custom edits, then you may need to move it back. For example, to move `server.cnf` back in place:

```
$ sudo mv /etc/my.cnf.d/server.cnf /etc/my.cnf.d/server.cnf.original
```

```
$ sudo mv /etc/my.cnf.d/server.cnf.rpmsave /etc/my.cnf.d/server.cnf
```

Starting the Server

MariaDB Enterprise Server includes configuration to start, stop, restart, enable/disable on boot, and check the status of the Server using the operating system default process management system.

For distributions that use systemd, you can manage the Server process using the `systemctl` command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Upgrading the Data Directory

MariaDB Enterprise Server ships with a utility that can be used to identify and correct compatibility issues in the new version. After you upgrade your Server and start the server process, run this utility to upgrade the data directory.

The utility is called mariadb-upgrade in MariaDB Enterprise Server 10.4 and later:

```
$ sudo mariadb-upgrade
```

And the utility is called `mysql\_upgrade` in MariaDB Enterprise Server 10.3 and 10.2:

```
$ sudo mysql_upgrade
```

Testing

When MariaDB Enterprise Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the server using MariaDB Client using the `root@localhost` user account.

   MariaDB Client is called mariadb (ES10.4 and later) or `mysql` (ES10.3 and earlier):

   ```
   $ sudo mariadb
   ```

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MariaDB connection id is 9
   Server version: 11.4.5-3-MariaDB MariaDB Server

   Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

   MariaDB [(none)]>
   ```

2. You can also verify the server version by checking the value of the version system variable with the SHOW GLOBAL STATUS statement:

   ```
   SHOW GLOBAL VARIABLES LIKE 'version';
   ```

   ```
   +---------------+-----------------+
   | Variable_name | Value           |
   +---------------+-----------------+
   | version       | 11.4.5-3-MariaDB |
   +---------------+-----------------+
   ```

3. You can also verify the server version by calling the VERSION() function:

```
SELECT VERSION();
```

```
+------------------+
| VERSION()        |
+------------------+
| 11.4.5-3-MariaDB |
+------------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Server 10.6

See Upgrade to MariaDB Enterprise Server 10.6.

Upgrade documentation for MariaDB Enterprise Server 10.6, featuring Atomic DDL support, JSON_TABLE function, improved Oracle compatibility modes, and the removal of older storage engines.

See Upgrade MariaDB Enterprise Server from 10.6.X to 10.6.Y.

Upgrade documentation for MariaDB Enterprise Server 10.6, featuring Atomic DDL support, JSON_TABLE function, improved Oracle compatibility modes, and the removal of older storage engines.

See Upgrade from MariaDB Community Server to MariaDB Enterprise Server 10.6.

Upgrade documentation for MariaDB Enterprise Server 10.6, featuring Atomic DDL support, JSON_TABLE function, improved Oracle compatibility modes, and the removal of older storage engines.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade to MariaDB Enterprise Server 10.6

Overview

These instructions detail the **upgrade** from a previous version of **MariaDB Enterprise Server** to **MariaDB Enterprise Server 10.6** on a range of supported Operating Systems.

When MariaDB Enterprise Server is upgraded, the old version needs to be uninstalled, and the new version needs to be installed.

See What's New in MariaDB Enterprise Server 10.6.

Incompatible Changes

Reserved Words

- **`OFFSET`**: This is now a reserved word. If you have tables or columns named `OFFSET`, you must quote them (e.g., `` `OFFSET` ``) in your SQL queries.

Character Sets

- **`utf8` changes**: The `utf8` character set is now an alias for `utf8mb3` (3-byte) by default. If you need 4-byte support (for emojis), ensure your application explicitly uses `utf8mb4`.

Removed Variables

The following variables have been removed. Remove them from `my.cnf` before upgrading:

- **`innodb_checksum_algorithm`**: The `innodb` and `none` options are removed. Use `crc32` or `full_crc32`.
- **`innodb_log_checksums`**: Removed.

- `innodb_undo_logs`: Removed.

Storage Engines

- **TokuDB**: The TokuDB storage engine was disabled in 10.5 and removed in 10.6. Migrating TokuDB tables to InnoDB or MyRocks is required before upgrading.

Data Backup

Occasionally, issues can be encountered during upgrades. These issues can even potentially corrupt the database's data files, preventing you from easily reverting to the old installation. Therefore, it is generally best to perform a backup before upgrading. If an issue is encountered during the upgrade, you can use the backup to restore your MariaDB Server database to the old version. If the upgrade finishes without issue, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, please see the Recovery Guide.

1. Take a full backup. On MariaDB Enterprise Server 10.6 and later:

```
sudo mariadb-backup --backup \
      --user=mariadb-backup_user \
      --password=mariadb-backup_passwd \
      --target-dir=/data/backup/preupgrade_backup
```

Confirm successful completion of the backup operation.

2. The backup must be prepared. On the MariaDB Enterprise Server 10.4 and later:

```
sudo mariadb-backup --prepare \
      --target-dir=/data/backup/preupgrade_backup
```

Confirm successful completion of the prepare operation.

3. Backups should be tested before they are trusted.

Audit Plugin Considerations

If you have the MariaDB Audit Plugin installed and are upgrading from MariaDB Enterprise Server 10.2 or 10.3, the audit plugin should be removed prior to the upgrade to prevent conflicts with the MariaDB Enterprise Audit Plugin, which MariaDB Enterprise Audit Plugin that is present in MariaDB Enterprise Server 10.4 or later. It can be removed by using the UNINSTALL SONAME statement:

```
UNINSTALL SONAME 'server_audit';
```

And if you load the plugin in a configuration file using the `plugin\_load\_add` option, then the option should also be removed. The MariaDB Enterprise Audit Plugin will automatically be installed after installing MariaDB Enterprise Server 10.4 or later.

Uninstall the Old Version

When upgrading to a new major release of MariaDB Enterprise Server, it is necessary to remove the existing installation of MariaDB Enterprise Server before installing the new version of MariaDB Enterprise Server. Otherwise, the package manager will refuse to install the new version of MariaDB Enterprise Server.

Stop the MariaDB Server Process

Before the old version can be uninstalled, we first need to stop the current MariaDB Server process.

1. Set the innodb_fast_shutdown system variable to 1:

```
SET GLOBAL innodb_fast_shutdown = 1;
```

2. Use XA RECOVER to confirm that there are no external XA transactions in a prepared state:

```
XA RECOVER;
```

Commit or roll back any open XA transactions before stopping the node for upgrade.

3. Stop the server process: For distributions that use `systemd` (most supported OSes), you can manage the server process using the `systemctl` command:

```
sudo systemctl stop mariadb
```

Uninstall via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Uninstall all of the MariaDB Enterprise Server packages. Note that a wildcard character is used to ensure that all MariaDB Enterprise Server packages are uninstalled:

```
sudo yum remove "MariaDB-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications:

2. Uninstall the Galera package as well. The name of the package depends on the specific version of MariaDB Enterprise Server. When upgrading from MariaDB Enterprise Server 10.4 or later, the package is called `galera-enterprise-4`:

```
sudo yum remove galera-enterprise-4
```

3. Before proceeding, verify that all MariaDB Enterprise Server packages are uninstalled. The following command should not return any results:

```
rpm --query --all | grep -i -E "mariadb|galera"
```

Uninstall via APT (Debian, Ubuntu)

1. Uninstall all of the MariaDB Enterprise Server packages. Note that a wildcard character is used to ensure that all MariaDB Enterprise Server packages are uninstalled:

```
sudo apt-get remove "mariadb-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well. The name of the package depends on the specific version of MariaDB Enterprise Server. When upgrading from MariaDB Enterprise Server 10.4 or later, the package is called galera-enterprise-4:

```
sudo apt remove galera-enterprise-4
```

3. Before proceeding, verify that all MariaDB Enterprise Server packages are uninstalled. The following command should not return any results:

```
apt list --installed | grep -i -E "mariadb|galera"
```

Uninstall via ZYpp (SLES)

1. Uninstall all of the MariaDB Enterprise Server packages. Note that a wildcard character is used to ensure that all MariaDB Enterprise Server packages are uninstalled:

```
sudo zypper remove "MariaDB-*"
```

Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well. The name of the package depends on the specific version of MariaDB Enterprise Server. When upgrading from MariaDB Enterprise Server 10.4 or later, the package is called galera-enterprise-4:

```
sudo zypper remove galera-enterprise-4
```

3. Before proceeding, verify that all MariaDB Enterprise Server packages are uninstalled. The following command should not return any results:

```
rpm --query --all | grep -i -E "mariadb|galera"
```

Install the New Version

MariaDB Corporation provides package repositories for YUM (RHEL, AlmaLinux, CentOS, Rocky Linux), APT (Debian, Ubuntu), and ZYpp (SLES).

Install via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Retrieve your Customer Download Token atand substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the YUM package repository. Installable versions of MariaDB Enterprise Server are `11.8, 11.4, 10.6, 10.5, 10.4, and 10.3.` Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

   To configure YUM package repositories:

```
sudo yum install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

> **Note:**
>
> *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="10.6"
```

3. Install MariaDB Enterprise Server and package dependencies:

```
sudo yum install MariaDB-server MariaDB-backup
```

4. Installation of additional packages may be required for some plugins.
5. Configure MariaDB. Installation only loads MariaDB Enterprise Server onto the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Install via APT (Debian, Ubuntu)

1. Retrieve your Customer Download Token atand substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the APT package repository. Installable versions of MariaDB Enterprise Server are `11.8 11.4, 10.6, 10.5, 10.4, and 10.3.` Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

To configure APT package repositories:

```
sudo apt install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

Note:

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="10.6"
```

```
sudo apt update
```

3. Install MariaDB Enterprise Server and package dependencies:

```
$ sudo apt install mariadb-server mariadb-backup galera-enterprise-4
```

4. Installation of additional packages may be required for some plugins. Configure MariaDB. Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Install via ZYpp (SLES)

1. Retrieve your Customer Download Token atand substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the ZYpp package repository. Installable versions of MariaDB Enterprise Server are `11.8, 11.4, 10.6, 10.5, 10.4, and 10.3.` Pass the version to install using the `--mariadb-server-version` flag to `mariadb_es_repo_setup`. The following directions reference 11.8.

To configure ZYpp package repositories:

```
sudo zypper install curl
```

```
curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

Note:

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

```
chmod +x mariadb_es_repo_setup
```

```
sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="10.6"
```

3. Install MariaDB Enterprise Server and package dependencies:

```
sudo zypper install MariaDB-server MariaDB-backup
```

4. Installation of additional packages may be required for some plugins.
5. Configure MariaDB. Installation only loads MariaDB Enterprise Server onto the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Configuration

For platforms that use YUM or ZYpp as a package manager: MariaDB Enterprise Server's packages bundle several configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mariadb-enterprise.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

If your version of any of these configuration files contained any custom edits, then the package manager may save your edited version with the `.rpmsave` extension during the upgrade process. If you want to continue using your version with the custom edits, then you may need to move it back.

For example, to move `server.cnf` back in place:

```
sudo mv /etc/my.cnf.d/server.cnf /etc/my.cnf.d/server.cnf.original
```

```
sudo mv /etc/my.cnf.d/server.cnf.rpmsave /etc/my.cnf.d/server.cnf
```

Starting the Server

MariaDB Enterprise Server includes configuration to `start, stop, restart, enable/disable` on boot, and check the status of the Server using the operating system's default process management system.

For distributions that use `systemd`, you can manage the Server process using the `systemctl` command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Upgrading the Data Directory

MariaDB Enterprise Server ships with a utility that can be used to identify and correct compatibility issues in the new version. After you upgrade your Server and start the server process, run this utility to upgrade the data directory.

The utility is called `mariadb-upgrade` in MariaDB Enterprise Server 10.4 and later:

```
sudo mariadb-upgrade
```

Testing

When MariaDB Enterprise Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the server using MariaDB Client using the `root@localhost` user account. MariaDB Client is called `mariadb` (ES 10.4 and later):

```
sudo mariadb
```

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.6.21-17-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

2. You can also verify the server version by checking the value of the version system variable with the SHOW GLOBAL STATUS statement:

```
SHOW GLOBAL VARIABLES LIKE 'version';
```

```
+---------------+----------------------------+
| Variable_name | Value                      |
+---------------+----------------------------+
| version       | 10.6.21-17-MariaDB-Enterprise |
+---------------+----------------------------+
```

3. You can also verify the server version by calling the VERSION() function:

```
SELECT VERSION();
```

```
+----------------------------+
| VERSION()                  |
+----------------------------+
| 10.6.21-17-MariaDB-Enterprise |
+----------------------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade MariaDB Enterprise Server from 10.6.X to 10.6.Y

These instructions detail a **minor release upgrade** with **MariaDB Enterprise Server 10.6** on a range of supported Operating Systems.

A minor release upgrade is a change from an earlier release of MariaDB Enterprise Server 10.6 to a later release in the same release series.

For example, it would be a minor release upgrade to upgrade from MariaDB Enterprise Server 10.6.20-16 to MariaDB Enterprise Server 10.6.21-17.

Data Backup

Occasionally, issues can be encountered during upgrades. These issues can even potentially corrupt the database's data files, preventing you from easily reverting to the old installation. Therefore, it is generally best to perform a backup prior to upgrading. If an issue is encountered during the upgrade, you can use the backup to restore your MariaDB Server database to the old version. If the upgrade finishes without issue, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, please see the Recovery Guide.

1. Take a full backup.

   On MariaDB Enterprise Server 10.4 and later:

   ```
   $ sudo mariadb-backup --backup \
         --user=mariadb-backup_user \
         --password=mariadb-backup_passwd \
         --target-dir=/data/backup/preupgrade_backup
   ```

   On MariaDB Enterprise Server 10.3 and earlier:

   ```
   $ sudo mariadb-backup --backup \
         --user=mariadb-backup_user \
         --password=mariadb-backup_passwd \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Confirm successful completion of the backup operation.

2. The backup must be prepared.

   On MariaDB Enterprise Server 10.4 and later:

   ```
   $ sudo mariadb-backup --prepare \
         --target-dir=/data/backup/preupgrade_backup
   ```

   On MariaDB Enterprise Server 10.3 and earlier:

   ```
   $ sudo mariadb-backup --prepare \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Confirm successful completion of the prepare operation.

3. Backups should be tested before they are trusted.

Stop the MariaDB Server Process

Before the new version can be installed, we first need to stop the current MariaDB Server process.

1. Set the innodb_fast_shutdown system variable to `1`:

   ```
   SET GLOBAL innodb_fast_shutdown = 1;
   ```

2. Use XA RECOVER to confirm that there are no external XA transactions in a prepared state:

   ```
   XA RECOVER;
   ```

   Commit or rollback any open XA transactions before stopping the node for upgrade.

3. Stop the server process:

   For distributions that use `systemd` (most supported OSes), you can manage the Server process using the `systemctl` command:

   ```
   $ sudo systemctl stop mariadb
   ```

Install the New Version

MariaDB Corporation provides package repositories for YUM (RHEL, AlmaLinux, CentOS, Rocky Linux), APT (Debian, Ubuntu), and ZYpp (SLES).

> **INSTALL VIA YUM**
>
> **Install via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)**
>
> 1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
> 2. Configure the YUM package repository. Installable versions of MariaDB Enterprise Server are `11.4`, `10.6`, `10.5`, `10.4`, and `10.3`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `10.6`.
>
>   To configure YUM package repositories:
>
>    ```
>    $ sudo yum install curl
>    ```
>
>    ```
>    $ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
>    ```
>
>    ```
>    $ echo "${checksum}  mariadb_es_repo_setup" \
>        | sha256sum -c -
>    ```
>
>    ```
>    $ chmod +x mariadb_es_repo_setup
>    ```
>
>    ```
>    $ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
>       --mariadb-server-version="10.6"
>    ```
>
>   1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*
>
> 3. Update MariaDB Enterprise Server and package dependencies:

```
$ sudo yum update "MariaDB-*" "galera*"
```

INSTALL VIA APT

Install via APT (Debian, Ubuntu)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the APT package repository.

   Installable versions of MariaDB Enterprise Server are `11.4`, `10.6`, `10.5`, `10.4`, and `10.3`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `10.6`.

   To configure APT package repositories:

   ```
   $ sudo apt install curl
   ```

   ```
   $ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
   ```

   ```
   $ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
   ```

   ```
   $ chmod +x mariadb_es_repo_setup
   ```

   ```
   $ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --mariadb-server-version="10.6"
   ```

   ```
   $ sudo apt update
   ```

   1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Update MariaDB Enterprise Server and package dependencies:

   ```
   $ sudo apt install --only-upgrade "mariadb-*" "galera*"
   ```

INSTALL VIA ZYPP

Install via ZYpp (SLES)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the ZYpp package repository.

   Installable versions of MariaDB Enterprise Server are `11.4`, `10.6`, `10.5`, `10.4`, and `10.3`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `11.4`.

   To configure ZYpp package repositories:

   ```
   $ sudo zypper install curl
   ```

   ```
   $ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
   ```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="10.6"
```

1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Update MariaDB Enterprise Server and package dependencies:

```
$ sudo zypper update "MariaDB-*" "galera*"
```

Configuration

For platforms that use YUM or ZYpp as a package manager:

MariaDB Enterprise Server's packages bundle several configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mariadb-enterprise.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

If your version of any of these configuration files contained any custom edits, then the package manager may save your edited version with the `.rpmsave` extension during the upgrade process. If you want to continue using your version with the custom edits, then you may need to move it back. For example, to move `server.cnf` back in place:

```
$ sudo mv /etc/my.cnf.d/server.cnf /etc/my.cnf.d/server.cnf.original
```

```
$ sudo mv /etc/my.cnf.d/server.cnf.rpmsave /etc/my.cnf.d/server.cnf
```

Starting the Server

MariaDB Enterprise Server includes configuration to start, stop, restart, enable/disable on boot, and check the status of the Server using the operating system default process management system.

For distributions that use systemd, you can manage the Server process using the `systemctl` command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Upgrading the Data Directory

MariaDB Enterprise Server ships with a utility that can be used to identify and correct compatibility issues in the new version. After you upgrade your Server and start the server process, run this utility to upgrade the data directory.

The utility is called mariadb-upgrade in MariaDB Enterprise Server 10.4 and later:

```
$ sudo mariadb-upgrade
```

And the utility is called mysql_upgrade in MariaDB Enterprise Server 10.3 and 10.2:

```
$ sudo mysql_upgrade
```

Testing

When MariaDB Enterprise Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the server using MariaDB Client using the `root@localhost` user account.

   MariaDB Client is called mariadb (ES10.4 and later) or `mysql` (ES10.3, ES10.2):

   ```
   $ sudo mariadb
   ```

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MariaDB connection id is 9
   Server version: 10.6.21-17-MariaDB-Enterprise MariaDB Enterprise Server

   Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

   MariaDB [(none)]>
   ```

2. You can also verify the server version by checking the value of the version system variable with the SHOW GLOBAL STATUS statement:

   ```
   SHOW GLOBAL VARIABLES LIKE 'version';
   ```

   ```
   +---------------+-------------------------------+
   | Variable_name | Value                         |
   +---------------+-------------------------------+
   | version       | 10.6.21-17-MariaDB-Enterprise |
   +---------------+-------------------------------+
   ```

3. You can also verify the server version by calling the VERSION() function:

   ```
   SELECT VERSION();
   ```

   ```
   +-------------------------------+
   | VERSION()                     |
   +-------------------------------+
   | 10.6.21-17-MariaDB-Enterprise |
   +-------------------------------+
   ```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Upgrade from MariaDB Community Server to MariaDB Enterprise Server 10.6

These instructions detail the **upgrade** from **MariaDB Community Server 10.6** to **MariaDB Enterprise Server 10.6** on a range of supported Operating Systems.

> **Warning:**
>
> MariaDB Community Server 10.6 reached its end of life in July 2026 and no longer receives maintenance or security releases. Upgrading to MariaDB Enterprise Server 10.6 lets you stay on the 10.6 release series while moving to a maintained and supported server.

> **Note:**
>
> This page describes the minimal-change path: switching products while staying on the 10.6 release series. The recommended path is to upgrade directly to MariaDB Enterprise Server 11.8, the newest long-term release series, for a longer support runway — see Upgrade from MariaDB Community Server 10.6 to MariaDB Enterprise Server 11.8.

When switching from MariaDB Community Server to MariaDB Enterprise Server, the old version needs to be uninstalled, and the new version needs to be installed.

MariaDB Enterprise Server 10.6 is not identical to MariaDB Community Server 10.6:

- MariaDB Enterprise Server includes features and fixes backported from later MariaDB versions. For details, see What's New in MariaDB Enterprise Server 10.6.
- MariaDB Enterprise Server does not include plugins that are not supported for enterprise use. Before upgrading, review Differences in Available Plugins for Enterprise Server to confirm that the plugins you rely on are available.

Data Backup

Occasionally, issues can be encountered during upgrades. These issues can even potentially corrupt the database's data files, preventing you from easily reverting to the old installation. Therefore, it is generally best to perform a backup prior to upgrading. If an issue is encountered during the upgrade, you can use the backup to restore your MariaDB Server database to the old version. If the upgrade finishes without issue, then the backup can be deleted.

The instructions below show how to perform a backup using MariaDB Backup. For more information about backing up and restoring the database, please see the Recovery Guide.

1. Take a full backup:

   ```
   $ sudo mariadb-backup --backup \
         --user=mariadb-backup_user \
         --password=mariadb-backup_passwd \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Confirm successful completion of the backup operation.

2. Prepare the backup so that it is ready for immediate restoration if required:

   ```
   $ sudo mariadb-backup --prepare \
         --target-dir=/data/backup/preupgrade_backup
   ```

   Confirm successful completion of the prepare operation.

3. Verify that the backup is recoverable by restoring it to a non-production environment before proceeding with the upgrade.

Audit Plugin Considerations

If you have the MariaDB Audit Plugin installed, then the audit plugin should be removed prior to the upgrade to prevent conflict with the MariaDB Enterprise Audit Plugin that is included in MariaDB Enterprise Server 10.6.

The two plugins differ mainly in how audit logging can be filtered:

Capability	MariaDB Audit Plugin	MariaDB Enterprise Audit
Event selection	Global event types (connect, query, table)	Event filters defined per audit filter
Per-user control	Include/exclude user lists	Default and named per-user audit filters
Per-object control	Not available	Object filters for specific databases and tables
Configuration auditing	Not available	Changes to the audit configuration are logged

It can be removed by using the UNINSTALL SONAME statement:

```
UNINSTALL SONAME 'server_audit';
```

And if you load the plugin in a configuration file using the plugin_load_add option — a mariadbd startup option that loads a plugin library when the server starts — then the option should also be removed.

The MariaDB Enterprise Audit Plugin will automatically be installed after installing MariaDB Enterprise Server 10.6.

Uninstall the Old Version

When upgrading to MariaDB Enterprise Server, it is necessary to remove the existing installation of MariaDB Community Server, before installing MariaDB Enterprise Server. Otherwise, the package manager will refuse to install MariaDB Enterprise Server.

Stop the MariaDB Server Process

Before the old version can be uninstalled, we first need to stop the current MariaDB Server process.

1. Set the innodb_fast_shutdown system variable to `1`:

   ```
   SET GLOBAL innodb_fast_shutdown = 1;
   ```

2. Use XA RECOVER to confirm that there are no external XA transactions in a prepared state:

   ```
   XA RECOVER;
   ```

   Commit or rollback any open XA transactions before stopping the node for upgrade.

3. Stop the server process:

   For distributions that use systemd (most supported OSes), you can manage the Server process using the `systemctl` command:

   ```
   $ sudo systemctl stop mariadb
   ```

UNINSTALL VIA YUM

Uninstall via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Uninstall all of the MariaDB Community Server packages. Note that a wildcard character is used to ensure that all MariaDB Community Server packages are uninstalled:

   ```
   $ sudo yum remove "MariaDB-*"
   ```

   Be sure to check that this wildcard does not unintentionally refer to any of your custom applications:

2. Uninstall the Galera package as well.

   For MariaDB Community Server 10.6, the package is called `galera-4`:

```
$ sudo yum remove galera-4
```

3. Before proceeding, verify that all MariaDB Community Server packages are uninstalled. The following command should not return any results:

```
$ rpm --query --all | grep -i -E "mariadb|galera"
```

UNINSTALL VIA APT

Uninstall via APT (Debian, Ubuntu)

1. Uninstall all of the MariaDB Community Server packages. Note that a wildcard character is used to ensure that all MariaDB Community Server packages are uninstalled:

```
$ sudo apt-get remove "mariadb-*"
```

   Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well.

   For MariaDB Community Server 10.6, the package is called `galera-4`:

```
$ sudo apt remove galera-4
```

3. Before proceeding, verify that all MariaDB Community Server packages are uninstalled. The following command should not return any results:

```
$ apt list --installed | grep -i -E "mariadb|galera"
```

UNINSTALL VIA ZYPP

Uninstall via ZYpp (SLES)

1. Uninstall all of the MariaDB Community Server packages. Note that a wildcard character is used to ensure that all MariaDB Community Server packages are uninstalled:

```
$ sudo zypper remove "MariaDB-*"
```

   Be sure to check that this wildcard does not unintentionally refer to any of your custom applications.

2. Uninstall the Galera package as well.

   For MariaDB Community Server 10.6, the package is called `galera-4`:

```
$ sudo zypper remove galera-4
```

3. Before proceeding, verify that all MariaDB Community Server packages are uninstalled. The following command should not return any results:

```
$ rpm --query --all | grep -i -E "mariadb|galera"
```

Install the New Version

MariaDB Corporation provides package repositories for YUM (RHEL, AlmaLinux, CentOS, Rocky Linux), APT (Debian, Ubuntu), and ZYpp (SLES).

Note:

If the server is a node in a Galera Cluster, also install the Enterprise Galera provider package, `galera-enterprise-4`, which replaces MariaDB Community Server's `galera-4` package. Cluster nodes are also upgraded one node at a time — see Upgrading Galera Cluster for the cluster-specific procedure.

INSTALL VIA YUM

Install via YUM (RHEL, AlmaLinux, CentOS, Rocky Linux)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the YUM package repository. Installable versions of MariaDB Enterprise Server are `10.6`, `11.4`, and `11.8`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `10.6`.

   To configure YUM package repositories:

   ```
   $ sudo yum install curl
   ```

   ```
   $ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
   ```

   ```
   $ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
   ```

   ```
   $ chmod +x mariadb_es_repo_setup
   ```

   ```
   $ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --mariadb-server-version="10.6"
   ```

   1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Install MariaDB Enterprise Server and package dependencies:

   ```
   $ sudo yum install MariaDB-server MariaDB-backup
   ```

   Installation of additional packages may be required for some plugins.

4. Configure MariaDB.

   Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

INSTALL VIA APT

Install via APT (Debian, Ubuntu)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.
2. Configure the APT package repository.

   Installable versions of MariaDB Enterprise Server are `10.6`, `11.4`, and `11.8`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `10.6`.

To configure APT package repositories:

```
$ sudo apt install curl
```

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="10.6"
```

```
$ sudo apt update
```

1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Install MariaDB Enterprise Server and package dependencies:

```
$ sudo apt install mariadb-server mariadb-backup
```

Installation of additional packages may be required for some plugins.

4. Configure MariaDB.

Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

INSTALL VIA ZYPP

Install via ZYpp (SLES)

1. Retrieve your Customer Download Token at https://customers.mariadb.com/downloads/token/ and substitute for `CUSTOMER_DOWNLOAD_TOKEN` in the following directions.

2. Configure the ZYpp package repository.

Installable versions of MariaDB Enterprise Server are `10.6`, `11.4`, and `11.8`. Pass the version to install using the `--mariadb-server-version` flag to mariadb_es_repo_setup. The following directions reference `10.6`.

To configure ZYpp package repositories:

```
$ sudo zypper install curl
```

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
    | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
   --mariadb-server-version="10.6"
```

1. *Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

3. Install MariaDB Enterprise Server and package dependencies:

   ```
   $ sudo zypper install MariaDB-server MariaDB-backup
   ```

   Installation of additional packages may be required for some plugins.

4. Configure MariaDB.

   Installation only loads MariaDB Enterprise Server to the system. MariaDB Enterprise Server requires configuration before the database server is ready for use.

Configuration

For platforms that use YUM or ZYpp as a package manager:

MariaDB Community Server's packages bundle several configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

If your version of any of these configuration files contained any custom edits, then the package manager may save your edited version with the `.rpmsave` extension during the upgrade process. If you want to continue using your version with the custom edits, then you may need to move it back. For example, to move `server.cnf` back in place:

```
$ sudo mv /etc/my.cnf.d/server.cnf /etc/my.cnf.d/server.cnf.original
```

```
$ sudo mv /etc/my.cnf.d/server.cnf.rpmsave /etc/my.cnf.d/server.cnf
```

In addition to the configuration files listed above, MariaDB Enterprise Server installs its own configuration file:

- On platforms that use YUM or ZYpp: `/etc/my.cnf.d/mariadb-enterprise.cnf`
- On platforms that use APT: `/etc/mysql/mariadb.conf.d/mariadb-enterprise.cnf`

This file provides MariaDB Enterprise Server defaults, some of which change behavior compared to a default MariaDB Community Server configuration:

- Only plugins of stable maturity are loaded (`plugin-maturity=stable`).
- The MariaDB Enterprise Audit Plugin is preloaded (with audit logging off by default).
- The simple_password_check plugin is enabled, so new passwords must be at least 8 characters and include a digit, a special character, and both upper- and lowercase letters.
- The InnoDB adaptive hash index is kept disabled (also the default in MariaDB Community Server 10.6).

Do not edit this file directly. To override its settings, add your own options in a separate configuration file whose name sorts later than `mariadb-enterprise.cnf` .

Starting the Server

MariaDB Enterprise Server includes configuration to start, stop, restart, enable/disable on boot, and check the status of the Server using the operating system default process management system.

For distributions that use systemd, you can manage the Server process using the `systemctl` command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Upgrading the Data Directory

MariaDB Enterprise Server ships with a utility that can be used to identify and correct compatibility issues in the new version. After you upgrade your Server and start the server process, run the mariadb-upgrade utility to upgrade the data directory:

```
$ sudo mariadb-upgrade
```

Testing

When MariaDB Enterprise Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the server with MariaDB Client using the `root@localhost` user account:

   ```
   $ sudo mariadb
   ```

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MariaDB connection id is 9
   Server version: 10.6.28-24-MariaDB-enterprise MariaDB Enterprise Server

   Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

   MariaDB [(none)]>
   ```

   The `-enterprise` suffix in the version string confirms that you are connected to MariaDB Enterprise Server rather than MariaDB Community Server.

2. You can also verify the server version by checking the value of the version system variable with the SHOW GLOBAL VARIABLES statement:

   ```
   SHOW GLOBAL VARIABLES LIKE 'version';
   ```

   ```
   +---------------+-------------------------------+
   | Variable_name | Value                         |
   +---------------+-------------------------------+
   | version       | 10.6.28-24-MariaDB-enterprise |
   +---------------+-------------------------------+
   ```

3. You can also verify the server version by calling the VERSION() function:

```
SELECT VERSION();
```

```
+------------------------------+
| VERSION()                    |
+------------------------------+
| 10.6.28-24-MariaDB-enterprise |
+------------------------------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Downgrading MariaDB

> **Note:**
>
> Downgrade planning should ideally be taken into consideration before upgrading in settings where data loss cannot be acceptable. This may need setting up replication between versions in advance to enable controlled switching between versions if necessary.

Overview

Downgrading MariaDB is not officially supported between major versions. Downgrades, unlike upgrades, pose a potential risk since the on-disk data format, internal system tables, and storage engine architecture may not be backwards compatible.

However, a downgrade is achievable as long as you plan properly and avoid using any features that are exclusively available to the higher versions.

Before You Begin

If backported features are used with MariaDB Enterprise Server, downgrading after upgrading to a newer maintenance version may not be allowed. For more information, see the release notes for your version of MariaDB Enterprise Server, specifically the **Backports** section of the **What's New** page.

See MariaDB Enterprise Server Considerations.

Before attempting any downgrade, ensure the following factors:

Feature Compatibility

The downgrade will not succeed if any features introduced in the higher version are in use, such as new InnoDB table formats, storage engine behaviors, or system table structures.

Common version-specific features that may prevent downgrading include:

- **Instant ADD/DROP COLUMN** (`ALTER TABLE ALGORITHM=INSTANT`): Introduced across versions 10.3 and 10.4.
- **New InnoDB redo log formats**: Changed across multiple versions 10.2, 10.3, 10.5, 10.8, 11.0.
- `mysql.global_priv` table: Replaced `mysql.user` in MariaDB 10.4; incompatible with 10.3.
- **InnoDB change buffer removal**: MariaDB 11.0 removed it; cannot downgrade to 10.4 or earlier.

See Version-Specific Incompatibilities for the full list.

Configuration Compatibility

If the older version does not support configuration options added or enabled in the current version, downgrading (and replication to a lower version) may fail.

In such circumstances, configuration variables may need to be modified to match the target version's defaults or supported options before proceeding with the downgrade.

Maintenance Releases

Downgrades within the same release series are generally possible, but compatibility should be verified, especially if changes affecting storage formats or system tables are involved. There are relatively few exceptions; for example, MariaDB 10.1.21 fixed a file format bug that prevented downgrading to previous MariaDB 10.1 versions.

Why Major Version Downgrades Break

The main reasons for a major version downgrade failure are:

- **System table schema changes**: As the privilege system improves, the privilege and status tables in the mysql schema change between most major versions.
- **Format changes on on-disk data**: These are less common and generally table-specific, but when they occur (e.g., using Instant add column in MariaDB 10.3), the affected tables cannot be opened in earlier versions.
- **Internal changes to storage engines**: Both MariaDB 10.2 and MariaDB 10.3, for example, introduced new versions of the InnoDB redo log that are not backward-compatible.

Generic Downgrade Process

The following is a general approach to attempting a downgrade with minimal risk. This method, which minimizes downtime and data risk, is similar to the recommended upgrade path: you set up a replica running the target lower version, make sure it replicates well, and then switch to it.

1. Before attempting downgrade process, verify that features exclusive to the higher version are not used.
2. Use `mariadb-dump` to create a full logical backup of your entire database.

```
mariadb-dump --all-databases > dumpfile
```

3. Additionally, back up the `mysql` schema separately, as this is where the majority of version-specific upgrading changes occur.
4. Store both backups securely and verify their readability before proceeding. These backups serve as your recovery path if anything goes wrong.
5. Install MariaDB's target lower version on a separate server or environment. Then, set up the lower-version server as a replica and the current (higher version) server as the primary.
6. Verify replication is working. On the lower version replica, run:

```
SHOW REPLICA STATUS\G
```

5. Once validation is complete:
    1. Stop writes to the current (higher version) server.
    2. Promote the downgraded server to the primary.
    3. Redirect application traffic to the downgraded server.

Restore from Backup (Fallback Method)

If the replica-based method is not possible, the most reliable alternative is to restore from a full backup created in Step 2 of the Generic Downgrade Process.

1. Install a lower version of MariaDB on a clean server.
2. Launch MariaDB, then restore the backup:
3. To update system tables for the previous version, run mariadb-upgrade.

4. Restart MariaDB and check the connectivity of the application.

When used on a clean installation, this method is reliable but involves more downtime than the replica-based approach.

In-Place Downgrade Method

The MariaDB developers have not tested this method, so it is not advised. A lot of things can go wrong. Whenever possible, use the backup-restore or replica-based methods mentioned above.

In general, an in-place downgrade to a previous major version is only allowed if you have not yet executed mariadb-upgrade on the new version.

Note: It is recommended to run mariadb-upgrade after upgrading to ensure security and collation correctness, which limits the available downgrade window.

If you have to attempt an in-place downgrade process, perform the following steps:

1. Shut down MariaDB cleanly. Ensure:
    1. innodb_fast_shutdown≠2.
    2. You use SHUTDOWN command, mariadb-admin shutdown or the operating system official commands, like systemctl stop mariadb.service.
2. Start the old server binary with –skip-privilege-tables to bypass the incompatible privilege tables.
3. Restore the mysql schema tables to the old definitions using `ALTER TABLE`, or drop and recreate them. To find the old definitions, run mariadb-install-db on a temporary data directory, start a temporary server, and use SHOW CREATE TABLE.
4. Execute FLUSH PRIVILEGES to reload the restored privilege tables.

This procedure will **not** work if the table format has changed in an incompatible manner. In this case the affected tables may not be accessible in the earlier version. See Version-Specific Incompatibilities below.

Version-Specific Considerations

> **Warning:**
>
> Many of the examples that follow use outdated MariaDB versions that are currently End of Life (EOL). For production environments, downgrading to EOL versions is not recommended. These examples are provided for reference and to demonstrate potential incompatibilities.

The following is an incomplete list of cases where a table or component cannot be used in an earlier major version. Before proceeding, always check the Release Notes and Changes and Improvements pages for your target version to ensure that there are no additional incompatibilities specific to your version pair.

- MariaDB 11.0 or later
    - A downgrade to MariaDB 10.4 or earlier is not possible, because MDEV-29694 removed the InnoDB change buffer. making the format incompatible.
    - A downgrade to MariaDB 10.5 or later is only possible if innodb_change_buffering=none (the default starting with MDEV-27734).
- MariaDB 10.8 or later
    - The InnoDB redo log file `ib_logfile0` would need to be replaced with a logically equivalent file, or the shutdown LSN written to the `FIL_PAGE_FILE_FLUSH_LSN` field in the system tablespace (see MDEV-27199), or the data may be accessed read-only when using innodb_force_recovery=6.
- MariaDB 10.5 → MariaDB 10.4

    - The InnoDB redo log file `ib_logfile0` must be deleted between a clean shutdown of the 10.5 and a startup of 10.4. This is **not recommended**.

- MariaDB 10.4 → MariaDB 10.3
    - Any InnoDB table where `ALTER TABLE ALGORITHM=INSTANT DROP COLUMN` was used while innodb_instant_alter_column_allowed=add_drop_reorder.
    - Any InnoDB table created or rebuilt while innodb_checksum_algorithm=full_crc32.
    - In MariaDB 10.4, the `mysql.user` table was replaced by the mysql.global_priv table, which may cause problems if one wants to downgrade to 10.3.

- MariaDB 10.3 → MariaDB 10.2
    - Any InnoDB table where `ALTER TABLE…ADD COLUMN` was used (unless innodb_instant_alter_column_allowed=never).
    - A prior shutdown with innodb_fast_shutdown=0 is required in order to empty the undo logs whose format changed in MDEV-12288. Even then, setting innodb_force_recovery=3 may be necessary.

MariaDB Enterprise Server Considerations

MariaDB Enterprise Server may include backported features, which are functionalities from newer versions added to a previous maintenance release. Downgrading to a version without a backport (even within the same release series) might not be feasible if you use a backported feature.

Before downgrading Enterprise Server:

- Review the **Backports section** of the relevant **What's New** page for your current version.
- Check for enabled backported features.

See Also

- Upgrading MariaDB
- Backup and Restore Overview
- mariadb-dump
- Setting Up Replication
- mariadb-upgrade

This page is licensed: CC BY-SA / Gnu FDL

Migrating to MariaDB

See Differences Between MariaDB and Other DBMSs.

A high-level comparison of MariaDB's architecture, including its pluggable storage engines, transaction log handling, and buffer pool management versus other databases.

See DDL Export Guide.

Learn how to export SQL DDL[16] to MariaDB from database management systems like Oracle Database, Microsoft SQL Server, MySQL, and IBM DB2.

See Migrating to MariaDB from MySQL.

Complete MySQL to MariaDB migration: `mysqldump` / `mysql` import steps, SQL syntax compatibility, user/privilege migration, and replication configuration.

See Migrating to MariaDB from PostgreSQL.

Strategies for moving data from PostgreSQL to MariaDB, using the CONNECT storage engine with ODBC/JDBC or foreign data wrappers.

See Migrating to MariaDB from Oracle.

Migrate from Oracle to MariaDB Server. This section provides detailed guidance, tools, and best practices for a smooth and efficient transition of your databases and applications.

See Migrating to MariaDB from SQL Server.

Migrate from SQL Server to MariaDB. This section provides detailed guidance, tools, and best practices for a smooth and efficient transition of your databases and applications to MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

Differences Between MariaDB and Other DBMSs

This page lists *general* differences between MariaDB and other DBMSs (database management systems). For *specific* differences (for instance, between MariaDB and PostgreSQL), refer to the respective pages in this section.

Database vs. Schema

In MariaDB, the terms *schema* and *database* are synonymous and used interchangeable. The `CREATE SCHEMA` statement is a synonym for `CREATE DATABASE`, meaning they create the same object—a container for database objects like tables and views.

MariaDB does not support a distinction between schema and database as seen in other systems like SQL Server or PostgreSQL, where a schema is a logical container within a database. Instead, a database in MariaDB serves as both a namespace and a logical container to separate objects, and it has a default character set and collation that are inherited by its tables.

This page is licensed: CC BY-SA / Gnu FDL

DDL Export Guide

This page covers DDL[17] export instructions for Oracle Database, Microsoft SQL Server, MySQL, and IBM DB2.

Oracle Database

There are two basic ways of getting a schema from Oracle, with different features and advantages. The instructions here are based on **Oracle 11.2**. In old versions of Oracle, there was no `SCHEMAS` parameter; instead, only a single schema to export was specified with an `OWNER` parameter.

Using Oracle Data Pump

Oracle Data Pump is a tool used for backing up and restoring Oracle contents. The Data Pump commands (`expdp` and `impdp`) are just frontends that instruct the backend to do the actual dump and restore, which means that the output and input is not necessarily in the current directory. There is a default Data Pump destination directory, though.

> **Note:**
>
> Be aware that, in order to get the schema, you cannot just export it. Rather than that, you export what you are looking for, and then use `impdp` to extract an SQL file with the contents that you have dumped.

What this means is that if you are working from a backup, or a dump as it is called in Oracle, created by `expdp`, this method of extracting a schema is useful. There is not a lot of flexibility though.

Here is an example for extracting a schema called *APP*. First we create a dump – this usually must be done by a system user, as access is needed to the export directory mentioned before.

> **Note:**
>
> In Oracle, every user has a schema named the same as the user, and all objects have “owners”.
>
> All schema names are case sensitive.

The first parameter is the login. You log in as `sysdba`, which is a default DBA[18] user authenticated by the operating system, hence no username/password is specified, just a slash (`/`).

```
$ expdp \"/ as sysdba\" SCHEMAS=APP CONTENT=METADATA_ONLY EXCLUDE=STATISTICS DUMPFILE=app.dmpimpdp \"/ as
sysdba\" dumpfile=app.dmp sqlfile=app.sql
```

When run, the progress of the export is shown. When the process ends, it tells you where the export file is placed, which is the default Data Pump directory. In this case, it is `/u01/app/oracle/admin/XE/dpdump/`.

To extract the schema, run the import tool, like this:

```
$ impdp \"/ as sysdba\" dumpfile=app.dmp sqlfile=app.sql
```

The app.sql file in the default Data Pump directory is `/u01/app/oracle/admin/XE/dpdump/app.sql`.

The advantage of this method is that you can work from a backup. The disadvantage is that this method is not very flexible, and all facts on all objects are exported, even things that are not relevant for a migration.

To export all schema data, skip the `SCHEMAS` parameter, and instead add `FULL=Y`, like this:

```
$ expdp \"/ as sysdba\" FULL=Y CONTENT=METADATA_ONLY EXCLUDE=STATISTICS DUMPFILE=app.dmpimpdp \"/ as
sysdba\" dumpfile=app.dmp sqlfile=app.sql
```

To get the DDL for the appropriate objects, use the `impdp` command like above, where you can use the `SCHEMAS` parameter to filter which schemas you are looking for. If you don’t do this, you get the SQL for every object in the Oracle database, including system schemas.

Using dbms_metadata.get_ddl

This method is much more flexible and creates more reasonable output, but it requires a running Oracle database. It uses a built-in PL/SQL package called `dbms_metadata`, where the `get_ddl` function is used to retrieve the DDL for a specific object. There is also a function called `set_transform_param` that is used to pass parameters to the `get_ddl` function.

This method only exports objects relevant for a migration – it does not export Oracle specific objects nor any system owned objects.

A similar operation as with Data Pump can be executed using a script like this – put it in an SQL file called `exp.sql`:

```
# exp.sql
set long 99999999
set pagesize 0
set feedback off
set echo off
set heading off
SET TRIMSPOOL ON

BEGIN
   dbms_metadata.set_transform_param(dbms_metadata.session_transform, 'PRETTY', TRUE);
   dbms_metadata.set_transform_param(dbms_metadata.session_transform, 'TABLESPACE', FALSE);
   dbms_metadata.set_transform_param(dbms_metadata.session_transform, 'SQLTERMINATOR', TRUE);
   dbms_metadata.set_transform_param(dbms_metadata.session_transform, 'SEGMENT_ATTRIBUTES', FALSE);
   dbms_metadata.set_transform_param(dbms_metadata.session_transform, 'FORCE', FALSE);
END;
/

SELECT dbms_metadata.get_ddl(object_type, object_name, user)
  FROM user_objects
  WHERE object_type IN ('TABLE', 'VIEW', 'SEQUENCE', 'FUNCTION', 'PACKAGE',
   'PACKAGE_BODY', 'PROCEDURE', 'INDEX', 'SYNONYM', 'TRIGGER', 'TYPE')
  ORDER BY DECODE(object_type, 'TABLE', 1, 'VIEW', 2, 'INDEX', 3,
    'TRIGGER', 4, 'PACKAGE', 5, 'PACKAGE BODY', 6, 7), object_type;

EXIT
```

When done, run this script through the Oracle `SQL*Plus` utility. You can run that tool as an ordinary user with privileges appropriate to the schema to export. In other words, you do not need DBA privileges. Issue this command:

```
$ sqlplus -S app/app @exp.sql > schema.sql
```

To export all schemas using this method, excluding known system schemas, a slightly different SQL script is used, `expall.sql`:

```
# expall.sql
set long 99999999
set pagesize 50000
set feedback off
set echo off
set heading off
SET TRIMSPOOL ON

BEGIN
   dbms_metadata.set_transform_param(dbms_metadata.session_transform,
     'PRETTY', TRUE);
   dbms_metadata.set_transform_param(dbms_metadata.session_transform,
     'TABLESPACE', FALSE);
   dbms_metadata.set_transform_param(dbms_metadata.session_transform,
     'SQLTERMINATOR', TRUE);
   dbms_metadata.set_transform_param(dbms_metadata.session_transform,
     'SEGMENT_ATTRIBUTES', FALSE);
   dbms_metadata.set_transform_param(dbms_metadata.session_transform,
     'FORCE', FALSE);
   dbms_metadata.set_transform_param(dbms_metadata.session_transform,
     'CONSTRAINTS_AS_ALTER', TRUE);
END;
/

column OWNER noprint new_value owner_name
break on OWNER skip page
ttitle LEFT '-- Dumping objects for user ' owner_name

SELECT owner, dbms_metadata.get_ddl(object_type, object_name, owner)
  FROM dba_objects
  WHERE object_type IN ('TABLE', 'VIEW', 'SEQUENCE', 'FUNCTION', 'PACKAGE',
     'PACKAGE_BODY', 'PROCEDURE', 'INDEX', 'SYNONYM', 'TRIGGER', 'TYPE')
    AND owner NOT IN('CTXSYS', 'DBSNMP', 'DMSYS', 'MDDATA', 'MDSYS',
      'OLAPSYS', 'ORDSYS', 'ORDPLUGINS', 'OUTLN', 'PUBLIC',
      'SI_INFORMTN_SCHEMA', 'SYS', 'SYSMAN', 'SYSTEM', 'XDB')
    AND temporary = 'N' AND generated = 'N'
    AND (secondary = 'N' OR secondary IS NULL)
  ORDER BY owner, DECODE(object_type, 'TABLE', 1, 'VIEW', 2, 'INDEX', 3,
    'TRIGGER', 4, 'PACKAGE', 5, 'PACKAGE BODY', 6, 7), object_type;

EXIT
```

Run this script as a DBA user:

```
$ sqlplus -S / as sysdba @expall.sql > schema.sql
```

Microsoft SQL Server

For most existing SQL server environments, admin work is done using the GUI tool SQL server Management Studio (SSMS). Use the generate scripts option in SSMS: See this documentation for more details.

Command-line options exist – the SQL server team released a Python-based tool called `mssql-scripter`. Examples of usage are in this document. Full `mssql-scripter` usage options are listed here.

MySQL

Due to the similarities between MySQL and MariaDB, we are able to assess server configurations, too, for different types of `mysqldump` exports (`--all-databases` and `--databases`` *list-of-databases*).

Use `mysqldump` with the connection information to back up all your databases on the existing MySQL Server:

```
mysqldump --user db_user --password \
--host localhost \
--no-data \
--events \
--routines \
--triggers \
-r mydump.sql \
--all-databases
```

You can also choose to export only select databases:

```
mysqldump --user user_db --password \
--host localhost \
--no-data \
--events \
--routines \
--triggers \
-r mydump.sql \
--databases mydb1 mydb2 …
```

Optionally, you can upload your configuration variables to find differences between your MySQL and MariaDB server configurations. Run this command:

```
mysql --user db_user --password --host localhost -e "SHOW GLOBAL VARIABLES;" > variables.log
```

The `variables.log` file can be used for offline comparison between MySQL and MariaDB, to identify any differences.

IBM DB2

Log on to the DB2 server with a user that has `SELECT` privileges on the system catalog tables. In some specific cases, additional grants might be required, such as `SYSADM`, `SYSCTRL`, `SYSMAINT`, `SYSMON`, and `DBADM`. For our specific requirements, the `SELECT` privilege on system catalog tables suffices, though.

> **Note:**
>
> In case of problems, request the DBA to export the schema DDL.

The following list gives examples of various extract options available when extracting DDL from a DB2 server.

- Generate the DDL script for all the objects created by user `foo` in `world` database:
  - `-d` defines the database/alias of the database that you want to extract.
  - `-u` defines the user name you want to export objects of.
  - `-e` is used to export all the possible objects including tables, stored procedures, aliases, triggers, etc.
  - `-o ddl_export.sql` represents the output file.
  - This is the complete command:
    `$ db2look -d world -u foo -e -o ddl_export.sql`
- Generate the DDL script for objects in a particular schema:
  - `-d` defines the database/alias of the database that you want to extract.
  - `-u` defines the user name we want to export objects of.
  - `-z` defines the schema name.
  - `-e` is used to export all the possible objects including tables, stored procedures, aliases, triggers, etc.
  - `-o ddl_export.sql` represents the output file name.
  - This is the complete command:
    `$ db2look -d world -u foo -z globe -e -o ddl_export.sql`

- Generate the DDL script of all the objects regardless of the user they are created with. This is the one most of the exports will be done with, for the sake of completion of the DDL script:
  - `-d` defines the database/alias of the database that you want to extract.
  - `-a` is used to extract objects for all users on the particular database.
  - `-o ddl_export.sql` represents the output file name.
  - `-e` is used to export all the possible objects including tables, stored procedures, aliases, triggers, etc.
  - This is the complete command:
    ```
    $ db2look -d world -a -e -o ddl_export.sql
    ```

The `db2look` command arguments mentioned before work for all DB2 versions starting from DB2 9.7 to 11.5, but for specifics details, refer to the following DB2 pages:

- 9.7 - db2look - DB2 statistics and DDL extraction tool command
- 10.5 - db2look - DB2 statistics and DDL extraction tool command
- 11.5 - db2look - Db2 statistics and DDL extraction tool command

This page is licensed: CC BY-SA / Gnu FDL

Migrating to MariaDB from MySQL

MariaDB Server is designed to be a high-performance, drop-in replacement for MySQL. Whether you are moving from legacy versions (5.5, 5.6, 5.7) or modern releases (8.0, 8.4 LTS), this section provides the tools and workflows needed for a successful transition.

Evaluate Compatibility

See MySQL to MariaDB Compatibility Matrix.

Before moving data, understand the technical differences between your current MySQL version and your target MariaDB version. Our Compatibility Matrix covers changes in authentication, SQL syntax, and system variables. It's a detailed reference for DBAs and developers.

Choose Your Migration Path

See MySQL to MariaDB Migration: The Master Guide.

The best method for migration depends on your downtime requirements and server architecture. We have consolidated these workflows into a single authoritative guide.

- MySQL to MariaDB Migration: The Master Guide — Step-by-step instructions for:
  - In-Place Upgrades: Fast, same-server binary swaps.
  - Logical Migrations: Safe, cross-server dump and restore.
  - Cloud & RDS Migrations: Specialized paths for managed environments.

Specialized Migration Scenarios

See Migrating Galera Cluster.

See Migrating from MySQL to MariaDB Galera Cluster (Node-by-Node In-Place).

Guides for migrating MySQL Galera Cluster to MariaDB Galera Cluster.

This page is licensed: CC BY-SA / Gnu FDL

MySQL to MariaDB Migration: The Master Guide

> **Note:**
>
> **Environment Scope**
>
> This guide focuses on **migrations within Linux-based environments** (RHEL/CentOS/Alma, Debian/Ubuntu, etc.), as these represent the vast majority of production MySQL and MariaDB deployments.

Preparation

Pre-Migration Checklist

Before touching the production server, verify these requirements:

- Version Target: Identify your destination.
- *Coming from MySQL 5.7?* MariaDB 10.11 LTS or 11.8 LTS are safe harbors with the longest remaining support runways.
- *Coming from MySQL 8.0/8.4?* MariaDB 11.8 LTS or the new 12.3 LTS are recommended. These versions have the most modern feature parity required for 8.x-era applications.
- Compatibility Check: Review the MySQL to MariaDB Compatibility Matrix for potential high-impact differences in authentication and SQL syntax.

Mandatory Backup

Never start a migration without two types of backups:

1. Physical/Binary Backup: A copy of the `/var/lib/mysql` directory (while the server is stopped). This allows for the fastest “undo” if the binary swap fails.
2. Logical Backup: A full dump of your data.
   Use the MySQL `mysqldump` program, not MariaDB’s `mariadb-dump`, because the latter hasn’t been extensively tested against MySQL.
   Don’t dump `--all-databases`, because that would include the MySQL system databases, which are different from the MariaDB ones – loading them would fail on data import. Instead, specify which databases to export as arguments to `--databases`.

   ```
   # The Universal Master Dump Command
   mysqldump --user=root --password --databases db1 db2 ... \
   --single-transaction --routines --events \
   --triggers --hex-blob > migration_dump.sql
   ```

Porting Users and Privileges

Standard data dumps often fail to capture complex user permissions correctly when moving from MySQL 8.0+. Instead of migrating the entire `mysql` system database, use these scripts on your MySQL source server to generate the exact SQL commands needed to recreate your users and their permissions on MariaDB.

Creating Users on MariaDB

The following SQL can be used to generate CREATE USER statements with a default password that can be executed on MariaDB. It does the following:

- Authentication method mysql_native_password – the script supports passwords porting.
- Authentication method caching_sha2_password or authentication string is `NULL` – passwords are reset to default with expiry.

The PARSEC authentication plugin is intended to be the default in a future release, hence it is recommended to use this during user migration.

```
SELECT
    CASE
        WHEN plugin = 'mysql_native_password'
             AND authentication_string IS NOT NULL
             AND authentication_string <> ''
        THEN CONCAT(
            'CREATE USER IF NOT EXISTS ''', user, '''@''', host,
            ''' IDENTIFIED VIA mysql_native_password USING ''',
            authentication_string, ''';'
        )

        WHEN plugin = 'mysql_native_password'
             AND (authentication_string IS NULL OR authentication_string = '')
        THEN CONCAT(
            'CREATE USER IF NOT EXISTS ''', user, '''@''', host,
            ''' IDENTIFIED BY ''Welcome@123'' PASSWORD EXPIRE; ',
            '-- no source hash available'
        )

        WHEN plugin IN ('caching_sha2_password', 'sha256_password')
        THEN CONCAT(
            'CREATE USER IF NOT EXISTS ''', user, '''@''', host,
            ''' IDENTIFIED BY ''Welcome@123'' PASSWORD EXPIRE; ',
            '-- original plugin: ', plugin
        )

        ELSE CONCAT(
            '-- SKIPPED unsupported plugin for ''', user, '''@''', host,
            ''': ', COALESCE(plugin, 'NULL')
        )
    END AS migration_sql
FROM mysql.user
WHERE user NOT IN ('mysql.sys','mysql.session','mysql.infoschema','root');
```

Copying User Privileges to MariaDB

Use the following SQL to generate user privilege statements for execution in MariaDB from MySQL.

Privileges covered are:

- Global privileges
- Database privileges
- Table privileges
- Procedure privileges

> **Note:**
>
> The SQL may vary depending on the version of MySQL you're coming from.

```
SET SESSION group_concat_max_len = 1000000;

SELECT '-- GLOBAL PRIVILEGES' AS '';

SELECT CONCAT(
'GRANT ',
TRIM(TRAILING ',' FROM CONCAT(
IF(Select_priv='Y','SELECT,',''),
IF(Insert_priv='Y','INSERT,',''),
IF(Update_priv='Y','UPDATE,',''),
IF(Delete_priv='Y','DELETE,',''),
IF(Create_priv='Y','CREATE,',''),
IF(Drop_priv='Y','DROP,',''),
IF(Grant_priv='Y','GRANT OPTION,',''),
IF(References_priv='Y','REFERENCES,',''),
IF(Index_priv='Y','INDEX,',''),
IF(Alter_priv='Y','ALTER,',''),
IF(Create_view_priv='Y','CREATE VIEW,',''),
IF(Show_view_priv='Y','SHOW VIEW,',''),
IF(Create_routine_priv='Y','CREATE ROUTINE,',''),
IF(Alter_routine_priv='Y','ALTER ROUTINE,',''),
IF(Execute_priv='Y','EXECUTE,',''),
IF(Event_priv='Y','EVENT,',''),
IF(Trigger_priv='Y','TRIGGER,','')
)),
' ON *.* TO ''', user, '''@''', host, ''';'
)
FROM mysql.user
WHERE CONCAT(
Select_priv,Insert_priv,Update_priv,Delete_priv,
Create_priv,Drop_priv,Grant_priv,References_priv,
Index_priv,Alter_priv,Create_view_priv,Show_view_priv,
Create_routine_priv,Alter_routine_priv,Execute_priv,
Event_priv,Trigger_priv
) REGEXP 'Y';

SELECT '-- DATABASE PRIVILEGES' AS '';

SELECT CONCAT(
'GRANT ',
TRIM(TRAILING ',' FROM CONCAT(
IF(Select_priv='Y','SELECT,',''),
IF(Insert_priv='Y','INSERT,',''),
IF(Update_priv='Y','UPDATE,',''),
IF(Delete_priv='Y','DELETE,',''),
IF(Create_priv='Y','CREATE,',''),
IF(Drop_priv='Y','DROP,',''),
IF(Grant_priv='Y','GRANT OPTION,',''),
IF(References_priv='Y','REFERENCES,',''),
IF(Index_priv='Y','INDEX,',''),
IF(Alter_priv='Y','ALTER,',''),
IF(Create_view_priv='Y','CREATE VIEW,',''),
IF(Show_view_priv='Y','SHOW VIEW,',''),
IF(Create_routine_priv='Y','CREATE ROUTINE,',''),
IF(Alter_routine_priv='Y','ALTER ROUTINE,',''),
IF(Execute_priv='Y','EXECUTE,',''),
IF(Event_priv='Y','EVENT,',''),
IF(Trigger_priv='Y','TRIGGER,','')
)),
' ON `', db, '`.* TO ''', user, '''@''', host, ''';'
)
FROM mysql.db;

SELECT '-- TABLE PRIVILEGES' AS '';

SELECT CONCAT(
'GRANT ', Table_priv,
' ON `', Db, '`.`', Table_name,
```

```
'` TO ''', User, '''@''', Host, ''';'
)
FROM mysql.tables_priv;

SELECT '-- PROCEDURE PRIVILEGES' AS '';

SELECT CONCAT(
'GRANT ', Proc_priv,
' ON PROCEDURE `', Db, '`.`', Routine_name,
'` TO ''', User, '''@''', Host, ''';'
)
FROM mysql.procs_priv;
```

Proactive SQL Validation

If your application is mission-critical:

- Enable the General Query Log on your MySQL server for a period of time to capture real-world traffic.
- Replay those queries against a MariaDB test instance.
- Check the MariaDB error log for any “Syntax Error” or “Unknown Function” messages. This prevents surprises on migration night.

Step by Step Migration Procedure

Logical migration (dump and restore) is the gold standard for safety. It is the best choice if you are moving to a new server, a different Linux distribution, or a cloud-managed environment.

By exporting the data into an SQL script, you effectively “filter” out any binary-level incompatibilities between MySQL and MariaDB.

A logical migration involves exporting your data into a text-based SQL file (a “dump”) and importing it into a fresh MariaDB instance (“restore”). This is the safest way to ensure data integrity, as it completely recreates your tables and indexes in the MariaDB format.

> **Step 1**
>
> **Export Data from MySQL**
>
> On your existing MySQL server, create a complete dump of all databases, including stored procedures, triggers, and events. (Note this is the same dump command used in the Preparation step, so you can use that one instead of re-creating it.)
>
> ```
> # The Universal Master Dump Command
> mysqldump --user=root --password --databases db1 db2 ... \
> --single-transaction --routines --events \
> --triggers --hex-blob > migration_dump.sql
> ```
>
> - `--single-transaction` : Ensures a consistent backup without locking your tables (for InnoDB).
> - `--hex-blob` : Properly handles binary data like images or encrypted strings.
> - `--routines` and `--events` ensures that the “logic” of the database moves, not just the “rows”.

> **Step 2**
>
> **Prepare the MariaDB Environment**
>
> On your new server, install MariaDB using the Installation Guide for MariaDB Enterprise Server or the Installation Guide for MariaDB Community Server.
>
> Before importing, ensure your MariaDB configuration (`my.cnf`) has enough resources to handle a large import.
>
> - `max_allowed_packet` : Set this to at least `64M` or higher if you have large `BLOB` columns.
> - `innodb_buffer_pool_size` : Temporarily increase this to 70-80% of available RAM to speed up index creation.

> **Step 3**
>
> **Import Data into MariaDB**
>
> Transfer your `mysql_full_dump.sql` file to the new server and run the import:
>
> ```
> # Import the data
> mariadb -u root -p < mysql_full_dump.sql
> ```

Alternative: Direct Network Streaming (The Pipe Method)

If you have limited disk space or a high-speed network connection between your servers, you can “stream” the data directly from the MySQL source to the MariaDB target. This avoids creating a large intermediate `.sql` file on your local disk.

> **Warning:**
>
> Streaming can take a long time for large databases. If it is aborted for any reason (like network timeout), it usually cannot be resumed. This means you’d have to start again from the beginning.

```
mysqldump --user=root --password --databases db1 db2 ... \
--single-transaction --routines --events \
--triggers --hex-blob | mariadb --host=new_server_ip --user=root --password
```

> Pro Tip: To monitor the progress of your transfer in real-time, install the `pv` (Pipe Viewer) utility and insert it into the command: `mysqldump ... | pv | mariadb ...`

Verification and Optimization

Once your data is moved (via either method), complete these three steps to ensure your new server is running at peak performance.

> **Step 1**
>
> **Rebuild Optimizer Statistics**
>
> MariaDB uses a sophisticated cost-based optimizer that may differ from MySQL’s. To ensure your queries use the most efficient execution plans, force a fresh analysis of all tables:
>
> ```
> # Run ANALYZE TABLE on every table in every database to refresh statistics
> mariadb-check -u root -p --analyze --all-databases
> ```
>
> This makes the difference between a migration that “works” and a migration that “works fast”.

Step 2

Character Set & Collation Validation

Ensure your server and your applications agree on the character set and collation to use.

If you notice unexpected behavior or performance drops in specific queries, verify your collations:

```
-- Check for collation issues in your migrated database
SHOW GLOBAL VARIABLES LIKE 'collation%';
SHOW  VARIABLES LIKE 'collation%';
SELECT table_schema, table_name, column_name, collation_name FROM information_schema.columns WHERE
collation_name IS NOT NULL and table_schema not
in('information_schema','performance_schema','mysql','sys') ORDER BY table_schema, table_name;
```

Check if you have queries that may lead to a mismatch error when joining 2 columns with different collations. Alter your table or database to use the same collation across all your databases. See ALTER TABLE and ALTER DATABASE for details.

> **Note:**
>
> **Why this matters**
>
> If a user has one table set to the old MySQL default and another table set to a MariaDB default, a `JOIN` between them might suddenly stop using indexes because of the "collation mismatch." The above snippet gives you the SQL command to find the problem immediately.

Step 3

Check the Error Log

Check for any "Deprecated Variable" or "Ignored Option" warnings that might have appeared during startup.

```
sudo tail -f /var/log/mysql/error.log | grep -e 'Deprecated|Ignored'
```

Step 4

Application Smoke Test

Verify that your application can connect. Pay special attention to character sets: Ensure your apps and the server agree on `utf8mb4` or whatever character set you choose.

Step 5

Security Hardening: Transition to PARSEC

For migrations to MariaDB 11.8 LTS and later, consider transitioning your users to the PARSEC authentication plugin. While `mysql_native_password` and `caching_sha2_password` are supported for compatibility, PARSEC is the modern successor designed for significantly stronger password hashing and better long-term security.

```
-- Example: Update a user to use PARSEC
ALTER USER 'app_user'@'localhost' IDENTIFIED VIA parsec USING PASSWORD('secure_password');
```

Troubleshooting & FAQ

Even with careful preparation, migrations can encounter specific hurdles. Here are the most common issues and how to resolve them.

Common Troubleshooting Scenarios

Issue	Likely Cause	Resolution
“Table ‘mysql.user’ doesn’t exist”	Missing `mariadb-upgrade` step.	The system tables must be converted. Run `sudo mariadb-upgrade -u root -p` immediately after starting the service.
Replication fails with “relay log” errors.	GTID Incompatibility.	MariaDB cannot use MySQL GTIDs. You must switch to position-based replication (File and Position) to link the two systems.
Slow queries after migration.	Outdated optimizer statistics.	MariaDB’s optimizer needs fresh data. Run `ANALYZE TABLE` on all large tables, or use `mariadb-check --analyze --all-databases`.

Frequently Asked Questions

Q: Can I go back to MySQL if the MariaDB migration fails? A: Yes, provided you took a physical backup (the `/var/lib/mysql` folder) before running `mariadb-upgrade`. Once `mariadb-upgrade` has modified the system tables, you cannot simply point MySQL binaries back at that data directory.

Q: Do I need to change my application’s client libraries? A: Usually, no. MariaDB is protocol-compatible with MySQL. Your existing MySQL connectors (JDBC, PHP PDO, Python mysqlclient) will continue to work. However, for the best performance and features (like IAM authentication), switching to the official MariaDB Connectors is recommended.

Q: What about the JSON data type? A: Your `JSON` columns will be treated as `LONGTEXT` with a `CHECK` constraint. Your queries using `JSON_EXTRACT()` will continue to work exactly as they did in MySQL.

Q: Is MariaDB 11.8 compatible with MySQL 8.4? A: Yes, for standard SQL and data. However, MySQL 8.4 removed many “legacy” behaviors that MariaDB still supports.

Q: Can I perform an in-place upgrade (binary swap)? A: No. MariaDB stores metadata (table structures etc.) in individual `.frm` files. In contrast, MySQL removed `.frm` files in MySQL 8.0, replacing it with a global data dictionary. If you stop a MySQL 8.0/8.4 server and point MariaDB binaries at that data directory, MariaDB will look for `.frm` files to understand what tables exist. Since those files no longer exist in MySQL 8.0+, MariaDB will see an “empty” data directory or throw critical errors. It has no way to read MySQL’s new internal dictionary tables.

Q: Can I perform an in-place upgrade for MySQL versions prior to 5.7? After all, those versions still use `.frm` files instead of a global data dictionary. A: In theory, that’s possible, but in practice, you shouldn’t do that either. Those old MySQL versions are built for operating systems like RHEL 6 which are out of support. You’d end up on an unsupported version of the operating system, on which you shouldn’t install supported versions of MariaDB.

Further Reading

See MySQL to MariaDB Compatibility Matrix.

The reference guide companion page to the Migration Master Guide.

See MariaDB versus MySQL - Features.

Compare MariaDB vs. MySQL features. Learn about exclusive storage engines, speed enhancements and binary compatibility.

See MariaDB versus MySQL - Compatibility.

Complete MariaDB Community Server release notes. Complete version history with features, bug fixes, and upgrade compatibility details for production use.

See Function Differences Between MariaDB and MySQL.

Functions in MariaDB that are not present in MySQL, or vice versa.

See System Variable Differences between MariaDB and MySQL.

Explore system variable differences between MariaDB Rolling Release and MySQL 8.0. This section details how configuration options vary, aiding in compatibility and migration planning.

See Replication Compatibility Between MariaDB and MySQL.

Describes replication compatibility between MariaDB and MySQL.

This page is licensed: CC BY-SA / Gnu FDL

MySQL to MariaDB Compatibility Matrix

Introduction & Executive Summary

MariaDB Server was originally designed as a drop-in replacement for MySQL, maintaining high compatibility across library binary protocols, file structures, and APIs. For most applications, moving from MySQL to MariaDB is a straightforward process that requires minimal changes to your code or configuration.

However, as both projects have evolved—particularly with the release of MySQL 8.0, the latest MySQL 8.4 LTS, and MariaDB 10.11/11.4 LTS—they have introduced unique features and default behaviors. While MariaDB often maintains support for "legacy" MySQL behaviors for compatibility, MySQL 8.4 has moved to aggressively remove them. This guide highlights the essential differences you must understand to ensure a seamless transition.

The TL;DR of MySQL to MariaDB Migration

If you are migrating from MySQL 8.0 or 8.4 LTS to a modern version of MariaDB, these are the four "deal-breaker" areas where you are most likely to encounter friction:

Authentication & Security

MySQL 8.4 has accelerated the push toward `caching_sha2_password`. Notably, the older `mysql_native_password` plugin is now disabled by default in MySQL 8.4 and requires explicit configuration to enable. MariaDB continues to support and often default to `mysql_native_password` or `unix_socket`.

- Action: Check your client connector versions. When moving from 8.4 to MariaDB, you may need to re-enable or transition user accounts back to `mysql_native_password` or `ed25519`.

Feature & Variable Removals

A key difference in MySQL 8.4 is the removal of many system variables and syntax options that were deprecated in 8.0. MariaDB often retains these for backward compatibility.

- Action: If you have optimized your configuration specifically for 8.4 by removing "removed" variables, MariaDB will likely accept them, but it may also still support the older variants you previously used in MySQL 5.7 or 8.0.

Global Transaction IDs

MariaDB and MySQL use entirely different formats for GTIDs. In MySQL 8.4, replication syntax has been further modernized (e.g., removal of `MASTER` in favor of `SOURCE` keywords). MariaDB maintains its own unique GTID structure (`DomainID:ServerID:Sequence`).

- Action: You cannot "mix and match" GTID replication directly. Plan to use position-based replication if running a hybrid environment during your migration.

Data Types & JSON Evolution

Both systems have diverged in how they handle complex data. MySQL 8.0 and 8.4 use a native binary format for JSON, optimized for their internal engine. MariaDB treats JSON as an alias for `LONGTEXT` with `CHECK` constraints.

- Action: While SQL-level JSON functions are highly compatible, performance and storage characteristics differ. Test any application logic that relies on high-speed JSON attribute extraction.

How to use this Matrix

The table below provides a granular breakdown of these differences. We recommend focusing on the “High Impact” items first, as these are the most likely to affect your application’s ability to connect or execute queries immediately after the migration.

> Target Version Tip: If you are migrating from MySQL 8.4 LTS, the most logical targets are MariaDB 10.11 LTS or MariaDB 11.4 LTS to ensure you are moving from one stable long-term environment to another.

> **Note:**
>
> **Key changes from the 8.4 Nutshell you should keep in mind**
>
> - Authentication: The fact that `mysql_native_password` is now “off” by default in 8.4 is a huge point of divergence. Most MariaDB installs still rely on it.
> - Syntax: 8.4 officially removed `SET OPTION`, the `LOW_PRIORITY` prefix for some statements, and various SSL variables. If a user’s app was updated to work with 8.4, it’s “cleaner” but might be using syntax that MariaDB handles differently.
> - Default Changes: Some InnoDB defaults changed in 8.4 that might not match MariaDB’s defaults.

Authentication & Security

Component	MySQL 8.0 / 8.4 Behavior	MariaDB Behavior	Migration Impact
Default Authentication Plugin	`caching_sha2_password` (MySQL 8.4 requires explicit config to even *enable* native password).	`mysql_native_password` or `unix_socket`.	High. New MySQL users cannot connect to MariaDB without altering the account or updating the client.
Connection Security	`REQUIRE SSL` is common.	Supports SSL/TLS but syntax for specific ciphers can vary.	Medium. Review `GRANT` statements for SSL requirements.
Password Complexity	`validate_password` component.	`simple_password_check` or `cracklib_password_check` plugins.	Low. Logic is similar, but plugin names and variables differ.
User Locking	Supports `ACCOUNT LOCK` / `UNLOCK`.	Supports `ACCOUNT LOCK` / `UNLOCK` (10.4.2+).	None. Highly compatible syntax.

SQL Syntax & Data Types

Component	MySQL 8.0 / 8.4 Behavior	MariaDB Behavior	Migration Impact
JSON Support	Native binary format (`JSON` type).	Alias for `LONGTEXT` with `CHECK` constraints.	Medium. SQL syntax is compatible, but binary-level storage and some indexing optimizations differ.
Reserved Words	Frequent additions (e.g., `WINDOW`, `EXCEPT`, `INTERSECT` in 8.0+).	Also uses these reserved words, but some unique MariaDB keywords may exist.	Low. Most modern reserved words are now synchronized between both.
Invisible Columns	Supported since 8.0.23.	Supported since 10.3.3.	None. Highly compatible syntax.
Common Table Expr (CTE)	Non-recursive and Recursive support.	Non-recursive and Recursive support.	None. Functionally identical for standard use cases.
Window Functions	Standard support since 8.0.	Standard support since 10.2.	Low. Standard syntax is compatible; specific MySQL-only extensions are rare.

System Variables & Defaults

This section helps the DBA avoid “Unknown variable” errors when they first try to start the MariaDB service using a MySQL configuration file.

Feature / Variable	MySQL 8.0 / 8.4 Defaults	MariaDB Defaults	Migration Impact
Configuration File	Reads `my.cnf` / `my.ini`.	Reads `my.cnf` / `my.ini` and `mariadb.cnf`.	None. MariaDB maintains compatibility with existing MySQL config files.
Removed Variables	Many 5.7 variables removed in 8.4 (e.g., `expire_logs_days`).	Often retains these legacy variables as aliases.	Low. MariaDB is generally more forgiving of “old” variables than MySQL 8.4.
Default Charset	`utf8mb4_0900_ai_ci` (MySQL 8.0+).	`utf8mb4_uca1400_ai_ci` (MariaDB 11.4+).	Medium. Sorting/Collation behavior may differ slightly for modern Unicode characters.
Explicit Defaults	`explicit_defaults_for_timestamp` is `ON`.	`explicit_defaults_for_timestamp` defaults to `OFF` (for legacy compatibility).	Medium. Can cause different behavior for `TIMESTAMP` columns without explicit defaults.

Replication & GTIDs

Feature	MySQL 8.0 / 8.4 Behavior	MariaDB Behavior	Migration Impact
GTID Format	`UUID:Sequence`	`Domain:ServerID:Sequence`	Critical. Formats are incompatible. GTID replication cannot be mixed directly.
Repl. Syntax	Uses `SOURCE` / `REPLICA` (8.0.22+).	Uses `MASTER` and `REPLICA`.	None. MariaDB is bi-lingual regarding replication syntax.
Binary Logging	Defaulted to `ON`.	Defaulted to `ON` in recent versions.	Low. Standard binary log events are generally compatible.

Next Steps

Now that you’ve reviewed the compatibility differences, you are ready to begin the migration.

Proceed to MySQL to MariaDB Migration: The Master Guide for step-by-step instructions on performing an in-place upgrade or a logical migration.

This page is licensed: CC BY-SA / Gnu FDL

Migrating Galera Cluster

This guide covers the migration of a live database from a MySQL Galera Cluster to a MariaDB Galera Cluster. The migration strategy requires setting up a new MariaDB Cluster, then using asynchronous replication to sync it with an existing MySQL Cluster. The method includes a reliable failback route.

> **Note:**
>
> This document focuses on the migration process itself. For detailed comparisons between the two database systems, please refer to Incompatibilities and Feature Differences Between MariaDB and MySQL.

Prerequisites

- **MySQL Server**: A running MySQL Galera Cluster (version 8.0 or 8.4) to migrate.
- **MariaDB Server**: The target MariaDB version that supports Galera Cluster. This guide assumes the latest stable version.
- **`mysqldump`**: A command-line utility by MySQL to create initial database backup.

> **Warning:**
>
> Due to changes in MySQL 8.0+, `mariadb-dump` may not be compatible for this initial step.

- **Root or Sudo Access**: Administrative privileges are required on all servers involved in the migration.

Migration Steps

Step 1

Check for Incompatibilities

> **Warning:**
>
> While MariaDB maintains a high degree of compatibility with MySQL, it is crucial to review any differences that could impact your applications.

- **Version Compatibility**: Select a MariaDB version compatible with your MySQL version for replication.
- **Feature Differences**: Review the Incompatibilities and Feature Differences to identify any deprecated features or changes in system tables, user accounts, and authentication that might affect your setup.

Replication Compatibility

MySQL Version	Compatible MariaDB Version
5.7	10.2 and newer
8.0	10.6.21, 10.11.11, 11.4.5, 11.7.2 and newer

For more details, see MySQL to MariaDB Replication Compatibility.

Step 2

Install and Configure a New MariaDB Galera Cluster

Set up a new, empty MariaDB Galera Cluster that will become the target for the migration.

1. **Install MariaDB Server and Galera Provider**

Install MariaDB Server package and Galera wsrep provider library on each node of the new cluster. Refer Install and Upgrade MariaDB for detailed instructions.

2. **Configure Each MariaDB Node**

Create a configuration file at `/etc/mysql/conf.d/galera.cnf` on each node. The configuration will be mostly identical across nodes, except for node-specific details like `wsrep_node_name` and `wsrep_node_address`.

Example `galera.cnf`:

```
[mysqld]
# MariaDB settings
binlog_format=ROW
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so # Adjust path if necessary

# Galera Cluster Configuration
wsrep_cluster_name="new_mariadb_cluster"
wsrep_cluster_address="gcomm://node1_ip,node2_ip,node3_ip"

# Node-specific Configuration
wsrep_node_name="mariadb_node1"
wsrep_node_address="node1_ip"

# SST Configuration
wsrep_sst_method=mariadb-backup
```

Note:

It is recommended to set `wsrep_sst_method` on all nodes to ensure consistency during State Snapshot Transfers(SST).

Step 3

Back Up the MySQL Database

A full logical backup of the MySQL database is required to seed the new MariaDB Cluster.

1. **Enable Binary Logging on MySQL**

   Asynchronous replication requires the binary log to be enabled on the MySQL source. If not enabled, enable it on at least one node in your MySQL cluster.

   1. **Edit `my.cnf`** : In the `[mysqld]` section of your MySQL configuration file, add `log-bin=mysql-bin` .
   2. **Restart MySQL**: A server restart is required for the change to take effect.

> **Important:**
>
> Restarting MySQL will cause a brief outage for the node.

2. **Create the Backup with `mysqldump`**

Use `mysqldump` from the source MySQL server to create a full backup. Using `--all-databases` and `--master-data` will include all databases and the binary log position needed to start replication.

```
mysqldump --user=backup_user --password --all-databases --master-data=2 > /path/to/backup.sql
```

> **Note:**
>
> This above command generates a backup file and logs the source's binary log file name and position in the output.

Step 4

Bootstrap the MariaDB Cluster and Restore Data

Start the first node of the MariaDB cluster and load the MySQL backup.

1. **Bootstrap the First Node**

Start the first MariaDB node with the `--wsrep-new-cluster` option. This initializes the cluster.

```
sudo galera_new_cluster
```

2. **Load the Backup File**

Transfer the `backup.sql` file to the first MariaDB node and import it:

```
mariadb -u root -p < /path/to/backup.sql
```

3. **Start Subsequent Nodes**

Start the MariaDB service normally on the remaining nodes. They will connect to the first node, and a State Snapshot Transfer (SST) will automatically sync their data.

```
sudo systemctl start mariadb
```

4. **Verify the Cluster**

Connect to any node and check the cluster size to ensure all nodes have joined successfully.

```
SHOW STATUS LIKE 'wsrep_cluster_size';
```

> **Note:**
>
> Cluster size should match the number of nodes in your cluster.

Step 5

Set Up Asynchronous Replication

Configure one of the MariaDB nodes to act as a replica of the source MySQL cluster.

1. **Create a Replication User on MySQL**

On the MySQL source node where binlogging is enabled, create a dedicated user for replication.

```
CREATE USER 'repl_user'@'mariadb_node_ip' IDENTIFIED BY 'your_password';
GRANT REPLICATION SLAVE ON *.* TO 'repl_user'@'mariadb_node_ip';
```

2. **Configure the MariaDB Replica**

On one of the MariaDB nodes, configure and start the replication process. Use the binary log file and position noted from the `mysqldump` output (`MASTER_LOG_FILE` and `MASTER_LOG_POS`) or use GTID-based replication.

```
CHANGE MASTER TO
  MASTER_HOST='mysql_source_ip',
  MASTER_PORT=3306,
  MASTER_USER='repl_user',
  MASTER_PASSWORD='your_password',
  MASTER_USE_GTID=slave_pos;

START SLAVE;
```

> **Step 6**
>
> **Monitor and Failover**
>
> With replication running, the MariaDB cluster will catch up with any changes made to the MySQL cluster since the backup was taken.
>
> 1. **Monitor Replication Lag**
>
> Check the replica status on the MariaDB node to ensure it is synchronized.
>
> ```
> SHOW SLAVE STATUS\G
> ```
>
> Ensure the following conditions are met on the replica:
>
> - `Slave_IO_Running: Yes`
> - `Slave_SQL_Running: Yes`
> - `Seconds_Behind_Master: 0`
>
> A `Seconds_Behind_Master` value of `0` indicates the replica is completely synchronized.
>
> 2. **Perform the Failover**
>
> Once the MariaDB cluster is fully synchronized, follow these steps:
>
> 1. **Stop Application Traffic:** Cease traffic to the MySQL cluster.
> 2. **Verify Synchronization:** Confirm the MariaDB replica has processed all events from the MySQL binary log.
> 3. **Promote MariaDB:**
>   - Stop replication on the MariaDB node using `STOP SLAVE;`.
>   - Update your application to connect to the MariaDB Galera Cluster.
> 4. **Resume Application Traffic:** Redirect traffic to the MariaDB cluster.
> 5. **Decommission MySQL:** After confirming the application runs smoothly on MariaDB, decommission the MySQL cluster.

This page is licensed: CC BY-SA / Gnu FDL

Migrating from MySQL to MariaDB Galera Cluster (Node-by-Node In-Place)

> **Warning:**
>
> **Version Requirement**
>
> This guide provides instructions for migrating a **MySQL 8.0 Galera Cluster** to a **MariaDB Galera Cluster 11.4**. Ensure that your systems meet or exceed these version requirements before proceeding. Please refer to the Prerequisites for detailed version information.
>
> The **End-of-Life (EOL)** date for continued maintenance and regular binary releases of MySQL Galera Cluster will be **September 30, 2026**.

This guide outlines the procedure for migrating a live MySQL Galera Cluster to a MariaDB Galera Cluster. It follows the process for migrating a live MySQL Galera Cluster to a MariaDB Galera Cluster by replacing the binaries on each node sequentially. This “In-Place” method maintains cluster availability during the migration, although the cluster capacity will be reduced while individual nodes are being processed.

Prerequisites

- **Source:** MySQL Galera Cluster 8.0.43-26.4.24 or later.
- **Target:** MariaDB Community Server 11.4 (LTS) or later.

- **Library:** MariaDB Galera library 26.4.24 or latest stable edition.
- **Tools:**
  - `mysqldump` (installed with the server).
  - `mariadb-backup` (installed with MariaDB).
- **Access:** Root access to the operating system and the database.

> **Important:**
>
> **Cluster Backup**
>
> This migration involves uninstalling software and wiping data directories on live nodes. **A full cluster backup (using Xtrabackup or Volume Snapshot) is mandatory before proceeding.**

Incompatibility Check & Data Preparation

> **Warning:**
>
> **Perform these steps on the running MySQL Cluster BEFORE shutting down any nodes.**

Incompatibilities and Feature Differences

MariaDB maintains high levels of compatibility with MySQL, but you still need to check for incompatibilities and feature differences to select the most suitable MariaDB version compatible with your current MySQL server.

While MariaDB and MySQL share a common history, they have diverged significantly. You must “clean” the schema and configurations to ensure data compatibility before the migration begins.

1. Authentication Protocol

MySQL 8.0 defaults to `caching_sha256_password`. MariaDB does not support the `caching_sha256_password` authentication protocol in all released versions (which is cumbersome and not secure for in-house attacks, since clear-text passwords are available inside the server).

Support Status:

- Implemented via MDEV-9804 for version 12.1.1.
- Available in the CS release based on MDEV-37600.
- Already supported in 11.8 Enterprise Server via MENT-2359.
- Available in 11.4 Enterprise Server with the December 2025 release as a rebase of MDEV-37600.

> **Note:**
>
> **For MySQL 8.4 Users**
>
> Ensure `mysql_native_password=ON` is set in your configuration, or you will receive the error: `Plugin 'mysql_native_password' is not loaded`.

You can check which MySQL users are using SHA-256 or caching_sha256_password by executing:

```
SELECT user, plugin FROM mysql.user WHERE plugin LIKE "%sha%";
```

You can change the user to use a protocol compatible with both MySQL and MariaDB with:

```
ALTER USER user_name IDENTIFIED WITH mysql_native_password BY 'new_password';
```

2. JSON Column Conversion

MySQL stores JSON as a native binary type. MariaDB stores JSON as an alias for `LONGTEXT` with constraint checks. Galera replication may break if you attempt to replicate binary JSON objects from MySQL to MariaDB.

Check if you have tables that use the MySQL JSON type:

```
SELECT table_schema, table_name FROM information_schema.COLUMNS
WHERE data_type="JSON";
```

You can convert the JSON column to text with:

```
ALTER TABLE table_name MODIFY json_column LONGTEXT;
```

3. Encryption and Compression

Encryption and compression table options are very different in MySQL compared to MariaDB. Logical dump will naturally de-encrypt/de-compress table contents but syntax used for creating tables is not supported by MariaDB.

For example:

```
CREATE TABLE `t1` (
`c1` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci COMPRESSION='lz4'
```

leads to: `ERROR 1911 (HY000) at line 1174: Unknown option 'COMPRESSION'`

Therefore, encrypted/compressed tables need to be de-encrypted/de-compressed before starting the conversion, and then encrypted/compressed again afterwards using MariaDB syntax.

Detect encrypted tables:

```
SELECT table_schema, table_name, create_options FROM information_schema.TABLES WHERE
create_options LIKE "%ENCRYPT%";
```

Detect compressed tables:

```
SELECT table_schema, table_name, create_options FROM information_schema.TABLES WHERE
create_options LIKE "%COMP%";
```

Action: Run `ALTER TABLE` to remove the conflicting **`ENCRYPTION`** and **`COMPRESSION`** attributes. They can be re-enabled using MariaDB syntax after the full migration is complete.

4. General Tablespaces

MariaDB does not support MySQL general tablespaces. In MySQL, these are used to store tables in specific external files rather than the default data directory. During migration, this `TABLESPACE` argument is ignored.

Example Scenario: In MySQL, a user creates a custom tablespace file at a specific path and assigns a table to it:

```
-- MySQL: Creating a tablespace in a custom file location
mysql> CREATE TABLESPACE `ts1` ADD DATAFILE '/home/jan/galeradb/node1/ts1.ibd' Engine=InnoDB;

-- MySQL: Creating a table assigned to that tablespace
mysql> CREATE TABLE t1 (c1 INT PRIMARY KEY) TABLESPACE ts1;
```

If you examine the table definition in MySQL, you will see the tablespace assignment:

```
mysql> show create table t1;
| t1 | CREATE TABLE `t1` (
 `c1` int NOT NULL,
 PRIMARY KEY (`c1`)
) /*!50100 TABLESPACE `ts1` */ ENGINE=InnoDB ... |
```

Migration Behavior: When this table is migrated to MariaDB (via `mysqldump`), the `TABLESPACE 'ts1'` attribute is ignored.

- **Result:** The table `t1` will be created as a standard file-per-table in the default MariaDB data directory (e.g., `/var/lib/mysql/dbname/t1.ibd`).
- **Loss of Path:** It will **not** use the custom path `/home/jan/galeradb/node1/ts1.ibd`.sq

> **Warning:**
>
> During cluster migration make sure that **no new tablespaces are created** and **no new tables created inside a general tablespaces**.

5. XA Transactions

Ensure no XA transactions are in a `PREPARED` state, as these cannot be migrated cleanly.

```
XA RECOVER; -- Should return an empty set
```

Global Cluster Preparation

> **Note:**
>
> **Perform these steps on the running MySQL Cluster.**

1. Create the SST User (Crucial)

The `mysql.user` system table structure differs between vendors and **will not migrate automatically**. You must create the SST user now so it exists in the data stream before the switch.

```
-- Create user compatible with both engines
CREATE USER 'sst_user'@'localhost' IDENTIFIED WITH mysql_native_password BY 'strong_password';
GRANT ALL PRIVILEGES ON *.* TO 'sst_user'@'localhost';
FLUSH PRIVILEGES;
```

2. Configure InnoDB Settings

Ensure these settings are active on the MySQL cluster.

- **`innodb_file_per_table = 1`**
  - This is the default setting for MySQL and MariaDB. If this is not set, you must use `mysqldump` for the entire migration, as specific recommendations regarding tablespaces will not work.
- **`innodb_flush_log_at_trx_commit = 1`**
  - InnoDB writes the log buffer to the log file and flushes it to disk after every transaction commit. This is the safest option, ensuring no data loss in a crash, though it comes at the cost of slower performance due to frequent disk I/O.

3. Galera Provider Options

- **`gcache.size`**: Should be large enough to avoid SST (State Snapshot Transfer) when a node is temporarily out of the cluster. A larger cache increases the chance of a faster IST (Incremental State Transfer).
  - See: Customizing gcache.size

Migrating the First Node (“The Bridge”)

This is the most complex step. This node will bridge the gap between the MySQL cluster and the new MariaDB environment.

> **Step 1**
>
> **Isolate and Shutdown**
>
> 1. **Remove from Load Balancer:** Ensure no application traffic is hitting this node.
> 2. **Clean Shutdown Prep:**
>
>    ```
>    SET GLOBAL innodb_fast_shutdown = 0;
>    ```
>
> 3. **Stop the Service:**
>
>    ```
>    mysqladmin -u root -p shutdown
>    ```

> **Step 2**
>
> **Swap Binaries & Wipe Data**
>
> 1. **Uninstall MySQL:** Remove all MySQL server and client packages using your OS package manager (e.g., `apt remove`, `dnf remove`).
> 2. **Clean Data Directory:** One way is to move all files under `datadir` to a new directory.
>
>    ```
>    mv /var/lib/mysql /var/lib/mysql_backup_timestamp
>    mkdir /var/lib/mysql
>    chown mysql:mysql /var/lib/mysql
>    ```
>
> 3. **Install MariaDB:** Install `mariadb-server` and `mariadb-backup`.

Step 3

Configure MariaDB (`my.cnf`)

Update the configuration file to work with both MariaDB and MySQL.

Verify Configuration: Before starting the service, verify that your configuration file does not contain unsupported options (such as `regexp-time-limit`).

```
mariadbd --help --verbose > /tmp/log 2>&1
```

Check `/tmp/log` for any errors or warnings about unknown variables and adjust your configuration files if needed.

Required Settings: You **must** add these specific settings for the first node:

```
[galera]
# ... standard galera settings ...

# 1. IGNORE PROTOCOL MISMATCH
# Required to avoid application protocol mismatch messages during migration.
wsrep_provider_options="gcs.check_appl_proto=0;gcache.size=2G"

# 2. USE LOGICAL SST (First Node Only)
# wsrep_sst_method should be set to mysqldump for the first MariaDB node
# to join the MySQL cluster.
wsrep_sst_method=mysqldump

# 3. SST AUTHENTICATION
# Should contain user and password and this user should exist on both
# MySQL and MariaDB databases with sufficient access rights.
wsrep_sst_auth=sst_user:strong_password
```

Step 4

Start and Join

Start the MariaDB service:

```
systemctl start mariadb
```

When initiated, the node connects to the MySQL cluster and automatically triggers a `mysqldump` on a designated MySQL donor node. It then imports the SQL dump stream, effectively reconstructing the database in MariaDB's native format.

Step 5

Post-Join Upgrade

Once the node is `Synced` , run `mariadb-upgrade` to fix system tables.

> **Warning:**
>
> Manually re-create the `sst_user` (and other system users) if `mariadb-upgrade` reports issues, as the `mysql.user` table structure is not fully compatible.

Migrating Subsequent Nodes

Once the first node (Node A) is successfully running MariaDB, you can migrate Node B, Node C, etc.

Step 1

Shutdown and Replace

1. **Set Fast Shutdown:**

   ```
   SET GLOBAL innodb_fast_shutdown = 0;
   ```

   Required because undo/redo log formats may change between versions.

2. **Shutdown:** `mysqladmin -u root -p shutdown`
3. **Wipe & Install:** Uninstall MySQL, Wipe Datadir, Install MariaDB.

Step 2

Configure MariaDB

The configuration differs slightly for subsequent nodes. We switch back to **Physical Backups** for speed.

```
[galera]
# ... standard galera settings ...

# 1. Point to the MariaDB Donor
# After the first MariaDB node has successfully joined, this variable
# should be set on all other nodes to the value of wsrep_node_name
# on the first MariaDB node.
wsrep_sst_donor=Name_of_First_MariaDB_Node

# 2. Use Binary SST
# For other joining nodes, mariabackup value should be used.
wsrep_sst_method=mariabackup
wsrep_sst_auth=sst_user:strong_password

wsrep_provider_options="gcs.check_appl_proto=0"
```

Step 3

Start and Join

Start the service. The node will perform a binary snapshot transfer (`mariabackup`) from the first MariaDB node.

Repeat the above 3 steps for all remaining nodes.

Finalization

Once all nodes are running MariaDB:

1. **Cleanup Config:** Remove `gcs.check_appl_proto=0` and `wsrep_sst_donor` from `my.cnf` on all nodes. Reset `wsrep_sst_method` to `mariabackup` on the first node.
2. **Security:** Run `mariadb-secure-installation` to secure the root account and remove anonymous users.
3. **Features:** Re-enable encryption or compression using MariaDB-supported syntax if required.

Known Issues & Limitations

1. `mysql.user` Migration

The `mysql.user` table structure is not fully compatible and cannot be migrated automatically.

- **Impact:** The administrator must create necessary users for the MariaDB database manually.
- **Workaround:** In testing, using `--skip-table=mysql.user` during dump/restore and manually creating the necessary SST user was required.
- **Reference:** MDEV-33486

2. `regexp-time-limit` Parameter

MariaDB currently does not support the `regexp-time-limit` parameter.

- **Impact:** If this variable exists in your configuration or scripts, the server may fail to start or throw errors.
- **Reference:** MySQL Sysvar: regexp_time_limit, MDEV-37403

3. `wsrep_start_position` Mismatch

MariaDB does not understand the `wsrep_start_position` format provided by MySQL due to GTID formatting differences.

- **Example:** MySQL node will use a set `wsrep_start_position` like: `"60e63a7-734c-11f0-bd90-97669e82ccf4:2284/2281/11/0747b7c5-734c-11f0-a69b-00e04ca2f8fe"`
- **Reference:** MGL-57

4. Replication Direction

Replication from MariaDB → MySQL is **not supported**.

- **Impact:** During migration, there can be **no load** directed to the MariaDB nodes until the entire cluster is migrated.

5. Testing Scope

Currently, migration from 8.0.x has been verified to work on a simple `sysbench` database workload.

Required Code Changes (References)

For developers or those compiling from source, the following changes were relevant to this migration path:

- codership-mariadb-server Pull Request #519
- codership-mysql Pull Request #2062

This page is licensed: CC BY-SA / Gnu FDL

MySQL to MariaDB Migrator

> **Note:**
>
> **MariaDB tool.** The MySQL to MariaDB Migrator is proprietary MariaDB software, provided free to MariaDB customers and partners under approved usage terms. It is distributed from the MariaDB community downloads page.

The **MySQL to MariaDB Migrator** is a MariaDB tool that automates end-to-end migrations from MySQL to MariaDB in a repeatable, auditable way. It orchestrates schema migration, data transfer, user and privilege migration, and post-migration validation, and it drives the standard MariaDB client tools (`mariadb-dump` , the `mariadb` client, and the `mariadb-mtk` data-transfer engine) under a single launcher.

> **Note:**
>
> The migrator is currently in **beta**. All four migration modes have been exercised end-to-end against representative source and target pairs, including AWS RDS sources and MariaDB Cloud targets. Validate it in your own environment before a production migration.

The migrator complements the manual workflows in the MySQL to MariaDB Migration: The Master Guide: the Master Guide explains the migration paths and the compatibility considerations, while the migrator automates the dump, load, user-migration, and validation steps for you.

Supported Versions

- **Source:** MySQL 8.0 and 8.4 (some modes have version-specific requirements — see Migration Modes).
- **Target:** supported MariaDB Enterprise and Community editions.
- **MariaDB Cloud** is validated as a target for the Offline Copy (`staged`), Parallel Restartable Streaming Copy (`two_step`), and Serial Streaming Copy (`one_step`) modes.
- **Platforms:** the migrator is built and tested for Linux on x86-64 and ARM64.

Migration Modes at a Glance

The launcher presents four migration modes as a numbered menu. The internal identifier in parentheses is the canonical form used in configuration files, environment variables (`MODE=...`), and the command line (`--mode <id>`).

Mode	Internal ID	Type	Best For
Serial Streaming Copy	`one_step`	Offline	Smaller databases and standard maintenance windows
Parallel Restartable Streaming Copy	`two_step`	Offline	Larger datasets that need schema-then-parallel-data loading
Offline Copy	`staged`	Offline	Source and target not network-reachable, or a deferred / two-host load
Replication	`binlog`	Online	Low-downtime cutover with ongoing replication

Each mode links to its step-by-step guide above. For the variable-level detail on all four modes in one place, see Migration Modes.

Choose a Guide

If you are not sure which mode fits, match your situation to a guide:

Your situation	Guide
A small or medium database, a standard maintenance window is acceptable, and one host can reach both the source and the target	Serial Streaming Copy
The source and target cannot reach each other (air-gapped or separate networks), or you want a checkpoint between the dump and the load	Offline Copy
A large database where a single serial transfer is too slow, you can install the `mariadb-mtk` engine, and you can start from a clean target	Parallel Restartable Streaming Copy
Downtime must be minimal, the source uses `binlog_format=ROW` with no JSON columns, and you can perform a cutover	Replication

If more than one row fits, use this order: minimal downtime points to Replication (when the source qualifies); otherwise, if one host cannot reach both sides, use Offline Copy; otherwise choose Serial Streaming Copy for a small or medium database and Parallel Restartable Streaming Copy for a large one.

How the Tool Runs

When launched interactively, the migrator first offers two top-level choices:

- **Assess & Plan** — inspect the source and target, validate connectivity and compatibility, and produce an assessment report and a migration plan. No data is moved and nothing is written to the target.
- **Assess + Run** — assess the source, then proceed to the full migration, with confirmation steps between phases.

Preview a migration with **Assess & Plan** before committing to it; the assess and plan phases write their artifacts under `artifacts/` and leave the target untouched.

In This Section

Start with the reference pages to understand how the tool installs, runs, and handles users:

See Installation and First Run.

See Migration Modes.

See Application User Migration.

See Environment Variables.

Then follow the step-by-step guide for your mode:

See Migrate with Serial Streaming Copy.

See Migrate with Offline Copy.

See Migrate with Parallel Restartable Streaming Copy.

See Migrate with Replication.

Feedback

Join the MariaDB Community on Slack to share your feedback.

License

The MariaDB Software License Terms apply to all MariaDB Software unless otherwise stated. They do not alter the license terms of any free and open-source software (FOSS) or software subject to the Business Source License (BSL); see Section 7 of the MariaDB Software License Terms.

For additional legal information, see the MariaDB Terms.

See Also

An Easy Path from MySQL to MariaDB • blog post • 2026 • 4 minutes

Migrating from MySQL to MariaDB? Meet the MariaDB Migrator • video • 2026 • 8 minutes

This page is licensed: CC BY-SA / Gnu FDL

Installation and First Run

> **Note:**
>
> **MariaDB tool.** The MySQL to MariaDB Migrator is proprietary MariaDB software, provided free to MariaDB customers and partners under approved usage terms.

The MySQL to MariaDB Migrator is distributed as a release archive (`.tar.gz` or `.zip`) from the MariaDB community downloads page. Download the latest version (for example, `v1.3.1-beta`), extract it, and run the launcher — it bootstraps its own Python environment, so there is no manual setup beyond the prerequisites below.

```
tar -xzf mariadb-migrator-<version>.tar.gz
cd Mysql-to-MariaDB-Migration-<version>
./mariadb-migrator
```

The `.zip` archive is equivalent: `unzip` it, then change into the extracted directory and run `./mariadb-migrator`. The version embedded in the archive and directory names matches the release you download.

The data-transfer engine `mariadb-mtk` used by Parallel Restartable Streaming Copy is available from the same MariaDB community downloads page. See Prerequisites below.

Prerequisites

Required

- **MariaDB must be installed and running on the target host before you run the tool.** The migrator verifies the target version during preflight, but it does not install MariaDB.
- **Python 3.9 or later** on the host that runs the migrator. On the first run, the launcher creates a project-local virtual environment (`.venv`) and installs its Python dependencies into it automatically — no manual `pip install` is needed. On Debian and Ubuntu, install the venv module first: `sudo apt-get install -y python3-venv` .
- **The `mariadb` client** on the host that runs the migrator, used for connectivity, version, and database checks. If it is missing, the launcher detects your platform and offers to install it on the first run. You can also install it manually, for example with `dnf install mariadb` , `apt-get install mariadb-client` , `zypper install mariadb-client` , or `brew install mariadb` .
- **Network connectivity** from the host that runs the migrator to both the source MySQL server and the target MariaDB server. The exception is Offline Copy (`staged`) in its `dump_only` or `load_only` phase, which only needs connectivity to one side.
- For **Parallel Restartable Streaming Copy** (`two_step`), the `mariadb-mtk` engine must be installed by extracting its release archive:

  ```
  tar -xzf mariadb-mtk-<version>_<x86_64/arm_64>_linux.tar.gz
  ```

  It is available from the MariaDB community downloads page. Set the `SQLINESDATA_BIN` environment variable to the full path of the `mariadb-mtk` binary.

> **Note:**
>
> `pv` is recommended for live progress visibility but is optional. When it is missing, Offline Copy (`staged`) falls back to a 60-second file-size probe and Serial Streaming Copy (`one_step`) falls back to a 60-second heartbeat.

User Privileges

The tool expects valid privileges to already exist for the source and target admin users you provide (`SRC_ADMIN_USER` and `TGT_ADMIN_USER`). Each must be able to:

- Connect from the host that runs the migrator.
- Check, create, and drop target database objects as required by the workflow.
- Run dump and restore operations, and configure replication where applicable.

In practice this means admin-level privileges, including the ability to grant.

First Run

On the first run, the launcher:

1. Creates a project-local virtual environment at `./.venv` (unless one is already active) and installs the Python dependencies into it. Your system Python is never modified.
2. Detects the `mariadb` client and, if it is missing, shows the correct install command for your platform and offers to run it. The optional `pv` tool is offered the same way, without blocking the run.
3. Presents the interactive menu.

Later runs reuse `./.venv` and go straight to the menu.

Unattended and CI Hosts

The first-run prompts can be controlled with environment variables for unattended or CI hosts:

Variable	Effect
`MIGRATOR_ASSUME_YES=1`	Accept install prompts automatically.
`MIGRATOR_NO_SYSTEM_INSTALL=1`	Never run system installs; print the commands only.
`MIGRATOR_NO_AUTO_VENV=1`	Do not auto-create `.venv`; print the manual venv steps and exit.

Next Steps

- Migration Modes — choose and run a migration mode.
- Environment Variables — the variables required to run each mode.

This page is licensed: CC BY-SA / Gnu FDL

Migration Modes

> **Note:**
>
> **MariaDB tool.** The MySQL to MariaDB Migrator is proprietary MariaDB software, provided free to MariaDB customers and partners under approved usage terms.

The MySQL to MariaDB Migrator offers four migration modes. This page describes how to select a mode and the playbook for each one. For the variables each mode requires, see Environment Variables.

Selecting a Mode

Run the launcher interactively:

```
./mariadb-migrator
```

It first offers two top-level choices:

- **Assess & Plan** — inspect the source and target, validate connectivity and compatibility, and produce an assessment report and a migration plan. No data is moved.
- **Assess + Run** — assess the source, then proceed to the full migration (plan and run), with confirmation steps between phases.

It then presents the four modes as a numbered menu:

```
Select a migration mode:

  1) Serial Streaming Copy (mariadb-dump)            [OFFLINE]
  2) Parallel Restartable Streaming Copy (sqldata)   [OFFLINE]
  3) Offline Copy (mariadb-dump)                     [OFFLINE]
  4) Replication (binlog)                            [ONLINE]
```

The assess, plan, and run phases are also reachable non-interactively with the `--assess`, `--plan`, and `--run` flags; run `./mariadb-migrator --help` for the full list. Command-line flags bypass the menu.

> **Note:**
>
> The internal mode identifiers — `one_step`, `two_step`, `staged`, and `binlog` — are the canonical form used in the configuration file, the `MODE` environment variable, and the `--mode <id>` command-line option.

Serial Streaming Copy (`one_step`)

Best for smaller databases and standard maintenance windows. For a step-by-step walkthrough, see Migrate with Serial Streaming Copy.

- Uses `mariadb-dump` on the source and streams directly to the target `mariadb` client.
- A single pipe transfers tables sequentially, giving a predictable memory and network profile with no concurrency tuning.
- Supports a single database (`SRC_DB`) or multiple databases (`SRC_DBS="db1,db2"`).
- Strips `DEFINER` clauses by default to avoid permission errors on the target.
- Shows live progress through `pv` when available, and falls back to a 60-second heartbeat otherwise.

Parallel Restartable Streaming Copy (`two_step`)

Best for larger datasets or tighter migration windows. This mode requires the `mariadb-mtk` engine (see Installation and First Run). For a step-by-step walkthrough, see Migrate with Parallel Restartable Streaming Copy.

The mode runs in stages: a schema-only dump first, then a parallel data load through `mariadb-mtk`, a post-load row-count validation, and finally an object-finalization step for triggers, routines, and events. `mariadb-mtk` uses multiple concurrent worker sessions per database for the data phase.

Restart Behavior

Restartable, chunked loading is native to `mariadb-mtk` and on by default. Large tables (by default those with 1,000,000 or more rows) are loaded as parallel chunks, and transient errors are retried automatically within a run. Chunking requires an `AUTO_INCREMENT` column; tables without one transfer as a single stream.

`mariadb-mtk` does not resume a load across separate invocations. An interrupted or failed `two_step` run is restarted by dropping the target database or databases and re-running from a clean target. The launcher always starts a fresh run for this mode and reports a leftover target database as a conflict to drop.

Disabling Constraint Checks During the Data Load

If you hit foreign key or unique-constraint ordering errors during the parallel data load, run the following on the **target** MariaDB before starting the load:

```
SET GLOBAL FOREIGN_KEY_CHECKS=0;
SET GLOBAL UNIQUE_CHECKS=0;
```

After the object-finalization step completes, re-enable both:

```
SET GLOBAL FOREIGN_KEY_CHECKS=1;
SET GLOBAL UNIQUE_CHECKS=1;
```

This is a manual DBA step performed with an account that can set global variables — the migrator does not toggle these globals automatically.

Row-Count Validation

After the parallel data load completes successfully, the migrator validates row counts between the source and the target for each migrated database, using `mariadb-mtk`'s own validation command.

This is a post-load **report, not a gate**. The data transfer itself is the gate: a failed load stops the run before validation is reached, so reaching validation means every database transferred successfully. A row-count mismatch is recorded and surfaced, but does not fail the migration. Each database's per-table comparison and summary are written to both the per-database load log and the run log, and a concise verdict per database is echoed to the console. Set `MIGRATOR_SKIP_ROWCOUNT_VALIDATE=1` to skip the validation entirely.

Offline Copy (`staged`)

Best when the source and target are not directly network-reachable, or when you want a checkpoint between the dump and the load. For a step-by-step walkthrough, see Migrate with Offline Copy.

This mode is phase-driven through the `STAGED_PHASE` variable:

- `dump_and_load` (default) — full end-to-end migration from source to target.
- `dump_only` — dump from the source to a directory of files, then exit; the target is untouched.
- `load_only` — load existing dumps into the target; no source connection is needed, and the database list comes from the dump manifest.

Other characteristics:

- Per-database compressed dumps (`<db>.sql.gz`) are written under the run directory by default, or to a custom location set with `STAGED_DUMP_DIR` .
- A manifest tracks each database's SHA-256 hash, byte size, and approximate row count. By default each dump file's integrity is verified against this hash before it is loaded.
- Per-database dump parallelism is configurable (`STAGED_PARALLEL` , default `4`); loads run sequentially by default.
- A post-load finalize step compares the manifest against the target's `information_schema.tables` . It hard-fails on missing or empty databases and warns on row-count variance above a configurable threshold (`STAGED_FINALIZE_VARIANCE_PCT` , default 50%). Both source and target counts are InnoDB sampled estimates, so small variance is expected and does not indicate data loss.

Dump Now, Load Later on Another Host

```
# On the source host:
STAGED_PHASE=dump_only STAGED_DUMP_DIR=/mnt/transfer/dumps ./mariadb-migrator

# Transfer the directory:
scp -r /mnt/transfer/dumps target-host:/mnt/load/

# On the target host:
STAGED_PHASE=load_only STAGED_DUMP_DIR=/mnt/load/dumps ./mariadb-migrator
```

> **Warning:**
>
> Offline Copy does not capture writes made to the source during the dump. If downtime is unacceptable, use Replication (`binlog`) instead.

> **Warning:**
>
> Target connections negotiate TLS where the server requires it (for example, MariaDB Cloud), but server-certificate verification on the target side is not yet configurable: connections are encrypted in transit but not authenticated against a trusted CA. Source-side TLS verification works as expected through `SRC_SSL_MODE` .

Replication (`binlog`)

Best for low-downtime cutover. For a step-by-step walkthrough, see Migrate with Replication.

- Seeds the target from a consistent dump snapshot with embedded binlog coordinates, then starts MariaDB replication from the MySQL binary log using the `REPL_USER` and `REPL_PASS` credentials.
- Verifies replication thread health and lag after replication starts.
- For MySQL 8.4 sources, an upstream `mysqldump` 8.4 or later must be available, because 8.4 servers reject `SHOW MASTER STATUS` . The migrator detects this and fails fast at preflight.

> **Warning:**
>
> **Replication requires `binlog_format=ROW` on the source and does not support schemas that contain JSON columns.** Both conditions are checked upfront — at the launcher, the assessment, and preflight — and an operator with either configuration is blocked and routed to one of the offline modes, which are unaffected by these limitations. To use Replication, set `binlog_format = ROW` under `[mysqld]` in the source `my.cnf` and restart the source MySQL server.

Post-Load Optimizer Statistics

After a dump-and-restore, the target's optimizer statistics are empty or stale, which can lead to poor query plans until they are refreshed. The migrator can run `ANALYZE TABLE` across the migrated databases on the target as a final step, so the optimizer has accurate statistics immediately.

A prompt — `Run ANALYZE TABLE on target after load? (y/n)` — appears during setup, just after the application-user prompt. It defaults to `y` and is shown only for the modes where a fresh load occurs: Serial Streaming Copy (`one_step`), Parallel Restartable Streaming Copy (`two_step`), and Offline Copy (`staged`). It is not part of Replication (`binlog`), where replication keeps changing the tables after the seed.

- Set `ANALYZE_TARGET=0` (or answer `n`) to skip it.
- The step is skipped automatically when Offline Copy runs as `dump_only` , because no load happens on that host.
- A connection failure to the target stops the run; an `ANALYZE TABLE` error on an individual table is reported but does not fail the migration.
- A report listing the tables analyzed, status counts, and any per-table errors is written to the run's artifacts directory.

Notes and Limitations

- The migration fails by default if the target database already exists. Set `ALLOW_TARGET_DB_OVERWRITE=1` only when an overwrite is intentional. (For `staged` `load_only` , the database list comes from the manifest, and the same overwrite check applies before the load begins.)
- `root` source and target users are blocked by default; set `ALLOW_ROOT_USERS=1` to allow them.
- Offline Copy per-database load resume is not supported. If a multi-database load fails partway through, drop the partially loaded databases on the target and re-run with `STAGED_PHASE=load_only` .
- The migrator is built and tested for Linux on x86-64 and ARM64. Validate it in your target environment before production use.

This page is licensed: CC BY-SA / Gnu FDL

Application User Migration

> **Note:**
>
> **MariaDB tool.** The MySQL to MariaDB Migrator is proprietary MariaDB software, provided free to MariaDB customers and partners under approved usage terms.

When you answer `y` to the `Migrate application users? (y/n)` prompt, the migrator migrates application users and roles from the source MySQL server to the target MariaDB server during the run phase. This runs in every mode (`one_step` , `two_step` , `staged` , and `binlog`), and the assess phase produces a prediction report ahead of the run.

What Happens to Each User

User creation on the target is plugin-aware: the source authentication plugin determines how each user is migrated.

Source Plugin	Outcome on Target
`mysql_native_password` (with hash)	Migrated with the original password preserved. The hash is ported using MariaDB's `IDENTIFIED VIA ... USING` syntax.
`mysql_native_password` (no hash)	Created with the default password supplied at the prompt, with `PASSWORD EXPIRE` set so the user must change it on first login.
`caching_sha2_password`, `sha256_password`	Created with the default password and `PASSWORD EXPIRE`. These hash formats are not portable across the engine boundary.
`auth_socket`, `unix_socket`, `auth_pam`, `mysql_no_login`, anything else	**Skipped** — not created on the target. Configure these manually after the migration if needed.

Roles are detected separately and replayed on the target with `CREATE ROLE IF NOT EXISTS`. They are not migrated as users.

Grants

After each user is created, the migrator runs `SHOW GRANTS` against the source and replays each grant on the target. Grants that fail at replay time are dropped from the migration and listed in the per-user report. Common causes are:

- MySQL 8.0 / 8.4 dynamic privileges that MariaDB does not recognize, such as `APPLICATION_PASSWORD_ADMIN`, `SYSTEM_USER`, or `SET_ANY_DEFINER`.
- A grant on a database that does not exist on the target.
- A grant to a role that was not migrated (because the source user was identified as a role and skipped from the user loop).
- The target admin user lacking `GRANT OPTION` on the database being granted on.

Default Password Handling

The default password supplied at the `Default password for app users:` prompt is used for every user that cannot have its original password preserved — that is, all non-native-password users and native-password users without a hash. Every such user is also marked `PASSWORD EXPIRE`, so they must change their password on first login.

> **Warning:**
>
> The default password is recorded in plain text in the user-migration report. Treat that file as sensitive, and rotate the default password after the migration completes.

Reports

Two artifacts are produced when user migration is enabled:

- **`user_assessment_report.txt`**, written by the assess phase, predicts which users would be preserved, defaulted, or skipped. No writes to the target are performed.
- **`user_migration_report.txt`**, written by the run phase, records what actually happened: roles created, users preserved, users whose password was reset to the default, users skipped, users that failed to migrate, and the full list of grants that were dropped.

The two reports use a parallel structure, so an `Assess + Run` migration can diff them afterward to confirm the prediction matched the outcome.

Limitations and Known Behavior

- Replication, monitoring, and backup-tool accounts (for example `repl_user`, `pmm_*`, or `xtrabackup`) are migrated as application users in this release. Clean them up manually on the target after the migration.

- MySQL 8.4 sources default to `caching_sha2_password` and ship with `mysql_native_password` disabled, so most or all users on a fresh 8.4 source land on the default-password path. Plan a password-rotation pass before re-enabling application traffic on the target.
- The dump phase may replay user-related rows from `mysql.user` as part of the data load, which can produce duplicate or conflicting entries alongside what the user-migration step created. If the user set looks off, review `SELECT user, host, plugin, is_role FROM mysql.user` on the target after the migration.

This page is licensed: CC BY-SA / Gnu FDL

Environment Variables

> **Note:**
>
> **MariaDB tool.** The MySQL to MariaDB Migrator is proprietary MariaDB software, provided free to MariaDB customers and partners under approved usage terms.

The migrator reads its configuration from environment variables (or the configuration file it writes from your interactive answers). This page lists the variables **required** to run each mode. A complete reference of every variable, with defaults and descriptions, ships with the tool.

> **Note:**
>
> The interactive launcher prompts for any required input that is not already set, so you do not have to export these variables by hand. Set them explicitly for scripted or unattended runs.

Common Variables

Most modes share the same source and target connection variables:

Variable	Purpose
`SRC_HOST`, `SRC_PORT`	Source MySQL host and port.
`SRC_ADMIN_USER`, `SRC_ADMIN_PASS`	Source admin credentials.
`SRC_DB` or `SRC_DBS`	A single source database, or a comma-separated list.
`TGT_HOST`, `TGT_PORT`	Target MariaDB host and port.
`TGT_ADMIN_USER`, `TGT_ADMIN_PASS`	Target admin credentials.
`SRC_SSL_MODE`	Optional source TLS mode (`DISABLED`, `PREFERRED`, `REQUIRED`, `VERIFY_CA`, or `VERIFY_IDENTITY`) for TLS-required sources.

Serial Streaming Copy (`one_step`)

Requires the common source and target variables above. Optional:

- `ANALYZE_TARGET` — `1` by default; runs `ANALYZE TABLE` on the target after the load to refresh optimizer statistics. Set to `0` to skip. (Applies to `one_step`, `two_step`, and `staged`.)

Parallel Restartable Streaming Copy (`two_step`)

Requires the common source and target variables. Optional:

- `SQLINESDATA_BIN` — full path to the `mariadb-mtk` binary; auto-detected from a `sqldata` or `sqlinesdata` binary on `PATH`.
- `MIGRATOR_SKIP_ROWCOUNT_VALIDATE` — set to `1` to skip the post-load row-count validation.

Replication (`binlog`)

Requires the common source and target variables, plus:

- `REPL_USER` , `REPL_PASS` — replication credentials.

Optional:

- `SRC_BINLOG_FILE` , `SRC_BINLOG_POS` — binlog coordinates; auto-captured during the seed if not set.
- `BINLOG_COORD_FILE` — where captured coordinates are written (default: `artifacts/binlog_coords.env`).
- `BINLOG_MAX_LAG_SECS` — replication lag threshold in seconds (default: `30`).

Offline Copy (`staged`)

The phase selector and connection requirements depend on the phase:

- `STAGED_PHASE` — `dump_and_load` (default), `dump_only` , or `load_only` .
- Source variables are required for `dump_and_load` and `dump_only` .
- Target variables are required for `dump_and_load` and `load_only` .

Dump and load tuning (all optional):

Variable	Default	Purpose
`STAGED_DUMP_DIR`	run directory	Dump location; required for `load_only` to point at the manifest directory.
`STAGED_COMPRESS`	`1`	Gzip-compress the dump.
`STAGED_PV`	`1`	Show a progress meter through `pv` or the fallback probe.
`STAGED_PARALLEL`	`4`	Concurrent per-database dumps.
`STAGED_LOAD_PARALLEL`	`1`	Concurrent per-database loads.
`STAGED_VERIFY_CHECKSUM`	`1`	Verify each dump file's integrity before load.
`STAGED_DISK_HEADROOM_FACTOR`	`2`	Multiplier on the source data size for the free-space check.
`STAGED_FINALIZE_VARIANCE_PCT`	`50`	Row-count variance threshold above which finalize warns.
`STAGED_CONFIRM_OFFLINE`	—	Set to bypass the interactive offline-acknowledgment prompt for non-interactive runs.

Run-Control Variables

These apply across modes:

Variable	Purpose
`MODE`	Selects the mode (`one_step` , `two_step` , `staged` , or `binlog`) for non-interactive runs.
`ALLOW_TARGET_DB_OVERWRITE`	Set to `1` to allow migrating into an existing target database.
`ALLOW_ROOT_USERS`	Set to `1` to allow `root` as a source or target admin user.
`FORCE_NEW_RUN`	Set to `1` to force a fresh run instead of resuming a previous one.

This page is licensed: CC BY-SA / Gnu FDL

Migrate with Serial Streaming Copy

This guide walks through a complete MySQL to MariaDB migration in **Serial Streaming Copy** (`one_step`) mode. You preview the migration with Assess & Plan, run it, and verify the result on the target.

Serial Streaming Copy is the most direct mode: it pipes `mariadb-dump` from the source straight into the `mariadb` client on the target, one table at a time. It needs no extra data-transfer engine and no replication setup, which

makes it the right choice for a first migration and for smaller databases that can move within a standard maintenance window.

> **Note:**
>
> The migrator is in **beta**. Run this procedure against a non-production target first, and validate the result before you migrate a production database.

This guide uses the `sakila` sample database as the example. Substitute your own database name wherever `sakila` appears.

> **Note:**
>
> **Following along with the sample?** `sakila` is MySQL's official sample database. To set it up on your source server, download the archive from the MySQL example databases page, extract it, and load the two files:
>
> ```
> mysql -u root -p < sakila-schema.sql   # tables, views, routines, triggers
> mysql -u root -p < sakila-data.sql     # data
> ```
>
> See the Sakila documentation for the full schema. If you are migrating your own database instead, skip this step.

Before You Begin

You need a source **MySQL 8.0 or 8.4** server with the database you want to migrate, and a target **MariaDB** server that is already installed and running. The migrator checks the target version but does not install MariaDB. The host running the migrator needs network connectivity to both servers.

You also need **admin credentials** on both servers that can connect from the migrator host, create and drop databases, and run dump and restore operations. Avoid `root`; the migrator blocks `root` by default.

Finally, the migrator host needs **Python 3.9 or later** and the **`mariadb` client**. The launcher bootstraps its own Python environment and offers to install the client if it is missing. On Debian or Ubuntu, install the venv module first with `sudo apt-get install -y python3-venv`. The optional `pv` tool improves progress visibility during the transfer but is not required.

Step 1: Download and Start the Migrator

See Installation and First Run for details on downloading and installing the MariaDB Migrator.

On the first run, the launcher creates a project-local Python environment (`.venv`), installs its dependencies into it, and checks for the `mariadb` client. Your system Python is never modified, and later runs reuse `.venv` and go straight to the menu.

```
--------------------------------------------------------------------
 Welcome to the MySQL to MariaDB Migration Tool
 Tool Version: 1.3.1-beta (Build 20260622)
 Supported Sources: MySQL 8.0, 8.4
--------------------------------------------------------------------
==> Validating execution environment...
==> No active virtualenv detected. Creating one at ./.venv ...
Installing missing dependencies: typer click rich PyYAML
Successfully installed PyYAML-6.0.2 click-8.1.7 rich-13.7.1 typer-0.12.3 ...
==> Environment ready.
==> Checking system dependencies...
==> Database client found: /usr/bin/mariadb
==> pv found (dump progress will be shown).
==> System dependencies ready.

What would you like to do?
  1) Assess & Plan    Inspect source and target, validate connectivity and
                      compatibility, and produce an assessment report and the
                      migration plan. No data is moved.
  2) Assess + Run     Assess the source, then proceed to the full migration
                      (plan + run, with confirm steps between phases).
  q) Quit
```

Step 2: Preview with Assess & Plan

Choose **1) Assess & Plan** for the preview. This phase never writes to the target. The migrator prompts for the connection details it needs:

Prompt	Example value
Source host / port	`mysql.example.com` / `3306`
Source admin user / password	`migadmin` / (entered at the prompt)
Source database	`sakila`
Target host / port	`mariadb.example.com` / `3306`
Target admin user / password	`migadmin` / (entered at the prompt)

Then select mode **1) Serial Streaming Copy (mariadb-dump) [OFFLINE]**. The assess phase checks connectivity and source-to-target compatibility, inventories the schema, and writes its reports under `artifacts/assess_<timestamp>/`.

```
== Precheck runner ==
Host: mysql.example.com  Port: 3306  User: migadmin
---- mysql_version ----
8.0.46
---- json_columns ----
(no rows)
---- engines_summary ----
InnoDB  59
Precheck complete.
==> Assess application users on source
Roles found on source                : 0
Users with portable native password  : 0
Users requiring default password     : 0
Users that would be SKIPPED          : 0
ASSESSMENT: PASS (see artifacts/report.json and run.log)
```

Review the reports before you run the migration. The **assessment report** confirms connectivity and compatibility. The **`user_assessment_report.txt`** predicts which application users would be migrated with their password preserved, which would be reset to a default password, and which would be skipped. In this example the `sakila` source has no application accounts of its own, so the user counts are zero; on a real database this report lists exactly which users fall into each category. For how the migrator classifies users, see Application User Migration.

Step 3: Run the Migration

Start the launcher again and choose **2) Assess + Run**, then select **1) Serial Streaming Copy** again. The migrator re-runs the assessment, then prompts for a few decisions before it moves any data.

The **`Migrate application users? (y/n)`** prompt controls whether application users and roles are migrated along with the data. If you answer `y`, the migrator also asks for a **`Default password for app users:`** used for any user whose original password cannot be carried over. The **`Run ANALYZE TABLE on target after load? (y/n)`** prompt defaults to `y`; it refreshes the target's optimizer statistics after the load so queries get good plans immediately.

The migrator then streams the data. Because Serial Streaming Copy is a logical dump and restore, it carries the full schema across: tables, views, stored procedures, functions, triggers, and events. It strips `DEFINER` clauses by default so those objects restore without permission errors on the target.

Output is captured to a run log rather than streamed to your terminal, and each phase prints a `tail -f` hint. To watch live progress, open a second shell and run `tail -f artifacts/run_one_step_<timestamp>/run.log`.

```
==> One-step migration (mariadb-dump | mariadb)
Source MySQL: 8.0.46  Dump tool: mariadb-dump
Source: mysql.example.com:3306  DB: sakila
Target: mariadb.example.com:3306
0:00:00 [ <=>                                                                ]
One-step migration completed.
==> Post-load ANALYZE TABLE on target (refresh optimizer statistics)
    [done] sakila: 16 table(s) in 0s
RUN: PASS
```

The `pv` line is the live transfer heartbeat. For `sakila` the transfer finishes in under a second, so you see a single heartbeat; on a larger database `pv` prints a progress line every 10 seconds, and a 60-second heartbeat is used only when `pv` is not installed.

> **Note:**
>
> By default the migration stops if the target database already exists, so an accidental overwrite cannot happen. To migrate into an existing database, set `ALLOW_TARGET_DB_OVERWRITE=1` before the run.

Step 4: Verify the Result

When the run finishes, confirm the migration on the target. The run artifacts under `artifacts/run_one_step_<timestamp>/` include **`user_migration_report.txt`** (what happened to each user) and **`analyze_target_report.txt`** (the tables analyzed and any per-table errors).

```
# user_migration_report.txt
Roles created on target              : 0
Users migrated with original password: 0
Users migrated with default password : 0  <- PASSWORD EXPIRE set
Grants dropped (incompatible)        : 0

# analyze_target_report.txt
Databases analyzed   : 1  (sakila)
Tables analyzed      : 16
  status OK          : 16
Overall              : PASS
```

Then spot-check the data on the target and compare against the source:

```
mariadb -h mariadb.example.com -u migadmin -p \
  -e "SELECT COUNT(*) FROM sakila.film; SHOW TABLES FROM sakila;"
```

```
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
actor
address
category
... (16 base tables and 7 views)
```

Object fidelity is the important check for Serial Streaming Copy. On the migrated target, `sakila` has **16 base tables, 7 views, 6 stored routines, and 6 triggers**, matching the source, and the stored functions and procedures execute and return the same results as the source. Confirm the stored objects with:

```
mariadb -h mariadb.example.com -u migadmin -p -N -e "
  SELECT COUNT(*) FROM information_schema.routines WHERE routine_schema='sakila';
  SELECT COUNT(*) FROM information_schema.triggers WHERE trigger_schema='sakila';"
```

After the Migration

Rotate any default passwords. A user that could not keep its original password was created with the default password you supplied and marked `PASSWORD EXPIRE`. That default is recorded in plain text in `user_migration_report.txt`, so treat the file as sensitive and rotate the password before re-enabling application traffic. This is common when migrating from MySQL 8.4, which defaults to `caching_sha2_password`.

Clean up infrastructure accounts. Replication, monitoring, and backup accounts are migrated as application users in this release; remove any that do not belong on the target. Once you have verified the data and credentials, repoint your application at the target MariaDB server.

Run Without Prompts

For a scripted, non-interactive run, set the connection details as environment variables and the launcher skips the prompts:

```
MODE=one_step \
SRC_HOST=mysql.example.com SRC_PORT=3306 SRC_ADMIN_USER=migadmin SRC_ADMIN_PASS=... SRC_DB=sakila \
TGT_HOST=mariadb.example.com TGT_PORT=3306 TGT_ADMIN_USER=migadmin TGT_ADMIN_PASS=... \
./mariadb-migrator --run
```

See Environment Variables for the full list.

Other Modes

If Serial Streaming Copy does not fit your situation, see the migrator overview to choose another mode: Offline Copy for hosts that cannot reach each other, Parallel Restartable Streaming Copy for large databases, or Replication for a low-downtime cutover.

This page is licensed: CC BY-SA / Gnu FDL

Migrate with Offline Copy

This guide walks through a complete MySQL to MariaDB migration in **Offline Copy** (`staged`) mode. You dump the source database to compressed files with a manifest, move that directory to where the target lives, and load it there.

Offline Copy is the right mode when the source and target cannot reach each other directly, for example across an air gap or between separate networks, or when you simply want a verifiable checkpoint between the dump and the load. It uses the same `mariadb-dump` and `mariadb` tooling as Serial Streaming Copy, but writes to disk in between instead of streaming through a single pipe.

> **Note:**
>
> The migrator is in **beta**. Run this procedure against a non-production target first, and validate the result before you migrate a production database.

This guide uses the `sakila` sample database as the example. Substitute your own database name wherever `sakila` appears.

> **Note:**
>
> **Following along with the sample?** `sakila` is MySQL's official sample database. Download it from the MySQL example databases page, extract it, and load `sakila-schema.sql` then `sakila-data.sql` into your source server.

Before You Begin

You need a source **MySQL 8.0 or 8.4** server and a target **MariaDB** server already installed and running. Unlike the online modes, Offline Copy does not need a single host that can reach both: the dump phase needs only the source, and the load phase needs only the target. You do need a directory you can move between the two sides, with free space for the compressed dump.

You also need **admin credentials** on each server that can connect from the host running that phase and can dump or restore databases. Avoid `root`; the migrator blocks it by default.

The host that runs each phase needs **Python 3.9 or later** and the **`mariadb` client**. The launcher bootstraps its own Python environment and offers to install the client if it is missing. The optional `pv` tool improves progress visibility.

Step 1: Download and Start the Migrator

See Installation and First Run for details on downloading and installing the MariaDB Migrator.

Step 2: Understand the Phases

Offline Copy is driven by the `STAGED_PHASE` setting, which selects which part of the migration runs:

- `dump_and_load` (default) runs the whole migration on a single host that can reach both the source and the target.
- `dump_only` dumps from the source to a directory of files and then exits. The target is untouched, and only source connectivity is needed.
- `load_only` loads existing dump files into the target. Only target connectivity is needed, and the database list comes from the manifest in the dump directory.

The two-host workflow below uses `dump_only` and `load_only`. If one host can reach both servers, you can run `dump_and_load` instead and skip the transfer step.

Step 3: Dump on the Source

On the host that can reach the source, run the dump phase, pointing `STAGED_DUMP_DIR` at a directory you can move. `STAGED_CONFIRM_OFFLINE=1` acknowledges without an interactive prompt that writes to the source during the dump are not captured.

```
STAGED_PHASE=dump_only STAGED_DUMP_DIR=/mnt/transfer/dumps \
  STAGED_CONFIRM_OFFLINE=1 ./mariadb-migrator
```

The dump phase reports each database as it finishes and prints a summary:

```
==> Staged dump (mariadb-dump | pv | gzip > file, per database)
Source MySQL : 8.0.46
Dump tool    : mariadb-dump
Compress     : gzip
Databases    : sakila

==> Dumping: sakila
   sakila: 0:00:00 [================>          ]  51%
    [done] sakila: 702KB, 47685 approx rows, 0s

==> Staged dump complete.
    Databases : 1
    Total size: 702KB
    Manifest  : /mnt/transfer/dumps/manifest.txt
```

The directory now holds one compressed file per database (`sakila.sql.gz`) and a `manifest.txt` that records each database's SHA-256, byte size, and approximate row count:

```
# staged-migration-manifest v1
# source_host: mysql.example.com
# source_version: 8.0.46
# dump_tool: mariadb-dump
# database  filename    sha256  size_bytes  approx_rows
sakila  sakila.sql.gz   c1040adb994b9a6c85a1c777b2abba7ea19d5dd9f86cc53e5f3b09b971fc8720    718005  47685
```

Step 4: Transfer the Dump

Move the dump directory to the target side by whatever transport your environment allows: `scp` , a shared volume, removable media, or an object-store bucket. The manifest travels with the dump files and is used to verify them at load time.

```
scp -r /mnt/transfer/dumps target-host:/mnt/load/
```

Step 5: Load on the Target

On the host that can reach the target, run the load phase against the transferred directory:

```
STAGED_PHASE=load_only STAGED_DUMP_DIR=/mnt/load/dumps \
  STAGED_CONFIRM_OFFLINE=1 ./mariadb-migrator
```

The load phase reads the manifest, verifies each file's SHA-256 before loading it, loads each database, refreshes optimizer statistics with `ANALYZE TABLE` , and finalizes:

```
==> Preflight checks (staged)
Phase: load_only
Manifest: 1 database(s) — sakila

==> Loading: sakila (file: sakila.sql.gz)
    checksum: OK
    [done] sakila: loaded 702KB in 2s

==> Staged load complete.
```

Step 6: Verify the Result

After the load, a finalize step compares the manifest against the target's `information_schema.tables` . It hard-fails on a missing or empty database and warns on row-count variance above the threshold:

```
==> Staged finalize (post-load sanity counts and manifest verification)
Per-DB verification:
  Database                 Exists    Tables  Source rows     Target rows     Variance
  ------------------------ --------- ------- --------------- --------------- --------
  sakila                   OK        16      ~47685          ~47375          -0.7%

==> Staged finalize complete.
    Databases verified: 1
```

The -0.7% variance is expected. Both row counts are InnoDB sampled estimates from `information_schema`, not exact `COUNT(*)` results, so small differences are normal and do not indicate data loss. The checksum verified during load is the authoritative file-integrity check. For an exact check, run `SELECT COUNT(*)` on a few tables on both servers.

After the Migration

Review the run reports under `artifacts/run_staged_<timestamp>/`, rotate any default passwords recorded in the user migration report, and remove replication, monitoring, or backup accounts that do not belong on the target. Repoint your application at the target MariaDB server once you have verified the data and credentials.

Known Limitations

> **Warning:**
>
> Offline Copy does not capture writes made to the source during the dump. For a consistent cutover, stop application traffic to the source before the dump and resume it on the target only after the load completes. If downtime is unacceptable, use Replication instead.

> **Warning:**
>
> Target-side TLS certificate verification is not yet configurable. Connections to TLS-required targets, such as MariaDB Cloud, are encrypted but not verified against a trusted CA. Source-side TLS works as expected through `SRC_SSL_MODE`.

Per-database load resume is not supported. If a multi-database load fails partway, drop the partially loaded databases on the target and re-run with `STAGED_PHASE=load_only`.

Reference

The dump phase accepts tuning variables including `STAGED_PARALLEL` (concurrent per-database dumps, default 4), `STAGED_COMPRESS` (gzip on by default), and `STAGED_FINALIZE_VARIANCE_PCT` (the warning threshold, default 50). See Environment Variables for the full list.

Other Modes

If Offline Copy does not fit your situation, see the migrator overview to choose another mode: Serial Streaming Copy for a single-pipe transfer when both servers are reachable, Parallel Restartable Streaming Copy for large databases, or Replication for a low-downtime cutover.

This page is licensed: CC BY-SA / Gnu FDL

Migrate with Parallel Restartable Streaming Copy

This guide walks through a complete MySQL to MariaDB migration in **Parallel Restartable Streaming Copy** (`two_step`) mode. The migrator dumps the schema first, loads the data in parallel through the `mariadb-mtk` engine, validates row counts, and finalizes the remaining objects.

This mode is for larger databases where a single serial transfer is too slow. It dumps the schema, then loads table data through multiple concurrent worker sessions, then applies triggers, routines, and events at the end.

> **Note:**
>
> The migrator is in **beta**. Run this procedure against a non-production target first, and validate the result before you migrate a production database.

This guide uses the `employees` sample database, which is large enough (about 3.9 million rows across six tables) to show the parallel load and the row-count validation doing real work. Substitute your own database name wherever `employees` appears.

> **Note:**
>
> **Following along with the sample?** `employees` is MySQL's sample database for larger datasets, from the datacharmer/test_db project. Clone it and load it into your source with `mysql -u root -p < employees.sql` from inside the repository directory.

Before You Begin

You need a source **MySQL 8.0 or 8.4** server with the database you want to migrate, and a target **MariaDB** server already installed and running. The host running the migrator needs network connectivity to both servers.

You need **admin credentials** on both servers that can connect from the migrator host and create, drop, dump, and restore databases. Avoid `root`; the migrator blocks it by default. The migrator host needs **Python 3.9 or later** and the **`mariadb` client**.

The target must be **clean** for this mode: Parallel Restartable Streaming Copy does not resume across separate invocations, so if a target database already exists from a previous attempt, drop it before you start.

The mariadb-mtk engine

This mode requires the `mariadb-mtk` data-transfer engine. The other three modes do not use it. See the "Required" section of the Installation and First Run page for details on downloading and installing mariadb-mtk.

Step 1: Download and Start the Migrator

See Installation and First Run for details on downloading and installing the MariaDB Migrator.

Step 2: Preview with Assess & Plan

Choose **1) Assess & Plan**, supply the source and target connection details when prompted, and select mode **2) Parallel Restartable Streaming Copy (sqldata) [OFFLINE]**. This phase checks connectivity and compatibility and writes its reports under `artifacts/assess_<timestamp>/` without touching the target. Review the assessment before running.

Step 3: Run the Migration

Start the launcher again, choose **2) Assess + Run**, and select Parallel Restartable Streaming Copy. The run executes four steps in order: a schema-only dump, the parallel data load, row-count validation, and object finalization.

The schema is dumped in two parts. The pre-data DDL (tables, indexes, foreign keys) is loaded first, with triggers, routines, and events held back. Those post-data objects are applied during finalization rather than before the load, so a trigger cannot fire and re-insert rows while data is still streaming in.

Watch live progress with `tail -f artifacts/run_two_step_<timestamp>/run.log` from a second shell.

How the parallel load behaves

The engine transfers the database with several concurrent sessions (the `-ss` value, default 4), working on different tables at once. In this run of `employees`, four sessions moved all six tables and 3.9 million rows transferred in about three seconds:

```
Transferring database (4 concurrent sessions):
employees.departments  - Started (1 of 6, 1 chunk, session 1)
employees.dept_emp     - Started (2 of 6, 1 chunk, session 2)
employees.employees    - Started (4 of 6, 1 chunk, session 3)
employees.salaries     - Started (5 of 6, 1 chunk, session 4)
employees.titles       - Started (6 of 6, 1 chunk, session 1)
  Rows written:  3919015 (3.9M rows)
  Transfer time: 3.4 sec (1144238 rows/s, 134.9 MB, 39.4 MB/s)
```

That parallelism is across tables. The engine can also split a single large table into parallel chunks, but that is a separate, opt-in behavior: it is gated by `large_tables_parallel` (off in the shipped `sqldata.cfg`) and additionally requires the table to have an `AUTO_INCREMENT` column. The `employees` tables use composite primary keys with no `AUTO_INCREMENT`, so each table, `salaries` included, transfers as a single stream within its session, which is what the `1 chunk` markers show. To split one very large table across sessions, set `large_tables_parallel=yes` and ensure the table has an `AUTO_INCREMENT` column.

Row-count validation

After the load, the tool validates source and target row counts per database using the engine's own validate command. This is a report, not a gate: the data transfer is the gate, so reaching validation means every database loaded successfully. A mismatch is recorded and surfaced but does not fail the migration.

```
===== row-count validation: employees =====
  Total number of tables:         6
    With the different row count: 0
  Total number of rows:           3919015 in source, 3919015 in target
[employees] row counts OK (all tables equal)
==> Row-count validation: all selected database(s) match
```

Set `MIGRATOR_SKIP_ROWCOUNT_VALIDATE=1` to skip this step.

Step 4: Verify the Result

When the run finishes, confirm the migration on the target. In this run the target `employees` matched the source exactly: 6 base tables and 2 views, with 300024 rows in `employees` and 2844047 in `salaries`. The views are part of the schema dump and are created before the data load; only triggers, routines, and events are applied during finalize.

```
mariadb -h mariadb.example.com -u migadmin -p -t -e "
  SELECT table_type, COUNT(*) FROM information_schema.tables
   WHERE table_schema='employees' GROUP BY table_type;
  SELECT COUNT(*) AS salaries FROM employees.salaries;"
```

Restart Behavior

> **Warning:**
>
> `mariadb-mtk` does not resume a load across separate invocations. An interrupted or failed run is restarted by dropping the target database, then re-running from a clean target. The launcher always starts a fresh run for this mode and reports a leftover target database as a conflict to drop.

Chunked loading is still restartable within a run: transient errors are retried automatically (`restart_attempts`, default 10). The no-resume limitation applies only across separate invocations.

Constraint Checks During Load

By default the tool disables foreign-key checks on the target during the parallel load (`TWO_STEP_KEEP_FK_CHECKS=0`) so cross-table load order does not cause failures. If you keep them on and hit foreign-key or unique-constraint ordering errors, disable the checks on the target before the load and re-enable them after finalize. The tool does not toggle these for you in that case.

```
-- Before the load:
SET GLOBAL FOREIGN_KEY_CHECKS=0;
SET GLOBAL UNIQUE_CHECKS=0;
-- After finalize:
SET GLOBAL FOREIGN_KEY_CHECKS=1;
SET GLOBAL UNIQUE_CHECKS=1;
```

After the Migration

Review the run reports under `artifacts/run_two_step_<timestamp>/` , rotate any default passwords from the user migration report, and remove infrastructure accounts that do not belong on the target. Repoint your application at the target MariaDB server once you have verified the data and credentials.

Other Modes

If Parallel Restartable Streaming Copy does not fit your situation, see the migrator overview to choose another mode: Serial Streaming Copy for a smaller database, Offline Copy for hosts that cannot reach each other, or Replication for a low-downtime cutover.

This page is licensed: CC BY-SA / Gnu FDL

Migrate with Replication

This guide walks through a complete MySQL to MariaDB migration in **Replication** (`binlog`) mode. The migrator seeds the target from a consistent snapshot with embedded binlog coordinates, starts MariaDB replication from the MySQL binary log, and verifies it. The target then stays current with the source until you perform a short cutover.

This is the mode to use when downtime must be minimal. Unlike the offline modes, it does not finish in a single pass; it establishes ongoing replication that you cut over when you are ready.

> **Note:**
>
> The migrator is in **beta**. Run this procedure against a non-production target first, and validate the result before you migrate a production database.

This guide uses the `sakila` sample database as the example. Substitute your own database name wherever `sakila` appears.

> **Note:**
>
> **Following along with the sample?** `sakila` is MySQL's official sample database. Download it from the MySQL example databases page, extract it, and load `sakila-schema.sql` then `sakila-data.sql` into your source server.

Before You Begin

You need a source **MySQL 8.0 or 8.4** server and a target **MariaDB** server already installed and running. The host running the migrator needs network connectivity to both, and the target MariaDB must be able to reach the source MySQL directly, because the replica connects to the source to read its binary log.

You need **admin credentials** on both servers (not `root` , which is blocked by default), plus a **replication user** the migrator will use; you supply its name and password as `REPL_USER` and `REPL_PASS` , and the migrator creates it on

the source if it does not exist. The migrator host needs **Python 3.9 or later** and the `mariadb` **client**.

Source requirements

> **Warning:**
>
> Replication requires `binlog_format=ROW` on the source and does not support schemas that contain JSON columns. Both conditions are enforced at the launcher, the assessment, and preflight, so a source with either is blocked upfront and routed to an offline mode. To use Replication, set `binlog_format = ROW` under `[mysqld]` in the source `my.cnf` and restart the source MySQL server.

A **MySQL 8.4** source additionally needs an upstream `mysqldump` 8.4 or later available, because 8.4 servers reject `SHOW MASTER STATUS`. An **8.0** source uses `mariadb-dump` directly and needs no extra client, which makes 8.0 the simpler source for this mode.

Step 1: Download and Start the Migrator

See Installation and First Run for details on downloading and installing the MariaDB Migrator.

On the first run, the launcher creates a project-local Python environment (`.venv`) and checks for the `mariadb` client. Your system Python is never modified.

Step 2: Preview with Assess & Plan

Choose **1) Assess & Plan**, supply the source and target connection details and the replication user when prompted, and select mode **4) Replication (binlog) [ONLINE]**. The assessment inventories the source and checks compatibility before anything is written. For Replication the decisive check is JSON columns: `sakila` has none, so the source clears the gate (the `binlog_format=ROW` requirement is confirmed when the migration runs).

```
== Precheck runner ==
Host: mysql.example.com  Port: 3306  User: migadmin
---- mysql_version ----
8.0.46
---- json_columns ----
(no rows)
---- engines_summary ----
InnoDB  59
Precheck complete.
ASSESSMENT: PASS (see artifacts/report.json and run.log)
```

If the source has a JSON column, the assessment stops and lists the offending columns, then re-prompts for the database name(s) so you can drop or exclude the offending database and retry. If the JSON columns are ones you need to migrate, choose one of the offline modes instead.

Step 3: Run the Migration

Start the launcher again, choose **2) Assess + Run**, and select Replication. The migrator first re-checks source compatibility with a Replication-specific preflight, then runs three steps: seed, start replication, and verify. Watch progress with `tail -f artifacts/run_binlog_<timestamp>/run.log` .

```
==> Preflight checks (binlog)
Source MySQL: 8.0.46 (using: SHOW MASTER STATUS;)
Checking source schemas for JSON columns...
Checking source binary logging...
Checking source binlog format...
Checking source master/binary-log status visibility...
Preflight complete.
```

Seed the target

A consistent snapshot is taken with `mariadb-dump --single-transaction --master-data=2`, which embeds the binlog coordinates, and is restored to the target. The coordinates are saved for the next step.

```
==> Binlog migration: seed target from source snapshot
Source MySQL: 8.0.46  Dump tool: mariadb-dump
Creating source snapshot with binlog coordinates (--master-data=2)...
Restoring snapshot to target...
Seed completed.
```

```
# binlog_coords.env
SRC_BINLOG_FILE=mysql-bin.000004
SRC_BINLOG_POS=66378596
```

Start replication

The migrator ensures the replication user exists on the source (created `WITH mysql_native_password`, which cross-vendor replication requires), assigns the target a distinct `server_id` if it collides with the source, and points MariaDB at the captured coordinates.

```
==> Binlog migration: configure and start replication
Ensuring replication user exists on source...
Applying replication coordinates: mysql-bin.000004:66378596
Replication started.
```

Verify

The tool polls replication status until both threads are running and lag is within `BINLOG_MAX_LAG_SECS` (default 30).

```
==> Binlog migration: verify replication
IO running: Yes
SQL running: Yes
Seconds behind master/source: 0
Replication verify passed.
```

There is no `ANALYZE TABLE` step in this mode, because replication keeps changing the tables after the seed.

Step 4: Confirm Changes Are Flowing

The target is now live-following the source, so a write on the source appears on the target within seconds. Inserting one row on the source and reading it back on the target confirms the pipeline end to end:

```
# on the source:
INSERT INTO sakila.film (title, language_id) VALUES ('LIVE REPLICATION TEST', 1);

# on the target, moments later:
+---------+-----------------------+
| film_id | title                 |
+---------+-----------------------+
|    1001 | LIVE REPLICATION TEST |
+---------+-----------------------+

# SHOW REPLICA STATUS on the target:
Master_Host: mysql.example.com
Seconds_Behind_Master: 0
```

Step 5: Cut Over

The tool stops after verification with replication running; the cutover is yours to perform when you are ready. The migrator does not stop replication or repoint your application for you. A safe cutover:

1. Stop application writes to the source. Put the application in maintenance mode or revoke write access so no new transactions are generated.
2. Wait for zero lag. Confirm `Seconds_Behind_Master: 0` on the target with `SHOW REPLICA STATUS\G`, so every committed source change has been applied.
3. Stop replication on the target with `STOP REPLICA;`, then `RESET REPLICA ALL;` once you are committed, to clear the replication configuration.
4. Repoint the application at the target MariaDB server and resume traffic.
5. Clean up the replication user on the source and any infrastructure accounts that were migrated.

Downtime is only the window between stopping writes on the source and resuming them on the target, which is short because the target is already caught up before you begin.

Reference

Beyond the common source and target variables, Replication requires `REPL_USER` and `REPL_PASS`. Optional variables include `SRC_BINLOG_FILE` and `SRC_BINLOG_POS` (auto-captured during the seed if unset), `BINLOG_MAX_LAG_SECS` (default 30), and `BINLOG_CREATE_REPL_USER` (default 1; set to 0 if the replication user already exists). See Environment Variables for the full list.

Other Modes

If Replication does not fit your situation, see the migrator overview to choose another mode: Serial Streaming Copy or Parallel Restartable Streaming Copy for an offline transfer when both servers are reachable, or Offline Copy when they are not.

This page is licensed: CC BY-SA / Gnu FDL

Migrating to MariaDB from PostgreSQL

There are many different ways to migrate from PostgreSQL to MariaDB. This article will discuss some of those options.

MariaDB's CONNECT Storage Engine

MariaDB's CONNECT storage engine can be used to migrate from PostgreSQL to MariaDB. There are two primary ways that this can be done.

See Loading the CONNECT Storage Engine for information on how to install the CONNECT storage engine.

Tables with ODBC table_type

The CONNECT storage engine allows you to create tables that refer to tables on an external server, and it can fetch the data using a compatible ODBC driver. PostgreSQL does have a freely available ODBC driver called psqlODBC. Therefore, if you install `psqlODBC` on the MariaDB Server, and then configure the system's ODBC framework (such as unixODBC), then the MariaDB server will be able to connect to the remote PostgreSQL server. At that point, you can create tables with the ENGINE=CONNECT and table_type=ODBC table options set, so that you can access the PostgreSQL tables from MariaDB.

See CONNECT ODBC Table Type: Accessing Tables From Another DBMS for more information on how to do that.

Once the remote table is setup, you can migrate the data to local tables very simply. For example:

```
CREATE TABLE psql_tab (
   id INT,
   str VARCHAR(50)
) ENGINE = CONNECT
table_type=ODBC
tabname='tab'
CONNECTION='DSN=psql_server';

CREATE TABLE tab (
   id INT,
   str VARCHAR(50)
) ENGINE = InnoDB;

INSERT INTO tab SELECT * FROM psql_tab;
```

Tables with JDBC table_type

The CONNECT storage engine allows you to create tables that refer to tables on an external server, and it can fetch the data using a compatible JDBC driver. PostgreSQL does have a freely available JDBC driver. If you install this JDBC driver on the MariaDB server, then the MariaDB server will be able to connect to the remote PostgreSQL server via JDBC. At that point, you can create tables with the ENGINE=CONNECT and table_type=JDBC table options set, so that you can access the PostgreSQL tables from MariaDB.

See CONNECT JDBC Table Type: Accessing Tables from Another DBMS for more information on how to do that.

Once the remote table is setup, you can migrate the data to local tables very simply. For example:

```
CREATE TABLE psql_tab (
   id INT,
   str VARCHAR(50)
) ENGINE = CONNECT
table_type=JDBC
tabname='tab'
CONNECTION='jdbc:postgresql://psql_server/db1';

CREATE TABLE tab (
   id INT,
   str VARCHAR(50)
) ENGINE = InnoDB;

INSERT INTO tab SELECT * FROM psql_tab;
```

PostgreSQL's Foreign Data Wrappers

PostgreSQL's foreign data wrappers can also be used to migrate from PostgreSQL to MariaDB.

mysql_fdw

mysql_fdw allows you to create a table in PostgreSQL that actual refers to a remote MySQL or MariaDB server. Since MySQL and MariaDB are compatible at the protocol level, this should also support MariaDB.

The foreign data wrapper also supports writes, so you should be able to write to the remote MariaDB table to migrate your PostgreSQL data. For example:

```
CREATE TABLE tab (
   id INT,
   str text
);

INSERT INTO tab VALUES (1, 'str1');

CREATE SERVER mariadb_server
   FOREIGN DATA WRAPPER mysql_fdw
   OPTIONS (host '10.1.1.101', port '3306');

CREATE USER MAPPING FOR postgres
   SERVER mariadb_server
   OPTIONS (username 'foo', password 'bar');

CREATE FOREIGN TABLE mariadb_tab (
   id INT,
   str text
)
SERVER mariadb_server
OPTIONS (dbname 'db1', table_name 'tab');

INSERT INTO mariadb_tab SELECT * FROM tab;
```

PostgreSQL's COPY TO

PostgreSQL's COPY TO allows you to copy the data from a PostgreSQL table to a text file. This data can then be loaded into MariaDB with LOAD DATA INFILE.

MySQL Workbench

MySQL Workbench has a migration feature that requires an ODBC driver. PostgreSQL does have a freely available ODBC driver called psqlODBC.

See Set up and configure PostgreSQL ODBC drivers for the MySQL Workbench Migration Wizard for more information.

Known Issues

Migrating Functions and Procedures

PostgreSQL's functions and procedures use a language called PL/pgSQL. This language is quite different than the default `SQL/PSM` language used for MariaDB's stored procedures. `PL/pgSQL` is more similar to `PL/PSQL` from Oracle, so you may find it beneficial to try migrate with SQL_MODE=ORACLE set.

MariaDB / PostgreSQL Comparison

https://www.youtube.com/watch?v=l_5AgRPTa54

MariaDB and PostgreSQL: A technical deep-dive into how they differ (54 minutes • 2024)

This deep-dive explores the architectural and philosophy differences between MariaDB and PostgreSQL. It is particularly useful for users who are troubleshooting performance issues or deciding which database engine best suits their specific workload requirements.

Core Topics Covered:

- Storage Engine Architecture: Explains MariaDB's unique "pluggable storage engine" architecture (like InnoDB, MyRocks, and Aria) compared to PostgreSQL's single integrated storage engine approach.
- SQL Compatibility & Standards: A look at how PostgreSQL strictly adheres to SQL standards, whereas MariaDB prioritizes high performance and "pragmatic" features (like temporal tables and virtual columns).

- Concurrency Models: A comparison of how each handles multi-user traffic. The video highlights MariaDB's thread-pool approach versus PostgreSQL's process-per-connection model.
- JSON & Modern Data Types: How both databases have evolved to handle NoSQL-style data, comparing MariaDB's dynamic columns and JSON functions against PostgreSQL's JSONB.

Key Takeaway for Troubleshooting:

The video emphasizes that because MariaDB is a multi-engine database, connection or performance issues are often solved by tuning the specific engine being used (usually InnoDB) rather than the server as a whole – a major difference from the more "monolithic" troubleshooting approach in PostgreSQL.

See Also

- Set up and configure PostgreSQL ODBC drivers for the MySQL Workbench Migration Wizard
- MariaDB and PostgreSQL: A technical deep-dive into how they differ (54 minutes • 2024)

This page is licensed: CC BY-SA / Gnu FDL

Migrating to MariaDB from Oracle

See Oracle XE 11.2. and MariaDB 10.1 integration on Ubuntu 14.04 and Debian systems.

Migrate from Oracle to MariaDB Server. This page provides detailed guidance, tools, and best practices for a smooth and efficient transition of your databases and applications.

This page is licensed: CC BY-SA / Gnu FDL

Oracle XE 11.2. and MariaDB 10.1 integration on Ubuntu 14.04 and Debian systems

Part 1: Installing Oracle Database Express Edition (XE)

This section covers the installation of Java and the Oracle XE database on a Debian-based system like Ubuntu.

Step 1: Download Oracle Database XE

1. Sign up for an Oracle account (if you don't have one) and log in.
2. Navigate to the download page for legacy versions of the database. A good starting point is the Oracle Database Express Edition (XE) 11gR2 Archive Downloads Page.
3. Accept the license agreement.
4. Download Oracle Database Express Edition 11g Release 2 for Linux x64. The file will be named similarly to `oracle-xe-11.2.0-1.0.x86_64.rpm.zip`.

Step 2: Install and Configure Java 8

Oracle XE 11gR2 requires a Java environment.

A. Add the Java PPA and Install

```
sudo add-apt-repository ppa:webupd8team/java
# Press Enter to accept when prompted
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

B. Verify Java Installation

After the installation completes, verify the version.

```
java -version
```

The output should be similar to:

```
java version "1.8.0_121"
```

C. Set JAVA_HOME Environment Variable

Edit the system-wide bash configuration file.

```
sudo nano /etc/bash.bashrc
```

Scroll to the bottom of the file and add the following lines:

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export PATH=$JAVA_HOME/bin:$PATH
```

Save the file and exit. Then, source the file to apply the changes and verify the variable is set.

```
source /etc/bash.bashrc
echo $JAVA_HOME
```

The output should be:

```
/usr/lib/jvm/java-8-oracle
```

Step 3: Install Prerequisite Packages

Additional packages are required to convert and install the Oracle RPM.

```
sudo apt-get install alien libaio1 unixodbc
```

Step 4: Convert the Oracle RPM to a DEB Package

The downloaded file is an RPM, which must be converted to a DEB package for Debian-based systems.

```
unzip oracle-xe-11.2.0-1.0.x86_64.rpm.zip
cd Disk1/
sudo alien --scripts -d oracle-xe-11.2.0-1.0.x86_64.rpm
```

> **Note:**
>
> This conversion step might take some time. You can proceed with the following configuration steps in a separate terminal window while it runs.

Step 5: Create `chkconfig` and Kernel Configuration

A. Create a `chkconfig` Compatibility Script

The Oracle installer expects the chkconfig utility, which doesn't exist on Debian-based systems. Create a compatibility script.

```
sudo nano /sbin/chkconfig
```

Add the following content to the new file:

```
#!/bin/bash
# Oracle 11gR2 XE installer chkconfig for Ubuntu
file=/etc/init.d/oracle-xe
if [[ ! `tail -n1 $file | grep INIT` ]]; then
  echo >> $file
  echo '### BEGIN INIT INFO' >> $file
  echo '# Provides: OracleXE' >> $file
  echo '# Required-Start: $remote_fs $syslog' >> $file
  echo '# Required-Stop: $remote_fs $syslog' >> $file
  echo '# Default-Start: 2 3 4 5' >> $file
  echo '# Default-Stop: 0 1 6' >> $file
  echo '# Short-Description: Oracle 11g Express Edition' >> $file
  echo '### END INIT INFO' >> $file
fi
update-rc.d oracle-xe defaults 80 01
#EOF
```

Save the file, then make it executable:

```
sudo chmod 755 /sbin/chkconfig
```

B. Configure Kernel Parameters

Create a configuration file for Oracle-specific kernel parameters.

```
sudo nano /etc/sysctl.d/60-oracle.conf
```

Copy and paste the following into the file. The value for `kernel.shmmax` should be approximately half of your system's physical RAM, specified in bytes (e.g., 512MB = 536870912 bytes).

```
# Oracle 11g XE kernel parameters
fs.file-max=6815744
net.ipv4.ip_local_port_range=9000 65000
kernel.sem=250 32000 100 128
kernel.shmmax=536870912
```

Apply the new kernel parameters and verify the `fs.file-max` setting.

```
sudo service procps start
sudo sysctl -q fs.file-max
```

The expected output is `6815744`.

Step 6: Final System Adjustments

A. Create Compatibility Links and Directories

```
sudo ln -s /usr/bin/awk /bin/awk
sudo mkdir /var/lock/subsys
sudo touch /var/lock/subsys/listener
```

B. Configure Shared Memory (/dev/shm)

To avoid a potential ORA-00845: MEMORY_TARGET error, re-mount the /dev/shm directory with a sufficient size. Replace 4096m with the amount of RAM on your machine.

```
sudo rm -rf /dev/shm
sudo mkdir /dev/shm
sudo mount -t tmpfs shmfs -o size=4096m /dev/shm
```

To make this change persistent across reboots, create a startup script.

```
sudo nano /etc/rc2.d/S01shm_load
```

Add the following content, again replacing `4096m` with your machine's RAM size.

```
#!/bin/sh
case "$1" in
start)
  mkdir /var/lock/subsys 2>/dev/null
  touch /var/lock/subsys/listener
  rm /dev/shm 2>/dev/null
  mkdir /dev/shm 2>/dev/null
  mount -t tmpfs shmfs -o size=4096m /dev/shm
  ;;
*)
  echo "Usage: $0 start"
  exit 1
  ;;
esac
```

Save the file and make it executable:

```
sudo chmod 755 /etc/rc2.d/S01shm_load
```

Step 7: Install and Configure Oracle XE

A. Install the Converted Package

By now, the .deb package should have been created. Install it using `\`.

```
sudo dpkg --install oracle-xe_11.2.0-2_amd64.deb
```

B. Run the Oracle Configuration Script

Configure the database by running the script and answering the prompts. The default answers are usually acceptable.

```
sudo /etc/init.d/oracle-xe configure
```

You will be prompted for:

- HTTP port for Oracle Application Express (default: `8080`)
- Database listener port (default: `1521`)
- A password for the `SYS` and `SYSTEM` accounts.
- Whether to start the database on boot (default: `y`)

C. Set Oracle Environment Variables

Edit `/etc/bash.bashrc` again to add the Oracle-specific variables.

```
sudo nano /etc/bash.bashrc
```

Add the following to the end of the file:

```
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/xe
export ORACLE_SID=XE
export NLS_LANG=`$ORACLE_HOME/bin/nls_lang.sh`
export ORACLE_BASE=/u01/app/oracle
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
export PATH=$ORACLE_HOME/bin:$PATH
```

Apply and verify the changes:

```
source /etc/bash.bashrc
echo $ORACLE_HOME
```

D. Start Oracle

```
sudo service oracle-xe start
```

Part 2: Installing SQL Developer

Step 1: Download and Install SQL Developer

1. Go to the Oracle site and download the Linux RPM package for Oracle SQL Developer.
2. Use `alien` to convert the package and `dpkg` to install it.

```
sudo alien --scripts -d sqldeveloper-4.1.5.21.78-1.noarch.rpm
sudo dpkg --install sqldeveloper_4.1.5.21.78-2_all.deb
```

3. Create a configuration directory.

```
mkdir ~/.sqldeveloper
```

Step 2: Run and Connect to Oracle XE

1. Launch SQL Developer.

   Bash

   ```
   sudo /opt/sqldeveloper/sqldeveloper.sh
   ```

2. If it asks for the Java path, provide it: `/usr/lib/jvm/java-8-oracle`.
3. Inside SQL Developer, create a new connection with the following details:
   - Click: `Connections`, then `Add new connection`
   - Connection Name: `XE`
   - Username: `SYSTEM`
   - Password: `<your-password>`
   - Connection Type: `Basic`, Role: `Default`
   - Hostname: `localhost`
   - Port: `1521`
   - SID: `xe`

Part 3: Installing MariaDB and Connecting to Oracle

Step 1: Install MariaDB

Follow the official instructions on the MariaDB website for your specific OS version.

> **Important:**
>
> Do NOT copy the commands below. Go to the MariaDB Repository Configuration Tool to generate the correct commands for your system. The following is only an example.

```
# Example commands for Ubuntu 14.04 Trusty
sudo apt-get install software-properties-common
sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xcbcb082a1bb943db
sudo add-apt-repository 'deb [arch=amd64,i386,ppc64el]
http://mirror.netinch.com/pub/mariadb/repo/10.1/ubuntu trusty main'
sudo apt-get update
sudo apt-get install mariadb-server
```

Step 2: Install MariaDB ODBC Connector

1. Download the “MariaDB Connector/ODBC” for Linux from the MariaDB website.
2. Extract the archive and copy the library file.

```
tar xfz mariadb-connector-odbc-2.0.13-ga-linux-x86_64.tar.gz
sudo cp -p mariadb-connector-odbc-2.0.13-ga-linux-x86_64/lib/libmaodbc.so /lib
sudo ldconfig
```

Step 3: Install UnixODBC and CONNECT Engine

```
sudo apt-get install unixodbc-dev unixodbc-bin unixodbc libodbc1 mariadb-connect-engine-10.1
```

Step 4: Configure ODBC Drivers

A. Configure the Oracle Driver

Edit `/etc/odbcinst.ini` and add the following section:

```
[Oracle ODBC driver for Oracle 11.2]
Description     = Oracle 11.2 ODBC driver
Driver          = /u01/app/oracle/product/11.2.0/xe/lib/libsqora.so.11.1
```

B. Configure the Oracle DSN

Edit `/etc/odbc.ini` and add a Data Source Name (DSN) for your Oracle XE database.

```
[XE]
Driver          = Oracle ODBC driver for Oracle 11.2
ServerName      = //localhost:1521/xe
DSN             = XE
UserName        = SYSTEM
Password        = <your-password>
```

Step 5: Test the Oracle ODBC Connection

Use `isql` to test the connection and create a sample table in Oracle.

```
isql -v XE SYSTEM <your-password>

CREATE TABLE t1 (i INT);
INSERT INTO t1 (i) VALUES (1);
INSERT INTO t1 (i) VALUES (3);
INSERT INTO t1 (i) VALUES (5);
INSERT INTO t1 (i) VALUES (8);
SELECT i FROM t1;
```

You should see the rows `1, 3, 5, 8` returned.

Step 6: Configure and Test the MariaDB CONNECT Engine

A. Set Environment Variables for MariaDB

For the CONNECT engine to find the Oracle libraries, you must add the Oracle environment variables to MariaDB's startup script. Edit /etc/init.d/mysql.

```
sudo nano /etc/init.d/mysql
```

Immediately after `### END INIT INFO`, add the following block:

```
# Oracle Environment for MariaDB CONNECT Engine
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export PATH=$JAVA_HOME/bin:$PATH
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/xe
export CLIENT_HOME=$ORACLE_HOME
export ORACLE_SID=XE
export NLS_LANG=`$ORACLE_HOME/bin/nls_lang.sh`
export ORACLE_BASE=/u01/app/oracle
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
export PATH=$ORACLE_HOME/bin:$PATH
```

B. Restart MariaDB and Test the Connection

```
sudo /etc/init.d/mysql restart
mysql -uroot -p
```

Once logged into the MariaDB client, run the following SQL commands:

```
CREATE DATABASE mdb;
USE mdb;
INSTALL SONAME 'ha_connect';
CREATE TABLE t1 ENGINE=CONNECT TABLE_TYPE=ODBC tabname='T1' CONNECTION='DSN=XE;UID=SYSTEM;PWD=<your-
password>';

SELECT I FROM t1;
```

You should see the values `1, 3, 5, 8` retrieved from the Oracle table.

Part 4: Connecting to MariaDB via JDBC

Step 1: Install the MariaDB JDBC Driver

1. Download the "MySQL Connector/J" from the MySQL or MariaDB website.
2. Extract the archive and copy the `.jar` file to a known location.

```
tar xvfz mysql-connector-java-5.0.8.tar.gz
cd mysql-connector-java-5.0.8/
sudo mkdir -p /usr/lib/jvm/java-8-oracle/lib/mariadb/
sudo cp -p mysql-connector-java-5.0.8-bin.jar /usr/lib/jvm/java-8-oracle/lib/mariadb/
```

Step 2: Configure SQL Developer

1. Launch SQL Developer: `sudo /opt/sqldeveloper/sqldeveloper.sh`
2. Go to Tools -> Preferences.
3. Expand Database and click on Third Party JDBC Drivers.
4. Click Add Entry… and navigate to the `.jar` file you copied: `/usr/lib/jvm/java-8-oracle/lib/mariadb/mysql-connector-java-5.0.8-bin.jar`.
5. Click OK to close the preferences.

Step 3: Connect to MariaDB

1. In SQL Developer, create a new connection.

2. Switch to the MySQL tab.
3. Enter the connection details:
   - Connection Name: `MariaDB via MySQL Conn`
   - Username: `root`
   - Password: `<your-root-password>`
   - Hostname: `localhost`
   - Port: `3306`
4. Click Test Connection. It should report `Status: Success`.
5. Save and connect. You can now run queries against your MariaDB server in the worksheet.

This page is licensed: CC BY-SA / Gnu FDL

Migrating to MariaDB from SQL Server

- MariaDB Backups Overview for SQL Server Users
- Understanding MariaDB Architecture
- MariaDB Features Not Available in SQL Server
- SQL Server Features Not Available in MariaDB
- SQL Server Features Implemented Differently in MariaDB
- SQL Server and MariaDB Types Comparison
- Syntax Differences between MariaDB and SQL Server
- MariaDB Authorization and Permissions for SQL Server Users
- MariaDB Replication Overview for SQL Server Users
- MariaDB Transactions and Isolation Levels for SQL Server Users
- Moving Data Between SQL Server and MariaDB
- Repairing MariaDB Tables for SQL Server Users
- Setting Up MariaDB for Testing for SQL Server Users

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Backups Overview for SQL Server Users

MariaDB has the following types of backups:

- Logical backups (dumps).
- Hot backups with mariadb-backup.
- Snapshots.
- Incremental backups.

Logical Backups (Dumps)

A *dump*, also called a *logical backup*, consists of the SQL statements needed to recreate MariaDB databases and their data into another server. A dump is the slowest form of backup to restore, because it implies executing all the SQL statements needed to recreate data. However it is also the most flexible, because restoring will work on any MariaDB version, because the SQL syntax is usually compatible. It is even possible to restore a dump into an older version, though the incompatible syntax (new features) will be ignored. Under certain conditions, MariaDB dumps may also be restored on other DBMSs, including SQL Server.

The compatibility between different versions and technologies is achieved by using executable comments, but we should be aware of how they work. If we use a feature introduced in version 11.1, for example, it will be included in

the dump inside an executable comment. If we restore that backup on a server with MariaDB 10.11, the 11.1 feature will be ignored. This is the only way to restore backups in older MariaDB versions.

`mariadb-dump`

Logical backups are usually taken with mariadb-dump (previously called mysqldump).

mariadb-dump allows one to dump all databases, a single database, or a set of tables from a database. It is even possible to specify a `WHERE` clause, which under certain circumstances allows to obtain incremental dumps.

For consistency reasons, when using the default storage engine InnoDB, it is important to use the `--single-transaction` option. This will read all data in a single transaction. It's important however to understand that long transactions may have a big impact on performance.

The `--master-data` option adds the statements to setup a slave to the dump.

MariaDB also supports statements which make easy to write applications to obtain custom types of dumps. For most `CREATE <object_type>` statement, a corresponding `SHOW CREATE <object_type>` exists. For example, SHOW CREATE TABLE returns the `CREATE TABLE` statement that can be used to recreate a certain table, without data.

`mydumper`

mydumper is a 3rd party tools to take dumps from MariaDB and MySQL databases. It is much faster than `mariadb-dump` because it takes backups with several parallel threads, usually one thread for each available CPU core. It produces several files, that can be used to restore a database using the related tool myloader.

Since is it a 3rd party tool, it could be incompatible with some present or future MariaDB features.

Hot Backups (`mariadb-backup`)

`mariadb-backup` is a tool for taking a backup of MariaDB files while MariaDB is working. A lock is only held for a small amount of time, so it is suitable to backup a server without causing disruptions. It works by taking corrupted backups and then bringing them to a consistent state by using the InnoDB undo log. `mariadb-backup` also properly backups MyRocks tables and non-transactional storage engines.

Cold Backups and Snapshots

A copy of all MariaDB files is a working backup. Therefore, the easiest way to backup a dataset is to shutdown the server and copy all its files. It will be entirely possible to start another server with a copy of those files. This is often referred to as a *cold backup*. However, in most cases we don't want to do this, because it implies downtime for the server: it will not be working at least for the time necessary to copy the files.

Snapshots are usually a better idea, as they are a consistent copy of the files at a given moment in time, taken without stopping the normal operations.

A snapshot of the files can be taken at several levels: filesystem level, if the filesystem supports snapshots, for example `zfs`; Linux Logical Volume Manager (LVM) also supports snapshots; and virtual machines also allow one to take snapshots. Windows shadow copies are also snapshots, with a benefit: it is possible to restore a single file from a shadow copy. A snapshot is not an expensive operation, because it does not imply a copy of the files. The current files will not be modified anymore, and changes to them will be written in separate places.

The problem with snapshots is that they behave like a logical copy of the files as they are in a given point in time. But database files are not guaranteed to be consistent in every moment, because contents can be buffered before being flushed to the disk. You can think a database snapshot like a database after an operating system crash.

With non-transactional tables, some data is typically lost. Data changes that are present in a buffer before the snapshot, but not written on a disk, cannot be recovered in any way. Data changes in transactional tables, like

InnoDB tables, can always be recovered after restoring a snapshot (or after a crash), as long as a commit was done. Tables will still need to be repaired, just like it happens after an SQL Server crash.

Snapshots can be taken while MariaDB is running. To restore them, stop MariaDB first - or kill the process, because you don't really care of the consequences in this case. Then restore a snapshot and start MariaDB again.

For more information about snapshots, check your filesystem, LVM or virtual machine documentation.

Incremental Backups

The term incremental backup in MariaDB indicates what SQL Server calls a *differential backup*. An important difference is that in SQL Server such backups are based on the transaction log, which wouldn't be possible in MariaDB because transaction logs are handled at storage engine level.

As mentioned here, MariaDB can use the binary log instead for backup purposes. Such incremental backups can be done manually. This means that:

- The binary log files are copied just like any other regular file.
- To copy those files it is necessary to have the proper permissions at filesystem level, not in MariaDB.
- Backups do not expire until we delete the last needed complete backup.

Replaying the Binary Log

The page Using mariadb-binlog shows how to use the mariadb-binlog utility to replay a binary log file.

The page also shows how to edit the binary log before replaying it. This allows one to undo an SQL statement that was executed by mistake, for example a `DROP TABLE` against a wrong table. The high level procedure is the following:

- Restore a backup that is older than the SQL statement to undo.
- Use `mariadb-binlog` to generate a file with the SQL statements that were executed after the backup.
- Edit the SQL file, erasing the unwanted statement.
- Run the SQL file.

Incremental Backups with mariadb-backup

The simplest way to take an incremental backup is to use mariadb-backup. This tool is able to take and restore incremental backups. For the complete procedure to use, see Incremental Backup and Restore with mariadb-backup.

mariadb-backup can run on both Linux and Windows systems.

Flashback

Flashback is a feature that allows one to bring all databases, some databases or some tables back to a certain point in time. This can only be done if the binary log is enabled. Flashback is not a proper backup, but it can be used to restore a certain set of data.

Copying Individual Tables

It is entirely possible to restore a single table from a physical backup, or to copy the table to another server.

With the MyISAM storage engine it was very easy to move tables between different servers, as long as the MySQL or MariaDB version was the same.

InnoDB is nowadays the default storage engine, and it is more complex, as it supports transactions for example. It still supports restoring a table from a physical file, this feature is called *transportable tablespaces*. There is a

particular procedure to follow, and some limitations. This is basically the MariaDB equivalent of detaching and re-attaching tables in SQL Server.

For more information, see InnoDB File-Per-Table Tablespaces.

By default. all table files are located in the *data directory*, which is defined by the system variable datadir. There may be exceptions, because a table's files can be located elsewhere using the DATA DIRECTORY and INDEX DIRECTORY options in `CREATE TABLE`.

Regardless of the storage engine used, each table's structure is generally stored in a file with the `.frm` extension.

The files used for partitioned tables are different from the files used for non-partitioned tables. See Partitions Files for details.

This page is licensed: CC BY-SA / Gnu FDL

Understanding MariaDB Architecture

MariaDB architecture is partly different from the architecture of traditional DBMSs, like SQL Server. Here we will examine the main components that a new MariaDB DBA needs to know. We will also discuss a bit of history, because this may help understand MariaDB philosophy and certain design choices.

This section is an overview of the most important components. More information is included in specific sections of this migration guide, or in other pages of the MariaDB documentation (see the links scattered over the text).

Storage Engines

MariaDB was born from the source code of MySQL, in 2008. Therefore, its history begins with MySQL.

MySQL was born at the beginning of the 90s. Back in the days, if compared to its existing competitors, MySQL was lightweight, simple to install, easy to learn. While it had a very limited set of features, it was also fast in certain common operations. And it was open source. These characteristics made it suitable to back the simple websites that existed at that time.

The web evolved rapidly, and the same happened to MySQL. Being open source helped a lot in this respect, because the community needed functionalities that weren't supported at that time.

MySQL was probably the first database system to support a pluggable storage engine architecture. Basically, this means that MySQL knows very little about creating or populating a table, reading from it, building proper indexes and caches. It just delegated all these operations to a special plugin type called a storage engine.

One of the first plugins developed by third parties was InnoDB. It is very fast, and it adds two important features that are not otherwise supported: transactions and foreign keys.

Note that when MariaDB asks a storage engine to write or read a row, the storage engine could theoretically do anything. This led to the creation of very interesting alternative engines, like BLACKHOLE (which doesn't write or read any data, acting like the /dev/null file in Linux), or CONNECT (which can read and write to files written in many different formats, or remote DBMSs, or some other special data sources).

Nowadays InnoDB is the default MariaDB storage engine, and it is the best choice for most use cases. But for particular needs, sometimes using a different storage engine is desirable. In case of doubts about the best storage engine to use for a specific case, check the Choosing the Right Storage Engine page.

When we create a table, we specify its storage engine or use the default one. It is possible to convert an existing table to another storage engine, though this is a blocking operation which requires a complete table copy. Third-party storage engines can also be installed while MariaDB is running.

Note that it is perfectly possible to use tables with different storage engines in the same transaction (even if some engines are not transactional). It is even possible to use different engines in the same query, for example with JOINs

and subqueries.

The default storage engine can be changed by changing the default_storage_engine variable. A different default can be specified for temporary tables by setting default_tmp_storage_engine. MariaDB uses Aria for system tables and temporary tables created internally to store the intermediate results of a query.

InnoDB

It is worth spending some more words here about InnoDB, the default storage engine.

Primary Key and Indexes

InnoDB primary keys are always the equivalent of SQL Server clustered indexes. In other words, an InnoDB table is always ordered by the primary key.

If an InnoDB table doesn't have a user-defined primary key, the first `UNIQUE` index whose columns are all `NOT NULL` is used as a primary key. If there is no such index, the table will have a *clustered index*. The terminology here can be a bit confusing for SQL Server and other DBMS users. A clustered index in InnoDB is a 6 bytes value that is added to the table. This index and its values are completely invisible to the users. It's important to note that clustered indexes are governed by a global mutex that greatly reduces their scalability.

Secondary indexes are ordered by the columns that are part of the index, and contain a reference to each entry's corresponding primary key value.

Some consequences of these design choices are the following:

- For performance reasons, a primary key value should be inserted in order. In other words, the last inserted value should be the highest. This order is normally followed when inserting values into an `AUTO_INCREMENT` primary key. The reason is that inserting values in the middle of an ordered data structure is slower, unless they fit into existing holes. If we insert primary key values randomly, InnoDB often has to rearrange pages to make some room for the new data.
- A big primary keys means that all secondary indexes are also big.
- A query by primary key will require a single search. A query on a secondary index that also reads columns not contained in the index will require one search on the index, plus one more search for each row that satisfies the index condition.
- We shouldn't explicitly include the primary key in a secondary index. If we do so, the primary key column will be duplicated in the index.

Tablespaces

For InnoDB, a *tablespace* is a file containing data (not a file group as in SQL Server). The types of tablespaces are:

- System tablespace.
- File-per-table tablespaces.
- Temporary tablespaces.

The system tablespace is stored in the file `ibdata`. It contains information used by InnoDB internally, like rollback segments, as well as some system tables. Historically, the system tablespace also contained all tables created by the user. In modern MariaDB versions, a table is created in the system tablespace only if the innodb_file_per_table system variable is set to 0 at the moment of the table creation. By default, innodb_file_per_table is 1.

Tables created while `innodb_file_per_table=1` are written into their own tablespace. These are `.ibd` files.

Starting from MariaDB 10.2, temporary tables are written into temporary tablespaces, which means `ibtmp*` files. Previously, they were created in the system tablespace or in file-per-table tablespaces according to the value of `innodb_file_per_table`, just like regular tables. Temporary tablespaces, if present, are deleted when MariaDB starts.

It is important to remember that tablespaces can never shrink. If a file-per-table tablespace grows too much, deleting data won't recover space. Instead, a new table must be created and data needs to be copied. Finally, the old table will be deleted. If the system tablespace grows too much, the only solution is to move data into a new MariaDB installation.

Transaction Logs

In SQL Server, the transaction log contains both the undo log and the redo log. Usually we have only one transaction log.

In MariaDB the undo log and the redo log are stored separately. By default, the redo log is written to two files, called `ib_logfile0` and `ib_logfile1`. The undo log by default is written to the *system tablespace*, which is in the `ibdata1` file. However, it is possible to write it in separate files in a specified directory.

MariaDB provides no way to inspect the contents of the transaction logs. However, it is possible to inspect the binary log.

InnoDB transaction logs are written in a circular fashion: their size is normally fixed, and when the end is reached, InnoDB continues to write from the beginning. However, if very long transactions are running, InnoDB cannot overwrite the oldest data, so it has to expand the log size instead.

InnoDB Buffer Pool

MariaDB doesn't have a central buffer pool. Each storage engine may or may not have a buffer pool. The InnoDB buffer pool is typically assigned a big amount of memory. See MariaDB Memory Allocation.

MariaDB has no extension like the SQL Server buffer pool extension.

A part of the buffer pool is called the change buffer. It contains dirty pages that have been modified in memory and not yet flushed.

InnoDB Background Threads

InnoDB has background threads that take care of flushing dirty pages from the change buffer to the tablespaces. They don't directly affect the latency of queries, but they are very important for performance.

SHOW ENGINE InnoDB STATUS shows information about them in the `BACKGROUND THREAD` section. They can also be seen using the threads table, in the performance_schema.

InnoDB flushing is similar to *lazy writes* and *checkpoints* in SQL Server. It has no equivalent for *eager writing*.

For more information, see InnoDB Page Flushing and InnoDB Purge.

Checksums and Doublewrite Buffer

InnoDB pages have checksums. After writing pages to disk, InnoDB verifies that the checksums match. The checksum algorithm is determined by innodb_checksum_algorithm. Check the variable documentation for its consequences on performance, backward compatibility and encryption.

In case of a system crash, hardware failure or power outage, a page could be half-written on disk. For some pages, this causes a disaster. Therefore, InnoDB writes essential pages to disk twice. A backup copy of the new page version is written first. Then, the old page is overwritten. The backup copies are written into a file called the *doublewrite buffer*.

- If an event prevents the first page from being written, the old version of the page will still be available.
- If an event prevents the old page from being completely overwritten by its new version, the page can still be recovered using the doublewrite buffer.

The doublewrite buffer can disabled using the innodb_doublewrite variable, but this usually doesn't bring big performance benefits. The doublewrite buffer location can be changed with innodb_doublewrite_file.

Aria

Even if we only create InnoDB tables, we use Aria indirectly, in two ways:

- For system tables.
- For internal temporary tables.

Aria is a non-transactional storage engine. By default it is crash-safe, meaning that all changes to data are durably written to a write-ahead log and can always be recovered in case of a crash.

Aria caches indexes into the pagecache. Data are not directly cached by Aria, so it's important that the underlying filesystem caches reads and writes.

The pagecache size is determined by the aria_pagecache_buffer_size system variable. To know if it is big enough we can check the proportion of free pages (the ratio between Aria_pagecache_blocks_used and Aria_pagecache_blocks_unused) and the proportion of cache misses (the ratio between Aria_pagecache_read_requests and Aria_pagecache_reads.

The proportion of dirty pages is the ratio between Aria_pagecache_blocks_used and Aria_pagecache_blocks_not_flushed tells us if the log file is big enough.

The size of Aria log is determined by aria_log_file_size.

Databases

MariaDB does not support the concept of schema. In MariaDB SQL, *schema* and *schemas* are synonyms for *database* and *databases*.

When a user connects to MariaDB, they don't connect to a specific database. Instead, they can access any table they have permissions for. There is however a concept of *default database*, see below.

A database is a container for database objects like tables and views. A database serves the following purposes:

- A database is a namespace.
- A database is a logical container to separate objects.
- A database has a default character set and collation, which are inherited by their tables.
- Permissions can be assigned on a whole database, to make permission maintenance simpler.
- Physical data files are stored in a directory which has the same name as the database to which they belong.

System Databases

MariaDB has the following system databases:

- mysql is for internal use only, and should not be read or written directly.
- information_schema contains all information that can be found in SQL Server's information_schema and more. However, while SQL Server's `information_schema` is a schema containing information about the local database, MariaDB's `information_schema` is a database that contains information about all databases.
- performance_schema contains information about MariaDB runtime. It is disabled by default. Enabling it requires setting the performance_schema system variable to 1 and restarting MariaDB.

Default Database

When a user connects to MariaDB, they can optionally specify a default database. A default database can also be specified or changed later, with the USE command.

Having a default database specified allows one to specify tables without specifying the name of the database where they are located. If no default database is specified, all table names must be fully qualified.

For example, the two following snippets are equivalent:

```
SELECT * FROM my_database.my_table;

-- is equivalent to:
USE my_database;
SELECT * FROM my_table;
```

Even if a default database is specified, tables from other databases can be accessed by specifying their fully qualified names:

```
-- this query joins my_database.my_table to your_database.your_table
USE my_database;
SELECT m.*
    FROM my_table m
    JOIN your_database.your_table y
        ON m.xyz = y.xyz;
```

MariaDB has the DATABASE() function to determine the current database:

```
SELECT DATABASE();
```

Stored procedures and triggers don't inherit a default database from the session, nor by a caller procedure. In that context, the default database is the database which contains the procedure. `USE` can be used to change it. The default database will only be valid for the rest of the procedure.

The Binary Log

Different tables can be built using different storage engines. It is important to note that not all engines are transactional, and that different engines implement the transaction logs in different ways. For this reason, MariaDB cannot replicate data from a primary to a replica using an equivalent of SQL Server transactional replication.

Instead, it needs a global mechanism to log the changes that are applied to data. This mechanism is the binary log, often abbreviated to binlog.

The binary log can be written in the following formats:

- STATEMENT logs SQL statements that modify data;
- ROW logs a reference to the rows that have been modified, if any (usually it's the primary key), and the new values that have been added or modified, in a binary format.
- MIXED is a combination of the above formats. It means that ROW is used for statements that can safely be logged in this way (see below), and STATEMENT is used in other cases. This is the default format from MariaDB 10.2.

In most cases, STATEMENT is slower because the SQL statement needs to be re-executed by the replica, and because certain statements may produce a different result in the replica (think about queries that use LIMIT without ORDER BY, or the CURRENT_TIMESTAMP() function). But there are exceptions, and besides, DDL statements are always logged as STATEMENT to avoid flooding the binary log. Therefore, the binary log may well contain both ROW and STATEMENT entries.

See Binary Log Formats.

The binary log allows:

- replication, if enabled on the primary;
- promoting a replica to a primary, if enabled on that replica;

- incremental backups;
- seeing data as they were in a point of time in the past (flashback);
- restoring a backup and re-appling the binary log, with the exception of a data change which caused problems (human mistake, application bug, SQL injection);
- Capture Data Changes (CDC), by streaming the binary log to technologies like Apache Kafka.

If you don't plan to use any of these features on a server, it is possible to disable the binary log to slightly improve the performance.

The binary log can be inspected using the mariadb-binlog utility, which comes with MariaDB. Enabling or disabling the binary log requires restarting MariaDB.

See also MariaDB Replication Overview for SQL Server Users and MariaDB Backups Overview for SQL Server Users for a better understanding of how the binary log is used.

Plugins

Storage engines are a special type of plugin. But others exist. For example, plugins can add authentication methods, new features, SQL syntax, functions, informative tables, and more.

A plugin may add some server variables and some status variables. Server variables can be used to configure the plugin, and status variables can be used to monitor its activities and status. These variables generally use the plugin's name as a prefix. For example InnoDB has a server variable called innodb_buffer_pool_size to configure the size of its buffer pool, and a status variable called Innodb_pages_read which indicates the number of memory pages read from the buffer pool. The category system variables of the MariaDB documentation has specific pages for system and status variables associated with various plugins.

Many plugins are installed by default, or available but not installed by default. They can be installed or uninstalled at runtime with SQL statements, like `INSTALL PLUGIN`, `UNINSTALL PLUGIN` and others; see Plugin SQL Statements. 3rd party plugins can be made available for installation by simply copying them to the plugin_dir.

It is important to note that different plugins may have different maturity levels. It is possible to prevent the installation of plugins we don't consider production-ready by setting the plugin_maturity system variable. For plugins that are distributed with MariaDB, the maturity level is determined by the MariaDB team based on the bugs reported and fixed.

Some plugins are developed by 3rd parties. Even some 3rd party plugins are included in MariaDB official distributions - the ones available on mariadb.org.

In MariaDB every authorization method (including the default one) is provided by an authentication plugin. A user can be required to use a certain authentication plugin. This gives us much flexibility and control. Windows users may be interested in gsapi (which supports Windows authentication, Kerberos and NTLM) and named_pipe (which uses named pipe impersonation).

Other plugins that can be very useful include userstat, which includes statistics about resources and table usage, and METADATA_LOCK_INFO, which provides information about metadata locks.

Thread Pool

MariaDB supports thread pool. It works differently on UNIX and on Windows. On Windows, it is enabled by default and its implementation is quite similar to SQL Server. It uses the Windows native CreateThreadpool API.

If we don't use the thread pool, MariaDB will use its traditional method to handle connections. It consists of using a dedicated thread for each client connection. Creating a new thread has a cost in terms of CPU time. To mitigate this cost, after a client disconnects, the thread may be preserved for a certain time in the thread cache.

Whichever connection method we use, MariaDB has a maximum number of simultaneous connections, which can be changed at runtime. When the limit is reached, if more clients try to connect they will receive an error. This prevents MariaDB from consuming all the server resources and freezing or crashing. See Handling Too Many Connections.

Configuration

MariaDB has many settings that control the server behavior. These can be set up when starting mysqld (mysqld options), and the vast majority are also accessible as server system variables. These can be classified in these ways:

- Dynamic or static;
- Global, session, or both.

Note that server system variables are not to be confused with user-defined variables. The latter are not used for MariaDB configuration.

Configuration Files

MariaDB can use several configuration files. Configuration files are searched in several locations, including in the user directory, and if present they all are read and used. They are read in a consistent order. These locations depend on the operating system; see Default Option File Locations. It is possible to tell MariaDB which files it should read; see Global Options Related to Option Files.

On Linux, by default the configuration files are called `my.cnf`. On Windows, by default the configuration files can be called `my.ini` or `my.cnf`. The former is more common.

If a variable is mentioned multiple times in different files, the occurrence that is read last will overwrite the others. Similarly, if a variable is mentioned several times in a single file, the occurrence that is read last overwrites the others.

The contents of each configuration file are organized by *option groups*. MariaDB Server and client programs read different groups. The read groups also depend on the MariaDB version. See Option Groups for the details. Most commonly, the `[server]` or `[mysqld]` groups are used to contain all server configuration. The `[client-server]` group can be used for options that are shared by the server and the clients (like the port to use), to avoid repeating those variables multiple times.

Dynamic and Static Variables

Dynamic variables have a value that can be changed at runtime, using the SET SQL statement. Static variables have a value that is decided at startup (see below) and cannot be changed without a restart.

The Server System Variables page states if variables are dynamic or static.

Scope

A global system variable is one that affects the general behavior of MariaDB. For example innodb_buffer_pool_size determines the size of the InnoDB buffer pool, which is used by read and write operations, no matter which user issued them. A session system variable is one that affects MariaDB behavior for the current connection; changing it will not affect other connected users, or future connections from the current user.

A variable could exist in both the global and session scopes. In this case, the session value is what affects the current connection. When a user connects, the current global value is copied to the session scope. Changing the global value afterward will not change existing connections.

The Server System Variables page states the scope of each variable.

Global variables and some session variables can only be modified by a user with the SUPER privilege (typically root).

Syntax

To see the value of a system variable:

```
-- global variables:
SELECT @@global.variable_name;
-- session variables:
SELECT @@session.variable_name;
-- or just use the shortcut:
SELECT @@variable_name;
```

A longer syntax, which is mostly useful to get multiple variables, makes use of the same pattern syntax that is used by the LIKE operator:

```
-- global variables whose name starts with 'innodb':
SHOW GLOBAL VARIABLES LIKE 'innodb%';
-- session variables whose name starts with 'innodb':
SHOW SESSION VARIABLES LIKE 'innodb%';
SHOW VARIABLES LIKE 'innodb%';
```

To modify the global or session value of a dynamic variable:

```
SET @@global.variable_name = 'new 'value';
SET @@session.variable_name = 'new 'value';
```

Notice that if we modify a global variable in this way, the new value will be lost at server restart. For this reason we probably want to change the value in the configuration file too.

For further information see:

- The SET statement.
- The SHOW VARIABLES statement.

Setting System Variables with Startup Parameters

System variables can be set at server startup without writing their values into a configuration file. This is useful if we want a value to be set once, until we change it or restart MariaDB. Values passed in this way override values written in the configuration files.

The general rule is that every global variable can be passed as an argument of `mysqld` by prefixing its name with `--` and by replacing every occurrence of `_` with `-` in its name.

For example, to pass `bind_address` as a startup argument:

```
mysqld --bind-address=127.0.0.1
```

Debugging Configuration

Mistyping a variable can prevent MariaDB from starting. We cannot set a variable that doesn't exist in the MariaDB version in use. In these cases, an error is written in the error log.

Having several configuration files and configuration groups, as well as being able to pass variables as command-line arguments, brings a lot of flexibility but can sometimes be confusing. When we are unsure about which values will be used, we can run:

```
mysqld --print-defaults
```

Status Variables

MariaDB status variables and some system tables allow external tools to monitor a server, building graphs on how they change over time, and allow the user to inspect what is happening inside the server.

Status variables cannot be directly modified by the user. Their values indicate how MariaDB is operating. Their scope can be:

- Global, meaning that the value is about some MariaDB activity.
- Session, meaning that the value measures activities taking place in the current session.

Many status variables exist in both scopes. For example,Cpu_time at global level indicates how much time the CPU was used by the MariaDB process (including all user sessions and all the background threads). At session level, it indicates how much time the CPU was used by the current session.

The status variables created by a plugin, usually, use the plugin name as a prefix.

The SHOW STATUS statement prints the values of the status variables that match a certain pattern.

```
-- Show all InnoDB global status variables
SHOW GLOBAL STATUS LIKE 'innodb%';
-- Show all InnoDB session status variables
SHOW SESSION STATUS LIKE 'innodb%';
SHOW STATUS LIKE 'innodb%';
-- Show global variables that contain the "size" substring:
SHOW GLOBAL STATUS LIKE '%size%';
```

Some status variables values are reset when FLUSH STATUS is executed. A possible use:

```
DELIMITER ||
BEGIN NOT ATOMIC
SET @i = 0;
WHILE @i < 60 DO
    SHOW GLOBAL STATUS LIKE 'Com_select';
    FLUSH STATUS;
    DO SLEEP(1);
    SET @i = @i + 1;
END WHILE;
END ||
```

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Features Not Available in SQL Server

Some MariaDB features are not available in SQL Server.

At first glance, it is not important to know about those features to migrate from SQL Server to MariaDB. However, this is not the case. Using MariaDB features that are not in SQL Server allows one to obtain more advantages from the migration, getting the most from MariaDB.

This page has a list of MariaDB features that are not supported in SQL Server. The list is not exhaustive.

Plugin Architecture

- Storage engines.
- Authentication plugins.
- Encryption plugins.
- ColumnStore is a columnar storage engine designed to scale horizontally. It runs on a specific edition of MariaDB, so currently it cannot be used in combination with other engines.

SQL

- The sql_mode variable determines in which cases an SQL statement should fail with an error, and in which cases it should succeed with a warning even if it is not entirely correct. For example, when a statement tries to insert a string in a column which is not big enough to contain it, it could fail, or it could insert a truncated string and emit a warning. It is a tradeoff between reliability and flexibility.
  - SQL_MODE=MSSQL allows one to use a small subset of SQL Server proprietary syntax.
- The CREATE … IF EXISTS, CREATE OR REPLACE, DROP … IF NOT EXISTS options are supported for most DDL statements.
- SHOW statements.
- SHOW CREATE statements.
- SHOW PROCESSLIST and PERFORMANCE_SCHEMA THREAD table provide much richer information, compared to SQL Server `sp_who()` and `sp_who2()` procedures.
- CHECKSUM TABLE statement.
- PL/SQL support (only for stored procedures and stored functions).
- Row constructors.
- `BEFORE` triggers.
- HANDLER statements, to scroll table rows ordered by an index or in their physical order.
- DO statement, to call functions without returning a result set.
- BENCHMARK() function, to measure the speed of an SQL expression.

See also Syntax Differences between MariaDB and SQL Server.

Types

- Character sets and collations don't depend on column type. They can be set globally, or at database, table or column level.
- Columns may use non-constant expressions as the DEFAULT value. TIMESTAMP columns may have a `DEFAULT` value.
- UNSIGNED numeric types.
- Dynamic columns (note that JSON is usually preferred to this feature).

See also SQL Server and MariaDB Types Comparison.

JSON

For compatibility with some other database systems, MariaDB supports the JSON pseudo-type. However, it is just an alias for:

```
LONGTEXT CHECK (JSON_VALID(column_name))
```

JSON_VALID() is the MariaDB equivalent of SQL Server's `ISJSON()`.

Features

- Flashback functionality allows one to "undo" the changes that happened after a certain point in time.
- Partitioned tables support the following features:
  - Tables can be partitioned based on multiple columns.
  - Several partitioning types are available.
  - Subpartitions.
- Progress reporting for some typically expensive statements.

This page is licensed: CC BY-SA / Gnu FDL

SQL Server Features Not Available in MariaDB

When planning a migration between different DBMSs, one of the most important aspects to consider is that the new database system will probably miss some features supported by the old one. This is not relevant for all users. The most widely used features are supported by most DBMSs. However, it is important to make a list of unsupported features and check which of them are currently used by applications. In most cases it is possible to implement such features on the application side, or simply stop using them.

This page has a list of SQL Server features that are not supported in MariaDB. The list is not exhaustive.

Introduced in SQL Server versions older than 2016

- Full outer joins.
- `GROUP BY CUBE` syntax.
- `MERGE` statement.
- In MariaDB, indexes are always ascending. Defining them as `ASC` or `DESC` has no effect.
  - For single-column indexes, the performance difference between an `ORDER BY ... ASC` and `DESC` is negligible.
  - For multiple-column indexes, an index may be unusable for certain queries because `DESC` is not supported. In some cases, a generated column can be used to invert the order of an index (for example, the expression `0 - price` can be indexed to index the prices in a descending order).
- The WITH syntax is currently only supported for the `SELECT` statement.
- Filtered indexes (`CREATE INDEX ... WHERE`).
- Autonomous transactions.
- User-defined types.
- Rules.
- Triggers don't support the following features:
  - Triggers on DDL and login.
  - `INSTEAD OF` triggers.
  - The `DISABLE TRIGGER` syntax.
- Cursors advanced features.
  - Global cursors.
  - `DELETE ... CURRENT OF`, `UPDATE ... CURRENT OF` statements: MariaDB cursors are read-only.
  - Specifying a direction (MariaDB cursors can only advance by one row).
- Synonyms.
- Table variables.
- Queues.
- XML indexes, XML schema collection, XQuery.
- User access to system functionalities, for example:
  - Running system commands (`xp_cmdshell()`).
  - Sending emails (`sp_send_dbmail()`).
  - Sending HTTP requests.
- External languages, external libraries (MariaDB only supports procedural SQL and PL/SQL).
- Negative permissions (the `DENY` command).
- Snapshot replication. See Provisioning a Slave.

Introduced in SQL Server 2016

- Native data masking
- PolyBase (however, MariaDB 10.5 supports accessing Amazon S3 via the S3 storage engine and several DBMSs via CONNECT)
- R and Python services
- ColumnStore indexes. MariaDB has a storage engine called ColumnStore, but this is a completely different feature.

Introduced in SQL Server 2017

- Adaptive joins
- Graph SQL

See Also

- SQL Server Features Implemented Differently in MariaDB

This page is licensed: CC BY-SA / Gnu FDL

SQL Server Features Implemented Differently in MariaDB

Modern DBMSs implement several advanced features. While an SQL standard exists, the complete feature list is different for every database system. Sometimes different features allow achieving the same purpose, but with a different logic and different limitations. This is something to take into account when planning a migration.

Some features are implemented by different DBMSs, with a similar logic and similar syntax. But there could be important differences that users should be aware of.

This page has a list of SQL Server features that MariaDB implements in a different way, and SQL Server features for which MariaDB has an alternative feature. Minor differences are not taken into account here. The list is not exhaustive.

SQL

- The list of supported data types is different.
- There are relevant differences in transaction isolation levels.
- `SNAPSHOT` isolation level is not supported. Instead, you can use `START TRANSACTION WITH CONSISTENT SNAPSHOT` to acquire a snapshot at the beginning of the transaction. This is compatible with all isolation levels. See How Isolation Levels are Implemented in MariaDB.
- JSON support is different.

Indexes and Performance

- Clustered indexes. In MariaDB, the physical order of rows is delegated to the storage engine. InnoDB uses the primary key as a clustered index.
- Hash indexes. Only some storage engines support `HASH` indexes.
  - The InnoDB storage engine has a feature called adaptive hash index, enabled by default. It means that in InnoDB all indexes are created as `BTREE`, and depending on how they are used, InnoDB could convert them from BTree to hash indexes, or the other way around. This happens in the background.
  - The MEMORY storage engine uses hash indexes by default, if we don't specify the `BTREE` keyword.
  - See Storage Engine Index Types for more information.
- Query store. MariaDB allows query performance analysis using the slow log and performance_schema. Some open source or commercial 3rd party tools read that information to produce statistics and make it easy to identify slow queries.

Tables

- Computed columns are called generated columns in MariaDB and are created with a different syntax. See also Implementation Differences Compared to Microsoft SQL Server.
- Temporal tables use a different (more standard) syntax on MariaDB. In MariaDB, the history is stored in the same table as current data (but optionally in different partitions). MariaDB supports both SYSTEM_TIME and APPLICATION_TIME.
- Hidden columns are Invisible columns in MariaDB.
- Temporary tables are implemented and used differently.

High Availability

- `NOT FOR REPLICATION`
  - MariaDB supports replication filters to exclude some tables or databases from replication
  - It is possible to keep a table empty in a slave (or in the master) by using the BLACKHOLE storage engine.
  - The master can have columns that are not present in a slave (the other way around is also supported). Before using this feature, carefully read the Replication When the Master and Slave Have Different Table Definitions page.
  - With MariaDB it's possible to prevent a trigger from running on slaves.
  - It's possible to run events without replicating them. The same applies to some administrative statements.
  - MariaDB superusers can run statements without replicating them, by using the sql_log_bin system variable.
  - Constraints and triggers cannot be disabled for replication, but it is possible to drop them on the slaves.
  - The `IF EXISTS` syntax allows one to easily create a table on the master that already exists (possibly in a different version) on a slave.
- pollinginterval option. See Delayed Replication.

Security

- The list of permissions is different.
- Security policies. MariaDB allows one to achieve the same results by assigning permissions on views and stored procedures. However, this is not a common practice and it's more complicated than defining security policies. See Other Uses of Views.
- MariaDB does not support an `OUTPUT` clause. Instead, we can use DELETE RETURNING and, since MariaDB 10.5, INSERT RETURNING and REPLACE RETURNING.

Other Features

- Linked servers. MariaDB supports storage engines to read from, and write to, remote tables. When using the CONNECT engine, those tables could be in different DBMSs, including SQL Server.
- Job scheduler: MariaDB uses an event scheduler to schedule events instead.

See Also

- SQL Server features not available in MariaDB

This page is licensed: CC BY-SA / Gnu FDL

SQL Server and MariaDB Types Comparison

This page helps to map each SQL Server type to the matching MariaDB type.

Numbers

In MariaDB, numeric types can be declared as `SIGNED` or `UNSIGNED`. By default, numeric columns are `SIGNED`, so not specifying either will not break compatibility with SQL Server.

When using `UNSIGNED` values, there is a potential problem with subtractions. When subtracting an `UNSIGNED` valued from another, the result is usually of an `UNSIGNED` type. But if the result is negative, this will cause an error. To solve this problem, we can enable the NO_UNSIGNED_SUBTRACTION flag in sql_mode.

For more information see Numeric Data Type Overview.

Integer Numbers

SQL Server Types	Size (bytes)	MariaDB Types	Size (bytes)	Notes
tinyint	1	TINYINT	1	
smallint	2	SMALLINT	2	
		MEDIUMINT	3	Takes 3 bytes on disk, but 4 bytes in memory
int	4	INT / INTEGER	4	
bigint	8	BIGINT	8	

Real Numbers (approximated)

SQL Server Types	Precision	Size	MariaDB Types	Size
float(1-24)	7 digits	4	FLOAT(0-23)	4
float(25-53)	15 digits	8	FLOAT(24-53)	8

MariaDB supports an alternative syntax: `FLOAT(M, D)`. M is the total number of digits, and D is the number of digits after the decimal point.

See also: Floating-point Accuracy.

Aliases

In SQL Server `real` is an alias for `float(24)`.

In MariaDB DOUBLE, and DOUBLE PRECISION are aliases for `FLOAT(24-53)`.

Normally, `REAL` is also a synonym for `FLOAT(24-53)`. However, the sql_mode variable can be set with the `REAL_AS_FLOAT` flag to make `REAL` a synonym for `FLOAT(0-23)`.

Real Numbers (Exact)

SQL Server Types	Precision	Size (bytes)	MariaDB Types	Precision	Size (bytes)
decimal	0 - 38	Up to 17	DECIMAL	0 - 38	See table

MariaDB supports this syntax: `DECIMAL(M, D)`. M and D are both optional. M is the total number of digits (10 by default), and D is the number of digits after the decimal point (0 by default). In SQL Server, defaults are 18 and 0, respectively. The reason for this difference is that SQL standard imposes a default of 0 for D, but it leaves the implementation free to choose any default for M.

SQL Server `DECIMAL` is equivalent to MariaDB `DECIMAL(18)`.

Aliases

The following aliases for `DECIMAL` are recognized in both SQL Server and MariaDB: `DEC`, `NUMERIC`. MariaDB also allows one to use `FIXED`.

Money

SQL Server `money` and `smallmoney` types represent real numbers guaranteeing a very low level of approximation (five decimal digits are accurate), optionally associated with one of the supported currencies.

MariaDB doesn't have monetary types. To represent amounts of money:

- Store the currency in a separate column, if necessary. It's possible to use a foreign key to a currencies table, or the ENUM type.
- Use a non-approximated type:
  - DECIMAL is very convenient, as it allows one to store the number as-is. But calculations are potentially slower.
  - An integer type is faster for calculations. It is possible to store, for example, the amount of money multiplied by 100.

There is a small incompatibility that users should be aware about. `money` and `smallmoney` are accurate to about 4 decimal digits. This means that, if you use enough decimal digits, operations on these types may produce different results than the results they would produce on MariaDB types.

Bits

The BIT type is supported in MariaDB. Its maximum size is `BIT(64)`. The `BIT` type has a fixed length. If we insert a value which requires less bits than the ones that are allocated, zero-bits are padded on the left.

In MariaDB, binary values can be written in one of the following ways:

- `b'value'`
- `0value` where `value` is a sequence of 0 and 1 digits. Hexadecimal syntax can also be used. For more details, see Binary Literals and Hexadecimal Literals.

MariaDB and SQL Server have different sets of bitwise operators. See Bit Functions and Operators.

BOOLEAN Pseudo-Type

In SQL Server, it is common to use `bit` to represent boolean values. In MariaDB it is possible to do the same, but this is not a common practice.

A column can also be defined as BOOLEAN or `BOOL`, which is just a synonym for TINYINT. `TRUE` and `FALSE` keywords also exist, but they are synonyms for 1 and 0. To understand what this implies, see Boolean Literals.

In MariaDB `'True'` and `'False'` are always strings.

Date and Time

SQL Server Types	Range	Precision	Size (bytes)	MariaDB Types	Range	Size (bytes)	Precision	Notes
date	0001-01-01 - 9999-12-31	3	/	DATE	0001-01-01 - 9999-12-31	3	/	They cover the same range
datetime	1753-01-01 - 9999-12-31	8	0 to 3, rounded	DATETIME	001-01-01 - 9999-12-31	8	0 to 6	MariaDB values are not approximated, see below.
datetime2	001-01-01 - 9999-12-31	8	6 to 8	DATETIME	001-01-01 - 9999-12-31	8	0 to 6	MariaDB values are not approximated, see below.
smalldatetime				DATETIME				
datetimeoffset				DATETIME				
time				TIME				

You may also consider the following MariaDB types:

- TIMESTAMP has little to do with SQL Server's `timestamp`. In MariaDB it is the number of seconds elapsed since the beginning of 1970-01-01, with a decimal precision up to 6 digits (0 by default). The maximum allowed value is '2038-01-19 03:14:07'. Values are always stored in UTC. A TIMESTAMP column can optionally be automatically set to the current timestamp on insert, on update, or both. It is not meant to be a unique row identifier. Also, in MariaDB the range of TIMESTAMP values is
- YEAR is a 1-byte type representing years between 1901 and 2155, as well as 0000.

Zero Values

MariaDB allows a special value where all the parts of a date are zeroes: `'0000-00-00'`. This can be disallowed by setting sql_mode=NO_ZERO_DATE.

It is also possible to use values where only some date parts are zeroes, for example `'1994-01-00'` or `'1994-00-00'`. These values can be disallowed by setting sql_mode=NO_ZERO_IN_DATE. They are not affected by `NO_ZERO_DATE`.

Syntax

Several different date formats are understood. Typically used formats are `'YYYY-MM-DD'` and `YYYYMMDD`. Several separators are accepted.

The syntax defined in standard SQL and ODBC are understood - for example, `DATE '1994-01-01'` and `{d '1994-01-01'}`. Using these eliminates possible ambiguities in contexts where a temporal value could be interpreted as a string or as an integer.

See Date and Time Literals for the details.

Precision

For temporal types that include a day time, MariaDB allows a precision from 0 to 6 (microseconds), 0 being the default. The subsecond part is never approximated. It adds up to 3 bytes. See Data Type Storage Requirements for the details.

String and Binary

Binary Strings

SQL Server Types	Size (bytes)	MariaDB Types	Notes
binary	1 to 8000	VARBINARY or BLOB	See below for BLOB types
varbinary	1 to 8000	VARBINARY or BLOB	See below for BLOB types
image	2^31-1	VARBINARY or BLOB	See below for BLOB types

The `VARBINARY` type is similar to `VARCHAR`, but stores binary byte strings, just like SQL Server `binary` does.

For large binary strings, MariaDB has four `BLOB` types, with different sizes. See BLOB and TEXT Data Types for more information.

Character Strings

One important difference between SQL Server and MariaDB is that **in MariaDB character sets do not depend on types and collations**. Character sets can be set at database, table or column level. If this is not done, the default character sets applies, which is specified by the character_set_server system variable.

To create a MariaDB table that is identical to a SQL Server table, **it may be necessary to specify a character set for each string column**. However, in many cases using UTF-8 will work.

SQL Server Types	Size (bytes)	MariaDB Types	Size (bytes)	Character set
char	1 to 8000	CHAR	0 to 255	utf8mb4 (1, 4)
varchar	1 to 8000	VARCHAR	0 to 65,532 (2)	utf8mb4 (1)
text	2^31-1	TEXT	2^31-1	ucs2
nchar	2 to 8000	CHAR	0 to 255	utf16 or ucs2 (3, 4)
nvarchar	2 to 8000	VARCHAR	0 to 65,532 (2) (5)	utf16 or ucs2 (1) (3)
ntext	2^30 - 1	TEXT	2^31-1	ucs2

Notes:

1. If SQL Server uses a non-unicode collation, a subset of UTF-8 is used. So it is possible to use a smaller character set on MariaDB too.
2. InnoDB has a maximum row length of 65,535 bytes. TEXT columns do not contribute to the row size, because they are stored separately (except for the first 12 bytes).
3. In SQL Server, UTF-16 is used if data contains Supplementary Characters, otherwise UCS-2 is used. If not sure, use `utf16` in MariaDB.
4. In SQL Server, the value of `ANSI_PADDING` determines if `char` values should be padded with spaces to their maximum length. In MariaDB, this depends on the PAD_CHAR_TO_FULL_LENGTH sql_mode flag.
5. See JSON, below.

SQL Server Special Types

rowversion

MariaDB does not have the `rowversion` type.

If the only purpose is to check if a row has been modified since its last read, a TIMESTAMP column can be used instead. Its default value should be `ON UPDATE CURRENT_TIMESTAMP`. In this way, the timestamp will be updated whenever the column is modified.

A way to preserve much more information is to use a temporal table. Past versions of the row will be preserved.

sql_variant

MariaDB does not support the `sql_variant` type.

MariaDB is quite flexible about implicit and explicit type conversions. Therefore, for most cases storing the values as a string should be equivalent to using `sql_variant`.

Be aware that the maximum length of an `sql_variant` value is 8,000 bytes. In MariaDB, you may need to use `TINYBLOB`.

uniqueidentifier

While MariaDB does not support the `uniqueidentifier` type, the UUID type can typically be used for the same purpose.

`uniqueidentifier` columns contain 16-bit GUIDs. MariaDB UUID columns store UUIDv1 values (128 bits).

The UUID type was implemented in MariaDB 10.7. On older versions, you can generate unique values with the UUID() or UUID_SHORT() functions, and store them in `BIT(128)` or `BIT(64)` columns, respectively.

xml

MariaDB does not support the `xml` type.

XML data can be stored in string columns. MariaDB supports several XML functions.

JSON

With SQL Server, typically JSON documents are stored in `nvarchar` columns in a text form.

MariaDB has a JSON pseudo-type that maps to LONGTEXT. However, from MariaDB 10.5 the `JSON` pseudo-type also checks that the value is valid a JSON document.

MariaDB supports different JSON functions than SQL Server. MariaDB currently has more functions, and SQL Server syntax will not work. See JSON functions for more information.

MariaDB Specific Types

The following types are supported by MariaDB and don't have a direct equivalent in SQL Server. If you are migrating your database to MariaDB, you can consider using these types.

- INET6 - IPv6 addresses.
- INET4 - IPv4 addresses.

This page is licensed: CC BY-SA / Gnu FDL

Syntax Differences between MariaDB and SQL Server

This article contains a non-exhaustive list of syntax differences between MariaDB and SQL Server and is written for SQL Server users who are unfamiliar with MariaDB.

Compatibility Features

Some features are meant to improve syntax and semantics compatibility between MariaDB versions, between MariaDB and MySQL, and between MariaDB and other DBMSs. This section focuses on compatibility between MariaDB and SQL Server.

sql_mode and old_mode

SQL semantics and syntax, in MariaDB, are affected by the sql_mode variable. Its value is a comma-separated list of flags, and each of them, if specified, affects a different aspect of SQL syntax and semantics.

A particularly important flag for users familiar with SQL Server is MSSQL.

sql_mode can be changed locally, in which case it only affects the current session; or globally, in which case it will affect all new connections (but not the connections already established). sql_mode must be assigned a comma-separated list of flags.

A usage example:

```
# check the current global and local sql_mode values
SELECT @@global.sql_mode;
SELECT @@session.sql_mode;
# empty sql_mode for all users
SET GLOBAL sql_mode = '';
# add MSSQL flag to the sql_mode for the current session
SET SESSION sql_mode = CONCAT(sql_mode, ',MSSQL');
```

old_mode is very similar to sql_mode, but its purpose is to provide compatibility with older MariaDB versions. Its flags shouldn't affect compatibility with SQL Server (though it is theoretically possible that some of them do, as a side effect).

Executable Comments

MariaDB supports executable comments. These are designed to write generic queries that are only executed by MariaDB, and optionally only certain versions.

The following examples show how to insert SQL code that will be ignored by SQL Server but executed by MariaDB, or some of its versions.

- Executed by MariaDB and MySQL (see below):

```
SELECT * FROM tab /*! FORCE INDEX (idx_a) */ WHERE a = 1 OR b = 2;
```

- Executed by MariaDB only:

```
SELECT * /*M! , @in_transaction */ FROM tab;
```

- Executed by MariaDB starting from version 10.0.5:

```
DELETE FROM user WHERE id = 100 /*!M100005 RETURNING email */;
```

As explained in the Understanding MariaDB Architecture page, MariaDB was initially forked from MySQL. At that time, executable comments were already supported by MySQL. This is why the `/*! ... */` syntax is supported by both MariaDB and MySQL. But because MariaDB also supports specific syntax not supported by MySQL, it added the `/*M! ... */` syntax.

Generic Syntax

Here we discuss some differences between MariaDB and SQL Server syntax that may affect any user, as well as some hints to make queries compatible with a reasonable amount of work.

Delimiters

SQL Server uses two different terminators:

- The batch terminator is the `go` command. It tells Microsoft clients to send the text we typed to SQL Server.
- The query terminator is a semicolon (`;`) and it tells SQL Server where a query ends.

It is rarely necessary to use `;` in SQL Server. It is required for certain common table expressions, for example.

But the same doesn't apply to MariaDB. **Normally, with MariaDB, you only use `;`.**

However, MariaDB also has some situations where you want to use a `;` but you don't want the mariadb command-line client to send the query yet. This can be done in any situation, but it is particularly useful when creating stored routines or using BEGIN NOT ATOMIC.

The reason is better explained with an example:

```
CREATE PROCEDURE p()
BEGIN
    SELECT * FROM t1;
    SELECT * FROM t2;
END;
```

If we enter this procedure in this way in the `mariadb` client, as soon as we type the first `;` (after the first `SELECT`) and press enter, the statement will be sent. MariaDB will try to parse it and will return an error.

To avoid this, `mariadb` implements the DELIMITER statement. This client statement is never sent to MariaDB. Instead, the client uses it to find out when the typed query should be sent. Let's correct the above example:

```
DELIMITER ||

CREATE PROCEDURE p()
BEGIN
    SELECT * FROM t1;
    SELECT * FROM t2;
END;

DELIMITER ;
```

Names

In MariaDB, most names have a maximum length of 64 characters. When migrating an SQL Server database to MariaDB, check if some names exceed this limit (SQL Server maximum length is 128).

By default, MariaDB names are case-sensitive if the operating system has case-sensitive file names (Linux), and case-insensitive if the operating system is case-insensitive (Windows). SQL Server is case-insensitive by default on all operating systems.

When migrating a SQL Server database to MariaDB on Linux, to avoid problems, you may want to set the lower_case_table_names system variable to 1, making table names, database names, and aliases case-insensitive.

Names can be quoted inside backtick characters (```). This character can be used in names, in which case it should be doubled. By default, this is the only way to quote names.

To also enable the use of double quotes (`"`), modify sql_mode by adding the `ANSI_QUOTES` flag. This is the equivalent of setting QUOTED_IDENTIFIER ON in SQL Server.

To also enable the use of SQL Server style quotes (`[` and `]`), modify sql_mode by adding the `MSSQL` flag.

The case-sensitivity of stored procedures and functions is never a problem, as they are case-insensitive in SQL Server.

Quoting Strings

In SQL Server, by default, strings can only be quoted with single-quotes (`'`), and to use a double quote in a string, it should be doubled (`''`). This also works by default in MariaDB.

SQL Server also allows to use of double quotes (`"`) to quote strings. This works by default in MariaDB, but as mentioned before, it won't work if sql_mode contains the `ANSI_QUOTES` flag.

NULL

The default semantics of NULL in SQL Server and MariaDB are the same, by default.

However, SQL Server allows one to change it globally with SET ANSI_NULLS OFF, or at the database level with ALTER DATABASE.

There is no way to achieve exactly the same result in MariaDB. To perform NULL-safe comparisons in MariaDB, one should replace the = operator with the <=> operator.

Also, note that MariaDB doesn't support the `UNKNOWN` pseudo-value. An expression like `NULL OR 0` returns `NULL` in MariaDB.

LIKE

In MariaDB, LIKE expressions only have two characters with special meanings: `%` and `_`. These two characters have the same meanings they have in SQL Server.

The additional characters recognized by SQL Server (`[`, `]` and `^`) are part of regular expressions. MariaDB supports the REGEXP operator that supports the full regular expression syntax.

Data Definition Language

Here we discuss some DDL differences that database administrators will want to be aware of.

While this section is meant to highlight the most noticeable DDL differences between MariaDB and SQL Server, there are many others, both in the syntax and in the semantics. See the ALTER statement documentation.

Altering Tables Online

Altering tables online can be a problem, especially when the tables are big and we don't want to cause a disruption.

MariaDB offers the following solutions to help:

- The ALTER TABLE … ALGORITHM clause allows one to specify which algorithm should be used to run a certain operation. For example, `INPLACE` tells MariaDB not to create a table copy (perhaps because we don't have enough disk space), and `INSTANT` tells MariaDB to execute the operation instantaneously. Not all algorithms are supported for certain operations. If the algorithm we've chosen cannot be used, the ALTER TABLE statement will fail with an error.
- The ALTER TABLE … LOCK clause allows one to specify which lock type should be used. For example, `NONE` tells MariaDB to avoid any lock on the table and `SHARED` only allows one to acquire a share lock. If the operation requires a lock that is more strict than the one, we are requesting, the ALTER TABLE statement will fail with an error. Sometimes this happens because the `LOCK` level we want is not available for the specified `ALGORITHM`.

To find out which operations require a table copy and which lock levels are necessary, see InnoDB Online DDL Overview.

An `ALTER TABLE` can be queued because a long-running statement (even a `SELECT`) requires a metadata lock. Since this may cause trouble, sometimes we want the operation to simply fail if the wait is too long. This can be achieved with the WAIT and NOWAIT clauses, whose syntax is a bit different from SQL Server.

SQL Server `WITH ONLINE = ON` is equivalent to MariaDB `LOCK = NONE`. However, note that most ALTER TABLE statements support `ALGORITHM = INSTANT`, which is non-blocking and much faster (almost instantaneous, as the syntax suggests).

IF EXISTS, IF NOT EXISTS, OR REPLACE

Most DDL statements, including ALTER TABLE, support the following syntax:

- `DROP IF EXISTS`: A warning (not an error) is produced if the object does not exist.
- `OR REPLACE`: If the object exists, it is dropped and recreated; otherwise, it is created. This operation is atomic, so at no point in time does the object not exist.
- `CREATE IF NOT EXISTS`: If the object already exists, a warning (not an error) is produced. The object will not be replaced.

These statements are functionally similar (but less verbose) than SQL Server snippets similar to the following:

```
IF NOT EXISTS (
        SELECT name
            FROM sysobjects
            WHERE name = 'my_table' AND xtype = 'U'
    )
    CREATE TABLE my_table (
        ...
    )
go
```

Altering Columns

With SQL Server, the only syntax to alter a table column is `ALTER TABLE ... ALTER COLUMN`. MariaDB provides more ALTER TABLE commands to obtain the same result:

- CHANGE COLUMN allows one to perform any change by specifying a new column definition, including the name.
- MODIFY COLUMN allows any change, except renaming the column. This is a slightly simpler syntax that we can use when we don't want to change a column name.
- ALTER COLUMN allows one to change or drop the `DEFAULT` value.
- RENAME COLUMN allows one to only change the column name.

Using a more specific syntax is less error-prone. For example, by using `ALTER TABLE ... ALTER COLUMN`,we will not accidentally change the data type.

The word `COLUMN` is usually optional, except in the case of `RENAME COLUMN`.

SHOW Statements

MariaDB supports SHOW statements to quickly list all objects of a certain type (tables, views, triggers…). Most `SHOW` statements support a `LIKE` clause to filter data. For example, to list the tables in the current database whose name begins with 'wp_':

```
SHOW TABLES LIKE 'wp\_%';
```

This is the equivalent of this query, which would work on both MariaDB and SQL Server:

```
SELECT TABLE_SCHEMA, TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_NAME LIKE 'wp\_';
```

SHOW CREATE Statements

In general, for each `CREATE` statement, MariaDB also supports a `SHOW CREATE` statement. For example, there is a SHOW CREATE TABLE that returns the CREATE TABLE statement that can be used to recreate a table.

Though SQL Server has no way to show the DDL statement to recreate an object, `SHOW CREATE` statements are functionally similar to `sp_helptext()`.

Database Comments

MariaDB does not support extended properties. Instead, it supports a `COMMENT` clause for most CREATE and ALTER statements.

For example, to create and then change a table comment:

```
CREATE TABLE counter (
    c INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
)
    COMMENT 'Monotonic counter'
;
ALTER TABLE counter COMMENT
    'Counter. It can contain many values, we only care about the max';
```

Comments can be seen with `SHOW CREATE` statements or by querying `information_schema` tables. For example:

```
SELECT TABLE_COMMENT
    FROM information_schema.TABLES
    WHERE TABLE_SCHEMA = 'test' AND TABLE_NAME = 'counter';
+-------------------------------------------------------------------+
| TABLE_COMMENT                                                     |
+-------------------------------------------------------------------+
| Counter. It can contain many values, we only care about the max |
+-------------------------------------------------------------------+
```

Error Handling

MariaDB SHOW ERRORS and SHOW WARNINGS statements can be used to show errors, or warnings and errors. This is convenient for clients, but stored procedures cannot work with the output of these commands.

Instead, inside stored procedures, you can:

- Use the GET DIAGNOSTICS command to assign error properties to variables. This is the equivalent of using SQL Server functions like `ERROR_NUMBER()` or `ERROR_STATE()`.
- Add a DECLARE HANDLER block to handle all errors, a class of errors, or a specific error. This is the equivalent of SQL Server `TRY ... CATCH`.
- An error or warning can be generated on purpose using SIGNAL. Inside a `DECLARE HANDLER` block, RESIGNAL can be used to issue the error again and interrupt the execution of the block. These are the equivalents of SQL Server `RAISERROR()`.

Administration

Administration and maintenance commands in MariaDB use a different syntax from SQL Server.

- OPTIMIZE TABLE rebuilds table data and indexes. It can be considered as the MariaDB equivalent of SQL Server's `ALTER INDEX REBUILD`. See Defragmenting InnoDB Tablespaces for more information. This statement is always locked. It supports WAIT and NOWAIT syntax,
- MariaDB has an ANALYZE TABLE command, which is equivalent to `UPDATE STATISTICS`.

BULK INSERT

MariaDB has no `BULK INSERT` statement. Instead, it supports:

- LOAD DATA INFILE to load data from files in CSV or similar formats;
- LOAD XML INFILE to load data from XML files.

See also How to Quickly Insert Data Into MariaDB.

See Also

- SQL Server and MariaDB Types Comparison
- MariaDB Transactions and Isolation Levels for SQL Server Users
- SQL_MODE=MSSQL

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Authorization and Permissions for SQL Server Users

Understanding Accounts and Users

MariaDB authorizes access and check permissions on accounts, rather than users. Even if MariaDB supports standard SQL commands like CREATE USER and DROP USER, it is important to remember that it actually works with accounts.

An account is specified in the format `'user'@'host'`. The quotes are optional and allow one to include special characters, like dots. The host part can actually be a pattern, which follows the same syntax used in `LIKE` comparisons. Patterns are often convenient because they can match several hostnames.

Here are some examples.

Omitting the host part indicates an account that can access from any host. So the following statements are equivalent:

```
CREATE USER viviana;
CREATE USER viviana@'%';
```

However, such accounts may be unable to connect from localhost if an anonymous user `''@'%'` is present. See localhost and % for the details.

Accounts are not bound to a specific database. They are global. Once an account is created, it is possible to assign it permissions on any existing or non existing database.

The sql_mode system variable has a NO_AUTO_CREATE_USER flag. In recent MariaDB versions it is enabled by default. If it is not enabled, a GRANT statement specifying privileges for a non-existent account will automatically create that account.

For more information: Account Management SQL Commands.

Setting or Changing Passwords

Accounts with the same username can have different passwords.

By default, an account has no password. A password can be set, or changed, in the following way:

- By specifying it in CREATE USER.
- By the user, with SET PASSWORD.
- By root, with `SET PASSWORD` or ALTER USER.

With all these statements (`CREATE USER`, `ALTER USER`, `SET PASSWORD`) it is possible to specify the password in plain or as a hash:

```
-- specifying plain passwords:
CREATE USER tom@'%.example.com' IDENTIFIED BY 'plain secret';
ALTER USER tom@'%.example.com' IDENTIFIED BY 'plain secret';
SET PASSWORD = 'plain secret';
-- specifying hashes:
CREATE USER tom@'%.example.com' IDENTIFIED BY PASSWORD 'secret hash';
ALTER USER tom@'%.example.com' IDENTIFIED BY PASSWORD 'secret hash';
SET PASSWORD = PASSWORD('secret hash');
```

The PASSWORD() function uses the same algorithm used internally by MariaDB to generate hashes. Therefore it can be used to get a hash from a plain password. Note that this function should not be used by applications, as its output may depend on MariaDB version and configuration.

`SET PASSWORD` applies to the current account, by default. Superusers can change other accounts passwords in this way:

```
SET PASSWORD FOR tom@'%.example.com' = PASSWORD 'secret hash';
```

Passwords can have an expiry date, set by default_password_lifetime. To set a different date for a particular user:

```
CREATE USER 'tom'@'%.example.com' PASSWORD EXPIRE INTERVAL 365 DAY;
```

To set no expiry date for a particular user:

```
CREATE USER 'tom'@'%.example.com' PASSWORD EXPIRE NEVER;
```

For more details, see User Password Expiry.

It is also possible to lock an account with immediate effect:

```
CREATE USER 'tom'@'%.example.com' ACCOUNT LOCK;
```

See Account Locking for more details.

Authentication Plugins

MariaDB supports authentication plugins. These plugins implement user's login and authorization before they can use MariaDB.

Each user has one or more authentication plugins assigned. The default one is mysql_native_password. It is the traditional login using the username and password set in MariaDB, as described above.

On UNIX systems, root is also assigned the unix_socket plugin, which allows a user logged in the operating system to be recognized by MariaDB.

Windows users may be interested in the named pipe and GSSAPI plugins. GSSAPI also requires the use of a plugin on the client side.

A plugin can be assigned to a user with `CREATE USER`, `ALTER USER` or `GRANT`, using the `IDENTIFIED VIA` syntax. For example:

```
CREATE USER username@hostname IDENTIFIED VIA gssapi;
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA named_pipe;
```

TLS connections

A particular user can be required to use TLS connections. Additional requirements can be set:

- Having a valid X509 certificate.
- The certificate may be required to be issued by a particular authority.
- A particular certificate subject can be required.
- A particular certificate cipher suite can be required.

These requirements can be set with `CREATE USER`, `ALTER USER` or `GRANT`. For the syntax, see CREATE USER.

MariaDB can be bundled with several cryptography libraries, depending on its version. For more information about the libraries, see TLS and Cryptography Libraries Used by MariaDB.

For more information about secure connections, see Secure Connections Overview.

Permissions

Permissions can be granted to accounts. As mentioned before, the specified accounts can actually be patterns, and multiple accounts may match a pattern. For example, in this example we are creating three accounts, and we are assigning permissions to all of them:

```
CREATE USER 'tom'@'example.com';
CREATE USER 'tom'@'123.123.123.123;
CREATE USER 'tom'@'tomlaptop';
GRANT USAGE ON *.* TO tom@'%';
```

The following permission levels exist in MariaDB:

- Global privileges;

- Database privileges;
- Table privileges;
- Column privileges;
- Function and procedure privileges.

Note that database and schema are synonymous in MariaDB.

Permissions can be granted for non-existent objects that could exist in the future.

The list of supported privileges can be found in the GRANT page. Some highlights can be useful for SQL Server users:

- `USAGE` privilege has no effect. The `GRANT` command fails if we don't grant at least one privilege; but sometimes we want to run it for other purposes, for example to require a user to use TLS connections. In such cases, it is useful to grant `USAGE`.
- Normally we can obtain a list of all databases for which we have at least one permission. The `SHOW DATABASES` permission allows getting a list of all databases.
- There is no `SHOWPLAN` privilege in MariaDB. Instead, EXPLAIN requires the `SELECT` privilege for each accessed table and the `SHOW VIEW` privilege for each accessed view.
- The same permissions are needed to see a table structure (`SELECT`) or a view definition (`SHOW VIEW`).
- `REFERENCES` has no effect.

MariaDB does not support negative permissions (the `DENY` command).

Some differences concerning the SQL commands:

- In MariaDB `GRANT` and `REVOKE` statements can only assign/revoke permissions to one user at a time.
- While we can assign/revoke privileges at column level, we have to run a `GRANT` or `REVOKE` statement for each column. The `table (column_list)` syntax is not recognized by MariaDB.
- In MariaDB it is not needed (or possible) to specify a class type.

Roles

MariaDB supports roles. Permissions can be assigned to roles, and roles can be assigned to accounts.

An account may have zero or one default roles. A default role is a role that is automatically active for a user when they connect. To assign an account or remove a default role, these SQL statements can be used:

```
SET DEFAULT ROLE some_role FOR username@hostname;
SET DEFAULT ROLE NONE FOR username@hostname;
```

Normally a role is not a default role. If we assign a role in this way:

```
GRANT some_role TO username@hostname;
```

…the user will not have that role automatically enabled. They will have to enable it explicitly:

```
SET ROLE some_role;
```

MariaDB does not have predefined roles, like public.

For an introduction to roles, see Roles Overview.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Replication Overview for SQL Server Users

MariaDB supports the following types of replication:

- Asynchronous replication.
- Semi-synchronous replication.
- Galera Cluster.

MariaDB starting with 10.5.1

Note: in the snippets in this page, several SQL statements use the keyword `SLAVE`. This word is considered inappropriate by some persons or cultures, so from MariaDB 10.5 it is possible to use the `REPLICA` keyword, as a synonym.
Similar synonyms will be created in the future for status variables and system variables. See MDEV-18777 to track the status of these changes.

Asynchronous Replication

The original MariaDB replication system is asynchronous primary-replica replication.

A primary needs to have the binary log enabled. The primary logs all data changes in the binary log. Every *event* (a binary log entry) is sent to all the replicas.

For a high-level description of the binary log for SQL Server users, see Understanding MariaDB Architecture.

The events can be written in two formats: as an SQL statement (*statement-based replication*, or SBR), or as a binary representation of the change (*row-based replication*, or RBR). The former is generally slower, because the statement needs to be re-executed by the replicas. It is also less reliable, because some SQL statements are not deterministic, so they could produce different results on the replicas. On the other hand row-based replication could make the binary log much bigger, and require more network traffic. For this reason, DML statements are always logged in statement format.

For more details on replication formats, see binary log formats.

The replicas have an I/O thread that receives the binary log events and writes them to the relay log. These events are then read by the SQL thread. This thread could directly apply the changes to the local databases, and this was the only option before MariaDB 10.0.5. If parallel replication is enabled, the SQL thread hands the events to the worker thread, that apply them to the databases. The latter method is recommended for performance reasons.

When a replica cannot apply an event to the local data, the SQL thread stops. This happens, for example, if the event is a row deletion but that row doesn't exist on the replica. There can be several reasons for this, for example non-deterministic statements, or a user deleted the row in the replica. To reduce the risk, it is recommended to set read_only to 1 in the replicas.

SHOW SLAVE STATUS has columns named `Slave_SQL_State` and `Slave_IO_State` that show, respectively, if the SQL thread and the IO thread are running. If they are not, the column `Last_IO_Errno` and `Last_IO_Error` (for the IO thread) or `Last_SQL_Errno` and `Last_SQL_Error` (for the SQL thread) show what the problem is.

In a replication chain, every server must have a unique server_id.

For more information on replication, see standard replication.

Binary Log Coordinates, Relay Log Coordinates and GTID

The binary log coordinates provide a way to identify a certain data change made by a server. Coordinates consist of a file name and the position of the latest event, expressed as an integer. The last event coordinates can be seen with the SHOW MASTER STATUS columns `File` and `Position`. mariadb-dump includes them in a dump if the `--master-data` option is used.

A replica uses primary binary log coordinates to identify the last event it read. This can be seen with the SHOW SLAVE STATUS columns `Master_Log_File` and `Read_Master_Log_Pos`.

The columns `Relay_Master_Log_File` and `Exec_Master_Log_Pos` identify the primary event that corresponds to the last event applied by the SQL thread.

The replica relay log also has coordinates. The coordinates of the last applied event can be seen with the `SHOW SLAVE STATUS` columns `Relay_Log_File` and `Relay_Log_Pos`.

To easily find out how far the replica is lagging behind the primary, we can look at `Seconds_Behind_Master`.

Coordinates represented in this way have a problem: they are different on each server. Each server can use files with different (or the same) names, depending on its configuration. And files can be rotated at different times, including when a user runs FLUSH LOGS. By enabling the GTID (global transaction id) an event will have the same id on the primary and on all the replicas.

When GTID is enabled, `SHOW SLAVE STATUS` shows two GTIDs: `Gtid_IO_Pos` is the last event written into the relay log, and `Gtid_Slave_Pos` is the last event applied by the SQL thread. There is no need for a column identifying the same event in the primary, because the id is the same.

Provisioning a Replica

MariaDB does not have an equivalent to SQL Server's snapshot replication.

To setup a replica, it is necessary to manually provision it. It can be provisioned from the primary in this way:

- A backup from the primary must be restored on the new replica;
- The binary log coordinates at the moment of the backup should be set as replication coordinates in the replica, via CHANGE MASTER TO.

However, if there is at least one existing replica, it is better to use it to provision the new replica:

- A backup from the existing replica must be restored in the new replica;
- The backup should include the system tables. In this way it will not be necessary to set the correct coordinates manually.

For more information see Setting Up Replication and Setting up a Replica with mariadb-backup.

Replication and Permissions

A replica connects to a primary using its credentials. See CHANGE MASTER TO.

The appropriate account must be created in the primary, and it needs to have the `REPLICATION SLAVE` permission.

See Setting Up Replication for more information.

Parallel Replication and Group Commit

MariaDB uses group commit, which means that a group of events are physically written in the binary log altogether. This reduces the number of IOPS (input/output operations per second). Group commit cannot be disabled, but it can be tuned with variables like binlog_commit_wait_count and binlog_commit_wait_usec.

Replicas can apply the changes using multiple threads. This is known as parallel replication. Before MariaDB 10.0.5 only one thread was used to apply changes. Since a primary can use many threads to write data, mono-thread replication is a well-known bottleneck. Parallel replication is not enabled by default. To use it, set the slave_parallel_threads variable to a number greater than 1. If replication is running, the replica threads must be stopped in order to change this value:

```
STOP SLAVE SQL_THREAD;
SET GLOBAL slave_parallel_threads = 4;
START SLAVE SQL_THREAD;
```

There are different parallel replication styles available: in-order and out-of-order. The exact mode in use is determined by the slave_parallel_mode system variable. In parallel replication, the events are not replicated exactly in the same order as they occurred in the primary. But with an in-order replication mode the commit phase is always applied simultaneously. In this way data in the replica always reflect data as they have been in the primary at a certain point in time. Out-of-order replication is faster because there is less queuing, but it's not completely consistent with the primary. If two transactions modified different sets of rows in the primary, they could become visible in the replica in a different order.

`conservative` relies on primary group commit: events in different groups are executed in a parallel way.

`optimistic` does not try to find out which transaction can be executed in a parallel way - except for transactions that conflicted on the primary. Instead, it always tries to apply many events together, and rolls transactions back when there is a conflict.

`aggressive` is similar to optimistic, but it does not take into account which transactions conflicted in the primary.

`minimal` applies commits together, but all other events are applied in order.

Out-of-order replication cannot be enabled automatically by changing a variable in the replica. Instead, it must be enabled by the applications that run transactions in the primary. They can do this if the GTID is enabled. They can set different values for the gtid_domain_id variable in different transactions. This shifts a lot of responsibility to the application layer; however, if the application is aware of which transactions are not going to conflict and this information allows one to sensibly increase the parallelism, and using out-of-order replication can be a good idea.

Even if out-of-order replication is not normally used, it can be a good idea to use it for long running transactions or ALTER TABLEs, so they can be applied at the same time as normal operations that are not conflicting.

The impact of the number of threads and mode on performance can be partly seen with SHOW PROCESSLIST, which shows the state of all threads. This includes the replication worker threads, and shows if they are blocking each other.

Differences Between the Primary and the Replicas

As a general rule, we want the primary and the replicas to contain exactly the same data. In this way, no conflicts are possible. Conflicts are the most likely cause of replication outages.

To reduce the possible causes of conflicts, the following best practices are recommended:

- Users must not change data in the replica directly. Set read_only to 1. Note that this won't prevent root from making changes.
- Use the same table definitions in the primary and in the replica.
- Use `ROW` binary log format on the primary.

Another cause of inconsistencies include MariaDB bugs and failover in case the primary crashes.

An open source third party tool is available to check if the primary and a replica are consistent. It is called pt-table-checksum. Another tool, pt-table-sync, can be used to eliminate the differences. Both are part of Percona Toolkit. The advice is to run pt-table-checksum periodically, and use pt-table-sync if inconsistencies are found.

If a replication outage occurs because an inconsistency is found, sometimes we want to quickly bring the replica up again as quickly as possible, and solve the core problem later. If GTID is not used, a way to do this is to run SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1, which skips the problematic replication event.

If GTID is used, the gtid_slave_pos variable can be used instead. See the link for an explanation of how it works.

There are ways to have different data on the replicas. For example:

- Multi-source replication is possible. In this way, a replica will replicate data from multiple primaries. This feature is described below.

- Replication filters are supported. This allows one to exclude or include in replication specific tables, entire databases, or tables whose name matches a certain pattern. This allows one to avoid replicating data that is present in the primary but can always be rebuilt.
- Differences in table definitions are also possible. For example, a replica could have more columns or less columns compared to the primary. In this way we can avoid replicating columns whose values can be rebuilt. Or we can add columns for analytics purposes, without having them in the primary. Be sure to understand the limitations and risks of this technique.

Delayed Replication

MariaDB supports delayed replication. This is the equivalent of setting a `pollinginterval` in SQL Server.

To delay replication in a MariaDB replica, use CHANGE MASTER TO to specify a delay in seconds.

For more information, see Delayed Replication.

Multi-Source Replication

Multi-source replication is an equivalent to peer-to-peer replication, available in SQL Server Enterprise Edition.

A MariaDB replica can replicate from any number of primaries. It is very important that different primaries don't have the same tables. Otherwise there could be conflicts between data changes made on different primaries, and this will result in a replication outage.

In multi-source replication different channels exist, one for each primary.

This changed the way SQL replication statements work. SHOW PROCESSLIST returns a different row for each channel. Several statements, like CHANGE MASTER TO, START SLAVE or STOP SLAVE. accept a parameter which specifies which replication channel they should affect. For example, to stop a channel called `wp1`:

```
STOP SLAVE "wp1";
```

Furthermore, variables that affect parallel replication can be prefixed with a channel name. This allow one to only use parallel replication for certain channels, or to tune it differently for each channel. For example, to enable parallel replication on a channel called `wp1`:

```
SET GLOBAL wp1.slave_parallel_threads = 4;
```

Dual Primary

It is possible to configure two servers in a way that each of them acts as a primary for the other server.

In this way, data could theoretically be inserted into any of these servers, and will then be replicated to the other server. However, in such a configuration conflicts are very likely. So it is impractical to use this technique to scale writes.

A dual primary (or primary-primary) configuration however can be useful for failover. In this case we talk about an *active primary* that receives reads and writes from the clients, and a *passive primary* that is not used until the active primary crashes.

Several problems should be considered in this scenario:

- If the active primary crashes, it is very possible that the passive primary did not receive all events yet, because replication is asynchronous. If the primary data are lost (for example because the disk is damaged), some data are also lost.
- If data is not lost, when we bring the primary up again, the latest events will be replicated by the other server. There could be conflicts that will break replication.

- When is the active primary considered down? Even if a server cannot reach it, the active primary could be running and it could be able to communicate with the passive primary. Switching the clients to the passive primary could lead to unnecessary problems. It is a good idea to always check `SHOW SLAVE STATUS` to be sure that the two primary are not communicating.
- If we want to have more replicas, we should attach some of them to the active primary, and some of them to the passive primary. The reason is that when a server crashes, its replicas stop receiving any data. Failover is still possible, but it's better to have some servers that will not need any failover.

A safe primary-primary configuration where both servers accept writes, however, is possible. This is the case is data never conflicts. For example, the two servers could accept writes on different databases. We will have to decide what should happens in case of a server crash:

- Writes can be stopped until the server is up again. Reads can be sent to the other server, but keep in mind that the most recently written data could be missing.
- Both writes and reads can failover to the other server. All the problems mentioned above may apply to this situation.

See Sveta Smirnova's slides at MariaDB Day 2020: "How Safe is Asynchronous Master-Master Setup?".

Semi-Synchronous Replication

Semi-synchronous replication was initially implemented as a plugin, in MySQL. Two different plugins needed to be used, one on the primary and the other on the replicas. Starting from MariaDB 10.3.3 it is built-in, which improved its performance.

The problem with standard replication is that there is no guarantee that it will not lag, even by long amounts of time. Semi-synchronous replication reduces this problem, at the cost of reducing the speed of the primary.

In semi-synchronous replication, when a transaction is committed on the primary, the primary does not immediately return control to the client. Instead, it sends the event to the replicas. After one replica reported that the commit was executed with success, the primary reports success to the client.

Semi-synchronous replication is useful for failover, therefore a dual primary setup is not needed in this case. If the primary crashes, the most up-to-date replica can be promoted to primary without losing any data.

Enabling Semi-Synchronous Replication

Semi-synchronous replication can be enabled at runtime in this way on the primary:

```
SET GLOBAL rpl_semi_sync_master_enabled = ON;
```

Semi-synchronous replication is not used until it has been enabled on the replicas also. If the replicas are already replicating, the io_thread needs to be stopped and restarted. This can be done as follows:

```
SET GLOBAL rpl_semi_sync_slave_enabled = ON;
STOP SLAVE IO_THREAD;
START SLAVE IO_THREAD;
```

Tuning the Wait Point and the Primary Timeout

The most important aspects to tune are the wait point and the primary timeout.

When the binary log is enabled, transactions must be committed both in the storage engine (usually InnoDB) and in the binary log. Semi-synchronous replication requires that the transaction is also acknowledged by at least one replica before the primary can report success to the client.

The wait point determines at which point the primary must stop and wait for a confirmation from a replica. This is an important decision from disaster recovery standpoint, in case the primary crashes when a transaction is not fully

committed. The rpl_semi_sync_master_wait_point is used to set the wait point, Its allowed values are:

- `AFTER_SYNC`: After committing the transaction in the binary log, but before committing it to the storage engine. After a crash, a transaction may be present in the binary log even if it was not committed.
- `AFTER_COMMIT`. After committing a transaction both in the binary log and in the storage engine. In case of a crash, a transaction could possibly be committed in the primary but not replicated in the slaves. This is the default.

Primary timeout is meant to avoid that a primary remains stuck for a long time, or virtually forever, because no replica acknowledges a transaction. If primary timeout is reached, the primary switches to asynchronous replication. Before doing that, the primary writes an error in the error log and increments the Rpl_semi_sync_master_no_times status variable.

The timeout is set via the rpl_semi_sync_master_timeout variable.

Galera Cluster

Galera is a technology that implements virtually synchronous, primary-primary replication for a cluster of MariaDB servers.

Raft and the Primary Cluster

Nodes of the cluster communicate using the Raft protocol. In case the cluster is partitioned or some nodes crash, a cluster knows that it's still the *primary cluster* if it has the *quorum*: half of the nodes + 1. Only the primary cluster accepts reads and writes.

For this reason a cluster should consist of an odd number of nodes. Imagine for example that a cluster consists of two nodes: if one of them crashes of the connection between them is interrupted, there will be no primary cluster.

Transaction Certification

A transaction can be executed against any node. The node will use a 2-phase commit. After running the transaction locally, the node will ask other nodes to *certify* it. This means that other nodes will receive it, and will try to apply it, and will report success or a failure. The node that received the transaction will not wait for an answer from all the nodes. Once it succeeded on more than half of the nodes (the quorum) the node will run the final commit and data becomes visible.

It is desirable to write data on only one node (unless it fails), or write different databases on different nodes. This will minimize the risk of conflicts.

Galera Cache and SST

Data changes applied are recorded for some time in the *Galera cache*. This is an on-disk cache, written in a circularly written file.

The size of Galera cache can be tuned using the wsrep_provider_options system variable, which contains many flags. We need to tune gcache.size. To tune it, add a line similar to the following to a configuration file:

```
wsrep_provider_options = 'gcache.size=2G';
```

If a single transaction is bigger than half of the Galera cache, it needs to be written in a separate file, as *on-demand pages*. On-demand pages are regularly replaced. Whether a new page replaces an old one depends on another wsrep_provider_options flag: `wsrep_provider_options#gcachekeep_pages_size|gcache.keep_pages_size`, which limits the total size of on-demand pages.

When a node is restarted (after a crash or for maintenance reasons), it will need to receive all the changes that were written by other nodes since the moment it was unreachable. A node is therefore chosen as a donor, possibly using

the gcssync_donor wsrep_provider_options flag.

If possible, the donor will send all the recent changes, reading them from the Galera cache and on-demand pages. However, sometimes the Galera cache is not big enough to contain all the needed changes, or the on-demand pages have been overwritten because `gcache.keep_pages_size` is not big enough. In these cases, a State Snapshot Transfer (SST) needs to be sent. This means that the donor will send the whole dataset to the restarted node. Most commonly, this happens using the mariadb-backup method.

Flow Control

While transaction certification is synchronous, certified transactions are applied locally in asynchronous fashion. However, a node should never lag too much behind others. To avoid that, a node may occasionally trigger a mechanism called *flow control* to ask other nodes to stop replication until its situation improves. Several wsrep_provider_options flags affect flow control.

gcs.fc_master_slave should normally be set to 1 if all writes are sent to a single node.

gcs.fc_limit is tuned automatically, unless gcs.fc_master_slave is set to 0. The *receive queue* (the transactions received and not yet applied) should not exceed this limit. When this happens, flow control is triggered by the node to pause other node's replication.

Once flow control is activated, gcs.fc_factor determines when it is released. It is a number from 0 to 1, and it represents a fraction. When the receive queue is below this fraction, the flow control is released.

Flow control and the receive queue can and should be monitored. The most useful metrics are:

- wsrep_flow_control_paused indicates how many times the replication has been paused as requested by other nodes, since the last `FLUSH STATUS`.
- wsrep_flow_control_sent indicates how many times this node requested other nodes to pause replication.
- wsrep_local_recv_queue is the size of the receive queue.

Configuration

Galera is implemented as a plugin. Starting from version 10.1, MariaDB comes with Galera pre-installed, but not in use by default. To enable it one has to set the wsrep_on system variable.

Like asynchronous replication, Galera uses the binary log. It also requires that data changes are logged in the `ROW` format.

For other required settings, see Mandatory Options.

Galera Limitations

Galera is not suitable for all databases and workloads.

- Galera only replicates InnoDB tables. Other storage engines should not be used.
- For performance reasons, it is highly desirable that all tables have a primary key.
- Long transactions will damage performance.
- Some applications use an integer AUTO_INCREMENT primary key. In case of failover from a crashed node to another, Galera does not guarantee that `AUTO_INCREMENT` follows a chronological order. Therere, applications should use TIMESTAMP columns for chronological order instead.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Transactions and Isolation Levels for SQL Server Users

This page explains how transactions work in MariaDB, and highlights the main differences between MariaDB and SQL Server transactions.

Note that XA transactions are handled in a completely different way and are not covered in this page. See XA Transactions.

Missing Features

These SQL Server features are not available in MariaDB:

- Autonomous transactions;
- Distributed transactions.

Transactions, Storage Engines and the Binary Log

In MariaDB, transactions are optionally implemented by storage engines. The default storage engine, InnoDB, fully supports transactions. Another transactional storage engine is MyRocks. Most storage engines are not transactional, therefore they should not considered general purpose engines.

Most of the information in this page refers to generic MariaDB server behaviors or InnoDB. For MyRocks please check the proper documentation section.

Writing into a non-transactional table in a transaction can still be useful. The reason is that a metadata lock is acquired on the table for the duration of the transaction, so that ALTER TABLEs are queued.

It is possible to write into transactional and non-transactional tables within a single transaction. It is important to remember that non-transactional engines will have the following limitations:

- In case of rollback, changes to non-transactional engines won't be undone. We will receive a warning `1196` which reminds us of this.
- Data in transactional tables cannot be changed by other connections in the middle of a transaction, but data in non-transactional tables can.
- In case of a crash, committed data written into a transactional table can always be recovered, but this is not necessarily true for non-transactional tables.

If the binary log is enabled, writing into different transactional storage engines in a single transaction, or writing into transactional and non-transactional engines inside the same transaction, implies some extra work for MariaDB. It needs to perform a two-phase commit to be sure that changes to different tables are logged in the correct order. This affects the performance.

Transaction Syntax

The first read or write to an InnoDB table starts a transaction. No data access is possible outside a transaction.

By default autocommit is on, which means that the transaction is committed automatically after each SQL statement. We can disable it, and manually commit transactions:

```
SET SESSION autocommit = 0;
SELECT ... ;
DELETE ... ;
COMMIT;
```

Whether autocommit is enabled or not, we can start transactions explicitly, and they will not be automatically committed:

```
START TRANSACTION;
SELECT ... ;
DELETE ... ;
COMMIT;
```

`BEGIN` can also be used to start a transaction, but does not work in stored procedures.

Read-only transactions are also available using `START TRANSACTION READ ONLY`. This is a small performance optimization. MariaDB will issue an error when trying to write data in the middle of a read-only transaction.

Only DML statements are transactional and can be rolled back. This may change in a future version, see MDEV-17567 - Atomic DDL and MDEV-4259 - transactional DDL.

Changing autocommit and explicitly starting a transaction will implicitly commit the active transaction, if any. DDL statements, and several other statements, implicitly commit the active transaction. See SQL statements That Cause an Implicit Commit for the complete list of these statements.

A rollback can also be triggered implicitly, when certain errors occur.

You can experiment with transactions to check in which cases they implicitly commit or rollback. The in_transaction system variable can help: it is set to 1 when a transaction is in progress, or 0 when no transaction is in progress.

This section only covers the basic syntax for transactions. Much more options are available. For more information, see Transactions.

Constraint Checking

MariaDB supports the following constraints:

- Primary keys
- UNIQUE
- CHECK
- Foreign keys

In some databases, constraints can temporarily be violated during a transaction, and their enforcement can be deferred to the commit time. SQL Server does not support this, and always validates data against constraints at the end of each statement.

MariaDB does something different: it always checks constraints after each row change. There are cases this policy makes some statements fail with an error, even if those statements would work on SQL Server.

For example, suppose you have an `id` column that is the primary key, and you need to increase its value for some reason:

```
SELECT id FROM customer;
```

```
+----+
| id |
+----+
|  1 |
|  2 |
|  3 |
|  4 |
|  5 |
+----+
```

```
UPDATE customer SET id = id + 1;
```

```
ERROR 1062 (23000): Duplicate entry '2' for key 'PRIMARY'
```

The reason why this happens is that, as the first thing, MariaDB tries to change 1 to 2, but a value of 2 is already present in the primary key.

A solution is to use this non-standard syntax:

```
UPDATE customer SET id = id + 1 ORDER BY id DESC;
Query OK, 5 rows affected (0.00 sec)
Rows matched: 5  Changed: 5  Warnings: 0
```

Changing the ids in reversed order won't duplicate any value.

Similar problems can happen with `CHECK` constraints and foreign keys. To solve them, we can use a different approach:

```
SET SESSION check_constraint_checks = 0;
-- run some queries
-- that temporarily violate a CHECK clause
SET SESSION check_constraint_checks = 1;

SET SESSION foreign_key_checks = 0;
-- run some queries
-- that temporarily violate a foreign key
SET SESSION foreign_key_checks = 1;
```

The last solutions temporarily disable `CHECK` constraints and foreign keys. Note that, while this may solve practical problems, it is dangerous because:

- This doesn't disable a single `CHECK` or foreign key, but also others, that you don't expect to violate.
- This doesn't defer the constraint checks, but it simply disables them for a while. This means that, if you insert some invalid values, they will not be detected.

See check_constraint_checks and foreign_key_checks system variables.

Isolation Levels and Locks

For more information about MariaDB isolation levels see SET TRANSACTION.

Locking Reads

In MariaDB, the locks acquired by a read do not depend on the isolation level (with one exception noted below).

As a general rule:

- Plain SELECTs are not locking, they acquire snapshots instead.
- To force a read to acquire a shared lock, use SELECT … LOCK IN SHARED MODE.
- To force a read to acquire an exclusive lock, use SELECT … FOR UPDATE.

Changing the Isolation Level

The default, the isolation level in MariaDB is `REPEATABLE READ`. This can be changed with the tx_isolation system variable.

Applications developed for SQL Server and later ported to MariaDB may run with `READ COMMITTED` without problems. Using a stricter level would reduce scalability. To use `READ COMMITTED` by default, add the following line to the MariaDB configuration file:

```
tx_isolation = 'READ COMMITTED'
```

It is also possible to change the default isolation level for the current session:

```
SET SESSION tx_isolation = 'read-committed';
```

Or just for one transaction, by issuing the following statement before starting a transaction:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

How Isolation Levels are Implemented in MariaDB

MariaDB supports the following isolation levels:

- `READ UNCOMMITTED`
- `READ COMMITTED`
- `REPEATABLE READ`
- `SERIALIZABLE`

MariaDB isolation levels differ from SQL Server in the following ways:

- `REPEATABLE READ` does not acquire share locks on all read rows, nor a range lock on the missing values that match a `WHERE` clause.
- It is not possible to change the isolation level in the middle of a transaction.
- `SNAPSHOT` isolation level is not supported. Instead, you can use `START TRANSACTION WITH CONSISTENT SNAPSHOT` to acquire a snapshot at the beginning of the transaction. This is compatible with all isolation levels.

Here is an example of `WITH CONSISTENT SNAPSHOT` usage:

```
-- session 1
SELECT * FROM t1;
```

```
+----+
| id |
+----+
|  1 |
+----+
```

```
SELECT * FROM t2;
```

```
+----+
| id |
+----+
|  1 |
+----+
```

```
START TRANSACTION WITH CONSISTENT SNAPSHOT;
```

```
-- session 2
INSERT INTO t1 VALUES (2);
```

```
-- session 1
SELECT * FROM t1;
```

```
+----+
| id |
+----+
|  1 |
+----+
```

```
-- session 2
INSERT INTO t2 VALUES (2);
```

```
-- session 1
SELECT * FROM t2;
```

```
+----+
| id |
+----+
|  1 |
+----+
```

As you can see, session 1 uses `WITH CONSISTENT SNAPSHOT`, thus it sees all tables as they were when the transaction begun.

Avoiding Lock Waits

When we try to read or modify a row that is exclusive-locked by another transaction, our transaction is queued until that lock is released. There could be more queued transactions waiting to acquire the same lock, in which case we will wait even more.

There is a timeout for such waits, defined by the innodb_lock_wait_timeout variable. If it is set to 0, statements that encounter a row lock will fail immediately. When the timeout is exceeded, MariaDB produces the following error:

```
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
```

It is important to note that this variable has two limitations (by design):

- It only affects transactional statements, not statements like `ALTER TABLE` or `TRUNCATE TABLE`.
- It only concerns row locks. It does not put a timeout on metadata locks, or table locks acquired - for example - with the LOCK TABLES statement.

Note however that lock_wait_timeout can be used for metadata locks.

There is a special syntax that can be used with `SELECT` and some non-transactional statements including `ALTER TABLE`: the WAIT and NOWAIT clauses. This syntax puts a timeout in seconds for all lock types, including row locks, table locks, and metadata locks. For example:

```
Session 1:
START TRANSACTION;
-- let's acquire a metadata lock
SELECT id FROM t WHERE 0;

Session 2:
DROP TABLE t WAIT 0;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
```

InnoDB Transactions

InnoDB Lock Types

InnoDB locks are classified based on what exactly they lock, and which operations they lock.

The first classification is the following:

- Record Locks lock a row or, more precisely, an index entry.
- Gap Locks lock an interval between two index entries. Note that indexes have virtual values of -Infinum and Infinum, so a gap lock can cover the gap before the first or after the last index entry.
- Next-Key Locks lock an index entry and the gap between it and the next entry. They're a combination of record locks and gap locks.
- Insert Intention Locks are gap locks acquired before inserting a new row.

Lock modes are the following:

- Exclusive Locks (X) are generally acquired on writes, e.g. immediately before deleting a row. Only one exclusive lock can be acquired on a resource simultaneously.
- Shared Locks (S) can be acquired on reads. Multiple shared locks can be acquired at the same time (because the rows are not supposed to change when shared-locked) but are incompatible with exclusive locks.
- Intention locks (IS, XS) are acquired when it is not possible to acquire an exclusive lock or a shared lock. When a lock on a row or gap is released, the oldest intention lock on that resource (if any) is converted to an X or S lock.

For more information see InnoDB Lock Modes.

Information Schema

Querying the information_schema is the best way to see which transactions have acquired some locks and which transactions are waiting for some locks to be released.

In particular, check the following tables:

- INNODB_LOCKS: requests for locks not yet fulfilled, or that are blocking another transaction.
- INNODB_LOCK_WAITS: queued requests to acquire a lock.
- INNODB_TRX: information about all currently executing InnoDB transactions, including SQL queries that are running.

Here is an example of their usage.

```
-- session 1
START TRANSACTION;
UPDATE t SET id = 15 WHERE id = 10;

-- session 2
DELETE FROM t WHERE id = 10;

-- session 1
USE information_schema;
SELECT l.*, t.*
    FROM information_schema.INNODB_LOCKS l
    JOIN information_schema.INNODB_TRX t
        ON l.lock_trx_id = t.trx_id
    WHERE trx_state = 'LOCK WAIT' \G
```

```
*************************** 1. row ***************************
                   lock_id: 840:40:3:2
               lock_trx_id: 840
                 lock_mode: X
                 lock_type: RECORD
                lock_table: `test`.`t`
                lock_index: PRIMARY
                lock_space: 40
                 lock_page: 3
                  lock_rec: 2
                 lock_data: 10
                    trx_id: 840
                 trx_state: LOCK WAIT
               trx_started: 2019-12-23 18:43:46
     trx_requested_lock_id: 840:40:3:2
          trx_wait_started: 2019-12-23 18:43:46
                trx_weight: 2
       trx_mysql_thread_id: 46
                 trx_query: DELETE FROM t WHERE id = 10
       trx_operation_state: starting index read
         trx_tables_in_use: 1
         trx_tables_locked: 1
          trx_lock_structs: 2
     trx_lock_memory_bytes: 1136
           trx_rows_locked: 1
         trx_rows_modified: 0
   trx_concurrency_tickets: 0
       trx_isolation_level: REPEATABLE READ
         trx_unique_checks: 1
    trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
          trx_is_read_only: 0
trx_autocommit_non_locking: 0
```

Deadlocks

InnoDB detects deadlocks automatically. Since this consumes CPU time, some users prefer to disable this feature by setting the innodb_deadlock_detect variable to 0. If this is done, locked transactions will wait until the they exceed the innodb_lock_wait_timeout. Therefore it is important to set innodb_lock_wait_timeout to a very low value, like 1.

When InnoDB detects a deadlock, it kills the transaction that modified the least amount of data. The client will receive the following error:

```
ERROR 1213 (40001): Deadlock found when trying to get lock; try restarting transaction
```

The latest detected deadlock, and the killed transaction, can be viewed in the output of SHOW ENGINE InnoDB STATUS. Here's an example:

```
------------------------
LATEST DETECTED DEADLOCK
------------------------
2019-12-23 18:55:18 0x7f51045e3700
*** (1) TRANSACTION:
TRANSACTION 847, ACTIVE 10 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 4 lock struct(s), heap size 1136, 3 row lock(s), undo log entries 1
MySQL thread id 46, OS thread handle 139985942054656, query id 839 localhost root Updating
delete from t where id = 10
*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 40 page no 3 n bits 80 index PRIMARY of table `test`.`t` trx id 847 lock_mode X
locks rec but not gap waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 3; compact format; info bits 32
 0: len 4; hex 8000000a; asc     ;;
 1: len 6; hex 00000000034e; asc      N;;
 2: len 7; hex 760000019c0495; asc v      ;;

*** (2) TRANSACTION:
TRANSACTION 846, ACTIVE 25 sec starting index read
mysql tables in use 1, locked 1
3 lock struct(s), heap size 1136, 2 row lock(s), undo log entries 1
MySQL thread id 39, OS thread handle 139985942361856, query id 840 localhost root Updating
delete from t where id = 11
*** (2) HOLDS THE LOCK(S):
RECORD LOCKS space id 40 page no 3 n bits 80 index PRIMARY of table `test`.`t` trx id 846 lock_mode X
locks rec but not gap
Record lock, heap no 2 PHYSICAL RECORD: n_fields 3; compact format; info bits 32
 0: len 4; hex 8000000a; asc     ;;
 1: len 6; hex 00000000034e; asc      N;;
 2: len 7; hex 760000019c0495; asc v      ;;

*** (2) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 40 page no 3 n bits 80 index PRIMARY of table `test`.`t` trx id 846 lock_mode X
locks rec but not gap waiting
Record lock, heap no 3 PHYSICAL RECORD: n_fields 3; compact format; info bits 32
 0: len 4; hex 8000000b; asc     ;;
 1: len 6; hex 00000000034f; asc      O;;
 2: len 7; hex 770000019d031d; asc w      ;;

*** WE ROLL BACK TRANSACTION (2)
```

The latest detected deadlock never disappears from the output of `SHOW ENGINE InnoDB STATUS`. If you cannot see any, MariaDB hasn't detected any InnoDB deadlocks since the last restart.

Another way to monitor deadlocks is to set innodb_print_all_deadlocks to 1 (0 is the default). InnoDB will log all detected deadlocks into the error log.

This page is licensed: CC BY-SA / Gnu FDL

Moving Data Between SQL Server and MariaDB

There are several ways to move data between SQL Server and MariaDB. Here we will discuss them and we will highlight some caveats.

Moving Data Definition from SQL Server to MariaDB

To copy SQL Server data structures to MariaDB, one has to:

1. Generate a CSV file from SQL Server data.
2. Modify the syntax so that it works in MariaDB.
3. Run the file in MariaDB.

Variables That Affect DDL Statements

DDL statements are affected by some server system variables.

sql_mode determines the behavior of some SQL statements and expressions, including how strict error checking is, and some details regarding the syntax. Objects like stored procedures, stored functions triggers and views, are always executed with the sql_mode that was in effect during their creation. sql_mode='MSSQL' can be used to have MariaDB behaving as close to SQL Server as possible.

innodb_strict_mode enables the so-called InnoDB strict mode. Normally some errors in the CREATE TABLE options are ignored. When InnoDB strict mode is enabled, the creation of InnoDB tables will fail with an error when certain mistakes are made.

updatable_views_with_limit determines whether view updates can be made with an UPDATE or DELETE statement with a `LIMIT` clause if the view does not contain all primary or not null unique key columns from the underlying table.

Dumps and sys.sql_modules

SQL Server Management Studio allows one to create a working SQL script to recreate a database - something that MariaDB users refer to as a *dump*. Several options allow fine-tuning the generated syntax. It could be necessary to adjust some of these options to make the output compatible with MariaDB. It is possible to export schemas, data or both. One can create a single global file, or one file for each exported object. Normally, producing a single file is more practical.

Alternatively, the sp_helptext() procedure returns information about how to recreate a certain object. Similar information is also present in the sql_modules table (`definition` column), in the `sys` schema. Such information, however, is not a ready-to-use set of SQL statements.

Remember however that MariaDB does not support schemas. An SQL Server schema is approximately a MariaDB database.

To execute a dump, we can pass the file to mariadb, the MariaDB command-line client.

Provided that a dump file contains syntax that is valid in MariaDB, it can be executed in this way:

```
mariadb --show-warnings < dump.sql
```

`--show-warnings` tells MariaDB to output any warnings produced by the statements contained in the dump. Without this option, warnings will not appear on screen. Warnings don't stop the dump execution.

Errors will appear on screen. Errors will stop the dump execution, unless the `--force` option (or just `-f`) is specified.

For other `mariadb` options, see mariadb Command-line Client Options.

Another way to achieve the same purpose is to start the `mariadb` client in interactive mode first, and then run the `source` command. For example:

```
root@d5a54a082d1b:/# mariadb -uroot -psecret
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 22
Server version: 10.4.7-MariaDB-1:10.4.7+maria~bionic mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> \W
Show warnings enabled.
MariaDB [(none)]> source dump.sql
```

In this case, to show warnings we used the `\W` command, where "w" is uppercase. To hide warnings (which is the default), we can use `\w` (lowercase).

For other `mariadb` commands, see mariadb Commands.

CSV Data

If the table structures are already in MariaDB, we need only to import table data. While this can still be done as explained above, it may be more practical to export CSV files from SQL Server and import them into MariaDB.

SQL Server Management Studio and several other Microsoft tools allow one to export CSV files.

MariaDB allows importing CSV files with the LOAD DATA INFILE statement, which is essentially the MariaDB equivalent of `BULK INSERT`.

It can happen that we don't want to import the whole data, but some filtered or transformed version of it. In that case, we may prefer to use the CONNECT storage engine to access CSV files and query them. The results of a query can be inserted into a table using INSERT SELECT.

Moving Data from MariaDB to SQL Server

There are several ways to move data from MariaDB to SQL Server:

- If the tables don't exist at all in SQL Server, we need to generate a dump first. The dump can include data or not.
- If the tables are already in SQL Server, we can use CSV files instead of dumps to move the rows. CSV files are the most concise format to move data between different technologies.
- With the tables already in SQL Server, another way to move data is to insert the rows into CONNECT tables that "point" to remote SQL Server tables.

Using a Dump (Structure)

mariadb-dump can be used to generate dumps of all databases, a specified database, or a set of tables. It is even possible to only dump a set of rows by specifying the `WHERE` clause.

By specifying the `--no-data` option we can dump the table structures without data.

`--compatible=mssql` will produce an output that should be usable in SQL Server.

Using a Dump (Data)

mariadb-dump by default produces an output with both data and structure.

`--no-create-info` can be used to skip the CREATE TABLE statements.

`--compatible=mssql` will produce an output that should be usable in SQL Server.

`--single-transaction` should be specified to select the source data in a single transaction, so that a consistent dump is produced.

`--quick` speeds up the dump process when dumping big tables.

Using a CSV File

CSV files can also be used to export data to SQL Server. There are several ways to produce CSV files from MariaDB:

- The SELECT INTO OUTFILE statement.
- The CONNECT storage engine, with the CSV table type.
- The CSV storage engine (note that it doesn't support `NULL` and indexes).

Using CONNECT Tables

The CONNECT storage engine allows one to access external data, in many forms:

- Data files (CSV, JSON, XML, HTML and more).
- Remote databases, using the ODBC or JDBC standards, or MariaDB/MySQL native protocol.
- Some special data sources.

`CONNECT` was mentioned previously because it could allow one to read a CSV file and query it in SQL, filtering and transforming the data that we want to move into regular MariaDB tables.

However, `CONNECT` can also access remote SQL Server tables. We can read data from it, or even write data.

To enable `CONNECT` to work with SQL Server, we need to fulfill these requirements:

- Install the ODBC driver, downloadable form Microsoft website. The driver is also available for Linux and MacOS.
- Install unixODBC.
- Install CONNECT (unless it is already installed).

Here is an example of a `CONNECT` table that points to a SQL Server table:

```
CREATE TABLE city (
    id INT PRIMARY KEY,
    city_name VARCHAR(100),
    province_id INT NOT NULL
)
    ENGINE=CONNECT,
    TABLE_TYPE=ODBC,
    TABNAME='city'
    CONNECTION='Driver=SQL Server Native Client 13.0;Server=sql-server-
hostname;Database=world;UID=mariadb_connect;PWD=secret';
```

The key points here are:

- `ENGINE=CONNECT` tells MariaDB that we want to create a `CONNECT` table.
- `TABLE_TYPE` must be 'ODBC', so `CONNECT` knows what type of data source it has to use.
- `CONNECTION` is the connection string to use, including server address, username and password.
- `TABNAME` tells `CONNECT` what the remote table is called. The local name could be different.

`CONNECT` is able to query SQL Server to find out the remote table structure. We can use this feature to avoid specifying the column names and types:

```
CREATE TABLE city
    ENGINE=CONNECT,
    TABLE_TYPE=ODBC,
    TABNAME='city'
    CONNECTION='Driver=SQL Server Native Client 13.0;Server=sql-server-
hostname;Database=world;UID=mariadb_connect;PWD=secret';
```

However, we may prefer to manually specify the MariaDB types, sizes and character sets to use.

Linked Server

Instead of using MariaDB `CONNECT`, it is possible to use SQL Server Linked Server functionality. This will allow one to read data from a remote MariaDB database and copy it into local SQL Server tables. However, note that `CONNECT` allows more control on types and character sets mapping.

Refer to Linked Servers section in Microsoft documentation.

This page is licensed: CC BY-SA / Gnu FDL

Repairing MariaDB Tables for SQL Server Users

Repairing tables in MariaDB is not similar to repairing tables in SQL Server.

The first thing to understand is that every MariaDB table is handled by a storage engine. Storage engines are plugins that know how to physically read and write a table, so each storage engine allows one to repair tables in different ways. The default storage engine is InnoDB.

MariaDB provides specific SQL statements to deal with corrupted tables:

- CHECK TABLE checks if a table is corrupted;
- REPAIR TABLE repairs a table if it is corrupted.

As a general rule, there is no reason why a table that is corrupted on a master should also be corrupted on the slaves. Therefore, `REPAIR` is generally used with the `NO_WRITE_TO_BINLOG` option, to avoid replicating it to the slaves.

Partitioned Tables

Partitioned tables are normally split into multiple physical files (one per partition). Even if one of the partitions is corrupted, in most cases other partitions are healthy.

For this reason, `CHECK TABLE` and `REPAIR TABLE` don't work on partitioned tables. Instead, use ALTER TABLE to check or repair a single partition.

For example:

```
ALTER TABLE orders CHECK PARTITION p_2019, p_2020;
ALTER TABLE orders REPAIR PARTITION p_2019, p_2020;
```

Indexes

Indexes can get corrupted. However, as long as data is not corrupted, indexes can always be dropped and rebuilt with ALTER TABLE:

```
ALTER TABLE customer DROP INDEX idx_email;
ALTER TABLE customer ADD INDEX idx_email (email);
```

Checking and Repairing Tables

Here we discuss how to repair tables, depending on the storage engine.

InnoDB

InnoDB follows the "fail fast" philosophy. If table corruption is detected, by default InnoDB deliberately causes MariaDB to crash to avoid corruption propagation, logging an error into the error log. This happens even if the corruption is found with a `CHECK TABLE` statement. This behavior can be changed with the innodb_corrupt_table_action server variable.

To repair an InnoDB table after a crash:

1. Restart MariaDB with the –innodb-force-recovery option set to a low but non-zero value.
2. If MariaDB fails to start, retry with a higher value. Repeat until you succeed.

At this point, you can follow two different procedures, depending if you can use a backup or not. Provided that you have a usable backup, it is often the best option to bring the database up quickly. But if you want to reduce the data loss as much as possible, you prefer to follow the second method.

Restoring a backup:

1. Drop the whole database with DROP DATABASE.
2. Restore a backup of the database. The exact procedure depends on the type of backup.

Recovering existing data:

1. Dump data from the corrupter table, ordered by primary key. MariaDB could crash when it finds damaged data. Repeat the process skipping damaged data.
2. Save somewhere the table structure with SHOW CREATE TABLE.
3. Restart MariaDB.
4. Drop the table with DROP TABLE.
5. Recreate the table and restore the dump.

For more details, see InnoDB Recovery Modes.

Aria and MyISAM

MyISAM is not crash-safe. In case of a MariaDB crash, the changes applied to MyISAM tables but not yet flushed to the disk are lost.

Aria is crash-safe by default, which means that in case of a crash, after repairing any table that is damaged, no changes are lost. However, Aria tables are not crash-safe if created with `TRANSACTIONAL=0` or `ROW_FORMAT` set to `FIXED` or `DYNAMIC`.

System tables use the Aria storage engine and they are crash-safe.

To check if a MyISAM/Aria table is corrupted, we can use CHECK TABLE. To repair a MyISAM/Aria table, one can use REPAIR TABLE. Before running `REPAIR TABLE` against big tables, consider increasing myisam_repair_threads or aria_repair_threads.

MyISAM and Aria tables can also be automatically repaired when corruption is detected. This is particularly useful for Aria, in case corrupted system tables prevent MariaDB from starting. See myisam_recover_options and aria_recover_options. By default Aria runs the quickest repair type. Occasionally, to repair a system table, we may have to start MariaDB in this way:

```
mysqld --aria-recover-options=BACKUP,FORCE
```

It is also possible to stop MariaDB and repair MyISAM tables with myisamchk, and Aria tables with aria_chk. With default values, a repair can be unnecessarily very slow. Before running these tools, be sure to check the Memory and Disk Use With myisamchk page.

Other Storage Engines

Notes on the different storage engines:

- For MyRocks, see MyRocks and CHECK TABLE.
- With ARCHIVE, `REPAIR TABLE` also improves the compression rate.
- For CSV, see Checking and Rpairing CSV Tables.
- Some special storage engines, like MEMORY or BLACKHOLE, do not support any form of check and repair.

This page is licensed: CC BY-SA / Gnu FDL

Setting Up MariaDB for Testing for SQL Server Users

This page contains links and hints to setup MariaDB for testing. The page is designed for SQL Server users, assuming that they are mostly familiar with Windows and they are not familiar with MariaDB.

Choosing a MariaDB Version

As a general rule, for new installations it's better to choose the latest Generally Available (GA) version.

If you need a feature that is only present in a version that is not yet production-ready, and the project will surely not go to production before that version is GA, it could make sense to use a non-GA version. In this case however, keep in mind that you are using a version that is only suitable for testing.

If you need to work with an existing production instance, you should of course use the same version in testing. However, deprecated versions should not be used in production, because they could be exposed to vulnerabilities that will never be fixed. See deprecation policies if you are not sure about the version you are using.

Setting up MariaDB on Windows

There are two different ways to use MariaDB on Windows natively: using Zip packages or MSI packages.

In both cases, 32-bit platforms are still supported.

Check the page Installation issues on Windows to verify if current versions of MariaDB have troubles on Windows. More generally, it is a good idea to check the Troubleshooting Installation Issues category.

ZIP Packages

Windows users don't necessarily need to install MariaDB to use it. They can download ready-to-use ZIP packages to avoid any change in the system (except for downloading MariaDB and writing databases on the disk). This is very useful for testing without risking some undesired side effect on the machine in use. And it avoids the hassle of installing Docker or virtual machines.

It is necessary to run mysql_install_db.exe to install the data directory.

The drawback is that MariaDB will need to be started and stopped from the command line.

See Installing MariaDB Windows ZIP Packages.

MSI Packages

MSI packages provide a friendly graphical interface to install MariaDB. The installation process is easy but flexible. For example, the user can decide which components to install, whether to install it as a service or not, and if networking should be enabled. An interface to uninstall MariaDB is also provided.

See Installing MariaDB MSI Packages on Windows.

Installing MariaDB on Docker

Docker is a container platform that runs natively on Linux. A Docker image is a representation of a basic Linux system, which usually runs a single process - in our case, that process is MariaDB. A container is an instance of an image, which can be created or destroyed instantaneously. Once a container is started, it can be used just like a normal system.

Docker runs on all major operating systems. On Windows and MacOS it runs on a Linux virtual machine, but this additional complexity is transparent for the end user.

Docker's characteristics makes it optimal to test MariaDB functionalities without wasting time on installation and without making changes to the host system. However, it is not ideal to test MariaDB performance.

See Installing and Using MariaDB via Docker.

Reinitializing MariaDB Data Directory

While experimenting with MariaDB, you could end up with an unusable installation. This occurs for example if you deliberately delete files that you shouldn't delete. If it happens, there is no need to uninstall and reinstall MariaDB.

Instead, you can simply delete the contents of the data directory and run mariadb-install-db. The program will recreate your system tables and the essential files.

To know where your data directory is, check the datadir system variable.

This page is licensed: CC BY-SA / Gnu FDL

Migrating From MariaDB RDS to MariaDB Cloud

This guide covers migrating from a self-managed or AWS MariaDB RDS instance to MariaDB Cloud using AWS Database Migration Service (DMS), utilizing binary log streaming for Change Data Capture (CDC)[19] to enable a near-zero downtime cutover.

Phase 1: Preparing Your Source and Target Environments

DMS reads changes from MariaDB using binary log replication. These settings must be in place before creating DMS endpoints or starting any task.

Step 1

Source AWS MariaDB RDS Configuration

- Enable automated backups: Set *Backup Retention Period* to at least 1 day.
- Create a custom parameter group: Apply the following parameters to ensure DMS can read row-level changes:
- `binlog_format = ROW`
- `binlog_row_image = FULL`
- `binlog_checksum = NONE`

Option groups
default:mariadb-11-8 In sync

Amazon Resource Name (ARN)
arn:aws:rds:ap-south-1:221033228277:db:
database-1

Resource ID
db-Z6IHGACSNVD5Z63SP4IWC5JOMI

Created time
April 20, 2026, 10:31 (UTC+05:30)

DB instance parameter group
mariadb118custom In sync

- Apply and reboot: Associate the custom parameter group with your RDS instance and **reboot** to finalize the `ROW` mode.
- Set log retention: Provide a buffer for DMS to read the continuous logs. Run this command: `CALL mysql.rds_set_configuration('binlog retention hours', 24);`
- Verify that all of the following variables are set, by running the following statements:

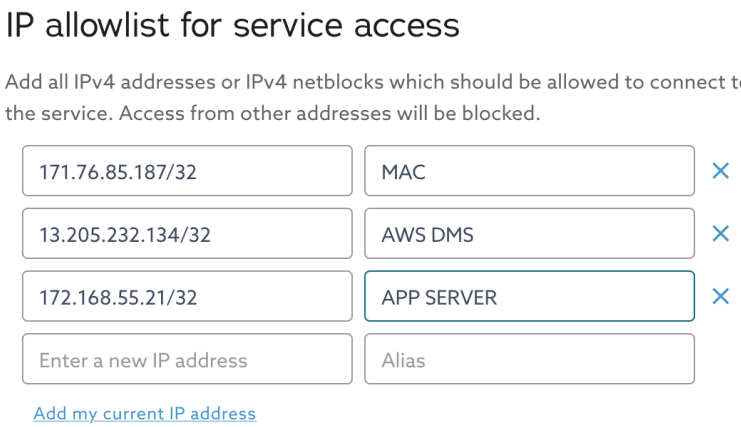
```
MariaDB [(none)]> SHOW VARIABLES LIKE 'net%';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| net_buffer_length | 16384 |
| net_read_timeout  | 30    |
| net_retry_count   | 10    |
| net_write_timeout | 60    |
+-------------------+-------+

MariaDB [(none)]> SHOW VARIABLES LIKE 'binlog_format';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | ROW   |
+---------------+-------+
```

Step 2

MariaDB Cloud Target Configuration

- Create database: Log into MariaDB Cloud and run `CREATE DATABASE db_name;`.
- Create user (optional): Use the default user to create additional users with privileges (for instance, users in RDS, `dms_admin` etc.), setting the Allowed Host to `%` or to the specific IP of your DMS instance.
- Verify settings: Issue the same `SHOW VARIABLES` statements as in the previous step.
- Enable connectivity: Open the MariaDB Cloud allowlist during setup, narrow it to the *NAT Gateway EIP* (or *DMS Public IP*) before go-live.

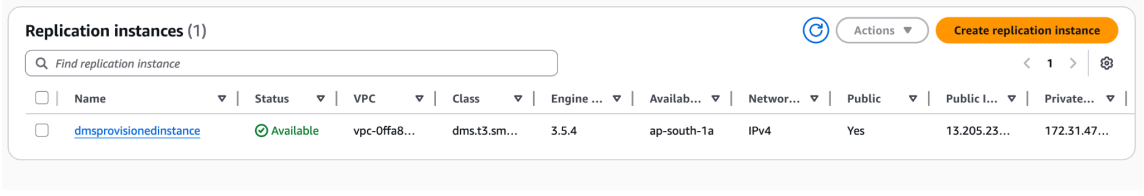


> **Note:**
>
> Note on Serverless: Unlike Provisioned instances, MariaDB Serverless creates temporary Elastic Network Interfaces (ENIs) in your subnets. These ENIs lack default Public IPs and cannot reach the internet via an Internet Gateway. A NAT Gateway in your VPC is required.

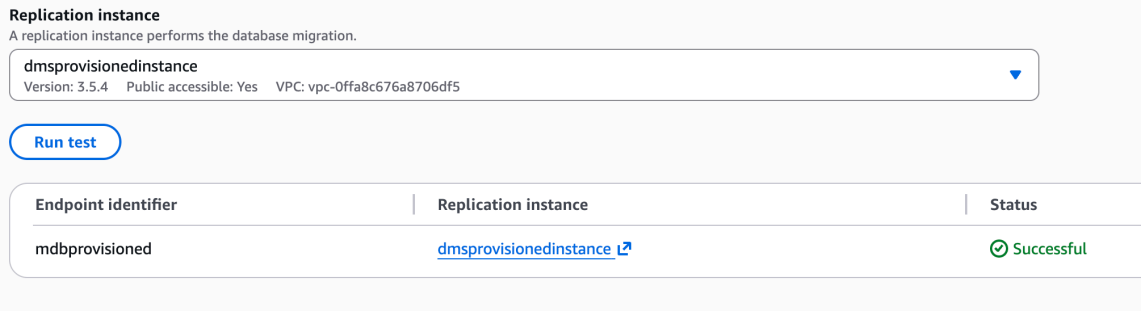
Phase 2: Provisioning the AWS DMS Networking and Compute

The replication instance is the compute resource that handles the data migration.

- Instance type: Using a *Provisioned* instance is highly recommended for static IP stability.
- VPC architecture: Deploy resources within private subnets for enhanced security, utilizing a NAT Gateway to establish the required outbound internet connection to MariaDB Cloud.
- Public accessibility: Select *Publicly Accessible: Yes* if your traffic is routing over the internet. Note the allocated *Public IP* and add it to your MariaDB Cloud allowlist if you are not using an open `0.0.0.0/0` rule.



- Test Connections: Ensure both endpoints return a *Successful* status.



> **Note:**
>
> Serverless Limits: Similar to the target database, DMS Serverless relies on ENIs without Public IPs and inherently struggles to reach MariaDB Cloud over the internet without a properly configured NAT Gateway.

Phase 3: Migrating Schema Definitions and Secondary Objects

AWS DMS only migrates table data and basic primary keys. It does not migrate secondary objects such as views, stored procedures, or triggers.

Export metadata from RDS source: Use `mariadb-dump` from your terminal to grab all schema objects except the table data. Run this command:

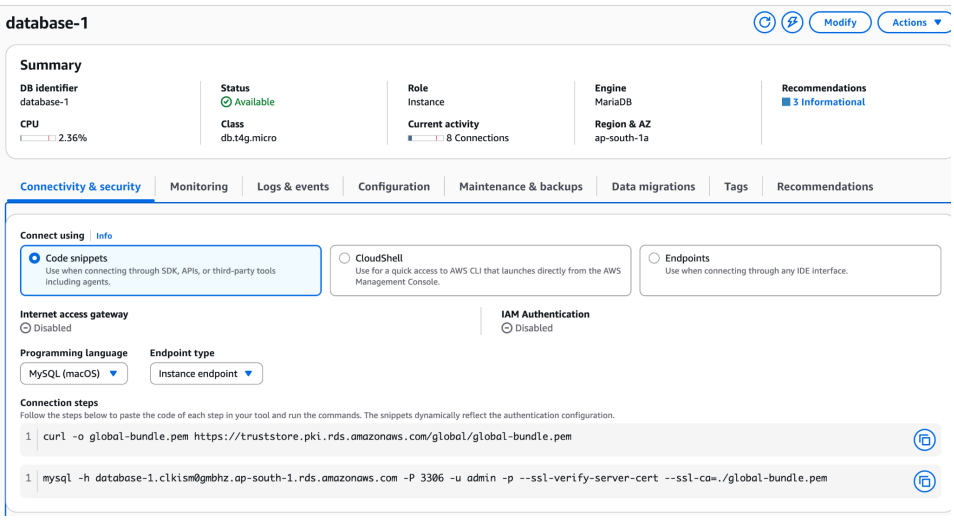
```
mariadb-dump
  -h [RDS_HOST] -u [USER] -p
  --no-data --no-create-info --routines --triggers
  --databases db_name | sed -e 's/DEFINER=`[^`]*`@`[^`]*`//g'  > metadata.sql
```

Phase 4: Configuring Migration Endpoints and Sync Tasks

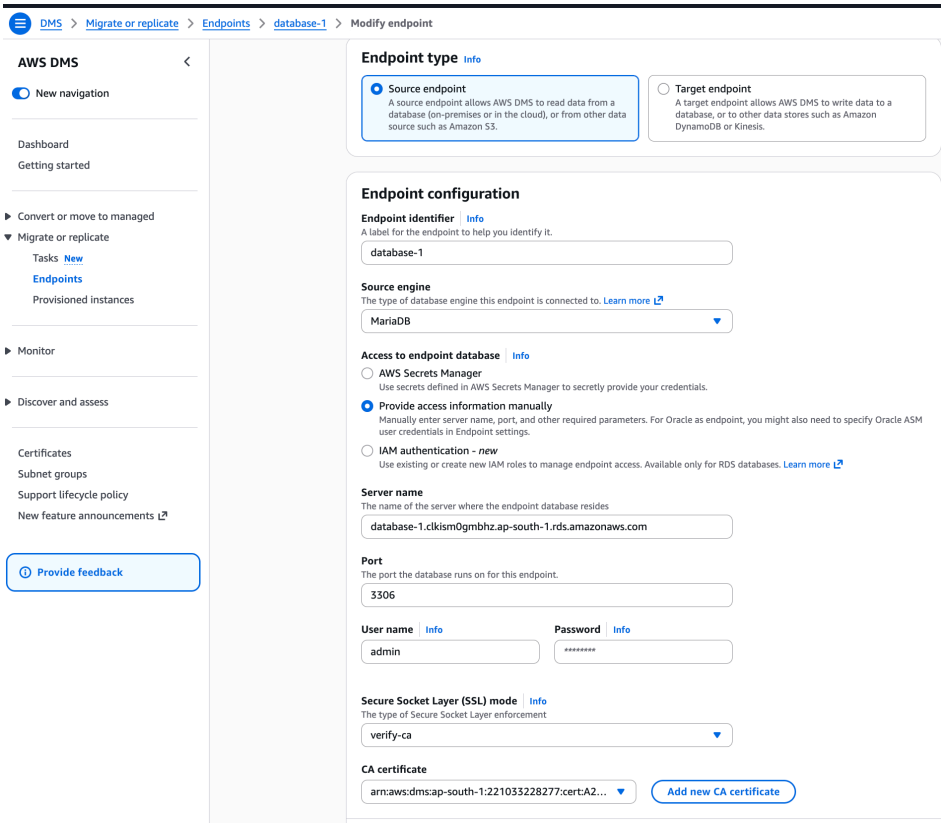
Before creating endpoints, establish secure CA certificate trust between AWS DMS and your database servers.

Source Endpoint (RDS)

1. Download the Global Bundle or Regional CA from the AWS RDS connection information section.

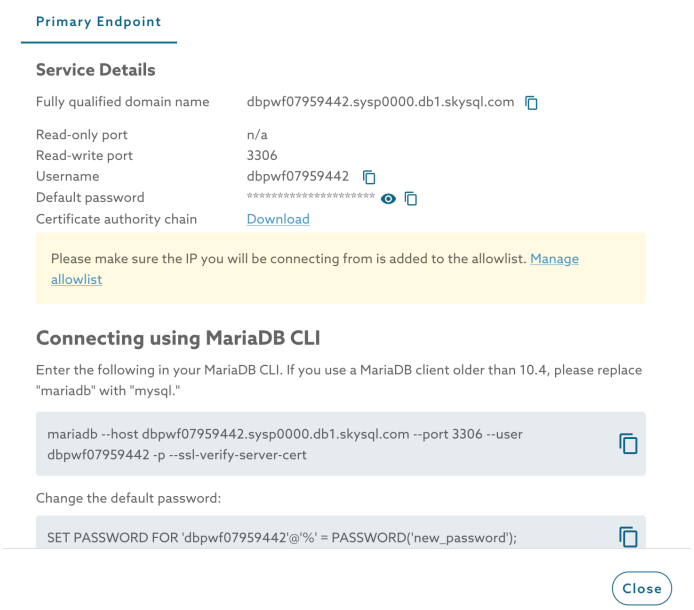


2. In the DMS Console, go to *Certificates* > *Import Certificate*. Upload the `global-bundle.pem` file and rename it to `rds-source-ca`.
3. In the *Source Endpoint* settings, select SSL mode: `verify-ca` and choose the `rds-source-ca` certificate. Point to your RDS endpoint using the MariaDB user.

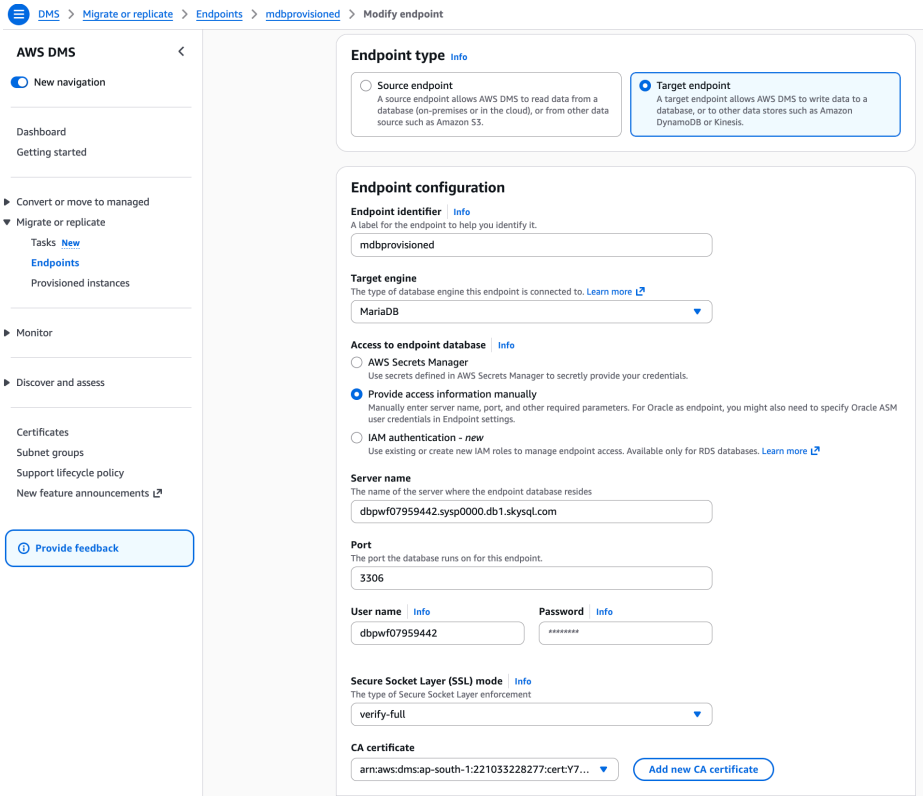


Target Endpoint (MariaDB Cloud)

1. Download the MariaDB Cloud CA Certificate from your MariaDB Cloud portal (found in the Security or Connect tab).



2. Import this into the DMS Console as MariaDB Cloud-target-ca.
3. Create the Target Endpoint, select SSL mode: `verify-full`, and point it to the MariaDB Cloud endpoint on Port `3306`.

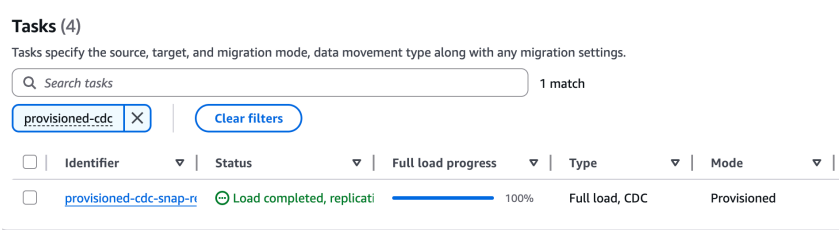


Create the DMS Migration Task

- Migration type: Select Migrate existing data and replicate ongoing changes (enables Snapshot + Replica mode).
- Target table preparation mode: Choose *Drop tables on target* if you want DMS to build the structures, or *Do nothing* if you precreated the tables.
- Stop task after full load completes: Select *Don't stop* so it transitions into CDC mode automatically.
- Include LOB columns in replication: Select *Limited LOB* mode.
- Table Mappings: Add a selection rule with *Schema <your Schema Name>*, *Table name* `%`, and *Action Include*.
- Transformation Rules: Add rules here if you need to rename schemas (for instance, rename `sakila` to `sakila_prod`).
- *Premigration Assessment*: Enable this to let AWS check for potential issues (unsupported data types, missing primary keys) before launching.

Phase 5: Live Replication, Target Cutover, and Decommissioning

1. Executing the *Snapshot + Replica:* Select *Start on create* to initiate the *Full Load*. Watch the *Full Load* progress bar until it hits 100%. The task status will transition to *Load complete, replication ongoing*. From this point forward, any `INSERT` or `UPDATE` statement on the RDS source is replicated to MariaDB Cloud instantaneously.



2. Post-table migration (views, triggers, and procedures): AWS DMS does not migrate secondary objects. Once DMS shows the load is complete, apply the metadata you exported in Part 3 to your MariaDB Cloud Instance:

```
mariadb -h [MariaDB Cloud_HOST] -P 4000 -u [USER] -p db_name < metadata.sql
```

> **Note:**
>
> Important CDC considerations for triggers: If you have triggers on the Target (MariaDB Cloud) that update other tables, DMS might accidentally trigger them again while replicating CDC events, causing duplicate data. It is best practice to keep triggers disabled on the target until the final cutover.

3. Cutover: Identify a low-traffic window, stop the application from writing to the RDS source, and wait for the CDC lag to hit zero. Update your connection strings to point to MariaDB Cloud, re-enable your triggers, and finally terminate the DMS task and infrastructure.

Phase 6 (Optional): Reverse Replication

Switch master and slave to make MariaDB Cloud the master.

Establishe MariaDB SkySQL as the primary (master) and AWS RDS as the replica (slave).

1. Prepare MariaDB Cloud (new master)

Before RDS can follow MariaDB Cloud, the target environment must be configured to allow inbound replication.

- Add the AWS RDS *Public IP* (or *NAT Gateway EIP*) to the SkySQL Allowlist (Port `3306`).
- Identity: Create a dedicated replication user on MariaDB Cloud (SkySQL):

```
CREATE USER 'repl_user'@'%' IDENTIFIED BY 'YourSecurePassword';
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl_user'@'%';
```

- Checkpoint: Capture the current binary log coordinates to tell RDS where to start:

```
SHOW MASTER STATUS; -- note the file and position
```

2. Configure AWS RDS (new slave): Because RDS is a managed service, standard `CHANGE MASTER` statements are restricted. You must use AWS stored procedures.

- Initialize: Clear old migration metadata and point RDS to the SkySQL endpoint using the file and position from step 1:

```
CALL mysql.rds_set_external_master ('skysql-hostname.net', 4000, 'repl_user', 'YourSecurePassword',
'[File_from_SkySQL]', [Position_from_SkySQL], 1 );
```

- Activate: Start the replication threads:

```
CALL mysql.rds_start_replication;
```

3. Monitoring & verification: The “source of truth” for the health of this link moves from the DMS Console to the database command line.

- Status check on RDS: Run `CALL mysql.rds_replica_status\G;` to verify connectivity.
- Success criteria: Ensure `Slave_IO_Running` is `Yes`, `Slave_SQL_Running` is `Yes`, and `Seconds_Behind_Master` is `0`.
- Status check on MariaDB Cloud: Run `SHOW MASTER STATUS\G` and `SHOW SLAVE STATUS\G` (this should return an empty set).

Decommissioning: Once `Seconds_Behind_Master` remains at `0`, the AWS DMS infrastructure is redundant. To eliminate unnecessary costs:

1. Stop and Delete the AWS DMS Task.
2. Terminate the DMS Replication Instance.
3. Delete the NAT Gateway (if no longer required for outbound RDS traffic).

> **Note:**
>
> The DMS Task Console will now show *Zero Activity* and may report *High Source Latency*. This is expected, as the data flow has successfully moved from the DMS pipe to native MariaDB replication.

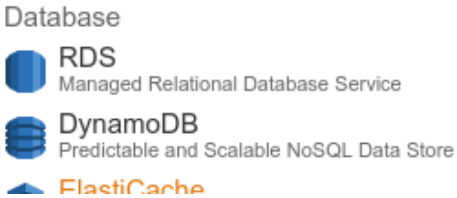
This page is licensed: CC BY-SA / Gnu FDL

MariaDB on Amazon RDS

MariaDB is available on Amazon AWS, as one of the database options when using Amazon's RDS service, or using a MariaDB AMI on Amazon EC2 from the AWS Marketplace.

Amazon RDS

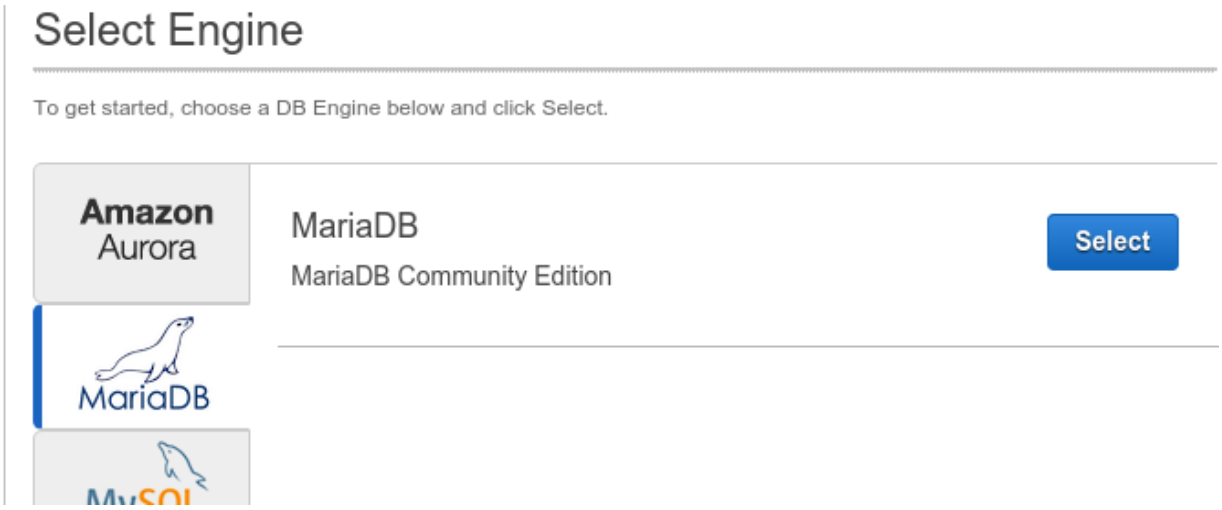
To get started with MariaDB on Amazon's RDS service, click on the RDS link in the Database section of the AWS console.



rds-link

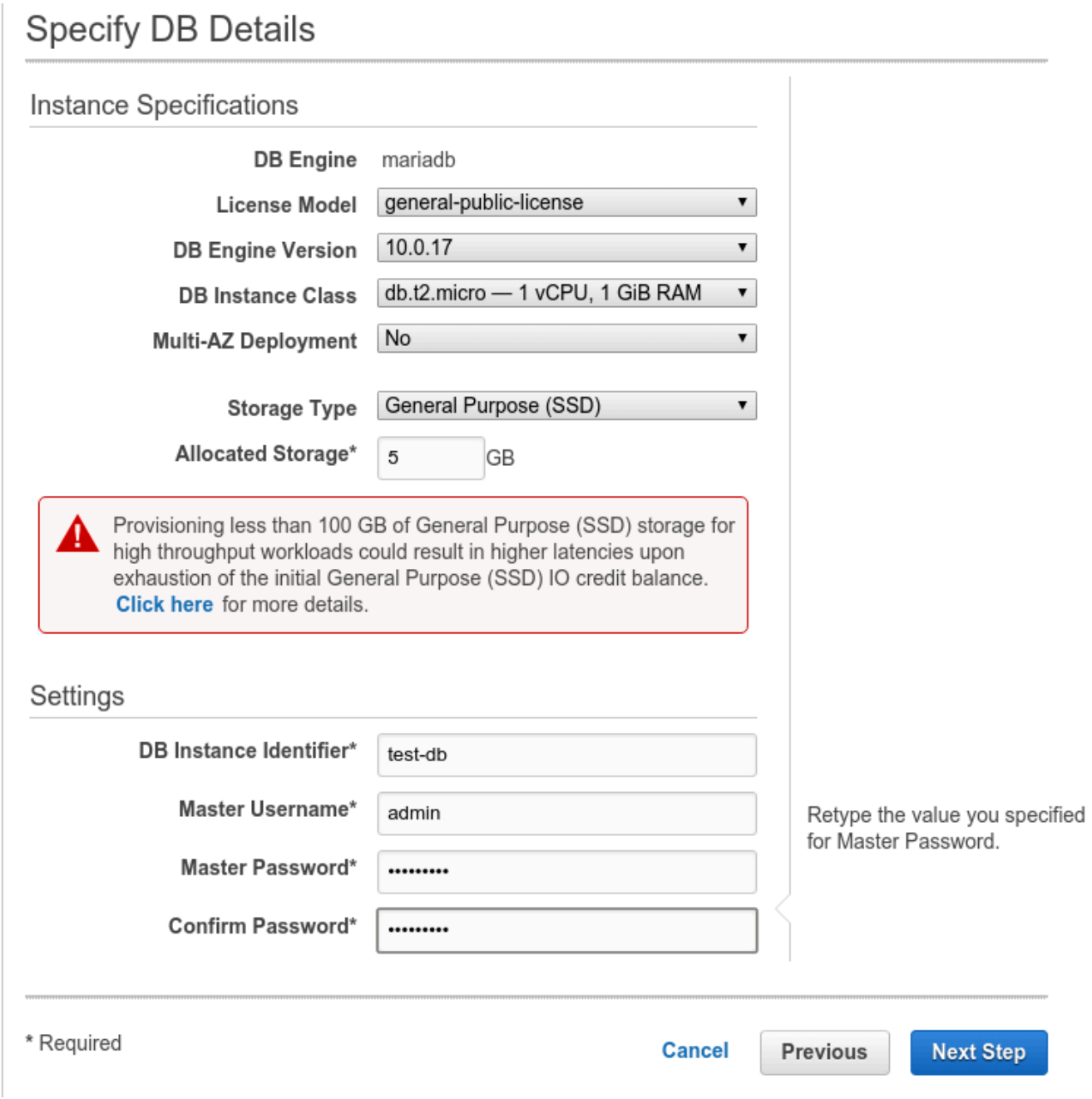
Next, click on the **Get Started Now** button. Alternatively, you can click on the **Launch DB Instance** button from the Instances section of the RDS Dashboard.

In either case, you will be brought to the page where you can select the database engine you want to use. Click on the **MariaDB** logo and then click on the **Select** button.



You will then move to step 2 where you choose whether or not you want to use your MariaDB instance for production or non-production usage. Amazon has links on this page to documentation on the various options.

After selecting the choice you want you will move to step 3 where you specify the details for your database, including setting up an admin user in the database.



You will then move to step 4 where you can configure advanced settings, including security settings, various options, backup settings, maintenance defaults, and so on.

Refer to Amazon's RDS documentation for complete documentation on all the various settings and for information on connecting to and using your RDS MariaDB instances.

AMI on EC2

MariaDB AMIs (Amazon Machine Images) are available in the AWS Marketplace. These AMIs, kept up-to-date with the most recently released versions of MariaDB, are a great way to try out the newest MariaDB versions before they make it to RDS and/or to use MariaDB in a more traditional server environment.

This page is licensed: CC BY-SA / Gnu FDL

Automated Deployment & Administration

See Why to Automate MariaDB Deployments and Management.

Explains the benefits of automating database deployment and management, such as consistency, scalability, and the adoption of Infrastructure as Code principles.

See Comparison Between Automation Systems.

Compares different automation tools like Ansible and Puppet, highlighting differences in their architecture (agentless vs. agent-based) and code structure to help users choose the right tool.

See Automating MariaDB Tasks with Events.

Describes how to use the MariaDB Event Scheduler to automate recurring SQL tasks directly within the database server, similar to cron jobs.

See Automating Upgrades with MariaDB.Org Downloads REST API.

A guide on using the MariaDB Downloads REST API to programmatically retrieve information about software versions, facilitating automated upgrade scripts.

See HashiCorp Vault and MariaDB.

Discusses the integration of HashiCorp Vault with MariaDB for managing secrets, including using MariaDB as a storage backend for Vault.

See Orchestrator Overview.

An introduction to Orchestrator, a tool for managing MySQL and MariaDB replication topologies, providing high availability and automated failover capabilities.

See Ansible and MariaDB.

Provides general information and resources for using Ansible to automate the deployment and configuration of MariaDB servers using playbooks.

See Puppet and MariaDB.

General information and hints on how to automate MariaDB deployments and configuration with Puppet, an open source tool for deployment, configuration, and operations.

See MariaDB Containers.

Discusses running MariaDB in Docker containers, covering the benefits of isolation and ease of deployment for development and testing environments.

See Kubernetes and MariaDB.

General information and hints on deploying MariaDB Kubernetes (K8s) containers, an open source container orchestration system which automates deployments, horizontal scaling, configuration, and operat

See Vagrant and MariaDB.

Vagrant is an open source tool to quickly setup machines that can be used for development and testing. They can be local virtual machines, Docker containers, AWS EC2 instances, and so on

This page is licensed: CC BY-SA / Gnu FDL

Starting & Stopping

See Starting and Stopping Overview.

Complete Starting and Stopping Overview guide for MariaDB. Complete reference documentation for implementation, configuration, and usage for production use.

See launchd (macOS).

Instructions for configuring MariaDB to start automatically on macOS using a launchd plist file in /Library/LaunchDaemons.

See systemd (Linux).

MariaDB on systemd (Linux): overview, supported distributions, and links to operational and configuration guidance.

See sysVinit.

Describes how to manage MariaDB using SysVinit scripts (mysql.server), common on older Linux distributions, using commands like `service` and `chkconfig`.

See mariadbd-safe.

Details the `mariadbd-safe` wrapper script, which adds safety features like auto-restart upon crash and error logging to syslog.

See mariadbd-multi.

Explains how to use `mariadbd-multi` to manage multiple MariaDB server processes on a single host using GNR groups in the configuration file.

See mariadbd Options.

A reference list of command-line options available for the `mariadbd` server binary, covering configuration, replication, and service installation.

See mariadbd.

Describes the `mariadbd` binary (formerly `mysqld`), which is the core database server executable.

See mysql.server.

Documentation for the `mysql.server` script, a SysVinit-style wrapper used to start and stop `mariadbd-safe`.

See Running MariaDB From the Build Directory.

Instructions for developers on how to run MariaDB directly from the source build directory without installing it to system paths.

See Running Multiple MariaDB Server Processes.

A guide on configuring and running multiple independent MariaDB instances on the same machine by isolating data directories, ports, and sockets.

See Specifying Permissions for Schema (Data) Directories and Tables.

Explains default file permissions for data directories and how to customize them using `UMASK` and `UMASK_DIR` environment variables.

See Switching Between Different Installed MariaDB Versions.

Techniques for managing parallel installations of different MariaDB versions, typically using symlinks and separate data directories for testing.

See What to Do if MariaDB Doesn't Start.

Complete What to Do if MariaDB Doesn't Start guide for MariaDB. Complete reference documentation for implementation, configuration, and usage.

This page is licensed: CC BY-SA / Gnu FDL

Starting and Stopping Overview

There are several different methods to start or stop the MariaDB Server process. There are two primary categories that most of these methods fall into: starting the process with the help of a service manager, and starting the process manually.

Service Managers

sysVinit and systemd are the most common Linux service managers. launchd is used in MacOS X. Upstart is a less common service manager.

Systemd

RHEL/CentOS 7 and above, Debian 8 Jessie and above, and Ubuntu 15.04 and above use systemd by default.

For information on how to start and stop MariaDB with this service manager, see systemd: Interacting with the MariaDB Server Process.

SysVinit

RHEL/CentOS 6 and below, and Debian 7 Wheezy and below use sysVinit by default.

For information on how to start and stop MariaDB with this service manager, see sysVinit: Interacting with the MariaDB Server Process.

launchd

launchd is used in MacOS X.

Upstart

Ubuntu 14.10 and below use Upstart by default.

Starting the Server Process Manually

mariadbd

mariadbd is the actual MariaDB Server binary. It can be started manually on its own.

mariadbd-safe

mariadbd-safe is a wrapper that can be used to start the mariadbd server process. The script has some built-in safeguards, such as automatically restarting the server process if it dies. See mariadbd-safe for more information.

mariadbd-multi

mariadbd-multi is a wrapper that can be used to start the mariadbd server process if you plan to run multiple server processes on the same host. See mariadbd-multi for more information.

mysql.server

mysql.server is a wrapper that works as a standard sysVinit script. However, it can be used independently of sysVinit as a regular `sh` script. The script starts the mariadbd server process by first changing its current working directory to the MariaDB install directory and then starting mariadbd-safe. The script requires the standard sysVinit arguments, such as `start`, `stop`, and `status`. See mysql.server for more information.

This page is licensed: CC BY-SA / Gnu FDL

launchd (macOS)

If you install MariaDB on macOS using Homebrew – which is the only option – starting and stopping `mariadbd` (MariaDB server) is done as described on this page. The below instructions don't need to be performed.

On macOS, create a file called `/Library/LaunchDaemons/com.mariadb.server.plist` with the following contents (edit to suit):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-
1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key> <string>com.mariadb.server</string>
  <key>KeepAlive</key><true/>
  <key>RunAtLoad</key><true/>
  <key>LaunchOnlyOnce</key><false/>
  <key>ExitTimeOut</key><integer>600</integer>
  <key>WorkingDirectory</key><string>/usr/local/var</string>
  <key>Program</key><string>/usr/local/bin/mysqld</string>
  <key>ProgramArguments</key>
  <array>
    <string>/usr/local/bin/mysqld</string>
    <string>--user=_mysql</string>
    <string>--basedir=/usr/local/opt/mariadb</string>
    <string>--plugin-dir=/usr/local/opt/mariadb/lib/plugin</string>
    <string>--datadir=/usr/local/var/mysql</string>
    <string>--log-error=/usr/local/var/mysql/Data-Server.local.err</string>
    <string>--pid-file=/usr/local/var/mysql/Data-Server.local.pid</string>
    <string>--sql-mode=ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION</string>
  </array>
</dict>
</plist>
```

Then from a shell, run `launchctl load /Library/LaunchDaemons/com.mariadb.server.plist` and MariaDB will run immediately, and also upon reboot.

See Also

- Creating Launch Daemons and Agents
- A launchd Tutorial

This page is licensed: CC BY-SA / Gnu FDL

systemd (Linux)

`systemd` is a sysVinit replacement that is the default service manager on the following Linux distributions:

- RHEL 7 and above
- CentOS 7 and above
- Fedora 15 and above
- Debian 8 and above
- Ubuntu 15.04 and above
- SLES 12 and above
- OpenSUSE 12.2 and above

MariaDB's `systemd` unit file is included in the server packages for RPMs and DEBs. It is also included in certain binary tarballs.

The service name is `mariadb.service`.

In this Section

See Starting MariaDB on systemd.

Installing, starting, stopping, restarting, and inspecting the MariaDB service, including multi-instance setups, Galera Cluster integration, and the systemd journal.

See Configuring MariaDB for systemd.

Drop-in configuration files, timeouts, open-file and core-size limits, `LimitMEMLOCK` (`io_uring` and `aio`), error-log redirection, home-directory access, umask, data directory, socket activation, and converting `mariadbd-safe` options to `systemd` options. Also includes a quick-setup script for developers running MariaDB from a build directory.

Contents of the MariaDB Service's Unit File

The contents of the `mariadb.service` file can be examined with `systemctl show mariadb.service`.

This page is licensed: CC BY-SA / Gnu FDL

Starting MariaDB on systemd

This page covers the operational side of running MariaDB under systemd. For drop-in configuration, limits, socket activation, and `mariadbd-safe`-to-systemd conversion, see Configuring MariaDB for systemd.

Installing & Starting MariaDB

When installing using the MariaDB server RPM or DEB package, it automatically runs the mariadb-install-db script which creates the initial databases and users.

When MariaDB is started with the `systemd` unit file, it directly starts the mariadbd process as the `mysql` user. Unlike with sysVinit, the mariadbd process is not started with mariadbd-safe. As a consequence, options are not read from the `[mariadbd-safe]` option group from option files.

Interacting with the MariaDB Server Process

The service can be interacted with by using the systemctl command.

Starting the MariaDB Server Process on Boot

MariaDB's `systemd` service can be configured to start at boot by executing the following:

```
sudo systemctl enable mariadb.service
```

Starting the MariaDB Server Process

MariaDB's `systemd` service can be started by executing the following:

```
sudo systemctl start mariadb.service
```

MariaDB's `systemd` unit file has a default startup timeout of about 90 seconds on most systems. If certain startup tasks, such as crash recovery, take longer than this default startup timeout, then `systemd` assumes that `mariadbd` has failed to startup, which causes `systemd` to kill the `mariadbd` process. To work around this, you can reconfigure the MariaDB `systemd` unit to have an infinite timeout.

Note that systemd 236 added the EXTEND_TIMEOUT_USEC environment variable that allows services to extend the startup timeout during long-running processes. On systems with systemd versions that support it, MariaDB uses this feature to extend the startup timeout during certain startup processes that can run long. Therefore, if you are using `systemd` 236 or later, then you should not need to manually override `TimeoutStartSec`, even if your startup tasks, such as crash recovery, run for longer than the configured value. See MDEV-14705 for more information.

Stopping the MariaDB Server Process

MariaDB's `systemd` service can be stopped by executing the following:

```
sudo systemctl stop mariadb.service
```

Restarting the MariaDB Server Process

MariaDB's `systemd` service can be restarted by executing the following:

```
sudo systemctl restart mariadb.service
```

Checking the Status of the MariaDB Server Process

The status of MariaDB's `systemd` service can be obtained by executing the following:

```
sudo systemctl status mariadb.service
```

Interacting with Multiple MariaDB Server Processes

On some operating systems, a `systemd` template unit file called `mariadb@.service` is installed in `INSTALL_SYSTEMD_UNITDIR`. See Contents of the MariaDB Service's Unit File for how to inspect the unit file on your system.

This template unit file allows you to interact with multiple MariaDB instances on the same system using the same template unit file. When you interact with a MariaDB instance using this template unit file, you have to provide an instance name as a suffix. For example, the following command tries to start a MariaDB instance with the name `node1`:

```
sudo systemctl start mariadb@node1.service
```

MariaDB's build system cannot include the `mariadb@.service` template unit file in RPM packages on platforms that have cmake versions older than 3.3.0, because these `cmake` versions have a bug that causes it to encounter errors when packaging a file in RPMs if the file name contains the `@` character. To use this functionality on a MariaDB version that does not have the file, you can copy the file from a package that contains the file.

Default Configuration of Multiple Instances

`systemd` also looks for an option file for a specific MariaDB instance based on the instance name.

It uses the `.%I` as the custom option group suffix that is appended to any server option group, in any configuration file included by default.

In all distributions, the `%I` is the MariaDB instance name. In the above `node1` case, it would use the option file at the path `/etc/mynode1.cnf`.

When using multiple instances, each instance also needs their own datadir, socket, and port (unless `skip_networking` is specified). Because mariadb-install-db reads the same sections as the server, and `ExecStartPre=run mariadb-install-db` within the service, the instances are automatically created if there are sufficient privileges.

Custom Configuration of Multiple Instances

Because users may want to do many various things with their multiple instances, we've provided a way to let the user define how they wish their multiple instances to run. The systemd environment variable `MYSQLD_MULTI_INSTANCE` can be set to anything that mariadbd and mariadb-install-db recognize.

A hosting environment where each user has their own instance looks like this (with `sudo systemctl edit mariadb@.service`):

```
[Service]
ProtectHome=false
Environment='MYSQLD_MULTI_INSTANCE=--defaults-file=/home/%I/my.cnf \
                        --user=%I \
                        --socket=/home/%I.sock \
                        --datadir=/home/%I/mariadb_data \
                        --skip-networking'
```

Here the instance name is the unix user of the service.

Systemd and Galera Cluster

Bootstrapping a New Cluster

When using Galera Cluster with systemd, the first node in a cluster has to be started with `galera_new_cluster`. See Getting Started with MariaDB Galera Cluster: Bootstrapping a New Cluster for more information.

Recovering a Node's Cluster Position

When using Galera Cluster with systemd, a node's position in the cluster can be recovered with `galera_recovery`. See Getting Started with MariaDB Galera Cluster: Determining the Most Advanced Node for more information.

SSTs and Systemd

MariaDB's `systemd` unit file has a default startup timeout of about 90 seconds on most systems. If an SST takes longer than this default startup timeout on a joiner node, then `systemd` assumes that `mariadbd` has failed to startup, which causes `systemd` to kill the `mariadbd` process on the joiner node. To work around this, you can reconfigure the MariaDB `systemd` unit to have an infinite timeout. See Introduction to State Snapshot Transfers (SSTs): SSTs and Systemd for more information.

Note that systemd 236 added the EXTEND_TIMEOUT_USEC environment variable that allows services to extend the startup timeout during long-running processes. On systems with `systemd` versions that support it, MariaDB uses this feature to extend the startup timeout during long SSTs. Therefore, if you are using `systemd` 236 or later, you do not need to manually override `TimeoutStartSec` even if your SSTs run for longer than the configured value. See MDEV-15607 for more information.

Systemd Journal

`systemd` has its own logging system called the `systemd` journal. The `systemd` journal contains information about the service startup process. It is a good place to look when a failure has occurred.

The MariaDB `systemd` service's journal can be queried by using the journalctl command:

```
$ sudo journalctl n 20 -u mariadb.service
-- Logs begin at Fri 2019-01-25 13:49:04 EST, end at Fri 2019-01-25 18:07:02 EST. --
Jan 25 13:49:15 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Starting MariaDB 10.1.37 database
server...
Jan 25 13:49:16 ip-172-30-0-249.us-west-2.compute.internal mysqld[2364]: 2019-01-25 13:49:16
140547528317120 [Note] /usr/sbin/mysqld (mysqld 10.1.37-MariaDB) starting as process 2364 ...
Jan 25 13:49:17 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Started MariaDB 10.1.37 database
server.
Jan 25 18:06:42 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Stopping MariaDB 10.1.37 database
server...
Jan 25 18:06:44 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Stopped MariaDB 10.1.37 database
server.
Jan 25 18:06:57 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Starting MariaDB 10.1.37 database
server...
Jan 25 18:08:32 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: mariadb.service start-pre
operation timed out. Terminating.
Jan 25 18:08:32 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Failed to start MariaDB 10.1.37
database server.
Jan 25 18:08:32 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Unit mariadb.service entered
failed state.
Jan 25 18:08:32 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: mariadb.service failed.
```

This page is licensed: CC BY-SA / Gnu FDL

Configuring MariaDB for systemd

This page covers configuration for MariaDB running under systemd. For starting, stopping, and inspecting the service, see Starting MariaDB on systemd.

> **Note:**
>
> The per-option sections on this page apply to **installed-service deployments** — MariaDB started via `mariadb.service`. If you are a developer running `mariadbd` from a build directory (for example, running `mysql-test-run`), most of those settings don't apply, because your workflow doesn't go through `mariadb.service`. See Configuring systemd for MariaDB Development below for a one-shot setup script that covers the developer case.

Configuring systemd for MariaDB Development

When developing MariaDB, you typically run `mariadbd` directly from a build directory rather than installing it as a service. In that case, the per-service drop-in files described below (under `/etc/systemd/system/mariadb.service.d/`) have no effect, because no `mariadb.service` is involved. Instead, the limits need to be set in `systemd`'s system-wide manager configuration (`/etc/systemd/system.conf.d/`), `sysctl`, and PAM.

The following script applies the settings that are typically needed to run `mariadbd` and `mysql-test-run` (`mtr`) on a developer workstation, including support for `io_uring` asynchronous I/O in InnoDB, a sufficient number of open files, large core dumps, the `aio` limit fix for `mtr`, predictable core file locations, and `perf` / `rr` access. Run it as root:

```
#!/bin/bash
#
# Fixes needed for openSUSE (systemd) to be able to run and develop MariaDB
# without having to install it as a service
# This script needs to be run as root

set -e

# ── systemd limits ──────────────────────────────────────────────────────────

mkdir -p /etc/systemd/system.conf.d

cat > /etc/systemd/system.conf.d/memlock.conf << 'EOF'
# Needed for io_uring asynchronous IO used by InnoDB
[Manager]
DefaultLimitMEMLOCK=infinity
EOF

cat > /etc/systemd/system.conf.d/nofile.conf << 'EOF'
# Databases need a lot of open files
[Manager]
DefaultLimitNOFILE=65535
EOF

cat > /etc/systemd/system.conf.d/bigcore.conf << 'EOF'
# Allow big cores (needed to be able to debug MariaDB core files)
[Manager]
LimitCORE=infinity
EOF

# Apply systemd changes
systemctl daemon-reexec

# ── sysctl ──────────────────────────────────────────────────────────────────

mkdir -p /etc/sysctl.d

cat > /etc/sysctl.d/50-aio.conf << 'EOF'
# Fix "io_setup(1024) returned Unknown error -11 when running mtr tests"
fs.aio-max-nr=1048576
EOF

cat > /etc/sysctl.d/50-coredump.conf << 'EOF'
# Store core files in the current directory
kernel.core_pattern=core.%p
EOF

cat > /etc/sysctl.d/50-perf_event_paranoid.conf << 'EOF'
# Allow usage of perf/RR
kernel.perf_event_paranoid=1
EOF

# Apply sysctl changes
sysctl --system

# ── PAM limits ──────────────────────────────────────────────────────────────

mkdir -p /etc/security/limits.d

cat > /etc/security/limits.d/nofile.conf << 'EOF'
# Takes effect on next login
*    soft    nofile    65535
*    hard    nofile    65535
root soft    nofile    65535
root hard    nofile    65535
EOF

echo ""
```

```
echo "Done. PAM (number of open files) limits (/etc/security/limits.d/nofile.conf) will take effect on
next login."
```

Warning:

The PAM limit change in `/etc/security/limits.d/nofile.conf` only takes effect on your next login. Log out and back in (or open a new shell session) before running `mtr`.

Configuring the Systemd Service

You can configure MariaDB's `systemd` service by creating a "drop-in" configuration file for the `systemd` service. On most systems, the `systemd` service's directory for "drop-in" configuration files is `/etc/systemd/system/mariadb.service.d/`. You can confirm the directory and see what "drop-in" configuration files are currently loaded by executing:

```
$ sudo systemctl status mariadb.service
● mariadb.service - MariaDB 10.1.37 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/mariadb.service.d
           └─migrated-from-my.cnf-settings.conf, timeoutstartsec.conf
...
```

If you want to configure the `systemd` service, you can create a file with the `.conf` extension in that directory. The configuration options you would like to change need to be placed in an appropriate section within the file, usually `[Service]`. If a `systemd` option is a list, you may need to set the option to empty before you set the replacement values:

```
[Service]

ExecStart=
ExecStart=/usr/bin/numactl --interleave=all  /usr/sbin/mariadbd $MYSQLD_OPTS $_WSREP_NEW_CLUSTER
$_WSREP_START_POSITION
```

After any configuration change, you need to execute the following for the change to take effect:

```
sudo systemctl daemon-reload
```

Useful Systemd Options

Useful `systemd` options are listed below. If an option is equivalent to a common mariadbd-safe option, then that is also listed. Use `systemctl edit mariadb.service` to create the systemd option under a `[Service]` section header.

mariadbd-safe option	systemd option	Comments
no option	ProtectHome=false	If any MariaDB files are in `/home/`
no option	PrivateDevices=false	If any MariaDB storage references raw block devices
no option	ProtectSystem=	If any MariaDB write any files to anywhere under `/boot`, `/usr` or `/etc`
no option	TimeoutStartSec={time}	Service startup timeout. See Configuring the Systemd Service Timeout.
no option (see MDEV-9264)	OOMScoreAdjust={priority}	For instance, `-600` to lower priority of OOM killer for `mariadbd`

mariadbd-safe option	systemd option	Comments
open-files-limit	LimitNOFILE={limit}	Limit on number of open files. See Configuring the Open Files Limit.
core-file-size	LimitCORE={size}	Limit on core file size. Useful when enabling core dumps. See Configuring the Core File Size.
	LimitMEMLOCK={size} or infinity	Limit on how much can be locked in memory. Useful when large-pages or memlock is used
nice	Nice={nice value}	
syslog	StandardOutput=syslog	See Configuring MariaDB to Write the Error Log to Syslog.
	StandardError=syslog	
	SyslogFacility=daemon	
	SyslogLevel=err	
syslog-tag	SyslogIdentifier	
flush-caches	ExecStartPre=/usr/bin/sync	
	ExecStartPre=/usr/sbin/sysctl -q -w vm.drop_caches=3	
malloc-lib	Environment=LD_PRELOAD=/path/to/library	
numa-interleave	NUMAPolicy=interleave	from systemd v243 onwards
	or: ExecStart=/usr/bin/numactl –interleave=all /usr/sbin/mariadbd $MYSQLD_OPTS $_WSREP_NEW_CLUSTER $_WSREP_START_POSITION	prepending ExecStart=/usr/bin/numactl –interleave=all to existing ExecStart setting
no-auto-restart	Restart={exit-status}	

> **Note:**
>
> The systemd manual contains the official meanings for these options. The manual also lists considerably more options than the ones listed above.

There are other options and the `mariadb-service-convert` script attempts to convert these as accurately as possible.

Configuring the Systemd Service Timeout

MariaDB's systemd unit file has a default startup timeout of about 90 seconds on most systems. If a service startup takes longer than this default startup timeout, then `systemd` assumes that `mariadbd` has failed to startup, which causes `systemd` to kill the `mariadbd` process. To work around this, it can be changed by configuring the TimeoutStartSec option for the `systemd` service.

A similar problem can happen when stopping the MariaDB service. Therefore, it may also be a good idea to set TimeoutStopSec.

For example, you can reconfigure the MariaDB `systemd` service to have an infinite timeout by executing one of the following commands:

If you are using `systemd` 228 or older, then you can execute the following to set an infinite timeout:

```
sudo systemctl edit mariadb.service

[Service]

TimeoutStartSec=0
TimeoutStopSec=0
```

Systemd 229 added the infinity option, so if you are using `systemd` 229 or later, you can execute the following to set an infinite timeout:

```
sudo systemctl edit mariadb.service

[Service]

TimeoutStartSec=infinity
TimeoutStopSec=infinity
```

Note that systemd 236 added the EXTEND_TIMEOUT_USEC environment variable that allows services to extend the startup timeout during long-running processes. On systems with systemd versions that support it, MariaDB uses this feature to extend the startup timeout during certain startup processes that can run long, like when upgrading MariaDB to a newer main version or recovering after a crash with a very big transaction that needs to roll back.

Configuring the Open Files Limit

When using `systemd`, rather than setting the open files limit by setting the open-files-limit option for `mariadbd-safe` or the open_files_limit system variable, the limit can be changed by configuring the LimitNOFILE option for the MariaDB `systemd` service. The default is set to `LimitNOFILE=16364` in `mariadb.service`.

For example, you can reconfigure the MariaDB `systemd` service to have a larger limit for open files by executing the following command – then restart the server for the changes to take effect:

```
sudo systemctl edit mariadb.service

[Service]
LimitNOFILE=infinity
```

> **Note:**
>
> Setting `LimitNOFILE=infinity` doesn't actually set the open file limit to *infinite*.

In `systemd` 234 and later, setting `LimitNOFILE=infinity` actually sets the open file limit to the value of the kernel's `fs.nr_open` parameter. Therefore, in these `systemd` versions, you may have to change this parameter's value.

The value of the `fs.nr_open` parameter can be changed permanently by setting the value in /etc/sysctl.conf and restarting the server.

The value of the `fs.nr_open` parameter can be changed temporarily by executing the sysctl utility:

```
sudo sysctl -w fs.nr_open=1048576
```

In `systemd` 233 and before, setting `LimitNOFILE=infinity` actually sets the open file limit to `65536`. See systemd issue #6559 for more information. Therefore, in these `systemd` versions, it is not generally recommended to set `LimitNOFILE=infinity`. Instead, it is generally better to set `LimitNOFILE` to a very large integer. For example:

```
sudo systemctl edit mariadb.service

[Service]
LimitNOFILE=1048576
```

Configuring the Core File Size

When using `systemd`, if you would like to enable core dumps, rather than setting the core file size by setting the core-file-size option for `mariadbd-safe`, the limit can be changed by configuring the LimitCORE option for the MariaDB `systemd` service. For example, you can reconfigure the MariaDB `systemd` service to have an infinite size for core files by executing the following commands:

```
sudo systemctl edit mariadb.service

[Service]
LimitCORE=infinity
```

Configuring MariaDB to Write the Error Log to Syslog

When using `systemd`, if you would like to redirect the error log to the syslog, then that can easily be done by doing the following:

- Ensure that log_error system variable is not set.
- Set StandardOutput=syslog.
- Set StandardError=syslog.
- Set SyslogFacility=daemon.
- Set SysLogLevel=err.

For example:

```
sudo systemctl edit mariadb.service

[Service]
StandardOutput=syslog
StandardError=syslog
SyslogFacility=daemon
SysLogLevel=err
```

If you have multiple instances of MariaDB, then you may also want to set SyslogIdentifier with a different tag for each instance.

Configuring LimitMEMLOCK

Configuring Support for io_uring Asynchronous IO in InnoDB

If using `--memlock`, or the `io_uring` `asynchronous` IO in InnoDB in MariaDB 10.6 or above, with a Linux Kernel version < 5.12, you need to raise the `LimitMEMLOCK` limit.

```
sudo systemctl edit mariadb.service

[Service]
LimitMEMLOCK=2M
```

Increasing the Limit for aio

When running the server with lots of users, the default `aio` value is too small. When running `mysql-test-run`, you could see an error like this:

```
io_setup(1024) returned Unknown error -11
```

The fix is to configure a higher aio value:

1. Create this file: `/etc/sysctl.d/50-aio.conf`

2. Add the following to that file:

```
# Fix "io_setup(1024) returned Unknown error -11 when running mtr tests"
fs.aio-max-nr=1048576
```

Configuring Access to Home Directories

MariaDB's systemd unit file restricts access to `/home`, `/root`, and `/run/user` by default. This restriction can be overridden by setting the ProtectHome option to `false` for the MariaDB `systemd` service. This is done by creating a "drop-in" directory `/etc/systemd/system/mariadb.service.d/` and in it a file with a `.conf` suffix that contains the `ProtectHome=false` directive.

You can reconfigure the MariaDB `systemd` service to allow access to `/home` by executing the following commands:

```
sudo systemctl edit mariadb.service

[Service]

ProtectHome=false
```

Configuring the umask

When using `systemd`, the default file permissions of `mariadbd` can be set by setting the `UMASK` and `UMASK_DIR` environment variables for the `systemd` service. For example, you can configure the MariaDB `systemd` service's umask by executing the following commands:

```
sudo systemctl edit mariadb.service

[Service]

Environment="UMASK=0750"
Environment="UMASK_DIR=0750"
```

These environment variables do not set the umask. They set the default file system permissions. See MDEV-23058 for more information.

Keep in mind that configuring the umask this way only affects the permissions of files created by the `mariadbd` process that is managed by `systemd`. The permissions of files created by components that are not managed by `systemd`, such as mariadb-install-db, are not affected.

See Specifying Permissions for Schema (Data) Directories and Tables for more information.

Configuring the Data Directory

When doing a standard binary tarball installation, the *datadir* is under `/usr/local/data`. However, the default `systemd` service file makes the whole `/usr` directory tree write protected.

When just copying the distributed service file a tarball install does not start up, complaining for instance about this:

```
[Warning] Can't create test file /usr/local/.../data/ubuntu-focal.lower-test
[ERROR] mariadbd: File '/usr/local/.../data/aria_log_control' not found (Errcode: 30 "Read-only file
system")
[ERROR] mariadbd: Got error 'Can't open file' when trying to use aria control file
'/usr/local/.../data/aria_log_control'
```

So when using a data directory under `/usr/local` that specific directory needs to be made writable for the service using the `ReadWritePaths` setting:

```
sudo systemctl edit mariadb.service

[Service]
ReadWritePaths=/usr/local/mysql/data
```

Systemd Socket Activation

> **Note:**
>
> Systemd socket activation is available from MariaDB 10.6.

MariaDB can use systemd's socket activation. This is an on-demand service for MariaDB that activates when required.

Systemd socket activation uses a `mariadb.socket` definition file to define a set of UNIX and TCP sockets. Systemd listens on these sockets, and when they are connected to, systemd starts the `mariadb.service` and hand over the socket file descriptors for MariaDB to process the connection.

MariaDB remains running at this point, and has all sockets available and process connections exactly like it did before 10.6.

When MariaDB is shut down, the systemd `mariadb.socket` remains active, and a new connection restarts the `mariadb.service`.

Using Systemd Socket Activation

To use MariaDB systemd socket activation, instead of enabling/starting `mariadb.service`, `mariadb.socket` is used instead.

So the following commands work exactly like the `mariadb.service` equivalents.

```
systemctl start mariadb.socket
systemctl enable mariadb.socket
```

These files alone only contain the UNIX and TCP sockets and basic network connection information to which the server is listening for connections. `@mariadb` is a UNIX abstract socket, which means it doesn't appear on the filesystem. Connectors based on MariaDB Connector/C is able to connect with these by using the socket name directly, provided the higher level implementation doesn't try to test for the file's existence first. Some connectors like PHP use `mysqlnd` that is a pure PHP implementation and as such are only able to connect to on filesystem UNIX sockets.

With systemd activated sockets there is only a file descriptor limit on the number of listening sockets that can be created.

When to Use Systemd Socket Activation

A common use case for systemd socket activated MariaDB is when there needs to be a quick boot up time. MariaDB needs to be ready to run, but it doesn't need to be running.

The ideal use case for systemd socket activation for MariaDB is for infrastructure providers running many multiple instances of MariaDB, where each instance is dedicated for a user.

Downsides to Using Systemd Socket Activation

From the time the connection occurs, the client is going to be waiting until MariaDB has fully initialized before MariaDB can process the awaiting connection. If MariaDB was previously hard shutdown and needs to perform an extensive InnoDB rollback, then the activation time may be larger than the desired wait time of the client connection.

Configuring Systemd Socket Activation

When MariaDB is run under systemd socket activation, the usual socket, port, and backlog system variables are ignored, as these settings are contained within the systemd socket definition file.

There is no configuration required in MariaDB to use MariaDB under socket activation.

The systemd options available are from the systemd documentation, however ListenStream and BackLog would be the most common configuration options.

As MariaDB isn't creating these sockets, the sockets don't need to be created with a `mysql` user. The sockets MariaDB may end up listening to under systemd socket activation, it may have not had the privileges to create itself.

Changes to the default `mariadb.socket` can be made in the same way as services, `systemctl edit mariadb.socket`, or using `/etc/systemd/system/mariadb.socket.d/someconfig.conf` files.

Extra Port

A `systemd` socket can be configured as an extra_port, by using `[FileDescriptorName=extra]` in the `.socket` file.

The `mariadb-extra.socket` file is already packaged and ready for use.

Multi-Instance Socket Activation

`mariadb@.socket` is MariaDB's packaged multi-instance definition. It creates multiple UNIX sockets based on the socket file started.

Starting `mariadb@bob.socket` uses the `mariadb@.socket` definition with `%I` within the definition replaced with "bob".

When something connects to a socket defined there, the `mariadb@bob.service` are started.

Systemd Socket Activation for Hosting Service Providers

A `systemd` socket activation service with multi-instance can provide an on-demand per user access to a hosting service provider's dedicated database. Here, "user" refers to the customer of the hosting service provider.

End User Benefits

This provides the following benefits for the user:

- Each user has their own dedicated instance with the following benefits:
  - The instance is free from the database contention of neighbors on MariaDB shared resources (table cache, connections, etc)
  - The user is free to change their own configuration of MariaDB, within the limits and permissions of the service provider.
  - Database service level backups, like mariadb-backup, are now directly available.
  - A user can install their own plugins.
  - The user can run a different database version to their neighbors.
  - If a user's neighbor triggers a fault in the server, the uder's instance isn't affected.
- The database runs as their unix user in the server facilitating:
  - User can directly migrate their MariaDB data directory to a different provider.
  - The user's data is protected from other users on a kernel level.

Hosting Service Provider Benefits

In addition to providing user benefits as a sales item, the following are additional benefits for the hosting service provider compared to a monolith service:

- Without passwords for the database, while still having security, support may be easier.
- When a user's database isn't active, there is no resource usage, only listening file descriptors by systemd.
- The socket activation transparently, with a minor startup time, starts the service as required.
- When the user's database hasn't had any activity after a time, it deactivates (MDEV-25282).
- Planned enhancements in InnoDB provide:
  - an on-demand consumption of memory (MDEV-25340 .
  - a proactive reduction in memory (MDEV-25341).
  - a memory resource pressure reduction in memory use (MDEV-24670).
- The service provider can still cap the user's database memory usage in a ulimit way that a user cannot override in settings.
- The service provider may choose a CPU/memory/IO based billing to the user on Linux `cgroup` accounting rather than the available comprared to the rather limited options in CREATE USER.
- Because a user's database shuts down when inactive, a database upgrade on the server does not take effect for the user until it passively shuts down, restarts, and then gets reactivated hence reducing user downtime..

Downsides to the Hosting Service Provider

The extra memory used by more instances. This is mitigated by the on-demand activation. The deactivation when idle, and improved InnoDB memory management.

With plenty of medium size database servers running, the Linux OOM kill has the opportunity to kill off only a small number of database servers running rather than everyones.

Configuration Items for a Per-User systemd Socket-Activitated Multi-Instance Service

From a server perspective the operation are as follows. To make the socket ready to connect and systemd listening to the socket, run these commands:

```
# systemctl start mariadb@username.socket
# systemctl start mariadb-extra@username.socket
```

To enable this on reboot (the same way as a `systemd` service):

```
# systemctl enable mariadb@username.socket
# systemctl enable mariadb-extra@username.socket
```

A MariaDB Template File

A global template file. Once installed as a user's `$HOME/.my.cnf` file, it becomes the default for many applications and for the MariaDB server itself.

```
# cat /etc/my.cnf.templ
[client]
socket=/home/USER/mariadb.sock

[client-server]
user=USER

[mariadbd]
datadir=/home/USER/mariadb-datadir
```

Custom Configuration for the Multi-Instance Service

This extends/modifies the MariaDB multi-instance service.

The features of this extension are:

- It automatically creates configuration file for user applications.
- It installs the database on first service start.
- `auth-root-*` in mariadb-install-db means that the user is their own privileged user with unix socket authentication active. This means non-that user cannot access another users service, even with access to the unix socket(s). For more information, see unix socket authentication security.
- If the MariaDB version was upgraded, the upgrade changes are made automatically.
- `LimitData` places a hard upper limit so the user doesn't exceed a portion of the server resources.

```
# cat /etc/systemd/system/mariadb@.service.d/user.conf
[Service]
User=%I
ProtectHome=false

Environment=MYSQLD_MULTI_INSTANCE="--defaults-file=/home/%I/.my.cnf"

ExecStartPre=
ExecStartPre=/bin/sh -c "[ -f /home/%I/.my.cnf ] || sed -e \"s/USER/%I/g\" /etc/my.cnf.templ >
/home/%I/.my.cnf"
ExecStartPre=mkdir -p /home/%I/mariadb-datadir
ExecStartPre=/usr/bin/mariadb-install-db $MYSQLD_MULTI_INSTANCE --rpm \
   --auth-root-authentication-method=socket --auth-root-socket-user=%I
ExecStartPost=/usr/bin/mariadb-upgrade $MYSQLD_MULTI_INSTANCE

# To limit user based tuning
LimitData=768M
# For io_uring use by innodb on < 5.12 kernels
LimitMEMLOCK=1M
```

Custom Configuration for the Multi-Instance Socket

This extends/modifies the MariaDB socket definition to be per user.

Create sockets based on the user of the instance (`%I`). Permissions are only necessary in the sense that the user can connect to them. It won't matter to the server. Access control is enforced within the server, however if the user web services are run as the user, `Mode=777` can be reduced. `@mariadb-%I` is a abstract unix socket not on the filesystem. It may help if a user is in a chroot. Not all applications can connect to abstract sockets.

```
# cat /etc/systemd/system/mariadb@.socket.d/user.conf
[Socket]
SocketUser=%I
SocketMode=777
ListenSteam=
ListenStream=@mariadb-%I
ListenStream=/home/%I/mariadb.sock
```

The extra socket provides the user the ability to access the server when all max-connections are used:

```
# cat /etc/systemd/system/mariadb-extra@.socket.d/user.conf
[Socket]
SocketUser=%I
SocketMode=777
ListenSteam=
ListenStream=@mariadb-extra-%I
ListenStream=/home/%I/mariadb-extra.sock
```

Converting mariadbd-safe Options to Systemd Options

`mariadb-service-convert` is a script included in many MariaDB packages that is used by the package manager to convert mariadbd-safe options to `systemd` options. It reads any explicit settings in the `[mariadbd-safe]` option group from option files, and its output is directed to `/etc/systemd/system/mariadb.service.d/migrated-from-my.cnf-settings.conf`. This helps to keep the configuration the same when upgrading from a version of MariaDB that does not use `systemd` to one that does.

Implicitly high defaults of open-files-limit may be missed by the conversion script and require explicit configuration. See Configuring the Open Files Limit.

This page is licensed: CC BY-SA / Gnu FDL

sysVinit

sysVinit is one of the most common service managers. On systems that use sysVinit, the mysql.server script is normally installed to `/etc/init.d/mysql`.

Interacting with the MariaDB Server Process

The service can be interacted with by using the service command.

Starting the MariaDB Server Process on Boot

On RHEL/CentOS and other similar distributions, the chkconfig command can be used to enable the MariaDB Server process at boot:

```
chkconfig --add mysql
chkconfig --level 345 mysql on
```

On Debian and Ubuntu and other similar distributions, the update-rc.d command can be used:

```
update-rc.d mysql defaults
```

Starting the MariaDB Server Process

```
service mysql start
```

Stopping the MariaDB Server Process

```
service mysql stop
```

Restarting the MariaDB Server Process

```
service mysql restart
```

Checking the Status of the MariaDB Server Process

```
service mysql status
```

Manually Installing mysql.server with SysVinit

If you install MariaDB from source or from a binary tarball that does not install mysql.server automatically, and if you are on a system that uses sysVinit, then you can manually install `mysql.server` with sysVinit. See mysql.server: Manually Installing with SysVinit for more information.

SysVinit and Galera Cluster

Bootstrapping a New Cluster

When using Galera Cluster with sysVinit, the first node in a cluster has to be started with `service mysql bootstrap`. See Getting Started with MariaDB Galera Cluster: Bootstrapping a New Cluster for more information.

This page is licensed: CC BY-SA / Gnu FDL

mariadbd-safe

The `mariadbd-safe` startup script is in MariaDB distributions on Linux and Unix. It is a wrapper that starts `mariadbd` with some extra safety features. For example, if `mariadbd-safe` notices that `mariadbd` has crashed, then `mariadbd-safe` automatically restarts `mariadbd`.

`mariadbd-safe` is the recommended way to start `mariadbd` on Linux and Unix distributions that do not support systemd. Additionally, the mysql.server init script used by sysVinit starts `mariadbd` with `mariadbd-safe` by default.

> **Note:**
>
> Previously, the client used to be called `mysqld_safe`, and can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Using mariadbd-safe

The command to use `mariadbd-safe` and the general syntax is:

```
mariadbd-safe [ --no-defaults | --defaults-file | --defaults-extra-file | --defaults-group-suffix | --
print-defaults ] <options> <mariadbd_options>
```

Options

Many of the options supported by `mariadbd-safe` are identical to options supported by mariadbd. If an unknown option is provided to `mariadbd-safe` on the command-line, then it is passed to `mariadbd`.

`mariadbd-safe` supports the following options:

Option	Description
–help	Display a help message and exit.
–autoclose	(NetWare only) On NetWare, mariadbd-safe provides a screen presence. When you unload (shut down) the mariadbd-safe NLM, the screen does not by default go away. Instead, it prompts for user input: NLM has terminated; Press any key to close the screen. If you want NetWare to close the screen automatically instead, use the –autoclose option to mariadbd-safe.
–basedir=path	The path to the MariaDB installation directory.
–core-file-size=size	The size of the core file that mariadbd should be able to create. The option value is passed to ulimit -c.
–crash-script=file	Script to call in the event of mariadbd crashing.
–datadir=path	The path to the data directory.
–defaults-extra-file=path	The name of an option file to be read in addition to the usual option files. This must be the first option on the command line if it is used. If the file does not exist or is otherwise inaccessible, the server will exit with an error.
–defaults-file=file_name	The name of an option file to be read instead of the usual option files. This must be the first option on the command line if it is used.

Option	Description
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.
–flush-caches	Flush and purge buffers/caches before starting the server.
–ledir=path	If mariadbd-safe cannot find the server, use this option to indicate the path name to the directory where the server is located.
–log-error=file_name	Write the error log to the given file.
–malloc-lib=lib	Preload shared library lib if available. See debugging MariaDB for an example.
–mysqld=prog_nam	The name of the server program (in the ledir directory) that you want to start. This option is needed if you use the MariaDB binary distribution but have the data directory outside of the binary distribution. If mariadbd-safe cannot find the server, use the –ledir option to indicate the path name to the directory where the server is located.
–mysqld-version=suffix	This option is similar to the –mysqld option, but you specify only the suffix for the server program name. The basename is assumed to be mysqld. For example, if you use–mysqld-version=debug, mariadbd-safe starts the mariadbd-debug program in the ledir directory. If the argument to –mysqld-version is empty, mariadbd-safe uses mysqld in the ledir directory.
–nice=priority	Use the nice program to set the server´s scheduling priority to the given value.
–no-defaults	Do not read any option files. This must be the first option on the command line if it is used.
–no-watch, –nowatch, –no-auto-restart	Exit after starting mariadbd.
–numa-interleave	Run mariadbd with its memory interleaved on all NUMA nodes.
–open-files-limit=count	The number of files that mariadbd should be able to open. The option value is passed to ulimit -n. Note that you need to start mariadbd-safe as root for this to work properly.
–pid-file=file_name	The path name of the process ID file.
–plugin-dir=dir_name	Directory for client-side plugins.
–port=port_num	The port number that the server should use when listening for TCP/IP connections. The port number must be 1024 or higher unless the server is started by the root system user.
–print-defaults	Print the program argument list and exit.
–skip-kill-mysqld	Do not try to kill stray mariadbd processes at startup. This option works only on Linux.
–socket=path	The Unix socket file that the server should use when listening for local connections.
–syslog, –skip-syslog	–syslog causes error messages to be sent to syslog on systems that support the logger program. –skip-syslog suppresses the use of syslog; messages are written to an error log file.
–syslog-tag=tag	For logging to syslog, messages from mariadbd-safe and mariadbd are written with a tag of mariadbd-safe and mariadbd, respectively. To specify a suffix for the tag, use –syslog-tag=tag, which modifies the tags to be mariadbd-safe-tag and mariadbd-tag.
–timezone=timezone	Set the TZ time zone environment variable to the given option value. Consult your operating system documentation for legal time zone specification formats. Also see Time Zones.
–user={user_name or user_id}	Run the mariadbd server as the user having the name user_name or the numeric user ID user_id. (“User” in this context refers to a system login account, not a MariaDB user listed in the grant tables.)

Option Files

In addition to reading options from the command-line, `mariadbd-safe` can also read options from option files. If an unknown option is provided to `mariadbd-safe` in an option file, then it is ignored.

The following options relate to how MariaDB command-line tools handles option files. They must be given as the **first argument** on the command-line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

Option Groups

`mariadbd-safe` reads options from the following option groups from option files:

Group	Description
[mysqld_safe]	Options read by mysqld_safe, which includes both MariaDB Server and MySQL Server.
[safe_mysqld]	Options read by mysqld_safe, which includes both MariaDB Server and MySQL Server.
[mariadbd-safe]	Options read by mariadbd_safe_safe from MariaDB Server. Available starting with MariaDB 10.4.6.
[mariadb-safe]	Options read by mysqld_safe from MariaDB Server. Deprecated, please avoid using this.

The `[safe_mariadbd]` option group is primarily supported for backward compatibility. You should rename such option groups to `[mariadbd-safe]` in MariaDB installations to prevent breakage in the future if this compatibility is removed.

`mariadbd-safe` also reads options from the following server option groups from option files:

Group	Description
[mysqld]	Options read by mysqld, which includes both MariaDB Server and MySQL Server.
[server]	Options read by MariaDB Server.
[mysqld-X.Y]	Options read by a specific version of mysqld, which includes both MariaDB Server and MySQL Server. For example, [mysqld-5.5].
[mariadb]	Options read by MariaDB Server.
[mariadb-X.Y]	Options read by a specific version of MariaDB Server.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[galera]	Options read by a galera-capable MariaDB Server. Available on systems compiled with Galera support.

For example, if you specify the log_error option in a server option group in an option file, like this:

```
[mariadb]
log_error=error.log
```

Then `mariadbd-safe` also uses this value for its own `--log-error` option:

Configuring the Open Files Limit

When using `mariadbd-safe`, the system's open files limit can be changed by providing the `--open-files-limit` option either on the command-line or in an option file. For example:

```
[mariadbd-safe]
open_files_limit=4294967295
```

The option value is passed to `ulimit -n`. Note that you need to start `mariadbd-safe` as root for this to work properly. However, you can't currently set this to `unlimited`. See MDEV-18410 about that.

When `mariadbd-safe` starts `mariadbd`, it also uses this option to set the value of the open_files_limit system variable for `mariadbd`.

Configuring the Core File Size

When using `mariadbd-safe`, if you would like to enable core dumps, the system's core file size limit can be changed by providing the `--core-file-size` option either on the command-line or in an option file. For example:

```
[mariadbd-safe]
core_file_size=unlimited
```

The option value is passed to `ulimit -c`. Note that you need to start `mariadbd-safe` as root for this to work properly.

Configuring MariaDB to Write the Error Log to Syslog

When using `mariadbd-safe`, if you would like to redirect the error log to the syslog, then that can easily be done by using the `--syslog` option. `mariadbd-safe` redirects two types of log messages to the syslog–its own log messages, and log messages for `mariadbd`.

- `mariadbd-safe` configures its own log messages to go to the `daemon` syslog facility. The log level for these messages is either `notice` or `error`, depending on the specific type of log message. The default tag is `mariadbd-safe`.
- `mariadbd-safe` also configures the log messages for `mariadbd` to go to the `daemon` syslog facility. The log level for these messages is `error`. The default tag is `mariadbd`.

Sometimes it can be helpful to add a suffix to the syslog tag, such as if you are running multiple instances of MariaDB on the same host. To add a suffix to each syslog tag, use the `--syslog-tag` option.

Specifying mariadbd

By default, `mariadbd-safe` tries to start an executable named `mariadbd`.

You can also specify another executable for `mariadbd-safe` to start instead of `mariadbd` by providing the `--mariadbd` or `--mariadbd-version` options either on the command-line or in an option file.

By default, it will look for `mariadbd` in the following locations in the following order:

- `$BASEDIR/libexec/mysqld`
- `$BASEDIR/sbin/mysqld`
- `$BASEDIR/bin/mysqld`
- `$PWD/bin/mysqld`
- `$PWD/libexec/mysqld`
- `$PWD/sbin/mysqld`
- `@libexecdir@/mysql`

Where `$BASEDIR` is set by the `--basedir` option, `$PWD` is the current working directory where `mariadbd-safe` was invoked, and `@libexecdir@` is set at compile-time by the `INSTALL_BINDIR` option for cmake.

You can also specify where the executable is located by providing the `--ledir` option either on the command-line or in an option file.

Specifying datadir

By default, `mariadbd-safe` will look for the `datadir` in the following locations in the following order:

- `$BASEDIR/data/mysql`
- `$BASEDIR/data`
- `$BASEDIR/var/mysql`
- `$BASEDIR/var`
- `@localstatedir@`

Where `$BASEDIR` is set by the `--basedir` option, and `@localstatedir@` is set at compile-time by the `INSTALL_MYSQLDATADIR` option for cmake.

You can also specify where the `datadir` is located by providing the `--datadir` option either on the command-line or in an option file.

Logging

When you use `mariadbd-safe` to start `mariadbd`, `mariadbd-safe` logs to the same destination as `mariadbd`.

`mariadbd-safe` has several log-related options:

- `--syslog`: Write error messages to syslog on systems that support the logger program.
- `--skip-syslog`: Do not write error messages to syslog.
  Messages are written to the default error log file (host_name.err in the data directory), or to a named file if the `--log-error` option is given.
- `--log-error=file_name`: Write error messages to the named error file.

If none of these options is provided, then the default is `--skip-syslog`.

If `--syslog` and `--log-error` are both provided, then a warning is issued and `--log-error` takes precedence.

`mariadbd-safe` also writes notices to `stdout` and errors to `stderr`.

Editing mariadbd-safe

`mariadbd-safe` is a `sh` script, so if you need to change its behavior, then it can easily be edited. However, you should not normally edit the script. A lot of behavior can be changed by providing options either on the command-line or in an option file.

If you do edit `mariadbd-safe`, then you should be aware of the fact that a package upgrade can overwrite your changes. If you would like to preserve your changes, be sure to have a backup.

NetWare

On NetWare, mariadbd-safe is a NetWare Loadable Module (NLM) that is ported from the original Unix shell script. It starts the server as follows:

1. Runs a number of system and option checks.
2. Runs a check on MyISAM tables.
3. Provides a screen presence for the MariaDB server.
4. Starts mariadbd, monitors it, and restarts it if it terminates in error.

5. Sends error messages from mariadbd to the host_name.err file in the data directory.
6. Sends mariadbd-safe screen output to the host_name.safe file in the data directory.

See Also

- How to increase max number of open files on Linux. This can be used to solve issues like this warning from mariadbd: `Changed limits: max_open_files: 1024 (requested 5000)"`
- mariadbd Options
- systemd

This page is licensed: CC BY-SA / Gnu FDL

mariadbd-multi

Before using mariadbd-multi be sure that you understand the meanings of the options that are passed to the mariadbd servers and why you would want to have separate mariadbd processes. Beware of the dangers of using multiple mariadbd servers with the same data directory. Use separate data directories, unless you know what you are doing. Starting multiple servers with the same data directory does not give you extra performance in a threaded system.

The `mariadbd-multi` startup script is in MariaDB distributions on Linux and Unix. It is a wrapper that is designed to manage several `mariadbd` processes running on the same host.

> **Note:**
>
> Previously, the client was called `mysqld_multi`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

In order for multiple `mariadbd` processes to work on the same host, these processes must:

- Use different Unix socket files for local connections.
- Use different TCP/IP ports for network connections.
- Use different data directories.
- Use different process ID files (specified by the `--pid-file` option) if using mariadbd-safe to start `mariadbd`.

`mariadbd-multi` can start or stop servers, or report their current status.

Using mariadbd-multi

The command to use `mariadbd-multi` and the general syntax is:

```
mariadbd-multi [options] {start|stop|report} [GNR[,GNR] ...]
```

`start`, `stop`, and `report` indicate which operation to perform.

You can specify which servers to perform the operation on by providing one or more `GNR` values. `GNR` refers to an option group number, and it is explained more in the option groups section below. If there is no `GNR` list, then `mariadbd-multi` performs the operation for all `GNR` values found in its option files.

Multiple `GNR` values can be specified as a comma-separated list. `GNR` values can also be specified as a range by separating the numbers by a dash. There must not be any whitespace characters in the `GNR` list.

For example:

This command starts a single server using option group `[mariadbd17]`:

```
mariadbd-multi start 17
```

This command stops several servers, using option groups `[mariadbd8]` and `[mariadbd10]` through `[mariadbd13]`:

```
mariadbd-multi stop 8,10-13
```

Options

`mariadbd-multi` supports the following options:

Option	Description
–example	Give an example of a config file with extra information.
–help	Display help and exit.
–log=filename	Specify the path and name of the log file. If the file exists, log output is appended to it.
–mysqladmin=prog_name	The mariadb-admin binary to be used to stop servers. Can be given within groups [mariadbd#].
–mariadbd=prog_name	The mariadbd binary to be used. Note that you can also specify mariadbd-safe as the value for this option. If you use mariadbd-safe to start the server, you can include the mariadbd or ledir options in the corresponding [mariadbdN] option group. These options indicate the name of the server that mariadbd-safe should start and the path name of the directory where the server is located. Example:\[mariadbd38]mariadbd = mariadbd-debugledir = /opt/local/mysql/libexec.
–no-log	Print to stdout instead of the log file. By default the log file is turned on.
–password=password	The password of the MariaDB account to use when invoking mariadb-admin. Note that the password value is not optional for this option, unlike for other MariaDB programs.
–silent	Silent mode; disable warnings.
–tcp-ip	Connect to the MariaDB server(s) via the TCP/IP port instead of the UNIX socket. This affects stopping and reporting. If a socket file is missing, the server may still be running, but can be accessed only via the TCP/IP port. By default connecting is done via the UNIX socket. This option affects stop and report operations.
–user=username	The user name of the MariaDB account to use when invoking mariadb-admin.
–verbose	Be more verbose.
–version	Display version information and exit.
–wsrep-new-cluster	Bootstrap a cluster. Added in MariaDB 10.1.15.

Option Files

In addition to reading options from the command-line, `mariadbd-multi` can also read options from option files. If an unknown option is provided to `mariadbd-multi` in an option file, then it is ignored.

The following options relate to how MariaDB command-line tools handles option files. They must be given as the first argument on the command-line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.

Option	Description
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

Option Groups

`mariadbd-multi` reads options from the following option groups from option files:

Group	Description
[mysqld_multi]	Options read by mysqld_multi, which includes both MariaDB Server and MySQL Server.

`mariadbd-multi` also searches option files for option groups with names like `[mariadbdN]`, where `N` can be any positive integer. This number is referred to in the following discussion as the option group number, or `GNR`:

Group	Description
[mysqld N]	Options read by a mysqld instance managed by mysqld_multi, which includes both MariaDB Server and MySQL Server. The N refers to the instance's GNR.

`GNR` values distinguish option groups from one another and are used as arguments to `mariadbd-multi` to specify which servers you want to start, stop, or obtain a status report for. The `GNR` value should be the number at the end of the option group name in the option file. For example, the `GNR` for an option group named `[mariadbd17]` is `17`.

Options listed in these option groups are the same that you would use in the regular server option groups used for configuring `mariadbd`. However, when using multiple servers, it is necessary that each one use its own value for options such as the Unix socket file and TCP/IP port number.

The `[mariadbd-multi]` option group can be used for options that are needed for `mariadbd-multi` itself. `[mariadbdN]` option groups can be used for options passed to specific `mariadbd` instances.

The regular server option groups can also be used for common options that are read by all instances:

Group	Description
[mysqld]	Options read by mysqld, which includes both MariaDB Server and MySQL Server.
[server]	Options read by MariaDB Server.
[mysqld-X.Y]	Options read by a specific version of mysqld, which includes both MariaDB Server and MySQL Server. For example, [mysqld-5.5].
[mariadb]	Options read by MariaDB Server.
[mariadb-X.Y]	Options read by a specific version of MariaDB Server.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[galera]	Options read by a galera-capable MariaDB Server. Available on systems compiled with Galera support.

For an example of how you might set up an option file, use this command:

```
mariadbd-multi --example
```

Authentication and Privileges

Make sure that the MariaDB account used for stopping the `mariadbd` processes (with the mariadb-admin utility) has the same user name and password for each server. Also, make sure that the account has the `SHUTDOWN` privilege. If the servers that you want to manage have different user names or passwords for the administrative accounts, you

might want to create an account on each server that has the same user name and password. For example, you might set up a common `multi_admin` account by executing the following commands for each server:

```
shell> mysql -u root -S /tmp/mysql.sock -p
Enter password:
mysql> GRANT SHUTDOWN ON *.*
 -> TO ´multi_admin´@´localhost´ IDENTIFIED BY ´multipass´;
```

Change the connection parameters appropriately when connecting to each one. Note that the host name part of the account name must allow you to connect as `multi_admin` from the host where you want to run `mariadbd-multi`.

User Account

Make sure that the data directory for each server is fully accessible to the Unix account that the specific `mariadbd` process is started as. If you run the `mariadbd-multi` script as the Unix `root` account, and if you want the `mariadbd` process to be started with another Unix account, then you can use the `--user` option with `mariadbd`. If you specify the `--user` option in an option file, and if you did not run the `mariadbd-multi` script as the Unix `root` account, then it will just log a warning and the `mariadbd` processes are started under the original Unix account.

Do not run the `mariadbd` process as the Unix `root` account, unless you know what you are doing.

Example

The following example shows how you might set up an option file for use with mariadbd-multi. The order in which the mariadbd programs are started or stopped depends on the order in which they appear in the option file. Group numbers need not form an unbroken sequence. The first and fifth [mariadbdN] groups were intentionally omitted from the example to illustrate that you can have “gaps” in the option file. This gives you more flexibility.

```
# This file should probably be in your home dir (~/.my.cnf)
           # or /etc/my.cnf
           # Version 2.1 by Jani Tolonen
           [mariadbd-multi]
           mariadbd     = /usr/local/bin/mariadbd-safe
           mysqladmin = /usr/local/bin/mysqladmin
           user       = multi_admin
           password   = multipass
           [mariadbd2]
           socket     = /tmp/mysql.sock2
           port       = 3307
           pid-file   = /usr/local/mysql/var2/hostname.pid2
           datadir    = /usr/local/mysql/var2
           language   = /usr/local/share/mysql/english
           user       = john
           [mariadbd3]
           socket     = /tmp/mysql.sock3
           port       = 3308
           pid-file   = /usr/local/mysql/var3/hostname.pid3
           datadir    = /usr/local/mysql/var3
           language   = /usr/local/share/mysql/swedish
           user       = monty
           [mariadbd4]
           socket     = /tmp/mysql.sock4
           port       = 3309
           pid-file   = /usr/local/mysql/var4/hostname.pid4
           datadir    = /usr/local/mysql/var4
           language   = /usr/local/share/mysql/estonia
           user       = tonu
           [mariadbd6]
           socket     = /tmp/mysql.sock6
           port       = 3311
           pid-file   = /usr/local/mysql/var6/hostname.pid6
           datadir    = /usr/local/mysql/var6
           language   = /usr/local/share/mysql/japanese
           user       = jani
```

This page is licensed: CC BY-SA / Gnu FDL

mariadbd Options

This page lists all of the options for `mariadbd` (called mysqld before MariaDB 10.5), ordered by topic. For a full alphabetical list of all mariadbd options, as well as server and status variables, see Full list of MariaDB options, system and status variables.

In many cases, the entry here is a summary, and links to the full description.

By convention, server variables have usually been specified with an underscore in the configuration files, and a dash on the command line. You can however specify underscores as dashes - they are interchangeable.

See Configuring MariaDB with Option Files for which files and groups mariadbd reads for it's default options.

> **Note:**
>
> Previously, the client used to be called `mysqld`, and can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Option Prefixes

`--autoset-*`

- Description: Sets the option value automatically. Only supported for certain options.

`--disable-*`

- Description: For all boolean options, disables the setting (equivalent to setting it to `0`). Same as `--skip`.

`--enable-*`

- Description: For all boolean options, enables the setting (equivalent to setting it to `1`).

`--loose-*`

- Description: Don't produce an error if the option doesn't exist.

`--maximum-*`

- Description: Sets the maximum value for the option.

`--skip-*`

- Description: For all boolean options, disables the setting (equivalent to setting it to `0`). Same as `--disable`.

Option File Options

`--defaults-extra-file`

- Command line: `--defaults-extra-file=name`
- Description: Read this extra option file after all other option files are read.
  - See Configuring MariaDB with Option Files.

`--defaults-file`

- Command line: `--defaults-file=name`
- Description: Only read options from the given option file.
  - See Configuring MariaDB with Option Files.

`--defaults-group-suffix`

- Command line: `--defaults-group-suffix=name`
- Description: In addition to the default option groups, also read option groups with the given suffix.
  - See Configuring MariaDB with Option Files.

`--no-defaults`

- Command line: `--no-defaults`
- Description: Don't read options from any option file.
  - See Configuring MariaDB with Option Files.

`--print-defaults`

- Command line: `--print-defaults`
- Description: Read options from option files, print all option values, and then exit the program.
  - See Configuring MariaDB with Option Files.

Compatibility Options

The following options have been added to MariaDB to make it more compliant with other MariaDB and MySQL versions. Options that are also system variables are listed after:

`-a, --ansi`

- Description: Use ANSI SQL syntax instead of MariaDB syntax. This mode will also set transaction isolation level serializable.

`--new`

- Description: Use new functionality that will exist in next version of MariaDB. This function exists to make it easier to prepare for an upgrade.

`--old-style-user-limits`

- Description: Enable old-style user limits (before MySQL 5.0.3, user resources were counted per each user+host vs. per account).

`--safe-mode`

- Description: Disable some potential unsafe optimizations. For 5.2, INSERT DELAYED is disabled, myisam_recover_options is set to DEFAULT (automatically recover crashed MyISAM files) and the query cache is disabled. For Aria tables, disable bulk insert optimization to enable one to use aria_read_log to recover tables even if tables are deleted (good for testing recovery).

`--skip-new`

- Description: Disables –new.

Compatibility Options and System Variables

- –new-mode
- –old
- –old-alter-table
- –old-mode
- –old-passwords
- –show-old-temporals

Locale Options

Options that are also system variables are listed after:

`--character-set-client-handshake`

- Command line: `--character-set-client-handshake`
- Description: Don't ignore client side character set value sent during handshake. `--skip-character-set-client-handshake` will ignore the client value and use the default server value.

`--default-character-set`

- Command line: `--default-character-set=name`
- Description: Still available as an option for setting the default character set for clients and their connections, it was deprecated and removed in MariaDB 10.2 as a server option. Use character-set-server instead.

`--language`

- Description: This option can be used to set the server's language for error messages. This option can be specified either as a language name or as the path to the directory storing the language's error message file. See Server Locales for a list of supported locales and their associated languages.
  - This option is deprecated. Use the lc_messages and lc_messages_dir system variables instead.

- See Setting the Language for Error Messages for more information.

Locale Options and System Variables

- character-set-filesystem
- character-set-client
- character-set-connection
- character-set-database
- character-set-filesystem
- character-set-results
- character-set-server
- character-set-system
- character-sets-dir
- collation-connection
- collation-database
- collation-server
- default-week-format
- default-time-zone
- lc-messages
- lc-messages-dir
- lc-time-names

Windows Options

Options that are also system variables are listed after:

`--console`

- Description: Windows-only option that keeps the console window open and for writing log messages to stderr and stdout. If specified together with –log-error, the last option will take precedence.

`--install`

- Description: Windows-only option that installs the `mariadbd` process as a Windows service.
  - The Windows service created with this option auto-starts. If you want a service that is started on demand, then use the –install-manual option.
  - This option takes a service name as an argument. If this option is provided without a service name, then the service name defaults to “MARIADB”.
  - This option is deprecated and may be removed in a future version. See MDEV-19358 for more information.

`--install-manual`

- Description: Windows-only option that installs the `mariadbd` process as a Windows service.
  - The Windows service created with this option is started on demand. If you want a service that auto-starts, use the –install option.
  - This option takes a service name as an argument. If this option is provided without a service name, then the service name defaults to “MARIADB”.
  - This option is deprecated and may be removed in a future version. See MDEV-19358 for more information.

`--remove`

- Description: Windows-only option that removes the Windows service created by the –install or –install-manual options.
  - This option takes a service name as an argument. If this option is provided without a service name, then the service name defaults to “MARIADB”.
  - This option is deprecated and may be removed in a future version. See MDEV-19358 for more information.

`--slow-start-timeout`

- Description: Windows-only option that defines the maximum number of milliseconds that the service control manager should wait before trying to kill the Windows service during startup. Defaults to `15000`.

`--standalone`

- Description: Windows-only option that has no effect. Kept for compatibility reasons.

Windows Options and System Variables

The following options and system variables are related to using MariaDB on Windows:

- –named-pipe

Replication and Binary Logging Options

The following options are related to replication and the binary log. Options that are also system variables are listed after:

`--abort-slave-event-count`

- Command line: `--abort-slave-event-count=#`
- Description: Option used by mysql-test for debugging and testing of replication.

`--binlog-do-db`

- Command line: `--binlog-do-db=name`
- Description: This option allows you to configure a replication master to write statements and transactions affecting databases that match a specified name into its binary log. Since the filtered statements or transactions will not be present in the binary log, its replicas will not be able to replicate them.
  - This option will not work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
  - This option can not be set dynamically. Available as a system variable from MariaDB 11.2.0.
  - When setting it on the command-line or in a server option group in an option file, the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times.
  - See Replication Filters for more information.

`--binlog-ignore-db`

- Command line: `--binlog-ignore-db=name`
- Description: This option allows you to configure a replication master to not write statements and transactions affecting databases that match a specified name into its binary log. Since the filtered statements or transactions will not be present in the binary log, its replicas will not be able to replicate them.
  - This option will not work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
  - This option can not be set dynamically. Available as a system variable from MariaDB 11.2.0.

    - When setting it on the command-line or in a server option group in an option file, the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times.
    - See Replication Filters for more information.

`--binlog-row-event-max-size`

- Command line: `--binlog-row-event-max-size=#`
- Description: The maximum size of a row-based binary log event in bytes. Rows will be grouped into events smaller than this size if possible. The value has to be a multiple of 256. Available as a system variable from MariaDB 11.2.0.
- Default value `8192`

`--disconnect-slave-event-count`

- Command line: `--disconnect-slave-event-count=#`
- Description: Option used by mysql-test for debugging and testing of replication.

`--flashback`

- Command line: `--flashback`
- Description: Setup the server to use flashback. This enables the binary log and sets `binlog_format=ROW`.

`--log-basename`

- Command line: `--log-basename=name`
- Description: Basename for all log files and the .pid file. This sets all log file names at once (in 'datadir') and is normally the only option you need for specifying log files. This is especially recommended to be set if you are using replication as it ensures that your log file names are not dependent on your host name. Sets names for the binary log, relay log, general query log, slow query log and error log. Note that if you explicitly set log file names with any of these other options; log-bin-index, relay-log, relay-log-index, general-log-file, log_slow_query_file (slow_query_log_file), log_error, and pid-file, these should be placed after `--log-basename` in the config files. Later settings override earlier settings, so `log-basename` will override any earlier log file name settings.

`--log-bin-trust-routine-creators`

- Command line: `--log-bin-trust-routine-creators`
- Description: Deprecated, use log-bin-trust-function-creators.

`--master-host`

- Command line: `--master-host=name`
- Description: Primary hostname or IP address for replication. If not set, the replica thread will not be started. Note that the setting of master-host will be ignored if there exists a valid master.info file.

`--master-info-file`

- Command line: `--master-info-file=name`
- Description: Name and location of the file on the replica where the `MASTER_LOG_FILE` and `MASTER_LOG_POS` options (i.e. the binary log position on the primary) and most other CHANGE MASTER options are written. The replica's I/O thread keeps this binary log position updated as it downloads events.
    - See CHANGE MASTER TO: Option Persistence for more information.

`--master-password`

- Command line: `--master-password=name`
- Description: The password the replica thread will authenticate with when connecting to the primary. If not set, an empty password is assumed. The value in master.info will take precedence if it can be read.

`--master-port`

- Command line: `--master-port=#`
- Description: The port the master is listening on. If not set, the compiled setting of MYSQL_PORT is assumed. If you have not tinkered with configure options, this should be 3306. The value in master.info will take precedence if it can be read.

`--master-retry-count`

- Command line: `--master-retry-count=#`
- Description: Number of times a replica will attempt to connect to a primary before giving up. The retry interval is determined by the MASTER_CONNECT_RETRY option for the CHANGE MASTER statement. A value of 0 means the replica will not stop attempting to reconnect. Reconnects are triggered when a replica has timed out. See slave_net_timeout.
- Default Value: `86400` through 10.5, `100000` as of 10.6
- Range - 32 bit: `0 to 4294967295`
- Range - 64 bit: `0 to 18446744073709551615`

`--master-ssl`

- Command line: `--master-ssl`
- Description: Enable the replica to connect to the master using TLS.

`--master-ssl-ca`

- Command line: `--master-ssl-ca[=name]`
- Description: Master TLS CA file. Only applies if you have enabled master-ssl.

`--master-ssl-capath`

- Command line: `--master-ssl-capath[=name]`
- Description: Master TLS CA path. Only applies if you have enabled master-ssl.

`--master-ssl-cert`

- Command line: `--master-ssl-cert[=name]`
- Description: Master TLS certificate file name. Only applies if you have enabled master-ssl.

`--master-ssl-cipher`

- Command line: `--master-ssl-cipher[=name]`
- Description: Master TLS cipher. Only applies if you have enabled master-ssl.

`--master-ssl-key`

- Command line: `--master-ssl-key[=name]`
- Description: Master TLS keyfile name. Only applies if you have enabled master-ssl.

`--master-user`

- Command line: `--master-user=name`
- Description: The username the replica thread will use for authentication when connecting to the primary. The user must have FILE privilege. If the primary user is not set, user test is assumed. The value in master.info will take precedence if it can be read.

`--max-binlog-dump-events`

- Command line: `--max-binlog-dump-events=#`
- Description: Option used by mysql-test for debugging and testing of replication.

`--replicate-same-server-id`

- Command line: `--replicate-same-server-id`
- Description: In replication, if set to 1, do not skip events having our server id. Default value is 0 (to break infinite loops in circular replication). Can't be set to 1 if log-slave-updates is used. Added as a system variable in MariaDB 12.0.

`--sporadic-binlog-dump-fail`

- Command line: `--sporadic-binlog-dump-fail`
- Description: Option used by mysql-test for debugging and testing of replication.

`--sysdate-is-now`

- Command line: `--sysdate-is-now`
- Description: Non-default option to alias SYSDATE() to NOW() to make it safe for replication. Since 5.0, SYSDATE() has returned a `dynamic' value different for different invocations, even within the same statement.

Replication and Binary Logging Options and System Variables

The options and system variables related to Replication and Binary Logging are described on Replication and Binary Logging System Variables.

Semisynchronous Replication Options and System Variables

The options and system variables related to Semisynchronous Replication are described here.

Optimizer Options

Options that are also system variables are listed after:

`--record-buffer`

- Command line: `--record-buffer=#`
- Description: Old alias for read_buffer_size.
- Removed: MariaDB 5.5

`--table-cache`

- Command line: `--table-open-cache=#`
- Description: Removed; use –table-open-cache instead.
- Removed: MariaDB 5.3.1

Optimizer Options and System Variables

- alter-algorithm
- analyze-max-length
- analyze-sample-percentage
- big-tables
- bulk-insert-buffer-size
- expensive-subquery-limit
- join-buffer-size
- join-buffer-space-limit
- join-cache-level
- max-heap-table-size
- max-join-size
- max-seeks-for-key
- max-sort-length
- mrr-buffer-size
- optimizer-adjust-secondary-key-costs
- optimizer-extra-pruning-depth
- optimizer-join-limit-pref-ratio
- optimizer-max-sel-arg-weight
- optimizer-max-sel-args
- optimizer-prune-level
- optimizer-record-context
- optimizer-search-depth
- optimizer-selectivity-sampling-limit
- optimizer-switch
- optimizer-trace
- optimizer-trace-max-mem-size
- optimizer-use-condition-selectivity
- query-alloc-block-size
- query-prealloc-size
- range-alloc-block-size
- read-buffer-size
- rowid-merge-buff-size
- table-definition-cache
- table-open-cache
- table-open-cache-instances
- tmp-disk-table-size
- tmp-memory-table-size
- tmp-table-size
- use-stat-tables

Storage Engine Options

`--skip-bdb`

- Command line: `----skip-bdb`

- Description: Deprecated option; Exists only for compatibility with very old my.cnf files.
- Removed: MariaDB 10.5.1

`--external-locking`

- Command line: `--external-locking`
- Description: Use system (external) locking (disabled by default). With this option enabled you can run myisamchk to test (not repair) tables while the server is running. Disable with –skip-external-locking. From MariaDB 10.2.40, MariaDB 10.3.31, MariaDB 10.4.21, MariaDB 10.5.12, MariaDB 10.6.4 and all later version, this effects InnoDB and can be used to prevent multiple instances running on the same data.

MyISAM Storage Engine Options

The options related to the MyISAM storage engine are described below. Options that are also system variables are listed after:

`--log-isam`

- Command line: `--log-isam[=file_name]`
- Description: Enable the MyISAM log, which logs all MyISAM changes to file. If no filename is provided, the default, myisam.log is used.

MyISAM Storage Engine Options and System Variables

Some options and system variables related to the MyISAM storage engine can be found here. Direct links to many of them can be found below.

- concurrent-insert
- delayed-insert-limit
- delayed-insert-timeout
- delayed-queue-size
- keep-files-on-create
- key-buffer-size
- key-cache-age-threshold
- key-cache-block-size
- key-cache-division-limit
- key-cache-file-hash-size
- key-cache-segments
- myisam-block-size
- myisam-data-pointer-size
- myisam-max-sort-file-size
- myisam-mmap-size
- myisam-recover-options
- myisam-repair-threads
- myisam-sort-buffer-size
- myisam-stats-method
- myisam-use-mmap

InnoDB Storage Engine Options

The options related to the InnoDB storage engine are described below. Options that are also system variables are listed after:

`--innodb`

- Command line: `--innodb=value`, `--skip-innodb`
- Description: This variable controls whether or not to load the InnoDB storage engine. Possible values are `ON`, `OFF`, `FORCE` or `FORCE_PLUS_PERMANENT` (from MariaDB 5.5). If set to `OFF` (the same as –skip-innodb), since InnoDB is the default storage engine, the server will not start unless another storage engine has been chosen with –default-storage-engine. `FORCE` means that the storage engine must be successfully loaded, or else the server won't start. `FORCE_PLUS_PERMANENT` enables the plugin, but if plugin cannot initialize, the server will not start. In addition, the plugin cannot be uninstalled while the server is running.

`--innodb-cmp`

- Command line: `--innodb-cmp`
- Description:
- Default: `ON`

`--innodb-cmp-reset`

- Command line: `--innodb-cmp-reset`
- Description:
- Default: `ON`

`--innodb-cmpmem`

- Command line: `--innodb-cmpmem`
- Description:
- Default: `ON`

`--innodb-cmpmem-reset`

- Command line: `--innodb-cmpmem-reset`
- Description:
- Default: `ON`

`--innodb-file-io-threads`

- Command line: `--innodb-file-io-threads`
- Description:
- Default: `4`
- Removed: MariaDB 10.3.0

`--innodb-index-stats`

- Command line: `--innodb-index-stats`
- Description:
- Default: `ON`
- Removed: MariaDB 10.0.0

`--innodb-lock-waits`

- Command line: `--innodb-lock-waits`
- Description:
- Default: `ON`

`--innodb-locks`

- Command line: `--innodb-locks`
- Description:
- Default: `ON`

`--innodb-rseg`

- Command line: `--innodb-rseg`
- Description:
- Default: `ON`
- Removed: MariaDB 10.0.0

`--innodb-status-file`

- Command line: `--innodb-status-file`
- Description:
- Default: `FALSE`

`--innodb-sys-indexes`

- Command line: `--innodb-sys-indexes`
- Description:
- Default: `ON`

`--innodb-sys-stats`

- Command line: `--innodb-sys-stats`
- Description:
- Default: `ON`
- Removed: MariaDB 10.0.0

`--innodb-sys-tables`

- Command line: `--innodb-sys-tables`
- Description:
- Default: `ON`

`--innodb-table-stats`

- Command line: `--innodb-table-stats`
- Description:
- Default: `ON`
- Removed: MariaDB 10.0.0

`--innodb-trx`

- Command line: `--innodb-trx`
- Description:
- Default: `ON`

InnoDB Storage Engine Options and System Variables

Some options and system variables related to the InnoDB storage engine can be found here. Direct links to many of them can be found below.

- ignore-builtin-innodb
- innodb-adaptive-checkpoint
- innodb-adaptive-flushing
- innodb-adaptive-flushing-lwm
- innodb-adaptive-flushing-method
- innodb-adaptive-hash-index
- innodb-adaptive-hash-index-partitions
- innodb-adaptive-hash-index-parts
- innodb-adaptive-max-sleep-delay
- innodb-additional-mem-pool-size
- innodb-alter-copy-bulk
- innodb-api-bk-commit-interval
- innodb-api-disable-rowlock
- innodb-api-enable-binlog
- innodb-api-enable-mdl
- innodb-api-trx-level
- innodb-auto-lru-dump
- innodb-autoextend-increment
- innodb-autoinc-lock-mode
- innodb-background-scrub-data-check-interval
- innodb-background-scrub-data-compressed
- innodb-background-scrub-data-interval
- innodb-background-scrub-data-uncompressed
- innodb-blocking-buffer-pool-restore
- innodb-buf-dump-status-frequency
- innodb-buffer-pool-chunk-size
- innodb-buffer-pool-dump-at-shutdown
- innodb-buffer-pool-dump-now
- innodb-buffer-pool-dump-pct
- innodb-buffer-pool-evict
- innodb-buffer-pool-filename
- innodb-buffer-pool-instances
- innodb-buffer-pool-load-abort
- innodb-buffer-pool-load-at-startup
- innodb-buffer-pool-load-now
- innodb-buffer-pool-load-pages-abort
- innodb-buffer-pool-populate
- innodb-buffer-pool-restore-at-startup
- innodb-buffer-pool-shm-checksum
- innodb-buffer-pool-shm-key
- innodb-buffer-pool-size

- innodb-buffer-pool-size-auto-min
- innodb-buffer-pool-size-max
- innodb-change-buffer-max-size
- innodb-change-buffering
- innodb-change-buffering-debug
- innodb-checkpoint-age-target
- innodb-checksum-algorithm
- innodb-checksums
- innodb-cleaner-lsn-age-factor
- innodb-cmp-per-index-enabled
- innodb-commit-concurrency
- innodb-compression-algorithm
- innodb-compression-failure-threshold-pct
- innodb-compression-level
- innodb-compression-pad-pct-max
- innodb-concurrency-tickets
- innodb-corrupt-table-action
- innodb-data-file-buffering
- innodb-data-file-path
- innodb-data-file-write-through
- innodb-data-home-dir
- innodb-deadlock-detect
- innodb-deadlock-report
- innodb-default-encryption-key-id
- innodb-default-page-encryption-key
- innodb-default-row-format
- innodb-defragment
- innodb-defragment-fill-factor
- innodb-defragment-fill-factor-n-recs
- innodb-defragment-frequency
- innodb-defragment-n-pages
- innodb-defragment-stats-accuracy
- innodb-dict-size-limit
- innodb_disable_sort_file_cache
- innodb-doublewrite
- innodb-doublewrite-file
- innodb-empty-free-list-algorithm
- innodb-enable-unsafe-group-commit
- innodb-encrypt-log
- innodb-encrypt-tables
- innodb-encrypt-temporary-tables
- innodb-encryption-rotate-key-age
- innodb-encryption-rotation_iops
- innodb-encryption-threads
- innodb-extra-rsegments

- innodb-extra-undoslots
- innodb-fake-changes
- innodb-fast-checksum
- innodb-fast-shutdown
- innodb-fatal-semaphore-wait-threshold
- innodb-file-format
- innodb-file-format-check
- innodb-file-format-max
- innodb-file-per-table
- innodb-fill-factor
- innodb-flush-log-at-trx-commit
- innodb-flush-method
- innodb-flush-neighbor-pages
- innodb-flush-neighbors
- innodb-flush-sync
- innodb-flushing-avg-loops
- innodb-force-load-corrupted
- innodb-force-primary-key
- innodb-force-recovery
- innodb-foreground-preflush
- innodb-ft-aux-table
- innodb-ft-cache-size
- innodb-ft-enable-diag-print
- innodb-ft-enable-stopword
- innodb-ft-max-token-size
- innodb-ft-min-token-size
- innodb-ft-num-word-optimize
- innodb-ft-result-cache-limit
- innodb-ft-server-stopword-table
- innodb-ft-sort-pll-degree
- innodb-ft-total-cache-size
- innodb-ft-user-stopword-table
- innodb-ibuf-accel-rate
- innodb-ibuf-active-contract
- innodb-ibuf-max-size
- innodb-idle-flush-pct
- innodb-immediate-scrub-data-uncompressed
- innodb-import-table-from-xtrabackup
- innodb-instant-alter-column-allowed
- innodb-instrument-semaphores
- innodb-io-capacity
- innodb-io-capacity-max
- innodb-large-prefix
- innodb-lazy-drop-table
- innodb-lock-schedule-algorithm

- innodb-locking-fake-changes
- innodb-locks-unsafe-for-binlog
- innodb-log-arch-dir
- innodb-log-arch-expire-sec
- innodb-log-archive
- innodb-log-block-size
- innodb-log-buffer-size
- innodb-log-checkpoint-now
- innodb-log-checksum-algorithm
- innodb-log-checksums
- innodb-log-compressed-pages
- innodb-log-file-buffering
- innodb-log-file-mmap
- innodb-log-file-size
- innodb-log-file-write-through
- innodb-log-files-in-group
- innodb-log-group-home-dir
- innodb-log-optimize-ddl
- innodb-log-spin-wait-delay
- innodb-log-write-ahead-size
- innodb-lru-flush-size
- innodb-lru-scan-depth
- innodb-max-bitmap-file-size
- innodb-max-changed-pages
- innodb-max-dirty-pages-pct
- innodb-max-dirty-pages-pct-lwm
- innodb-max-purge-lag
- innodb-max-purge-lag-delay
- innodb-max-purge-lag-wait
- innodb-max-undo-log-size
- innodb-merge-sort-block-size
- innodb-mirrored-log-groups
- innodb-monitor-disable
- innodb-monitor-enable
- innodb-monitor-reset
- innodb-monitor-reset-all
- innodb-mtflush-threads
- innodb-numa-interleave
- innodb-old-blocks-pct
- innodb-old-blocks-time
- innodb-online-alter-log-max-size
- innodb-open-files
- innodb-optimize-fulltext-only
- innodb-page-cleaners
- innodb-page-size

- innodb-pass-corrupt-table
- innodb-prefix-index-cluster-optimization
- innodb-print-all-deadlocks
- innodb-purge-batch-size
- innodb-purge-rseg-truncate-frequency
- innodb-purge-threads
- innodb-random-read-ahead
- innodb-read-ahead
- innodb-read-ahead-threshold
- innodb-read-io-threads
- innodb-read-only
- innodb-recovery-update-relay-log
- innodb-replication-delay
- innodb-rollback-on-timeout
- innodb-rollback-segments
- innodb-safe-truncate
- innodb-sched-priority-cleaner
- innodb-scrub-log
- innodb-scrub-log-interval
- innodb-scrub-log-speed
- innodb-show-locks-held
- innodb-show-verbose-locks
- innodb-snapshot-isolation
- innodb-sort-buffer-size
- innodb-spin-wait-delay
- innodb-stats-auto-recalc
- innodb-stats-auto-update
- innodb-stats-include-delete-marked
- innodb-stats-method
- innodb-stats-modified-counter
- innodb-stats-on-metadata
- innodb-stats-persistent
- innodb-stats-persistent-sample-pages
- innodb-stats-sample-pages
- innodb-stats-transient-sample-pages
- innodb-stats-traditional
- innodb-stats-update-need-lock
- innodb-status-output
- innodb-status-output-locks
- innodb-strict-mode
- innodb-support-xa
- innodb-sync-array-size
- innodb-sync-spin-loops
- innodb-table-locks
- innodb-temp-data-file-path

- innodb-thread-concurrency
- innodb-thread-concurrency-timer-based
- innodb-thread-sleep-delay
- innodb-tmpdir
- innodb-track-changed-pages
- innodb-track-redo-log-now
- innodb-truncate-temporary-tablespace-now
- innodb-undo-directory
- innodb-undo-log-truncate
- innodb-undo-logs
- innodb-undo-tablespaces
- innodb-use-atomic-writes
- innodb-use-fallocate
- innodb-use-global-flush-log-at-trx-commit
- innodb-use-mtflush
- innodb-use-native_aio
- innodb-use-purge-thread
- innodb-use-stacktrace
- innodb-use-sys-malloc
- innodb-use-sys-stats-table
- innodb-use-trim
- innodb-write-io-threads
- skip-innodb
- skip-innodb-checksums
- skip-innodb-doublewrite

Aria Storage Engine Options

Options related to the Aria storage engine are listed below:

Aria Storage Engine Options and System Variables

Some options and system variables related to the Aria storage engine can be found here. Direct links to many of them can be found below.

- aria-block-size
- aria-checkpoint-interval
- aria-checkpoint-log-activity
- aria-encrypt-tables
- aria-force-start-after-recovery-failures
- aria-group-commit
- aria-group-commit-interval
- aria-log-dir-path
- aria-log-file-size
- aria-log-purge-type
- aria-max-sort-file-size
- aria-page-checksum
- aria-pagecache-age-threshold

- aria-pagecache-buffer-size
- aria-pagecache-division-limit
- aria-pagecache-file-hash-size
- aria-pagecache-segments
- aria-recover
- aria-recover-options
- aria-repair-threads
- aria-sort-buffer-size
- aria-stats-method
- aria-sync-log-dir
- aria-used-for-temp-tables
- deadlock-search-depth-long
- deadlock-search-depth-short
- deadlock-timeout-long
- deadlock-timeout-short

MyRocks Storage Engine Options

The options and system variables related to the MyRocks storage engine can be found here.

S3 Storage Engine Options

The options and system variables related to the S3 storage engine can be found here.

CONNECT Storage Engine Options

The options related to the CONNECT storage engine are described below.

CONNECT Storage Engine Options and System Variables

Some options and system variables related to the CONNECT storage engine can be found here. Direct links to many of them can be found below.

- connect-class-path
- connect-cond-push
- connect-conv-size
- connect-default-depth
- connect-default-prec
- connect-enable-mongo
- connect-exact-info
- connect-force_bson
- connect-indx-map
- connect-java-wrapper
- connect-json-all-path
- connect-json-grp-size
- connect-json-null
- connect-jvm-path
- connect-type-conv
- connect-use-tempfile
- connect-work-size

- connect-xtrace

Spider Storage Engine Options

The options and system variables related to the Spider storage engine can be found here.

Mroonga Storage Engine Options

The options and system variables related to the Mroonga storage engine can be found here.

Vector Options

The options and system variables related to Vectors storage engine (beginning with `mhnsw`) can be found here.

Performance Schema Options

The options related to the Performance Schema are described below. Options that are also system variables are listed after:

`--performance-schema-consumer-events-stages-current`

- Command line: `--performance-schema-consumer-events-stages-current`
- Description: Enable the events-stages-current consumer.
- Default: `OFF`

`--performance-schema-consumer-events-stages-history`

- Command line: `--performance-schema-consumer-events-stages-history`
- Description: Enable the events-stages-history consumer.
- Default: `OFF`

`--performance-schema-consumer-events-stages-history-long`

- Command line: `--performance-schema-consumer-events-stages-history-long`
- Description: Enable the events-stages-history-long consumer.
- Default: `OFF`

`--performance-schema-consumer-events-statements-current`

- Command line: `--performance-schema-consumer-events-statements-current`
- Description: Enable the events-statements-current consumer. Use `--skip-performance-schema-consumer-events-statements-current` to disable.
- Default: `ON`

`--performance-schema-consumer-events-statements-history`

- Command line: `--performance-schema-consumer-events-statements-history`
- Description: Enable the events-statements-history consumer.
- Default: `OFF`

`--performance-schema-consumer-events-statements-history-long`

- Command line: `--performance-schema-consumer-events-statements-history-long`
- Description: Enable the events-statements-history-long consumer.
- Default: `OFF`

`--performance-schema-consumer-events-waits-current`

- Command line: `--performance-schema-consumer-events-waits-current`
- Description: Enable the events-waits-current consumer.
- Default: `OFF`

`--performance-schema-consumer-events-waits-history`

- Command line: `--performance-schema-consumer-events-waits-history`
- Description: Enable the events-waits-history consumer.
- Default: `OFF`

`--performance-schema-consumer-events-waits-history-long`

- Command line: `--performance-schema-consumer-events-waits-history-long`
- Description: Enable the events-waits-history-long consumer.
- Default: `OFF`

`--performance-schema-consumer-global-instrumentation`

- Command line: `--performance-schema-consumer-global-instrumentation`
- Description: Enable the global-instrumentation consumer. Use `--skip-performance-schema-consumer-global-instrumentation` to disable.
- Default: `ON`

`--performance-schema-consumer-statements-digest`

- Command line: `--performance-schema-consumer-statements-digest`
- Description: Enable the statements-digest consumer. Use `--skip-performance-schema-consumer-statements-digest` to disable.
- Default: `ON`

`--performance-schema-consumer-thread-instrumentation`

- Command line: `--performance-schema-consumer-thread-instrumentation`
- Description: Enable the statements-thread-instrumentation. Use `--skip-performance-schema-thread-instrumentation` to disable.
- Default: `ON`

Performance Schema Options and System Variables

Some options and system variables related to the Performance Schema can be found here. Direct links to many of them can be found below.

- performance-schema
- performance-schema-accounts-size
- performance-schema-digests-size
- performance-schema-events-stages-history-long-size
- performance-schema-events-stages-history-size
- performance-schema-events-statements-history-long-size
- performance-schema-events-statements-history-size
- performance-schema-events-waits-history-long-size
- performance-schema-events-waits-history-size

- performance-schema-hosts-size
- performance-schema-max-cond-classes
- performance-schema-max-cond-instances
- performance-schema-max-digest-length
- performance-schema-max-file-classes
- performance-schema-max-file-handles
- performance-schema-max-file-instances
- performance-schema-max-mutex-classes
- performance-schema-max-mutex-instances
- performance-schema-max-rwlock-classes
- performance-schema-max-rwlock-instances
- performance-schema-max-socket-classes
- performance-schema-max-socket-instances
- performance-schema-max-stage-classes
- performance-schema-max-statement-classes
- performance-schema-max-table-handles
- performance-schema-max-table-instances
- performance-schema-max-thread-classes
- performance-schema-max-thread-instances
- performance-schema-session-connect-attrs-size
- performance-schema-setup-actors-size
- performance-schema-setup-objects-size
- performance-schema-users-size

Galera Cluster Options

The options related to Galera Cluster are described below. Options that are also system variables are listed after:

`--wsrep-new-cluster`

- Command line: `--wsrep-new-cluster`
- Description: Bootstrap a cluster. It works by overriding the current value of wsrep_cluster_address. It is recommended not to add this option to the config file as this will trigger bootstrap on every server start.

Galera Cluster Options and System Variables

Some options and system variables related to Galera Cluster can be found here. Direct links to many of them can be found below.

- wsrep-allowlist
- wsrep_applier_retry_count
- wsrep-auto-increment-control
- wsrep-causal-reads
- wsrep-certify-nonPK
- wsrep-cluster-address
- wsrep-cluster-name
- wsrep-convert-LOCK-to-trx
- wsrep-data-home-dir
- wsrep-dbug-option

- wsrep-debug
- wsrep-desync
- wsrep-dirty-reads
- wsrep-drupal-282555-workaround
- wsrep-forced-binlog-format
- wsrep-gtid-domain-id
- wsrep-gtid-mode
- wsrep-ignore-apply-errors
- wsrep-load-data-splitting
- wsrep-log-conflicts
- wsrep-max-ws-rows
- wsrep-max-ws-size
- wsrep-mode
- wsrep-mysql-replication-bundle
- wsrep-node-address
- wsrep-node-incoming-address
- wsrep-node-name
- wsrep-notify-cmd
- wsrep-on
- wsrep-OSU-method
- wsrep-provider
- wsrep-provider-options
- wsrep-recover
- wsrep-reject_queries
- wsrep-retry-autocommit
- wsrep-slave-FK-checks
- wsrep-slave-threads
- wsrep-slave-UK-checks
- wsrep-sr-store
- wsrep-sst-auth
- wsrep-sst-donor
- wsrep-sst-donor-rejects-queries
- wsrep-sst-method
- wsrep-sst-receive-address
- wsrep-start-position
- wsrep-status-file
- wsrep-strict-ddl
- wsrep-sync-wait
- wsrep-trx_fragment_size
- wsrep-trx_fragment_unit

Options When Debugging mariadbd

`--debug-assert-if-crashed-table`

- Description: Do an assert in handler::print_error() if we get a crashed table.

`--debug-binlog-fsync-sleep`

- Description: `--debug-binlog-fsync-sleep=#` If not set to zero, sets the number of micro-seconds to sleep after running fsync() on the binary log to flush transactions to disk. This can thus be used to artificially increase the perceived cost of such an fsync().

`--debug-crc-break`

- Description: `--debug-crc-break=#` Call my_debug_put_break_here() if crc matches this number (for debug).

`--debug-flush`

- Description: Default debug log with flush after write.

`--debug-no-sync`

- Description: `debug-no-sync[=#]` Disables system sync calls. Only for running tests or debugging!

`--debug-sync-timeout`

- Description: `debug-sync-timeout[=#]` Enable the debug sync facility and optionally specify a default wait timeout in seconds. A zero value keeps the facility disabled.

`--gdb`

- Description: Set up signals usable for debugging.

`--silent-startup`

- Description: Don't print Notes to the error log during startup.

`--sync-sys`

- Description: Enable/disable system sync calls. Syncs should only be turned off (`--disable-sync-sys`) when running tests or debugging! Replaced by debug-no-sync from MariaDB 5.5.
- Removed: MariaDB 5.5

`--thread-alarm`

- Description: Enable/disable system thread alarm calls. Should only be turned off (`--disable-thread-alarm`) when running tests or debugging!

Debugging Options and System Variables

- core-file
- debug
- debug-no-thread-alarm

Other Options

Options that are also system variables are listed after:

`--allow-suspicious-udfs`

- Command line: `--allow-suspicious-udfs`
- Description: Allows use of user-defined functions consisting of only one symbol `x()` without corresponding `x_init()` or `x_deinit()` . That also means that one can load any function from any library, for example

`exit()` from `libc.so`. Not recommended unless you require old UDFs with one symbol that cannot be recompiled. From MariaDB 10.10, available as a system variable as well.

`--bootstrap`

- Command line: `--bootstrap`
- Description: Used by mariadb installation scripts, such as mariadb-install-db to execute SQL scripts before any privilege or system tables exist. Do no use while an existing MariaDB instance is running.

`--chroot`

- Command line: `--chroot=name`
- Description: Chroot mariadbd daemon during startup.

`--des-key-file`

- Command line: `--des-key-file=name`
- Description: Load keys for des_encrypt() and des_decrypt() from the given file.
- Removed: MariaDB 13.0, together with the `DES_ENCRYPT()` / `DES_DECRYPT()` functions and the `FLUSH DES_KEY_FILE` statement.

`--exit-info`

- Command line: `--exit-info[=#]`
- Description: Used for debugging. Use at your own risk.

`--getopt-prefix-matching`

- Command line: `--getopt-prefix-matching={0|1}`
- Description: Makes it possible to disable historical “unambiguous prefix” matching in the command-line option parsing.
- Default: TRUE
- Introduced: MariaDB 10.1.3

`--help`

- Command line: `--help`
- Description: Displays help with many commandline options described, and exits. From MariaDB 11.5, includes deprecation information.

`--log-ddl-recovery`

- Command line: `--log-ddl-recovery=name`
- Description: Path to file used for recovery of DDL statements after a crash.
- Default Value: `ddl-recover.log`
- Introduced: MariaDB 10.6.1

`--log-short-format`

- Command line: `--log-short-format`
- Description: Don’t log extra information to update and slow-query logs.

`--log-slow-file`

- Command line: `--log-slow-file=name`

- Description: Log slow queries to given log file. Defaults logging to hostname-slow.log

`--log-slow-time`

- Command line: `--log-slow-time=#`
- Description: Log all queries that have taken more than long-query-time seconds to execute to the slow query log, if active. The argument will be treated as a decimal value with microsecond precision.

`--log-tc`

- Command line: `--log-tc=name`
- Description: Defines the path to the memory-mapped file-based transaction coordinator log, which is only used if the binary log is disabled. If you have two or more XA-capable storage engines enabled, then a transaction coordinator log must be available. See Transaction Coordinator Log for more information. Also see the log_tc_size system variable and the –tc-heuristic-recover option.
- Default Value: `tc.log`

`--master-connect-retry`

- Command line: `--master-connect-retry=#`
- Description: Deprecated in 5.1.17 and removed in 5.5. The number of seconds the replica thread will sleep before retrying to connect to the master, in case the master goes down or the connection is lost.

`--memlock`

- Command line: `--memlock`
- Description: Lock mariadbd in memory.

`--ndb-use-copying-alter-table`

- Command line: `--ndb-use-copying-alter-table`
- Description: Force ndbcluster to always copy tables at alter table (should only be used if on-line alter table fails).

`--one-thread`

- Command line: `--one-thread`
- Description: (Deprecated): Only use one thread (for debugging under Linux). Use thread-handling=no-threads instead.
- Removed: MariaDB 10.0.4

`--plugin-load`

- Command line: `--plugin-load=name`
- Description: This option can be used to configure the server to load specific plugins. This option uses the following format:
  - Plugins can be specified in the format `name=library`, where `name` is the plugin name and `library` is the plugin library. This format installs a single plugin from the given plugin library.
  - Plugins can also be specified in the format `library`, where `library` is the plugin library. This format installs all plugins from the given plugin library.
  - Multiple plugins can be specified by separating them with semicolons.
- Special care must be taken when specifying the –plugin-load option multiple times, or when specifying both the –plugin-load option and the –plugin-load-add option together. The –plugin-load option resets the plugin load list, and this can cause unexpected problems if you are not aware. The –plugin-load-add option does not reset the

plugin load list, so it is much safer to use. See Plugin Overview: Specifying Multiple Plugin Load Options for more information.

- See Plugin Overview: Installing a Plugin with Plugin Load Options for more information.

`--plugin-load-add`

- Command line: `--plugin-load-add=name`
- Description: This option can be used to configure the server to load specific plugins. This option uses the following format:
  - Plugins can be specified in the format `name=library`, where `name` is the plugin name and `library` is the plugin library. This format installs a single plugin from the given plugin library.
  - Plugins can also be specified in the format `library`, where `library` is the plugin library. This format installs all plugins from the given plugin library.
  - Multiple plugins can be specified by separating them with semicolons.
- Special care must be taken when specifying both the –plugin-load option and the –plugin-load-add option together. The –plugin-load option resets the plugin load list, and this can cause unexpected problems if you are not aware. The –plugin-load-add option does not reset the plugin load list, so it is much safer to use. See Plugin Overview: Specifying Multiple Plugin Load Options for more information.
- See Plugin Overview: Installing a Plugin with Plugin Load Options for more information.

`--port-open-timeout`

- Command line: `--port-open-timeout=#`
- Description: Maximum time in seconds to wait for the port to become free. (Default: No wait).

`--safe-user-create`

- Command line: `--safe-user-create`
- Description: Don't allow new user creation by the user who has no write privileges to the mysql.user table.

`--safemalloc-mem-limit`

- Command line: `--safemalloc-mem-limit=#`
- Description: Simulate memory shortage when compiled with the `--with-debug=full` option.

`--show-slave-auth-info`

- Command line: `--show-slave-auth-info`
- Description: Show user and password in SHOW REPLICA HOSTS (SHOW SLAVE HOSTS) on this primary. Also added as a system variable in MariaDB 12.0.

`--skip-grant-tables`

- Command line: `--skip-grant-tables`
- Description: Start without grant tables. This gives all users FULL ACCESS to all tables, which is useful in case of a lost root password. Use mariadb-admin flush-privileges, mariadb-admin reload or FLUSH PRIVILEGES to resume using the grant tables. From MariaDB 10.10, available as a system variable as well.

Because the Event Scheduler also depends on the grant tables for its functionality, it is automatically disabled when running with `--skip-grant-tables`.

`--skip-host-cache`

- Command line: `--skip-host-cache`

- Description: Don't cache host names.

`--skip-partition`

- Command line: `--skip-partition`, `--disable-partition`
- Description: Disables user-defined partitioning. Previously partitioned tables cannot be accessed or modified. Tables can still be seen with SHOW TABLES or by viewing the INFORMATION_SCHEMA.TABLES table. Tables can be dropped with DROP TABLE, but this only removes .frm files, not the associated .par files, which will need to be removed manually.

`--skip-slave-start`

- Command line: `--skip-slave-start`
- Description: If set, replica is not autostarted. From MariaDB 12.0, server will display in the log if this option is set.

`--skip-ssl`

- Command line: `--skip-ssl`
- Description: Disable TLS connections.

`--skip-symlink`

- Command line: `--skip-symlink`
- Description: Don't allow symlinking of tables. Deprecated and removed in MariaDB 5.5. Use symbolic-links with the `skip` option prefix instead.
- Removed: MariaDB 5.5

`--skip-thread-priority`

- Command line: `--skip-thread-priority`
- Description: Don't give threads different priorities. Deprecated and removed in MariaDB 10.0.
- Removed: MariaDB 10.0

`--sql-bin-update-same`

- Command line: `--sql-bin-update-same=#`
- Description: The update log was deprecated in version 5.0 and replaced by the binary log, so this option did nothing since then. Deprecated and removed in MariaDB 5.5.
- Removed: MariaDB 5.5

`--ssl`

- Command line: `--ssl`
- Description: Enable TLS for connection (automatically enabled with other flags). Disable with '`--skip-ssl`'.

`--stack-trace`

- Command line: `--stack-trace`, `--skip-stack-trace`
- Description: Print a stack trace on failure. Enabled by default, disable with `-skip-stack-trace`.

`--symbolic-links`

- Command line: `--symbolic-links`
- Description: Enables symbolic link support. When set, the have_symlink system variable shows as `YES`. Silently ignored in Windows. Use `--skip-symbolic-links` to disable.

`--tc-heuristic-recover`

- Command line: `--tc-heuristic-recover=name`
- Description: If manual heuristic recovery is needed, this option defines the decision to use in the heuristic recovery process. Manual heuristic recovery may be needed if the transaction coordination log is missing or if it doesn't contain all prepared transactions. This option can be set to `OFF`, `COMMIT`, or `ROLLBACK`. The default is `OFF`. See also the –log-tc server option and the log_tc_size system variable.

`--temp-pool`

- Command line: `--temp-pool`
- Description: Using this option will cause most temporary files created to use a small set of names, rather than a unique name for each new file. This behavior works around a bug in old Linux kernels where the kernel appeared to "leak" memory. In a Docker environment it might look like an unbounded working-set memory growth.
  Defaults to `1` until MariaDB 10.5.6, use `--skip-temp-pool` to disable. Defaults to `0` from MariaDB 10.5.7, as benchmarking shows it causes a heavy mutex contention.

`--test-expect-abort`

- Command line: `--test-expect-abort`
- Description: Expect that server aborts with 'abort'; Don't write out server variables on 'abort'. Useful only for test scripts.

`--test-ignore-wrong-options`

- Command line: `--test-ignore-wrong-options`
- Description: Ignore wrong enums values in command line arguments. Useful only for test scripts.

`--user`

- Command line: `--user=name`
- Description: Run mariadbd daemon as user.

`--verbose`

- Command line: `-v`, `--verbose`
- Description: Used with help option for detailed help.

Other Options and System Variables

- allow-suspicious-udfs
- automatic-sp-privileges
- back-log
- basedir
- check-constraint-checks
- column-compression-threshold
- column-compression-zlib-level
- column-compression-zlib-strategy
- column-compression-zlib-wrap
- completion-type
- connect-timeout
- datadir

- date-format
- datetime-format
- deadlock-search-depth-long
- deadlock-search-depth-short
- deadlock-timeout-long
- deadlock-timeout-short
- default-password-lifetime
- default-regex-flags
- default-storage-engine
- default-table-type
- delay-key-write
- disconnect-on-expired-password
- div-precision-increment
- enable-named-pipe
- encrypt-binlog
- encrypt-tmp-disk-tables
- encrypt-tmp-files
- encryption-algorithm
- engine-condition-pushdown
- eq-range-index-dive-limit
- event-scheduler
- expire-logs-days
- explicit-defaults-for-timestamp
- extra-max-connections
- extra-port
- flush
- flush-time
- ft-boolean-syntax
- ft-max-word-len
- ft-min-word-len
- ft-query-expansion-limit
- ft-stopword-file
- general-log
- general-log-file
- group-concat-max-len
- histogram-size
- histogram-type
- host-cache-size
- idle-readonly-transaction-timeout
- idle-transaction-timeout
- idle-write-transaction-timeout
- ignore-db-dirs
- in-predicate-conversion-threshold
- init-connect
- init-file

- interactive-timeout
- large-pages
- local-infile
- lock-wait-timeout
- log
- log-disabled-statements
- log-error
- log-output
- log-queries-not-using-indexes
- log-slow-admin-statements
- log-slow-always-query-time
- log-slow-disabled-statements
- log-slow-filter
- log-slow-min-examined-row-limit
- log-slow-queries
- log-slow-query
- log-slow-query-file
- log-slow-query-time
- log-slow-rate-limit
- log-slow-slave-statements
- log-slow-verbosity
- log-tc-size
- log-warnings
- long-query-time
- low-priority-updates
- lower-case-table-names
- max-allowed-packet
- max-connections
- max-connect-errors
- max-delayed-threads
- max-digest-length
- max-error-count
- max-length-for-sort-data
- max-long-data-size
- max_open_cursors
- max-password-errors
- max-prepared-stmt-count
- max-recursive-iterations
- max-rowid-filter-size
- max-session-mem-used
- max-sp-recursion-depth
- max-statement-time
- max-tmp-session-space-usage
- max-tmp-tables
- max-tmp-total-space-usage

- max-user-connections
- max-write-lock-count
- metadata-locks-cache-size
- metadata-locks-hash-instances
- metadata-locks-instances
- min-examined-row-limit
- mrr-buffer-size
- multi-range-count
- –mysql56-temporal-format
- net-buffer-length
- net-read-timeout
- net-retry-count
- net-write-timeout
- open-files-limit
- pid-file
- plugin-dir
- plugin-maturity
- port
- preload-buffer-size
- profiling-history-size
- progress-report-time
- proxy-protocol-networks
- query-cache-limit
- query-cache-min-res-unit
- query-cache-strip-comments
- query-cache-wlock-invalidate
- read-rnd-buffer-size
- read-only
- redirect-url
- require-secure-transport
- safe-show-database
- secure-auth
- secure-file-priv
- secure-timestamp
- session-track-schema
- session-track-state-change
- session-track-system-variables
- session-track-transaction-info
- skip-automatic-sp-privileges
- skip-external-locking
- skip-large-pages
- skip-log-error
- skip-name-resolve
- skip-networking
- skip-show-database

- slow-launch-time
- slow-query-log
- slow-query-log-file
- socket
- sort-buffer-size
- sql-if-exists
- sql-mode
- ssl-ca
- ssl-capath
- ssl-cert
- ssl-cipher
- ssl-crl
- ssl-crlpath
- ssl-key
- ssl-passphrase
- standards_compliant_cte
- stored-program-cache
- strict_password_validation
- sync-frm
- system-versioning-alter-history
- system-versioning-asof
- system-versioning-innodb-algorithm-simple
- system-versioning-insert-history
- table-lock-wait-timeout
- tcp-keepalive-interval
- tcp-keepalive-probes
- tcp-keepalive-time
- tcp-nodelay
- thread-cache-size
- thread-concurrency
- thread-handling
- thread-pool-dedicated-listener
- thread-pool-exact-stats
- thread-pool-idle-timeout
- thread-pool-max-threads
- thread-pool-min-threads
- thread-pool-oversubscribe
- thread-pool-prio-kickup-timer
- thread-pool-priority
- thread-pool-size
- thread-pool-stall-limit
- thread-stack
- timed-mutexes
- time-format
- tls-version

- tmpdir
- transaction-isolation
- transaction-alloc-block-size
- transaction-prealloc-size
- transaction-read-only
- updatable-views-with-limit
- userstat
- version
- wait-timeout

Authentication Plugins - Options and System Variables

Authentication Plugin - `ed25519`

The options related to the ed25519 authentication plugin can be found here.

Authentication Plugin - `gssapi`

The system variables related to the gssapi authentication plugin can be found here.

The options related to the gssapi authentication plugin can be found here.

Authentication Plugin - `named_pipe`

The options related to the named_pipe authentication plugin can be found here.

Authentication Plugin - `pam`

The system variables related to the pam authentication plugin can be found here.

The options related to the pam authentication plugin can be found here.

Authentication Plugin - `unix_socket`

The options related to the unix_socket authentication plugin can be found here.

Encryption Plugins - Options and System Variables

Encryption Plugin - `aws_key_management`

The system variables related to the aws_key_management encryption plugin can be found here.

The options elated to the aws_key_management encryption plugin can be found here.

Encryption Plugin - `file_key_management`

The system variables related to the file_key_management encryption plugin can be found here.

The options related to the file_key_management encryption plugin can be found here.

Password Validation Plugins - Options and System Variables

Password Validation Plugin - `simple_password_check`

The system variables related to the c password validation plugin can be found here.

The options related to the simple_password_check password validation plugin can be found here.

Password Validation Plugin - `cracklib_password_check`

The system variables related to the cracklib_password_check password validation plugin can be found here.

The options related to the cracklib_password_check password validation plugin can be found here.

Audit Plugins - Options and System Variables

Audit Plugin - `server_audit`

Options and system variables related to the server_audit audit plugin can be found here.

Audit Plugin - `SQL_ERROR_LOG`

The options and system variables related to the SQL_ERROR_LOG audit plugin can be found here.

Audit Plugin - QUERY_RESPONSE_TIME_AUDIT

The options related to the QUERY_RESPONSE_TIME_AUDIT audit plugin can be found here.

Daemon Plugins - Options and System Variables

Daemon Plugin - `handlersocket`

The options for the HandlerSocket plugin are all described on the HandlerSocket Configuration Option page.

Information Schema Plugins - Options and System Variables

Information Schema Plugin - `DISKS`

The options related to the DISKS information schema plugin can be found here.

Information Schema Plugin - `feedback`

The system variables related to the feedback plugin can be found here.

The options related to the feedback plugin can be found here.

Information Schema Plugin - `LOCALES`

The options related to the LOCALES information schema plugin can be found here.

Information Schema Plugin - `METADATA_LOCK_INFO`

The options related to the METADATA_LOCK_INFO information schema plugin can be found here.

Information Schema Plugin - `QUERY_CACHE_INFO`

The options related to the QUERY_CACHE_INFO information schema plugin can be found here.

Information Schema Plugin - `QUERY_RESPONSE_TIME`

The system variables related to the QUERY_RESPONSE_TIME information schema plugin can be found here.

The options related to the QUERY_RESPONSE_TIME information schema plugin can be found here.

Information Schema Plugin - `user_variables`

The options related to the user_variables information schema plugin can be found here.

Information Schema Plugin - `WSREP_MEMBERSHIP`

The options related to the WSREP_MEMBERSHIP information schema plugin can be found here.

Information Schema Plugin - `WSREP_STATUS`

The options related to the WSREP_STATUS information schema plugin can be found here.

Replication Plugins - Options and System Variables

Replication Plugin - `rpl_semi_sync_master`

The system variables related to the rpl_semi_sync_master replication plugin can be found here.

The options related to the rpl_semi_sync_master replication plugin can be found here.

Replication Plugin - `rpl_semi_sync_slave`

The system variables related to the rpl_semi_sync_slave replication plugin can be found here.

The options related to the rpl_semi_sync_slave replication plugin can be found here.

Default Values

You can verify the default values for an option by doing:

```
mariadbd --no-defaults --help --verbose
```

This page is licensed: GPLv2

mariadbd

> **Note:**
>
> Previously, the client was called `mysqld`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

See mariadbd-options for details.

This page is licensed: CC BY-SA / Gnu FDL

mysql.server

The mysql.server startup script is in MariaDB distributions on Linux and Unix. It is a wrapper that works as a standard sysVinit script. However, it can be used independently of sysVinit as a regular `sh` script. The script starts the mariadbd server process by first changing its current working directory to the MariaDB install directory and then starting mariadbd-safe. The script requires the standard sysVinit arguments, such as `start`, `stop`, `restart`, and `status`. For example:

```
mysql.server start
mysql.server restart
mysql.server stop
mysql.server status
```

It can be used on systems such as Linux, Solaris, and Mac OS X.

The `mysql.server` script starts mariadbd by first changing to the MariaDB install directory and then calling mariadbd-safe.

Using mysql.server

The command to use `mysql.server` and the general syntax is:

```
mysql.server [ start | stop | restart | status ] <options> <mariadbd_options>
```

Options

If an unknown option is provided to `mariadbd-safe` on the command-line, it is passed to `mariadbd-safe`.

`mysql.server` supports the following options:

Option	Description
–basedir=path	The path to the MariaDB installation directory.
–datadir=path	The path to the MariaDB data directory.
–pid-file=file_name	The path name of the file in which the server should write its process ID. If not provided, the default, host_name.pid is used.
–service-startup-timeout=file_name	How long in seconds to wait for confirmation of server startup. If the server does not start within this time, mysql.server exits with an error. The default value is 900. A value of 0 means not to wait at all for startup. Negative values mean to wait forever (no timeout).
–use-mysqld_safe	Use mariadbd-safe to start the server. This is the default.
–use-manager	Use Instance Manager to start the server.
–user=user_name	The login user name to use for running mariadbd.

Option Files

In addition to reading options from the command-line, `mysql.server` can also read options from option files.

The following options relate to how MariaDB command-line tools handles option files. They must be given as the first argument on the command-line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.

Option Groups

`mysql.server` reads options from the following option groups from option files:

Group	Description
[mysql.server]	Options read by mysql.server, which includes both MariaDB Server and MySQL Server.

`mysql.server` also reads options from the following server option groups from option files:

Group	Description
[mysqld]	Options read by mysqld, which includes both MariaDB Server and MySQL Server.
[server]	Options read by MariaDB Server.
[mysqld-X.Y]	Options read by a specific version of mysqld, which includes both MariaDB Server and MySQL Server. For example, [mysqld-5.5].

Group	Description
[mariadbd]	Options read by MariaDB Server.
[mariadbd-X.Y]	Options read by a specific version of MariaDB Server.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[galera]	Options read by a galera-capable MariaDB Server. Available on systems compiled with Galera support.

Customizing mysql.server

If you have installed MariaDB to a non-standard location, then you may need to edit the `mysql.server` script to get it to work right.

If you do not want to edit the `mysql.server` script itself, then `mysql.server` also sources a few other `sh` scripts. These files can be used to set any variables that might be needed to make the script work in your specific environment. The files are:

- `/etc/default/mysql`
- `/etc/sysconfig/mysql`
- `/etc/conf.d/mysql`

Installed Locations

`mysql.server` can be found in the `support-files` directory under your MariaDB installation directory or in a MariaDB source distribution.

Installed SysVinit Locations

On systems that use sysVinit, `mysql.server` may also be installed in other locations and with other names.

If you installed MariaDB on Linux using RPMs, the `mysql.server` script is installed in the `/etc/init.d` directory with the name `mysql`. You need not install it manually.

Manually Installing with SysVinit

If you install MariaDB from source or from a binary tarball that does not install mysql.server automatically, and if you are on a system that uses sysVinit, then you can manually install `mysql.server` with sysVinit. This is usually done by copying it to `/etc/init.d/` and then creating specially named symlinks in the appropriate `/etc/rcX.d/` directories (where 'X' is a number between 0 and 6).

In the examples below we will follow the historical convention of renaming the `mysql.server` script to '`mysql`' when we copy it to `/etc/init.d/`.

The first step for most Linux distributions is to copy the `mysql.server` script to `/etc/init.d/` and make it executable:

```
cd /path/to/your/mariadb-version/support-files/
cp mysql.server /etc/init.d/mysql
chmod +x /etc/init.d/mysql
```

Now all that is needed is to create the specially-named symlinks. On both RPM and Debian-based Linux distributions there are tools which do this for you. Consult your distribution's documentation if neither of these work for you and follow their instructions for generating the symlinks or creating them manually.

On RPM-based distributions (like Fedora and CentOS), you use `chkconfig`:

```
chkconfig --add mysql
chkconfig --level 345 mysql on
```

On Debian-based distributions you use `update-rc.d`:

```
update-rc.d mysql defaults
```

On FreeBSD, the location for startup scripts is `/usr/local/etc/rc.d/` and when you copy the `mysql.server` script there you should rename it so that it matches the `*.sh` pattern, like so:

```
cd /path/to/your/mariadb/support-files/
cp mysql.server /usr/local/etc/rc.d/mysql.server.sh
```

As stated above, consult your distribution's documentation for more information on starting services like MariaDB at system startup.

See mariadbd startup options for information on configuration options for `mariadbd`.

This page is licensed: GPLv2

Running MariaDB From the Build Directory

You can run mariadbd directly from the build directory (without doing `make install`).

Starting mariadbd After Build on Windows

On Windows, the data directory is produced during the build.

The simplest way to start database from the command line is:

1. Go to the directory where mariadbd.exe is located (subdirectory `sql\Debug` or `sql\Relwithdebinfo` of the build directory)
2. From here, execute:

```
mariadbd.exe --console
```

Pass other server parameters on the command line, or store them in a `my.ini` configuration file and pass `--defaults-file=path\to\my.ini`.

The default search path on Windows for the `my.ini` file is:

- `GetSystemWindowsDirectory()`
- `GetWindowsDirectory()`
- `C:\`
- Directory where the executable is located

Starting mariadbd After Build on Unix

Copy the following to your '`~/.my.cnf`' file.

There are two lines you have to edit: '`datadir=`' and '`language=`'. Be sure to change them to match your environment.

```
# Example MariadB config file.
# You can copy this to one of:
# /etc/my.cnf to set global options,
# /mysql-data-dir/my.cnf to get server specific options or
# ~/my.cnf for user specific options.
#
# One can use all long options that the program supports.
# Run the program with --help to get a list of available options

# This will be passed to all MariaDB clients
[client]
#password=my_password
#port=3306
#socket=/tmp/mysql.sock

# Here is entries for some specific programs
# The following values assume you have at least 32M ram

# The mariadb server  (both [mysqld] and [mariadb] works here)
[mariadb]
#port=3306
#socket=/tmp/mysql.sock

# The following three entries caused mysqld 10.0.1-MariaDB (and possibly other versions) to abort...
# skip-locking
# set-variable  = key_buffer=16M

loose-innodb_data_file_path = ibdata1:1000M
loose-mutex-deadlock-detector
gdb

######### Fix the two following paths

# Where you want to have your database
datadir=/path/to/data/dir

# Where you have your mysql/MariaDB source + sql/share/english
language=/path/to/src/dir/sql/share/english

########## One can also have a different path for different versions, to simplify development.

[mariadb-10.1]
lc-messages-dir=/my/maria-10.1/sql/share

[mariadb-10.2]
lc-messages-dir=/my/maria-10.2/sql/share

[mysqldump]
quick
set-variable = max_allowed_packet=16M

[mysql]
no-auto-rehash

[myisamchk]
set-variable= key_buffer=128M
```

With the above file in place, go to your MariaDB source directory and execute:

```
./scripts/mariadb-install-db --srcdir=$PWD --datadir=/path/to/data/dir
```

- `$PWD` is the environment variable that points to your current directory. If you added `datadir` to your `my.cnf`, you don't have to specify this option.

Now you can start `mariadbd` in the debugger:

```
cd sql
ddd ./mariadbd &
```

Or start `mariadbd` on its own:

```
cd sql
./mariadbd &
```

After starting up `mariadbd` using one of the above methods (with the debugger or without), launch the client (as root if you don't have any users setup yet).

```
../client/mariadb
```

Using a Storage Engine Plugin

The simplest case is to compile the storage engine into MariaDB:

```
cmake -DWITH_PLUGIN_<plugin_name>=1` .
```

Another option is to point `mariadbd` to the storage engine directory:

```
./mariadbd --plugin-dir={build-dir-path}/storage/connect/.libs
```

This page is licensed: CC BY-SA / Gnu FDL

Running Multiple MariaDB Server Processes

It is possible to run multiple MariaDB Server processes on the same server, but there are certain things that need to be kept in mind. This page will go over some of those things.

Configuring Multiple MariaDB Server Processes

If multiple MariaDB Server process are running on the same server, then at minimum, you will need to ensure that the different instances do not use the same datadir, port, and socket. The following example shows these options set in an option file:

```
[client]
# TCP port to use to connect to mariadbd server
port=3306
# Socket to use to connect to mariadbd server
socket=/tmp/mysql.sock
[mariadb]
# TCP port to make available for clients
port=3306
#  Socket to make available for clients
socket=/tmp/mysql.sock
# Where MariaDB should store all its data
datadir=/usr/local/mysql/data
```

The above values are the defaults. If you would like to run multiple MariaDB Server instances on the same server, then you will need to set unique values for each instance.

There may be additional options that also need to be changed for each instance. Take a look at the full list of options for mariadbd.

To see the current values set for an instance, see Checking Program Options for how to do so.

To list the default values, check the end of the output of this command:

```
mariadbd --help --verbose
```

Starting Multiple MariaDB Server Processes

There are several different methods to start or stop the MariaDB Server process. There are two primary categories that most of these methods fall into: starting the process with the help of a service manager, and starting the process manually. See Starting and Stopping MariaDB for more information.

If you want to run different MariaDB versions on the same machine, using binary tarballs, Docker or using Virtual Machines (VMs) are the recommended ways. A binary tarball uses least resources, Docker a bit more and a VM uses most resources.

Service Managers

sysVinit and systemd are the most common Linux service managers. launchd is used in MacOS X. Upstart is a less common service manager.

Systemd

RHEL/CentOS 7 and above, Debian 8 Jessie and above, and Ubuntu 15.04 and above use systemd by default.

For information on how to start and stop multiple MariaDB Server processes on the same server with this service manager, see systemd: Interacting with Multiple MariaDB Server Processes.

Starting the Server Process Manually

mariadbd

mariadbd is the actual MariaDB Server binary. It can be started manually on its own.

If you want to force each instance to read only a single option file, then you can use the –defaults-file option:

```
mariadbd --defaults-file=/etc/my_instance1.cnf
```

mariadbd-safe

mariadbd-safe is a wrapper that can be used to start the mariadbd server process. The script has some built-in safeguards, such as automatically restarting the server process if it dies. See mariadbd-safe for more information.

If you want to force each instance to read only a single option file, then you can use the –defaults-file option:

```
mariadbd-safe --defaults-file=/etc/my_instance1.cnf
```

mariadbd-multi

mariadbd-multi is a wrapper that can be used to start the mariadbd server process if you plan to run multiple server processes on the same host. See mariadbd-multi for more information.

Other Options

In some cases, there may be easier ways to run multiple MariaDB Server instances on the same server, such as:

- Starting multiple Docker containers.
- Using dbdeployer (no longer maintained).

This page is licensed: CC BY-SA / Gnu FDL

Specifying Permissions for Schema (Data) Directories and Tables

Default File Permissions

By default MariaDB uses the following permissions for files and directories:

Object Type	Default Mode	Default Permissions
Files	0660	-rw-rw—-
Directories	0700	drwx——

Configuring File Permissions with Environment Variables

You can configure MariaDB to use different permissions for files and directories by setting the following environment variables before you start the server:

Object Type	Environment Variable
Files	UMASK
Directories	UMASK_DIR

In other words, if you would run the following in a shell:

```
export UMASK=0640
export UMASK_DIR=0750
```

These environment variables do not set the umask. They set the default file system permissions. See MDEV-23058 for more information.

Configuring File Permissions With systemd

If your server is started by systemd, then there is a specific way to configure the umask. See Systemd: Configuring the umask for more information.

This page is licensed: CC BY-SA / Gnu FDL

Switching Between Different Installed MariaDB Versions

This article is about managing many different installed MariaDB versions and running them one at a time. This is useful when doing benchmarking, testing, or for when developing different MariaDB versions.

This is most easily done using the tar files from mariadb.org/download/.

Stopping a Preinstalled MySQL/MariaDB From Interfering With Your Tests

If MySQL/MariaDB is already installed and running, you have two options:

1. Use test MariaDB servers with a different port & socket.

- In this case you are probably best off creating a specific section for MariaDB in your `~/.my.cnf` file.

2. Stop mysqld with `/etc/rc.d/mysql stop` or `mariadb-admin shutdown`.

Note that you don't have to uninstall or otherwise remove MySQL!

How to Create a Binary Distribution (tar File)

Here is a short description of how to generate a tar file from a source distribution. If you have downloaded a binary tar file, you can skip this section.

The steps to create a binary tar file are:

- Decide where to put the source. A good place is under `/usr/local/src/mariadb-5.#`.
- Get the source
- Compile the source
- Create the binary tar ball.

You will then be left with a tar file named something like: `mariadb-11.0.1-MariaDB-linux-x86_64.tar.gz`

Creating a Directory Structure for the Different Installations

Install the binary tar files under `/usr/local/` with the following directory names (one for each MariaDB version you want to use), for example:

- `mariadb-10.5`
- `mariadb-10.6`
- `mariadb-10.11`
- `mariadb-11.0`
- `mariadb-11.1`

The above assumes you are just testing major versions of MariaDB. If you are testing specific versions, use directory names like `mariadb-11.0.1`

With the directories in place, create a symlink named `mariadb` which points at the `mariadb-XXX` directory you are currently testing. When you want to switch to testing a different version, just update the symlink.

Example:

```
cd /usr/local
tar xfz /tmp/mariadb-11.0.1-linux-systemd-x86_64.tar.gz
mv -vi mariadb-11.0.1-MariaDB-systemd-linux-x86_64 mariadb-11.0
ln -vs mariadb-11.0 mariadb
```

Setting Up the Data Directory

When setting up the data directory, you have the option of either using a shared database directory or creating a unique database directory for each server version. For testing, a common directory is probably easiest. Note that you can only have one `mysqld` server running against one data directory.

Setting Up a Common Data Directory

The steps are:

1. Create the `mysql` system user if you don't have it already!
   (On Linux you do it with the `useradd` command).
2. Create the directory (we call it `mariadb-data` in the example below) or add a symlink to a directory which is in some other place.
3. Create the `mysql` permission tables with mariadb-install-db.

```
cd /usr/local/
mkdir mariadb-data
cd mariadb
./bin/mariadb-install-db --no-defaults --datadir=/usr/local/mariadb-data
chown -R mysql mariadb-data mariadb-data/*
```

The reason to use `--no-defaults` is to ensure that we don't inherit incorrect options from some old `my.cnf`.

Setting Up Different Data Directories

To create a different `data` directories for each installation:

```
cd mariadb
./scripts/mariadb-install-db --no-defaults
chown -R mysql mariadb-data mariadb-data/*
```

This will create a directory `data` inside the current directory.

If you want to use another disk, you should run this command:

```
cd mariadb
ln -s path-to-empty-directory-for-data data
./scripts/mariadb-install-db --no-defaults --datadir=./data
chown -R mysql mariadb-data mariadb-data/*
```

Running a MariaDB Server

The normal steps are:

```
rm mariadb
ln -s mariadb-# mariadb
cd mariadb
./bin/mysqld_safe --no-defaults --datadir=/usr/local/mariadb-data &
```

Setting Up a .my.cnf file for Running Multiple MariaDB Versions

If you are going to start/stop MariaDB a lot of times, you should create a `~/.my.cnf` file for the common options you are using.

The following example shows how to use a non-standard TCP-port and socket (to not interfere with a main MySQL/MariaDB server) and how to setup different options for each main server:

```
[client-server]
socket=/tmp/mysql.sock
port=3306
[mysqld]
datadir=/usr/local/mariadb-data

[mariadb-11.0]
# Options for MariaDB 11.0
[mariadb-11.1]
# Options for MariaDB 11.1
```

If you create an `~/.my.cnf` file, you should start `mysqld` with `--defaults-file=~/.my.cnf` instead of `--no-defaults` in the examples above.

This page is licensed: CC BY-SA / Gnu FDL

What to Do if MariaDB Doesn't Start

This page helps troubleshoot some of the more common reasons for MariaDB not starting, and provide solutions.

If you have tried everything here and still need help, you can ask for help on IRC or on the forums - see Where to find other MariaDB users and developers - or ask a question at the Starting and Stopping MariaDB page.

The Error Log and the Data Directory

The reason for the failure will almost certainly be written in the error log and, if you are starting MariaDB manually, to the console. By default, the error log is named *host-name*.err and is written to the data directory.

Common Locations:

- `/var/log/`
- `/var/log/mysql`
- `C:\Program Files\MariaDB x.y\data` (x.y refers to the version number)
- `C:\Program Files (x86)\MariaDB x.y\data` (32bit version on 64bit Windows)

It's also possible that the error log has been explicitly written to another location. This is often done by changing the datadir or log_error system variables in an option file. See the Option Files below for more information about that.

A quick way to get the values of these system variables is to execute the following commands:

```
mariadbd --help --verbose | grep 'log-error' | tail -1
mariadbd --help --verbose | grep 'datadir' | tail -1
```

Option Files

Another kind of file to consider when troubleshooting is option files. The default option file is called `my.cnf`. Option files contain configuration options, such as the location of the data directory mentioned above. If you're unsure where the option file is located, see Configuring MariaDB with Option Files: Default Option File Locations for information on the default locations.

You can check which configuration options MariaDB server will use from its option files by executing the following command:

```
mariadbd --print-defaults
```

You can also check by executing the following command:

```
my_print_defaults --mysqld
```

See Configuring MariaDB with Option Files: Checking Program Options for more information on checking configuration options.

Invalid Option or Option Value

Another potential reason for a startup failure is that an option file contains an invalid option or an invalid option value. In those cases, the error log should contain an error similar to this:

```
140514 12:19:37 [ERROR] /usr/local/mysql/bin/mariadbd: unknown variable 'option=value'
```

This is more likely to happen when you upgrade to a new version of MariaDB. In most cases, the option file from the old version of MariaDB will work just fine with the new version. However, occasionally, options are removed in new versions of MariaDB, or the valid values for options are changed in new versions of MariaDB. Therefore, it's possible for an option file to stop working after an upgrade.

Also, remember that option names are case sensitive.

Examine the specifics of the error. Possible fixes are usually one of the following:

- If the option is completely invalid, then remove it from the option file.
- If the option's name has changed, then fix the name.
- If the option's valid values have changed, then change the option's value to a valid one.
- If the problem is caused by a simple typo, then fix the typo.

Can't Open Privilege Tables

It is possible to see errors similar to the following:

```
System error 1067 has occurred.
Fatal error: Can't open privilege tables: Table 'mysql.host' doesn't exist
```

If errors like this occur, then critical system tables are either missing or are in the wrong location. The above error is quite common after an upgrade if the option files set the basedir or datadir to a non-standard location, but the new server is using the default location. Therefore, make sure that the basedir and datadir variables are correctly set.

If you're unsure where the option file is located, see Configuring MariaDB with Option Files: Default Option File Locations for information on the default locations.

If the system tables really do not exist, then you may need to create them with mariadb-install-db. See Installing System Tables (mariadb-install-db) for more information.

Can't Create Test File

One of the first tests on startup is to check whether MariaDB can write to the data directory. When this fails, it will log an error like this:

```
May 13 10:24:28 mariadb3 mariadbd[19221]: 2019-05-13 10:24:28 0 [Warning] Can't create test file
/usr/local/data/mariadb/mariadb3.lower-test
May 13 10:24:28 mariadb3 maridbd[19221]: 2019-05-13 10:24:28 0 [ERROR] Aborting
```

This is usually a permission error on the directory in which this file is being written. Ensure that the entire datadir is owned by the user running `mariadbd`, usually `mysql`. Ensure that directories have the "x" (execute) directory permissions for the owner. Ensure that all the parent directories of the datadir upwards have "x" (execute) permissions for all (`user`, `group`, and `other`).

Once this is checked, look at the systemd and SELinux documentation below, or AppArmor.

Can't Lock Aria Control File

On starting MariaDB, the `aria_log_control` file is locked. If a lock cannot be obtained, it will log an error like this:

```
2023-05-01 16:27:03 0 [ERROR] mariadbd: Can't lock aria control file '/var/lib/mysql/aria_log_control'
for exclusive use, error: 11. Will retry for 30 seconds
```

This almost always happens because there is already an existing MariaDB service running on this data directory. Recommend aborting this startup and looking closely for the other MariaDB instance.

A less likely case is that file locking is not available, which might occur on an NFS-mounted data directory without proper locking support.

Unable to Lock ./ibdata1 Error 11

Like the above for the Aria Control File, this is attempting to exclusively lock the `ibdata1` InnoDB system tablespace. Error 11 corresponds to the system error "OS error code 11: Resource temporarily unavailable," meaning the lock cannot be created.

```
2023-05-01 16:27:34 0 [ERROR] InnoDB: Unable to lock ./ibdata1 error: 11
2023-05-01 16:27:34 0 [Note] InnoDB: Check that you do not already have another mariadbd process using
the same InnoDB data or log files.
2023-05-01 16:27:34 0 [ERROR] InnoDB: Plugin initialization aborted with error Generic error
2023-05-01 16:27:35 0 [Note] InnoDB: Starting shutdown...
```

Like the above, this is an indication that a second MariaDB instance is already running on the data directory.

InnoDB

InnoDB is probably the MariaDB component that most frequently causes a crash. In the error log, lines containing InnoDB messages generally start with “InnoDB:”.

Cannot Allocate Memory for the InnoDB Buffer Pool

In a typical installation on a dedicated server, at least 70% of your memory should be assigned to the InnoDB buffer pool; sometimes it can even reach 85%. But be very careful: don’t assign to the buffer pool more memory than it can allocate. If it cannot allocate memory, InnoDB will use the disk’s swap area, which is very bad for performance. If swapping is disabled or the swap area is not big enough, InnoDB will crash. In this case, MariaDB will probably try to restart several times, and each time it will log a message like this:

```
140124 17:29:01 InnoDB: Fatal error: cannot allocate memory for the buffer pool
```

In that case, you will need to add more memory to your server/VM or decrease the value of the innodb_buffer_pool_size variable.

Remember that the buffer pool will slightly exceed that limit. Also, remember that MariaDB also needs to allocate memory for other storage engines and several per-connection buffers. The operating system also needs memory.

InnoDB Table Corruption

By default, InnoDB deliberately crashes the server when it detects table corruption. The reason for this behavior is preventing corruption propagation. However, in some situations, server availability is more important than data integrity. For this reason, we can avoid these crashes by changing the value of innodb_corrupt_table_action to ‘warn’.

If InnoDB crashes the server after detecting data corruption, it writes a detailed message in the error log. The first lines are similar to the following:

```
InnoDB: Database page corruption on disk or a failed
InnoDB: file read of page 7.
InnoDB: You may have to recover from a backup.
```

Generally, it is still possible to recover most of the corrupted data. To do so, restart the server in InnoDB recovery mode and try to extract the data that you want to backup. You can save them in a CSV file or in a non-InnoDB table. Then, restart the server in normal mode and restore the data.

MyISAM

Most tables in the mysql database are MyISAM tables. These tables are necessary for MariaDB to properly work or even start.

A MariaDB crash could cause system table corruption. With the default settings, MariaDB will simply not start if the system tables are corrupted. With myisam_recover_options, we can force MyISAM to repair damaged tables.

systemd

If you are using systemd, then there are a few relevant notes about startup failures:

- If MariaDB is configured to access files under `/home`, `/root`, or `/run/user`, then the default systemd unit file will prevent access to these directories with a `Permission Denied` error. This happens because the unit file sets ProtectHome=true. See Systemd: Configuring Access to Home Directories for information on how to work around this.
- The default systemd unit file also sets ProtectSystem=full, which places restrictions on writing to a few other directories. Overwriting this with `ProtectSystem=off` in the same way as above will restore access to these

directories.

- If MariaDB takes longer than 90 seconds to start, then the default systemd unit file will cause it to fail with an error. This happens because the default value for the TimeoutStartSec option is 90 seconds. See Systemd: Configuring the Systemd Service Timeout for information on how to work around this.
- The systemd journal may also contain useful information about startup failures. See Systemd: Systemd Journal for more information.

See systemd documentation for further information on systemd configuration.

SELinux

Security-Enhanced Linux (SELinux) is a Linux kernel module that provides a framework for configuring a mandatory access control (MAC) system for many resources on the system. It is enabled by default on some Linux distributions, including RHEL, CentOS, Fedora, and other similar Linux distributions. SELinux prevents programs from accessing files, directories, or ports unless it is configured to access those resources.

You might need to troubleshoot SELinux-related issues in cases such as:

- MariaDB is using a non-default port.
- MariaDB is reading from or writing to some files (datadir, log files, option files, etc.) located at non-default paths.
- MariaDB is using a plugin that requires access to resources that default installations do not use.

Setting SELinux state to `permissive` is a common way to investigate what is going wrong while allowing MariaDB to function normally. `permissive` is supposed to produce a log entry every time it should block a resource access, without actually blocking it. However, there are situations when SELinux blocks resource accesses even in `permissive` mode.

See SELinux for more information.

AppArmor

Add the following to `/etc/apparmor.d/tunables/alias` if you have moved the datadir:

```
alias /var/lib/mysql/ -> /data/mariadb/,
```

The restart AppArmor:

```
sudo systemctl restart apparmor
```

AppArmor on newer Ubuntu and Debian releases

In Ubuntu 26.04 “Resolute Raccoon”, Debian 14 “Forky”, and newer releases of these distributions, the MariaDB packages ship with a **custom AppArmor profile** which is enabled by default.

In the event of unexplained errors at startup or runtime, check for potential AppArmor denials with `aa-status | grep mariadb` or `sudo dmesg | grep -i apparmor`.

To disable enforcement, run `aa-complain /etc/apparmor.d/mariadbd`.

To add local overrides, put them in `/etc/apparmor.d/local/mariadbd`. For more information, see /usr/share/doc/mariadb-server/NEWS.Debian.gz.

This page is licensed: CC BY-SA / Gnu FDL

Why to Automate MariaDB Deployments and Management

MariaDB includes a powerful configuration system. This is enough when we need to deploy a single MariaDB instance, or a small number of instances. But many modern organisations have many database servers. Deploying

and upgrading them manually could require too much time, and would be error-prone.

Infrastructure as Code

Several tools exist to deploy and manage several servers automatically. These tools operate at a higher level, and execute tasks like installing MariaDB, running queries, or generating new configuration files based on a template. Instead of upgrading servers manually, users can launch a command to upgrade a group of servers, and the automation software will run the necessary tasks.

Servers can be described in a code repository. This description can include MariaDB version, its configuration, users, backup jobs, and so on. This code is human-readable, and can serve as a documentation of which servers exist and how they are configured. The code is typically versioned in a repository, to allow collaborative development and track the changes that occurred over time. This is a paradigm called **Infrastructure as Code**.

Automation code is high-level and one usually doesn't care how operations are implemented. Their implementation is delegated to modules that handle specific components of the infrastructure. For example a module could equally work with apt and yum package managers. Other modules can implement operations for a specific cloud vendor, so we declare we want a snapshot to be done, but we don't need to write the commands to make it happen. For special cases, it is of course possible to write Bash commands, or scripts in every language, and declare that they must be run.

Manual interventions on the servers will still be possible. This is useful, for example, to investigate performance problems. But it is important to leave the servers in the state that is described by the code.

This code is not something you write once and never touch again. It is periodically necessary to modify infrastructures to update some software, add new replicas, and so on. Once the base code is in place, making such changes is often trivial and potentially it can be done in minutes.

Automated Failover

Once replication is in place, two important aspects to automate are load balancing and failover.

Proxies can implement load balancing, redirecting the queries they receive to different server, trying to distribute the load equally. They can also monitor that MariaDB servers are running and in good health, thus avoiding sending queries to a server that is down or struggling.

However, this does not solve the problem with replication: if a primary server crashes, its replicas should point to another server. Usually this means that an existing replica is promoted to a master. This kind of changes are possible thanks to MariaDB GTID.

One can promote a replica to a primary by making change to existing automation code. This is typically simple and relatively quick to do for a human operator. But this operation takes time, and in the meanwhile the service could be down.

Automating failover will minimise the time to recover. A way to do it is to use Orchestrator, a tool that can automatically promote a replica to a primary. The choice of the replica to promote is done in a smart way, keeping into account things like the servers versions and the binary log format.

Resources

- Continuous configuration automation on Wikipedia.
- Infrastructure as code on Wikipedia.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Comparison Between Automation Systems

This page compares the automation systems that are covered by this section of the MariaDB documentation. More information about these systems are presented in the relevant pages, and more systems may be added in the future.

Code Structure Differences

Different automation systems provide different ways to describe our infrastructure. Understanding how they work is the first step to evaluate them and choose one for our organization.

ANSIBLE

Ansible code consists of the following components:

- An inventory determines which hosts Ansible should be able to deploy. Each host may belong to one or more groups. Groups may have children, forming a hierarchy. This is useful because it allows us to deploy on a group, or to assign variables to a group.
- A role describes the state that a host, or group of hosts, should reach after a deploy.
- A play associates hosts or groups to their roles. Each role/group can have more than one role.
- A role consists of a list of tasks. Despite its name a task is not necessarily something to do, but something that must exist in a certain state.
- Tasks can use variables. They can affect how a task is executed (for example a variable could be a file name), or even whether a task is executed or not. Variables exist at role, group or host level. Variables can also be passed by the user when a play is applied.
- Playbooks are the code that is used to define tasks and variables.
- Facts are data that Ansible retrieves from remote hosts before deploying. This is a very important step, because facts may determine which tasks are executed or how they are executed. Facts include, for example, the operating system family or its version. A playbook sees facts as pre-set variables.
- Modules implement actions that tasks can use. Action examples are file (to declare that files and directories must exist) or mysql_variables (to declare MySQL/MariaDB variables that need to be set).

See Ansible Overview - concepts for more details and an example.

PUPPET

Puppet code consists of the following components:

- An inventory file defines a set of groups and their targets (the members of a group). plugins can be used to retrieve groups and target dynamically, so they are equivalent to Ansible dynamic inventories.
- A manifest is a file that describes a configuration.
- A resource is a component that should run on a server. For example, “file” and “service” are existing support types.
- An attribute relates to a resource and affects the way it is applied. For example, a resource of type “file” can have attributes like “owner” and “mode”.
- A class groups resources and variables, describing a logical part of server configuration. A class can be associated to several servers. A class is part of a manifest.
- A module is a set of manifests and describes an infrastructure or a part of it.
- Classes can have typed parameters that affect how they are applied.
- Properties are variables that are read from the remote server, and cannot be arbitrarily assigned.
- Facts are pre-set variables collected by Puppet before applying or compiling a manifest.

Architectural Differences

The architecture of the various systems is different. Their architectures determine how a deploy physically works, and what is needed to be able to deploy.

ANSIBLE

Ansible architecture is simple. Ansible can run from any host, and can apply its playbooks on remote hosts. To do this, it runs commands via SSH. In practice, in most cases the commands will be run as superuser via `sudo`, though this is not always necessary.

Inventories can be dynamic. In this case, when we apply a playbook Ansible connects to remote services to discover hosts.

Ansible playbooks are applied via the `ansible-playbook` binary. Changes to playbooks are only applied when we perform this operation.

To recap, Ansible does not need to be installed on the server is administers. It needs an SSH access, and normally its user needs to be able to run `sudo`. It is also possible to configure a dynamic inventory, and a remote service to be used for this purpose.

PUPPET

Puppet Architecture

Puppet supports two types of architecture: agent-master or standalone. The agent-master architecture is recommended by Puppet Labs, and it is the most popular among Puppet users. For this reason, those who prefer a standalone architecture tend to prefer Ansible.

Agent-Master Architecture

When this architecture is chosen, manifests are sent to the **Puppet master**. There can be more than one master, for high availability reasons. All target hosts run a **Puppet agent**. Normally this is a service that automatically starts at system boot. The agent contacts a master at a given interval. It sends facts, and uses them to compile a **catalog** from the manifests. A catalog is a description of what exactly an individual server should run. The agent receives the catalog and checks if there are differences between its current configuration and the catalog. If differences are found, the agent applies the relevant parts of the catalog.

An optional component is **PuppetDB**. This is a central place where some data are stored, including manifests, retrieved facts and logs. PuppetDB is based on PostgreSQL and there are no plans to support MariaDB or other DBMSs.

If a manual change is made to a remove server, it will likely be overwritten the next time Puppet agent runs. To avoid this, the Puppet agent service can be stopped.

Standalone Architecture

As mentioned, this architecture is not recommended by Puppet Labs nor popular amongst Puppet users. It is similar to Ansible architecture.

Users can apply manifests from any host with Puppet installed. This could be their laptop but, in order to emulate the behavior of an agent-master architecture, normally Puppet runs on a dedicated node as a cronjob. The **Puppet apply** application will require facts from remote hosts, it will compile a catalog for each host, will check which parts of it need to be applied, and will apply them remotely.

If a manual change is made to a remove server, it will be overwritten the next time Puppet apply runs. To avoid this, comment out any cron job running Puppet apply, or comment out the target server in the inventory.

Inventory

> As mentioned, Puppet supports plugins to retrieve the inventory dynamically from remote services. In an agent-master architecture, one has to make sure that each target host has access to these services. In a standalone architecture, one has to make sure that the hosts running Puppet apply have access to these services.

Storing Secrets

Often our automation repositories need to contain secrets, like MariaDB user passwords or private keys for SSH authentication.

Both Ansible and Puppet support integration with secret stores, like Hashicorp Vault.

> **ANSIBLE**
>
> Ansible allows encrypting secrets in playbooks and decrypting them during execution using ansible-vault. This implies a minimal effort to handle secrets. However, it is not the most secure way to store secrets. The passwords to disclose certain secrets need to be shared with the users who have the right to use them. Also, brute force attacks are possible.
>
> **PUPPET**
>
> For Puppet integration, see Integrations with secret stores.

Ecosystems and Communities

Automation software communities are very important, because they make available a wide variety of modules to handle specific software.

> **ANSIBLE**
>
> Ansible is open source, released under the terms of the GNU GPL. It is produced by RedHat. RedHat has a page about Red Hat Ansible Automation Platform Partners, who can provide support and consulting.
>
> Ansible Galaxy is a big repository of Ansible roles produced by both the vendor and the community. Ansible comes with `ansible-galaxy`, a tool that can be used to create roles and upload them to Ansible Galaxy.
>
> At the time of this writing, Ansible does not have specific MariaDB official modules. MySQL official modules can be used. However, be careful not try to use features that only apply to MySQL. There are several community-maintained MariaDB roles.
>
> **PUPPET**
>
> Puppet is open source, released under the GNU GPL. It is produced by a homonym company. The page Puppet Partners lists partners that can provide support and consulting about Puppet.
>
> Puppet Forge is a big repository of modules produced by the vendor and by the community, as well as how-to guides.
>
> Currently Puppet has many MariaDB modules.

See Also

For more information about the systems mentioned in this page, from MariaDB users perspective:

- Ansible and MariaDB.
- Puppet and MariaDB.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Automating MariaDB Tasks with Events

MariaDB has an event scheduler that can be used to automate tasks, making them run at regular intervals of time. This page is about using events for automation. For more information about events themselves, and how to work with them, see event scheduler.

Pros and Cons of Using Events for Automation

Events can be compared to Unix cron jobs or Windows scheduled tasks. MariaDB events have at least the following benefits compared to those tools:

- Events are system-independent. The same code can run on any system.
- Events are written in procedural SQL. There is no need to install other languages or libraries.
- If you use user-defined functions, you can still take advantage of them in your events.
- Events run in MariaDB. An implication, for example, is that the results of queries remain in MariaDB itself and are not sent to a client. This means that network glitches don't affect events, there is no overhead due to data roundtrip, and therefore locks are held for a shorter time.

Some drawbacks of using events are the following:

- Events can only perform tasks that can be developed in SQL. So, for example, it is not possible to send alerts. Access to files or remote databases is limited.
- The event scheduler runs as a single thread. This means that events that are scheduled to run while another event is running will wait until the other event has stopped. This means that there is no guarantee that an event will run on exactly it's scheduled time. This should not be a problem as long as one ensures that events are short lived.
- For more events limitations, see Event Limitations.

In many cases you may prefer to develop scripts in an external programming language. However, you should know that simple tasks consisting of a few queries can easily be implemented as events.

Good Practices

When using events to automate tasks, there are good practices one may want to follow.

Move your SQL code in a stored procedure. All the event will do is to call a stored procedures. Several events may call the same stored procedure, maybe with different parameters. The procedure may also be called manually, if necessary. This will avoid code duplication. This will separate the logic from the schedule, making it possible to change an event without a risk of making changes to the logic, and the other way around.

Just like cron jobs, events should log whether if they succeed or not. Logging debug messages may also be useful for non-trivial events. This information can be logged into a dedicated table. The contents of the table can be monitored by a monitoring tool like Grafana. This allows to visualize in a dashboard the status of events, and send alerts in case of a failure.

Examples

Some examples of tasks that could easily be automated with events:

- Copying data from a remote table to a local table by night, using the CONNECT storage engine. This can be a good idea if many rows need be copied, because data won't be sent to an external client.
- Periodically delete historical data. For example, rows that are older than 5 years. Nothing prevents us from doing this with an external script, but probably this wouldn't add any value.

- Periodically delete invalid rows. In an e-commerce, they could be abandoned carts. In a messaging system, they could be messages to users that don't exist anymore.
- Add a new partition to a table and drop the oldest one (partition rotation).

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Automating Upgrades with MariaDB.Org Downloads REST API

The MariaDB Foundation maintains a Downloads REST API. See the Downloads API documentation to find out all the tasks that you can accomplish with this API. Generally speaking, we can say that it provides information about MariaDB products and available versions. This allows to easily automate upgrades for MariaDB and related products.

The Downloads API exposes HTTPS endpoints that return information in JSON format. HTTP and JSON are extremely common standards that can be easily used with any programming language. All the information provided by the API is public, so no authentication is required.

How to Use the API with a Unix Shell

Linux shells are great for writing simple scripts. They are compatible to each other to some extent, so simple scripts can be run on almost any Unix/Linux system. In the following examples we'll use Bash.

On Linux, some programs you'll generally need to work with any REST API are:

- curl, to call HTTP URLs and get their output.
- js, to extract or transform information from a JSON document.

Example: Check When a New Version Becomes GA

A trivial use case is to write a script that checks the list of MariaDB GA major versions and, when something changes, send us an email. So we can test the newest GA version and eventually install it.

The script in this example will be extremely simple. We'll do it this way:

- Retrieve the JSON object describing all MariaDB versions.
- For each element of the array, only show the `release_id` and `release_status` properties, and concatenate them.
- Apply a filter, so we only select the rows containing 'stable' but not 'old' (so we exclude 'Old Stable').
- From the remaining rows, only show the first column (the version number).
- If the results we obtained are different from the previously written file (see last point) send an email.
- Save the results into a file.

This is something that we can easily do with a Unix shell:

```
#!/bin/bash

current_ga_versions=$(
    curl https://downloads.mariadb.org/rest-api/mariadb/ | \
    jq -r '.major_releases[] | .release_id + " " + .release_status' | \
    grep -i 'stable' | grep -vi 'old' | \
    cut -d ' ' -f 1
)

# create file if it doesn't exist, then compare version lists
touch ga_versions
previous_ga_versions=$( cat ga_versions )

echo "$current_ga_versions" > ga_versions

if [ "$current_ga_versions" != "$previous_ga_versions" ];
then
    mail -s 'NOTE: New MariaDB GA Versions' devops@example.com <<< 'There seems to be a new MariaDB GA
version! Yay!'
fi
```

The only non-standard command here is jq. It is a great way to manipulate JSON documents, so if you don't know it you may want to take a look at jq documentation.

How to Use the API with a Python Script

To use the API with Python, we need a module that is able to send HTTP requests and parse a JSON output. The `requests` module has both these features. It can be installed as follows:

```
pip install requests
```

The following script prints stable versions to the standard output:

```
#!/usr/bin/env python

import requests

response = requests.get('https://downloads.mariadb.org/rest-api/mariadb/').json()

for x in response['major_releases']:
    if x['release_status'] == 'Stable':
        print(x['release_id'])
```

`requests.get()` makes an HTTP call of type GET, and `requests.json()` returns a dictionary representing the previously obtained JSON document.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

HashiCorp Vault and MariaDB

Vault is open source software for secret management provided by HashiCorp. It is designed to avoid sharing secrets of various types, like passwords and private keys. When building automation, Vault is a good solution to avoid storing secrets in plain text in a repository.

Important:

Unsafe Operations
It is unsafe to execute `vault secrets disable` or `vault token revoke` while MariaDB is running. Abruptly removing access to the Vault could prevent InnoDB from encrypting data pages during background flushes or log writes, which can lead to a server crash.

Tip:

Always ensure the MariaDB server has completed a clean shutdown before disabling Vault secrets or revoking access tokens.

MariaDB and Vault may relate to each other in several ways:

- MariaDB has a Hashicorp Key Management plugin, to manage and rotate SSH keys.
- Users passwords can be stored in Vault.
- MariaDB (and MySQL) can be used as a secret engine, a component which stores, generates, or encrypts data.
- MariaDB (and MySQL) can be used as a backend storage, providing durability for Vault data.

For information about how to install Vault, see Install Vault, as well as MySQL/MariaDB Database Secrets Engine.

Note:

Namespace support is available as of MariaDB Enterprise Server 11.8.

See the Hashicorp namespaces documentation for details.

Vault Features

Vault is used via an HTTP/HTTPS API.

Vault is identity-based. Users login and Vault sends them a token that is valid for a certain amount of time, or until certain conditions occur. Users with a valid token may request to obtain secrets for which they have proper permissions.

Vault encrypts the secrets it stores.

Vault can optionally audit changes to secrets and secrets requests by the users.

Vault Architecture

Vault is a server. This allows decoupling the secrets management logic from the clients, which only need to login and keep a token until it expires.

The sever can actually be a cluster of servers, to implement high availability.

The main Vault components are:

- Storage Backed: This is where the secrets are stored. Vault only send encrypted data to the backend storage.
- HTTP API: This API is used by the clients, and provides an access to Vault server.
- Barrier: Similarly to an actual barrier, it protects all inner Vault components. The HTTP API and the storage backend are outside of the barrier and could be accessed by anyone. All communications from and to these components have to pass through the barrier. The barrier verifies data and encrypts it. The barrier can have two states: sealed or unsealed. Data can only pass through when the barrier is unsealed. All the following components are located inside the barrier.
- Auth Method: Handles login attempts from clients. When a login succeeds, the auth method returns a list of security policies to Vault core.
- Token Store: Here the tokens generated as a result of a succeeded login are stored.

- Secrets Engines: These components manage secrets. They can have different levels of complexity. Some of them simply expect to receive a key, and return the corresponding secret. Others may generate secrets, including one-time-passwords.
- Audit Devices: These components log the requests received by Vault and the responses sent back to the clients.There may be multiple devices, in which case an Audit Broker sends the request or response to the proper device.

Dev Mode

It is possible to start Vault in dev mode:

```
vault server -dev
```

Dev mode is useful for learning Vault, or running experiments on some particular features. It is extremely insecure, because dev mode is equivalent to starting Vault with several insecure options. This means that Vault should never run in production in dev mode. However, this also means that all the regular Vault features are available in dev mode.

Dev mode simplifies all operations. Actually, no configuration is necessary to get Vault up and running in dev mode. It makes it possible to communicate with the Vault API from the shell without any authentication. Data is stored in memory by default. Vault is unsealed by default, and if explicitly sealed, it can be unsealed using only one key.

For more details, see “Dev” Server Mode in Vault documentation.

Vault Resources and References

- Documentation.
- MySQL/MariaDB Database Secrets Engine.
- MySQL Storage Backend.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Orchestrator Overview

Orchestrator is no longer actively maintained.

Orchestrator is a MySQL and MariaDB high availability and replication management tool. It is released by Shlomi Noach under the terms of the Apache License, version 2.0.

Orchestrator provides automation for MariaDB replication in the following ways:

- It can be used to perform certain operations, like repairing broken replication or moving a replica from one master to another. These operations can be requested using CLI commands, or via the GUI provided with Orchestrator. The actual commands sent to MariaDB are automated by Orchestrator, and the user doesn’t have to worry about the details.
- Orchestrator can also automatically perform a failover in case a master crashes or is unreachable by its replicas. If that is the case, Orchestrator will promote one of the replicas to a master. The replica to promote is chosen based on several criteria, like the server versions, the binary log formats in use and the datacenters locations.

Note that, if we don’t want to use Orchestrator to automate operations, we can still use it as a dynamic inventory. Other tools can use it to obtain a list of existing MariaDB servers via its REST API or CLI commands.

Orchestrator has several big users, listed in the documentation Users page. It is also included in the PMM monitoring solution.

To install Orchestrator, see:

- The install.md for a manual installation;
- The links in README.md, to install Orchestrator using automation tools.

Supported Topologies

Currently, Orchestrator fully supports MariaDB GTID, replication, and semi-synchronous replication. While Orchestrator does not support Galera specific logic, it works with Galera clusters. For details, see Supported Topologies and Versions in Orchestrator documentation.

Architecture

Orchestrator consists of a single executable called `orchestrator`. This is a process that periodically connects to the target servers. It will run SQL queries against target servers, so it needs a user with proper permissions. When the process is running, a GUI is available via a web browser, at the URL 'localhost:3000'. It also exposes a REST API (see Using the web API in the Orchestrator documentation).

Orchestrator expects to find a JSON configuration file called `orchestrator.conf.json`, in `/etc`.

A database is used to store the configuration and the state of the target servers. By default, this is done using built-in SQLite. However, it is possible to use an external MariaDB or MySQL server instance.

If a cluster of Orchestrator instances is running, only one central database is used. One Orchestrator node is active, while the others are passive and are only used for failover. If the active node crashes or becomes unreachable, one of the other nodes becomes the active instance. The `active_node` table shows which node is active. Nodes communicate between them using the Raft protocol.

CLI Examples

As mentioned, Orchestrator can be used from the command-line. Here you can find some examples.

List clusters:

```
orchestrator -c clusters
```

Discover a specified instance and add it to the known topology:

```
orchestrator -c discover -i <host>:<port>
```

Forget about an instance:

```
orchestrator -c topology -i <host>:<port>
```

Move a replica to a different master:

```
orchestrator -c move-up -i <replica-host>:<replica-port> -d <master-host>:<master-port>
```

Move a replica up, so that it becomes a "sibling" of its master:

```
orchestrator -c move-up -i <replica-host>:<replica-port>
```

Move a replica down, so that it becomes a replica of its"sibling":

```
orchestrator -c move-below -i <replica-host>:<replica-port> -d <master-host>:<master-port>
```

Make a node read-only:

```
orchestrator -c set-read-only -i <host>:<port>
```

Make a node writeable:

```
orchestrator -c set-writeable -i <host>:<port>
```

The `--debug` and `--stack` options can be added to the above commands to make them more verbose.

Orchestrator Resources and References

- Orchestrator on GitHub.
- Documentation.
- Raft consensus protocol website.

The README.md file lists some related community projects, including modules to automate Orchestrator with Puppet and other technologies.

On GitHub you can also find links to projects that allow the use of automation software to deploy and manage Orchestrator.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Ansible and MariaDB

See Ansible Overview for MariaDB Users.

Introduction to core Ansible concepts such as inventories, playbooks, and roles, with specific examples of how to structure them for MariaDB deployments like Galera Clusters and replicas.

See Deploying Docker Containers with Ansible.

Explains how to use Ansible's Docker modules to automate the deployment and configuration of MariaDB containers, serving as an alternative to Docker Compose.

See Deploying to Remote Servers with Ansible.

A guide to executing Ansible commands and playbooks on remote servers via SSH, covering basic connectivity tests (ping) and the application of roles to specific host groups.

See Existing Ansible Modules and Roles for MariaDB.

Lists and describes the standard Ansible modules available for managing MariaDB, such as `mysql_db`, `mysql_user`, and `mysql_variables`, highlighting their idempotent nature.

See Installing MariaDB .deb Files with Ansible.

Detailed instructions for automating the installation of MariaDB on Debian/Ubuntu systems, including tasks for adding repositories, importing GPG keys, and installing packages.

See Managing Secrets in Ansible.

Best practices for handling sensitive information like database passwords and SSH keys within Ansible, recommending the use of `ansible-vault` to encrypt secrets.

See Running mariadb-tzinfo-to-sql with Ansible.

Demonstrates how to automate the loading of time zone data into MariaDB using the `mysql_tzinfo_to_sql` utility, with techniques to ensure the task is idempotent.

This page is licensed: CC BY-SA / Gnu FDL

Ansible Overview for MariaDB Users

Ansible is a tool to automate servers configuration management. It is produced by Red Hat and it is open source software released under the terms of the GNU GPL.

It is entirely possible to use Ansible to automate MariaDB deployments and configuration. This page contains generic information for MariaDB users who want to learn, or evaluate, Ansible.

For information about how to install Ansible, see Installing Ansible in Ansible documentation.

Automation Hubs

Normally, Ansible can run from any computer that has access to the target hosts to be automated. It is not uncommon that all members of a team has Ansible installed on their own laptop, and use it to deploy.

Red Hat offers a commercial version of Ansible called Ansible Tower. It consists of a REST API and a web-based interface that work as a hub that handles all normal Ansible operations.

An alternative is AWX. AWX is the open source upstream project from which many Ansible Tower features are originally developed. AWX is released under the terms of the Apache License 2.0. However, Red Hat does not recommend to run AWX in production.

AWX development is fast. It has several features that may or may not end up in Ansible Tower. Ansible Tower is more focused on making AWS features more robust, providing a stable tool to automate production environments.

Design Principles

Ansible allows us to write **playbooks** that describe how our servers should be configured. Playbooks are lists of **tasks**.

Tasks are usually **declarative**. You don't explain *how* to do something, you declare *what* should be done.

Playbooks are **idempotent**. When you apply a playbook, tasks are only run if necessary.

Here is a task example:

```
- name: Install Perl
  package:
    name: perl
    state: present
```

"Install Perl" is just a description that will appear on screen when the task is applied. Then we use the `package` module to declare that a package called "perl" should be installed. When we apply the playbook, if Perl is already installed nothing happens. Otherwise, Ansible installs it.

When we apply a playbook, the last information that appears on the screen is a recap like the following:

```
PLAY RECAP
*********************************************************************************************
mariadb-01        : ok=6    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

This means that six tasks were already applied (so no action was taken), and two tasks were applied.

As the above example shows, Ansible playbooks are written in YAML.

Modules (like `package`) can be written in any language, as long as they are able to process a JSON input and produce a JSON output. However the Ansible community prefers to write them in Python, which is the language Ansible is written in.

Concepts

A piece of Ansible code that can be applied to a server is called a **playbook**.

A **task** is the smallest brick of code in a playbook. The name is a bit misleading, though, because an Ansible task should not be seen as “something to do”. Instead, it is a minimal description of a component of a server. In the example above, we can see a task.

A task uses a single **module**, which is an interface that Ansible uses to interact with a specific system component. In the example, the module is “package”.

A task also has attributes, that describe what should be done with that module, and how. In the example above, “name” and “state” are both attributes. The `state` attribute exists for every module, by convention (though there may be exceptions). Typically, it has at least the “present” and “absent” state, to indicate if an object should exist or not.

Other important code concepts are:

- An inventory determines which hosts Ansible should be able to deploy. Each host may belong to one or more groups. Groups may have children, forming a hierarchy. This is useful because it allows us to deploy on a group, or to assign variables to a group.
- A role describes the state that a host, or group of hosts, should reach after a deploy.
- A play associates hosts or groups to their roles. Each role/group can have more than one role.
- A role is a playbook that describes how certain servers should be configured, based on the logical role they have in the infrastructure. Servers can have multiple roles, for example the same server could have both the “mariadb” and the “mydumper” role, meaning that they run MariaDB and they have mydumper installed (as shown later).
- Tasks can use variables. They can affect how a task is executed (for example a variable could be a file name), or even whether a task is executed or not. Variables exist at role, group or host level. Variables can also be passed by the user when a play is applied.
- Facts are data that Ansible retrieves from remote hosts before deploying. This is a very important step, because facts may determine which tasks are executed or how they are executed. Facts include, for example, the operating system family or its version. A playbook sees facts as pre-set variables.
- Modules implement actions that tasks can use. Action examples are file (to declare that files and directories must exist) or mysql_variables (to declare MySQL/MariaDB variables that need to be set).

Example

Let’s describe a hypothetical infrastructure to find out how these concepts can apply to MariaDB.

The **inventory** could define the following groups:

- “db-main” for the cluster used by our website. All nodes belong to this group.
- “db-analytics” for our replicas used by data analysts.
- “dump” for one or more servers that take dumps from the replicas.
- “proxysql” for one or more hosts that run ProxySQL.

Then we’ll need the following nodes:

- `mariadb-node` for the nodes in `db-main`. This role describes how to setup nodes of a cluster using Galera.
- `mariadb-replica` for the members of `db-analytics`. It describes a running replica, and it includes the tasks that are necessary to provision the node if the data directory is empty when the playbook is applied. The hostname of the primary server is defined in a variable.
- `mariadb`. The aforementioned `mariadb-node` and `mariadb-replica` can be children of this group. They have many things in common (filesystem for the data directory, some basic MariaDB configuration, some installed tools…), so it could make sense to avoid duplication and describe the common traits in a super-role.
- A `mariadb-backup` role to take backups with mariadb-backup, running jobs during the night. We can associate this role to the “ `db-main` ” group, or we could create a child group for servers that will take the backups.

- `mariadb-dump` for the server that takes dumps with mariadb-dump. Note that we may decide to take dumps on a replica, so the same host may belong to `db-analytics` and `mariadb-dump`.
- `proxysql` for the namesake group.

Architecture

Ansible architecture is extremely simple. Ansible can run on any host. To apply playbooks, it connects to the target hosts and runs system commands. By default the connection happens via ssh, though it is possible to develop connection plugins to use different methods. Applying playbooks locally without establishing a connection is also possible.

Modules can be written in any language, though Python is the most common choice in the Ansible community. Modules receive JSON "requests" and facts from Ansible core, they are supposed to run useful commands on a target host, and then they should return information in JSON. Their output informs Ansible whether something has changed on the remote server and if the operations succeeded.

Ansible is not centralized. It can run on any host, and it is common for a team to run it from several laptops. However, to simplify things and improve security, it may be desirable to run it from a dedicated host. Users will connect to that host, and apply Ansible playbooks.

Ansible Resources and References

- Ansible.com
- AWX
- Ansible Tower
- Ansible Galaxy
- Ansible on Wikipedia
- Ansible Automation Platform YouTube channel
- Ansible: Getting Started
- MariaDB Deployment and Management with Ansible (video)

Further information about the concepts discussed in this page can be found in Ansible documentation:

- Basic Concepts.
- Glossary.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Deploying Docker Containers with Ansible

Ansible can be used to manage Docker container upgrades and configuration changes. Docker has native ways to do this, namely Dockerfiles and Docker Compose. But sometimes there are reasons to start basic containers from an image and then manage configuration with Ansible or similar software. See Benefits of Managing Docker Containers with Automation Software.

In this page we'll discuss how to use Ansible to manage Docker containers.

How to Deploy a Container with Ansible

Ansible has modules to manage the Docker server, Docker containers, and Docker Compose. These modules are maintained by the community.

A dynamic inventory plugin for Docker exists. It retrieves the list of existing containers from Docker.

Docker modules and the Docker inventory plugin communicate with Docker using its API. The connection to the API can use a TSL connection and supports key authenticity verification.

To communicate with Docker API, Ansible needs a proper Python module installed on the Ansible node (`docker` or `docker-py`).

Several roles exist to deploy Docker and configure it. They can be found in Ansible Galaxy.

References

Further information can be found in Ansible documentation.

- Docker Guide.
- docker_container module.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Deploying to Remote Servers with Ansible

If we manage several remote servers, running commands on them manually can be frustrating and time consuming. Ansible allows one to run commands on a whole group of servers.

This page shows some examples of ansible-playbook invocations. We'll see how to deploy roles or parts of them to remote servers. Then we'll see how to run commands on remote hosts, and possibly to get information from them. Make sure to read Ansible Overview first, to understand Ansible general concepts.

Pinging Remote Servers

Let's start with the simplest example: we just want our local Ansible to ping remote servers to see if they are reachable. Here's how to do it:

```
ansible -i production-mariadb all -m ping
```

Before proceeding with more useful examples, let's discuss this syntax.

- ansible is the executable we can call to run a command from remote servers.
- -i production-mariadb means that the servers must be read from an inventory called production-mariadb.
- all means that the command must be executed against all servers from the above inventory.
- -m ping specifies that we want to run the ping module. This is not the ping Linux command. It tells us if Ansible is able to connect a remote server and run a simple commands on them.

To run ping on a specific group or host, we can just replace "all" with a group name or host name from the inventory:

```
ansible -i production-mariadb main_cluster -m ping
```

Running Commands on Remote Servers

The previous examples show how to run an Ansible module on remote servers. But it's also possible to run custom commands over SSH. Here's how:

```
ansible -i production-mariadb all -a 'echo $PATH'
```

This command shows the value of `$PATH` on all servers in the inventory "production-mariadb".

We can also run commands as root by adding the `-b` (or `--become`) option:

```
# print a MariaDB variable
ansible -i production-mariadb all -b -a 'mysql -e "SHOW GLOBAL VARIABLES LIKE
\'innodb_buffer_pool_size\';"'

# reboot servers
ansible -i production-mariadb all -b -a 'reboot'
```

Applying Roles to Remote Servers

We saw how to run commands on remote hosts. Applying roles to remote hosts is not much harder, we just need to add some information. An example:

```
ansible-playbook -i production-mariadb production-mariadb.yml
```

Let's see what changed:

- ansible-playbook is the executable file that we need to call to apply playbooks and roles.
- production-mariadb.yml is the play that associates the servers listed in the inventory to their roles.

If we call ansible-playbook with no additional arguments, we will apply all applicable roles to all the servers mentioned in the play.

To only apply roles to certain servers, we can use the `-l` parameter to specify a group, an individual host, or a pattern:

```
# Apply to the mariadb-main role role
ansible-playbook -i production-mariadb -l mariadb-main production-mariadb.yml

# Apply to the mariadb-main-01 host
ansible-playbook -i production-mariadb -l mariadb-main-01 production-mariadb.yml

# Apply to multiple hosts whose name starts with "mariadb-main-"
ansible-playbook -i production-mariadb -l mariadb-main-* production-mariadb.yml
```

We can also apply tasks from roles selectively. Tasks may optionally have tags, and each tag corresponds to an operation that we may want to run on our remote hosts. For example, a "mariadb" role could have the "timezone-update" tag, to update the contents of the timezone tables. To only apply the tasks with the "timezone-update" tag, we can use this command:

```
ansible-playbook -i production-mariadb --tag timezone-update production-mariadb.yml
```

Using tags is especially useful for database servers. While most of the technologies typically managed by Ansible are stateless (web servers, load balancers, etc.) database servers are not. We must pay special attention not to run tasks that could cause a database server outage, for example destroying its data directory or restarting the service when it is not necessary.

Check mode

We should always test our playbooks and roles on test servers before applying them to production. However, if test servers and production servers are not exactly in the same state (which means, some facts may differ) it is still possible that applying roles will fail. If it fails in the initial stage, Ansible will not touch the remote hosts at all. But there are cases where Ansible could successfully apply some tasks, and fail to apply another task. After the first failure, ansible-playbook will show errors and exit. But this could leave a host in an inconsistent state.

Ansible has a *check mode* that is meant to greatly reduce the chances of a failure. When run in check mode, ansible-playbook will read the inventory, the play and roles; it will figure out which tasks need to be applied; then it will connect to target hosts, read facts, and value all the relevant variables. If all these steps succeed, it is unlikely that running ansible-playbook without check mode will fail.

To run ansible-playbook in check mode, just add the `--check` (or `-C`) parameter.

References

Further documentation can be found in the Ansible website:

- ansible tool.
- ansible-playbook tool.
- Validating tasks: check mode and diff mode.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Existing Ansible Modules and Roles for MariaDB

This page contains links to Ansible modules and roles that can be used to automate MariaDB deployment and configuration. The list is not meant to be exhaustive. Use it as a starting point, but then please do your own research.

Modules

At the time of writing, there are no MariaDB-specific modules in Ansible Galaxy. MySQL modules can be used. Trying to use MySQL-specific features may result in errors or unexpected behavior. However, the same applies when trying to use a feature not supported by the MySQL version in use.

Currently, the MySQL collection in Ansible Galaxy contains at least the following modules:

- mysql_db: manages MySQL databases.
- mysql_info: gathers information about a MySQL server.
- mysql_query: runs SQL queries against MySQL.
- mysql_replication: configures and operates asynchronous replication.
- mysql_user: creates, modifies and deletes MySQL users.
- mysql_variables: manages MySQL configuration.

Note that some modules only exist as shortcuts, and it is possible to use mysql_query instead. However, it is important to notice that mysql_query is not idempotent. Ansible does not understand MySQL queries, therefore it cannot check whether a query needs to be run or not.

To install this collection locally:

```
ansible-galaxy collection install community.mysql
```

MariaDB Corporation maintains a ColumnStore playbook on GitHub.

Other Useful Modules

Let's see some other modules that are useful to manage MariaDB servers.

shell and command

Modules like shell and command allow one to run system commands.

To deploy on Windows, win_shell and win_command can be used.

Among other things, it is possible to use one of these modules to run MariaDB queries:

```
- name: Make the server read-only
  # become root to log into MariaDB with UNIX_SOCKET plugin
  become: yes
  shell: $( which mysql ) -e "SET GLOBAL read_only = 1;"
```

The main disadvantage with these modules is that they are not idempotent, because they're meant to run arbitrary system commands that Ansible can't understand. They are still useful in a variety of cases:

- To run queries, because mysql_query is also not idempotent.
- In cases when other modules do not allow us to use the exact arguments we need to use, we can achieve our goals by writing shell commands ourselves.
- To run custom scripts that implement non-trivial logic. Implementing complex logic in Ansible tasks is possible, but it can be tricky and inefficient.
- To call command-line tools. There may be specific roles for some of the most common tools, but most of the times using them is an unnecessary complication.

copy and template

An important part of configuration management is copying configuration files to remote servers.

The copy module allows us to copy files to target hosts. This is convenient for static files that we want to copy exactly as they are. An example task:

```
- name: Copy my.cnf
  copy:
    src: ./files/my.cnf.1
    dest: /etc/mysql/my.cnf
```

As you can see, the local name and the name on remote host don't need to match. This is convenient, because it makes it easy to use different configuration files with different servers. By default, files to copy are located in a `files` subdirectory in the role.

However, typically the content of a configuration file should vary based on the target host, the group and various variables. To do this, we can use the template module, which compiles and copies templates written in Jinja.

A simple template task:

```
- name: Compile and copy my.cnf
  copy:
    src: ./templates/my.cnf.j2
    dest: /etc/mysql/my.cnf
```

Again, the local and the remote names don't have to match. By default, Jinja templates are located in a `templates` subdirectory in the role, and by convention they have the `.j2` extension. This is because Ansible uses Jinja version 2 for templating, at the time writing.

A simple template example:

```
## WARNING: DO NOT EDIT THIS FILE MANUALLY !!
## IF YOU DO, THIS FILE WILL BE OVERWRITTEN BY ANSIBLE

[mysqld]
innodb_buffer_pool_size = {{ innodb_buffer_pool_size }}

<div data-gb-custom-block data-tag="if" data-0='true' data-1='true' data-2='true' data-3='true'>

connect_work_size = {{ connect_work_size }}

</div>
```

Other Common Modules

The following modules are also often used for database servers:

- package, apt or yum. Package is package manager-agnostic. Use them to install, uninstall and upgrade packages.
- user, useful to create the system user and group that run MariaDB binary.
- file can be used to make sure that MariaDB directories (like the data directory) exist and have proper permissions. It can also be used to upload static files.
- template allows to create configuration files (like `my.cnf`) more dynamically, using the Jinja template language.
- service is useful after installing MariaDB as a service, to start it, restart it or stop it.

Roles

Specific roles exist for MariaDB in Ansible Galaxy. Using them for MariaDB is generally preferable, to be sure to avoid incompatibilities and to probably be able to use some MariaDB specific features. However, using MySQL or Percona Server roles is also possible. This probably makes sense for users who also administer MySQL and Percona Server instances.

To find roles that suits you, check Ansible Galaxy search page. Most roles are also available on GitHub.

You can also search roles using the ansible-galaxy tool:

```
ansible-galaxy search mariadb
```

See Also

- MariaDB Deployment and Management with Ansible (video)

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Installing MariaDB .deb Files with Ansible

This page refers to the operations described in Installing MariaDB .deb Files. Refer to that page for a complete list and explanation of the tasks that should be performed.

Here we discuss how to automate such tasks using Ansible. For example, here we show how to install a package or how to import a GPG key; but for an updated list of the necessary packages and for the keyserver to use, you should refer to Installing MariaDB .deb Files.

Adding apt Repositories

To add a repository:

```
- name: Add specified repository into sources list
  ansible.builtin.apt_repository:
    repo: deb [arch=amd64,arm64,ppc64el] http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.3/ubuntu
bionic main
    state: present
```

If you prefer to keep the repository information in a source list file in the Ansible repository, you can upload that file to the target hosts in this way:

```
- name: Create a symbolic link
  ansible.builtin.file:
    src: ./file/mariadb.list
    dest: /etc/apt/sources.list.d/
    owner: root
    group: root
    mod: 644
    state: file
```

Updating the Repository Cache

Both the Ansible modules ansible.builtin.apt and ansible.builtin.apt_repository have an `update_cache` attribute. In ansible.builtin.apt it is set to “no” by default. Whenever a task sets it to ‘yes’, `apt-get update` is run on the target system. You have three ways to make sure that repositories are updated.

The first is to use ansible.builtin.apt_repository to add the desired repository, as shown above. So you only need to worry about updating repositories if you use the file method.

The second is to make sure that `update_cache` is set to ‘yes’ when you install a repository:

```
- name: Install foo
  apt:
    name: foo
    update_cache: yes
```

But if you run certain tasks conditionally, this option may not be very convenient. So the third option is to update the repository cache explicitly as a separate task:

```
- name: Update repositories
  apt:
    - update_cache: yes
```

Importing MariaDB GPG Key

To import the GPG key for MariaDB we can use the ansible.builtin.apt_key Ansible module. For example:

```
- name: Add an apt key by id from a keyserver
  ansible.builtin.apt_key:
    keyserver: hkp://keyserver.ubuntu.com:80
    id: 0xF1656F24C74CD1D8
```

Installing Packages

To install Deb packages into a system:

```
- name: Install software-properties-common
  apt:
    name: software-properties-common
    state: present
```

To make sure that a specific version is installed, performing an upgrade or a downgrade if necessary:

```
- name: Install foo 1.0
  apt:
    name: foo=1.0
```

To install a package or upgrade it to the latest version, use: `state: latest`.

To install multiple packages at once:

```
- name: Install the necessary packages
  apt:
    pkg:
    - pkg1
    - pkg2=1.0
```

If all your servers run on the same system, you will always use ansible.builtin.apt and the names and versions of the packages will be the same for all servers. But suppose you have some servers running systems from the Debian family, and others running systems from the Red Hat family. In this case, you may find convenient to use two different task files for two different types of systems. To include the proper file for the target host's system:

```
- include: mariadb-debian.yml
  when: "{{ ansible_facts['os_family'] }} == 'Debian'
```

The variables you can use to run the tasks related to the proper system are:

- ansible_fact ['distribution']
- ansible_fact ['distribution_major_version']
- ansible_fact ['os_family']

There is also a system-independent package module, but if the package names depend on the target system using it may be of very little benefit.

See Also

- Installing MariaDB .deb Files

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Managing Secrets in Ansible

An Ansible role often runs commands that require certain privileges, so it must perform some forms of login, using passwords or key pairs. In the context of database automation, we normally talk about: SSH access, sudo access, and access to MariaDB. If we write these secrets (passwords or private keys) in clear text in an Ansible repository, anyone who has access to the repository can access them, and this is not what we want.

Let's see how we can manage secrets.

The SSH Password or Keys

Most of the times, Ansible connects to the target hosts via SSH. It is common to use the system username and the SSH keys installed in `/.ssh`, which is the SSH clients default. In this case, nothing has to be done on the clients to be able to allow Ansible to use SSH, as long as they are already able to connect to the target hosts.

It is also possible to specify a different username as ANSIBLE_REMOTE_USER and an SSH configuration file as ANSIBLE_NETCONF_SSH_CONFIG. These settings can be specified in Ansible configuration file or as environment variables.

ANSIBLE_ASK_PASS can be specified. If this is the case, Ansible will prompt the user asking to type an SSH password.

Avoiding Sharing Secrets

As a general rule, any configuration that implies communicating sensible information to the persons who will connects to a system implies some degree of risk. Therefore, the most common choice is to allow users to login into remote systems with their local usernames, using SSH keys.

Once Ansible is able to connect remote hosts, it can also be used to install the public keys of some users to grant them access. Sharing these keys implies no risk. Sharing private keys is never necessary, and must be avoided.

MariaDB has a UNIX_SOCKET plugin that can be used to let some users avoid entering a password, as far as they're logged in the operating system. This authentication method is used by default for the root user. This is a good way to avoid having one more password and possibly writing to a `.my.cnf` file so that the user doesn't have to type it.

Even for users who connect remotely, it is normally not necessary to insert passwords in an Ansible file. When we create a user with a password, a hash of the original password is stored in MariaDB. That hash can be found in the mysql.user table. To know the hash of a password without even creating a user, we can use the PASSWORD() function:

```
SELECT PASSWORD('my_password12') AS hash;
```

When we create a user, we can actually specify a hash instead of the password the user will have to type:

```
CREATE USER user@host IDENTIFIED BY PASSWORD '*54958E764CE10E50764C2EECBB71D01F08549980';
```

ansible-vault

Even if you try to avoid sharing secrets, it's likely you'll have to keep some in Ansible. For example, MariaDB users that connect remotely have passwords, and if we want Ansible to create and manage those users, the hashes must be placed somewhere in our Ansible repository. While a hash cannot be converted back to a password, treating hashes as secrets is usually a good idea. Ansible provides a native way to handle secrets: ansible-vault.

In the simplest case, we can manage all our passwords with a single ansible-vault password. When we add or change a new password in some file (typically a file in `host_vars` or `group_vars`) we'll use ansible-vault to crypt this password. While doing so, we'll be asked to insert our ansible-vault password. When we apply a role and Ansible needs to decrypt this password, it will ask us to enter again our ansible-vault password.

ansible-vault can use more than one password. Each password can manage a different set of secrets. So, for example, some users may have the password to manage regular MariaDB users passwords, and only one may have the password that is needed to manage the root user.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Running mariadb-tzinfo-to-sql with Ansible

For documentation about the `mariadb-tzinfo-to-sql` utility, see mysql_tzinfo_to_sql. This page is about running it using Ansible.

Installing or Upgrading the Package

First, we should install `mariadb-tzinfo-to-sql` if it is available on our system. For example, to install it on Ubuntu, we can use this task. For other systems, use the proper module and package name.

```
- name: Update timezone info
  tags: [ timezone-update ]
  apt:
    name: tzdata
    state: latest
    install_recommends: no
  register: timezone_info
```

This task installs the latest version of the `tzdata`, unless it is already installed and up to date. We register the `timezone_info` variables, so we can only run the next task if the package was installed or updated.

We also specify a `timezone-update` tag, so we can apply the role to only update the timezone tables.

Running the Script

The next task runs `mariadb-tzinfo-to-sql`.

```
- name: Move system timezone info into MariaDB
  tags: [ timezone-update ]
  shell: >
    mysql_tzinfo_to_sql /usr/share/zoneinfo \
      | grep -v "^Warning" \
      | mysql --database=mysql
  when: timezone_info.changed
```

We use the `shell` module to run the command. Running a command in this way is not idempotent, so we specify `when: timezone_info.changed` to only run it when necessary. Some warnings may be generated, so we pipe the output of `mysql_tzinfo_to_sql` to `grep` to filter warnings out.

Using Galera

If we're using MariaDB Galera Cluster we'll want to only update the timezone tables in one node, because the other nodes will replicate the changes. For our convenience, we can run this operation on the first node. If the nodes hostnames are defined in a list called `cluster_hosts`, we can check if the current node is the first in this way:

```
when: timezone_info.changed and inventory_hostname == cluster_hosts[0].hostname
```

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Puppet and MariaDB

See Puppet Overview.

Introduction to Puppet's architecture (agent-master vs. standalone), concepts like manifests and catalogs, and how it can be used for configuration management of MariaDB servers.

See Deploying Docker Containers with Puppet.

A guide to managing Docker container lifecycles using Puppet's `docker` resource type, covering image pulling, container execution, and upgrades for MariaDB.

See Existing Puppet Modules for MariaDB.

Lists available Puppet modules for MariaDB from the Puppet Forge and GitHub, noting that while no official module exists, community and generic MySQL modules are often used.

See Puppet hiera Configuration System.

Explains how to use Hiera, Puppet's hierarchical key/value lookup tool, to separate MariaDB configuration data from code and manage environment-specific settings.

See Bolt Examples.

Demonstrates how to use Bolt, an orchestration tool in the Puppet ecosystem, to run ad-hoc commands, scripts, and tasks on remote MariaDB servers without a permanent agent.

This page is licensed: CC BY-SA / Gnu FDL

Puppet Overview

Puppet is a tool to automate servers configuration management. It is produced by Puppet Inc, and released under the terms of the Apache License, version 2.

It is entirely possible to use Ansible to automate MariaDB deployments and configuration. This page contains generic information for MariaDB users who want to learn, or evaluate, Puppet.

Puppet modules can be searched using Puppet Forge. Most of them are also published on GitHub with open source licenses. Puppet Forge allows filtering modules to only view the most reliable: supported by Puppet, supported by a Puppet partner, or approved.

For information about installing Puppet, see Installing and upgrading in Puppet documentation.

Design Principles

With Puppet, you write **manifests** that describe the resources you need to run on certain servers and their **attributes**.

Therefore manifests are **declarative**. You don't write the steps to achieve the desired result. Instead, you describe the desired result. When Puppet detects differences between your description and the current state of a server, it decides what to do to fix those differences.

Manifests are also **idempotent**. You don't need to worry about the effects of applying a manifest twice. This may happen (see Architecture below) but it won't have any side effects.

Defining Resources

Here's an example of how to describe a resource in a manifest:

```
file { '/etc/motd':
  content => '',
  ensure => present,
}
```

This block describes a resource. The resource type is `file`, while the resource itself is `/etc/motd`. The description consists of a set of attributes. The most important is `ensure`, which in this case states that the file must exist. It is also common to use this resource to indicate that a file (probably created by a previous version of the manifest) doesn't exist.

These classes of resource types exist:

- Built-in resources, or Puppet core resources: Resources that are part of Puppet, maintained by the Puppet team.
- Defined resources: Resources that are defined as a combination of other resources. They are written in the Puppet domain-specific language.
- Custom resources: Resources that are written by users, in the Ruby language.

To obtain information about resources:

```
# list existing resource types
puppet resource --typesupp
# print information about the file resource type
puppet describe file
```

To group several resources in a reusable class:

```
class ssh_server {
  file { '/etc/motd':
    content => '',
    ensure => present,
  }
  file { '/etc/issue.net':
    content => '',
    ensure => present,
  }
}
```

There are several ways to include a class. For example:

```
include Class['ssh_server']
```

Defining Nodes

Puppet has a **main manifest** that could be a `site.pp` file or a directory containing `.pp` files. For simple infrastructures, we can define the nodes here. For more complex infrastructures, we may prefer to import other files that define the nodes.

Nodes are defined in this way:

```
node 'maria-1.example.com' {
  include common
  include mariadb
}
```

The resource type is `node`. Then we specify a hostname that is used to match this node to an existing host. This can also be a list of hostnames, a regular expression that matches multiple nodes, or the `default` keyword that matches all hosts. To use a regular expression:

```
node /^(maria|mysql)-[1-3]\.example\.com$/ {
  include common
}
```

Concepts

The most important Puppet concepts are the following:

Concepts	Definition
Target	A host whose configuration is managed via Puppet.
Group	A logical group of targets. Example: `mariadb` group.
Facts	Information collected from targets (e.g., system name/version) using Facter. Can be core or custom facts.
Manifest	A description that can be applied to a target.
Catalog	A compiled manifest.
Apply	Modifying a target's state to reflect its manifest description.
Module	A set of manifests.
Resource	A minimal piece of description in a manifest describing system components like files or services.
Resource Type	Determines the class of a resource, e.g., `file` resource type with multiple resources describing different files.
Attribute	Characteristic of a resource, like file owner or mode.
Class	A group of resources reusable across several manifests.

Architecture

Depending on how the user decides to deploy changes, Puppet can use two different architectures:

- An Agent-master architecture. This is the preferred way to use Puppet.
- A standalone architecture, that is similar to Ansible architecture.

Agent-master Architecture

A **Puppet master** stores a catalog for each target. There may be more than one Puppet master, for redundancy.

Each target runs a **Puppet agent** in background. Each Puppet agent periodically connects to the Puppet master, sending its facts. The Puppet master compiles the relevant manifest using the facts it receives, and send back a catalog. Note that it is also possible to store the catalogs in PuppetDB instead.

Once the Puppet agent receives the up-to-date catalog, it checks all resources and compares them with its current state. It applies the necessary changes to make sure that its state reflects the resources present in the catalog.

Standalone Architecture

With this architecture, the targets run **Puppet apply**. This application usually runs as a Linux cron job or a Windows scheduled task, but it can also be manually invoked by the user.

When Puppet apply runs, it compiles the latest versions of manifests using the local facts. Then it checks every resource from the resulting catalogs and compares it to the state of the local system, applying changes where needed.

Newly created or modified manifests are normally deployed to the targets, so Puppet apply can read them from the local host. However it is possible to use PuppetDB instead.

PuppetDB

PuppetDB is a Puppet node that runs a PostgreSQL database to store information that can be used by other nodes. PuppetDB can be used with both the Agent-master and the standalone architectures, but it is always optional. However it is necessary to use some advanced Puppet features.

PuppetDB stored the following information:

- The latest facts from each target.
- The latest catalogs, compiled by Puppet apply or a Puppet master.
- Optionally, the recent history of each node activities.

External Node Classifiers

With both architectures, it is possible to have a component called an External Node Classifier (ENC). This is a script or an executable written in any language that Puppet can call to determine the list of classes that should be applied to a certain target.

An ENC received a node name in input, and should return a list of classes, parameters, etc, as a YAML hash.

Bolt

Bolt can be used in both architectures to run operations against a target or a set of targets. These operations can be commands passed manually to Bolt, scripts, Puppet tasks or plans. Bolt directly connects to targets via ssh and runs system commands.

See Bolt Examples to get an idea of what you can do with Bolt.

hiera

hiera is a hierarchical configuration system that allows us to:

- Store configuration in separate files;
- Include the relevant configuration files for every server we automate with Puppet.

See Puppet hiera Configuration System for more information.

Puppet Resources

- Puppet documentation.
- forge.puppet.com.
- Puppet on GitHub.
- Puppet on Wikipedia.

More information about the topics discussed in this page can be found in the Ansible documentation:

- Puppet Glossary in Puppet documentation.
- Overview of Puppet's architecture in Puppet documentation.
- PuppetDB documentation.
- Classifying nodes in Puppet documentation.
- Hiera in Puppet documentation.
- Bolt documentation.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Deploying Docker Containers with Puppet

Puppet can also be used to manage Docker container upgrades and configuration changes. Docker has more specific tools for this purpose, but sometimes there are reasons to choose alternatives. See Benefits of Managing Docker Containers with Automation Software.

In this page you will find out what managing Docker with Puppet looks like. All the snippets in this page use the `docker` resource type, supported by the Puppet company.

How to Install, Upgrade or Uninstall Docker with Puppet

Installing or upgrading Docker is simple:

```
class { 'docker':
  use_upstream_package_source => false,
  version => '17.09.0~ce-0~debian',
}
```

In this example we are using our system's repositories instead of Docker official repositories, and we are specifying the desired version. To upgrade Docker later, all we need to do is to modify the version number. While specifying a version is not mandatory, it is a good idea because it makes our manifest more reproducible.

To uninstall Docker:

```
class { 'docker':
  ensure => absent
}
```

Check the `docker` resource type documentation to find out how to use more features: for example you can use Docker Enterprise Edition, or bind the Docker daemon to a TCP port.

How to Build or Pull Docker Images with Puppet

To pull an image from Dockerhub:

```
docker::image { 'mariadb:10.0': }
```

We specified the `10.0` tag to get the desired MariaDB version. If we don't, the image with the `latest` tag will be used. Note that this is not desirable in production, because it can lead to unexpected upgrades.

You can also write a Dockerfile yourself, and then build it to create a Docker image. To do so, you need to instruct Puppet to copy the Dockerfile to the target and then build it::

```
file { '/path/to/remote/Dockerfile':
  ensure => file,
  source => 'puppet:///path/to/local/Dockerfile',
}

docker::image { 'image_name':
  docker_file => '/path/to/remote/Dockerfile'
}
```

It is also possible to subscribe to Dockerfile changes, and automatically rebuild the image whenever a new file is found:

```
docker::image { 'image_name':
  docker_file => '/path/to/remote/Dockerfile'
  subscribe => File['/path/to/remote/Dockerfile'],
}
```

To remove an image that was possibly built or pulled:

```
docker::image { 'mariadb':
  ensure => absent
}
```

How to Deploy Containers with Puppet

To run a container:

```
docker::run { 'mariadb-01':
    image   => 'mariadb:10.5',
    ports   => ['3306:6606']
}
```

`mariadb-01` is the contained name. We specified the optional `10.5` tag, and we mapped the guest port `3306` to the host port `6606`. In production, you normally don't map ports because you don't need to connect MariaDB clients from the host system to MariaDB servers in the containers. Third-party tools can be installed as separate containers.

References

- docker resource type documentation, in Puppet documentation.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Existing Puppet Modules for MariaDB

This page contains links to Puppet modules that can be used to automate MariaDB deployment and configuration. The list is not meant to be exhaustive. Use it as a starting point, but then please do your own research.

Puppet Forge

Puppet Forge is the website to search for Puppet modules, maintained by the Puppet company. Modules are searched by the technology that needs to be automated, and the target operating system.

Search criteria include whether the modules are supported by Puppet or its partners, and whether a module is approved by Puppet. Approved modules are certified by Puppet based on their quality and maintenance standards.

Acceptance Tests

Some modules that support the Puppet Development Kit allow some types of acceptance tests.

We can run a static analysis on a module's source code to find certain bad practices that are likely to be a source of bugs:

```
pdk validate
```

If a module's authors wrote unit tests, we can run them in this way:

```
pdk test unit
```

Supported Modules for MariaDB

At the time of writing, there are no supported or approved modules for MariaDB.

However there is a mysql module supported by Puppet, that supports the Puppet Development Kit. Though it doesn't support MariaDB-specific features, it works with MariaDB. Its documentation shows how to use the module to install MariaDB on certain operating systems.

Several unsupported and not approved modules exist for MariaDB and MaxScale.

Resources and References

- Puppet Forge website.
- Puppet Development Kit documentation.
- Modules overview in Puppet documentation.
- Beginner's guide to writing modules in Puppet documentation.
- Puppet Supported Modules page in Puppet Forge.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Puppet hiera Configuration System

hiera is part of Puppet. It is a hierarchical configuration system that allows us to:

- Store configuration in separate files;
- Include the relevant configuration files for every server we automate with Puppet.

hiera Configuration Files

Each hierarchy allows to one choose the proper configuration file for a resource, based on certain criteria. For example criteria may include node names, node groups, operating systems, or datacenters. Hierarchies are defined in a `hiera.yaml` file, which also defines a path for the files in each hierarchy.

Puppet facts are commonly used to select the proper files to use. For example, a path may be defined as `"os/%{facts.os.name}.yaml"`. In this case, each resource will use a file named after the operating system it uses, in the

os directory. You may need to use custom facts, for example to check which microservices will use a MariaDB server, or in which datacenter it runs.

We do not have to create a file for each possible value of a certain fact. We can define a default configuration file with settings that are reasonable for most resources. Other files, when included, will override some of the default settings.

A hiera configuration file will look like this:

```
version: 5
defaults:
  datadir: global
  data_hash: yaml_data

hierarchy:
  - name: "Node data"
    path: "nodes/%{trusted.certname}.yaml"

  - name: "OS data"
    path: "os/%{facts.os.family}.yaml"

  - name: "Per-datacenter business group data" # Uses custom facts.
    path: "location/%{facts.whereami}/%{facts.group}.yaml"
```

This file would include the global files, the OS-specific files and the node-specific files. Each hierarchy will override settings from previous hierarchies.

We can actually have several hiera configuration files. `hiera.yaml` is the global file. But we will typically have additional hiera configuration files for each environment. So we can include the configuration files that apply to production, staging, etc, plus global configuration files that should be included for every environment.

Importantly, we can also have hiera configuration files for each module. So, for example, a separate `mariadb/hiera.yaml` file may defined the hierarchies for MariaDB servers. This allow us to define, for example, different configuration files for MariaDB and for MaxScale, as most of the needed settings are typically different.

Configuration Files

You probably noticed that, in the previous example, we defined `data_hash: yaml_data`, which indicates that configuration files are written in YAML. Other allowed formats are JSON and HOCON. The `data_hash` setting is defined in `defaults`, but it can be overridden by hierarchies.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Bolt Examples

This page shows some examples of what we can do with Bolt to administer a set of MariaDB servers. Bolt is a tool that is part of the Puppet ecosystem.

For information about installing Bolt, see Installing Bolt in Bolt documentation.

Inventory Files

The simplest way to call Bolt and instruct it to do something on some remote targets is the following:

```
bolt ... --nodes 100.100.100.100,200.200.200.200,300,300,300,300
```

However, for non-trivial setups it is usually better to use an inventory file. An example:

```
targets:
  - uri: maria-1.example.com
    name: maria_1
    alias: mariadb_main
  ...
```

In this way, it will be possible to refer the target by name or alias.

We can also define groups, followed by the group members. For example:

```
groups:
  - name: mariadb-staging
    targets:
        - uri: maria-1.example.com
        name: maria_1
        - uri: maria-2.example.com
        name: maria_2
  - name: mariadb-production
    targets:
      ...
...
```

With an inventory of this type, it will be possible to run Bolt actions against all the targets that are members of a group:

```
bolt ... --nodes mariadb-staging
```

In the examples in the rest of the page, the `--targets` parameter will be indicated in this way, for simplicity: `--targets <targets>`.

Running Commands on Targets

The simplest way to run a command remotely is the following:

```
bolt command run 'mariadb-admin start-all-slaves' --targets <targets>
```

Copying Files

To copy a file or a whole directory to targets:

```
bolt file upload /path/to/source /path/to/destination --targets <targets>
```

To copy a file or a whole directory from the targets to the local host:

```
bolt file download /path/to/source /path/to/destination --targets <targets>
```

Running Scripts on Targets

We can use Bolt to run a local script on remote targets. Bolt will temporarily copy the script to the targets, run it, and delete it from the targets. This is convenient for scripts that are meant to only run once.

```
bolt script run rotate_logs.sh --targets <targets>
```

Running Tasks on Targets

Puppet tasks are not always as powerful as custom scripts, but they are simpler and many of them are idempotent. The following task stops MariaDB replication:

```
bolt task run mysql::sql --targets <targets> sql="STOP REPLICA"
```

Applying Puppet Code on Targets

It is also possible to apply whole manifests or portions of Puppet code (resources) on the targets.

To apply a manifest:

```
bolt apply manifests/server.pp  --targets <targets>
```

To apply a resource description:

```
bolt apply --execute "file { '/etc/mysql/my.cnf': ensure => present }" --targets <targets>
```

Bolt Resources and References

- Bolt documentation.
- Bolt on GitHub.

Further information about the concepts explained in this page can be found in Bolt documentation:

- Inventory Files in Bolt documentation.
- Applying Puppet code in Bolt documentation.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Containers

Containers are an OCI standard format for software images and their specified time all bundled up into a single distributable time. They can be used for production, development or testing.

Docker Inc. run a Docker Official Images program to provide users with an essential base implementation of MariaDB in a container and to exemplify best practices of a container.

The containers are available on Docker Hub as **docker.io/library/mariadb** though many container runtime implementation will fill in the **docker.io/library** where the host/path isn't specified.

The containers are in a Open Container Initiative format that allows the containers to be interoperable with a number of container runtime implementations. Docker, or more fully Docker Engine, is just one of the many available runtimes.

Many people use MariaDB Docker Official Image containers in CI systems like GitHub Actions, though its possible to use these in production environments like kubernetes.

The MariaDB Server container images are available with a number of tags:

- A full version, like 10.11.5
- A major version like 10.11
- The most recent stable GA version - latest
- The most recent stable LTS version - lts

Versions that aren't stable will be suffixed with **-rc**, or **-alpha** to clearly show their release status, and enables Renovatebot and other that follow semantic versioning to follow updates.

For a consistent application between testing an production environment using the SHA hash of the image is recommended like **docker.io/library/mariadb@sha256:29fe5062baf36bae8ec68f21a3dce4f0372dadc185e687624f1252fc49d91c67**. There is a list of mapping and history of tags to SHA hash on the Docker Library repository.

See Adding Plugins to the MariaDB Docker Official Image.

Instructions on how to extend the official MariaDB Docker image by installing additional plugins and dependencies using a custom Dockerfile.

See Benefits of Managing MariaDB Containers with Orchestration Software.

Explains the advantages of using orchestration tools like Kubernetes or Docker Swarm for managing MariaDB containers, including automated failover, scaling, and rolling updates.

See Container Backup and Restoration.

Complete container backup guide: docker volume create, mariadb-dump –all-databases, mariadb-backup –backup/–prepare/–copy-back operations.

See Container Security.

Discusses security best practices for running MariaDB in containers, addressing topics like root user privileges, volume permissions, and network isolation.

See Creating a Custom Container Image.

Guide to building a custom MariaDB container image to include specific configuration files, scripts, or pre-loaded data.

See Deploy MariaDB Enterprise Server with Docker.

Instructions for deploying MariaDB Enterprise Server using the official enterprise Docker images, including handling license keys and entitlements.

See MariaDB Enterprise Docker Registry for MariaDB Enterprise Server.

How to authenticate with and pull images from the private MariaDB Enterprise Docker Registry.

See Docker and AWS EC2.

Specific considerations and steps for running MariaDB Docker containers on Amazon EC2 instances.

See Docker and Google Cloud.

Guide for deploying MariaDB containers on Google Cloud Platform (GCP) compute resources.

See Docker and Microsoft Azure.

Instructions for running MariaDB containers within the Microsoft Azure ecosystem.

See Docker Official Image Frequently Asked Questions.

Answers to common questions regarding the official MariaDB image, covering versioning, tagging, and default configurations.

See Installing and Using MariaDB via Docker.

Complete guide to MariaDB in Docker. Complete resource for container deployment, volume management, networking, and environment setup for production use.

See MariaDB Container Cheat Sheet.

A concise reference of common Docker commands and environment variables used with MariaDB containers.

See MariaDB Enterprise Server Docker Official Image Environment Variables.

Complete guide to MariaDB in Docker. Complete resource for container deployment, volume management, networking, and environment setup for production use.

See Setting Up a LAMP Stack with Docker Compose.

Complete LAMP stack Docker Compose: define docker-compose.yml services (web/mariadb), set volumes/env vars (${MARIADB_VERSION}), docker-compose up/down.

See Using Healthcheck.

Complete healthcheck.sh Docker reference: –connect, –innodb_initialized, –replication_* checks, .my-healthcheck.cnf config, and environment variables.

See MariaDB Enterprise Container Image Configuration Reference.

Configuration reference for the official MariaDB Enterprise container images — environment variables, command-line flags, and sidecar patterns for production deployments.

This page is licensed: CC BY-SA / Gnu FDL

Adding Plugins to the MariaDB Docker Official Image

MariaDB has many plugins. Most are not enabled by default, some are in the `mariadb` container, while others need to be installed from additional packages.

The following methods summarize Installing plugins in the MariaDB Docker Library Container (mariadb.org blog post) on this topic.

Which Plugins Does the Container Contain?

To see which plugins are available in the mariadb:

```
$ docker run --rm mariadb:latest ls -C /usr/lib/mysql/plugin
```

Enabling a Plugin Using Flags

Using the `--plugin-load-add` flag with the plugin name (can be repeated), the plugins will be loaded and ready when the container is started:

For example, to enable the `simple\_password\_check` plugin:

```
$ docker run --name some-%%REPO%% -e MARIADB_ROOT_PASSWORD=my-secret-pw --network=host -d mariadb:latest
--plugin-load-add=simple_password_check
```

Enabling a Plugin in the Configuration Files

plugin-load-add` can be used as a configuration option to load plugins. The example below loads the FederatedX Storage Engine.

```
$ printf "[mariadb]\nplugin-load-add=ha_federatedx\n" > /my/custom/federatedx.conf
$ docker run --name some-mariadb -v /my/custom:/etc/mysql/conf.d -e MARIADB_ROOT_PASSWORD=my-secret-pw -d
mariadb:latest
```

Install a Plugin Using SQL in /docker-entrypoint-initdb.d

INSTALL SONAME can be used to install a plugin as part of the database initialization.

Create the SQL file used in initialization:

```
$ echo 'INSTALL SONAME "disks";' > my_initdb/disks.sql
```

In this case, the `my\_initdb` is a `/docker-entrypoint-initdb.d` directory per “Initializing a fresh instance” section above.

Identifying Additional Plugins in Additional Packages

A number of plugins are in separate packages to reduce their installation size. The package names of MariaDB-created plugins can be determined using the following command:

```
$ docker run --rm mariadb:latest sh -c 'apt-get update -qq && apt-cache search mariadb-plugin'
```

Creating an Image With Plugins From Additional Packages

A new image needs to be created when using additional packages. The `mariadb` image can however be used as a base:

In the following, the CONNECT Storage Engine is installed:

```
FROM mariadb:latest
RUN apt-get update && \
    apt-get install mariadb-plugin-connect -y && \
    rm -rf /var/lib/apt/lists/*
```

Installing plugins from packages creates a configuration file in the directory `/etc/mysql/mariadb.conf.d/` that loads the plugin on startup.

This page is licensed: CC BY-SA / Gnu FDL

Benefits of Managing MariaDB Containers with Orchestration Software

In this page we'll discuss why automating containers with software like Ansible or Puppet may be desirable in some cases. To talk about this, we'll first need to discuss why containers are defined *ephemeral*, and how this applies to containerized database servers (particularly MariaDB).

During the discussion, we should keep in mind that Docker Engine, CRI-I, containerd, Mirantis Container Runtime, Podman and other OCI container runtimes can be used to setup production and/or development environments. These use cases are very different from a database perspective: a production database may be big, and typically contains data that we don't want to lose. Development environments usually contain small sample data that can be rebuilt relatively quickly. This page focuses on the latter case.

Container Ephemeral Nature

Images are an OCI specified format that can be compiled from Dockerfiles as one of the ways. Containers are the OCI runtime specified way of creating a runtime version of an images. Normally, a container is not modified from the moment it is created. In other words, containers are usually designed to be **ephemeral**, meaning that they can be destroyed and replaced with new containers at any time. Provided that there is proper redundancy (for example, there are several web servers running the same services) destroying one container and starting a new one of the same type won't cause any damage.

We will discuss a bit later how this applies to MariaDB, and more generally to database servers.

When something should change, for example some software version or configuration, normally Dockerfiles are updated and containers are recreated from the latest image versions. For this reason, containers shouldn't contain anything that shouldn't be lost, and recreating them should be an extremely cheap operation. **Docker Compose** or the **Swarm mode** are used to declare which containers form a certain environment, and how they communicate with each other.

On the contrary, Ansible and Puppet are mainly built to manage the configuration of existing servers. It doesn't recreate servers, it changes their configuration. So Docker and Ansible have very different approaches. For this reason, Ansible and Puppet are not frequently used to deploy containers to production. However, using them together can bring some benefits, especially for development environments.

More on this later in the page. First, we need to understand how these concepts apply to database servers.

Stateful Technologies

Using ephemeral containers works very well for *stateless* technologies, like web servers and proxies. These technologies virtually only need binaries, configuration and small amounts of data (web pages). If some data need to be restored after a container creation, it will be a fast operation.

In the case of a database, the problem is that data can be large and need to be written somewhere. We don't want all databases to disappear when we destroy a container. Even if we had an up-to-date backup, restoring it would take time.

However, OCI Containers has features called **volumes**. A volume is a directory in the host system mapped to a directory in one or more containers. Volumes are not destroyed when containers are destroyed. They can be used to share data between any number of containers and the host system. Therefore, they are also a good way to persist data.

Suppose a MariaDB container called `mariadb-main-01` uses a volume that is mapped to `/var/docker/volumes/mariadb-main`. At some point we want to use a more recent MariaDB version. As explained earlier, the container way to do this is to destroy the container and create a new one that uses a more recent version of the MariaDB image.

So, we will destroy `mariadb-main-01`. The volume is still there. Then we create a new container with the same name, but based on a newer image. We make sure to link the volume to the new container too, so it will be able to use `/var/docker/volumes/mariadb-main` again. At this point we may want to run mariadb-upgrade, but apart from that, everything should *just work*.

The container runtime implementations also provide the opportunity to create a volume with an explicit name and this is also persistent. The actual location on the filesystem is managed by the runtime.

The above described steps are simple, but running them manually is time consuming and error-prone. Automating them with some automation software like Ansible or Puppet is often desirable.

Ways to Deploy Containers

Containers can be deployed in the following ways:

- Manually. See Installing and Using MariaDB via Docker. This is not recommended for production, or for complex environments. However, it can easily be done for the simplest cases. If we want to make changes to our custom images, we'll need to modify the Dockerfiles, destroy the containers and recreate them.
- With Docker Compose. See Setting Up a LAMP Stack with Docker Compose for a simple example. When we modify a Dockerfile, we'll need to destroy the containers and recreate them, which is usually as simple as running `docker compose down` followed by `docker compose up`. After changing `docker-cmpose.yml` (maybe to add a container or a network) we'll simply need to run `docker compose up` again, because it is idempotent.
- Using Ansible, Puppet or other automation software, as mentioned before. We can use Ansible or Puppet to create the containers, and run them again every time we want to apply some change to the containers. This means that the containers are potentially created once and modified any number of times.

In all these cases, it is entirely possible to add Vagrant to the picture. Vagrant is a way to deploy or provision several hosts, including virtual machines (the most common case), and containers. It is agnostic in regarding the underlying technology, so it can deploy to a virtual machine, a container, or even a remote server in the same way. Containers can work with Vagrant in two ways:

- As a provisioner. In this case Vagrant will most commonly deploy a virtual machine, and will use Docker to setup the applications that need to run in it, as containers. This guarantees a higher level of isolation, compared to

running the containers in the local host. Especially if you have different environments to deploy locally, because you can have them on different virtual machines.

- As a provider. Vagrant will deploy one or more containers locally. Once each container is up, Vagrant can optionally use a provisioner on it, to make sure that the container runs the proper software with proper configuration. In this case, Ansible, Puppet or other automation software can be used as a provisioner. But again, this is optional: it is possible to make changes to the Dockerfiles and recreate the containers every time.

Benefits of Managing Containers with Automation Software

Containers can be entirely managed with Docker Compose or the Swarm mode. This is often a good idea.

However, choosing to use automation software like Ansible or Puppet has some benefits too. Benefits include:

- Containers allow working without modifying the host system, and their creation is very fast. Much faster than virtual machines. This makes containers desirable for development environments.
- As explained, making all containers ephemeral and using volumes to store important data is possible. But this means adding some complexity to adapt an ephemeral philosophy to technologies that are not ephemeral by nature (databases). Also, many database professionals don't like this approach. Using automation software allows easily triggering upgrades and configuration changes in the containers, treating them as non-ephemeral systems.
- Sometimes containers are only used in development environments. If production databases are managed via Ansible, Puppet, or other automation software, this could lead to some code duplication. Dealing with configuration changes using the same procedures will reduce the cost of maintenance.
- While recreating containers is fast, being able to apply small changes with Ansible or Puppet can be more convenient in some cases: particularly if we write files into the container itself, or if recreating a container bootstrap involves some lengthy procedure.
- Trying to do something non-standard with Dockerfiles can be tricky. For example, running two processes in a container is possible but can be problematic, as containers are designed to run single main process per container. However there are situations when this is desirable. For example PMM containers run several different processes. Launching additional processes with Ansible or Puppet may be easier than doing it with a Dockerfile.

With all this in mind, let's see some examples of cases when managing containers with Ansible, Puppet or other automation software is preferable, rather than destroying containers every time we want to make a change:

- We use Ansible or Puppet in production, and we try to keep development environments as similar as possible to production. By using Ansible/Puppet in development too, we can reuse part of the code.
- We make changes to the containers often, and recreating containers is not as fast as it should be (for example because a MariaDB dump needs to be restored).
- Creating a container implies some complex logic that does not easily fit a Dockerfile or Docker Compose (including, but not limited to, running multiple processes per container).

That said, every case is different. There are environments where these advantages do not apply, or bring a very small benefit. In those cases, the cost of adding some automation with Ansible, Puppet or similar software is probably not justified.

How to Deploy to Container from Orchestration Software

Suppose you want to manage containers configuration with Ansible.

At a first glance, the simplest way is to run Ansible in the host system. It will need to connect to the containers via SSH, so they need to expose the 22 port. But we have multiple containers, so we'll need to map the 22 port of each container to a different port in the host. This is hard to maintain and potentially insecure: in production you want to avoid exposing any container port to the host.

A better solution is to run Ansible itself in a container. The playbooks will be in a container volume, so we can access them from the host system to manage them more easily. The Ansible container will communicate with other containers using a container network, using the standard 22 port (or another port of your choice) for all containers.

See Also

See these pages on how to manage containers with different automation technologies:

- Deploying Containers with Ansible.
- Deploying Containers with Puppet.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Container Backup and Restoration

MariaDB databases in containers need backup and restore like their non-container equivalents.

Logical Backups

In this section, we will assume that the MariaDB container has been created as follows:

```
$ docker volume create mariadb_data
$ docker volume create mariadb_backup
$ docker run --rm \
  -v mariadb_data:/var/lib/mysql \
  -v mariadb_backup:/backup \
  mariadb \
  chown -R mysql:mysql /var/lib/mysql /backup
$ docker run -d --name mariadb \
  -v mariadb_data:/var/lib/mysql \
  -v mariadb_backup:/backup \
  -e MARIADB_ROOT_PASSWORD='MariaDB11!' \
  <mariadb-image>
```

Backup

mariadb-dump is in the Docker Official Image and can be used as follows:

```
$ docker exec mariadb \
  sh -c 'mariadb-dump --all-databases -u root -p"$MARIADB_ROOT_PASSWORD" > backup/db.sql'
```

Restoring Data from Dump Files

For restoring data, you can use the following `docker exec` command:

```
$ docker exec mariadb \
  sh -c 'mariadb -u root -p"$MARIADB_ROOT_PASSWORD" < backup/db.sql'
```

Physical Backups

mariadb-backup is in the Docker Official Image.

Backup

mariadb-backup can create a backup as follows:

To perform a backup using mariadb-backup, a second container is started that shares the original container's data directory. An additional volume for the backup needs to be included in the second backup instance. Authentication

against the MariaDB database instance is required to successfully complete the backup. In the example below, a `mysql@localhost` user is used with the MariaDB server's Unix socket shared with the backup container.

Note: Privileges listed here are for 10.5+. For an exact list, see mariadb-backup: Authentication and Privileges.

```
$ docker volume create mariadb_data
$ docker volume create mariadb_backup
$ docker run --rm \
  -v mariadb_data:/var/lib/mysql \
  -v mariadb_backup:/backup \
  mariadb \
  chown -R mysql:mysql /var/lib/mysql /backup
$ docker run -d --name mariadb \
  -v mariadb_data:/var/lib/mysql \
  -v mariadb_backup:/backup \
  -e MARIADB_ROOT_PASSWORD='MariaDB11!' \
  -e MARIADB_MYSQL_LOCALHOST_USER=1 \
  -e MARIADB_MYSQL_LOCALHOST_GRANTS='RELOAD, PROCESS, LOCK TABLES, BINLOG MONITOR' \
  <mariadb-image>
```

mariadb-backup will run as the `mysql` user in the container, so the permissions on `/backup` will need to ensure that it can be written to by this user:

```
$ docker exec --user mysql mariadb mariadb-backup --backup --target-dir=backup
```

Restore

These steps restore the backup made with mariadb-backup.

At some point before doing the restore, the backup needs to be prepared. The prepare must be done with the same MariaDB version that performed the backup. Perform the prepare like this:

```
$ docker run --rm \
  --name mariadb-restore \
  -v mariadb_backup:/backup \
  <mariadb-image> \
  mariadb-backup --prepare --target-dir=backup
```

Now that the image is prepared, start the container with both the data and the backup volumes and restore the backup. The data directory must be empty to perform this action:

```
$ docker volume create mariadb_restore
$ docker run --rm \
   -v mariadb_restore:/var/lib/mysql \
   --name mariadb-restore-change-permissions \
  <mariadb-image> \
  chown mysql: /var/lib/mysql
$ docker run --rm \
  --name mariadb-restore \
  -v mariadb_restore:/var/lib/mysql \
  -v mariadb_backup:/backup \
  --user mysql \
  <mariadb-image> \
  mariadb-backup --copy-back --target-dir=backup
```

With `mariadb_restore` volume containing the restored backup, start normally as this is an initialized data directory. At this point a later version of `<mariadb-image>` container can be used:

```
$ docker run -d --name mariadb \
  -v mariadb_restore:/var/lib/mysql \
  -e MARIADB_AUTO_UPGRADE=1 \
  -e MARIADB_ROOT_PASSWORD='MariaDB11!' \
  <mariadb-image>
```

On the environment variables here:

- MARIADB_AUTO_UPGRADE here in addition to upgrading the system tables ensures there is a healthcheck user.
- `MARIADB_ROOT_PASSWORD` is a convenience if any scripts, like logical backup above, use the environment variable. This environment variable is not strictly required.

For further information on mariadb-backup, see mariadb-backup Overview.

This page is licensed: CC BY-SA / Gnu FDL

Container Security

When using containers in production, it is important to be aware of container security concerns.

Host System Security

Depending on the container runtime, containers may be running on the host system's kernel or a kernel shared with other containers. If this kernel has security bugs, those bugs are also present in the containers. Malicious containers may attempt to explain a kernel vulnerability to impact the confidentiality, integrity or availability of other containers.

In particular, Linux based containers have a container runtime that can use the following features:

- Namespaces, to isolate containers from each other and make sure that a container can't establish unauthorized connections to another container.
- Seccomp security profiles.
- Rootless operation in Docker, or Rootless Podman
- cgroups, to limit the resources (CPU, memory, IO) that each container can consume.

The administrators of a system should be particularly careful to upgrade the kernel whenever security bugs to these features are fixed.

It is important to note that when we upgrade the kernel, runC or Docker itself we cause downtime for all the containers running on the system.

Images Security

Containers are built from images. If security is a major concern, you should make sure that the images you use are secure.

If you want to be sure that you are pulling authentic images, you should only pull images signed with Docker Content Trust. Signing only ensure authenticity or origin, it doesn't dictate that entity is trustworthy.

Updated images should be used. An image usually downloads packages information at build time. If the image is not recently built, a newly created container will have old packages. Updating the packages on container creation and regularly re-updating them will ensure that the container uses packages with the most recent versions. Rebuilding an image often will reduce the time necessary to update the packages the first time.

Security bugs are usually important for a database server, so you don't want your version of MariaDB to contain known security bugs. But suppose you also have a bug in Docker, in runC, or in the kernel. A bug in a user-facing application may allow an attacker to exploit a bug in those lower level technologies. So, after gaining access to the

container, an attacker may gain access to the host system. This is why system administrators should keep both the host system and the software running in the containers updated.

References

For more information, see the following links:

- Container Security from Red Hat.
- Docker security on Docker documentation.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Creating a Custom Container Image

OCI containers, frequently and incorrectly called Docker containers, are created from OCI images. An image contains software that can be launched, including the underlying system. A container is an instance of that software.

When we want to automate MariaDB, creating an image with MariaDB and the desired configuration, we may want to create an image by ourselves, which fulfils our needs.

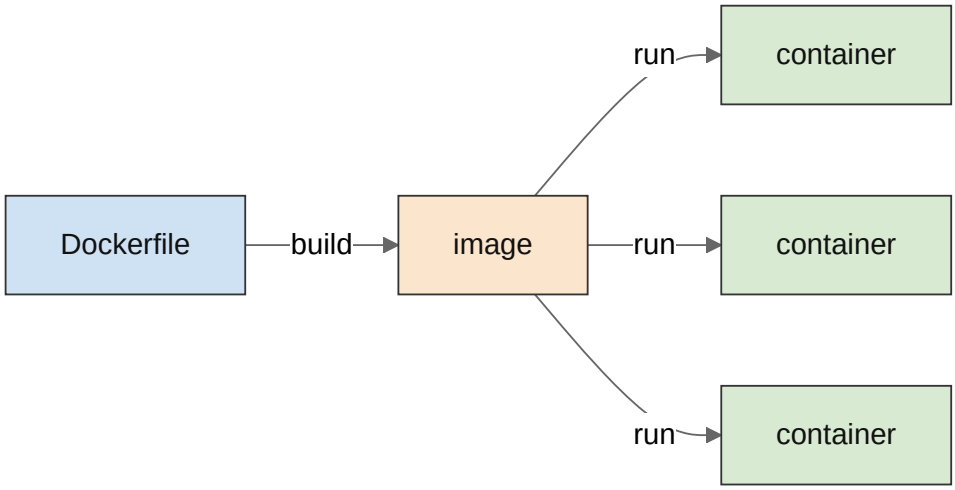
Images Architecture

One “source code” of an image is a Dockerfile. A Dockerfile is written in Docker specific language, and can be compiled into an image by the `docker` binary, using the `docker build` command. It can also be compiled by buildah using `buildah bud`.

Most images are based on another image. The base image is specified at the beginning of the Dockerfile, with the `FROM` directive. If the base image is not present in the local system, it is downloaded from the repository specified, or if not specified, from the default repository of the build program. This is often Docker Hub. For example, we can build a `mariadb-rocksdb:10.5` image starting from the `debian:13` image. In this way, we’ll have all the software included in a standard Debian image, and we’ll add MariaDB and its configuration upon that image.

All the following Dockerfile directives are compiled into a new Docker image, identified by an SHA256 string. Each of these images is based on the image compiled from the previous directive. A physical compiled image can serve as a base for any number of images. This mechanism saves a lot of disk space, download time and build time.

The following diagram shows the relationship between Dockerfiles, images and containers:



A Dockerfile is built into an image, which can then be run to create multiple containers.

Dockerfile Syntax

Here's a simple Dockerfile example:

```
FROM ubuntu:20.04

RUN apt-get update
RUN apt-get install -y mariadb-server

EXPOSE 3306

LABEL version="1.0"
LABEL description="MariaDB Server"

HEALTHCHECK --start-period=5m \
  CMD mariadb -e 'SELECT @@datadir;' || exit 1

CMD ["mariadbd"]
```

This example is not very good for practical purposes, but it shows what a Dockerfile looks like.

First, we declare that the base image to use is `ubuntu:20.04`.

Then we run some commands to install MariaDB from the Ubuntu default repositories and stop the MariaDB service.

We define some metadata about the image with `LABEL`. Any label is valid.

We declare that the port 3306 (MariaDB default port) should be exposed. However, this has no effect if the port is not exposed at container creation.

We also define a healthcheck. This is a command that is run to check if the container is healthy. If the return code is 0 the healthcheck succeeds, if it's 1 it fails. In the MariaDB specific case, we want to check that it's running and able to answer a simple query. This is better than just checking that MariaDB process is running, because MariaDB could be running but unable to respond, for example because max_connections was reached or data si corrupted. We read a system variable, because we should not assume that any user-created table exists. We also specify `--start-period` to allow some time for MariaDB to start, keeping in mind that restarting it may take some time if some data is corrupted. Note that there can be only one healthcheck: if the command is specified multiple times, only the last occurrence will take effect.

Finally, we start the container command: mariadbd. This command is run when a container based on this image starts. When the process stops or crashes, the container will immediately stop.

Note that, in a container, we normally run mariadbd directly or in an entrypoint script `exec mariadbd`, rather than running mysqld_safe or running MariaDB as a service. Containers restart can be handled by the container service. See automatic restart.

See the documentation links below to learn the syntax allowed in a Dockerfile.

Using Variables

It is possible to use variables in a Dockerfile. This allows us, for example, to install different packages, install different versions of a package, or configure software differently depending on how variables are set, without modifying the Dockerfile itself.

To use a variable, we can do something like this:

```
FROM ubuntu:20.04

ARG MARIADB_CONFIG_FILE

...

ENTRYPOINT mariadbd --defaults-file=$MARIADB_CONFIG_FILE
```

Here `ARG` is used after the `FROM` directive, thus the variable cannot be used in `FROM`. It is also possible to declare a variable before `FROM`, so we can use a variable to select the base image to use or its tag, but in this case the variable cannot be used after the `FROM` directive, unless `ARG` is redeclared after the `FROM`. Here is an example:

```
ARG UBUNTU_VERSION
FROM ubuntu:$UBUNTU_VERSION

# Uncomment for the build error to be avoided
# ARG UBUNTU_VERSION

# But this will cause a build error:
RUN echo 'Ubuntu version: $UBUNTU_VERSION' > /var/build_log
```

We'll have to assign variables a value when we build the Dockerfile, in this way:

```
docker build --build-arg UBUNTU_VERSION=20.04 .
```

Note that Dockerfile variables are just placeholders for values. Dockerfiles do not support assignment, conditionals or loops.

Versioning and Deploying Images

Dockerfiles are normally versioned, as well as the files that are copied to the images.

Once an image is built, it can be pushed to a container registry. Whenever an image is needed on a host to start containers from it, it is pulled from the registry.

Container registries

A default container registry for OCI images is Docker Hub. It contains Docker Official Images maintained by the Docker Library team and the community. Any individual or organization can open an account and push images to Docker Hub. Most Docker images are open source: the Dockerfiles and the needed files to build the images are usually on GitHub.

It is also possible to setup a self-hosted registry. Images can be pushed to that registry and pulled from it, instead of using Docker Hub. If the registry is not publicly accessible, it can be used to store images used by the organization without making them publicly available.

But a self-hosted registry can also be useful for open source images: if an image is available on Docker Hub and also on a self-hosted registry, in case Docker Hub is down or not reachable, it will still be possible to pull images.

Choosing Image Names and Tags

The names of images developed by the community follow this schema:

```
repository/maintainer/technology
```

It doesn't matter if the maintainer is an individual or an organization. For images available on Docker Hub, the maintainer is the name of a Docker Hub account.

Official images maintained by the Docker Library maintainers have the implicit name of `library` filled in by the container fetching tool. For example, the official MariaDB image is called `mariadb` which is an alias for `docker.io/library/mariadb`.

All images have a tag, which identifies the version or the variant of an image. For example, all MariaDB versions available on Docker are used as image tags. MariaDB 10.11 is called `mariadb:10.11`.

By conversion, tags form a hierarchy. So for example, there is a `10.1.1` tag whose meaning will not change over time. `10.5` will always identify the latest stable version in the 10.5 branch. For some time it was `10.5.1`, then it became `10.5.2`, and so on.

When we pull an image without specifying a tag (ie, `docker pull mariadb`), we are implicitly requiring the image with the `latest` tag. This is even more mutable: at different periods of time, it pointed to the latest `10.0` version, to the latest `10.1` version, and so on.

In production, it is always better to know for sure which version we are installing. Therefore it is better to specify a tag whose meaning won't change over time, like `10.5.21`. To keep to a latest LTS version, the `lts` can be used.

Pushing and Pulling Images

To pull an image from Docker Hub or a self-hosted registry, we use the `docker pull` command. For example:

```
docker pull mariadb:10.5
```

This command downloads the specified image if it is not already present in the system, or if the local version is not up to date.

After modifying a Dockerfile, we can build an image in this way:

```
docker build .
```

This step can be automated by services like Docker Hub and GitHub. Check those service's documentation to find out how this feature works.

Once an image is created, it can be pushed to a registry. We can do it in this way:

```
docker push <image_name>:<tag>
```

Docker Content Trust

Docker has a feature called Docker Content Trust (DCT). It is a system used to digitally sign images, based on PEM keys. For environments where security is a major concern, it is important to sign images before pushing them. This can be done with both Docker Hub and self-hosted registries.

Good Practices and Caveats

As mentioned, a Dockerfile is built by creating a new image for each directive that follows `FROM`. This leads to some considerations.

- Sometimes it can be a good idea to run several shell commands in a single `RUN` directive to avoid creating images that are not useful.
- Modifying a directive means that all subsequent directives also need to be rebuilt. When possible, directives that are expected to change often should follow directives that will change seldom.
- Directives like `LABEL` or `EXPOSE` should be placed close to the end of Dockerfiles. In this way they will be rebuilt often, but this operation is cheap. On the other side, changing a label should not trigger a long rebuild process.
- Variables should be used to avoid Dockerfiles proliferation. But if a variable is used, changing its value should be tested. So, be sure not to use variables without a good reason.

- Writing logic into a Dockerfile is impossible or very hard. Call shell scripts instead, and write your logic into them. For example, in a shell script it is easy to perform a certain operation only if a variable is set to a certain value.
- If you need MariaDB containers with different configurations or different sets of plugins, use the method explained above. Do not create several Dockerfiles, with different tags, for each desired configuration or plugin set. This may lead to undesired code duplication and increased maintenance costs.

References

More details can be found in the Docker documentation:

- Dockerfile reference.
- docker build.
- Repositories.
- Deploy a registry server.
- Content trust in Docker.

See also:

- Privacy-Enhanced Mail on Wikipedia.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Deploy MariaDB Enterprise Server with Docker

MariaDB Corporation provides Docker images for MariaDB Enterprise Server in the MariaDB Enterprise Docker Registry.

Docker provides multiple benefits:

- Docker is an open platform for developing, shipping, and running applications that allows you to separate your applications from your infrastructure.
- Docker images are portable. A Docker image can be deployed in a Docker container on any system using the Docker platform, regardless of the host operating system.
- Docker containers are isolated from the host operating system and from other Docker containers.

If you want to deploy MariaDB Enterprise Server without Docker, alternative deployment methods are available.

Use Cases

MariaDB Enterprise Server can be deployed with Docker to support use cases that require software to be rapidly deployed on existing infrastructure, such as:

- Continuously create and destroy automated testing environments as part of a continuous integration (CI) pipeline
- Create a small test environment on a local workstation
- Create multiple isolated test environments on the same host
- Deployment alongside related containers using Docker Compose

Compatibility

The following products and versions can be deployed using the MariaDB Enterprise Docker Registry:

- MariaDB Enterprise Server 11.8
- MariaDB Enterprise Server 11.4
- MariaDB Enterprise Server 10.6

For details about which storage engines and plugins are supported in the images for each version, see "MariaDB Enterprise Docker Registry".

Deploy Enterprise Server in a Docker Container

To deploy MariaDB Enterprise Server in a Docker container, follow the instructions below.

Step 1: Retrieve Customer Download Token

MariaDB Corporation requires customers to authenticate when logging in to the MariaDB Enterprise Docker Registry. A customer-specific Customer Download Token must be provided as the password.

Customer Download Tokens are available through the MariaDB Customer Portal.

To retrieve the customer download token for your account:

1. Navigate to the Customer Download Token at the MariaDB Customer Portal.
2. Log in using your MariaDB ID.
3. Copy the Customer Download Token to use as the password when logging in to the MariaDB Enterprise Docker Registry.

Step 2: Log In to Docker Registry

Log in to the MariaDB Enterprise Docker Registry by executing `docker login`:

```
$ docker login docker.mariadb.com
```

When prompted, enter the login details:

- As the user name, enter the email address associated with your MariaDB ID.
- As the password, enter your Customer Download Token.

The login details will be saved.

Confirm the login details were saved by checking the /.docker/config.json file for a JSON object named "docker.mariadb.com" inside an "auths" parent JSON object:

```
$ cat ~/.docker/config.json
```

```
{
   "auths": {
      "docker.mariadb.com": {
         "auth": "<auth_hash>"
      }
   }
}
```

Step 3: Choose an Image Tag

The `enterprise-server` repository in the MariaDB Enterprise Docker Registry contains images for different MariaDB Enterprise Server releases using specific tags. Before continuing, you will need to decide which tag to use.

To deploy a container using the most recent image for the latest MariaDB Enterprise Server release series (currently 11.8), use the `latest` tag.

For additional information, see "MariaDB Enterprise Docker Registry: Tags".

Step 4: Pull Docker Image

Pull the Docker image with the chosen tag by executing `docker pull`:

```
docker pull docker.mariadb.com/enterprise-server:latest
```

```
latest: Pulling from enterprise-server
5d87d5506868: Pull complete
Digest: sha256:68795ca747901e3402e30dab71d6d8bc72bce727db3b9e4888979468be77d250
Status: Downloaded newer image for docker.mariadb.com/enterprise-server:latest
docker.mariadb.com/enterprise-server:latest
```

Confirm the Docker image has been pulled by executing `docker images`:

```
docker images \
   --filter=reference='docker.mariadb.com/enterprise-server'
```

```
REPOSITORY                            TAG       IMAGE ID       CREATED        SIZE
docker.mariadb.com/enterprise-server   latest    dd17291aa340   3 months ago   451MB
```

Step 5: Create a Container

Create a container using the pulled Docker image by executing `docker run`:

```
docker run --detach \
   --name mariadb-es-latest \
   --env MARIADB_ROOT_PASSWORD='YourSecurePassword123!' \
   --publish '3307:3306/tcp' \
   docker.mariadb.com/enterprise-server:latest \
   --log-bin=mariadb-bin \
   <other mariadbd command-line options>
```

```
3082ab69e565be21c6157bb5a3d8c849ec03a2c51576778ac417a8a3aa9e7537
```

- Configure the container and set the root password using environment variables by setting the `--env` command-line option.
- Configure TCP port bindings for the container by setting the `--publish` or `--publish-all` command-line options.
- Configure MariaDB Enterprise Server by setting mariadbd command-line options.

Confirm the container is running by executing `docker ps`:

```
docker ps \
   --all \
   --filter ancestor='docker.mariadb.com/enterprise-server:latest'
```

```
CONTAINER ID   IMAGE                                          COMMAND                  CREATED
STATUS                     PORTS      NAMES
3082ab69e565   docker.mariadb.com/enterprise-server:latest   "/es-entrypoint.sh -…"   12 seconds ago   Up
11 seconds              3306/tcp   mariadb-es-latest
```

By default, Docker uses Docker bridge networking for new containers. For details on how to use host networking for new containers, see “Create a Container with Host Networking”.

Step 6: Connect to Container

Connect to the container by executing MariaDB Client on the container using docker exec:

```
docker exec --interactive --tty \
   mariadb-es-latest \
   mariadb \
   --user=root \
   --password
```

Confirm the container is using the correct version of MariaDB Enterprise Server by querying the version system variable with the SHOW GLOBAL VARIABLES statement:

```
SHOW GLOBAL VARIABLES
   LIKE 'version'\G
```

```
*************************** 1. row ***************************
Variable_name: version
        Value: 11.8.3-1-MariaDB-enterprise-log
```

Exit the container using `exit`:

```
exit
```

```
Bye
```

Step 7: Stop Container

Stop a Docker container using `docker stop`:

```
docker stop mariadb-es-latest
```

```
mariadb-es-latest
```

Confirm the container is stopped by executing `docker ps`:

```
docker ps \
   --all \
   --filter ancestor='docker.mariadb.com/enterprise-server:latest'
```

```
CONTAINER ID   IMAGE                                          COMMAND                  CREATED
STATUS                            PORTS       NAMES
3082ab69e565   docker.mariadb.com/enterprise-server:latest   "/es-entrypoint.sh -…"   2 minutes ago
Exited (143) About a minute ago               mariadb-es-latest
```

Step 8: Remove Container

Remove a Docker container using `docker rm`:

```
docker rm mariadb-es-latest
```

```
mariadb-es-latest
```

Confirm the container is removed by executing docker ps:

```
docker ps \
   --all \
   --filter ancestor='docker.mariadb.com/enterprise-server:latest'
```

```
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Docker Registry for MariaDB Enterprise Server

MariaDB Corporation provides the MariaDB Enterprise Docker Registry. The MariaDB Enterprise Docker Registry provides Docker images for MariaDB Enterprise Server.

This page contains reference material for the MariaDB Enterprise Docker Registry.

The reference material on this page applies to MariaDB Enterprise Server 10.6 and later.

Details

Docker is an open platform for developing, shipping, and running applications that allows you to separate your applications from your infrastructure.

MariaDB Corporation provides the MariaDB Enterprise Docker Registry. The MariaDB Enterprise Docker Registry provides Docker images for MariaDB Enterprise Server.

For examples that show how to use the Docker images in the MariaDB Enterprise Docker Registry for MariaDB Enterprise Server, see Examples.

Other details are listed in the sections below.

- Versions
- Storage Engines
- Plugins
- Repositories
- Tags

Versions

The MariaDB Enterprise Docker Registry provides Docker images for **MariaDB Enterprise Server 10.6 and later**.

Storage Engines

The Docker images for MariaDB Enterprise Server include all storage engines that are installed with MariaDB Enterprise Server by default. The following storage engines are currently included:

- Aria
- CSV
- InnoDB
- MEMORY
- MRG_MYISAM
- MyISAM

You can check which storage engines are installed by connecting to the Docker container and executing the SHOW ENGINES statement.

Plugins

The Docker images for MariaDB Enterprise Server include all plugins that are installed with MariaDB Enterprise Server by default.

You can check which plugins are installed by connecting to the Docker container and executing the SHOW PLUGINS statement.

The container contains shared libraries for additional plugins that can be installed with the INSTALL SONAME statement or the –plugin-load-add command-line option. You can check which plugins are available by querying the information_schema.ALL_PLUGINS table:

```
SELECT PLUGIN_NAME, PLUGIN_VERSION, PLUGIN_LIBRARY
FROM information_schema.ALL_PLUGINS
WHERE PLUGIN_STATUS = 'NOT INSTALLED';
```

```
+---------------------------+----------------+------------------------+
| PLUGIN_NAME               | PLUGIN_VERSION | PLUGIN_LIBRARY         |
+---------------------------+----------------+------------------------+
| pam                       | 1.0            | auth_pam_v1.so         |
| LOCALES                   | 1.0            | locales.so             |
| BLACKHOLE                 | 1.0            | ha_blackhole.so        |
| SQL_ERROR_LOG             | 1.0            | sql_errlog.so          |
| METADATA_LOCK_INFO        | 0.1            | metadata_lock_info.so  |
| pam                       | 2.0            | auth_pam.so            |
| SERVER_AUDIT              | 1.4            | server_audit.so        |
| file_key_management       | 1.0            | file_key_management.so |
| QUERY_RESPONSE_TIME       | 1.0            | query_response_time.so |
| QUERY_RESPONSE_TIME_AUDIT | 1.0            | query_response_time.so |
+---------------------------+----------------+------------------------+
```

Repositories

The MariaDB Enterprise Docker Registry contains a single repository, which provides images for MariaDB Enterprise Server:

- `enterprise-server`

Tags

The `enterprise-server` repository in the MariaDB Enterprise Docker Registry contains images for different MariaDB Enterprise Server releases using specific tags:

Type of release?	Tags	Description
Latest release series		This tag refers to the most recent image for the latest MariaDB Enterprise Server release series, which is currently MariaDB Enterprise Server 11.8.
Specific release series		These tags refer to the images for the most recent minor release of each specific MariaDB Enterprise Server release series.
Specific minor release		These tags refer to images for specific MariaDB Enterprise Server minor releases. The listed tags are examples of minor releases. For a full list of minor releases, see MariaDB Enterprise Server All Releases.

Examples

- Connect to Container with Docker Bridge Networking
- Connect to Container with Host Networking
- Create a Container with Docker Bridge Networking
- Create a Container with Host Networking
- Environment Variables
- Inspect Container
- Log In to Docker Registry
- Obtain a Shell in Container

- Pull Docker Image
- Remove Container
- Stop Container
- View Container Logs

Connect to Container with Docker Bridge Networking

To connect to a Docker container that uses Docker bridge networking, execute MariaDB Client on the container using `docker exec`:

```
docker exec --interactive --tty \
   mariadb-es-11.4 \
   mariadb \
   --user=root \
   --password
```

To confirm the client is connected to the Docker container and the container is using the correct version of MariaDB Enterprise Server, query the version system variable with the SHOW GLOBAL VARIABLES statement:

```
SHOW GLOBAL VARIABLES
   LIKE 'version'\G
```

```
*************************** 1. row ***************************
Variable_name: version
        Value: 11.4.5-3-MariaDB-enterprise-log
```

To exit the container, use `exit`:

```
exit
```

```
Bye
```

The example above shows how to connect to a Docker container using MariaDB Client on the container, but you can also connect using MariaDB Client on the host using TCP/IP. You can inspect the Docker container to find the container's IP address and TCP port bindings.

Connect to Container with Host Networking

To connect to a Docker container that uses host networking, execute MariaDB Client on the host node and specify the host's IP address and the container's port:

```
mariadb --host=192.0.2.1 \
   --port=3307 \
   --user=root \
   --password
```

To confirm the client is connected to the Docker container and the container is using the correct version of MariaDB Enterprise Server, query the version system variable with the SHOW GLOBAL VARIABLES statement:

```
SHOW GLOBAL VARIABLES
   LIKE 'version'\G
```

```
*************************** 1. row ***************************
Variable_name: version
        Value: 11.8.3-1-MariaDB-enterprise-log
```

To exit the container, use `exit`:

```
exit
```

```
Bye
```

Create a Container with Docker Bridge Networking

By default, Docker containers use Docker bridge networking. To make connecting to the container easier, Docker can use port bindings to publish the container's TCP ports to the host.

To create a Docker container using Docker bridge networking, execute `docker run`:

```
docker run --detach \
   --name mariadb-es-11.8 \
   --env MARIADB_ROOT_PASSWORD='Password123!' \
   --publish '3307:3306/tcp' \
   docker.mariadb.com/enterprise-server:11.8 \
   --log-bin=mariadb-bin \
   <other mariadbd command-line options>
```

```
3082ab69e565be21c6157bb5a3d8c849ec03a2c51576778ac417a8a3aa9e7537
```

- Configure the container using environment variables by setting the `--env` command-line option.
- Configure TCP port bindings for the container by setting the `--publish` or `--publish-all` command-line options.
- Configure MariaDB Enterprise Server by setting mariadbd command-line options.

To confirm the Docker container is running, execute `docker ps`:

```
docker ps \
   --all \
   --filter ancestor='docker.mariadb.com/enterprise-server:11.8'
```

```
CONTAINER ID   IMAGE                                          COMMAND                  CREATED
STATUS                     PORTS      NAMES
3082ab69e565   docker.mariadb.com/enterprise-server:11.8   "/es-entrypoint.sh -…"   12 seconds ago   Up
11 seconds                3306/tcp   mariadb-es-11.8
```

Create a Container with Host Networking

A Docker container can be configured to use the host operating system's network.

To create a Docker container using host networking, execute `docker run` and specify the `--network host` option:

```
docker run --detach \
   --network host \
   --name mariadb-es-11.8 \
   --env MARIADB_ROOT_PASSWORD='Password123!' \
   docker.mariadb.com/enterprise-server:11.8 \
   --port=3307 \
   --log-bin=mariadb-bin \
   <other mariadbd command-line options>
```

```
3082ab69e565be21c6157bb5a3d8c849ec03a2c51576778ac417a8a3aa9e7537
```

- Configure the container using environment variables by setting the `--env` command-line option.
- Configure the port for MariaDB Enterprise Server by setting the –port command-line option.
- Configure MariaDB Enterprise Server by setting mariadbd command-line options.

To confirm the Docker container is running, execute `docker ps`:

```
docker ps \
   --all \
   --filter ancestor='docker.mariadb.com/enterprise-server:11.8'
```

Environment Variables

A Docker container can be configured to perform the following tasks using environment variables:

- Create a custom database
- Create a custom database user account with a custom password
- Initialize the timezone tables
- Set the host for the `root` database user
- Set the password for the `root` database user

The following table contains details about the environment variables that are supported:

Environment Variable	Description	Default	Allowed Values
`MARIADB_DATABASE`	When the `MARIADB_DATABASE` environment variable is set to a valid database name, create the specified database if it does not already exist.	No default	Any valid database name
`MARIADB_INITDB_SKIP_TZINFO`	By default, the entrypoint script automatically loads the timezone data needed for the `CONVERT_TZ()` function. If it is not needed, any non-empty value disables timezone loading.	No default	
`MARIADB_PASSWORD`	When the `MARIADB_USER` and `MARIADB_PASSWORD` environment variables are both set, create a user account with the specified user name and password.	No default	Any valid password
`MARIADB_ROOT_HOST`	When the `MARIADB_ROOT_HOST` environment variable is set to a valid hostname, the `root` user account will be restricted to logins from that hostname.	`%`	Any valid hostname
`MARIADB_ROOT_PASSWORD`	When the `MARIADB_ROOT_PASSWORD` environment variable is set to `RANDOM`, a random password is generated for the `root` user account. When the `MARIADB_ROOT_PASSWORD` environment variable is set to `EMPTY`, the `root` user account uses an empty password. When the `MARIADB_ROOT_PASSWORD` environment variable is set to a valid password, the `root` user account uses the specified password.	No default	
`MARIADB_USER`	When the `MARIADB_USER` and `MARIADB_PASSWORD` environment variables are both set, create a user account with the specified user name and password.	No default	Any valid user name
`MARIADB_UNIX_SOCKET_AUTHENTICATION`	When set to 1, enables Unix socket-based authentication for the root user, allowing passwordless access when connecting locally as the mysql system user. This option is mutually exclusive with all MARIADB_ROOT_PASSWORD* variables.	Empty	1,0
`SKIP_INIT_WSREP_OFF`	By default, the temporary server used for initialization is started with Galera Cluster replication explicitly turned off (–wsrep_on=OFF). If this variable is set, that flag is omitted, allowing the temporary server to start with wsrep enabled.	No default	Any non-empty value

To create a Docker container using environment variables, execute `docker run`, environment variables and specify each environment variable using the `--env` option:

```
docker run --detach \
   --network host \
   --name mariadb-es-11.8 \
   --env MARIADB_ROOT_PASSWORD='Password123!' \
   docker.mariadb.com/enterprise-server:11.8
```

```
3082ab69e565be21c6157bb5a3d8c849ec03a2c51576778ac417a8a3aa9e7537
```

Inspect Container

A Docker contained can be inspected to find out internal details about the container, such as the following details:

- IP addresses
- MAC addresses
- Port bindings

To inspect all internal details about a Docker container, execute `docker inspect`:

```
docker inspect mariadb-es-11.8
```

```
[
    {
        "Id": "d784a193ad828cbd3db10dfbef3e3b7274465fa587453c3f6e2e27703b361173",
        "Created": "2021-10-13T20:25:04.994759266Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "--log-bin=mariadb-bin"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 33714,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-10-13T20:25:05.32049779Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        ..
    }
]
```

To inspect a specific internal detail about a Docker container, execute `docker inspect` and specify a filter using the `--format` option. Some examples are shown below.

To inspect the container's IP address:

```
docker inspect \
   --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' \
   mariadb-es-11.8
```

```
172.17.0.3
```

To inspect the container's TCP port bindings:

```
docker inspect \
   --format='{{range $p, $conf := .NetworkSettings.Ports}}Container port: {{$p}} -> Host port: {{(index
$conf 0).HostPort}} {{end}}' \
   mariadb-es-11.8
```

```
Container port: 3306/tcp -> Host port: 3307
```

Log In to Docker Registry

To log in to the MariaDB Enterprise Docker Registry, execute `docker login`:

```
docker login docker.mariadb.com
```

When prompted, enter the login details:

- As the user name, enter the email address associated with your MariaDB ID.
- As the password, enter your Customer Download Token.

The login details will be saved.

To confirm the login details were saved, check the `~/.docker/config.json` file for a JSON object named `"docker.mariadb.com"` inside an `"auths"` parent JSON object:

```
cat ~/.docker/config.json
{
   "auths": {
      "docker.mariadb.com": {
         "auth": "<auth_hash>"
      }
   }
}
```

Obtain a Shell in Container

To obtain a shell in the Docker container, execute the shell on the container using `docker exec`:

```
docker exec --interactive --tty \
   mariadb-es-11.8 \
   bash
```

Pull Docker Image

To pull a Docker image with the appropriate tag, execute `docker pull`:

```
docker pull docker.mariadb.com/enterprise-server:11.8
```

```
11.8: Pulling from enterprise-server
5d87d5506868: Pull complete
Digest: sha256:68795ca747901e3402e30dab71d6d8bc72bce727db3b9e4888979468be77d250
Status: Downloaded newer image for docker.mariadb.com/enterprise-server:11.8
docker.mariadb.com/enterprise-server:11.8
```

To confirm the Docker image has been pulled, execute `docker images`:

```
docker images \
   --filter=reference='docker.mariadb.com/enterprise-server'
```

```
REPOSITORY                            TAG       IMAGE ID       CREATED        SIZE
docker.mariadb.com/enterprise-server   11.8      dd17291aa340   3 months ago   451MB
```

Remove Container

To remove a Docker container, execute `docker rm`:

```
docker rm mariadb-es-11.8
```

```
mariadb-es-11.8
```

To confirm the container is removed, execute `docker ps`:

```
docker ps \
   --all \
   --filter ancestor='docker.mariadb.com/enterprise-server:11.8'
```

```
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Stop Container

To stop a Docker container, execute `docker stop`:

```
docker stop mariadb-es-11.8
```

```
mariadb-es-11.8
```

To confirm the container is stopped, execute `docker ps`:

```
docker ps \
   --all \
   --filter ancestor='docker.mariadb.com/enterprise-server:11.8'
```

```
CONTAINER ID   IMAGE                                          COMMAND                  CREATED
STATUS                             PORTS       NAMES
3082ab69e565   docker.mariadb.com/enterprise-server:11.8   "/es-entrypoint.sh -…"   2 minutes ago
Exited (143) About a minute ago               mariadb-es-11.8
```

View Container Logs

To view the logs in the Docker container, execute `docker logs`:

```
docker logs mariadb-es-11.8
```

This page is licensed: CC BY-SA / Gnu FDL

Docker and AWS EC2

This process shows how to deploy MariaDB in a Docker container running on an EC2 instance. First we'll create the EC2 VM, then we'll deploy Docker to it. After that, we'll pull the MariaDB Docker image which we'll use to create a running container with a MariaDB instance. Finally, we'll load a sample database into the MariaDB instance.

Create a VM in AWS EC2

1. Install MariaDB client on your local machine, either bundled with Maria DB server or standalone.
2. Login to AWS, navigate to EC2 service home
3. Choose Region for EC2 in the upper right corner of the console
4. Launch (1) Instance, giving instance a name (e.g. mrdb-ubuntu-docker-use1) and create or reuse a key pair
5. Choose Ubuntu 22.04 or similar free tier instance

6. Choose hardware, t2.micro or similar free tier instance
7. Create Key Pair with name (e.g. mrdb-docker-aws-pk.pem if using openSSH at the command line, or mrdb-docker-aws-pk..ppk for use with programs like PuTTY.)
8. Create or select a security group where SSH is allowed from anywhere 0.0.0.0/0. If you'd like to make this more secure, it can be restricted to a specific IP address or CIDR block.

```
{{aws-firewall}}
```

10. Accept remaining instance creation defaults and click "launch instance".
11. Save the *.pem or *.ppk keyfile on your local hard drive when prompted. You will need it later. If you're on Linux, don't forget to change permissions on the downloaded *.pem / *.ppk key file:

```
$ chmod 400 mrdb-docker-pk.pem
```

12. Click into the instance summary (EC2 > Instances > Instance ID) and click on the "security" tab towards the bottom.

```
{{security-group}}
```

13. In the relevant security group for your instance, Create an inbound rule so that TCP port 3306 is open, allowing external connections to Maria DB (like your local command line client for MariaDB). Double check that port 22 is open while you're there for SSH.

Install Docker on the EC2 VM

For more detailed instructions, refer to Installing and Using MariaDB via Docker

14. Back in the instance summary (EC2 > Instances > Instance ID), copy the public IP (e.g. ww.xx.yyy.zzz)

```
{{aws-instance-ip}}
```

15. Open terminal window, navigate to the directory with private key (*.pem or *.ppk) file and start a SSH remote shell session by typing:

```
$ ssh -i mrdb-docker-pk.pem ubuntu@ww.xx.yyy.zzz
```

(switch ww.xx.yyy.zzz for your IP address from step 14).

16. Are you sure you want to continue connecting (yes/no/[fingerprint])?
    Say yes
17. Escalate to root

```
$ sudo su
```

18. Install Docker

```
$ curl -fsSL https://get.docker.com | sudo sh
```

Pull the Mariadb Docker Image and Create the Container

19. Pull MariaDB Docker image

```
$ docker pull mariadb:lts
```

20. Start MDRB docker process at your terminal / command line

```
$ docker run --detach --name mariadb-docker -v \Users\YouUID\Documents\YourDirName:/var/lib/mysql:Z -p
3306:3306 -e MARIADB_ROOT_PASSWORD=yoursecurepassword mariadb:lts
```

To ensure persistent storage, use the `-v` flag to mount your chosen directory as `/var/lib/mysql`. For Windows, represent file paths like `C:\Users\YourUID\Documents\YourDirName` accordingly. On Linux, use absolute file paths. Always replace the root password with a secure one, especially outside of development environments.

21. Shell into container

```
$ docker exec -it mariadb-docker bash
```

22. Login to MRDB inside container using the root password specified in step 20.

```
$ mariadb -pyoursecurepassword
```

23. Setup admin account with permission for remote connection, configure access control

```
MariaDB [(none)]> CREATE USER 'admin'@'%' IDENTIFIED BY 'admin';

MariaDB [(none)]> GRANT ALL ON *.* to 'admin'@'%' WITH GRANT OPTION;

MariaDB [(none)]> SHOW GRANTS FOR admin;
```

Obviously replace these passwords with something that is a bit more secure than you see in this example for anything other than development purposes.

24. Setup service account for your app with permission for remote connection, configure access control

```
MariaDB [(none)]> CREATE USER 'yourappname'@'%' IDENTIFIED BY 'yoursecurepassword';

MariaDB [(none)]> GRANT INSERT, UPDATE, DELETE ON *.* to 'yourappname'@'%';

MariaDB [(none)]> SHOW GRANTS FOR yourappname;
```

Obviously replace these passwords with something that is a bit more secure than you see in this example for anything other than development purposes.

25. Load up your database from your preexisting SQL script that contains CREATE DATABASE; USE DATABASE; and CREATE TABLE statements.

Open a new terminal window (not your SSH session) and navigate to the directory containing your database creation script, for example, `init.sql`. Then, enter the command:

```
$ mariadb --host=ww.xx.yyy.zzz --port=3306 --user=admin --password=admin -e “SOURCE init.sql”
```

(switch ww.xx.yyy.zzz for your IP address from step 14).

This page is licensed: CC BY-SA / Gnu FDL

Docker and Google Cloud

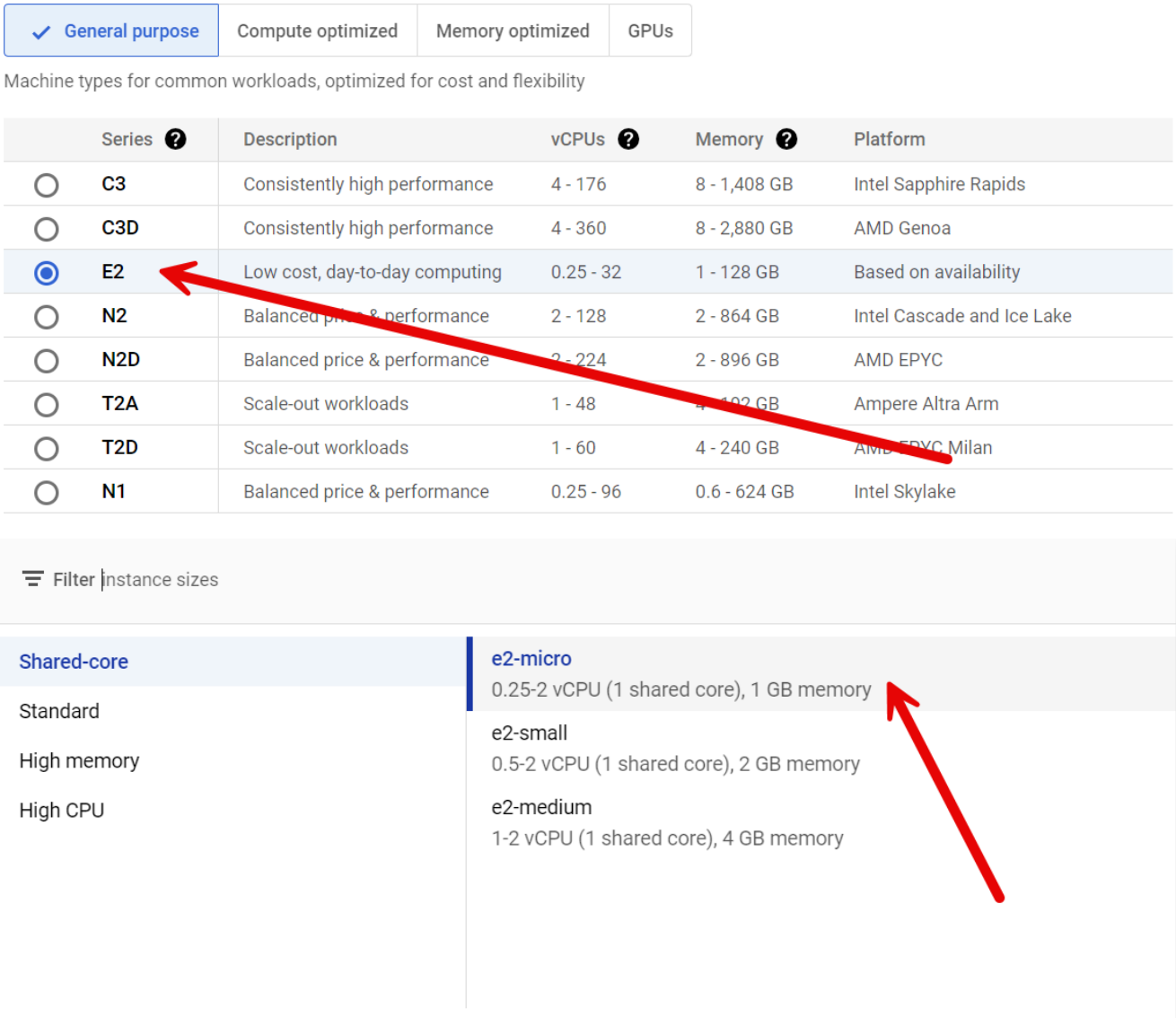
This process shows how to deploy MariaDB in a Docker container running on an GCE instance. First we’ll create the GCE VM, then we’ll deploy Docker to it. After that, we’ll pull the MariaDB Docker image which we’ll use to create a running container with a MariaDB instance. Finally, we’ll load a sample database into the MariaDB instance.

Create a VM in Google Cloud Compute Engine

1. Install MariaDB client on your local machine, either bundled with Maria DB server or standalone.
2. Login to Google Cloud, navigate to VM instances

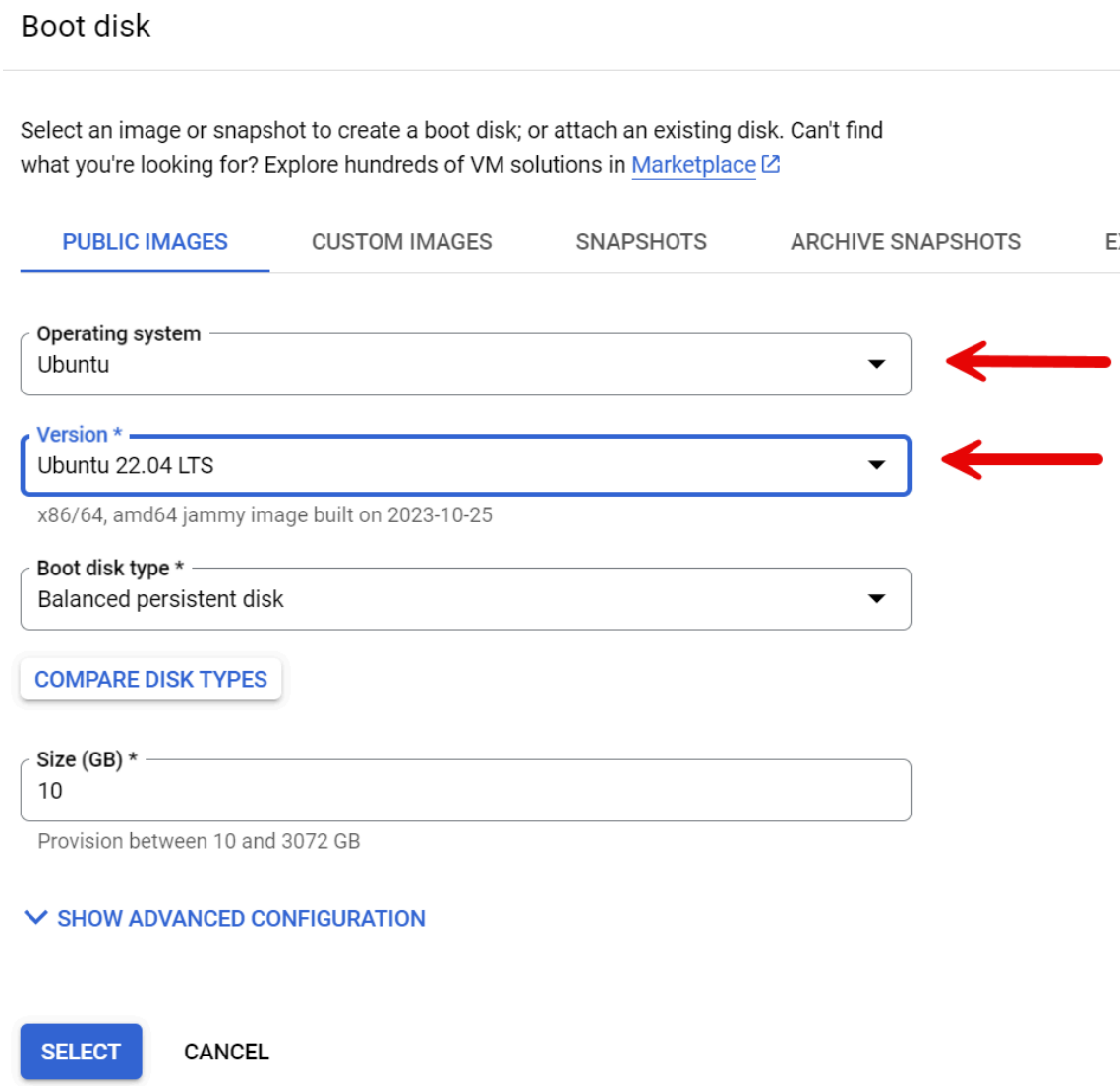
3. Enable Compute Engine API if you haven't already.
4. Click create instance, give instance a name (e.g. mrdb-ubuntu-docker-use1b), choose a region and zone.
5. Machine configuration: Choose general-purpose / E2 micro

Machine configuration



6. Boot Disk > Change

Switch the operating system to a modern Ubuntu release x86/64 CPU architecture, or similar free tier offering.



7. Create a firewall rule in the Firewall Policies section of the console. After naming it, change the targets, add 0.0.0.0/0 as a source IP range, and open TCP port 3306. Then Click create.

Direction of traffic

- (•) Ingress
- ( ) Egress

Action on match

- (•) Allow
- ( ) Deny

Targets
All instances in the network

Source filter
IPv4 ranges

Source IPv4 ranges *
0.0.0.0/0

Second source filter
None

Destination filter
None

Protocols and ports

- ( ) Allow all
- (•) Specified protocols and ports

[x] TCP

Ports
3306

E.g. 20, 50-60

[ ] UDP

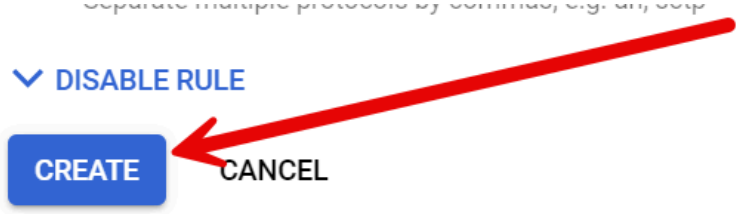
Ports

E.g. all

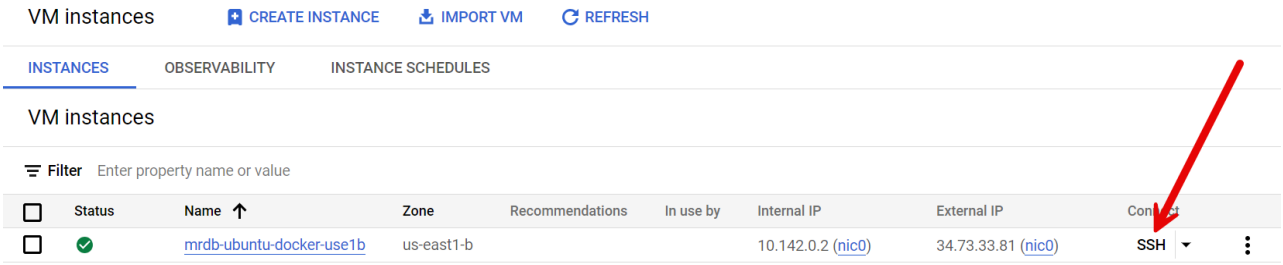
[ ] Other

Protocols

Separate multiple protocols by commas, e.g. ah, sctp



8. Connect using Google Cloud's built in browser SSH. Accept all prompts for authorization.



Install Docker on the GCE VM

For more detailed instructions, refer to Installing and Using MariaDB via Docker

9. Escalate to root Escalate to root

```
$ sudo su
```

10. Install Docker

```
$ curl -fsSL get.docker.com | sudo sh
```

11. Pull Docker image

```
$ docker pull mariadb:lts
```

12. Start MDRB docker process at your terminal / command line.

```
$ docker run --detach --name mariadb-docker -v \Users\YouUID\Documents\YourDirName:/var/lib/mysql:Z -p
3306:3306 -e MARIADB_ROOT_PASSWORD=yoursecurepassword mariadb:lts
```

Mounting a Directory for Persistent MySQL Storage

When using the `-v` flag to mount a directory as `/var/lib/mysql`, you ensure that your MySQL data is stored persistently.

File Path Instructions:

- **Windows**: Use a file path format similar to `C:\Users\YourUsername\Documents\YourDirectory`.
- **Linux**: Always use absolute paths instead of relative ones.

> **Important:**
>
> **Root Password**: Replace the default root password with a strong, secure password for production environments.

13. Shell into container

```
docker exec -it mariadb-docker bash
```

14. Login to MRDB inside container using the root password specified in step 12.

```
$ mariadb -pyoursecurepassword
```

15. Setup admin account with permission for remote connection, configure access control
    Execute these SQL commands in sequence:

```
MariaDB [(none)]> CREATE USER 'admin'@'%' IDENTIFIED BY 'admin';
MariaDB [(none)]> GRANT ALL ON _._ to 'admin'@'%' WITH GRANT OPTION;
MariaDB [(none)]> SHOW GRANTS FOR admin;
```

Obviously replace these passwords with something that is a bit more secure than you see in this example for anything other than development purposes.

16. Setup service account for your app with permission for remote connection, configure access control
    Execute these SQL commands in sequence:

```
MariaDB [(none)]> CREATE USER 'yourappname'@'%' IDENTIFIED BY 'yoursecurepassword';
MariaDB [(none)]> GRANT INSERT, UPDATE, DELETE ON *.* TO 'yourappname'@'%';
MariaDB [(none)]> SHOW GRANTS FOR yourappname;
```

Obviously replace these passwords with something that is a bit more secure than you see in this example for anything other than development purposes.

17. Load up your database from your preexisting SQL script that contains CREATE DATABASE; USE DATABASE; and CREATE TABLE statements.

Copy the external IP address of your VM instance from the Console in the VM instances list.

To run your database creation script using MariaDB, open a new terminal window and navigate to the directory containing the script, `init.sql`. Then, execute the following command, replacing `ww.xx.yyy.zzz` with your IP address:

```
$ mariadb --host=your_ip_address --port=3306 --user=admin --password=admin -e "SOURCE init.sql"
```

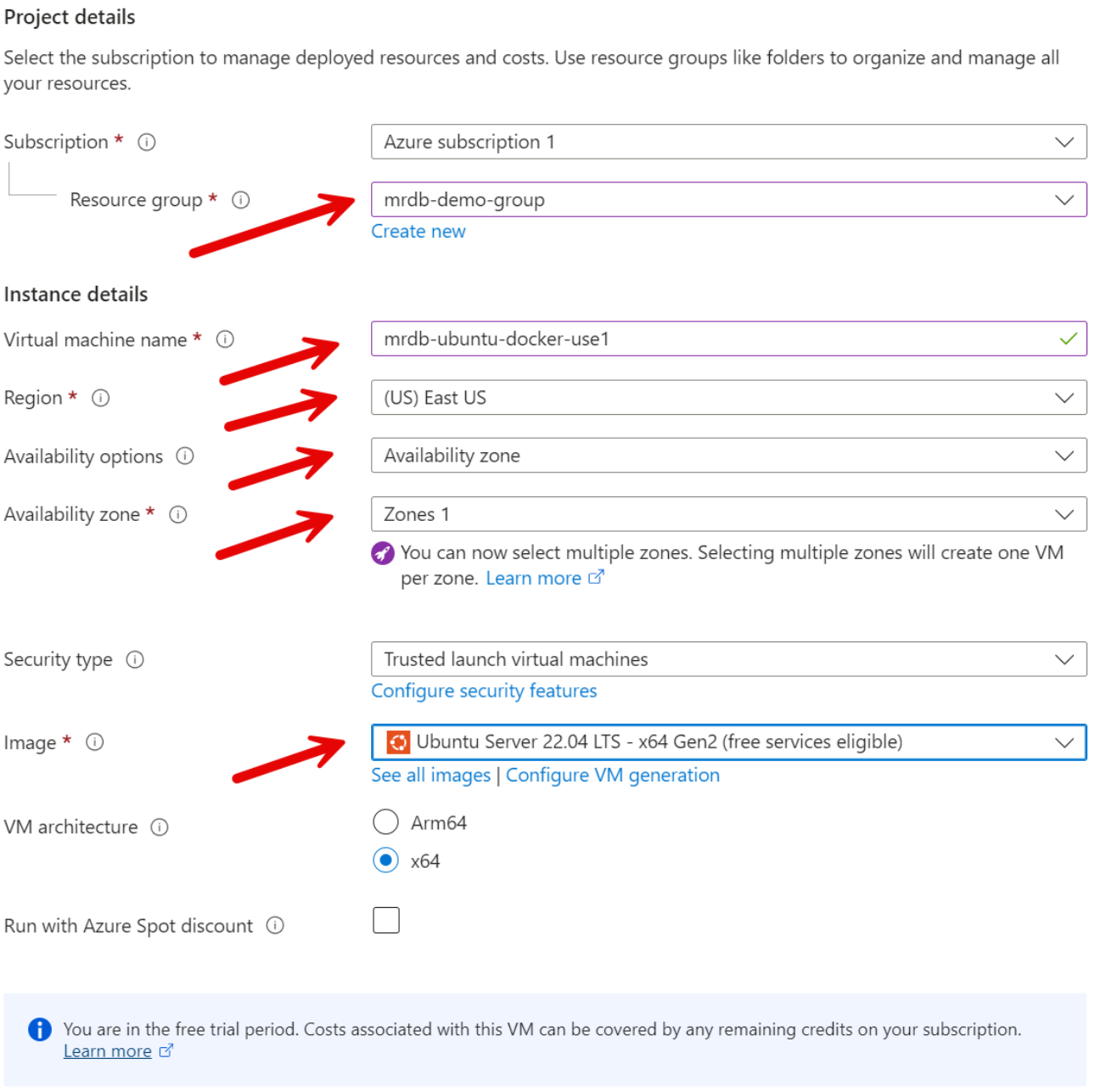
This page is licensed: CC BY-SA / Gnu FDL

Docker and Microsoft Azure

This process shows how to deploy MariaDB in a Docker container running on an Azure VM instance. First we'll create the Azure VM, then we'll deploy Docker to it. After that, we'll pull the MariaDB Docker image which we'll use to create a running container with a MariaDB instance. Finally, we'll load a sample database into the MariaDB instance.

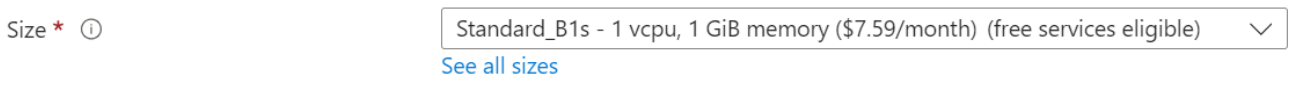
Create a VM in Azure

1. Install MariaDB client on your local machine, either bundled with Maria DB server or standalone.
2. Login to Azure, navigate to Azure Virtual Machine
3. Create VM. Give the VM a name (e.g. mrdb-ubuntu-docker-use1), and create new or use an existing resource group. Selection region and availability zone, and choose Ubuntu 22.04 LTS x64 (free services eligible).



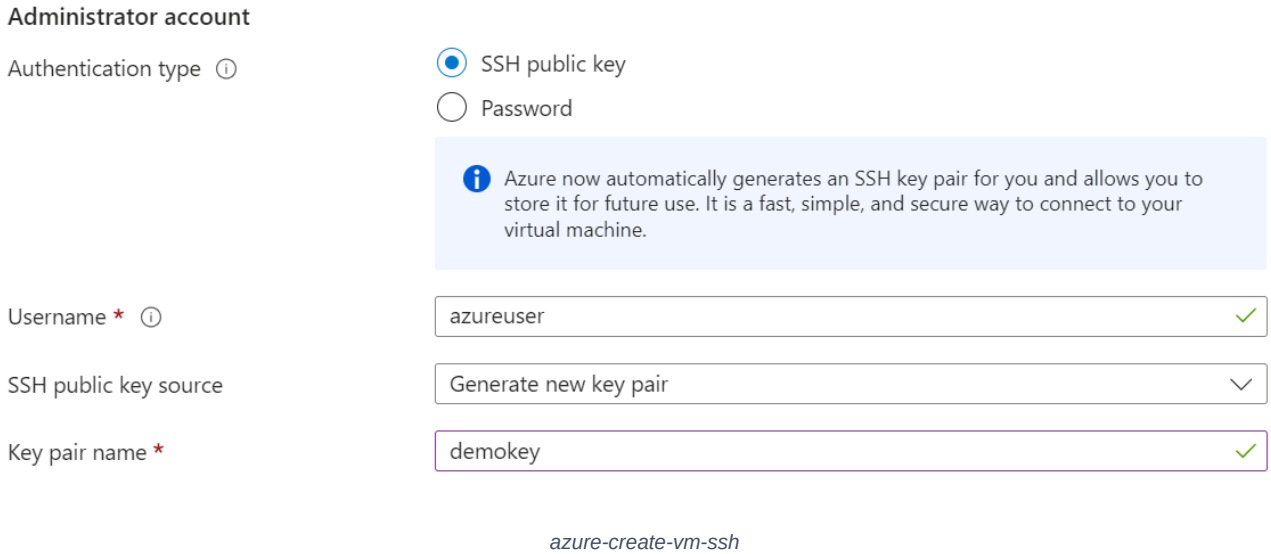
azure-create-vm

4. Choose the VM instance size, like a B1s or similar free tier. Note that Azure free works on a credit based system for new accounts



azure-create-vm-size

5. Configure an administrator account and generate a new key pair, and give the key pair a name.



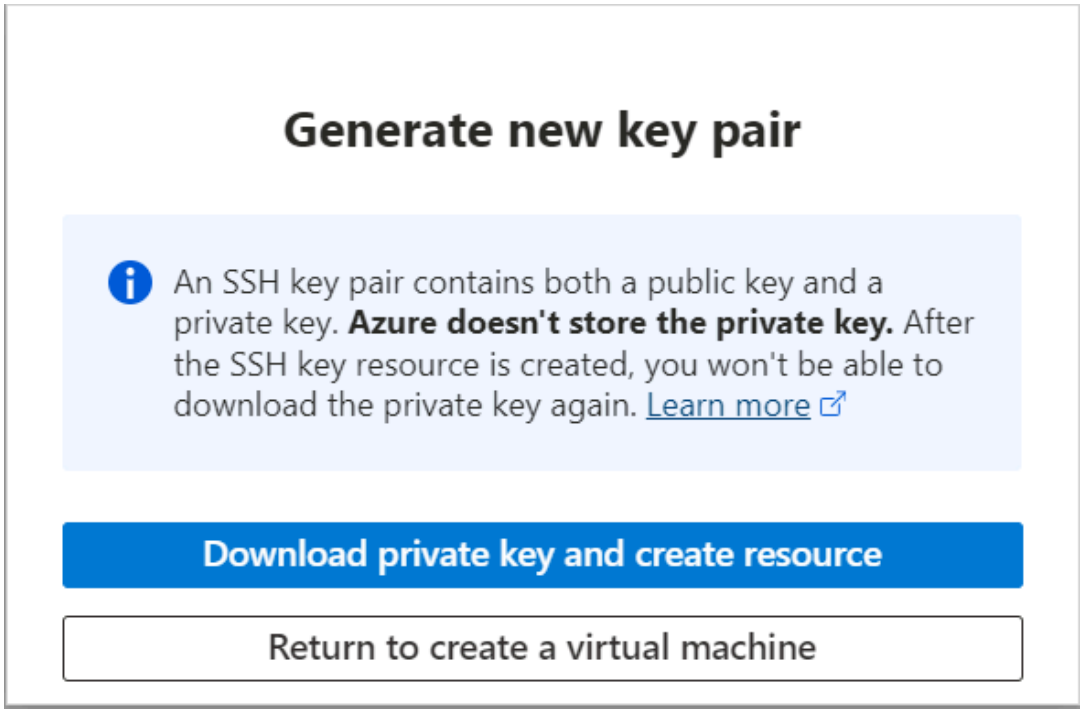
azure-create-vm-ssh

6. Click “Review + Create” at the very bottom of the “create virtual machine” page to create the VM.

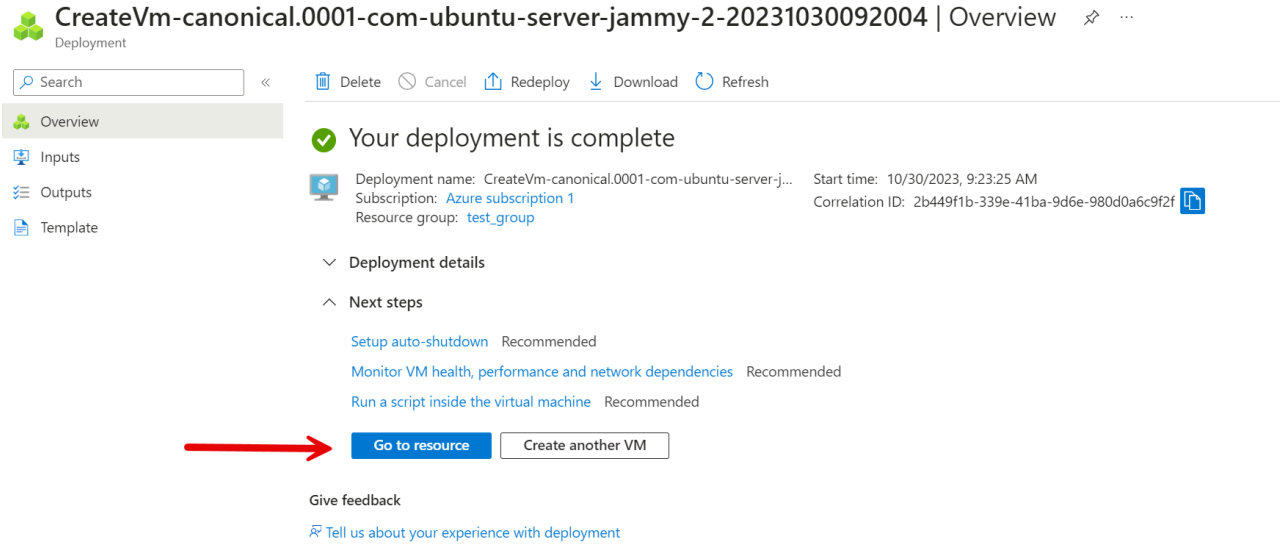
7. Download the SSH keys and them in a safe place, you will need them later. For this example, let’s name the key file `mrdb-docker-pk.pem`.

If your local machine is Linux or you are using WSL on Windows, open a terminal window and:

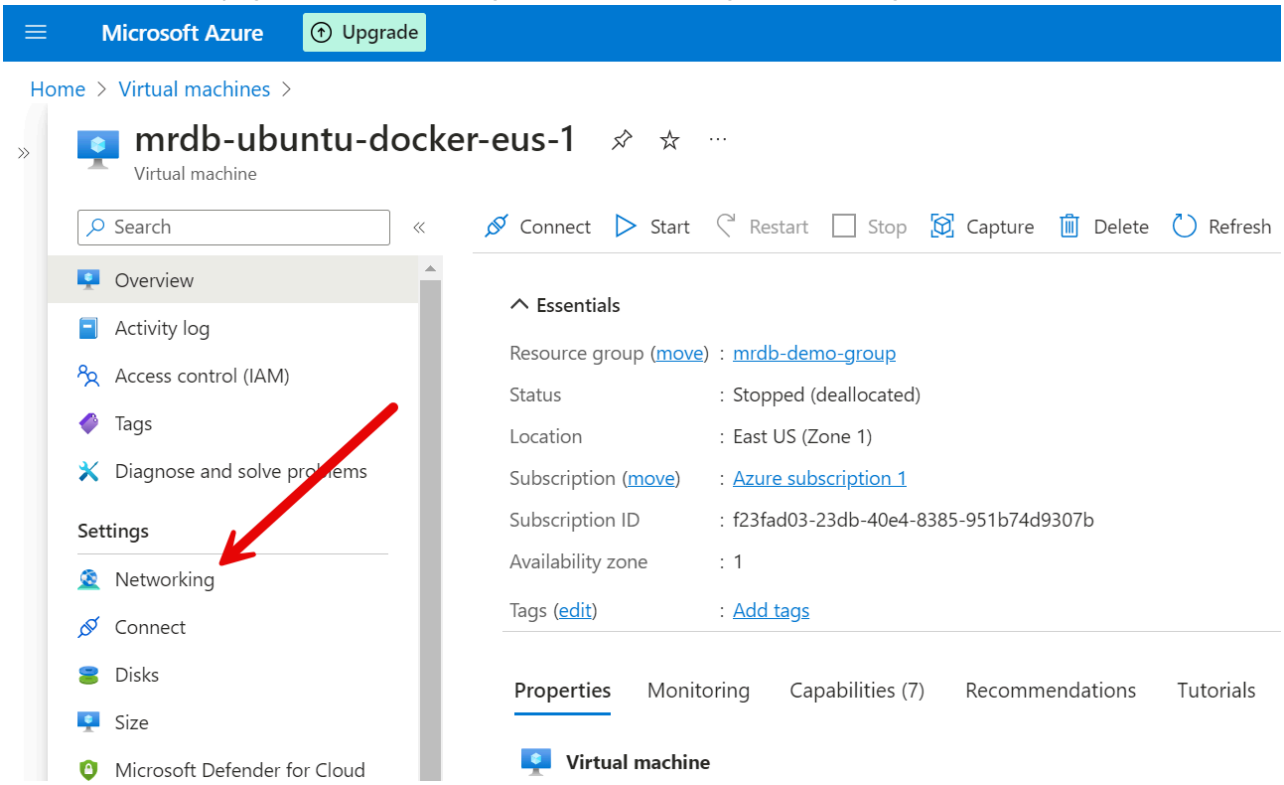
```
mv /mnt/c/ ~/.ssh/
chmod 400 ~/.ssh/
```



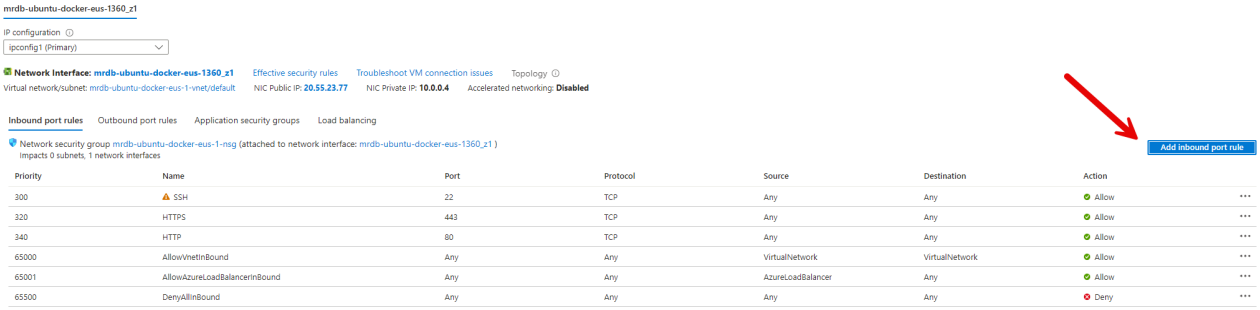
8. Once the VM is deployed, “click to resource” to get back to the virtual machine’s overview page.



9. From the overview page, the left-hand navigation, choose settings > networking.



10. Click “add inbound port rule”



11. Configure the port rule to allow port TCP 3306 inbound (mySQL) so that you can make external connections from your local Maria DB command line client, to the dockerized Maria DB instance in your Azure Linux VM.

AllowCidrBlockCustom3306Inbound

mrdb-ubuntu-docker-eus-1-nsg

Source
Any

Source port ranges *
*

Destination
Any

Service
MySQL

Destination port ranges
3306

Protocol
- Any
- TCP
- UDP
- ICMP

Action
- Allow
- Deny

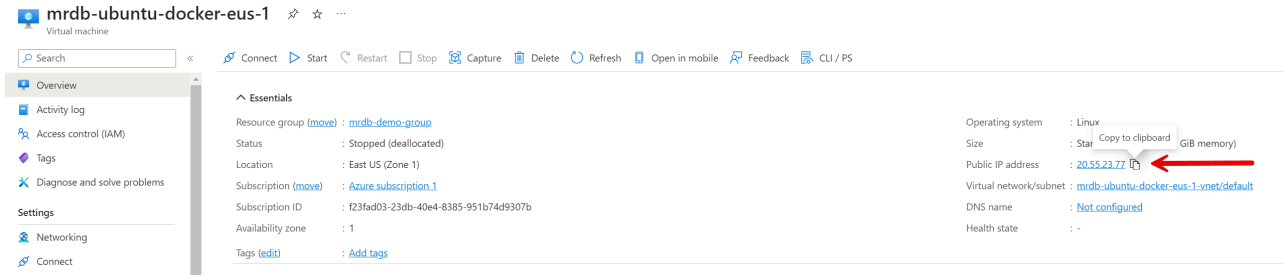
Priority *
350

Name
AllowCidrBlockCustom3306Inbound

Description
MariaDB

Mysql DB port 3306 is exposed to the Internet. We do not recommend exposing database ports to the Internet and suggest only exposing them to your front-end tier inside your virtual network.

12. Navigate back to the virtual machine's overview page. Then copy the public IP address to the clipboard.



Install Docker on the Azure VM

For more detailed instructions, refer to Installing and Using MariaDB via Docker

16. To connect to your remote server using SSH, open a terminal window and use the following command, replacing placeholders with your specific details:

```
ssh -i ~/.ssh/your-keyfile.pem azureuser@your.ip.address
```

    - Replace `your-keyfile.pem` with your private key file name.
    - Replace `your.ip.address` with your actual IP address from step 12.

    If you forget your admin details, go to Azure's left-hand navigation, select **Settings** > **Connect**. It will show the public IP address, admin username, and port.

17. Are you sure you want to continue connecting (yes/no/[fingerprint])?
    Say yes

18. To switch to the root user, use the following command:

```
sudo su
```

19.

```
Microsoft Azure offers two machine types with pre-installed Docker. If you need to reinstall Docker or
choose a machine type without Docker pre-installed, you can install it manually. Connect to your machine
via SSH and use the following command:
```

```
```bash
curl -fsSL get.docker.com | sudo sh
```
```

Pull the MariaDB Docker image and create the container

20. To download the latest MariaDB Docker image, run the following command:

```
docker pull mariadb:lts
```

21. To start the MariaDB Docker process, open your terminal or command line and enter the following command:

```
docker run --detach --name mariadb-docker -v C:\Users\YourUID\Documents\YourDirName:/var/lib/mysql:Z -
p 3306:3306 -e MARIADB_ROOT_PASSWORD=yoursecurepassword mariadb:lts
```

- Use the `-v` flag to mount a directory for persistent storage under `/var/lib/mysql`.
- Windows paths like `C:\Users\YourUID\Documents\YourDirName` should be specified as shown above.
- Ensure paths are absolute, not relative.
- Replace `yoursecurepassword` with a strong password, especially for environments beyond development.

22. To access the MariaDB container's shell, use the following command:

```
```bash
```

```
docker exec -it mariadb-docker bash
```
```

23. To login to MRDB inside the container, use the root password from step 20:

```
```bash
mariadb -p'yoursecurepassword'
```
```

24. Setup admin account with permission for remote connection, configure access control

```
MariaDB [(none)]> CREATE USER 'admin'@'%' IDENTIFIED BY 'admin';
MariaDB [(none)]> GRANT ALL ON *.* to 'admin'@'%' WITH GRANT OPTION;
MariaDB [(none)]> SHOW GRANTS FOR admin;
```

Obviously replace these passwords with something that is a bit more secure than you see in this example for anything other than development purposes.

25. Setup service account for your app with permission for remote connection, configure access control

```
MariaDB [(none)]> CREATE USER 'yourappname'@'%' IDENTIFIED BY 'yoursecurepassword';
MariaDB [(none)]> GRANT INSERT, UPDATE, DELETE ON _._ to 'yourappname'@'%';
MariaDB [(none)]> SHOW GRANTS FOR yourappname;
```

Obviously replace these passwords with something that is a bit more secure than you see in this example for anything other than development purposes.

26. Load up your database from your preexisting SQL script that contains CREATE DATABASE; USE DATABASE; and CREATE TABLE statements.

Open a new local terminal window and navigate to the directory containing your `init.sql` script. Then, run the command: `mariadb --host=ww.xx.yyy.zzz --port=3306 --user=admin --password=admin -e "SOURCE init.sql"`, replacing `ww.xx.yyy.zzz` with your server's IP address.

This page is licensed: CC BY-SA / Gnu FDL

Docker Official Image Frequently Asked Questions

Frequently asked questions about the Docker Official Image

How to Reset Passwords

If you have an existing data directory and wish to reset the root and user passwords, and to create a database which the user can fully modify, perform the following steps.

First create a `passwordreset.sql` file:

```
CREATE USER IF NOT EXISTS root@localhost IDENTIFIED BY 'thisismyrootpassword';
SET PASSWORD FOR root@localhost = PASSWORD('thisismyrootpassword');
GRANT ALL ON *.* TO root@localhost WITH GRANT OPTION;
GRANT PROXY ON ''@'%' ON root@localhost WITH GRANT OPTION;
CREATE USER IF NOT EXISTS root@'%' IDENTIFIED BY 'thisismyrootpassword';
SET PASSWORD FOR root@'%' = PASSWORD('thisismyrootpassword');
GRANT ALL ON *.* TO root@'%' WITH GRANT OPTION;
GRANT PROXY ON ''@'%' ON root@'%' WITH GRANT OPTION;
CREATE USER IF NOT EXISTS myuser@'%' IDENTIFIED BY 'thisismyuserpassword';
SET PASSWORD FOR myuser@'%' = PASSWORD('thisismyuserpassword');
CREATE DATABASE IF NOT EXISTS databasename;
GRANT ALL ON databasename.* TO myuser@'%';
```

Adjust `myuser`, `databasename` and passwords as needed.

Then:

```
$ docker run --rm -v /my/own/datadir:/var/lib/mysql -v /my/own/passwordreset.sql:/passwordreset.sql:z
%%IMAGE%%:latest --init-file=/passwordreset.sql
```

On restarting the MariaDB container in this `/my/own/datadir`, the `root` and `myuser` passwords will be reset.

Temp Server Start Timeout

Question, are you getting errors like the following where a temporary server start fails to succeed in 30 seconds?

Example of log:

```
2023-01-28 12:53:42+00:00 [Note] [Entrypoint]: Starting temporary server
2023-01-28 12:53:42+00:00 [Note] [Entrypoint]: Waiting for server startup
2023-01-28 12:53:42 0 [Note] mariadbd (server 10.10.2-MariaDB-1:10.10.2+maria~ubu2204) starting as
process 72 ...
....
2023-01-28 12:53:42 0 [Note] InnoDB: Setting file './ibtmp1' size to 12.000MiB. Physically writing the
file full; Please wait ...
2023-01-28 12:54:13 0 [Note] mariadbd: ready for connections.
Version: '10.10.2-MariaDB-1:10.10.2+maria~ubu2204'  socket: '/run/mysqld/mysqld.sock'  port: 0
mariadb.org binary distribution
2023-01-28 12:54:13+00:00 [ERROR] [Entrypoint]: Unable to start server.
```

The timeout on a temporary server start is a quite generous 30 seconds.

The lack of a message like the following indicates it failed to complete writing a temporary file of 12MiB in 30 seconds.

```
2023-01-28 12:53:46 0 [Note] InnoDB: File './ibtmp1' size is now 12.000MiB.
```

If the datadir where this is stored is remote storage maybe it's a bit slow. It's ideal to have an InnoDB temporary path local so this can be configured using the command or configuration setting:

```
innodb_temp_data_file_path=/dev/shm/ibtmp1:12M:autoextend
```

Note: depending on container runtime this space may be limited.

Creating a replication pair

`MARIADB_REPLICATION_USER` / `MARIADB_REPLICATION_PASSWORD` specify the authentication for the connection. The `MARIADB_MASTER_HOST` is the indicator that it is a replica and specifies the container aka hostname, of the master.

A `docker-compose.yml` example:

```
version: "3"
services:
  master:
    image: mariadb:latest
    command: --log-bin --log-basename=mariadb
    environment:
      - MARIADB_ROOT_PASSWORD=password
      - MARIADB_USER=testuser
      - MARIADB_PASSWORD=password
      - MARIADB_DATABASE=testdb
      - MARIADB_REPLICATION_USER=repl
      - MARIADB_REPLICATION_PASSWORD=replicationpass
    healthcheck:
      test: ["CMD", "healthcheck.sh", "--connect", "--innodb_initialized"]
      interval: 10s
      timeout: 5s
      retries: 3
  replica:
    image: mariadb:latest
    command: --server-id=2 --log-basename=mariadb
    environment:
      - MARIADB_ROOT_PASSWORD=password
      - MARIADB_MASTER_HOST=master
      - MARIADB_REPLICATION_USER=repl
      - MARIADB_REPLICATION_PASSWORD=replicationpass
      - MARIADB_HEALTHCHECK_GRANTS=REPLICA MONITOR
    healthcheck:
      test: ["CMD", "healthcheck.sh", "--connect", "--replication_io", "--replication_sql", "--
replication_seconds_behind_master=1", "--replication"]
      interval: 10s
      timeout: 5s
      retries: 3
    depends_on:
      master:
        condition: service_healthy
```

Event Scheduler: An error occurred when initializing system tables. Disabling the Event Scheduler.

This will show up in the container log as:

```
2024-01-29 17:38:13 0 [ERROR] Incorrect definition of table mysql.event: expected column 'definer' at
position 3 to have type varchar(, found type char(141).
2024-01-29 17:38:13 0 [ERROR] mariadbd: Event Scheduler: An error occurred when initializing system
tables. Disabling the Event Scheduler.
```

The cause is the underlying table has change structure from the last MariaDB version. The easiest solution to this is to start the container with the environment variable MARIADB_AUTO_UPGRADE=1 and system tables will be updated. This is safe to keep on as it detects the version installed. The next start should not show this error.

InnoDB: Upgrade after a crash is not supported. The redo log was created with MariaDB X.Y.Z

This will show up in the error log as:

```
2022-05-23 12:29:20 0 [ERROR] InnoDB: Upgrade after a crash is not supported. The redo log was created
with MariaDB 10.5.4.
2022-05-23 12:29:20 0 [ERROR] InnoDB: Plugin initialization aborted with error Generic error
```

This is attempting to start on a higher MariaDB version when the shutdown of the previous version crashed.

By crashed, it means the MariaDB was force killed or had a hard power failure. MariaDB, being a durable database, can recover from these, if started with the same version. The redo log however is a less stable format, so the recovery has to be on the same Major.Minor version, in this case 10.5. This error message is saying that you when from force killed MariaDB to a later version.

So whenever you encounter this message. Start with the again with the tag set to the version in the error message, like 10.5.4, or as the redo long format is consistent in the Major.Minor version 10.5 is sufficient. After this has been started correctly, cleanly shut the service down and it will be recovered.

The logs on shutdown should have a message like:

```
2023-11-06 10:49:23 0 [Note] InnoDB: Shutdown completed; log sequence number 84360; transaction id 49
2023-11-06 10:49:23 0 [Note] mariadbd: Shutdown complete
```

After you see this, you can update your MariaDB tag to a later version.

Every MariaDB start gives permission denied messages

```
2024-02-06 03:03:18+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server
1:10.11.6+maria~ubu2204 started.
/usr/local/bin/docker-entrypoint.sh: line 600: /var/lib/mysql//mysql_upgrade_info: Permission denied
2024-02-06 03:03:18+00:00 [Note] [Entrypoint]: MariaDB upgrade (mariadb-upgrade) required, but skipped
due to $MARIADB_AUTO_UPGRADE setting
2024-02-06  3:03:18 0 [Warning] Can't create test file '/var/lib/mysql/80a2bb81d698.lower-test' (Errcode:
13 "Permission denied")
2024-02-06  3:03:18 0 [Note] Starting MariaDB 10.11.6-MariaDB-1:10.11.6+maria~ubu2204 source revision
fecd78b83785d5ae96f2c6ff340375be803cd299 as process 1
2024-02-06  3:03:18 0 [ERROR] mariadbd: Can't create/write to file './ddl_recovery.log' (Errcode: 13
"Permission denied")
2024-02-06  3:03:18 0 [ERROR] DDL_LOG: Failed to create ddl log file: ./ddl_recovery.log
2024-02-06  3:03:18 0 [ERROR] Aborting
```

Or:

```
2024-08-16  4:54:05 0 [ERROR] InnoDB: Operating system error number 13 in a file operation.
2024-08-16  4:54:05 0 [ERROR] InnoDB: The error means mariadbd does not have the access rights to the
directory.
```

In this case, the container is running as a user that, inside the container, does not have write permissions on the datadir `/varlib/mysql`.

Bad magic header in tc log

From the transaction coordinator log this is a corrupted file. This will have a log message like the following:

```
2024-05-21  8:55:58 0 [Note] Recovering after a crash using tc.log
2024-05-21  8:55:58 0 [ERROR] Bad magic header in tc log
2024-05-21  8:55:58 0 [ERROR] Crash recovery failed. Either correct the problem (if it's, for example,
out of memory error) and restart, or delete tc log and start server with --tc-heuristic-recover=
{commit|rollback}
2024-05-21  8:55:58 0 [ERROR] Can't init tc log
2024-05-21  8:55:58 0 [ERROR] Aborting
```

The cause of this is headed by the first not, its a crash recovery. Like the Every MariaDB start is a crash recovery answer below, this is an indication that MariaDB wasn't given enough time by the container runtime to shutdown cleanly. While MariaDB was shutdown, the new version that was started is a newer MariaDB and doesn't recognise the updated magic information in the header.

MariaDB should always perform crash recovery with the same version that actually crashed, the same major/minor number at least.

As such the solution is to restart the container with the previous MariaDB version that was running, and configure the container runtime to allow a longer stop time. See the Every MariaDB start is a crash recovery answer below to see if the timeout is sufficiently extended.

Every MariaDB start is a crash recovery

Do you get on every start:

```
db-1  | 2023-02-25 19:10:02 0 [Note] Starting MariaDB 10.11.2-MariaDB-1:10.11.2+maria~ubu2204-log source
revision cafba8761af55ae16cc69c9b53a341340a845b36 as process 1
db-1  | 2023-02-25 19:10:02 0 [Note] mariadbd: Aria engine: starting recovery
db-1  | tables to flush: 3 2 1 0
db-1  |  (0.0 seconds);
db-1  | 2023-02-25 19:10:02 0 [Note] mariadbd: Aria engine: recovery done
...
db-1  | 2023-02-26 13:03:29 0 [Note] InnoDB: Initializing buffer pool, total size = 32.000GiB, chunk size
= 512.000MiB
db-1  | 2023-02-26 13:03:29 0 [Note] InnoDB: Completed initialization of buffer pool
db-1  | 2023-02-26 13:03:29 0 [Note] InnoDB: File system buffers for log disabled (block size=512 bytes)
db-1  | 2023-02-26 13:03:29 0 [Note] InnoDB: Starting crash recovery from checkpoint LSN=193796878816
```

Container runtimes are assume to start and stop very quickly. Check the shutdown logs. They may be a log like:

```
db-1  | 2023-02-26 13:03:17 0 [Note] InnoDB: Starting shutdown...
db-1  | 2023-02-26 13:03:17 0 [Note] InnoDB: Dumping buffer pool(s) to /var/lib/mysql/ib_buffer_pool
db-1  | 2023-02-26 13:03:17 0 [Note] InnoDB: Restricted to 519200 pages due to
innodb_buf_pool_dump_pct=25
db-1  | 2023-02-26 13:03:17 0 [Note] InnoDB: Buffer pool(s) dump completed at 230226 13:03:17
db-1 exited with code 0
```

Note that the logs didn't include the following messages:

```
db-1  | 2023-02-26 13:03:43 0 [Note] InnoDB: Shutdown completed; log sequence number 46590; transaction
id 15
db-1  | 2023-02-26 13:03:43 0 [Note] mariadbd: Shutdown complete
```

As these messages aren't here, the container was killed off before it could just down cleanly. When this happens, the startup will be a crash recovery and you won't be able to upgrade your MariaDB instance (previous FAQ) to the next Major.Minor version.

Solution is to extend the timeout in the container runtime to allow MariaDB to complete its shutdown.

How do I create a `mariaDB-backup` of the data?

```
docker volume create backup
docker run --name mdb -v backup:/backup -v datavolume:/var/lib/mysql mariadb
docker exec mdb mariadb-backup --backup --target-dir=/backup/d --user root --password soverysecret
docker exec mdb mariadb-backup --prepare --target-dir=/backup/d
docker exec mdb sh -c '[ ! -f /backup/d/.my-healthcheck.cnf ] && cp /var/lib/mysql/.my-healthcheck.cnf
/backup/d'
docker exec --workdir /backup/d mdb tar -Jcf ../backup.tar.xz .
docker exec mdb  rm -rf /backup/d
```

How do I restore from a MariaDB-backup

With the backup prepared like previously:

```
docker run -v backup:/docker-entrypoint-initdb.d -v newdatavolume:/var/lib/mysql mariadb
```

How to start MariaDB with Apptainer

Because Apptainer has all the filesystems readonly except or the volume, the /run/mysqld directory is used as a pidfile and socket directory. An easy way is to mark this as a scratch directory.

```
mkdir mydatadir
apptainer run --no-home --bind $PWD/mydatadir:/var/lib/mysql --env MARIADB_RANDOM_ROOT_PASSWORD=1 --net -
-network-args "portmap=3308:3306/tcp" --fakeroot --scratch=/run/mysqld  docker://mariadb:10.5
```

Alternately:

```
apptainer run --no-home --bind $PWD/mydatadir:/var/lib/mysql --env MARIADB_RANDOM_ROOT_PASSWORD=1 --net -
-network-args "portmap=3308:3306/tcp" --fakeroot   docker://mariadb:10.5 --
socket=/var/lib/mysql/mariadb.sock --pid-file=/var/lib/mysql/mariadb.pid
```

Why does the MariaDB container start as root?

The MariaDB entrypoint briefly starts as `root`, and if an explicit volume is there, the owner of this volume will be `root`. To allow MariaDB to use the `CHOWN` capability to change the volume owner to a user that can write to this volume, it needs to be briefly `root`. After this one action is taken, the entrypoint uses `gosu` to drop to a non-root user and continues execution. There is no accessible exploit vector to remotely affect the container startup when it is briefly running as the `root` user.

Can I run the MariaDB container as an arbitrary user?

In a Docker Compose file or when running a command, you can specify the user with `user: 2022` or `--user 2022` to execute the entrypoint as user ID 2022. Ensure the datadir volume has appropriate permissions for MariaDB access, aligning with local ownership by user ID 2022. Note that inside the container, user names are not defined as they are outside, so using numeric IDs offers better portability.

This page is licensed: CC BY-SA / Gnu FDL

Installing and Using MariaDB via Docker

Sometimes we want to install a specific version of MariaDB, MariaDB ColumnStore, or MaxScale on a certain system, but no packages are available. Or maybe, we simply want to isolate MariaDB from the rest of the system, to be sure that we won't cause any damage.

A virtual machine would certainly serve the scope. However, this means installing a system on the top of another system. It requires a lot of resources.

In many cases, the best solution is using containers. Docker is a framework that runs containers. A container is meant to run a specific daemon, and the software that is needed for that daemon to properly work. Docker does not virtualize a whole system; a container only includes the packages that are not included in the underlying system.

Docker requires a very small amount of resources. It can run on a virtualized system. It is used both in development and in production environments. Docker is an open source project, released under the Apache License, version 2.

Note that, while your package repositories could have a package called `docker`, it is probably not the Docker we are talking about. The Docker package could be called `docker.io` or `docker-engine`.

For information about installing Docker, see Get Docker in Docker documentation.

Installing Docker on Your System with the Universal Installation Script

The script below will install the Docker repositories, required kernel modules and packages on the most common Linux distributions:

```
curl -sSL https://get.docker.com/ | sh
```

Starting dockerd

On some systems you may have to start the `dockerd daemon` yourself:

```
sudo systemctl start docker
sudo gpasswd -a "${USER}" docker
```

If `dockerd` is not running, you will encounter the following error when using most `docker` commands:

```
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
```

Using MariaDB Images

The easiest way to use MariaDB on Docker is choosing a MariaDB image and creating a container.

Downloading an Image

You can download a MariaDB image for Docker from the Official Docker MariaDB, or choose another image that better suits your needs. You can search Docker Hub (the official set of repositories) for an image with this command:

```
docker search mariadb
```

Once you have found an image that you want to use, you can download it via Docker. Some layers including necessary dependencies will be downloaded too. Note that, once a layer is downloaded for a certain image, Docker will not need to download it again for another image.

For example, if you want to install the default MariaDB image, you can type:

```
docker pull mariadb:10.6
```

This will install the 10.6 version.

You will see a list of necessary layers. For each layer, Docker will say if it is already present, or its download progress.

To get a list of installed images:

```
docker images
```

Creating a Container

An image is not a running process; it is just the software needed to be launched. To run it, we must create a container first. The command needed to create a container can usually be found in the image documentation. For example, to create a container for the official MariaDB image:

```
docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=mypass -p 3306:3306 -d
docker.io/library/mariadb:10.3
```

`mariadbtest` is the name we want to assign the container. If we don't specify a name, an id will be automatically generated.

10.2 and 10.5 are also valid target versions:

```
docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=mypass -p 3306:3306 -d
docker.io/library/mariadb:10.2
```

```
docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=mypass -p 3306:3306  -d
docker.io/library/mariadb:10.5
```

Optionally, after the image name, we can specify some options for mariadbd. For example:

```
docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=mypass -p 3306:3306 -d mariadb:10.3 --log-bin --
binlog-format=MIXED
```

Docker will respond with the container's id. But, just to be sure that the container has been created and is running, we can get a list of running containers in this way:

```
docker ps
```

We should get an output similar to this one:

```
CONTAINER ID        IMAGE                     COMMAND                CREATED             STATUS
PORTS               NAMES
819b786a8b48        mariadb                   "/docker-entrypoint.   4 minutes ago       Up 4 minutes
3306/tcp            mariadbtest
```

Running and Stopping the Container

Docker allows us to restart a container with a single command:

```
docker restart mariadbtest
```

The container can also be stopped like this:

```
docker stop mariadbtest
```

The container will not be destroyed by this command. The data will still live inside the container, even if MariaDB is not running. To restart the container and see our data, we can issue:

```
docker start mariadbtest
```

With `docker stop`, the container will be gracefully terminated: a `SIGTERM` signal will be sent to the `mariadbd` process, and Docker will wait for the process to shutdown before returning the control to the shell. However, it is also possible to set a timeout, after which the process will be immediately killed with a `SIGKILL`. Or it is possible to immediately kill the process, with no timeout.

```
docker stop --time=30 mariadbtest
docker kill mariadbtest
```

In case we want to destroy a container, perhaps because the image does not suit our needs, we can stop it and then run:

```
docker rm mariadbtest
```

Note that the command above does not destroy the data volume that Docker has created for /var/lib/mysql. If you want to destroy the volume as well, use:

```
docker rm -v mariadbtest
```

Automatic Restart

When we start a container, we can use the `--restart` option to set an automatic restart policy. This is useful in production.

Allowed values are:

- `no`: No automatic restart.
- `on-failure`: The container restarts if it exits with a non-zero exit code.
- `unless-stopped`: Always restart the container, unless it was explicitly stopped as shown above.
- `always`: Similar to `unless-stopped`, but when Docker itself restarts, even containers that were explicitly stopped will restart.

It is possible to change the restart policy of existing, possibly running containers:

```
docker update --restart always mariadb
# or, to change the restart policy of all containers:
docker update --restart always $(docker ps -q)
```

A use case for changing the restart policy of existing containers is performing maintenance in production. For example, before upgrading the Docker version, we may want to change all containers restart policy to `always`, so they will restart as soon as the new version is up and running. However, if some containers are stopped and not needed at the moment, we can change their restart policy to `unless-stopped`.

Pausing Containers

A container can also be frozen with the `pause` command. Docker will freeze the process using croups. MariaDB will not know that it is being frozen and, when we `unpause` it, MariaDB will resume its work as expected.

Both `pause` and `unpause` accept one or more container names. So, if we are running a cluster, we can freeze and resume all nodes simultaneously:

```
docker pause node1 node2 node3
docker unpause node1 node2 node3
```

Pausing a container is very useful when we need to temporarily free our system's resources. If the container is not crucial at this moment (for example, it is performing some batch work), we can free it to allow other programs to run faster.

Troubleshooting a Container

If the container doesn't start, or is not working properly, we can investigate with the following command:

```
docker logs mariadbtest
```

This command shows what the daemon sent to the stdout since the last attempt of starting - the text that we typically see when we invoke `mariadbd` from the command line.

On some systems, commands such as `docker stop mariadbtest` and `docker restart mariadbtest` may fail with a permissions error. This can be caused by AppArmor, and even `sudo` won't allow you to execute the command. In this case, you will need to find out which profile is causing the problem and correct it, or disable it. **Disabling AppArmor altogether is not recommended, especially in production.**

To check which operations were prevented by AppArmor, see AppArmor Failures in AppArmor documentation.

To disable a profile, create a symlink with the profile name (in this example, `mariadbd`) to `etc/apparmor.d/disable`, and then reload profiles:

```
ln -s /etc/apparmor.d/usr.sbin.mariadbd /etc/apparmor.d/disable/
sudo apparmor_parser -R /etc/apparmor.d/usr.sbin.mariadbd
```

For more information, see Policy Layout in AppArmor documentation.

After disabling the profile, you may need to run:

```
sudo service docker restart
docker system prune --all --volumes
```

Restarting the system will then allow Docker to operate normally.

Accessing the Container

To access the container via Bash, we can run this command:

```
docker exec -it mariadbtest bash
```

Now we can use normal Linux commands like *cd*, *ls*, etc. We will have root privileges. We can even install our favorite file editor, for example:

```
apt-get update
apt-get install vim
```

In some images, no repository is configured by default, so we may need to add them.

Note that if we run mariadb-admin shutdown or the SHUTDOWN command to stop the container, the container will be deactivated, and we will automatically exit to our system.

Connecting to MariaDB from Outside the Container

If we try to connect to the MariaDB server on `localhost`, the client will bypass networking and attempt to connect to the server using a socket file in the local filesystem. However, this doesn't work when MariaDB is running inside a container because the server's filesystem is isolated from the host. The client can't access the socket file which is inside the container, so it fails to connect.

Therefore connections to the MariaDB server must be made using TCP, even when the client is running on the same machine as the server container.

Find the IP address that has been assigned to the container:

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mariadbtest
```

You can now connect to the MariaDB server using a TCP connection to that IP address.

Forcing a TCP Connection

After enabling network connections in MariaDB as described above, we will be able to connect to the server from outside the container.

On the host, run the client and set the server address ("-h") to the container's IP address that you found in the previous step:

```
mysql -h 172.17.0.2 -u root -p
```

This simple form of the connection should work in most situations. Depending on your configuration, it may also be necessary to specify the port for the server or to force TCP mode:

```
mysql -h 172.17.0.2 -P 3306 --protocol=TCP -u root -p
```

Port Configuration for Clustered Containers and Replication

Multiple MariaDB servers running in separate Docker containers can connect to each other using TCP. This is useful for forming a Galera cluster or for replication.

When running a cluster or a replication setup via Docker, we will want the containers to use different ports. The fastest way to achieve this is mapping the containers ports to different port on our system. We can do this when creating the containers (`docker run` command), by using the `-p` option, several times if necessary. For example, for Galera nodes we will use a mapping similar to this one:

```
-p 4306:3306 -p 5567:5567 -p 5444:5444 -p 5568:5568
```

Installing MariaDB on Another Image

It is possible to download a Linux distribution image, and to install MariaDB on it. This is not much harder than installing MariaDB on a regular operating system (which is easy), but it is still the hardest option. Normally we will try existing images first. However, it is possible that no image is available for the exact version we want, or we want a custom installation, or perhaps we want to use a distribution for which no images are available. In these cases, we will install MariaDB in an operating system image.

Daemonizing the Operating System

First, we need the system image to run as a daemon. If we skip this step, MariaDB and all databases will be lost when the container stops.

To demonize an image, we need to give it a command that never ends. In the following example, we will create a Debian Jessie daemon that constantly pings the 8.8.8.8 special address:

```
docker run --name debian -p 3306:3306 -d debian /bin/sh -c "while true; do ping 8.8.8.8; done"
```

Installing MariaDB

At this point, we can enter the shell and issue commands. First we will need to update the repositories, or no packages will be available. We can also update the packages, in case some of them are newer than the image. Then, we will need to install a text editor; we will need it to edit configuration files. For example:

```
# start an interactive Bash session in the container
docker exec -ti debian bash
apt-get -y update
apt-get -y upgrade
apt-get -y install vim
```

Now we are ready to install MariaDB in the way we prefer.

See Also

- Official MariaDB Docker Images Webinar.
- Docker official site.
- Docker Hub.
- Docker documentation.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Container Cheat Sheet

Get the images

Images can be found on MariaDB Docker Hub. To get the list of images, run this command:

```
$ docker images -a
```

Create the network

```
$ docker network create mynetwork
```

It is good practice to create the container network and attach the container to the network.

Start The Container With Server Options

To start the container in the background with the MariaDB server image, run:

```
$ docker run --rm --detach \
  --env MARIADB_ROOT_PASSWORD=sosecret \
  --network mynetwork \
  --name mariadb-server \
  mariadb:latest
```

Additionally, environment variables are also provided.

Get the list of running containers

```
$ docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED          STATUS          PORTS
NAMES
ad374ec8a272   mariadb:latest   "docker-entrypoint.s…"   3 seconds ago    Up 1 second     3306/tcp
mariadb-server
```

Note: Specify the flag `-a` in case you want to see all containers

Start the client from the container

To start the *mariadb* client inside the created container and run specific commands, run the following:

```
$ docker exec -it mariadb-server mariadb -psosecret -e "SHOW PLUGINS"
```

Inspect logs of a container

```
$ docker logs mariadb-server
```

In the logs you can find status information about the server, plugins, generated passwords, errors and so on.

Restart the container

```
$ docker restart mariadb-server
```

Run commands within the container

```
$ docker exec -it mariadb-server bash
```

Use a volume to specify configuration options

```
$ docker run --detach --env MARIADB_USER=anel \
  --env MARIADB_PASSWORD=anel \
  --env MARIADB_DATABASE=my_db \
  --env MARIADB_RANDOM_ROOT_PASSWORD=1 \
  --volume $PWD/my_container_config:/etc/mysql/conf.d:z \
  --network mynetwork \
  --name mariadb-server1 \
   mariadb:latest
```

One can specify custom configuration files through the `/etc/mysql/conf.d` volume during container startup.

Use a volume to specify grants during container start

```
$ docker run --detach --env MARIADB_USER=anel\
  --env MARIADB_PASSWORD=anel \
  --env MARIADB_DATABASE=my_db \
  --env MARIADB_RANDOM_ROOT_PASSWORD=1 \
  --volume $PWD/my_init_db:/docker-entrypoint-initdb.d \
  --network mynetwork \
  --name mariadb-server1 \
   mariadb:latest
```

User created with the environment variables has full grants only to the `MARIADB_DATABASE`. In order to override those grants, one can specify grants to a user, or execute any SQL statements from host file to **docker-entrypoint-initdb.d**. In the `local_init_dir` directory we can find the file, created like this:

```
$ echo "GRANT ALL PRIVILEGES ON *.* TO anel;" > my_init_db/my_grants.sql
```

See Also

- Installing and using MariaDB via Docker

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Enterprise Server Docker Official Image Environment Variables

When you start the image, you can adjust the initialization of the MariaDB Server instance by passing one or more environment variables on the docker run command line. Do note that all of the variables below, except `MARIADB_AUTO_UPGRADE`, will have no effect if you start the container with a data directory that already contains a database: any pre-existing database will always be left untouched on container startup.

All tags from `10.6` and above, `MARIADB_*` variables will be used in preference to `MYSQL_*` variables.

One of the following is required: `MARIADB_ROOT_PASSWORD_HASH`, `MARIADB_ROOT_PASSWORD`, `MARIADB_ALLOW_EMPTY_ROOT_PASSWORD`, or `MARIADB_RANDOM_ROOT_PASSWORD` (including `*_FILE` equivalents).

Other environment variables are optional.

Environment Variables

`MARIADB_ROOT_PASSWORD_HASH / MARIADB_ROOT_PASSWORD / MYSQL_ROOT_PASSWORD`

This specifies the password that will be set for the MariaDB root superuser account.

`MARIADB_ALLOW_EMPTY_ROOT_PASSWORD / MYSQL_ALLOW_EMPTY_PASSWORD`

Set to a non-empty value, like `1`, to allow the container to be started with a blank password for the root user.

> **Warning:**
>
> Setting this variable to yes is not recommended unless you really know what you are doing, since this will leave your MariaDB instance completely unprotected, allowing anyone to gain complete superuser access.

`MARIADB_RANDOM_ROOT_PASSWORD / MYSQL_RANDOM_ROOT_PASSWORD`

Define a non-empty value, such as “yes,” to auto-generate a random initial password for the root user. The password will be output to stdout, prefixed with “GENERATED ROOT PASSWORD: …”.

`MARIADB_ROOT_HOST / MYSQL_ROOT_HOST`

`%` is the default hostname part of the root user in MariaDB. This can be changed to any valid hostname. Setting it to `localhost` restricts root access to only the local machine via the Unix socket.

`MARIADB_USER_HOST`

`%` is the default hostname part of the user created through `MYSQL_USER / MARIADB_USER` in MariaDB. This can be changed to any valid hostname or IP. Setting it to `localhost` restricts user access to only the local machine via the Unix socket.

`MARIADB_DATABASE / MYSQL_DATABASE`

This variable allows you to specify the name of a database to be created on image startup.

`MARIADB_USER / MYSQL_USER, MARIADB_PASSWORD_HASH / MARIADB_PASSWORD / MYSQL_PASSWORD`

To create a new user with full access permissions in MariaDB, both `user` and `password` variables are required, along with a designated `database`. This new user will be granted comprehensive privileges (`GRANT ALL`) to the specified `MARIADB_DATABASE`. Note that this method should not be utilized for creating the root superuser, as this user is automatically created with the password provided by the `MARIADB_ROOT_PASSWORD` or `MYSQL_ROOT_PASSWORD` variable.

`MARIADB_MYSQL_LOCALHOST_USER / MARIADB_MYSQL_LOCALHOST_GRANTS`

Set `MARIADB_MYSQL_LOCALHOST_USER` to a non-empty value to create the `mysql@localhost` database user. This user is useful for health checks and backup scripts. The `mysql@localhost` user gets `USAGE` privileges by default. If more access is needed, additional global privileges can be provided as a comma-separated list. Be cautious when sharing a volume containing MariaDB's unix socket (`/var/run/mysqld` by default) as privileges beyond `USAGE` may pose security risks. This user can also be used with `mariadb-backup`. Refer to `healthcheck.sh` for required privileges for each health check test.

`MARIADB_HEALTHCHECK_GRANTS`

Set `MARIADB_HEALTHCHECK_GRANTS` to the grants required to be given to the `healthcheck@localhost`, `healthcheck@127.0.0.1`, `healthcheck@::1`, users. When not specified the default grant is USAGE.

The main value used here will be `[REPLICA MONITOR](#pg-rence-sql-statements-account-management-sql-statements-grant-7cefc036)` for the `[healthcheck --replication](#pg-t-and-administration-docker-and-mariadb-using-healthcheck-sh-db0da349)` test.

`MARIADB_INITDB_SKIP_TZINFO / MYSQL_INITDB_SKIP_TZINFO`

By default, the entrypoint script automatically loads the timezone data needed for the `CONVERT_TZ()` function. If it is not needed, any non-empty value disables timezone loading.

`MARIADB_AUTO_UPGRADE / MARIADB_DISABLE_UPGRADE_BACKUP`

Set `MARIADB_AUTO_UPGRADE` to a non-empty value to have the entrypoint check whether mariadb-upgrade needs to run, and if so, run the upgrade before starting the MariaDB server.

Before the upgrade, a backup of the system database is created in the top of the datadir with the name `system_mysql_backup_*.sql.zst`. This backup process can be disabled by setting `MARIADB_DISABLE_UPGRADE_BACKUP` to a non-empty value.

If `MARIADB_AUTO_UPGRADE` is set, and the `.my-healthcheck.cnf` file is missing, the `healthcheck` users are recreated if they don't exist, `MARIADB_HEALTHCHECK_GRANTS` grants are given, the passwords of the `healthcheck`

users are reset to a random value and the `.my-healthcheck.cnf` file is recreated with the new password populated.

`MARIADB_MASTER_HOST`

When specified, the container will connect to this host and replicate from it.

`MARIADB_REPLICATION_USER / MARIADB_REPLICATION_PASSWORD_HASH / MARIADB_REPLICATION_PASSWORD`

When `MARIADB_MASTER_HOST` is defined, `MARIADB_REPLICATION_USER` and `MARIADB_REPLICATION_PASSWORD` will be used to connect to the master. When not specified, the `MARIADB_REPLICATION_USER` will be created with the `REPLICATION REPLICA` grants needed for a client to initiate replication.

Timezone Configuration

By default, the container operates in the Coordinated Universal Time (UTC) timezone. To configure a specific timezone for the container's operating system and internal processes, utilize the `TZ` environment variable.

Supported Values

The `TZ` variable accepts standard tz database (IANA) timezone identifiers. Examples include `America/New_York`, `Europe/London`, or `Asia/Tokyo`.

Usage Examples

Using `docker run`

You can specify the timezone when starting the container via the command line interface using the `-e` flag:

```
docker run -d \
  --name database-container \
  -e TZ="America/New_York" \
  -e MARIADB_ROOT_PASSWORD="your_strong_password" \
  docker.mariadb.com/enterprise-server:11.8
```

Using `docker-compose.yml`

For deployments managed by Docker Compose, declare the `TZ` variable within the `environment` mapping of your service definition:

```
services:
  db:
    image: docker.mariadb.com/enterprise-server:11.8
    environment:
      - TZ=America/New_York
      - MARIADB_ROOT_PASSWORD=your_strong_password
```

Using `_FILE` Environment Variables for Secrets

When running Docker containers, it is a security best practice to avoid passing sensitive information, like database passwords, directly as plain-text environment variables. The MariaDB container images allow you to securely read secrets from mounted files by appending `_FILE` to the standard environment variable name.

List of supported `_FILE` Environment Variables

Name	Description
`MYSQL_ROOT_HOST_FILE`	Path to a file containing the host from which the root user is allowed to connect.
`MARIADB_ROOT_HOST_FILE`	Path to a file containing the host from which the root user is allowed to connect.

Name	Description
`MYSQL_DATABASE_FILE`	Path to a file containing the name of a database to be created on initialization.
`MARIADB_DATABASE_FILE`	Path to a file containing the name of a database to be created on initialization.
`MYSQL_USER_FILE`	Path to a file containing the username for a new user to be created on initialization.
`MARIADB_USER_FILE`	Path to a file containing the username for a new user to be created on initialization.
`MYSQL_PASSWORD_FILE`	Path to a file containing the password for the user defined by `MYSQL_USER`.
`MARIADB_PASSWORD_FILE`	Path to a file containing the password for the user defined by `MARIADB_USER`.
`MYSQL_ROOT_PASSWORD_FILE`	Path to a file containing the password for the root database user.
`MARIADB_ROOT_PASSWORD_FILE`	Path to a file containing the password for the root database user.
`MARIADB_PASSWORD_HASH_FILE`	Path to a file containing the hashed password for the created standard user.
`MARIADB_ROOT_PASSWORD_HASH_FILE`	Path to a file containing the hashed password for the root database user.
`MARIADB_UNIX_SOCKET_AUTHENTICATION_FILE`	Path to a file containing a value to enable/disable Unix socket authentication.
`MARIADB_REPLICATION_USER_FILE`	Path to a file containing the username to be used for setting up replication.
`MARIADB_REPLICATION_PASSWORD_FILE`	Path to a file containing the password for the replication user.
`MARIADB_REPLICATION_PASSWORD_HASH_FILE`	Path to a file containing the hashed password for the replication user.
`MARIADB_MASTER_HOST_FILE`	Path to a file containing the hostname or IP address of the replication master.
`MARIADB_MASTER_PORT_FILE`	Path to a file containing the port number of the replication master.

Example

Here is how to set up your MariaDB root password using the `MARIADB_ROOT_PASSWORD_FILE` environment variable.

Step 1: Create the Secret File

First, create a local file on your host machine that contains your desired environment variable. In this example, we will create a file named `mariadb_root_password_secret` containing the password `MariaDB11!`.

```
echo -n "MariaDB11!" > mariadb_root_password_secret
```

> **Note:**
>
> Using the `-n` flag with `echo` is highly recommended, as it ensures no hidden newline characters are accidentally appended to your password!

Step 2: Start the MariaDB Container

Next, run the container. You need to mount the local secret file into the container and set the environment variable to point to that newly mounted path.

```
docker run -d \
  -v "$(pwd)/mariadb_root_password_secret:/run/secrets/root_password" \
  -e MARIADB_ROOT_PASSWORD_FILE=/run/secrets/root_password \
  docker.mariadb.com/enterprise-server:11.8
```

Understanding the flags: * **`-v "$(pwd)/..."`**: This volume mount takes your local file (`mariadb_root_password_secret`) and securely maps it inside the container to `/run/secrets/root_password`. * **`-e MARIADB_ROOT_PASSWORD_FILE=...`**: This instructs the MariaDB initialization script to look inside `/run/secrets/root_password` and use its contents as the root password.

Step 3: Verify the Connection

Once the container is up and running, you can test the connection to ensure the password was applied correctly.

Grab your container ID (with `docker ps`) and use `docker exec` to start an interactive database session using the password you defined in Step 1:

```
docker exec -it <container_id> mariadb -u root -p'MariaDB11!'
```

If the secret was read successfully, you will be authenticated and greeted by the MariaDB:

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 5
```

This page is licensed: CC BY-SA / Gnu FDL

Setting Up a LAMP Stack with Docker Compose

Docker Compose is a tool that allows one to declare which Docker containers should run, and which relationships should exist between them. It follows the **infrastructure as code** approach, just like most automation software and Docker itself.

For information about installing Docker Compose, see Install Docker Compose in Docker documentation.

The `docker-compose.yml` File

When using Docker Compose, the Docker infrastructure must be described in a YAML file called `docker-compose.yml`.

Let's see an example:

```
version: "3"

services:
  web:
    image: "apache:${PHP_VERSION}"
    restart: 'always'
    depends_on:
      - mariadb
    restart: 'always'
    ports:
      - '8080:80'
    links:
      - mariadb
  mariadb:
    image: "mariadb:${MARIADB_VERSION}"
    restart: 'always'
    volumes:
      - "/var/lib/mysql/data:${MARIADB_DATA_DIR}"
      - "/var/lib/mysql/logs:${MARIADB_LOG_DIR}"
      - /var/docker/mariadb/conf:/etc/mysql
    environment:
      MYSQL_ROOT_PASSWORD: "${MYSQL_ROOT_PASSWORD}"
      MYSQL_DATABASE: "${MYSQL_DATABASE}"
      MYSQL_USER: "${MYSQL_USER}"
      MYSQL_PASSWORD: "${MYSQL_PASSWORD}"
```

In the first line we declare that we are using version 3 of the Docker compose language.

Then we have the list of services, namely the `web` and the `mariadb` services.

Let's see the properties of the services:

Property	Description
`port`	Maps the 8080 container port to the 80 host system port; useful for development, allows browser connection to containerized web server.
`links`	Declares container connectivity, ensuring that the container can connect to `mariadb`. The hostname is the container name.
`depends_on`	Specifies that `mariadb` must start before `web`, ensuring the application can connect once MariaDB is ready.
`restart`	Ensures containers restart automatically if they crash (`restart: always`).
`volumes`	Creates accessible directories for containers on the host system, facilitating data persistence even if a container is destroyed.
`environment`	Sets environment variables within the container, crucial for defining MariaDB root credentials.

About Volumes

It is good practice to create volumes for:

- The data directory, so we don't lose data when a container is created or replaced, perhaps to upgrade MariaDB.
- The directory where we put all the logs, if it is not the datadir.
- The directory containing all configuration files (for development environments), so we can edit those files with the editor installed in the host system. Normally no editor is installed in containers. In production we don't need to do this, because we can copy files from a repository located in the host system to the containers.

Note that Docker Compose variables are just placeholders for values. Compose does not support assignment, conditionals or loops.

Using Variables

In the above example you can see several variables, like `${MARIADB_VERSION}`. Before executing the file, Docker Compose will replace this syntax with the `MARIADB_VERSION` variable.

Variables allow making Docker Compose files more reusable: in this case, we can use any MariaDB image version without modifying the Docker Compose file.

The most common way to pass variables is to write them into a file. This has the benefit of allowing us to version the variable file along with the Docker Compose file. It uses the same syntax you would use in BASH:

```
PHP_VERSION=8.0
MARIADB_VERSION=10.6
...
```

For bigger setups, it could make sense to use different environment files for different services. To do so, we need to specify the file to use in the Compose file:

```
services:
  web:
    env_file:
      - web-variables.env
...
```

Docker Compose Commands

Docker Compose is operated using `docker-compose`. Here we'll see the most common commands. For more commands and for more information about the commands mentioned here, see the documentation.

Docker Compose assumes that the Composer file is located in the current directory and it's called `docker-compose.yml`. To use a different file, the `-f <filename>` parameter must be specified.

Command	Description
`docker-compose pull`	Pulls the necessary images.
`docker-compose up --no-recreate`	Creates containers only if they do not already exist.
`docker-compose up`	Creates containers if they don't exist and recreates them if their image or configuration has changed.
`docker-compose up --force-recreate`	Recreates containers in all cases.
`docker-compose up --no-start`	Creates containers without starting them.
`docker-compose restart`	Restarts containers without recreating them.
`docker-compose kill <service>`	Kills a specific container by sending it a SIGKILL signal.
`docker-compose rm -f <service>`	Instantly removes a running container.
`docker-compose down`	Tears down all containers, networks, and volumes created by the Compose file.

Docker Compose Resources and References

- Overview of Docker Compose in the Docker documentation.
- Compose file in the Docker documentation.
- Docker Compose on GitHub.

Further information about the concepts explained in this page can be found in Docker documentation:

- Environment variables in Compose.
- Overview of docker-compose CLI.
- Compose command-line reference.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Using Healthcheck

Using Healthcheck.sh

The healthcheck.sh script is part of the Docker Official Images of MariaDB Server. The script is part of the repository of the Docker Official Image of MariaDB Server.

The script processes a number of argument and tests, together, in strict order. Arguments pertaining to a test must occur before the test name. If a test fails, no further processing is performed. Both arguments and tests begin with a double-hyphen.

By default, (since 2023-06-27), official images will create `healthcheck@localhost`, `healthcheck@127.0.0.1`, `healthcheck@::1` users with a random password and USAGE privileges. MARIADB_HEALTHCHECK_GRANTS can be used for `--replication` where additional grants are required. This is stored in `.my-healthcheck.cnf` in the datadir of the container and passed as the `--defaults-extra-file` to the healthcheck.sh script if it exists. The `.my-healthcheck.cnf` also sets `protocol=tcp` for the `mariadb` so `--connect` is effectively there on all tests.

The MARIADB_AUTO_UPGRADE=1 will regenerate the `.my-healthcheck.cnf` file if missing and recreate the healthcheck users of the database with a new random password. The current port configuration of the MariaDB container is written into this file.

The MARIADB_MYSQL_LOCALHOST_USER=1, MARIADB_MYSQL_LOCALHOST_GRANTS environment variables can also be used, but with the creation of the healthcheck user, these are backwards compatible.

Compose File Example

An example of a compose file that uses the healthcheck.sh to determine a healthy service as a dependency before starting a WordPress service:

```
version: "3"
services:
  mariadb:
    image: mariadb:lts
    environment:
      - MARIADB_DATABASE=testdb
      - MARIADB_USER=testuser
      - MARIADB_PASSWORD=password
      - MARIADB_RANDOM_ROOT_PASSWORD=1
    healthcheck:
      test: ["CMD", "healthcheck.sh", "--connect", "--innodb_initialized"]
      start_period: 10s
      interval: 10s
      timeout: 5s
      retries: 3
  wordpress:
    image: wordpress
    environment:
      - WORDPRESS_DB_HOST=mariadb
      - WORDPRESS_DB_NAME=testdb
      - WORDPRESS_DB_USER=testuser
      - WORDPRESS_DB_PASSWORD=password
    depends_on:
      mariadb:
        condition: service_healthy
```

Tests

The various health check flags used for monitoring the status and readiness of a MariaDB server.

Health Check Flag	Description	Required Privileges
`--connect`	Active when an external user can connect to the TCP port of MariaDB Server. This strictly tests the TCP connection, not authentication.	
`--innodb_initialized`	True when InnoDB has completed initializing, including any rollback or crash recovery.	The connecting user must have `USAGE` privileges.
`--innodb_buffer_pool_loaded`	Indicates that the previously saved buffer pool dump has been completely loaded, meaning the server has a hot cache ready for use.	The connecting user must have `USAGE` privileges.
`--galera_online`	Indicates that the Galera node is online (based on the `wsrep_local_state` variable). This includes states like “joining” and “donor” where it cannot serve SQL queries.	The connecting user must have `USAGE` privileges.
`--replication`	Tests a replica based on the `--replication_*` parameters. All subtests must pass: - io: The IO thread is running. - sql: The SQL thread is running. - seconds_behind_master: The replica is less than a specified number of seconds behind the primary. - sql_remaining_delay: The delayed replica is less than a specified number of seconds behind the primary's execution.	`REPLICATION_CLIENT` (before MariaDB 10.5) or `REPLICA MONITOR` (MariaDB 10.5+).
`--mariadbupgrade`	Indicates that MariaDB is upgraded to the current version.	

Parameters

To customize the health checks described previously, you can use the following parameters:

Parameter	Description
`--replication_all`	Checks all replication sources.
`--replication_name=n`	Sets the multisource connection name to be tested. This unsets `--replication_all`.
`--replication_io`	Checks if the replication IO thread is running.
`--replication_sql`	Checks if the replication SQL thread is running.
`--replication_seconds_behind_master=n`	Checks if the replication delay is less than or equal to `n` seconds.
`--replication_sql_remaining_delay=n`	Checks if the remaining SQL delay is less than or equal to `n` seconds.
`--su=n`	Changes to the specified user (`n`). Can only be used once from the default `root` user.
`--su-mysql`	Changes to the `mysql` unix user. Respawns the script, resetting all parameters, so it should be the first argument.
`--datadir=n`	Specifies the data directory location (`n`) for the `--mariadbupgrade` test.
	These parameters are passed to the `mariadb` shell for all tests except `--mariadbupgrade`.

Examples

```
healthcheck.sh --su-mysql --connect --innodb_initialized
```

Switch to `mysql` user, and check if can connect and the innodb is initialized.

```
healthcheck.sh --su-mysql --connect --replication_io --replication_sql --
replication_seconds_behind_master=600  --replication_sql_remaining_delay=30 ----replication_name=archive
--replication --replication_seconds_behind_master=10  --replication_name=channel1 --replication
```

Switch to `mysql` user, check if connections can be made. For the replication channel `archive`, ensure IO and SQL threads are running, the seconds behind master is less than `600` seconds, and the SQL remaining delay is less than `30` seconds. For the `channel1`, the seconds behind master is limited to `10` seconds maximum.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Enterprise Container Image Configuration Reference

Orchestrating a MariaDB Enterprise deployment within containerized environments involves more than just launching a database instance; it requires the precise alignment of server logic, intelligent traffic routing, and robust observability tools. This reference serves as the definitive technical roadmap for configuring the official images provided by the MariaDB Enterprise Docker Registry. It details the standardized environment variables, specialized sidecar patterns, and command-line flags required to transform raw images into a production-ready, cloud-native ecosystem.

MariaDB Enterprise Server (Standard and Tiered)

The MariaDB Enterprise Docker Registry provides Docker images for MariaDB Enterprise Server 10.6 and later. These images include standard storage engines such as InnoDB, Aria, MyISAM, and others by default.

Environment Variables

MariaDB Enterprise Server containers are primarily configured using environment variables during initialization:

- `MARIADB_ROOT_PASSWORD`: Specifies the password for the MariaDB `root` user. It can be set to a specific string, `RANDOM` to generate a password, or `EMPTY` for no password.
- `MARIADB_DATABASE`: If provided, the container creates this database on startup if it does not already exist.
- `MARIADB_USER` and `MARIADB_PASSWORD`: These must be used together to create a non-root user account with the specified credentials.
- `MARIADB_ROOT_HOST`: Defines the host from which the `root` user is allowed to connect, defaulting to `%` (any host).
- `MARIADB_UNIX_SOCKET_AUTHENTICATION`: When set to `1`, this enables Unix socket-based authentication for the root user, allowing passwordless access when connecting locally.
- `MARIADB_INITDB_SKIP_TZINFO`: Setting this to any non-empty value prevents the automatic loading of timezone data into the system tables.
- `SKIP_INIT_WSREP_OFF`: When set, the temporary server used during initialization is allowed to start with Galera Cluster replication (`wsrep`) enabled.

MariaDB Prometheus Exporter

> **Warning:**
>
> **Configuration Methodology**
> For optimal security and management, exporter configuration is executed via command-line flags or configuration files. Environment variables are utilized strictly for sensitive data, ensuring credentials remain protected from exposure in configuration logs.

The MariaDB Prometheus Exporter (based on `mysqld_exporter`) is a tool for scraping metrics from MariaDB instances.

Database Permissions

A dedicated database user is required to allow the exporter to scrape metrics:

```
CREATE USER 'exporter'@'localhost' IDENTIFIED BY 'XXXXXXXX' WITH MAX_USER_CONNECTIONS 3;
GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'localhost';
```

> Note: Limiting connections is recommended to prevent monitoring traffic from impacting server performance, though some versions like MariaDB 10.1 do not support resource limits.

Configuration and Usage

- Configuration File (`.my.cnf`): Credentials should be placed in a `[client]` section. The exporter reads this file from the current directory or a path specified by `--config.my-cnf`.
- Environment Variables: The `MYSQLD_EXPORTER_PASSWORD` variable can be used to provide the connection password securely.
- Collector Flags: Specific metric groups can be toggled using flags such as `--collect.binlog_size`, `--collect.engine_innodb_status`, and `--collect.info_schema.userstats`.
- Multi-Target Support: The exporter can act as a proxy for multiple MariaDB instances via the `/probe` endpoint using the `target` parameter. Authentication for different targets is managed using `auth_module` sections in the configuration file.

MaxScale Prometheus Exporter

> **Warning:**
>
> **Configuration Methodology**
> For optimal security and management, exporter configuration is executed via command-line flags or configuration files. Environment variables are utilized strictly for sensitive data, ensuring credentials remain protected from exposure in configuration logs.

The MaxScale Exporter collects metrics from the MaxScale REST API.

Prerequisites

- A MaxScale REST API user with read privileges must be created using `maxctrl`: `maxctrl create user monitor-user monitor-password`.

Configuration and Security

Configuration options follow a hierarchy where environment variables override command-line flags, which override configuration files.

- Environment Variables: Use `MAXSCALE_USERNAME` and `MAXSCALE_PASSWORD` to provide REST API credentials.
- Command Line Flags: Standard flags include `--maxscale-host`, `--maxscale-port` (default `8989`), and listener settings.
- TLS/SSL: Encrypted connections to MaxScale require the flags `--tls-ca-cert-file`, `--tls-private-key-file`, and `--tls-key-cert-file`. Use `--tls-insecure-skip-verify` for self-signed certificates.
- Encryption: The exporter can generate and use encrypted passwords via the `--secrets` and `--encrypt` flags.

MariaDB Enterprise nslcd Sidecar

The nslcd sidecar is utilized for integrating MariaDB with LDAP or Active Directory authentication through the Pluggable Authentication Module (PAM) framework.

Configuration Requirements

- `nslcd.conf`: This file configures the `nslcd` daemon, specifying the LDAP server URI, search base, and service account credentials (`binddn` / `bindpw`).
- Identity Management: The `nslcd` daemon must be configured to run as the `mysql` user and group (`uid mysql`, `gid mysql`).
- `nsswitch.conf`: This file determines the sources for name-service information and must include `ldap` for `passwd`, `group`, and `shadow` entries.
- Shared Socket: The daemon communicates with the MariaDB container via a shared Unix socket, typically mounted at `/var/run/nslcd`.

Integration with MariaDB

To enable this functionality, the MariaDB Custom Resource must be configured to load the `auth_pam` plugin using the `plugin_load_add` option in the `myCnf` section.

MaxScale and MaxScale Enterprise

The core MaxScale images do not rely on Docker-specific environment variables for internal configuration. Instead, MaxScale is configured using standard `.cnf` files or through the REST API. For monitoring deployments, it is paired with the MaxScale Prometheus Exporter as detailed above.

This page is licensed: CC BY-SA / Gnu FDL

Kubernetes and MariaDB

See Kubernetes Operators for MariaDB.

Official Kubernetes Operators for MariaDB: Enterprise Operator for MariaDB Server+MaxScale, CRDs/controllers, HA topologies, and backup/restore automation.

See Kubernetes Overview for MariaDB Users.

Introduction to core Kubernetes concepts relevant to database administrators, such as StatefulSets, Persistent Volumes, and Services, and how they apply to MariaDB storage and networking.

This page is licensed: CC BY-SA / Gnu FDL

Kubernetes Operators for MariaDB

Operators basically instruct Kubernetes about how to manage a certain technology. Kubernetes comes with some default operators, but it is possible to create custom operators. Operators created by the community can be found on OperatorHub.io.

Custom Operators

Kubernetes provides a declarative API. To support a specific (i.e. MariaDB) technology or implement a desired behavior (i.e. provisioning a replica), we extend Kubernetes API. This involves creating two main components:

- A custom resource.
- A custom controller.

A custom resource adds an API endpoint, so the resource can be managed via the API server. It includes functionality to get information about the resource, like a list of the existing servers.

A custom controller implements the checks that must be performed against the resource to check if its state should be corrected using the API. In the case of MariaDB, some reasonable checks would be verifying that it accepts connections, replication is running, and a server is (or is not) read only.

MariaDB Enterprise Operator

MariaDB Enterprise Operator provides a seamless way to run and operate containerized versions of MariaDB Enterprise Server and MaxScale on Kubernetes, allowing you to leverage Kubernetes orchestration and automation capabilities. This document outlines the features and advantages of using Kubernetes and the MariaDB Enterprise Operator to streamline the deployment and management of MariaDB and MaxScale instances.

> **Tip:**
>
> Find the documentation here.

MariaDB Community Operator

mariadb-operator is a Kubernetes operator that allows you to run and operate MariaDB in a cloud native way. It aims to declaratively manage MariaDB instances using Kubernetes CRDs instead of imperative commands.

It's available in both Artifact Hub and Operator Hub and supports the following features:

- Easily provision standalone MariaDB servers in Kubernetes.
- Multiple highly available topologies supported:
  - Asynchronous replication
  - Synchronous multi-master via Galera
  - Multi-cluster: cross-cluster replication for multi-region deployments, disaster recovery, and blue-green upgrades.

    - MaxScale as database proxy to load balance requests and perform failover/switchover operations
- Flexible storage configuration. Volume expansion.
- Physical backups based on mariadb-backup and Kubernetes VolumeSnapshots.
- Logical backups based on mariadb-dump.
- Multiple backup storage types: S3 compatible, Azure Blob Storage, PVCs, Kubernetes volumes and `VolumeSnapshots`.
- Flexible backup configuration: scheduling, compression, encryption, retention policy, target policy, timeout, staging area…
- Target recovery time: restore the closest available backup to the specified time.
- Bootstrap new instances from: Physical backups, logical backups, S3, Azure Blob Storage, PVCs, `VolumeSnapshots`…
- Point-In-Time-Recovery: Archive binary logs to enable point-in-time restoration and significantly reduce RPO.
- Cluster-aware rolling update: roll out replica Pods one by one, wait for each of them to become ready, and then proceed with the primary Pod, using `ReplicasFirstPrimaryLast`.
- Zero-downtime blue-green updates: leverage the multi-cluster topology to maintain two identical cluster topologies, upgrade one in the background, and switch between them without downtime.
- Manual update strategies: `OnDelete` and `Never`.
- Automated data-plane updates.
- my.cnf change detection. Automatically trigger updates when my.cnf changes.
- Suspend operator reconciliation for maintenance operations.
- Maintenance mode for safe operational windows: cordon, drain connections, and read-only modes.
- Issue, configure and rotate TLS certificates and CAs.
- Automated root password rotation by updating the referenced `Secret`.
- Native integration with cert-manager. Automatically create `Certificate` resources.
- Prometheus metrics via mysqld-exporter and maxscale-exporter.
- Native integration with prometheus-operator. Automatically create `ServiceMonitor` resources.
- Declaratively manage SQL resources: users, grants and logical databases.
- Declaratively manage resources in external MariaDB instances.
- Configure connections for your applications.
- Orchestrate and schedule sql scripts.
- Validation webhooks to provide CRD immutability.
- Additional printer columns to report the current CRD status.
- CRDs designed according to the Kubernetes API conventions.
- Install it using helm, OLM or static manifests.
- Multiple deployment modes: cluster-wide and single namespace.
- Helm charts published as OCI artifacts.
- Helm chart to deploy MariaDB clusters and its associated CRs.
- Multi-arch distroless image.
- GitOps friendly.

Please, refer to the documentation, the API reference and the example suite for further detail.

This page is licensed: CC BY-SA / Gnu FDL

Kubernetes Overview for MariaDB Users

Kubernetes, or K8s, is software to orchestrate containers. It is released under the terms of an open source license, Apache License 2.0.

Kubernetes was originally developed by Google. Currently it is maintained by the Cloud Native Computing Foundation (CNCF), with the status of Graduated Project.

For information about how to setup a learning environment or a production environment, see Getting started in Kubernetes documentation.

Architecture

Kubernetes runs in a **cluster**. A cluster runs a **workload**: a set of servers that are meant to work together (web servers, database servers, etc).

A Kubernetes cluster consists of the following components:

- Nodes run containers with the servers needed by our applications.
- Controllersconstantly check the cluster nodes current state, and compare it with the desired state.
- A Control Plane is a set of different components that store the cluster desired state and take decisions about the nodes. The Control Plane provides an API that is used by the controllers.

For more information on Kubernetes architecture, see Concepts and Kubernetes Components in Kubernetes documentation.

Nodes

A node is a system that is responsible to run one or more pods. A pod is a set of containers that run a Kubernetes workload or part of it. All containers that run in the same pod are also located on the same node. Usually identical pods run on different nodes for fault tolerance.

For more details, see Nodes in the Kubernetes documentation.

Every node must necessarily have the following components:

Component	Description
`kubelet`	`kubelet` has a set of `PodSpecs` which describe the desired state of pods. It checks that the current state of the pods matches the desired state. It especially takes care that containers don't crash.
`kube-proxy`	
`Container Runtime`	

Controllers

Controllers constantly check if there are differences between the pod's current state and their desired state. When differences are found, controllers try to fix them. Each node type controls one or more resource types. Several types of controllers are needed to run a cluster.

Most of the actions taken by the controllers user the API server in the Control Plane. However, this is not necessarily true for custom controllers. Also, some actions cannot be performed via the Control Plane. For example, if some nodes crashed, adding new nodes involves taking actions outside of the Kubernetes cluster, and controllers will have to do this themselves.

It is possible to write custom controllers to perform checks that require knowledge about a specific technology. For example, a MariaDB custom controller may want to check if replication is working by issuing SHOW REPLICA STATUS commands. This logic is specific to the way MariaDB works, and can only be implemented in a customer controller. Custom controllers are usually part of operators.

For more information, see Controllers in the Kubernetes documentation.

Control Plane

The control plane consists of the following components.

For more information about the control plane, see Control Plane Components in Kubernetes documentation.

API Server

An API Server exposes API functions both internally and externally. It is essential to coordinate Kubernetes components so that they react to node's change of state, and it allows the user to send commands.

The default implementation of the API Server is kube-apiserver. It is able to scale horizontally and to balance the load between its instances.

Component	Description
`kube-controller-manager`	Most controllers run in this component.
`etcd`	Contains all data used by a Kubernetes cluster. Taking regular backups of etcd data is advisable.
`kube-scheduler`	Decides which node should host a new pod based on criteria like resource requirements.
`cloud-controller-manager`	Implements logic and API of a cloud provider. Handles requests from the API Server and performs actions specific to a cloud vendor, like creating instances.

Clients and Tools

Kubernetes comes with a set of tools that allow us to communicate with the API server and test a cluster.

Tool	Description
`kubectl`	Enables communication with the API server and allows running commands on a Kubernetes cluster.
`kubeadm`	Facilitates the creation of a Kubernetes cluster ready to receive commands from kubectl.
`kind`	Used to create and manage test clusters on a personal machine. It creates a cluster consisting of Docker containers, requiring Docker to be installed. Available on Linux, MacOS, and Windows.
`minikube`	Runs a single-node cluster on the local machine. Available on Linux, MacOS, and Windows.

Kubernetes Resources and References

- Kubernetes website.
- Kubernetes on Wikipedia.
- Kubernetes organization on GitHub.
- The Official MariaDB Operator
- OperatorHub.io
- Kubernetes Community Forums.
- (video) MariaDB database clusters on Kubernetes, by Pengfei Ma, at MariaDB Server Fest 2020.
- Series of posts by Anel Husakovic on the MariaDB Foundation blog:
  - Start MariaDB in K8s
  - MariaDB & K8s: Communication between containers/Deployments
  - MariaDB & K8s: Create a Secret and use it in MariaDB deployment
  - MariaDB & K8s: Deploy MariaDB and WordPress using Persistent Volumes

- Create statefulset MariaDB application in K8s
- MariaDB replication using containers
- MariaDB & K8s: How to replicate MariaDB in K8s

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Vagrant and MariaDB

See Creating a Vagrantfile.

A guide on creating and configuring a `Vagrantfile` to define the characteristics of a MariaDB virtual machine, including box selection and provisioning steps.

See Vagrant Overview for MariaDB Users.

Introduction to Vagrant's workflow and terminology for database administrators, explaining how it simplifies the creation of reproducible MariaDB development environments.

See Vagrant Security.

Discusses security considerations when using Vagrant with MariaDB, such as default insecure keys, port forwarding risks, and ensuring production-grade settings are not used in dev boxes.

This page is licensed: CC BY-SA / Gnu FDL

Creating a Vagrantfile

In this page we discuss how to create a Vagrantfile, which you can use to create new boxes or machines. This content is specifically written to address the needs of MariaDB users.

A Basic Vagrantfile

A Vagrantfile is a Ruby file that instructs Vagrant to create, depending on how it is executed, new Vagrant machines or boxes. You can see a box as a compiled Vagrantfile. It describes a type of Vagrant machines. From a box, we can create new Vagrant machines. However, while a box is easy to distribute to a team or to a wider public, a Vagrantfile can also directly create one or more Vagrant machines, without generating any box.

Here is a simple Vagrantfile example:

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/bionic64"
  config.vm.provider "virtualbox"
  config.vm.provision :shell, path: "bootstrap.sh"
end
```

`Vagrant.configure("2")` returns the Vagrant configuration object for the new box. In the block, we'll use the `config` alias to refer this object. We are going to use version 2 of Vagrant API.

`vm.box` is the base box that we are going to use. It is Ubuntu BionicBeaver (18.04 LTS), 64-bit version, provided by HashiCorp. The schema for box names is simple: the maintainer account in Vagrant Cloud followed by the box name.

We use `vm.provision` to specify the name of the file that is going to be executed at the machine creation, to provision the machine. `bootstrap.sh` is the conventional name used in most cases.

To create new Vagrant machines from the Vagrantfile, move to the directory that contains the Vagrant project and run:

```
vagrant up
```

To compile the Vagrantfile into a box:

```
vagrant package
```

These operations can take time. To preventively check if the Vagrantfile contains syntax errors or certain types of bugs:

```
vagrant validate
```

Providers

A provider allows Vagrant to create a Vagrant machine using a certain technology. Different providers may enable a virtual machine manager (VirtualBox, VMWare, Hyper-V…), a container manager (Docker), or remote cloud hosts (AWS, Google Compute Engine…).

Some providers are developed by third parties. app.vagrant.com supports search for boxes that support the most important third parties providers. To find out how to develop a new provider, see Plugin Development: Providers.

Provider options can be specified. Options affect the type of Vagrant machine that is created, like the number of virtual CPUs. Different providers support different options.

It is possible to specify multiple providers. In this case, Vagrant will try to use them in the order they appear in the Vagrantfile. It will try the first provider; if it is not available it will try the second; and so on.

Here is an example of providers usage:

```
Vagrant.configure("2") do |config|
    config.vm.box = "hashicorp/bionic64"
    config.vm.provider "virtualbox" do |vb|
        vb.customize ["modifyvm", :id, "--memory", 1024 * 4]
    end
    config.vm.provider "vmware_fusion"
end
```

In this example, we try to use VirtualBox to create a virtual machine. We specify that this machine must have 4G of RAM (1024M * 4). If VirtualBox is not available, Vagrant will try to use VMWare.

This mechanism is useful for at least a couple of reasons:

- Different users may use different systems, and maybe they don't have the same virtualization technologies installed.
- We can gradually move from one provider to another. For a period of time, some users will have the new virtualization technology installed, and they will use it; other users will only have the old technology installed, but they will still be able to create machines with Vagrant.

Provisioners

We can use different methods for provisioning. The simplest provisioner is `shell`, that allows one to run a Bash file to provision a machine. Other provisioners allow setting up the machines using automation software, including Ansible, Puppet, Chef and Salt.

To find out how to develop a new provisioner, see Plugin Development: Provisioners.

The `shell` Provisioner

In the example above, the shell provisioner runs bootstrap.sh inside the Vagrant machine to provision it. A simple bootstrap.sh may look like the following:

```
#!/bin/bash

apt-get update
apt-get install -y
```

To find out the steps to install MariaDB on your system of choice, see the Getting, Installing, and Upgrading MariaDB section.

You may also want to restore a database backup in the new Vagrant machine. In this way, you can have the database needed by the application you are developing. To find out how to do it, see Backup and Restore Overview. The most flexible type of backup (meaning that it works between different MariaDB versions, and in some cases even between MariaDB and different DBMSs) is a dump.

On Linux machines, the `shell` provisioner uses the default shell. On Windows machines, it uses PowerShell.

Uploading Files

If we use the `shell` provisioner, we need a way to upload files to the new machine when it is created. We could use the `file` provisioner, but it works by connecting the machine via ssh, and the default user doesn't have permissions for any directory except for the synced folders. We could change the target directory owner, or we could add the default user to a group with the necessary privileges, but these are not considered good practices.

Instead, we can just put the file we need to upload somewhere in the synced folder, and then copy it with a shell command:

```
cp ./files/my.cnf /etc/mysql/conf.d/
```

Provisioning Vagrant with Ansible

Here is an example of how to provision a Vagrant machine or box by running Ansible:

```
Vagrant.configure("2") do |config|
  ...
  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "vagrant.yml"
  end
end
```

With the Ansible provisioner, Ansible runs in the host system and applies a playbook in the guest system. In this example, it runs a playbook called `vagrant.yml`. The Ansible Local provisioner runs the playbook in the vagrant machine.

For more information, see Using Vagrant and Ansible in the Ansible documentation. For an introduction to Ansible for MariaDB users, see Ansible and MariaDB.

Provisioning Vagrant with Puppet

To provision a Vagrant machine or box by running Puppet:

```
Vagrant.configure("2") do |config|
  ...
  config.vm.provision "puppet" do |puppet|
    puppet.manifests_path = "manifests"
    puppet.manifest_file = "default.pp"
  end
end
```

In this example, Puppet Apply runs in the host system and no Puppet Server is needed. Puppet expects to find a `manifests` directory in the project directory. It expects it to contain `default.pp`, which will be used as an entry

point. Note that `puppet.manifests_path` and `puppet.manifest_file` are set to their default values.

Puppet needs to be installed in the guest machine.

To use a Puppet server, the `puppet_server` provisioner can be used:

```
Vagrant.configure("2") do |config|
  ...
  config.vm.provision "puppet_server" do |puppet|
    puppet.puppet_server = "puppet.example.com"
  end
end
```

See the Puppet Apply provisioner and the Puppet Agent Provisioner.

For an introduction to Puppet for MariaDB users, see Puppet and MariaDB.

Sharing Files Between the Host and a Guest System

To restore a backup into MariaDB, in most cases we need to be able to copy it from the host system to the box. We may also want to occasionally copy MariaDB logs from the box to the host system, to be able to investigate problems.

The project directory (the one that contains the Vagrantfile) by default is shared with the virtual machine and mapped to the `/vagrant` directory (the synced folder). It is a good practice to put there all files that should be shared with the box when it is started. Those files should normally be versioned.

The synced folder can be changed. In the above example, we could simply add one line:

```
config.vm.synced_folder "/host/path", "/guest/path"
```

The synced folder can also be disabled:

```
config.vm.synced_folder '.', '/vagrant', disabled: true
```

Note that multiple Vagrant machines may have synced folders that point to the same directory on the host system. This can be useful in some cases, if you prefer to test some functionalities quickly, rather that replicating production environment as faithfully as possible. For example, to test if you're able to take a backup from one machine and restore it to another, you can store the backup in a common directory.

Network Communications

It is often desirable for a machine to be able to communicate with "the outside". This can be done in several ways:

- Private networks;
- Public networks;
- Exposing ports to the host.

Remember that Vagrant doesn't create machines itself; instead, it asks a provider to create and manage them. Some providers support all of these communication methods, while others may only support some of them, or even none at all. When you create a Vagrantfile that uses one of these networking features, it is implicit that this can only happen if the provider you are using supports them. Check your provider's documentation to find out which features it supports.

The default provider, VirtualBox, supports all of these communication methods, including multiple networks.

Private Networks

A private network is a networks that can only be accesses by machines that run on the same host. Usually this also means that the machines must run on the same provider (for example, they all must be VirtualBox virtual machines).

Some providers support multiple private networks. This means that every network has a different name and can be accessed by different machines.

The following line shows how to create or join a private network called “example”, where this machine’s IP is assigned by the provider via DHCP:

```
config.vm.network 'private_network', name: 'example', type: 'dhcp'
```

While this is very convenient to avoid IP conflicts, sometimes you prefer to assign some IP’s manually, in this way:

```
config.vm.network 'private_network', name: 'example', ip: '111.222.111.222'
```

Public Networks

As explained above, public networks are networks that can be accessed by machines that don’t run on the same host with the same provider.

To let a machine join a public network:

```
# use provider DHCP:
config.vm.network "public_network", use_dhcp_assigned_default_route: true

# assign ip manually:
config.vm.network "public_network", ip: "111.222.111.222"
```

To improve security, you may want to configure a gateway:

```
config.vm.provision "shell", run: "always", inline: "route add default gw 111.222.111.222"
```

Exposing Ports

Vagrant allows us to map a TCP or UDP port in a guest system to a TCP or UDP port in the host system. For example, you can map a virtual machine port 3306 to the host port 12345. Then you can connect MariaDB in this way:

```
mariadb -hlocalhost -P12345 -u<user> -p<password>
```

You are not required to map a port to a port with a different number. In the above example, if the port 3306 in your host is not in use, you are free to map the guest port 3306 to the host port 3306.

There are a couple of caveats:

- You can’t map a single host port to multiple guest ports. If you want to expose the port 3306 from multiple Vagrant machines, you’ll have to map them to different host ports. When running many machines this can be hard to maintain.
- Ports with numbers below 1024 are privileged ports. Mapping privileged ports requires root privileges.

To expose a port:

```
config.vm.network 'forwarded_port', guest: 3306, host: 3306
```

Use Cases

Suppose you run MariaDB and an application server in two separate Vagrant machines. It's usually best to let them communicate via a private network, because this greatly increases your security. The application server will still need to expose ports to the host, so the application can be tested with a web browser.

Suppose you have multiple environments of the same type, like the one described above. They run different applications that don't communicate with each other. In this case, if your provider supports this, you will run multiple private networks. You will need to expose the applications servers ports, mapping them to different host ports.

You may even want to implement different private networks to create an environment that reflects production complexity. Maybe in production you have a cluster of three MariaDB servers, and the application servers communicate with them via a proxy layer (ProxySQL, HAProxy, or MaxScale). So the applications can communicate with the proxies, but have no way to reach MariaDB directly. So there is a private network called "database" that can be accessed by the MariaDB servers and the proxy servers, and another private network called "application" that can be accessed by the proxy servers and the application servers. This requires that your provider supports multiple private networks.

Using public networks instead of private one will allow VMs that run on different hosts to be part of your topology. In general this is considered as an insecure practice, so you should probably ask yourself if you really need to do this.

References

The vagrant-mariadb-examples repository is an example of a Vagrantfile that creates a box containing MariaDB and some useful tools for developers.

Further information can be found in Vagrant documentation.

- Vagrantfile.
- Providers.
- Synced Folders.
- Ansible Provisioner.
- Puppet Apply Provisioner.
- Puppet Agent Provisioner.

See also Ruby documentation.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Vagrant Overview for MariaDB Users

Vagrant is a tool to create and manage development machines (Vagrant *boxes*). They are usually virtual machines on the localhost system, but they could also be Docker containers or remote machines. Vagrant is open source software maintained by HashiCorp and released under the MIT license.

Vagrant benefits include simplicity, and a system to create test boxes that is mostly independent from the technology used.

For information about installing Vagrant, see Installation in Vagrant documentation.

In this page we discuss basic Vagrant concepts.

Vagrant Concepts

Following are the core Vagrant concepts:

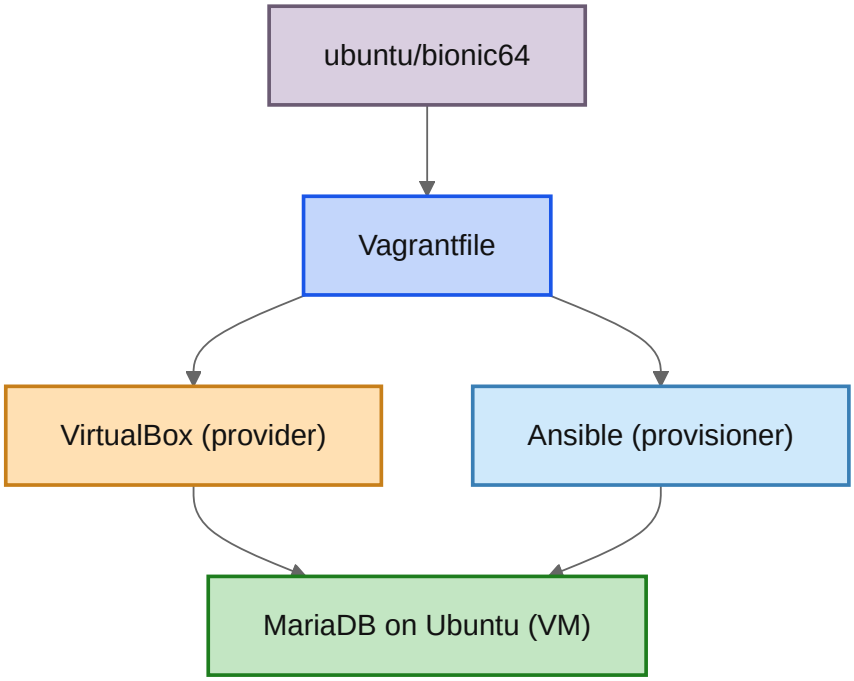
Concept	Description
Vagrant Machine	The final environment created from a box. It can be a virtual machine, a container, or a remote cloud server.
Box	A reusable package used to create Vagrant machines. It can be a base OS image downloaded from Vagrant Cloud or a custom-built template.
Provider	The underlying virtualization technology that runs the machine (e.g., VirtualBox, VMware, Hyper-V, Docker).
Provisioner	A tool or script (e.g., Shell, Ansible, Puppet) that automatically installs and configures software on the machine after it is created.

Example

The above concepts are probably easier to understand with an example.

We can use an Ubuntu box as a base to build a Vagrant machine with MariaDB. So we write a Vagrantfile for this purpose. In the Vagrantfile we specify VirtualBox as a provider. And we use the Ansible provisioner to install and configure MariaDB. Once we finish this Vagrantfile, we can run a Vagrant command to start a Vagrant machine, which is actually a VirtualBox VM running MariaDB on Ubuntu.

The following diagram should make the example clear:



Vagrant builds a machine from the ubuntu/bionic64 box using a Vagrantfile, which selects VirtualBox as the provider and Ansible as the provisioner to install MariaDB on the resulting VM.

Vagrantfiles

A Vagrantfile is a file that describes how to create one or more Vagrant machines. Vagrantfiles use the Ruby language, as well as objects provided by Vagrant itself.

A Vagrantfile is often based on a box, which is usually an operating system in which we are going to install our software. For example, one can create a MariaDB Vagrantfile based on the `ubuntu/trusty64` box. A Vagrantfile can describe a box with a single server, like MariaDB, but it can also contain a whole environment, like LAMP. For most practical use cases, having the whole environment in a single box is more convenient.

Boxes can be searched in Vagrant Cloud. Most of their Vagrantfiles are available on GitHub. Searches can be made, among other things, by keyword to find a specific technology, and by provider.

Providers

A provider adds support for creating a specific type of machines. Vagrant comes with several providers, for example:

- `VirtualBox` allows one to create virtual machines with VirtualBox.
- `Microsoft-Hyper-V` allows one to create virtual machines with Microsoft Hyper-V.
- Docker allows one to create Docker containers. On non-Linux systems, Vagrant will create a VM to run Docker.

Alternative providers are maintained by third parties or sold by HashiCorp. They allow one to create different types of machines, for example using VMWare.

Some examples of useful providers, recognized by the community:

- Vagrant AWS Provider.
- Vagrant Google Compute Engine (GCE) Provider.
- Vagrant Azure Provider.
- OpenVZ.
- vagrant-lxc.

If you need to create machines with different technologies, or deploy them to unsupported cloud platforms, you can develop a custom provider in Ruby language. To find out how, see Plugin Development: Providers in Vagrant documentation. The Vagrant AWS Provider was initially written as an example provider.

Provisioners

A provisioner is a technology used to deploy software to the newly created machines.

The simplest provisioner is `shell`, which runs a shell file inside the Vagrant machine. `powershell` is also available.

Other providers use automation software to provision the machine. There are provisioners that allow one to use Ansible, Puppet, Chef or Salt. Where relevant, there are different provisioners allowing the use of these technologies in a distributed way (for example, using Puppet apply) or in a centralized way (for example, using a Puppet server).

It is interesting to note that there is both a Docker provider and a Docker provisioner. This means that a Vagrant machine can be a Docker container, thanks to the `docker` provisioner. Or it could be any virtualisation technology with Docker running in it, thanks to the `docker` provisioner. In this case, Docker pulls images and starts containers to run the software that should be running in the Vagrant machine.

If you need to use an unsupported provisioning method, you can develop a custom provisioner in Ruby language. See Plugin Development: Provisioners in Vagrant documentation.

Plugins

It is possible to install a plugin with this command:

```
vagrant plugin install <plugin_name>
```

A Vagrantfile can require that a plugin is installed in this way:

```
require 'plugin_name'
```

A plugin can be a Vagrant plugin or a Ruby gem installable from rubygems.org. It is possible to install a plugin that only exists locally by specifying its path.

Changes in Vagrant 3.0

HashiCorp published an article that describes its plans for Vagrant 3.0.

Vagrant will switch to a client-server architecture. Most of the logic will be stored in the server, while the development machines will run a thin client that communicates with the server. It will be possible to store the configuration in a central database.

Another notable change is that Vagrant is switching from Ruby to Go. For some time, it will still be possible to use Vagrantfiles and plugins written in Ruby. However, in the future Vagrantfiles and plugins should be written in one of the languages that support gRPC (not necessarily Go). Vagrantfiles can also be written in HCL, HashiCorp Configuration Language.

Vagrant Commands

This is a list of the most common Vagrant commands. For a complete list, see Command-Line Interface in Vagrant documentation.

To list the available machines:

```
vagrant box list
```

To start a machine from a box:

```
cd /box/directory
vagrant up
```

To connect to a machine:

```
vagrant ssh
```

To see all machines status and their id:

```
vagrant global-status
```

To destroy a machine:

```
vagrant destroy <id>
```

Vagrant Resources and References

Here are some valuable websites and pages for Vagrant users.

- Vagrant Up.
- app.vagrantup.com.
- Vagrant Community.
- Vagrant on Wikipedia.
- Vagrant on HashiCorp Learn.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Vagrant Security

Databases typically contain information to which access should be restricted. For this reason, it's worth discussing some security concerns that Vagrant users should be aware of.

Access to the Vagrant Machine

By default, Vagrant machines are only accessible from the localhost. SSH access uses randomly generated key pairs, and therefore it is secure.

The password for `root` and `vagrant` is “vagrant” by default. Consider changing it.

Synced Folders

By default, the project folder in the host system is shared with the machine, which sees it as `/vagrant`. This means that whoever has access to the project folder also has read and write access to the synced folder. If this is a problem, make sure to properly restrict the access to the synced folder.

If we need to exchange files between the host system and the Vagrant machine, it is not advisable to disable the synced folder. This is because the only alternative is to use the `file` provider, which works by copying files to the machine via ssh. The problem is that the default ssh user does not have permissions to write to any directory by default, and changing this would be less secure than using a synced folder.

When a machine is provisioned, it should read the needed files from the synced folder or copy them to other places. Files in the synced folder should not be accessed by the Vagrant machine during its normal activities. For example, it is fine to load a dump from the synced folder during provisioning; and it is fine to copy configuration files from the synced folder to directories in `/etc` during provisioning. But it is a bad practice to let MariaDB use table files located in the synced folder.

Reporting Security Bugs

Note that security bugs are not reported as normal bugs. Information about security bugs are not public. See Security at HashiCorp for details.

Content initially contributed by Vettabase Ltd.

This page is licensed: CC BY-SA / Gnu FDL

Server Monitoring & Logs

See Overview of MariaDB Logs.

An introductory guide to the various logs available in MariaDB, including the Error Log, General Query Log, Slow Query Log, and Binary Log, and how to enable or disable them.

See Securing MariaDB Logs.

Learn how to harden MariaDB log files by implementing at-rest encryption, TLS for transit, strict OS permissions, and automated rotation to ensure data integrity and regulatory compliance.

See Binary Log.

Understand the binary log in MariaDB Server. This section explains its role in replication and point-in-time recovery, detailing its format, management, and use for data integrity.

See Error Log.

Complete Error Log guide for MariaDB. Complete reference documentation for implementation, configuration, and usage with comprehensive examples and best.

See General Query Log.

Complete General Query Log guide for MariaDB. Complete reference documentation for implementation, configuration, and usage with comprehensive examples.

See MyISAM Log.

Explains the MyISAM log (`myisam.log`), a specialized log for recording changes to MyISAM tables for debugging purposes, enabled via the `--log-isam` option.

See Rotating Logs on Unix and Linux.

A guide to using the `logrotate` utility on Linux to manage MariaDB log files, ensuring they don't consume excessive disk space by rotating, compressing, and archiving them.

See Slow Query Log.

Utilize the slow query log in MariaDB Server. This section helps you identify and optimize inefficient queries, improving overall database performance and responsiveness.

See SQL Error Log Plugin.

Documentation for the SQL Error Log Plugin, which allows logging of errors sent to clients to a file, enabling analysis of application-side errors that might otherwise be missed.

See Transaction Coordinator Log.

Explains the Transaction Coordinator Log (tc.log), used to maintain consistency in distributed transactions (XA) across multiple storage engines or servers.

See Writing Logs Into Tables.

Instructions on directing the General Query Log and Slow Query Log to tables (`mysql.general_log`, `mysql.slow_log`) instead of files using the `log_output=TABLE` system variable.

This page is licensed: CC BY-SA / Gnu FDL

Overview of MariaDB Logs

There are many variables in MariaDB that you can use to define what to log and when to log.

This page will give you an overview of the different logs and how to enable/disable logging to these.

> **Note:**
>
> Storage engines and plugins can have their logs too. For example, InnoDB keeps an Undo Log and a Redo Log which are used for rollback and crash recovery. However, this page only lists MariaDB server logs.

Standardized Log Naming

Use the –log-basename startup setting to establish a common base name for all log files. This option configures the base name for the *Error Log*, *General Query Log*, *Slow Query Log*, and *Binary Log* simultaneously. Using a standardized base name ensures consistency across different logs and simplifies management in multi-instance or clustered environments.

Error Log

- Always enabled by default.
- Records all critical errors encountered by the server.
  - Useful for diagnosing startup failures, crashes, and internal server errors.
- Can be configured to record varying levels of warnings and notes.

General Query Log

- Records a record of when clients connect or disconnect, and every SQL statement received from clients.
- Useful for debugging queries and monitoring client activity.

- For security and compliance auditing, use the MariaDB Audit Plugin or MariaDB Enterprise Audit instead.

Slow Query Log

- Records all SQL queries that take more than a defined threshold of time to execute.
- Useful for identifying performance bottlenecks and queries that require optimization.
- Can be filtered or sampled to reduce log volume on high-traffic servers.

Binary Log

- Records all statements that change data (DML and DDL) in a binary format.
- Essential for replication topologies where the server acts as a primary.
- Required for point-in-time recovery and database restoration.

Parsing Reference for Tool Developers

To support the development of log parsers (e.g., Fluentd, Logstash, Splunk), the following tables standardize identification fields and structural characteristics across all MariaDB Server logs.

Cross-Log Field Standardization

The following data points are represented differently across various logs but refer to the same internal identifiers.

Standardized Name	General/Slow Log Field	Audit Log Field	Error Log Field	Binary Log (Text)
Thread ID	`thread_id`	`connectionid`	`Thread ID`	`at {ID}`
Server ID	`server_id`	N/A (Uses Hostname)	N/A (Uses Process ID)	`server id`

Structural Characteristics for Parsers

Log Type	Primary Format	Record Start Indicator	Multi-line Support
Error Log	Positional String	Timestamp (`YYYY-MM-DD`)	No (Except Stack Traces)
General Query Log	Positional String	Timestamp	Yes (SQL Payloads)
Slow Query Log	Tagged Header	`# User@Host:`	Yes (Always)
Binary Log (Text)	Tagged Header	`# at {Position}`	Yes (Event Chunks)
Enterprise Audit	CSV	N/A (Standard Line)	No

Parser Developer Checklist

- Handle Multi-line Records: Assume that General and Slow Query logs will contain newlines within SQL statements.
- Standardize Identifiers: Always map `connectionid` and `thread_id` to a single `thread_id` field in your monitoring backend to allow cross-log correlation.
- Identify Record Starts: For Slow Query logs, use `# User@Host:` as the only reliable indicator of a new record.

This page is licensed: CC BY-SA / Gnu FDL

Securing MariaDB Logs

Introduction

In any hardened database environment, logs are more than just a diagnostic tool; they are a critical trail of evidence and a blueprint of your data's movement. Ensuring the security of these logs is paramount for maintaining data integrity, achieving regulatory compliance (such as GDPR or HIPAA), and performing effective forensic analysis in the event of a breach.

Log security involves protecting data in two distinct states:

- Data-at-Rest: Preventing unauthorized users or malicious actors from reading or tampering with log files stored on the disk.
- Data-in-Transit: Ensuring that when logs are moved—specifically during replication—they cannot be intercepted or spoofed.

Built-in Security Mechanisms

MariaDB provides several native features to help you secure this information:

- Access Control: Replication relies on a strictly defined username and password. Both the primary and the replica require specific privileges (`REPLICATION REPLICA`) to transfer, receive, and read log data.
- Native Encryption: You can protect the contents of your binary logs and relay logs at the storage level by enabling the `encrypt_binlog=ON` system variable.
- Database Table Security: While this guide focuses on file-based logging, MariaDB allows the General and Slow Query logs to be written to internal database tables. When logged to tables, these records inherit the same robust security, backup, and access control (via `GRANT` statements) as any other data in your database.
- On Linux, specifying `log_error` without a log file location is secure by default, because the systemd journal is used. See this section for context.

Log File Locations

By default, MariaDB log files are located within the Data Directory (`datadir`; often `/var/lib/mysql/` on Linux systems). However, for both security and performance reasons, it is common practice to move these files to a dedicated, restricted volume.

This page focuses on the security of log *files*, providing best practices for file system permissions, encryption, and rotation strategies to ensure your “digital paper trail” remains uncompromised.

To control where logs are written, MariaDB uses the global system variable `log_output`. The possible values are:

- `FILE`: (Default) Logs to a file.
- `TABLE`: Logs to the `mysql.general_log` and `mysql.slow_log` tables.
- `NONE`: Disables logging even if the logs are globally enabled.
- `FILE,TABLE`: Logs to both destinations simultaneously.

Table logging can be enabled for these log types:

1. General Query Log: This logs all established client connections and all statements received from clients. Table logging uses the CSV storage engine by default. However, this can be changed; see Writing Logs Into Tables.
2. Slow Query Log: This logs all SQL statements that took more than `long_query_time` seconds to execute and required at least `min_examined_row_limit` rows to be examined. Table logging uses the CSV storage engine by default. However, this can be changed; see Writing Logs Into Tables.
3. Binary Log (as of MariaDB 12.3): See InnoDB-Based Binary Log.

Since MariaDB is cross-platform, securing the physical log files requires a “defense-in-depth” approach tailored to each operating system’s permission model. The goal is the same across all platforms: only the MariaDB service account should have read/write access, and only authorized administrators should have read access.

Implementation Pro-Tip: The Principle of Least Privilege

When setting up these logs, always remember that the MariaDB Service Account (for instance, `mysql` or `NT Service\MariaDB`) is the only entity that needs write access. Human administrators should only have read access, and only when necessary for troubleshooting.

Summary of This Guide

By following this guide, you move from default, “open” logging to a hardened architecture:

- Logs are isolated on a dedicated partition.
- Access is restricted via OS-level permissions.
- Content is encrypted both on disk and over the wire.
- Retention is automated to prevent disk exhaustion.
- Activity is audited to detect tampering.

Log Security Best Practices in a Nutshell

If you only have five minutes to harden your MariaDB logging environment, focus on these five pillars:

- Isolation: Move all log files (`log_error` , `general_log_file` , etc.) out of the default data directory and onto a dedicated, encrypted partition to prevent “Disk Full” stalls and unauthorized access.
- Encryption: Enable `encrypt_binlog = ON` to protect your data-at-rest and enforce `require_secure_transport = ON` to ensure log data-in-transit (replication) is wrapped in TLS/SSL.
  As of MariaDB 11.4, TLS is enabled, even on a replication connection, by default (certificate-free TLS).
- Least Privilege: Restrict file-system permissions so that only the MariaDB service account can write to logs, and only high-level administrators can read them.
- Automated Rotation: Use `logrotate` (Linux) or scripts (Windows) for file-based logs, and `binlog_expire_logs_seconds` for binary logs to ensure you don’t run out of storage.
- Integrity Monitoring: Use the MariaDB Audit Plugin to log any attempts to disable logging or modify security-sensitive system variables.

Key Configuration Quick-Reference

Feature	Variable / Tool	Recommended Setting
At-Rest Encryption	`encrypt_binlog`	`ON`
In-Transit Security	`require_secure_transport`	`ON`
Linux Permissions	`chmod`	`660` for files; `750` for dirs
Log Retention	`binlog_expire_logs_seconds`	`604800` (7 Days)
Auditing	`server_audit_events`	`CONNECT, QUERY, TABLE`

Securing Log Files at the OS Level

LINUX

On Linux, the MariaDB process usually runs under a dedicated `mysql` user and group.

- Ownership: Ensure the data directory and log files are owned by the `mysql` user.
  - `chown -R mysql:mysql /var/lib/mysql`
- Permissions: Use restrictive octal permissions. Log files should generally be `660` (read/write for user/group, nothing for others) and directories should be `750` .
  - `chmod 660 /var/lib/mysql/mariadb.log`
- AppArmor/SELinux: If your distribution uses these, ensure policies are defined so that even if the MariaDB process is compromised, it cannot write logs to unauthorized locations (like `/tmp`).

WINDOWS

Windows uses Access Control Lists (ACLs) which are more granular than Linux permissions.

- Service Account: MariaDB typically runs as `NetworkService` or a specific local user.
- Inheritance: Disable permission inheritance on the `data` folder to prevent “Authenticated Users” from browsing your logs.
- Configuration:
  1. Right-click the log folder > Properties > Security > Advanced.
  2. Remove *Everyone* and *Users*.
  3. Add the specific MariaDB service account with Full Control.
  4. Add Administrators with Read/Execute.

MACOS

macOS follows a Unix-like permission structure similar to Linux, though the default user might be your local username if installed via Homebrew.

- Homebrew Path: Logs are often in `/opt/homebrew/var/mysql/`.
- Hardening: Use `chmod 700` on the log directory to ensure other local user accounts on the Mac cannot peek at your query history (which might contain sensitive data in plain text).

Sanitization of Backups

Logs are often backed up alongside data.

- Redaction: Be mindful that logs converted to `.sql` files for auditing (see this example) are now plain text and need to be deleted or encrypted immediately after use.
- Retention: Implement an automated purge policy (`expire_logs_days`) to ensure sensitive data does not persist on disk longer than legally or operationally required.

Changing the Log Location

As mentioned in the introduction, keeping logs on the same partition as your data is a security and availability risk (for instance, a “log bomb” filling up the disk and crashing the database).

To change the location, edit your `my.cnf` or `my.ini` file:

```
[mariadb]
# Example: Moving the Error Log
log_error = /var/log/mariadb/error.log

# Example: Moving the General Log
general_log_file = /var/log/mariadb/general.log
```

Note:

When moving logs on Linux, you must manually create the new directory and set the ownership to `mysql` before restarting the service, or MariaDB will fail to start due to “Permission Denied.”

Note:

On **Linux**, adding `log_error` without specifying a location logs to the `systemd` journal. This is a viable alternative to the above, and even more secure than a log file on disk, since it cannot be changed by the MariaDB process.

Because logs can grow indefinitely, an unmanaged log file is a security risk—not just for data exposure, but as a Denial of Service (DoS) vector. If a log file fills the storage partition, the MariaDB server will crash or hang.

Log Rotation and Retention Strategies

Log rotation is the process of periodically archiving the current log file and starting a fresh one. This limits the size of individual files and allows for the automatic deletion of old data after a set period.

LINUX

Linux: Using `logrotate`

On Linux, the standard way to handle this is the `logrotate` utility. It allows you to compress old logs (saving space) and set a retention policy (for instance, keep only the last 30 days).

A typical MariaDB `logrotate` configuration (usually found in `/etc/logrotate.d/mariadb`) looks like this:

```
/var/lib/mysql/*.log {
    daily
    rotate 30
    missingok
    compress
    sharedscripts
    postrotate
        # Tell MariaDB to close and reopen log files
        mariadb-admin --local flush-error-log  flush-engine-log flush-general-log flush-slow-log
    endscript
}
```

- `compress`: Encrypting logs is good, but compressing them is essential for storage.
- `postrotate`: This is critical. MariaDB holds a file handle on the log. If you move the file without telling MariaDB, it will keep writing to the old (now renamed) file. `flush-logs` forces it to start writing to the new filename.

WINDOWS

Windows: Scripted Rotation

Windows does not have a native `logrotate` equivalent. Most administrators use a PowerShell script triggered by the Task Scheduler.

A basic security workflow for Windows logs:

1. Stop Logging: Briefly disable the log via SQL (`SET GLOBAL general_log = 'OFF';`).
2. Rename: Move `mariadb.log` to `mariadb_2026_02_25.log`.
3. Start Logging: Re-enable logging; MariaDB will automatically create a new, empty file.
4. Archive: Move the old file to a secure, long-term storage volume with restricted NTFS permissions.

INTERNAL ROTATION (BINARY LOGS)

MariaDB Internal Rotation (Binary Logs)

The Binary Log is unique because MariaDB manages its rotation internally. You should never use external tools like `logrotate` on binary logs, as it will break replication.

Instead, use these system variables in your configuration file:

- `max_binlog_size`: Defines the threshold at which a new file is created (default is 1GB).
- `binlog_expire_logs_seconds`: Defines how long (in seconds) MariaDB keeps binary logs before automatically deleting them.

Security Best Practices for Rotated Logs

- Off-site Logging: For high-security environments, don't just rotate logs locally. Use a logging agent (like Fluentd or Logstash) to stream logs to a centralized, hardened log server or a SIEM (Security Information and Event Management) system.
  On Linux, an alternative is to log to systemd-journal-remote.
- Read-Only Archives: Once a log file is rotated/archived, change its permissions to Read-Only. There is no reason for the MariaDB process to have write access to a log from last month.
- Checksums: Periodically generate SHA-256 checksums of archived logs. This allows you to prove in an audit that the logs have not been tampered with since they were rotated.

Securing Binary Log Files

Generally, the same security concepts and measures apply as for other log files. Here are some extra notes on binary log files and relay log files.

Encryption at Rest

If your disk is compromised or a backup of the logs is stolen, raw binary logs are vulnerable.

- MariaDB Encryption: Enable the built-in MariaDB transparent encryption for binary logs. This ensures that even if someone copies a binary log file, they cannot decode it without the encryption keys.
- See detailed instructions for encrypting binary logs.

Securing Logs in Transit

When a replica connects to a primary server, it "pulls" binary log events. Without encryption, these events (which contain your raw SQL data) travel across the network in plain text.

Beyond the standard "use a strong password," there are three specific MariaDB security features you should definitely include to ensure log data isn't intercepted or faked during transmission.

> **Note:**
>
> SSL/TLS: Always use encrypted connections for replication traffic to prevent "sniffing" of the log data as it moves across the network.

Mandatory TLS/SSL for Replication

The most critical step is ensuring the log data is wrapped in an encrypted tunnel. You can enforce this on the primary server so it rejects any replica trying to connect without a secure certificate.

- Set `require_secure_transport = ON` globally, or specifically for the replication user:

  ```
  ALTER USER 'repl_user'@'%' REQUIRE SSL;
  ```

- This prevents "Man-in-the-Middle" (MITM) attacks where an attacker on the network could sniff the binary log stream to steal sensitive data.

Binary Log Checksums

While encryption protects against *reading* the logs, checksums protect against *corruption* or *tampering* during the transfer.

- In your configuration file (`my.cnf` or `my.ini`), set the `binlog_checksum` variable: `binlog_checksum = CRC32`
- With that setting, the primary calculates a CRC32 hash for every event in the binary log. The replica verifies this hash before applying the event. If a packet is altered or corrupted during transit, the replica stops and reports an

error rather than writing “junk” or malicious data to its database.

Master Key Rotation (for Encrypted Logs)

If you have enabled `encrypt_binlog = ON`, you must consider how the encryption keys are handled during replication.

- Only the *file* is encrypted on disk. When the log is read to be sent to a replica, it is decrypted by the primary and then re-encrypted by the network protocol (TLS).
- Security Tip: Ensure you are using a robust Key Management Service (KMS) plugin (like the AWS, HashiCorp Vault, or File Key Management plugins). Regularly rotating the Master Key ensures that, even if an old archived log file and an old key were somehow compromised, the current “live” logs remain secure.

Replica Side: Securing the Relay Log

Once the log data arrives at the replica, it is stored in Relay Logs before being executed.

- Security Risk: The replica’s relay logs are just as sensitive as the primary’s binary logs.
- Ensure `encrypt_binlog = ON` is also set on the replica. This ensures that the moment the log data hits the replica’s disk, it is encrypted immediately.

To wrap up the technical documentation, we should address the “human element.” Even with the best encryption and file permissions, simple configuration oversights can leave your logs vulnerable.

Here are the most common pitfalls to avoid when hardening MariaDB log security.

To finalize your security documentation, it is important to show users how to monitor the monitors. The MariaDB Audit Plugin is the gold standard for this, as it can track who accessed the server, what queries they ran, and—crucially—who attempted to change log settings.

Audit Policy for Log Integrity

Unlike the General Log, the Audit Plugin allows for fine-grained filtering. This prevents your logs from becoming bloated while ensuring that any “meta-changes” to your security posture are recorded.

Enabling the Plugin

First, the plugin must be installed and enabled. Add this to your configuration:

```
[mariadb]
plugin_load_add = server_audit
server_audit_logging = ON
server_audit_events = CONNECT,QUERY,TABLE
server_audit_file_path = /var/log/mariadb/audit.log
server_audit_file_rotate_size = 100M
server_audit_file_rotations = 5
```

Defining the Security Filter

You don’t want to log every `SELECT` statement, but you do want to log any attempt to disable logging or change security variables. You can filter for specific keywords.

- Filter for Sensitive Operations: You should monitor for any `SET GLOBAL` commands that affect security variables like `general_log`, `slow_query_log`, or `encrypt_binlog`.
- Filter for Privilege Changes: Use the `TABLE` event to track who is modifying the `mysql.user` or `mysql.general_log` tables.

Protecting the Audit Log Itself

The `audit.log` is your “black box” recorder. It requires the highest level of protection:

- Immutable Bit (Linux): On high-security systems, you can use `chattr +a /var/log/mariadb/audit.log` to make the file append-only. Even the `mysql` user won’t be able to delete or modify past entries, only add new ones.
- Remote Syslog: Redirect the audit events to a remote, secure syslog server so that even if the local server is wiped, the audit trail persists.

```
server_audit_output_type = SYSLOG
server_audit_syslog_facility = LOG_LOCAL6
server_audit_syslog_priority = LOG_INFO
```

Summary Table: Who Watches the Watchmen?

Action	Security Goal	Log Source
User Login/Logout	Tracking access windows.	Audit Log (`CONNECT`)
Changing `log_output`	Detecting attempts to hide tracks.	Audit Log (`QUERY`)
Direct Table Edits	Detecting tampering with `mysql.slow_log`.	Audit Log (`TABLE`)
System Errors	Detecting “Log Bomb” or disk space attacks.	Error Log

Verifying Your Security Setup

After restarting MariaDB with your new settings, run these queries to confirm the hardening is active.

Confirm Data-at-Rest Encryption

To verify that your binary logs are actually being encrypted on the disk, check the global status of the encryption threads.

```
SHOW GLOBAL VARIABLES LIKE 'encrypt_binlog';
-- This should return 'ON'

SELECT * FROM INFORMATION_SCHEMA.INNODB_TABLESPACES_ENCRYPTION WHERE NAME='file_binlog';
-- This confirms the encryption plugin is actively managing the binlog files.
```

Confirm Data-in-Transit Encryption (SSL/TLS)

If you are on a replica or a client machine, you can verify that your current connection is secure.

```
SHOW STATUS LIKE 'Ssl_cipher';
-- If this is empty, your connection is NOT encrypted.
-- It should return a cipher suite like 'TLS_AES_256_GCM_SHA384'.

SHOW SESSION STATUS LIKE 'Ssl_version';
-- Ensures you are using a modern protocol (TLSv1.2 or TLSv1.3).
```

Verify Log Destinations

Ensure the server is writing to the correct files (and not to tables, unless intended).

```
SHOW VARIABLES LIKE 'log_output';
-- Should return 'FILE'.

SHOW VARIABLES LIKE '%_log_file';
-- This lists the absolute paths for the Error, General, and Slow logs
-- so you can verify they moved to your secure partition.
```

Test the Audit Plugin

To ensure the “Black Box” is recording, perform a test action that should be caught by your filter (like changing a global variable), then check the log file.

```
-- Run a test change
SET GLOBAL slow_query_log = ON;

-- Then, from the OS terminal, check the audit log:
-- sudo tail /var/log/mariadb/audit.log
```

Common Configuration Pitfalls

Hardcoding Credentials in Configuration Files

When setting up replication or automated log rotation, it is tempting to put the username and password directly into the `[client]` or `[mariadb-dump]` sections of `my.cnf` or `my.ini`.

- This is risky since any user with read access to the config file (often world-readable by default in some environments) can steal the replication or administrative password.
- To close that loophole, use a Secret Store or restricted-access option files. If you must use a file, ensure its permissions are set to `400` or `600` and owned by the `mysql` user.

Leaving the General Log Enabled in Production

The General Query Log records *every* statement sent to the server.

- If an application sends sensitive data (like `INSERT INTO users (password) VALUES ('plain_text_pass')`), that password is now stored in plain text in your log file.
- Only enable the General Log during active debugging. For long-term auditing, use the MariaDB Audit Plugin, which allows you to filter exactly what is logged (for instance, only DDL changes) and mask sensitive information.

Neglecting the Relay Log on Replicas

Don’t just focus all security efforts on the primary server but leave the replicas wide open.

- The Relay Log on the replica contains a mirrored copy of the primary’s binary log. If an attacker gains access to the replica’s file system, they have the same data access as if they had breached the primary.
- Always mirror your security settings (permissions, `encrypt_binlog`, and `log_error` locations) across the entire replication topology.

Overlooking “Insecure Transport” Warnings

If you configure `REQUIRE SSL` for a user, but don’t set `require_secure_transport = ON` globally, the server might still allow unencrypted connections from other users or administrative tools.

- This risks accidental “downgrade” attacks where a tool connects via a non-encrypted port, exposing log-related metadata.
- To close that loophole, explicitly disable non-TLS connections in the `[mariadb]` section of your configuration.

Inadequate “Log Bomb” Protection

Failing to set a maximum size for your error logs or general logs.

- A flood of connection errors (perhaps from a brute-force attack) can cause the Error Log to balloon to hundreds of gigabytes, crashing the server by exhausting disk space.

- Always use `logrotate` (Linux) or a monitoring script (Windows) to alert you when log files exceed a specific percentage of disk capacity.

Configuration Template

Here is a hardened `my.cnf` (Linux) or `my.ini` (Windows) template. This configuration applies the “Defense in Depth” principles, covering encryption, location, and transmission security. Adjust paths and filenames according to your environment.

```
# MariaDB Log Security Hardened Template
# Target: MariaDB 10.6+

[mariadb]
# --- 1. FILE LOCATIONS & AVAILABILITY ---
# Move logs to a dedicated, restricted directory
log_error           = /var/log/mariadb/error.log
general_log_file    = /var/log/mariadb/queries.log
slow_query_log_file = /var/log/mariadb/slow.log

# --- 2. DATA-AT-REST ENCRYPTION ---
# Requires a Key Management Plugin (e.g., file_key_management)
encrypt_binlog          = ON
encrypt_tmp_disk_tables = ON

# --- 3. DATA-IN-TRANSIT SECURITY ---
# Force all connections (including replication) to use TLS/SSL
require_secure_transport = ON
ssl_ca   = /etc/mysql/ssl/ca-cert.pem
ssl_cert = /etc/mysql/ssl/server-cert.pem
ssl_key  = /etc/mysql/ssl/server-key.pem

# --- 4. INTEGRITY & RETENTION ---
# Enable checksums to detect tampering or corruption during transit
binlog_checksum     = CRC32

# Set binary log expiration (e.g., 7 days in seconds)
# 7 days * 24h * 60m * 60s = 604800
binlog_expire_logs_seconds = 604800
max_binlog_size = 100M

# --- 5. OPERATIONAL HYGIENE ---
# Log warnings to the error log (useful for spotting failed login attempts)
log_warnings = 2

# Ensure the slow query log only captures what is necessary
long_query_time = 2.0
log_slow_admin_statements = ON
```

Final Implementation Checklist for your Users

1. Directory Creation: Before restarting MariaDB with these settings, users must create the `/var/log/mariadb/` directory.
2. Permissions: Run `chown -R mysql:mysql /var/log/mariadb` and `chmod 750 /var/log/mariadb`.
3. Key Management: Remember that `encrypt_binlog = ON` requires an active encryption plugin; otherwise, the server will not start.

Log File Security Checklist

Security Category	Action Item	Target Logs	Platform
Access Control	Set permissions to `660` (Files) and `750` (Folders).	All Log Files	Linux / macOS
Access Control	Remove `Everyone` and `Users` groups from ACLs.	All Log Files	Windows

Security Category	Action Item	Target Logs	Platform
Data Integrity	Enable `encrypt_binlog = ON` in `my.cnf` or `my.ini`.	Binary / Relay Logs	All
Availability	Move logs to a dedicated partition/volume.	All Log Files	All
Availability	Configure `logrotate` with `compress` enabled.	Error / General / Slow	Linux
Availability	Set `binlog_expire_logs_seconds` for auto-purge.	Binary Logs	All
Audit Trail	Use `sha256sum` to baseline archived (rotated) logs.	All Archives	All
Replication	Require SSL/TLS for log transmission to replicas.	Binary / Relay Logs	All

This page is licensed: CC BY-SA / Gnu FDL

Binary Log

The binary log (binlog) contains a record of all changes to the databases.

See Overview of the Binary Log.

Introduction to the purpose and structure of the binary log, explaining how it records data changes (DML[20]) and structure changes (DDL[21]) for replication and recovery.

See Activating the Binary Log.

Instructions for enabling the binary log using the `--log-bin` option and configuring the log file basename and index file.

See Binary Log Formats.

Detailed comparison of the three binary logging formats: Statement-based (SBR), Row-based (RBR), and Mixed, including their pros, cons, and configuration via binlog_format.

See Using and Maintaining the Binary Log.

Complete binary log maintenance: `PURGE BINARY LOGS` / `RESET MASTER`, `expire_logs_days` & `binlog_expire_logs_seconds`, `FLUSH BINARY LOGS`, and `SHOW SLAVE STATUS`.

See Row Binlog Events.

The `Row_log_event` structure that row-based logging writes to the binary log, and the event types it uses to record inserts, updates, and deletes.

See InnoDB-Based Binary Log.

MariaDB 12.3 introduces a new binary log implementation that stores binlog events directly in InnoDB-managed tablespaces instead of separate files on disk.

See Compressing Events to Reduce Size of the Binary Log.

Guide to using the `log_bin_compress` system variable to compress binary log events, reducing storage usage and network bandwidth during replication.

See Flashback.

Explains how to use the Flashback feature (via `mysqlbinlog --flashback`) to rollback transactions by reversing the binary log events, useful for recovering from accidental data modifications.

See Group Commit for the Binary Log.

Describes the group commit optimization, which improves performance by committing multiple transactions to the binary log in a single disk I/O operation.

See Relay Log.

Overview of the relay log, a set of log files created by a replica server to store events received from the primary's binary log before executing them.

See Selectively Skipping Replication of Binlog Events.

Learn techniques to bypass specific replication events. This guide explains how to ignore individual transactions or errors to restore replication flow after a stoppage.

This page is licensed: CC BY-SA / Gnu FDL

Overview of the Binary Log

The binary log contains a record of all changes to the databases, both data and structure, as well as how long each statement took to execute. It consists of a set of binary log files and an index.

This means that statements such as CREATE, ALTER, INSERT, UPDATE and DELETE are logged, whereas statements that have no effect on the data, such as `SELECT` and `SHOW`, are not be logged. If you want to log these, too (at a cost in performance), use the general query log.

If a statement may potentially have an effect, but doesn't, such as an `UPDATE` or `DELETE` that returns no rows, it is still logged (this applies to the default statement-based logging, not to row-based logging - see Binary Log Formats).

The purpose of the binary log is:

- To enable replication, where data is sent from one or more primary servers to one or more replica servers based on the contents of the binary log.
- To assist in backup operations.

A MariaDB server with the binary log enabled runs slightly more slowly.

It is important to protect the binary log, as it may contain sensitive information, including passwords.

Binary logs are stored in a binary, not plain text, format, and so are not readable with a regular editor. However, MariaDB includes mariadb-binlog, a command-line tool for plain-text processing of binary logs.

> **Note:**
>
> From MariaDB 12.3, InnoDB-based binary logs can be used. (This is configurable, and not the default.)
>
> If configured, binary logs are stored in InnoDB tablespaces, rather than binary files. This removes the need of protecting binary logs separately, since they'll "inherit" the same protection as other MariaDB database tables. Also, any other information about log **files** doesn't apply – for example, you cannot specify a storage location for binary logs stored in an InnoDB tablespace.
>
> InnoDB-based binary logs are enabled by setting `binlog_storage_engine=innodb` in the server configuration. See InnoDB-Based Binary Log for more information.

This page is licensed: CC BY-SA / Gnu FDL

Activating the Binary Log

Turning on Binary Logging

To enable binary logging, start the server with the –log-bin option. Alternatively, add this to the configuration file (for instance, `my.cnf`), then restart the server:

```
[client-server]
log_bin
```

Optionally, you can specify a **basename** (with or without a path). If you don't, MariaDB assigns a default basename (for instance, `c525d37c-b2ff-4543-b06f-87012d142d44-bin)`, derived from the UUID or hostname of the computer the server runs on. To the basename, file extensions are added, determining the nature of the log files. See this section for details.

> **Note:**
>
> From MariaDB 12.3, InnoDB-based binary logs can be used. (This is configurable, and not the default.)
>
> If configured, binary logs are written to InnoDB-managed, page-structured files (with the `.ibb` extension) that are integrated with InnoDB's redo log and crash recovery, rather than the traditional flat binary log files. This removes the need to protect the binary log separately, since it "inherits" the same crash safety as other InnoDB data, and it removes the need for expensive two-phase commit between InnoDB transactions and the binary log. Some options that apply to traditional binary log files behave differently or no longer apply – for example, `sync_binlog` is effectively ignored.
>
> InnoDB-based binary logs are enabled by setting `binlog_storage_engine=innodb` in the server configuration. See InnoDB-Based Binary Log for more information.

On server start, you can set a basename (in the example, with a path) like this:

```
mariadbd --log-bin=/var/log/mariadb/mariadb-logs
```

The following applies:

- If you specify a binary log basename with an extension (for example `.log`), the extension is silently ignored.
- If you don't specify a path with the basename, the server logs into *`datadir`*. (*`Datadir`* is determined by the value of the datadir system variable.)
- If you don't specify a basename, it is strongly recommended to use `--log-basename` or to ensure that replication doesn't stop if the hostname of the computer changes.

Verifying Logging is On

Once `log-bin` or `log_bin` is configured and the server has been restarted, binary logging is enabled. To verify that, issue this statement:

```
SHOW VARIABLES LIKE 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
```

Viewing Log Files

> **Note:**
>
> From MariaDB 12.3, InnoDB-based binary logs can be used. (This is configurable, and not the default.)
>
> If configured, binary logs are written to InnoDB-managed, page-structured files (with the `.ibb` extension) that are integrated with InnoDB's redo log and crash recovery, rather than the traditional flat binary log files. This removes the need to protect the binary log separately, since it "inherits" the same crash safety as other InnoDB data, and it removes the need for expensive two-phase commit between InnoDB transactions and the binary log. Some options that apply to traditional binary log files behave differently or no longer apply – for example, `sync_binlog` is effectively ignored.
>
> InnoDB-based binary logs are enabled by setting `binlog_storage_engine=innodb` in the server configuration. See InnoDB-Based Binary Log for more information.

To view the log files created by the server, issue this statement:

```
SHOW BINARY LOGS;
+-----------------------------------------------+-----------+
| Log_name                                      | File_size |
+-----------------------------------------------+-----------+
| c525d37c-b2ff-4543-b06f-87012d142d44-bin.000001 |       437 |
| c525d37c-b2ff-4543-b06f-87012d142d44-bin.000002 |       433 |
+-----------------------------------------------+-----------+
```

To see which log file is currently used, issue this statement:

```
SHOW BINLOG STATUS \G
*************************** 1. row ***************************
            File: c525d37c-b2ff-4543-b06f-87012d142d44-bin.000002
        Position: 433
    Binlog_Do_DB:
Binlog_Ignore_DB:
```

To find out where log files are stored, issue this statement:

```
SHOW VARIABLES LIKE 'log_bin_basename' \G
*************************** 1. row ***************************
Variable_name: log_bin_basename
        Value: /opt/homebrew/var/mysql/c525d37c-b2ff-4543-b06f-87012d142d44-bin
```

- `/opt/homebrew/var/mysql/` is the **storage location**.
- `c525d37c-b2ff-4543-b06f-87012d142d44-bin` is the **basename** of the binary log files.

Log File Organization

> **Note:**
>
> From MariaDB 12.3, InnoDB-based binary logs can be used. (This is configurable, and not the default.)
>
> If configured, binary logs are written to InnoDB-managed, page-structured files (with the `.ibb` extension) that are integrated with InnoDB's redo log and crash recovery, rather than the traditional flat binary log files. This removes the need to protect the binary log separately, since it "inherits" the same crash safety as other InnoDB data, and it removes the need for expensive two-phase commit between InnoDB transactions and the binary log. Some options that apply to traditional binary log files behave differently or no longer apply – for example, `sync_binlog` is effectively ignored.
>
> InnoDB-based binary logs are enabled by setting `binlog_storage_engine=innodb` in the server configuration. See InnoDB-Based Binary Log for more information.

Knowing the **basename**, you can view the log files on the file system, too. Change directory (`cd`) to the **storage location**, and issue this command:

```
ls -1 c525d37c-b2ff-4543-b06f-87012d142d44-bin.*
c525d37c-b2ff-4543-b06f-87012d142d44-bin.000001
c525d37c-b2ff-4543-b06f-87012d142d44-bin.000001.idx
c525d37c-b2ff-4543-b06f-87012d142d44-bin.000002
c525d37c-b2ff-4543-b06f-87012d142d44-bin.000002.idx
c525d37c-b2ff-4543-b06f-87012d142d44-bin.index
```

- **The binary log index** is the file containing an `.index` extension. It is a plain-text file, containing a master list of the binary log files, in order.
  By default, the name of the index file is *basename.index*. This can be overridden with the `--log-bin-index` option.
- **The binary log files** have an extension using consecutive numbers, starting with `.000001`. (The higher the number, the newer the log file is.)
- **The binary log files for GTID binlog indexing** (available from MariaDB 11.4) have an `.idx` extension.

A new binary log file with a new extension (number) is created:

- Every time the server starts.
- When the logs are flushed with a `FLUSH LOGS` statement.
- When the maximum size for a binary log file is reached. (This is determined by max_binlog_size.)

Turning off Logging per Session

Clients with the BINLOG ADMIN privilege can disable and re-enable binary logging for the current session by setting the sql_log_bin variable:

```
SET sql_log_bin = 0; -- turns off logging
SET sql_log_bin = 1; -- turns on logging
```

Reading Log Files

> **Note:**
>
> From MariaDB 12.3, InnoDB-based binary logs can be used. (This is configurable, and not the default.)
>
> If configured, binary logs are written to InnoDB-managed, page-structured files (with the `.ibb` extension) that are integrated with InnoDB's redo log and crash recovery, rather than the traditional flat binary log files. This removes the need to protect the binary log separately, since it "inherits" the same crash safety as other InnoDB data, and it removes the need for expensive two-phase commit between InnoDB transactions and the binary log. Some options that apply to traditional binary log files behave differently or no longer apply – for example, `sync_binlog` is effectively ignored.
>
> InnoDB-based binary logs are enabled by setting `binlog_storage_engine=innodb` in the server configuration. See InnoDB-Based Binary Log for more information.

Log files, with the exception of the *index* log file, are binary-encoded. To display them in a human-readable format, change directory to the storage location of the log files, and issue this mariadb-binlog command:

```
mariadb-binlog c525d37c-b2ff-4543-b06f-87012d142d44-bin.000001
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!40019 SET @@session.max_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#260205 17:13:09 server id 1  end_log_pos 256 CRC32 0x59977cc7  Start: binlog v 4, server v 12.1.2-
MariaDB-log created 260205 17:13:09 at startup
ROLLBACK/*!*/;
…
```

To store that output permanently (for instance, for later processing), issue a command like this:

```
mariadb-binlog c525d37c-b2ff-4543-b06f-87012d142d44-bin.000001 > binlog.sql
```

Log File Security

> **Note:**
>
> From MariaDB 12.3, InnoDB-based binary logs can be used. (This is configurable, and not the default.)
>
> If configured, binary logs are written to InnoDB-managed, page-structured files (with the `.ibb` extension) that are integrated with InnoDB's redo log and crash recovery, rather than the traditional flat binary log files. This removes the need to protect the binary log separately, since it "inherits" the same crash safety as other InnoDB data, and it removes the need for expensive two-phase commit between InnoDB transactions and the binary log. Some options that apply to traditional binary log files behave differently or no longer apply – for example, `sync_binlog` is effectively ignored.
>
> InnoDB-based binary logs are enabled by setting `binlog_storage_engine=innodb` in the server configuration. See InnoDB-Based Binary Log for more information.

For instructions how to secure binary log **files**, see Securing MariaDB Logs.

Binary Log Format

There are three formats for the binary log. The default is mixed logging, which is a mix of statement-based and row-based logging. See Binary Log Formats for a detailed discussion.

See Also

- Setting sql_log_bin
- PURGE LOGS - Delete logs
- FLUSH LOGS - Close and rotate logs
- GTID binlog indexing

This page is licensed: CC BY-SA / Gnu FDL

Binary Log Formats

Supported Binary Log Formats

There are three supported formats for binary log events:

- Statement-Based Logging
- Row-Based Logging
- Mixed Logging (Default)

To see which log format is used, issue this statement:

```
SHOW VARIABLES LIKE 'binlog_format';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | MIXED |
+---------------+-------+
```

Regardless of the format, binary log events are always stored in a binary format, rather than in plain text. MariaDB includes the mariadb-binlog utility that can be used to output binary log events in a human-readable format.

You may want to set the binary log format in the following cases:

- If you execute single statements that update many rows, statement-based logging is more efficient than row-based logging for the replica to download.
- If you execute many statements that don't affect any rows, row-based logging is more efficient than statement-based logging for the replica to download.

- If you execute statements that take a long time to complete, but they ultimately only insert, update, or delete a few rows in the table, row-based logging is more efficient than statement-based logging for the replica to apply.

The default is mixed logging which is replication-safe and requires less storage space than row logging.

The storage engine API also allows storage engines to set or limit the logging format, which helps reduce errors with replicating between primaries and replicas with different storage engines.

Statement-Based Logging

When statement-based logging is enabled, statements are logged to the binary log exactly as they were executed. Temporary tables created on the primary will also be created on the replica.
This mode is only recommended where one needs to keep the binary log as small as possible, the primary and replica have identical data (including using the same storage engines for all tables), and all functions being used are deterministic (repeatable with the same arguments).
Statements and tables using timestamps or auto_increment are safe to use with statement-based logging.

This mode can be enabled by setting the binlog_format system variable to `STATEMENT`.

In certain cases when it would be impossible to execute the statement on the replica, the server will switch to row-based logging for the statement. Some cases of this are:

- When replication has been changed from row-based to statement-based and a statement uses data from a temporary table created during row-based mode. In this case, the temporary tables are not stored on the replica, so row logging is the only alternative.
- ALTER TABLE of a table using a storage engine that stores data remotely, such as the S3 storage engine, to another storage engine.
- One is using SEQUENCEs in the statement or the CREATE TABLE definition.

In certain cases, a statement may not be deterministic, and therefore not safe for replication. If MariaDB determines that an unsafe statement has been executed, it issues a warning like this:

```
[Warning] Unsafe statement written to the binary log using statement format since BINLOG_FORMAT =
STATEMENT. The statement is unsafe because it uses a LIMIT clause. This is unsafe because the set of rows
included cannot be predicted.
```

See Unsafe Statements for Statement-based Replication for more information.

If you need to execute non-deterministic statements, then it is safer to use mixed logging or row-based.

Caveats of Statement-Based Logging

Some tables, like temporary tables created in row mode, do not support statement-based logging (as the data is not in the binary log). Any statement that uses a table that does not support statement logging will use row based logging. This is to ensure that the data on master and the replica are consistent.

Mixed Logging

Mixed logging is the default binary log format.

When mixed logging is enabled, the server uses a combination of statement-based logging and row-based logging. Statement-based logging is used where possible, but when the server determines a statement may not be safe for statement-based logging, it will use row-based logging instead. See Unsafe Statements for Statement-based Replication: Unsafe Statements for a list of unsafe statements.

During one transaction, some statements may be logged with row logging while others are logged with statement-based logging.

This mode can be enabled by setting the binlog_format system variable to `MIXED`.

Row-Based Logging

When row-based logging is enabled, DML statements are **not** logged to the binary log. Instead, each insert, update, or delete performed by the statement for each row is logged to the binary log separately. DDL statements are still logged to the binary log in statement format.

Row-based logging uses more storage than the other log formats but is the safest to use. In practice mixed logging should be as safe.

If you want to be able to see the original query that was logged, you can enable annotated rows events, that is shown with mariadb-binlog, with –binlog-annotate-row-events. This option is on by default.

This mode can be enabled by setting the binlog_format system variable to `ROW`.

> **Warning:**
>
> When the primary server (master) writes a transaction to the binary log that exceeds the `max_allowed_packet` size, replication breaks, issuing an error like this:
>
> ```
> Last_IO_Error: Got fatal error 1236 from master when reading data from binary log: 'log event entry exceeded max_allowed_packet; Increase max_allowed_packet on master; the first event 'mariadb-bin.109824' at 4, the last event read from 'mariadb-bin.109825' at 15113554, the last byte read from 'mariadb-bin.109825' at 15113573.'
> ```
>
> When that happens, the only way to recover is by manually extracting the transaction from the primary server's binary log, and manually applying it to the replica server.
>
> **This issue has been addressed in MariaDB 12.3**, and is described in the following.

Splitting Large Row-Format Replication Events

> **Note:**
>
> This functionality is available from MariaDB 12.3.

The maximum value of the `max_allowed_packet` system variable is 1GB. It can be set to smaller values, but not values bigger than that. For row-based replication, this means that a `ROW`-format replication event bigger than the configured variable value cannot be written to the primary server's binary log. As a consequence, replication breaks. (See the warning above.)

To overcome this limitation (and breakage), a variable named `binlog_row_event_fragment_threshold` can be configured. A valid value for this variable would be equal to or smaller than a connecting replica's configured `slave_max_allowed_packet` value.

Once configured, the primary (master) server behaves differently when encountering large `ROW`-format replication events: When writing a transaction to the binary log that exceeds `binlog_row_event_fragment_threshold`, it splits up the transaction event into smaller chunks. (Those chunks have a type of `Partial_rows_log_event`.) None of those chunks exceed the value of the variable, and so replica servers can read binary log entries without issues.

Caveats of Row-Based Logging

When using row-based logging, some statements work different on the master.

- `DELETE FROM`` *table_name*
  - In row-based mode, the table always uses deletion row-by-row, which can take a long time if the table is big. It can also use a lot of space in the binary log.
  - In `STATEMENT` or `MIXED` mode, truncate is used, if possible (no triggers, no foreign keys etc). This is much faster and uses less space in the binary log.

Compression of the Binary Log

Compression of the binary log can be used with any of the binary log formats, but the best results come from using mixed or row-based logging. You can enable compression by using the `--log_bin_compress` startup option.

Configuring the Binary Log Format

The format for binary log events can be configured by setting the binlog_format system variable. If you have the SUPER privilege, then you can change it dynamically with SET GLOBAL. For example:

```
SET GLOBAL binlog_format='ROW';
```

You can also change it dynamically for just a specific session with SET SESSION. For example:

```
SET SESSION binlog_format='ROW';
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
binlog_format=ROW
```

Warning:

Be careful when changing the binary log format when using replication. When you change the binary log format on a server, it only changes the format for that server. Changing the binary log format on a primary has no effect on the replica's binary log format. This can cause replication to give inconsistent results or to fail.

Be careful changing the binary log format dynamically when the server is a replica and parallel replication is enabled. If you change the global value dynamically, then that does not also affect the session values of any currently running threads. This can cause problems with parallel replication, because the worker threads will remain running even after STOP REPLICA is executed. This can be worked around by resetting the slave_parallel_threads system variable. For example:

```
STOP REPLICA;
SET GLOBAL slave_parallel_threads=0;
SET GLOBAL binlog_format='ROW';
SET GLOBAL slave_parallel_threads=4;
START REPLICA;
```

For considerations when replicating temporary tables, see Replicating temporary tables.

Effect of the Binary Log Format on Replicas

A replica applies any events it gets from the primary, regardless of the binary log format. The binlog_format system variable only applies to normal (not replicated) updates.

The binary log format is upwards-compatible. This means replication should always work if the replica is the same or a newer version of MariaDB than the primary.

The mysql Database

Statements that affect the `mysql` database can be logged in a different way to that expected.

If the mysql database is edited directly, logging is performed as expected according to the binlog_format. Statements that directly edit the mysql database include INSERT, UPDATE, DELETE, REPLACE, DO, LOAD DATA INFILE, SELECT, and TRUNCATE TABLE.

If the `mysql` database is edited indirectly, statement logging is used regardless of binlog_format setting. Statements editing the `mysql` database indirectly include GRANT, REVOKE, SET PASSWORD, RENAME USER, ALTER,

DROP and CREATE (except for the situation described below).

`CREATE TABLE ... SELECT` can use a combination of logging formats. The CREATE TABLE portion of the statement is logged using statement-based logging, while the SELECT portion is logged according to the value of `binlog_format`.

Metadata Field Definitions (mariadb-binlog Output)

When using the mariadb-binlog utility to view the binary log, events are presented with a standardized header and specific metadata tags. Understanding these is essential for building replication monitors or audit tools.

Event Header Template: `# at {Position} #YYMMDD HH:MM:SS server id {ID} end_log_pos {Pos} CRC32 {Value} {Event_Type}: ...`

Field	Component	Description
Timestamp	`YYMMDD HH:MM:SS`	The time the event was recorded on the originating server.
Server ID	`Unsigned Integer`	The unique ID of the server where the event originated. This matches the `server_id` used in General and Slow Query logs.
end_log_pos	`Unsigned Integer`	The byte offset (position) of the *next* event in the log file.
Thread ID	`at {ID}`	(Positional) The connection identifier that executed the statement. Matches `Thread ID` in other logs.
GTID	`Domain-ServerID-SeqNo`	Global Transaction ID (e.g., `0-1-100`). Uniquely identifies a transaction across a replication topology.
Xid	`Unsigned Integer`	The internal transaction ID used for 2-Phase Commit (2PC) between storage engines (e.g., InnoDB) and the binary log.
cid	`Unsigned Integer`	Group Commit ID. Transactions with the same `cid` were flushed to disk in the same group.

Impact of Configuration on Log Structure

The structure and verbosity of `ROW` events are significantly altered by several system variables:

- **binlog_row_image**:
  - `FULL`: Logs all columns for both the “before” and “after” images.
  - `MINIMAL`: Logs only the changed columns in the after image and identifying columns (PK/Unique) in the before image. Parsers must handle variable column counts per event.
- **binlog_row_metadata**:
  - `MINIMAL`: Only logs data.
  - `FULL`: Includes column names and data types in the log. This is essential for parsers to decode the log without querying the live database schema.
- **encrypt_binlog**:
  - If `ON`, events are encrypted on disk. For `mariadb-binlog` to display this data, it must directly connect to the server hosting the encrypted binlogs via option `--read-from-remote-server`. `mariadb-binlog` cannot directly decrypt binary logs (see [MDEV-8813|https://jira.mariadb.org/browse/MDEV-8813] for details).

See Also

- Setting up replication
- Compressing the binary log
- Replicating temporary tables

This page is licensed: CC BY-SA / Gnu FDL

Using and Maintaining the Binary Log

See Overview of the Binary Log for a general overview of what the binary log is, and Activating the Binary Log for how to make sure it's running on your system.

Binary log file security is documented on Securing MariaDB Logs.

For details on using the binary log for replication, see the Replication section.

Purging Log Files

To delete all binary logs on the server, run the RESET MASTER command. To delete all binary logs before a certain datetime, or up to a certain number, use PURGE BINARY LOGS.

If a replica is active but has yet to read from a binary log you attempt to delete, the statement fails with an error. However, if the replica is not connected and has yet to read from a log you delete, the file is deleted, but the replica is unable to continue replicating once it connects again.

Logs can also be removed automatically with the expire_logs_days system variable. This is set to `0` by default (no removal). It can be set to a time, in days, after which a binary log file is automatically removed. Log files are only checked for being older than expire_logs_days upon log rotation, so if your binary log fills up slowly and does not reach max_binlog_size on a daily basis, you may see older logs still being kept. You can also force log rotation, and so expiry deletes, by running FLUSH BINARY LOGS on a regular basis. Always set expire_logs_days higher than any possible replica lag.

From MariaDB 10.6, the binlog_expire_logs_seconds variable allows more precise control over binlog deletion and takes precedence if both are non-zero.

> **Note:**
>
> From MariaDB 12.3, InnoDB-based binary logs can be used. (This is configurable, and not the default.)
>
> If configured, binary logs are written to InnoDB-managed, page-structured files (with the `.ibb` extension) that are integrated with InnoDB's redo log and crash recovery, rather than the traditional flat binary log files. This removes the need to protect the binary log separately, since it "inherits" the same crash safety as other InnoDB data, and it removes the need for expensive two-phase commit between InnoDB transactions and the binary log. Some options that apply to traditional binary log files behave differently or no longer apply – for example, `sync_binlog` is effectively ignored.
>
> InnoDB-based binary logs are enabled by setting `binlog_storage_engine=innodb` in the server configuration. See InnoDB-Based Binary Log for more information.

If the binary log index file has been removed, or incorrectly manually edited, all of the above forms of purging logs fail. The .index file is a plain text file and can be manually recreated or edited so that it lists only the binary log files that are present, in numeric/age order.

Examples of Log File Purging

```
PURGE BINARY LOGS TO 'mariadb-bin.000063';
```

```
PURGE BINARY LOGS BEFORE '2013-04-22 09:55:22';
```

Safely Purging Binary Log Files While Replicating

To be sure replication is not broken while deleting log files, perform the following steps:

- Get a listing of binary log files on the primary by running SHOW BINARY LOGS.
- Go to each replica server and run SHOW REPLICA STATUS to check which binary log file each replica is currently reading.

- Find the earliest log file still being read by a replica. No log files before this one will be needed.
- If you wish, make a backup of the log files to be deleted
- Purge all log files before (not including) the file identified above.

Limiting the Binlog Size

> **Note:**
>
> This feature is available from MariaDB 11.4.

You can limit the size of the binlog by setting the max_binlog_total_size system variable. If not set to zero, the total size of the binlog is stored in the binlog_disk_use status variable. It's also possible to limit the size of a single binlog file by setting max_binlog_size.

Selectively Logging to the Binary Log

By default, all changes to data or data structure are logged. This behavior can be changed by starting the server with the `--binlog-ignore-db=database_name` or `--binlog-do-db=database_name` options.

`--binlog-ignore-db=database_name` specified a database to ignore for logging purposes, while `--binlog-do-db=database_name` will not log any statements unless they apply to the specified database.

Neither option accepts comma-delimited lists of multiple databases as an option, since a database name can contain a comma. To apply to multiple databases, use the option multiple times.

`--binlog-ignore-db=database_name` behaves differently depending on whether statement-based or row-based logging is used. For statement-based logging, the server will not log any statement where the *default database* is database_name. The default database is set with the USE statement.

Similarly, `--binlog-do-db=database_name` also behaves differently depending on whether statement-based or row-based logging is used.

For statement-based logging, the server will only log statement where the *default database* is database_name. The default database is set with the USE statement.

For row-based logging, the server will log any updates to any tables in the named database/s, irrespective of the current database.

Examples of Selective Logging

Assume the server has started with the option `--binlog-ignore-db=employees`. The following example *is* logged if statement-based logging is used, and *is not* logged with row-based logging.

```
USE customers;
UPDATE employees.details SET bonus=bonus*1.2;
```

This is because statement-based logging examines the default database, in this case, `customers`. Since `customers` is not specified in the ignore list, the statement is logged. If row-based logging is used, the example is not logged, because updates are written to the tables in the `employees` database.

Assume now that the server started with the option `--binlog-do-db=employees`. The following example *is not* logged if statement-based logging is used, and *is* logged with row-based logging.

```
USE customers;
UPDATE employees.details SET bonus=bonus*1.2;
```

This is again because statement-based logging examines the default database, in this case, `customers`. Since `customers` is not specified in the do list, the statement is not logged. If row-based logging is used, the example is

logged as updates are written to the tables in the `employees` database.

Effects of Full Disk Errors on Binary Logging

> **Note:**
>
> From MariaDB 12.3, InnoDB-based binary logs can be used. (This is configurable, and not the default.)
>
> If configured, binary logs are written to InnoDB-managed, page-structured files (with the `.ibb` extension) that are integrated with InnoDB's redo log and crash recovery, rather than the traditional flat binary log files. This removes the need to protect the binary log separately, since it "inherits" the same crash safety as other InnoDB data, and it removes the need for expensive two-phase commit between InnoDB transactions and the binary log. Some options that apply to traditional binary log files behave differently or no longer apply – for example, `sync_binlog` is effectively ignored.
>
> InnoDB-based binary logs are enabled by setting `binlog_storage_engine=innodb` in the server configuration. See InnoDB-Based Binary Log for more information.

If MariaDB encounters a full disk error while trying to write to a binary log file, then it will keep retrying the write every 60 seconds. Log messages will get written to the error log every 600 seconds. For example:

```
2018-11-27  2:46:46 140278181563136 [Warning] mysqld: Disk is full writing '/var/lib/mariadb-bin.00001'
(Errcode: 28 "No space left on device"). Waiting for someone to free space... (Expect up to 60 secs delay
for server to continue after freeing disk space)
2018-11-27  2:46:46 140278181563136 [Warning] mysqld: Retry in 60 secs. Message reprinted in 600 secs
```

However, if MariaDB encounters a full disk error while trying to open a new binary log file, then it will disable binary logging entirely. A log message like the following will be written to the error log:

```
2018-11-27  3:30:49 140278181563136 [ERROR] Could not open '/var/lib/mariadb-bin.00002 for logging (error
28). Turning logging off for the whole duration of the MySQL server process. To turn it on again: fix the
cause, shutdown the MySQL server and restart it.
2018-11-27  3:30:49 140278181563136 [ERROR] mysqld: Error writing file '(null)' (errno: 9 "Bad file
descriptor")
2018-11-27  3:30:49 140278181563136 [ERROR] mysqld: Error writing file '(null)' (errno: 28 "No space left
on device")
```

See Also

- PURGE LOGS

This page is licensed: CC BY-SA / Gnu FDL

Row Binlog Events

A `Row_log_event` is a fundamental structure used in Row-Based Replication (RBR).

When your database is configured to use row-based logging (`binlog_format=ROW`), the binary log doesn't store the actual SQL statements (like `UPDATE users SET status='active'`). Instead, it stores the specific changes made to individual rows. The `Row_log_event` is the container for that data.

Row_log_event

How it Works

When a transaction occurs, MariaDB records the state of the row before and/or after the change. This ensures that the replica database stays perfectly in sync with the primary, regardless of non-deterministic functions (like `NOW()` or `UUID()`) that might cause issues in statement-based replication.

The Three Main Subtypes

While *Row_log_event* is the general category, you usually see it manifested as one of these three specific events in binary logs:

1. `Write_rows` event: Triggered by an `INSERT`. It contains the data for the new rows.
2. `Update_rows` event: Triggered by an `UPDATE`. It contains both the “before” image (to find the row on the replica) and the “after” image (the new data).
3. `Delete_rows` event: Triggered by a `DELETE`. It contains the data of the deleted rows so the replica can identify exactly which ones to remove.

Anatomy of a Row_log_event

Each event contains specific metadata to ensure the data lands in the right place:

- **Table ID:** A numeric identifier mapping the event to a specific table.
- **Columns Present Bitmap:** A sequence of bits identifying which columns are actually included in the event (useful for saving space if only one column in a wide table changed).
- **Row Data:** The actual payload, encoded in a binary format.

Why Does it Matter?

- Consistency: Row-based logging is much more “expensive” in terms of log size than statement-based logging, but it is significantly more reliable for data integrity.
- Point-in-Time Recovery: Tools like `mysqlbinlog` read these events to reconstruct data if a crash occurs.
- Performance: On the primary, it saves the overhead of complex query parsing for the logs, but it can create very large log files if you perform bulk updates (for instance, updating 1 million rows creates 1 million row events).

How to View Them

Log events can be seen by inspecting binary log files with `mariadb-binlog -v` (`-v` stands for verbose). Consider having issued these statements:

```
MariaDB [test]> INSERT INTO mynation VALUES (1001, 'Wonderland', 3, '2026-12-02', 'AL', 'ALC', 5);
Query OK, 1 row affected (0.004 sec)

MariaDB [test]> UPDATE mynation SET name = 'Aliceland' WHERE name = 'Wonderland';
Query OK, 1 row affected (0.006 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [test]> DELETE FROM mynation WHERE name = 'Aliceland';
Query OK, 1 row affected (0.006 sec)
```

The binary log now contains a `Write_rows`, an `Update_rows`, and a `Delete_rows` entry:

```
mariadb-binlog -v 6f6d5cb3-8a83-4f4d-a61a-6f3816cfbf93-bin.000002
…
#### at 489
#### at 593
#260212 21:36:26 server id 1  end_log_pos 0 CRC32 0xea542edb    Annotate_rows:
#Q> INSERT INTO mynation VALUES (1001, 'Wonderland', 3, '2026-12-02', 'AL', 'ALC', 5)
#260212 21:36:26 server id 1  end_log_pos 0 CRC32 0x984e0b89    Table_map: `test`.`mynation` mapped to
number 34
#### at 658
#260212 21:36:26 server id 1  end_log_pos 0 CRC32 0x9963032c    Write_rows: table id 34 flags: STMT_END_F
…
#### at 799
#### at 886
#260212 21:37:00 server id 1  end_log_pos 0 CRC32 0xee76b32e    Annotate_rows:
#Q> UPDATE mynation SET name = 'Aliceland' WHERE name = 'Wonderland'
#260212 21:37:00 server id 1  end_log_pos 0 CRC32 0xf6c030e3    Table_map: `test`.`mynation` mapped to
number 34
#### at 951
#260212 21:37:00 server id 1  end_log_pos 0 CRC32 0x5a8bc928    Update_rows: table id 34 flags:
STMT_END_F
…
#### at 1127
#### at 1195
#260212 21:38:19 server id 1  end_log_pos 0 CRC32 0x5321ba3a    Annotate_rows:
#Q> DELETE FROM mynation WHERE name = 'Aliceland'
#260212 21:38:19 server id 1  end_log_pos 0 CRC32 0x48c7ef4f    Table_map: `test`.`mynation` mapped to
number 34
#### at 1260
#260212 21:38:19 server id 1  end_log_pos 0 CRC32 0xd4ba019d    Delete_rows: table id 34 flags:
STMT_END_F
```

Partial_rows_log_event

> **Note:**
>
> This log event is available from MariaDB 12.3.

Overview

The `Partial_rows_log_event` was introduced to solve the issue of oversized binary log events. Previously, a single `Rows_log_event` containing a large number of row changes (or very large `BLOB` data) had to be written and transmitted as a single, contiguous block.

A `Partial_rows_log_event` represents a fragment of a larger logical row change. When a standard `Rows_log_event` exceeds the system-defined threshold, MariaDB splits the payload into a sequence of partial events.

> **Note:**
>
> The configuration of splitting log events is described here.

> **Note:**
>
> Note on memory usage: While `Partial_rows_log_event` allows the server to write large updates in fragments to stay within `max_allowed_packet` limits, it does not reduce the memory consumption of the primary server. The transaction still maintains the full row event contiguously in the binary log cache memory before it is fragmented for writing. For reducing memory overhead of large transactions, see InnoDB-based Binary Log.

How Fragmentation Works

Instead of one massive event, the binary log contains a sequence of events structured like this:

1. `Table_map_event`: Identifies the table being modified.
2. `Partial_rows_log_event` (`N` times): Each event is exactly the size of the configured maximum threshold.
3. `Rows_log_event`: The concluding fragment which contains the remaining data and signals the end of the logical group.

Key Characteristics

- Atomic Reassembly: Replicas and log parsers (like `mariadb-binlog`) must buffer all related `Partial_rows_log_event` entries until the final event in the group is received before the rows can be applied.
- Size Determinism: All partial events in a sequence (except the last one) are of a uniform size, defined by the system variable `binlog_row_event_max_size` (or the equivalent threshold setting).
- Order Preservation: Because the binary log is a sequential stream, these fragments always appear contiguously for a single statement within a transaction.

Configuration

Splitting log events is enabled by setting the `binlog_row_event_max_size` variable. See this section for details.

Use Case: Large BLOBs and Bulk Inserts

This feature is particularly beneficial for environments with:

- Limited network buffers: Prevents “packet too large” errors during replication.
- High memory concurrency: Allows the master to stream log data to disk in chunks rather than allocating one massive buffer for a multi-megabyte row change.

Best Practices & Troubleshooting: `Partial_rows_log_event`

Best Practices

- **Align with network MTU:** If you are experiencing “Packet too large” errors or network instability in your replication stream, set `binlog_row_event_max_size` to a value slightly lower than your network’s maximum transmission unit (MTU) or the `max_allowed_packet` setting.
- **Monitor disk I/O:** While fragmentation prevents massive memory allocations, writing many small `Partial_rows_log_event` entries can increase the number of small I/O operations. If you see an I/O bottleneck on the binary log disk, consider increasing the fragment size slightly.
- **Buffer memory on replicas:** Ensure your replicas have sufficient memory. Because the replica must buffer all `Partial_rows_log_event` fragments before applying the final `Rows_log_event`, very large row changes will still consume memory on the “subscriber” side during the reassembly phase.

Troubleshooting & Common Issues

1. Increased Replication Lag

If you notice an increase in replication lag after upgrading to MariaDB 12.3, check the frequency of partial events.

- Cause: The replica is spending more time “assembling” the fragments in memory before the SQL thread can execute them.
- Solution: Increase the threshold size to reduce the number of fragments per row change.

2. `mariadb-binlog` Compatibility

Older versions of `mariadb-binlog` (pre-12.3) will likely fail to parse these events or skip them entirely because they do not recognize the `Partial_rows_log_event` type code.

- Symptom: “Unknown event type” errors or truncated output when reading logs.

- Solution: Always use the `mariadb-binlog` binary that matches or exceeds the version of the MariaDB server generating the logs.

3. Monitoring Fragmentation Frequency

You can monitor how often fragmentation is occurring by checking the binary log headers. If almost every `Rows_log_event` is preceded by a `Partial_rows_log_event`, your threshold is likely set too low for your typical row size (for instance, you have many wide tables or `JSON`/`BLOB` columns).

Observation	Likely Cause	Recommended Action
Frequent small fragments	Threshold too low	Increase `binlog_row_event_max_size`
“Packet too large” errors	Threshold too high	Decrease `binlog_row_event_max_size`
High replica memory usage	Massive row updates	Break up large transactions into smaller batches

Implementation Summary

> **Warning:**
>
> The introduction of `Partial_rows_log_event` does *not* change the transactional nature of MariaDB. The fragments are part of a single unit of work; if the connection is lost mid-fragment, the replica discards the partial buffer and waits for a re-transmission or restart of that event group.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB-Based Binary Log

> **Note:**
>
> This feature is available from MariaDB 12.3.

Description

Traditionally, MariaDB treated the binary log (binlog) and the InnoDB storage engine as separate components, requiring a complex two-phase commit (2PC) protocol to keep them synchronized. With this new binlog implementation, binlog events are written to InnoDB-managed, page-structured files that are integrated with the InnoDB redo log and crash recovery. This change eliminates the need for an expensive 2PC between the binlog and InnoDB, allowing significant performance improvements and simplified crash recovery behavior.

The binlog files remain separate files on disk (with the `.ibb` extension), but instead of the traditional flat, sequential format they use InnoDB’s page format and durability machinery.

InnoDB-based binary logs entirely replace traditional file-based binary logs. When enabled, the server stops writing traditional binlog files, and the new format becomes the only binary log implementation in use.

Additionally, this feature is limited to the binary log on the server. The relay log on replicas continues to use the traditional implementation and is not affected by the use of InnoDB-based binary logs.

Benefits

The InnoDB-based binlog offers these key performance advantages:

1. **Crash-safe recovery integrated with InnoDB**: The binlog integrates with InnoDB’s crash recovery mechanism, so the binary log and InnoDB always recover to a mutually consistent state without a separate sync of the binlog. You can therefore set `--innodb-flush-log-at-trx-commit=0` or `--innodb-flush-log-at-trx-commit=2` for better performance. As with regular InnoDB tables, those settings trade durability of the most recently committed transactions for speed — but the binlog and InnoDB stay consistent with each other after a crash.

2. **Efficient commits with reduced `fsync` operations**: When using the `--innodb-flush-log-at-trx-commit=1` setting, the system performs only a single coordinated `fsync` per commit, instead of the multiple syncs required by the traditional 2PC.
3. The `sync_binlog` option is no longer required, as binlog files are now crash-safe without separate syncing.

Comparison: Traditional vs. New Binlog

Feature	Traditional Binlog	New InnoDB-Based Binlog
Files on Disk	Numbered files (e.g. `binlog.000001`) plus a `.index` listing file	`.ibb` files (e.g. `binlog-000001.ibb`)
Crash Safety	Requires `sync_binlog=1`	Native — inherited from InnoDB crash recovery
Commit Protocol	Two-Phase Commit (slower)	Integrated (faster)
Positioning	Filename/Offset or GTID	GTID only
File Allocation	Incremental	Pre-allocated (`max_binlog_size`)

When to Use InnoDB-Based Binlogs

Use **InnoDB-based binlogs** when:

- The system requires high transactional throughput with strict durability guarantees.
- Replication topologies are already using GTIDs.
- You are running MariaDB 12.3 or later with GTID replication enabled.

Stay with **traditional binlogs** when:

- The system requires Galera Cluster support.
- Applications depend on filename/offset-based replication positions.
- The environment uses third-party tools that read binlog files directly.

Enabling the InnoDB Binary Log

Configuration

To enable the InnoDB-based binlog, add these options to your MariaDB configuration file:

```
[mariadb]
log_bin
binlog_storage_engine=innodb
```

- The `log_bin` option must be specified without any value.
- The new binlog format is mutually exclusive with the traditional file-based binlog. Old binlog files are ignored after switching to the new format.

For additional configuration options, see Replication and Binary Log System Variables.

Optional Configuration

Custom Binlog Directory

The directory for binary log files can be configured with:

```
[mariadb]
binlog_directory=/path/to/binlog
```

If not specified, the data directory is used by default.

Binary Log Files

Binary log files are written to InnoDB-managed, page-structured files using the .ibb extension. Files follow this naming pattern:

```
binlog-000000.ibb
binlog-000001.ibb
```

Characteristics

- Binlog files are pre-allocated for efficiency. When binlog file N is filled up, event data continues in file N+1, and an empty file N+2 is pre-allocated in the background. Binlog files are therefore always exactly `--max-binlog-size` bytes long; if the server restarts, binlog writing continues at the point reached before shutdown.
- The file size is managed by `max_binlog_size` (by default 1GB).
- Files are page-based (16 KB pages) with a CRC32 checksum at the end of each page.
- There are no binlog index files (`.index`), GTID index files (`.idx`), or GTID state files (`.state`).
- The `--binlog-checksum` option is no longer used. Binlog files are always checksummed with a CRC32 at the end of each page. To checksum data sent over the network between primary and replica, use the `MASTER_SSL` option in `CHANGE MASTER` to enable SSL.

GTID Handling

The new binlog implementation requires GTID-based replication. Instead of GTID index and state files, the binlog periodically writes GTID state records into the binlog itself. When a replica connects (or the server starts up), the binlog is scanned from the last GTID state record to find or recover the correct GTID position.

The interval between state records is controlled by:

```
[mariadb]
innodb-binlog-state-interval=2097152
```

The interval is specified in bytes (default: 2 MB) and must be a power-of-two multiple of the binlog page size (16 KB). This option can be increased to reduce the overhead of state records, or decreased to speed up finding the initial GTID position at replica connect. The overhead is small either way, so normally there is little reason to change the default.

Note: The status variables `binlog_gtid_index_hit` and `binlog_gtid_index_miss` are not used with the new binlog implementation.

For additional configuration options, see Replication and Binary Log System Variables.

Replication

Replication configuration from the primary can be performed as usual. Requirements:

- Replicas must use GTID to connect to the primary (this is the default). The old filename/offset-based replication position is not available when using the new binlog on the primary.
- Replicas should be upgraded to at least MariaDB 12.3 before switching the primary to the new binlog format.
- The primary and replica can independently use either the traditional or new binlog format.

Viewing Binlog Events

Binary log events can be verified from within the server using SQL statements, or from outside the server using the `mariadb-binlog` client utility.

Within the Server

To inspect binary log events directly from a running server instance, use SQL statements:

```
-- View all binlog events
SHOW BINLOG EVENTS;

-- View events from a specific file
SHOW BINLOG EVENTS IN 'binlog-000001.ibb';
```

Note: Event offsets are generally reported as zero; GTID positions should be used instead for navigation.

Using the `mariadb-binlog` Utility

The `mariadb-binlog` utility allows inspection of binary log files either locally or remotely:

```
# Read binary log from a remote server
mariadb-binlog --read-from-remote-server \
               --host=master.example.com --user=repl binlog-000001.ibb

# Read a local binary log file
mariadb-binlog /path/to/binlog-000000.ibb

# Read multiple binary log files (pass all at once, in order)
mariadb-binlog /path/to/binlog-000000.ibb \
               /path/to/binlog-000001.ibb \
               /path/to/binlog-000002.ibb
```

When viewing events across multiple binlog files, all binlog files should be passed to the `mariadb-binlog` program at once in the correct order; this ensures that events that cross file boundaries are included in the output exactly once.

When using `--start-position` and `--stop-position`, it is recommended to use GTID positions. File offsets used in the traditional binlog format are not used in the new binlog and will mostly be reported as zero.

Flushing and Rotating Binary Log Files

To regulate the file size and disk use, binary log files can be manually rotated or purged:

```
-- Flush the current binary log file and create a new one
FLUSH BINARY LOGS;

-- Flush the current binary log and remove a specific GTID domain
FLUSH BINARY LOGS DELETE_DOMAIN_ID=N;

-- Remove binary log files up to a specific file
PURGE BINARY LOGS TO 'binlog-000001.ibb';
```

- `FLUSH BINARY LOGS` pads the current file to the next page boundary, truncates it, and switches to the next file. This can leave a binlog file smaller than `--max-binlog-size` (but always a multiple of the binlog page size).
- `FLUSH BINARY LOGS DELETE_DOMAIN_ID=N` removes an old domain ID from `@@gtid_binlog_pos`. This requires the domain not be in use in any existing binlog files. If the domain ID is already deleted, a warning is issued but the `FLUSH BINARY LOGS` operation still runs.

Note: This behavior differs from the old binlog implementation, where FLUSH is skipped if the domain ID was already deleted.

Automatic Purging

Binlog files can be automatically removed based on age or disk usage, provided they are no longer needed by active replication replicas or for crash recovery:

```
-- Retain binary logs for 7 days (in seconds)
SET GLOBAL binlog_expire_log_seconds = 604800;

-- Retain binary logs for 14 days
SET GLOBAL binlog_expire_log_days = 14;

-- Limit total binary log disk usage to 100 GB
SET GLOBAL max_binlog_total_size = 107374182400;

-- Allow purge in setups where no replicas are configured
slave_connections_needed_for_purge=0
```

Binlog files are only purged when limits are exceeded and files are not required for crash recovery or by active replicas.

Backup using mariadb-backup

Default Behavior

By default, binlog files are included in backups, resolving a long-standing limitation of the traditional binlog.

Key features:

- Binlog files are backed up in a transactionally consistent way, just like other InnoDB data.
- Backups are non-blocking, meaning the server continues running normally. By default, only `RESET MASTER`, `PURGE BINARY LOGS`, and `FLUSH BINARY LOGS` are blocked during the backup. This blocking can be disabled with the `--no-lock` option.
- Restored backups can serve as replicas using `MASTER_DEMOTE_TO_SLAVE=1` in `CHANGE MASTER`.

Setting Up a Replica from Backup

```
# Default backup includes binlog files
mariadb-backup --backup --target-dir=/backup/path

# Exclude binlog files (save space)
mariadb-backup --backup --target-dir=/backup/path --skip-binlog

# Restore on replica
mariadb-backup --prepare --target-dir=/backup/path
mariadb-backup --copy-back --target-dir=/backup/path

# Start MariaDB server
systemctl start mariadb
```

```
# Configure replication
CHANGE MASTER TO
   MASTER_HOST='master.example.com',
   MASTER_USER='repl',
   MASTER_PASSWORD='secret',
   MASTER_USE_GTID=slave_pos,
   MASTER_DEMOTE_TO_SLAVE=1;
START SLAVE;
```

When the `MASTER_DEMOTE_TO_SLAVE=1` option is set, the restored backup can automatically start replicating from the point the backup was taken, using the binlog files included in the backup.

Note that when binlog files are omitted from the backup (`--skip-binlog`), the restored server behaves as if `RESET MASTER` was run at the point of the backup. In that case, the `MASTER_DEMOTE_TO_SLAVE=1` option cannot be used to initialize the replication starting position; instead, the `gtid_slave_pos` variable must be set explicitly from the

information saved by `mariadb-backup`. Any transactions that were prepared but not yet committed at backup time will be rolled back on first startup.

Performance Characteristics

The InnoDB-based binary log eliminates the traditional 2PC protocol between the binary log and the InnoDB storage engine, which results in:

- Reduced commit latency.
- Removal of `sync_binlog` overhead.
- When `--innodb-flush-log-at-trx-commit=1` is set, a single synchronized flush occurs for both table data and binlog.

Commit durability is now controlled solely by –innodb-flush-log-at-trx-commit, which now applies to both binlog files and InnoDB table data.

Flush Policy

- `--innodb-flush-log-at-trx-commit=1`: Performs a single coordinated flush of both table data and binary log, offering greater efficiency compared to the traditional binlog implementation
- `--innodb-flush-log-at-trx-commit=0` or `=2`: Provides higher throughput. As with regular InnoDB tables, these settings trade durability of the most recently committed transactions for speed, but the binlog and InnoDB remain consistent with each other after a crash.

The `sync_binlog` option is effectively ignored when using the InnoDB-based binary log.

GTID State Interval

The `--innodb-binlog-state-interval` configuration option controls how often GTID state records are written to the binary log (default: 2 MB). Lowering this value helps replicas reconnect faster during high traffic. The overhead is small in either direction, so the default is usually appropriate.

Migration and Upgrades

New Installations

For new MariaDB 12.3 and later installations, add the following to your configuration file:

```
[mariadb]
log-bin
binlog-storage-engine=innodb
```

When you enable the InnoDB-based binary log on a new installation, no more migration steps are required.

Migration from Traditional Binlog

When switching an existing server to the new binlog format, the old binlog files will not be available after the switch, as the two formats are mutually exclusive. Choose the method that best suits your environment.

Method 1: Direct Restart Migration (Simple)

This method stops the server, switches the configuration to the new binlog format, and restarts with an empty binlog. It is the simplest migration approach and is applicable for setups where:

- The server is standalone or not part of an active replication topology.
- The old binary log files are no longer needed after the transition.
- Point-in-time recovery from the old binary log files is not required.

To perform a direct restart migration, follow the steps below:

1. Stop the MariaDB server.
2. If `log-bin` is already present in your configuration with a filename (e.g. `log-bin=my-old-binlog`), remove the filename so it reads as just `log-bin` without any value. Then add `binlog-storage-engine=innodb`.
3. Start the server. The new binlog starts empty.
4. Remove the old binlog files from the data directory manually.

Method 2: Replication State Migration (GTID Preservation)

This approach is used to switch a primary server while ensuring connected replicas can continue replicating without full reconfiguration. This method switches the primary to the new binlog format while preserving the GTID state, so that connected replicas can reconnect and continue replicating from where they left off without requiring a full reconfiguration.

This applies to the following types of setups:

- Active replication topologies where replicas must continue replicating after the primary switches format.
- Setups where all replicas have already been upgraded to at least MariaDB 12.3.

Follow the steps below to perform a replication state migration:

1. Stop all writes to the primary.
2. Wait for all replicas to catch up to the primary's current position.
3. Capture the current GTID state by noting down the value of `@@gtid_binlog_state`.
4. Restart the primary with the following configuration:

```
[mariadb]
log-bin
binlog-storage-engine=innodb
```

5. Immediately execute `SET GLOBAL gtid_binlog_state=<old value>` using the value saved in step 3.
6. Allow replicas to reconnect; they will continue from where they left off.

Method 3: Live Migration (Zero Downtime)

This method promotes a replica, already running the new binlog format, to become the new primary, avoiding any downtime on the original primary during the format transition.

It is applicable for the following types of setups:

- Production environments where primary downtime is not acceptable.
- Active replication topologies with at least one available replica that can be promoted to primary.
- Configurations where all replicas have been upgraded to MariaDB 12.3 or higher.

To perform a live migration, follow these steps:

1. Ensure that all replicas are upgraded to at least MariaDB version 12.3 before switching the primary.
2. Choose a replica and restart it with `--binlog-storage-engine=innodb`.
3. Allow the replica to replicate from the old-format primary until it has enough binlog data.
4. Promote this replica to be the new primary.
5. Stop the remaining replicas one at a time, update their configuration to the new binlog format, and restart them.

Using the New Binlog with 3rd-Party Programs

The new binlog uses a different on-disk format than the traditional binlog. The format of individual replication events is the same; however, the files stored on disk are page-based, and each page encapsulates event data to support

splitting events into multiple pieces.

Existing 3rd-party programs that read binlog files directly will need to be modified to support the new format. Until then, such programs require using the traditional binlog format.

The protocol for reading binlog data from a running server (e.g., for a connecting replica) is mostly unchanged. Existing programs that read binlog events from a running server may be able to function unmodified. Similarly, `mariadb-binlog --read-from-remote-server` works as usual.

Key differences for client programs:

- File offsets and file boundaries are no longer meaningful and are no longer reported to connecting clients.
- There are no rotate events at the end of a file specifying the name of the next file.
- There is no new format description event at the start of each new file.
- Effectively, the binlog appears as a single unbroken stream of events to clients.
- The starting position for receiving binlog events must be specified using a GTID position; specifying a filename and file offset is not available.

Binlog File Format Reference

This section describes the on-disk format of binlog files for developers building tools that interact with binlog files directly.

A binlog file consists of a sequence of pages. The page size is currently fixed at 16 KB. The file size is set with `--max-binlog-size`. Each page has a CRC32 in the last 4 bytes; all remaining bytes are used for data.

Numbers are stored in little-endian format. Some numbers are stored as compressed integers consisting of 1–9 bytes. The lower 3 bits determine the number of bytes used (one more than the value in the lower 3 bits, except that a value of 7 means 9 bytes are used). The stored value is the little-endian value of the used bytes, right-shifted by 3.

File Header Page

The first page in each binlog file is a file header page:

Offset	Size	Description
0	4	Magic value `0x010dfefe` identifying the file as a binlog file
4	4	Log-2 of the page size (currently fixed at 14 for 16 KB)
8	4	Major file version (currently 1). A new major version is not readable by older server versions
12	4	Minor file version (currently 0). New minor versions are backwards-compatible
16	8	File number (same as the number in the `binlog-NNNNNN.ibb` filename), for consistency check
24	8	Size of the file, in pages
32	8	InnoDB LSN corresponding to the start of the file, used for crash recovery
40	8	Value of `--innodb-binlog-state-interval` used in the file
48	8	File number of the earliest file this file may reference (used to prevent purging files still needed for large split events)
56	8	File number of the earliest file containing pending XA transactions that may still be active
64	448	Unused
512	4	Extra CRC32 (the header can be read as a 512-byte page to determine the real page size)

Data Pages and Chunks

Remaining pages are data pages. Data is structured as a sequence of records; each record consists of one or more chunks. A page contains one or more chunks; a record can span multiple pages, but a chunk always fits within one

page. Chunks are a minimum of 4 bytes; any remaining 1–3 bytes of a page are filled with `0xff`. Unused bytes are set to `0x00`.

A chunk consists of a type byte, two length bytes (little-endian, counting only data bytes), and the data bytes.

The high two bits of the type byte collect chunks into records:

- Bit 7 is clear for the first chunk in a record, and set for all following chunks.
- Bit 6 is set for the last chunk in a record, and clear for all prior chunks.

Chunk Types

Type	Description
0	Not a real type. `0x00` is an unused byte and denotes end-of-data in the current binlog file.
1	**Commit record** — contains binlog event data. Starts with a compressed integer for the count of out-of-band (oob) records referenced; if non-zero, four more compressed integers give the file number and offset of the first and last oob reference. A second similar group follows for mixed transactional/non-transactional event groups. The remainder is payload data.
2	**GTID state record** — written every `--innodb-binlog-state-interval` bytes. Contains: the number of GTIDs in the state (compressed int); one more than the earliest file number with possibly active XA transactions (or 0 for none); then N×3 compressed integers, each representing one GTID in the state.
3	**Out-of-band (oob) data record** — used to split large event groups into smaller pieces organized as a forest of perfect binary trees. Starts with 5 compressed integers: node index; file number and offset of the left child oob node; file number and offset of the right child oob node. The remainder is payload event data.
4	**Filler record** — pads out the last page of a binlog file truncated by `FLUSH BINARY LOGS`.
5	**XA PREPARE record** — for consistent crash recovery of user XA transactions. Starts with 1 byte counting the number of participating storage engines, then the XID (4-byte formatID; 1-byte gtrid length; 1-byte bqual length; XID name characters). Followed by 5 compressed integers: oob node count; file number and offset of first and last oob node.
6	**XA complete record** — for recovery of internal 2PC and user XA. First byte is type: `0` = commit, `1` = XA rollback. Followed by 6–134 bytes of XID in the same format as the XA PREPARE record.

Limitations and Unsupported Features

The following list of features are not supported when using the InnoDB-based binary log implementation:

Replication Positioning

Old-style filename/offset replication positions are not supported. Replication must use GTID-based replication (the default). The following are not available with the new binlog:

- `MASTER_POS_WAIT()` — use `MASTER_GTID_WAIT()` instead.
- `BINLOG_GTID_POS()`
- `RESET MASTER TO` (plain `RESET MASTER` remains available)

Semi-Synchronous Replication

Semi-synchronous replication is not supported in the initial implementation. The `AFTER_SYNC` option cannot be supported because the traditional 2PC no longer exists. The `--init-rpl-role` option is also not supported.

Savepoints Inside Triggers

Using savepoints inside triggers is not supported, due to bugs and inconsistencies (see MDEV-38465). Executing a `SAVEPOINT` or `ROLLBACK TO SAVEPOINT` statement inside a trigger will consistently error and roll back the entire statement.

Galera Cluster

The InnoDB-based binary log cannot be used with MariaDB Galera Cluster. Users running Galera must continue using the traditional binlog format.

XA Transaction Replication

External XA transactions are fully supported and crash-safe on the primary. The new binlog stores `XA PREPARE` and `XA COMMIT` / `XA ROLLBACK` records internally (see chunk types 5 and 6 in the Binlog File Format Reference), and the server will always recover prepared transactions to a state consistent with the binlog after a crash.

However, replicating `XA PREPARE` to a replica is not supported. This means that if the primary has transactions in a prepared-but-not-yet-committed state at the time of a failover, it is not possible to promote the replica and rescue those prepared transactions.

Binary Log Encryption

Binary log encryption (`--encrypt-binlog`) is not supported with the InnoDB-based binary log. Use full-disk encryption at the operating-system level instead, and/or SSL for the replica's connection to the primary (use the `MASTER_SSL` option in `CHANGE MASTER`).

Heuristic Recovery

The `--tc-heuristic-recover` option is not supported. Any pending prepared transactions will always be rolled back or committed to be consistent with the binlog. If the binary log is empty (i.e., deleted manually), pending transactions will be rolled back.

SHOW BINLOG EVENTS FROM Behavior

`SHOW BINLOG EVENTS FROM` does not return an error if the specified position falls in the middle of an event group. Instead, it starts from the first GTID event following the position, or returns empty if the position is past the end of the log.

`sync_binlog` Option

The `sync_binlog` option is no longer needed and is effectively ignored. Durability of commits is now controlled solely by `--innodb-flush-log-at-trx-commit` , which applies to both binlog files and InnoDB table data.

Multiple Storage Engines

In the initial version, only InnoDB is available as a storage engine for the binlog (`--binlog-storage-engine=innodb`).

See Also

- MariaDB Innovation: InnoDB-Based Binary Log • Blog post • 4 minutes
- Global Transaction IDs
- System Variables
- CHANGE MASTER TO
- InnoDB

This page is licensed: CC BY-SA / Gnu FDL

Compressing Events to Reduce Size of the Binary Log

Selected events in the binary log can be optionally compressed, to save space in the binary log on disk (or, from MariaDB 12.3, in InnoDB tablespaces) and in network transfers.

Events that can be compressed are those that can be of significant size:

- Query events (for DDL[22] and DML[23] in statement-based replication), and
- Row events (for DML[24] in row-based replication).

Compression is fully transparent. Events are compressed on the primary before being written to the binary log, and are uncompressed by the I/O thread on the replica before being written to the relay log. The mariadb-binlog command will likewise uncompress events for its output.

The `zlib` compression algorithm is used to compress events.

Compression has the greatest impact when events are of a non-negligible size, as each event is compressed individually – like batch `INSERT` statements that add many rows or large values, or row-based events that affect a number of rows in one query.

The log_bin_compress option is used to enable compression of events. Only events with data (query text or row data) above a certain size are compressed; the limit is set with the log_bin_compress_min_len option.

This page is licensed: CC BY-SA / Gnu FDL

Flashback

Flashback is a feature that allows instances, databases or tables to be rolled back to an old snapshot.

Flashback is supported only for DML statements (INSERT, DELETE, UPDATE). (An upcoming version of MariaDB will add support for flashback for DDL statements (DROP, TRUNCATE, ALTER, etc.) by copying or moving the current table to a reserved and hidden database, and then copying or moving back when using flashback. See MDEV-10571.)

Flashback is achieved in MariaDB Server using existing support for full image format binary logs (binlog_row_image=FULL), so it supports all engines.

The real work of Flashback is done by mariadb-binlog with `--flashback`. This causes events to be translated: `INSERT` to `DELETE`, `DELETE` to `INSERT`, and for `UPDATE` statements, the before and after images are swapped.

> **Note:**
>
> When executing `mariadb-binlog` with `--flashback`, the Flashback events are stored in memory. Make sure your server has enough memory for this feature.

Arguments

- mariadb-binlog has the `--flashback` or `-B` option that makes it work in flashback mode.
- mariadbd has the option –flashback that enables the binary log and sets `binlog_format=ROW`. It is not mandatory to use this option if you have already enabled those options directly.

Do not use the `-v` or `-vv` options, as they add verbose information to the binary log which can cause problems when importing. See MDEV-12066 and MDEV-12067.

Example

With a *`mytable`* table in a *`test`* database, you can compare the output with `--flashback` and without:

```
mariadb-binlog /var/lib/mysql/mysql-bin.000001 -vv -d test -T mytable \
    --start-datetime="2013-03-27 14:54:00" > review.sql
```

```
mariadb-binlog /var/lib/mysql/mysql-bin.000001 -vv -d test -T mytable \
    --start-datetime="2013-03-27 14:54:00" --flashback > flashback.sql
```

If you know the exact position, `--start-position` can be used instead of `--start-datetime`.

Then, by importing the output file (`mariadb < flashback.sql`), you can flash your database/table back to the specified time or position.

Common Use Case

A common use case for Flashback is the following scenario:

- You have one primary and two replicas, one started with `--flashback` (i.e., with binary logging enabled, using binlog_format=ROW, and binlog_row_image=FULL).
- Something goes wrong on the primary (like a wrong update or delete) and you would like to revert to a state of the database (or just a table) at a certain point in time.
- Remove the flashback-enabled replica from replication.
- Invoke mariadb-binlog to find the exact log position of the first offending operation after the state you want to revert to.
- Run `mariadb-binlog --flashback --start-position=xyz | mariadb` to pipe the output of `mariadb-binlog` directly to the `mariadb` client, or save the output to a file and then direct the file to the command-line client.

This page is licensed: CC BY-SA / Gnu FDL

Group Commit for the Binary Log

Overview

The server supports group commit. This is an important optimization that helps MariaDB reduce the number of expensive disk operations that are performed.

Durability

In ACID terminology, the “D” stands for durability. In order to ensure durability with group commit, innodb_flush_log_at_trx_commit=1 and/or sync_binlog=1 should be set. These settings are needed to ensure that if the server crashes, any transaction that was committed prior to the time of crash is still present in the database after crash recovery.

Durable InnoDB Data and Binary Logs

Setting both innodb_flush_log_at_trx_commit=1 and sync_binlog=1 provides the most durability and the best guarantee of replication consistency after a crash.

Non-Durable InnoDB Data

If sync_binlog=1 is set, but innodb_flush_log_at_trx_commit is not set to `1` or `3`, it is possible after a crash to end up in a state where a transaction present in a server binary log is missing from the server’s InnoDB redo log. If the server is a replication primary, this means that the server can become inconsistent with its replicas, since the replicas may have replicated transactions from the primary’s binary log that are no longer present in the primary’s local InnoDB data.

Non-Durable Binary Logs

If innodb_flush_log_at_trx_commit is set to `1` or `3`, but sync_binlog=1 is not set, it is possible after a crash to end up in a state where a transaction present in a server’s InnoDB redo log is missing from the server’s binary log. If the server is a replication primary, this also means that the server can become inconsistent with its replicas, since the server’s replicas would not be able to replicate the missing transactions from the server’s binary log.

Non-Durable InnoDB Data and Binary Logs

Setting innodb_flush_log_at_trx_commit=1 when sync_binlog=1 is not set can also cause the transaction to be missing from the server's InnoDB redo log due to some optimizations added in those versions. In that case, it is recommended to always set sync_binlog=1. If you can't do that, it is recommended to set innodb_flush_log_at_trx_commit to `3`, rather than `1`. See Non-durable Binary Log Settings for more information.

Amortizing Disk Flush Costs

> **Note:**
>
> From MariaDB 12.3, InnoDB-based binary logs can be used. (This is configurable, and not the default.)
>
> If configured, binary logs are written to InnoDB-managed, page-structured files (with the `.ibb` extension) that are integrated with InnoDB's redo log and crash recovery, rather than the traditional flat binary log files. This removes the need to protect the binary log separately, since it "inherits" the same crash safety as other InnoDB data, and it removes the need for expensive two-phase commit between InnoDB transactions and the binary log. Some options that apply to traditional binary log files behave differently or no longer apply – for example, `sync_binlog` is effectively ignored.
>
> InnoDB-based binary logs are enabled by setting `binlog_storage_engine=innodb` in the server configuration. See InnoDB-Based Binary Log for more information.

After every transaction COMMIT, the server normally has to persist any changes the transaction made to the InnoDB redo log and the binary log to non-volatile storage. This helps ensure that the data changes made by the transaction are durable. Disk flushing is a time-consuming operation, and can easily impose limits on throughput in terms of the number of transactions-per-second (TPS) which can be committed.

The idea with group commit is to amortize the costs of each flush to disk over multiple commits from multiple parallel transactions. For example, if there are 10 transactions trying to commit in parallel, then we can force all of them to be flushed disk at once with a single system call, rather than do one system call for each commit. This can greatly reduce the need for flush operations, and can consequently greatly improve the throughput of transactions-per-second (TPS).

However, to see the positive effects of group commit, the workload must have sufficient parallelism. A good rule of thumb is that at least three parallel transactions are needed for group commit to be effective. For example, while the first transaction is waiting for its flush operation to complete, the other two transactions will queue up waiting for their turn to flush their changes to disk. When the first transaction is done, a single system call can be used to flush the two queued-up transactions, saving in this case one of the three system calls.

In addition to sufficient parallelism, it is also necessary to have enough transactions per second wanting to commit that the flush operations are a bottleneck. If no such bottleneck exists (i.e., transactions never or rarely need to wait for the flush of another to complete), then group commit will provide little to no improvement.

Changing Group Commit Frequency

The frequency of group commits can be changed by configuring the binlog_commit_wait_usec and binlog_commit_wait_count system variables.

Measuring Group Commit Ratio

Two status variables are available for checking how effective group commit is at reducing flush overhead. These are the Binlog_commits and Binlog_group_commits status variables. We can obtain those values with the following query:

```
SHOW GLOBAL STATUS WHERE Variable_name IN('Binlog_commits', 'Binlog_group_commits');
```

Binlog_commits is the total number of transactions committed to the binary log.

Binlog_group_commits is the total number of groups committed to the binary log. As explained in the previous sections of this page, a group commit is when a group of transactions is flushed to the binary log together by sharing a single flush system call. When sync_binlog=1 is set, this is also the total number of flush system calls executed in the process of flushing commits to the binary log.

Thus, the extent to which group commit is effective at reducing the number of flush system calls on the binary log can be determined by the ratio between these two status variables. Binlog_commits are always equal to or greater than Binlog_group_commits. The greater the difference is between these status variables, the more effective group commit was at reducing flush overhead.

To calculate the group commit ratio, we need the values of these status variables from two snapshots. With those, we can calculate the ratio with the following formula:

`transactions/group commit` = (`Binlog_commits` (*snapshot2*) - `Binlog_commits` (*snapshot1*))/(`Binlog_group_commits` (*snapshot2*) - `Binlog_group_commits` (*snapshot1*))

For example, if we had the following first snapshot:

```
SHOW GLOBAL STATUS WHERE Variable_name IN('Binlog_commits', 'Binlog_group_commits');
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Binlog_commits       | 120   |
| Binlog_group_commits | 120   |
+----------------------+-------+
2 rows in set (0.00 sec)
```

And the following second snapshot:

```
SHOW GLOBAL STATUS WHERE Variable_name IN('Binlog_commits', 'Binlog_group_commits');
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Binlog_commits       | 220   |
| Binlog_group_commits | 145   |
+----------------------+-------+
2 rows in set (0.00 sec)
```

Then we would have:

`transactions/group commit` = (220 - 120) / (145 - 120) = 100 / 25 = 4 `transactions/group commit`

If your group commit ratio is too close to `1`, it may help to change your group commit frequency.

Use of Group Commit with Parallel Replication

Group commit is also used to enable conservative mode of in-order parallel replication.

Effects of Group Commit on InnoDB Performance

When both innodb_flush_log_at_trx_commit=1 (the default) is set and the binary log is enabled, there is now one less sync to disk inside InnoDB during commit (2 syncs shared between a group of transactions instead of 3). See Binary Log Group Commit and InnoDB Flushing Performance for more information.

Status Variables

Binlog_commits is the total number of transactions committed to the binary log.

Binlog_group_commits is the total number of groups committed to the binary log.

Binlog_group_commit_trigger_count is the total number of group commits triggered because of the number of binary log commits in the group reached the limit set by the system variable binlog_commit_wait_count.

Binlog_group_commit_trigger_lock_wait is the total number of group commits triggered because a binary log commit was being delayed because of a lock wait where the lock was held by a prior binary log commit. When this happens the later binary log commit is placed in the next group commit.

Binlog_group_commit_trigger_timeout is the total number of group commits triggered because of the time since the first binary log commit reached the limit set by the system variable binlog_commit_wait_usec.

Issue this statement to query these variables:

```
SHOW GLOBAL STATUS LIKE 'Binlog_%commit%';
```

See Also

- Parallel Replication
- Binary Log Group Commit and InnoDB Flushing Performance
- Group commit benchmark

This page is licensed: CC BY-SA / Gnu FDL

Relay Log

The relay log is a set of log files created by a replica during replication.

It's the same format as the binary log, containing a record of events that affect the data or structure; thus, mariadb-binlog can be used to display its contents. It consists of a set of relay log files and an index file containing a list of all relay log files.

Events are read from the primary's binary log and written to the replica's relay log. They are then applied on the replica. Old relay log files are automatically removed once they are no longer needed.

Creating Relay Log Files

New relay log files are created by the replica under the following circumstances:

- When the IO thread starts.
- When the logs are flushed, with FLUSH LOGS or mariadb-admin flush-logs.
- When the maximum size has been reached, determined by the max_relay_log_size system variable.

Relay Log Names

By default, the relay log will be given a name `host_name-relay-bin.nnnnnn`, with `host_name` referring to the server's host name, and #nnnnnn `the sequence number.`

This causes problems if the replica's host name changes, returning this error:

```
Failed to open the relay log and Could not find target log during relay log initialization
```

To prevent this, you can specify the relay log file name by setting the relay_log and relay_log_index system variables.

If you need to overcome this issue while replication is already underway, you can stop the replica, prepend the old relay log index file to the new relay log index file, and restart the replica.

For example:

```
shell> cat NEW_relay_log_name.index >> OLD_relay_log_name.index
shell> mv OLD_relay_log_name.index NEW_relay_log_name.index
```

Viewing Relay Logs

The SHOW RELAYLOG EVENTS shows events in the relay log, and, since relay log files are the same format as binary log files, they can be read with the mariadb-binlog utility.

Removing Old Relay Logs

Old relay logs are automatically removed once all events have been implemented on the replica, and the relay log file is no longer needed. This behavior can be changed by adjusting the relay_log_purge system variable from its default of `1` to `0`, in which case the relay logs will be left on the server.

Relay logs are also removed by the CHANGE MASTER statement unless a relay log option is used.

One can also flush the logs with the FLUSH RELAY LOGS statements.

If the relay logs are taking up too much space on the replica, the relay_log_space_limit system variable can be set to limit the size. The IO thread stops until the SQL thread has cleared the backlog. By default there is no limit.

See also

- FLUSH RELAY LOGS

This page is licensed: CC BY-SA / Gnu FDL

Selectively Skipping Replication of Binlog Events

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Normally, all changes that are logged as events in the binary log are also replicated to all replicas (though still subject to filtering by replicate-do-db, replicate-ignore-db, and similar options). However, sometimes it may be desirable to have certain events logged in the binlog, but not replicated to all or to some replicas, where the distinction between events that should be replicated or not is under the control of the application making the changes.

This can be useful if an application does some replication external to the server outside of the built-in replication, or if it has data that for some reason should not be replicated.

This is possible with the following system variables.

Primary Session Variable: skip_replication

When the skip_replication variable is set to true, changes are logged into the binary log with the flag `@@skip_replication` set. Such events are not replicated by replicas that run with `--replicate-events-marked-for-skip` set differently from the default (which is `REPLICATE`).

Variable Name	skip_replication
Scope	Session only
Access Type	Dynamic
Data Type	bool
Default Value	`OFF`

The `skip_replication` option only has effect if binary logging is enabled and sql_log_bin is true.

Attempting to change `@@skip_replication` in the middle of a transaction fails. This is to avoid getting half of a transaction replicated while the other half is not replicated. Be sure to end any current transaction with `COMMIT` / `ROLLBACK` before changing the variable.

Replica Option: –replicate-events-marked-for-skip

The replicate_events_marked_for_skip option tells the replica whether to replicate events that are marked with the `@@skip_replication` flag. Default is `REPLICATE`, to ensure that all changes are replicated to the replica. If set to `FILTER_ON_SLAVE`, events so marked will be skipped on the replica and not replicated. If set to `FILTER_ON_MASTER`, the filtering is done on the primary, saving network bandwidth as the events are not received by the replica at all.

Variable Name	replicate_events_marked_for_skip
Scope	Global
Access Type	Dynamic
Data Type	enum: `REPLICATE` | `FILTER_ON_SLAVE` | `FILTER_ON_MASTER`
Default Value	`REPLICATE`

> **Note:**
>
> `replicate_events_marked_for_skip` is a dynamic variable (it can be changed without restarting the server). However, the replica threads must be stopped when it is changed, otherwise an error is thrown.

When events are filtered due to `@@skip_replication`, the filtering happens on the primary side; in other words, the event is never sent to the replica. If many events are filtered like this, a replica can sit a long time without receiving any events from the primary. This is not a problem in itself, but must be kept in mind when inquiring on the replica about events that are filtered. For example, `START SLAVE UNTIL <some position>` stops when the first event that is **not** filtered is encountered at the given position or beyond. If the event at the given position is filtered, then the replica thread only stops when the next non-filtered event is encountered. In effect, if an event is filtered, to the replica it appears that it was never written to the binlog on the primary.

Note that when events are filtered for a replica, the data in the database is different on the replica and on the primary. It is the responsibility of the application to replicate the data outside of the built-in replication or otherwise ensure consistency of operation. If this is not done, it is possible for replication to encounter, for example, UNIQUE constraint violations or other problems which will cause replication to stop and require manual intervention to fix.

The session variable `@@skip_replication` can be changed without requiring special privileges. This makes it possible for normal applications to control it without requiring `SUPER` privileges. But it must be kept in mind when using replicas with `--replicate-events-marked-for-skip` set different from `REPLICATE`, as it allows any connection to do changes that are not replicated.

skip_replication and sql_log_bin

@@sql_log_bin and `@@skip_replication` are somewhat related, as they can both be used to prevent a change on the primary from being replicated to the replica. The difference is that with `@@skip_replication`, changes are still written into the binlog, and replication of the events is only skipped on replicas that explicitly are configured to do so, with `--replicate-events-marked-for-skip` different from `REPLICATE`. With `@@sql_log_bin`, events are not logged into the binlog, and so are not replicated by any replica.

skip_replication and the Binlog

When events in the binlog are marked with the `@@skip_replication` flag, the flag will be preserved if the events are dumped by the mariadb-binlog program and re-applied against a server with the mariadb client program. Similarly, the BINLOG statement will preserve the flag from the event being replayed. And a replica which runs

with `--log-slave-updates` and does not filter events (`--replicate-events-marked-for-skip=REPLICATE`) also preserves the flag in the events logged into the binlog on the replica.

See Also

- Using SQL_SLAVE_SKIP_COUNTER - How to skip a number of events on the replica

This page is licensed: CC BY-SA / Gnu FDL

Error Log

The error log contains a record of critical errors that occurred during the server's operation, table corruption, start and stop information.

SQL errors can also be logged in a separate file using the SQL_ERROR_LOG plugin.

Configuring the Error Log Output Destination

MariaDB always writes its error log, but the destination is configurable.

Writing the Error Log to a File

To configure the error log to be written to a file, you can set the log_error system variable. You can configure a specific file name. However, if a specific file name is not configured, the log will be written to the `${hostname}.err` file in the datadir directory by default.

The log_error system variable can be set in a server option group in an option file prior to starting up the server. For example, to write the error log to the default `${hostname}.err` file, you could configure the following:

```
[mariadb]
...
log_error
```

If you configure a specific file name as the log_error system variable, and if it is not an absolute path, then it will be relative to the datadir directory. For example, if you configured the following, then the error log would be written to `mariadb.err` in the datadir directory:

```
[mariadb]
...
log_error=mariadb.err
```

If it is a relative path, then the log_error is relative to the datadir directory.

However, the log_error system variable can also be an absolute path:

```
[mariadb]
...
log_error=/var/log/mysql/mariadb.err
```

Another way to configure the error log file name is to set the log-basename option, which configures MariaDB to use a common prefix for all log files (e.g. general query log, slow query log, error log, binary logs, etc.). The error log file name will be built by adding a `.err` extension to this prefix. For example, if you configured the following, then the error log would still be written to `mariadb.err` in the datadir directory:

```
[mariadb]
...
log-basename=mariadb
log_error
```

The log-basename cannot be an absolute path. The log file name is relative to the datadir directory.

Writing the Error Log to Stderr on Unix

On Unix, if the log_error system variable is not set, errors are written to `stderr`, which usually means that the log messages are output to the terminal that started `mariadbd`.

If the log_error system variable was set in an option file or on the command-line, it can still be unset by specifying `--skip-log-error`.

Writing the Error Log to Syslog on Unix

On Unix, the error log can also be redirected to the syslog. How this is done depends on how you start MariaDB.

Syslog with mariadbd-safe

If you start MariaDB with mariadbd-safe, the error log can be redirected to the syslog. See mariadbd-safe: Configuring MariaDB to Write the Error Log to Syslog for more information.

Syslog with Systemd

If you start MariaDB with systemd, the error log can also be redirected to the syslog. See Systemd: Configuring MariaDB to Write the Error Log to Syslog for more information.

systemd also has its own logging system called the `journal`, and some errors may get logged there instead. See Systemd:Systemd Journal for more information.

Writing the Error Log to Console on Windows

On Windows, if the console option is specified, and if the log_error system variable is not used, errors are written to the console. If both options are specified, the last option takes precedence.

Writing the Error Log to the Windows Event Viewer

On Windows, error log messages are also written to the Windows Event Viewer. You can find MariaDB's error log messages by browsing **Windows Logs**, then selecting **Application** or **Application Log**, depending on the Windows version.

You can find MariaDB's error log messages by searching for the **Source** `MariaDB` (prior to MariaDB 10.4, this was `MySQL`).

Finding the Error Log

To find where the error log is stored, one can find the options used for the error log with:

```
mariadbd --print-defaults
```

or (from MariaDB 10.11)

```
my_print_defaults --mariadbd | grep log-error
```

or

```
my_print_defaults --mysqld | grep log-error
```

If the above don't help, check also if your system is set to write to syslog, in which case you need to use journalctl to access it.

Configuring the Error Log Verbosity

The default value of the log_warnings system variable is `2`.

The log_warnings system variable can be used to configure the verbosity of the error log. It can be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL log_warnings=3;
```

It can also be set either on the command-line or in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
log_warnings=3
```

Some of the warnings included in each verbosity level are described below.

The log_warnings system variable only has an effect on some log messages. Some log messages are **always** written to the error log, regardless of the error log verbosity. For example, most warnings from the InnoDB storage engine are not affected by log_warnings. For a complete list of log messages affected by log_warnings, see the description of the log_warnings system variable.

Verbosity Level 0

If log_warnings is `0`, many optional warnings will not be logged. However, this does not prevent all warnings from being logged, because there are certain core warnings that will always be written to the error log. For example:

- If InnoDB strict mode is disabled, and if DDL is performed on a table that triggers a "Row size too large" error, InnoDB logs a warning:

```
[Warning] InnoDB: Cannot add field col25 in table db1.tab because after
  adding it, the row size is 8477 which is greater than maximum allowed
  size (8126) for a record on index leaf page.
```

However, if InnoDB strict mode is enabled, the same message will be logged as an error.

Verbosity Level 1

Default until MariaDB 10.2.3. If log_warnings is `1`, many types of warnings are logged. Some useful warnings are:

- Replication-related messages:

```
[Warning] A server restart has recreated the memory table `db`.`tbl`; a TRUNCATE query is written to the
binary log at GTID 0-1-2. As this server is a read-only slave, this event may diverge replication in
domain 0.
[Note] Error reading relay log event: slave SQL thread was killed
[Note] Slave SQL thread exiting, replication stopped in log
  'dbserver-2-bin.000033'   at position 181420;
  GTID position '0-263316466-368886'
[Note] Slave I/O thread exiting, read up to log
  'dbserver-2-bin.000034', position 642;
  GTID position 0-263316466-368887
```

- Messages related to DNS lookup failures:

```
[Warning] IP address '192.168.1.193'
  could not be resolved: Name or service not known
```

- Messages related to the event scheduler:

```
[Note] Event Scheduler: Loaded 0 events
```

- Messages related to unsafe statements for statement-based replication:

```
[Warning] Unsafe statement written to the binary log using statement format since
  BINLOG_FORMAT = STATEMENT. The statement is unsafe because
  it uses a LIMIT clause. This
  is unsafe because the set of rows included cannot be predicted.
```

Frequent warnings about unsafe statements for statement-based replication can cause the error log to grow very large. MariaDB will automatically detect frequent duplicate warnings about unsafe statements for statement-based replication. After 10 identical warnings are detected, MariaDB will prevent that same warning from being written to the error log again for the next 5 minutes.

Verbosity Level 2

Default from MariaDB 10.2.4. If log_warnings is `2`, a couple other different kinds of warnings are printed. For example:

- Messages related to access denied errors:

```
[Warning] Access denied for user 'root'@'localhost' (using password: YES)
```

- Messages related to connections that are aborted due to errors or timeouts:

```
[Warning] Aborted connection 35 to db: 'unconnected' user:
  'user1@host1' host: '192.168.1.40' (Got an error writing communication packets)
[Warning] Aborted connection 36 to db: 'unconnected' user:
  'user1@host2' host: '192.168.1.230' (Got an error writing communication packets)
[Warning] Aborted connection 38 to db: 'db1' user:
  'user2' host: '192.168.1.60' (Unknown error)
[Warning] Aborted connection 51 to db: 'db1' user:
  'user2' host: '192.168.1.50' (Got an error reading communication packets)
[Warning] Aborted connection 52 to db: 'db1' user:
  'user3' host: '192.168.1.53' (Got timeout reading communication packets)
```

- Messages related to table handler errors:

```
[Warning] Can't find record in 'tab1'.
[Warning] Can't write; duplicate key in table 'tab1'.
[Warning] Lock wait timeout exceeded; try restarting transaction.
[Warning] The number of locks exceeds the lock table size.
[Warning] Update locks cannot be acquired during a READ UNCOMMITTED transaction.
```

- Messages related to the files used to persist replication state:
  - Either the default `master.info` file or the file that is configured by the master_info_file option.
  - Either the default `relay-log.info` file or the file that is configured by the relay_log_info_file system variable.

```
[Note] Reading Master_info: '/mariadb/data/master.info'
  Relay_info:'/mariadb/data/relay-log.info'
[Note] Initialized Master_info from '/mariadb/data/master.info'
[Note] Reading of all Master_info entries succeeded
[Note] Deleted Master_info file '/mariadb/data/master.info'.
[Note] Deleted Master_info file '/mariadb/data/relay-log.info'.
```

- Messages about a master's binary log dump thread:

```
[Note] Start binlog_dump to slave_server(263316466), pos(, 4)
```

Verbosity Level 3

If log_warnings is `3`, a couple other different kinds of warnings are printed. For example:

- Messages related to old-style language options:

```
[Warning] An old style --language value with language specific
  part detected: /usr/local/mysql/data/
[Warning] Use --lc-messages-dir without language specific part instead.
```

- Messages related to progress of InnoDB online DDL:

```
[Note] InnoDB: Online DDL : Start
[Note] InnoDB: Online DDL : Start reading clustered index of the table and
  create temporary files
[Note] InnoDB: Online DDL : End of reading clustered index of the table and
  create temporary files
[Note] InnoDB: Online DDL : Start merge-sorting index PRIMARY (1 / 3),
  estimated cost : 18.0263
[Note] InnoDB: Online DDL : merge-sorting has estimated 33 runs
[Note] InnoDB: Online DDL : merge-sorting current run 1 estimated 33 runs
[Note] InnoDB: Online DDL : merge-sorting current run 2 estimated 17 runs
[Note] InnoDB: Online DDL : merge-sorting current run 3 estimated 9 runs
[Note] InnoDB: Online DDL : merge-sorting current run 4 estimated 5 runs
[Note] InnoDB: Online DDL : merge-sorting current run 5 estimated 3 runs
[Note] InnoDB: Online DDL : merge-sorting current run 6 estimated 2 runs
[Note] InnoDB: Online DDL : End of  merge-sorting index PRIMARY (1 / 3)
[Note] InnoDB: Online DDL : Start building index PRIMARY (1 / 3),
  estimated cost : 27.0395
[Note] InnoDB: Online DDL : End of building index PRIMARY (1 / 3)
[Note] InnoDB: Online DDL : Completed
[Note] InnoDB: Online DDL : Start merge-sorting index ux1 (2 / 3),
  estimated cost : 5.7895
[Note] InnoDB: Online DDL : merge-sorting has estimated 2 runs
[Note] InnoDB: Online DDL : merge-sorting current run 1 estimated 2 runs
[Note] InnoDB: Online DDL : End of  merge-sorting index ux1 (2 / 3)
[Note] InnoDB: Online DDL : Start building index ux1 (2 / 3),
  estimated cost : 8.6842
[Note] InnoDB: Online DDL : End of building index ux1 (2 / 3)
[Note] InnoDB: Online DDL : Completed
[Note] InnoDB: Online DDL : Start merge-sorting index ix1 (3 / 3),
  estimated cost : 6.1842
[Note] InnoDB: Online DDL : merge-sorting has estimated 3 runs
[Note] InnoDB: Online DDL : merge-sorting current run 1 estimated 3 runs
[Note] InnoDB: Online DDL : merge-sorting current run 2 estimated 2 runs
[Note] InnoDB: Online DDL : End of  merge-sorting index ix1 (3 / 3)
[Note] InnoDB: Online DDL : Start building index ix1 (3 / 3),
  estimated cost : 9.2763
[Note] InnoDB: Online DDL : End of building index ix1 (3 / 3)
[Note] InnoDB: Online DDL : Completed
```

Verbosity Level 4

If log_warnings is `4`, a couple other different kinds of warnings are printed. For example:

- Messages related to killed connections:

```
[Warning] Aborted connection 53 to db: 'db1' user:
  'user2' host: '192.168.1.50' (KILLED)
```

- Messages related to all closed connections:

```
[Warning] Aborted connection 56 to db: 'db1' user:
  'user2' host: '192.168.1.50' (CLOSE_CONNECTION)
```

- Messages related to released connections, such as when a transaction is committed and completion_type is set to `RELEASE`:

```
[Warning] Aborted connection 58 to db: 'db1' user:
  'user2' host: '192.168.1.50' (RELEASE)
```

Verbosity Level 9

If log_warnings is `9`, some **very** verbose warnings are printed. For example:

- Messages about initializing plugins:

```
[Note] Initializing built-in plugins
[Note] Initializing plugins specified on the command line
[Note] Initializing installed plugins
```

MySQL's log_error_verbosity

MariaDB does not support the log_error_verbosity system variable added in MySQL 5.7.

Format

The error log begins with a line indicating where the data is being read from, for example:

```
240919 22:58:50 mysqld_safe Starting mariadbd daemon with databases from
/home/ian/sandboxes/msb_11_7_0/data
```

Then, each item (note, warning or error) consists of a single line, containing the date (yyyy-mm-dd) and time, the thread ID, followed by the type of error (Note, Warning or Error) and the error message, for example:

```
2016-06-15 16:53:33 139651251140544 [Note] InnoDB: The InnoDB memory heap is disabled
```

Until MariaDB 10.1.4, the format only consisted of the date (yymmdd) and time, followed by the type of error (Note, Warning or Error) and the error message, for example:

```
160615 16:53:08 [Note] InnoDB: The InnoDB memory heap is disabled
```

The first item will always contain the source revision, a unique server id (from MariaDB 10.5.26, MariaDB 10.6.19, MariaDB 10.11.9, MariaDB 11.1.6, MariaDB 11.2.5, MariaDB 11.4.3, MariaDB 11.5.2, MariaDB 11.6.1) and the process_id, for example:

```
2024-09-19 22:58:50 0 [Note] Starting MariaDB 11.7.0-preview-MariaDB source revision
  7391f7143b462b29ecdaee049c5ebdfd1aefa6d7 server_uid UdJQ5BCzAQiTopMoCz3yYsfU1lA=
  as process 2307166
```

or

```
2024-05-18 16:05:33 0 [Note] Starting MariaDB 10.11.8-MariaDB source revision
  3a069644682e336e445039e48baae9693f9a08ee as process 50774
```

Formal Specification

For modern MariaDB versions (10.1.5 and later), the standard file-based error log follows this strict anatomy:

Template: `YYYY-MM-DD HH:MM:SS ThreadID [Level] Message`

Field	Component	Data Type	Description
1	Timestamp	`DateTime`	Formatted as `YYYY-MM-DD HH:MM:SS`.

Field	Component	Data Type	Description
2	Thread ID	`Unsigned Integer`	The internal connection ID. System messages use `0`.
3	Level	`String`	Enclosed in brackets. Valid values: `[Note]`, `[Warning]`, `[Error]`.
4	Message	`String`	The actual descriptive text of the event.

Deviations and Anomalies

Parsers should account for the following scenarios where the log structure may deviate:

- **Galera SST Logs:** When a Galera Cluster node performs a State Snapshot Transfer (SST), messages from external scripts (like `rsync` or `mariabackup`) are piped directly into the error log. These lines often lack the `ThreadID` and `[Level]` prefix.
- **Crash Stack Traces:** Following a server crash, MariaDB writes a multi-line stack trace. This consists of a header line followed by multiple lines of hex offsets and function names (e.g., `handle_fatal_signal (sig=11)`). These do not follow the standard single-line format.
- **Syslog Format:** If logging to syslog, the standard MariaDB prefix is replaced or preceded by the system's own header: `Month Day HH:MM:SS Hostname Ident[PID]: [Level] Message`

Rotating the Error Log on Unix and Linux

Unix and Linux distributions offer the logrotate utility, which makes it very easy to rotate log files. See Rotating Logs on Unix and Linux for more information on how to use this utility to rotate the error log.

Error Messages File

Many error messages are read from an error message file that contains localized error messages. If the server can't find this file when it starts up, you might see errors like the following:

```
[ERROR] Can't find messagefile '/usr/share/errmsg.sys'
```

If this error is occurring because the file is in a custom location, you can configure this location by setting the lc_messages_dir system variable either on the command-line or in a server option group in an option file prior to starting up the server:

```
[mariadb]
...
lc_messages_dir=/usr/share/mysql/
```

If you want to use a different locale for error messages, you can also set the lc_messages system variable:

```
[mariadb]
...
lc_messages_dir=/usr/share/mysql/
lc_messages=en_US
```

See Setting the Language for Error Messages for more information.

See Also

- sql error log plugin

This page is licensed: CC BY-SA / Gnu FDL

General Query Log

The general query log is a log of every SQL query received from a client, as well as each client connect and disconnect. Since it's a record of every query received by the server, it can grow large quite quickly.

However, if you only want a record of queries that change data, it might be better to use the binary log instead. One important difference is that the binary log only logs a query when the transaction is committed by the server, but the general query log logs a query immediately when it is received by the server.

Enabling the General Query Log

The general query log is disabled by default.

To enable the general query log, set the general_log system variable to `1`. It can be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL general_log=1;
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
general_log
```

Configuring the General Query Log Filename

By default, the general query log is written to `${hostname}.log` in the datadir directory. However, this can be changed.

One way to configure the general query log filename is to set the general_log_file system variable. It can be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL general_log_file='mariadb.log';
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
general_log
general_log_file=mariadb.log
```

If it is a relative path, then the general_log_file is relative to the datadir directory.

However, the general_log_file system variable can also be an absolute path. For example:

```
[mariadb]
...
general_log
general_log_file=/var/log/mysql/mariadb.log
```

Another way to configure the general query log filename is to set the log-basename option, which configures MariaDB to use a common prefix for all log files (e.g. general query log, slow query log, error log, binary logs, etc.). The general query log filename will be built by adding a `.log` extension to this prefix. This option cannot be set dynamically. It can be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
log-basename=mariadb
general_log
```

The log-basename cannot be an absolute path. The log file name is relative to the datadir directory.

Choosing the General Query Log Output Destination

The general query log can either be written to a file on disk, or it can be written to the general_log table in the mysql database. To choose the general query log output destination, set the log_output system variable.

Writing the General Query Log to a File

The general query log is output to a file by default. However, it can be explicitly chosen by setting the log_output system variable to `FILE`. It can be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL log_output='FILE';
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
log_output=FILE
general_log
general_log_file=queries.log
```

Writing the General Query Log to a Table

The general query log can either be written to the general_log table in the mysql database by setting the log_output system variable to `TABLE`. It can be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL log_output='TABLE';
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
log_output=TABLE
general_log
```

Some rows in this table might look like this:

```
SELECT * FROM mysql.general_log\G
*************************** 1. row ***************************
  event_time: 2014-11-11 08:40:04.117177
   user_host: root[root] @ localhost []
   thread_id: 74
   server_id: 1
command_type: Query
    argument: SELECT * FROM test.s
*************************** 2. row ***************************
  event_time: 2014-11-11 08:40:10.501131
   user_host: root[root] @ localhost []
   thread_id: 74
   server_id: 1
command_type: Query
    argument: SELECT * FROM mysql.general_log
...
```

See Writing logs into tables for more information.

Formal Specification (File Format)

When the General Query Log is written to a file, it follows a positional format. For tool developers (e.g., Fluentd, Logstash), it is important to treat this log as a continuous stream of events that can contain multi-line data.

Template: `Timestamp ThreadID Command Argument`

Field	Component	Data Type	Description
1	Timestamp	`DateTime`	The time the query was received. Format varies by version (e.g., `yymmdd h:mm:ss` or ISO 8601).
2	Thread ID	`Unsigned Integer`	The standardized connection identifier. This matches the `Thread ID` field in the Error Log and Slow Query Log.
3	Command	`String`	The command type (e.g., `Connect`, `Query`, `Quit`, `Prepare`).
4	Argument	`String`	The raw SQL statement or command details.

Parsing Considerations for Tool Developers

- **Multi-line SQL:** The `Argument` field contains the literal SQL query received by the server. Because SQL queries often contain newlines, a single log entry can span multiple lines. Parsers must be configured to handle multiline events, typically by identifying a new record when a line begins with a valid timestamp.
- **Standardized Identification:** While the log file header may label the second column as `Id`, it is functionally identical to the `thread_id` found in system tables (e.g., `mysql.general_log`) and the `Thread ID` in other server logs.
- **Immediate Logging:** Unlike the Binary Log, which logs at commit time, the General Query Log records queries immediately upon receipt.

Disabling the General Query Log for a Session

A user with the SUPER privilege can disable logging to the general query log for a connection by setting the SQL_LOG_OFF system variable to `1`. For example:

```
SET SESSION SQL_LOG_OFF=1;
```

Disabling the General Query Log for Specific Statements

In MariaDB 10.3.1 and later, it is possible to disable logging to the general query log for specific types of statements by setting the log_disabled_statements system variable. This option cannot be set dynamically. It can be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
log_output=FILE
general_log
general_log_file=queries.log
log_disabled_statements='slave,sp'
```

Rotating the General Query Log on Unix and Linux

Unix and Linux distributions offer the logrotate utility, which makes it very easy to rotate log files. See Rotating Logs on Unix and Linux for more information on how to use this utility to rotate the general query log.

See Also

- MariaDB audit plugin

This page is licensed: CC BY-SA / Gnu FDL

MyISAM Log

The MyISAM log records all changes to MyISAM tables. It is not enabled by default. To enable it, start the server with the –log-isam option, for example:

```
--log-isam=myisam.log
```

The *isam* instead of *myisam* above is not a typo - it's a legacy from when the predecessor to the MyISAM format, called ISAM. The option can be used without specifying a filename, in which case the default, *myisam.log* is used.

To process the contents of the log file, use the myisamlog utility.

This page is licensed: CC BY-SA / Gnu FDL

Rotating Logs on Unix and Linux

Unix and Linux distributions offer the logrotate utility, which makes it very easy to rotate log files. This page will describe how to configure log rotation for the error log, general query log, and the slow query log.

Configuring Locations and File Names of Logs

The first step is to configure the locations and file names of logs. To make the log rotation configuration easier, it can be best to put these logs in a dedicated log directory.

We will need to configure the following:

- The error log location and file name is configured with the log_error system variable.
- The general query log location and file name is configured with the general_log_file system variable.
- The slow query log location and file name is configured with the slow_query_log_file system variable.

If you want to enable the general query log and slow query log immediately, then you will also have to configure the following:

- The general query log is enabled with the general_log system variable.
- The slow query log is enabled with the slow_query_log system variable.

These options can be set in a server option group in an option file prior to starting up the server. For example, if we wanted to put our log files in `/var/log/mysql/`, then we could configure the following:

```
[mariadb]
...
log_error=/var/log/mysql/mariadb.err
general_log
general_log_file=/var/log/mysql/mariadb.log
slow_query_log
slow_query_log_file=/var/log/mysql/mariadb-slow.log
long_query_time=5
```

We will also need to create the relevant directory:

```
sudo mkdir /var/log/mysql/
sudo chown mysql:mysql /var/log/mysql/
sudo chmod 0770 /var/log/mysql/
```

If you are using SELinux, then you may also need to set the SELinux context for the directory. See SELinux: Setting the File Context for Log Files for more information. For example:

```
sudo semanage fcontext -a -t mysqld_log_t "/var/log/mysql(/.*)?"
sudo restorecon -Rv /var/log/mysql
```

After MariaDB is restarted, it will use the new log locations and file names.

Configuring Authentication for Logrotate

The logrotate utility needs to be able to authenticate with MariaDB in order to flush the log files.

The easiest way to allow the logrotate utility to authenticate with MariaDB is to configure the `root@localhost` user account.

Note

The `root@localhost` user account is configured to use unix_socket authentication by default

The `root@localhost` user account can be altered to use unix_socket authentication with the ALTER USER statement. For example:

```
ALTER USER 'root'@'localhost' IDENTIFIED VIA unix_socket;
```

<>

Configuring Logrotate

At this point, we can configure the logrotate utility to rotate the log files.

On many systems, the primary logrotate configuration file is located at the following path:

- `/etc/logrotate.conf`

And the logrotate configuration files for individual services are located in the following directory:

- `/etc/logrotate.d/`

We can create a logrotate configuration file for MariaDB by executing the following command in a shell:

```
$ sudo tee /etc/logrotate.d/mariadb <<EOF
/var/log/mysql/* {
        su mysql mysql
        missingok
        create 660 mysql mysql
        notifempty
        daily
        minsize 1M # only use with logrotate >= 3.7.4
        maxsize 100M # only use with logrotate >= 3.8.1
        rotate 30
        # dateext # only use if your logrotate version is compatible with below dateformat
        # dateformat .%Y-%m-%d-%H-%M-%S # only use with logrotate >= 3.9.2
        compress
        delaycompress
        sharedscripts
        olddir archive/
        createolddir 770 mysql mysql # only use with logrotate >= 3.8.9
    postrotate
        # just if mysqld is really running
        if test -x /usr/bin/mysqladmin && \
           /usr/bin/mysqladmin ping &>/dev/null
        then
           /usr/bin/mysqladmin --local flush-error-log \
              flush-engine-log flush-general-log flush-slow-log
        fi
    endscript
}
EOF
```

You may have to modify this configuration file to use it on your system, depending on the specific version of the logrotate utility that is installed. See the description of each configuration directive below to determine which logrotate versions support that configuration directive.

Each specific configuration directive does the following:

- `missingok`: This directive configures it to ignore missing files, rather than failing with an error.

- `create 660 mysql mysql`: This directive configures it to recreate the log files after log rotation with the specified permissions and owner.
- `notifempty`: This directive configures it to skip a log file during log rotation if it is empty.
- `daily`: This directive configures it to rotate each log file once per day.
- `minsize 1M`: This directive configures it to skip a log file during log rotation if it is smaller than 1 MB. This directive is only available with logrotate 3.7.4 and later.
- `maxsize 100M`: This directive configures it to rotate a log file more frequently than daily if it grows larger than 100 MB. This directive is only available with logrotate 3.8.1 and later.
- `rotate 30`: This directive configures it to keep 30 old copies of each log file.
- `dateext`: This directive configures it to use the date as an extension, rather than just a number. This directive is only available with logrotate 3.7.6 and later.
- `dateformat .%Y-%m-%d-%H-%M-%S`: This directive configures it to use this date format string (as defined by the format specification for strftime) for the date extension configured by the `dateext` directive. This directive is only available with logrotate 3.7.7 and later. Support for `%H` is only available with logrotate 3.9.0 and later. Support for `%M` and `%S` is only available with logrotate 3.9.2 and later.
- `compress`: This directive configures it to compress the log files with gzip.
- `delaycompress`: This directive configures it to delay compression of each log file until the next log rotation. If the log file is compressed at the same time that it is rotated, then there may be cases where a log file is being compressed while the MariaDB server is still writing to the log file. Delaying compression of a log file until the next log rotation can prevent race conditions such as these that can happen between the compression operation and the MariaDB server's log flush operation.
- `olddir archive/`: This directive configures it to archive the rotated log files in `/var/log/mysql/archive/`.
- `createolddir 770 mysql mysql`: This directive configures it to create the directory specified by the `olddir` directive with the specified permissions and owner, if the directory does not already exist. This directive is only available with logrotate 3.8.9 and later.
- `sharedscripts`: This directive configures it to run the `postrotate` script just once, rather than once for each rotated log file.
- `postrotate`: This directive configures it to execute a script after log rotation. This particular script executes the mariadb-admin utility, which executes the FLUSH statement, which tells the MariaDB server to flush its various log files. When MariaDB server flushes a log file, it closes its existing file handle and reopens a new one. This ensure that MariaDB server does not continue writing to a log file after it has been rotated. This is an important component of the log rotation process.

If our system does not have logrotate 3.8.9 or later, which is needed to support the `createolddir` directive, then we will also need to create the relevant directory specified by the `olddir` directive:

```
sudo mkdir /var/log/mysql/archive/
sudo chown mysql:mysql /var/log/mysql/archive/
sudo chmod 0770 /var/log/mysql/archive/
```

Testing Log Rotation

We can test log rotation by executing the logrotate utility with the `--force` option. For example:

```
sudo logrotate --force /etc/logrotate.d/mariadb
```

Keep in mind that under normal operation, the logrotate utility may skip a log file during log rotation if the utility does not believe that the log file needs to be rotated yet. For example:

- If you set the `notifempty` directive mentioned above, then it will be configured to skip a log file during log rotation if the log file is empty.

- If you set the `daily` directive mentioned above, then it will be configured to only rotate each log file once per day.
- If you set the `minsize 1M` directive mentioned above, then it will be configured to skip a log file during log rotation if the log file size is smaller than 1 MB.

However, when running tests with the `--force` option, the logrotate utility does not take these options into consideration.

After a few tests, we can see that the log rotation is indeed working:

```
$ sudo ls -l /var/log/mysql/archive/
total 48
-rw-rw---- 1 mysql mysql  440 Mar 31 15:31 mariadb.err.1
-rw-rw---- 1 mysql mysql  138 Mar 31 15:30 mariadb.err.2.gz
-rw-rw---- 1 mysql mysql  145 Mar 31 15:28 mariadb.err.3.gz
-rw-rw---- 1 mysql mysql 1007 Mar 31 15:27 mariadb.err.4.gz
-rw-rw---- 1 mysql mysql 1437 Mar 31 15:32 mariadb.log.1
-rw-rw---- 1 mysql mysql  429 Mar 31 15:31 mariadb.log.2.gz
-rw-rw---- 1 mysql mysql  439 Mar 31 15:28 mariadb.log.3.gz
-rw-rw---- 1 mysql mysql  370 Mar 31 15:27 mariadb.log.4.gz
-rw-rw---- 1 mysql mysql 3915 Mar 31 15:32 mariadb-slow.log.1
-rw-rw---- 1 mysql mysql  554 Mar 31 15:31 mariadb-slow.log.2.gz
-rw-rw---- 1 mysql mysql  569 Mar 31 15:28 mariadb-slow.log.3.gz
-rw-rw---- 1 mysql mysql  487 Mar 31 15:27 mariadb-slow.log.4.gz
```

Logrotate in Ansible

Let's see an example of how to configure logrotate in Ansible.

First, we'll create a couple of tasks in our playbook:

```
- name: Create mariadb_logrotate_old_dir
  file:
    path: "{{ mariadb_logrotate_old_dir }}"
    owner: mysql
    group: mysql
    mode: '770'
    state: directory

- name: Configure logrotate
  template:
    src: "../templates/logrotate.j2"
    dest: "/etc/logrotate.d/mysql"
```

The first task creates a directory to store the old, compressed logs, and set proper permissions.

The second task uploads logrotate configuration file into the proper directory, and calls it `mysql`. As you can see the original name is different, and it ends with the `.j2` extension, because it is a Jinja 2 template.

The file will look like the following:

```
{{ mariadb_log_dir }}/* {
        su mysql mysql
        missingok
        create 660 mysql mysql
        notifempty
        daily
        minsize 1M {{ mariadb_logrotate_min_size }}
        maxsize 100M {{ mariadb_logrotate_max_size }}
        rotate {{ mariadb_logrotate_old_dir }}
        dateformat .%Y-%m-%d-%H-%M-%S # only use with logrotate >= 3.9.2
        compress
        delaycompress
        sharedscripts
        olddir archive/
        createolddir 770 mysql mysql # only use with logrotate >= 3.8.9
    postrotate
        # just if mysqld is really running
        if test -x /usr/bin/mysqladmin && \
           /usr/bin/mysqladmin ping &>/dev/null
        then
           /usr/bin/mysqladmin --local flush-error-log \
              flush-engine-log flush-general-log flush-slow-log
        fi
    endscript
}
```

The file is very similar to the file shown above, which is obvious because we're still uploading a logrotate configuration file. Ansible is just a tool we've chosen to do this.

However, both in the tasks and in the template, we used some variables. This allows to use different paths and rotation parameters for different hosts, or host groups.

If we have a group host called `mariadb` that contains the default configuration for all our MariaDB servers, we can define these variables in a file called `group_vars/mariadb.yml`:

```
# MariaDB writes its logs here
mariadb_log_dir: /var/lib/mysql/logs

# logrotate configuration

mariadb_logrotate_min_size: 500M
mariadb_logrotate_max_size: 1G
mariadb_logrotate_old_files: 7
mariadb_logrotate_old_dir: /var/mysql/old-logs
```

After setting up logrotate in Ansible, you may want to deploy it to a non-production server and test it manually as explained above. Once you're sure that it works fine on one server, you can be confident in the new Ansible tasks and deploy them on all servers.

For more information on how to use Ansible to automate MariaDB configuration, see Ansible and MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

Slow Query Log

See Slow Query Log Overview.

Complete Slow Query Log Overview guide for MariaDB. Complete reference documentation for implementation, configuration, and usage for production use.

See EXPLAIN in the Slow Query Log.

Describes how to configure MariaDB to automatically write the `EXPLAIN` plan for slow queries to the log using the `log_slow_verbosity` system variable.

See log_slow_always_query_time System Variable.

Documentation for the `log_slow_always_query_time` variable, which forces queries executed by a specific function or user to be logged regardless of their execution time.

This page is licensed: CC BY-SA / Gnu FDL

Slow Query Log Overview

The slow query log is a record of SQL queries that took a long time to perform. The log also provides the number of rows affected by slow queries.

> **Note:**
>
> If your queries contain passwords, the slow query log can contain sensitive information, too. Ensure to store the log files in a safe location.
>
> While MariaDB natively encrypts Binary Logs and Data files, it does not provide built-in encryption for the Slow Query or General Query logs. To secure these, administrators should either log to Internal Tables (combined with Data-at-Rest encryption) or utilize OS-level filesystem encryption.

Enabling the Slow Query Log

The slow query log is disabled by default. To enable it, set log_slow_query to `1`, using one of the following methods.

This turns on slow query logging immediately, but only until the next server restart:

```
SET GLOBAL slow_query_log=1;
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
slow_query_log
```

To verify that slow query logging is enabled, and to see which log file is used, issue this statement:

```
SHOW GLOBAL VARIABLES LIKE 'slow_query_log%';
+---------------------+----------------------------------------------+
| Variable_name       | Value                                        |
+---------------------+----------------------------------------------+
| slow_query_log      | ON                                           |
| slow_query_log_file | c525d37c-b2ff-4543-b06f-87012d142d44-slow.log |
+---------------------+----------------------------------------------+
```

Configuring the Filename

By default, the slow query log is written to `${hostname}-slow.log` (for instance, `c525d37c-b2ff-4543-b06f-87012d142d44-slow.log`), and stored in the datadir directory. However, this can be changed.

Configure the slow query log filename by setting the slow_query_log_file system variable (or, from MariaDB 10.11, log_slow_query_file). It can be changed dynamically with SET GLOBAL:

```
SET GLOBAL slow_query_log_file='mariadb-slow.log';
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
slow_query_log
slow_query_log_file=mariadb-slow.log
```

Setting a relative path, or just the filename, puts the log file in the datadir directory. You can put it somewhere else by using an absolute path:

```
[mariadb]
slow_query_log
slow_query_log_file=/var/log/mysql/mariadb-slow.log
```

Another way to configure the slow query log filename is to set the log-basename option, which configures MariaDB to use a common prefix for all log files (e.g. slow query log, general query log, error log, binary logs, etc.). The slow query log filename will be built by adding `-slow.log` to this prefix. This option cannot be set dynamically. It can be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
log-basename=mariadb
slow_query_log
```

> **Note:**
>
> If you configure all log file basenames using –log-basename, you cannot use an absolute path – the log file name is always relative to the datadir directory.

Choosing the Output Destination

The slow query log can either be written to a file or to the slow_log table in the mysql database. To choose the slow query log output destination, set the log_output system variable.

To verify what logging method is used, issue this query:

```
SHOW GLOBAL VARIABLES LIKE 'log_output';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_output    | TABLE |
+---------------+-------+
```

File Logging

File logging is the default. You can configure it explicitly by setting the log_output system variable to `FILE`. (For instance, to switch back from the table logging method.)

Enable file logging until server restart:

```
SET GLOBAL log_output='FILE';
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
log_output=FILE
slow_query_log
slow_query_log_file=slow-queries.log
```

Parsing Specification (File Format)

When logging to a file, the Slow Query Log uses a multi-line format. Parsers must identify the start of a record and correctly separate the header metadata from the SQL query payload.

Record Start Indicator: Every log entry begins with the literal string: `# User@Host:`

Header Specification: The metadata section consists of several lines starting with a hash (`#`). These fields should be mapped as follows:

Line	Component Template	Key Fields
1	`# User@Host: {User}[{User}] @ {Host} [{IP}]`	`User` , `Host` , `IP`
2	`# Thread_id: {ID} Schema: {DB} QC_hit: {No/Yes}`	`Thread ID` , `Schema` , `QC_hit`
3	`# Query_time: {Float} Lock_time: {Float} ...`	`Query_time` , `Lock_time` , `Rows_sent` , `Rows_examined`

SQL Payload: The lines following the `#` headers contain the raw SQL statement.

- **Note:** The SQL query can span multiple lines.
- **Termination:** The record ends when the next `# User@Host:` indicator is encountered or the file ends.

Parsing Example for Tool Developers

A standard parser regex should look for the following sequence:

1. **Start:** `^# User@Host: .*`
2. **Metadata Capture:** Capture all lines starting with `#` until a line without a `#` is reached.
3. **Query Capture:** Everything from the first non-`#` line until the next record start.

Table Logging

The slow query log can either be written to the slow_log table in the mysql database by setting the log_output system variable to `TABLE` . It can be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL log_output='TABLE';
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
log_output=TABLE
slow_query_log
```

Display the slow query log by issuing this statement:

```
SELECT * FROM mysql.slow_log\G
...
*************************** 2. row ***************************
    start_time: 2014-11-11 07:56:28.721519
     user_host: root[root] @ localhost []
    query_time: 00:00:12.000215
     lock_time: 00:00:00.000000
     rows_sent: 1
 rows_examined: 0
            db: test
last_insert_id: 0
     insert_id: 0
     server_id: 1
      sql_text: SELECT SLEEP(12)
     thread_id: 74
...
```

See Writing logs into tables for more information.

Disabling the Log for a Session

Any user can disable logging for a connection by setting the slow_query_log system variable (or, from MariaDB 10.11, log_slow_query) to `0` :

```
SET SESSION slow_query_log=0;
```

Disabling the Log for Specific Statements

You can disable logging to the slow query log for specific types of statements by setting the log_slow_disabled_statements system variable. This option can only be set permanently, in a server option group in an option file prior, then restarting the server:

```
[mariadb]
log_output=FILE
slow_query_log
log_slow_disabled_statements='admin,call,slave,sp'
```

Configuring the Slow Query Log Time

The time that defines a slow query can be configured by setting the long_query_time system variable (or, from MariaDB 10.11, log_slow_query_time). It uses a units of seconds, with an optional milliseconds component. The default value is `10`. It can be changed dynamically with SET GLOBAL:

```
SET GLOBAL long_query_time=5.0;
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
slow_query_log
long_query_time=5.0
```

To verify this works as intended, you can force a “slow” query to happen. By default, MariaDB won’t log a query unless it takes longer than a certain amount of time (10 seconds by default). Check your settings first (output abbreviated):

```
SHOW GLOBAL VARIABLES LIKE 'slow_query_log';
SHOW GLOBAL VARIABLES LIKE 'log_output';
SHOW GLOBAL VARIABLES LIKE 'long_query_time';
+----------------+-------+
| Variable_name  | Value |
+----------------+-------+
| slow_query_log | ON    |
+----------------+-------+
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_output    | TABLE |
+---------------+-------+
+-----------------+-----------+
| Variable_name   | Value     |
+-----------------+-----------+
| long_query_time | 10.000000 |
+-----------------+-----------+
```

Run a sleep statement longer than the threshold of 10 seconds, then inspect the slow query log:

```
SELECT SLEEP(11);
+-----------+
| SLEEP(11) |
+-----------+
|         0 |
+-----------+
1 row in set (11.074 sec)

MariaDB [(none)]> SELECT * FROM mysql.slow_log \G
*************************** 1. row ***************************
    start_time: 2026-02-06 18:38:23.071314
     user_host: stefanhinz[stefanhinz] @ localhost []
    query_time: 00:00:11.073655
     lock_time: 00:00:00.000000
     rows_sent: 1
 rows_examined: 0
            db:
last_insert_id: 0
     insert_id: 0
     server_id: 1
      sql_text: SELECT SLEEP(11)
     thread_id: 7
 rows_affected: 0
```

If you use FILE logging, check the contents of the log file in your terminal.

Logging Queries That Don't Use Indexes

It can be beneficial to log queries that don't use indexes to the slow query log, since such queries can usually be optimized, either by adding an index or by doing a slight rewrite. The slow query log can be configured to log queries that don't use indexes regardless of their execution time, by adding the option `not_using_index` to log_slow_filter, or by setting the log_queries_not_using_indexes system variable to `1`. It can be changed dynamically with SET GLOBAL:

```
SET @@log_slow_filter=concat(@@log_slow_filter,",not_using_index");
SET GLOBAL log_queries_not_using_indexes=ON;
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
log_output=FILE
slow_query_log
slow_query_log_file=slow-queries.log
long_query_time=5.0
log_queries_not_using_indexes=ON
```

As a significant number of queries can run quickly even without indexes, you can use the min_examined_row_limit system variable (or, from MariaDB 10.11, log_slow_min_examined_row_limit) with log_queries_not_using_indexes, to limit the logged queries to those having a material performance impact on the server.

Excluding Queries That Examine Fewer Than a Minimum Row Limit

It can be beneficial to exclude queries that examine fewer than a minimum number of rows from the log. This can be done by setting the min_examined_row_limit system variable, or, from MariaDB 10.11, log_slow_min_examined_row_limit. It can be changed dynamically with SET GLOBAL:

```
SET GLOBAL min_examined_row_limit=100000;
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
log_output=FILE
slow_query_log
slow_query_log_file=slow-queries.log
long_query_time=5.0
min_examined_row_limit=100000
```

Logging Slow Administrative Statements

By default, the slow query log logs administrative statements. To disable that, remove `admin` from the log_slow_filter system variable. Alternatively, set the log_slow_admin_statements system variable to `OFF`. The slow query log considers the following statements administrative: ALTER TABLE, ANALYZE TABLE, CHECK TABLE, CREATE INDEX, DROP INDEX, OPTIMIZE TABLE, and REPAIR TABLE. This also includes ALTER SEQUENCE statements.

You can dynamically enable this feature using a SET GLOBAL statement and setting it for just the current connection with `LOCAL`. Some examples:

```
SET SESSION log_slow_filter=replace(@@log_slow_filter,"admin","");
SET GLOBAL log_slow_admin_statements=ON;
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
...
log_output=FILE
slow_query_log
slow_query_log_file=slow-queries.log
long_query_time=5.0
log_slow_admin_statements=ON
```

Enabling the Slow Query Log for Specific Criteria

It is possible to enable logging to the slow query log for queries that meet specific criteria by configuring the log_slow_filter system variable. It can be changed dynamically with SET GLOBAL:

```
SET GLOBAL log_slow_filter='filesort,filesort_on_disk,tmp_table,tmp_table_on_disk';
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
log_output=FILE
slow_query_log
slow_query_log_file=slow-queries.log
long_query_time=5.0
log_slow_filter=filesort,filesort_on_disk,tmp_table,tmp_table_on_disk
```

You can find all options for log_slow_filter at log_slow_filter system variable or at Slow Query Log Extended Statistics.

Throttling the Slow Query Log

The slow query log can create a lot of I/O[25], so it can be beneficial to throttle it in some cases. The slow query log can be throttled by configuring the log_slow_rate_limit system variable. It can be changed dynamically with SET GLOBAL:

```
SET GLOBAL log_slow_rate_limit=5;
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
log_output=FILE
slow_query_log
slow_query_log_file=slow-queries.log
long_query_time=5.0
log_slow_rate_limit=5
```

Configuring the Verbosity

There are a few optional pieces of information that can be included in the slow query log for each query. This optional information can be included by configuring the log_slow_verbosity system variable. It can be changed dynamically with SET GLOBAL:

```
SET GLOBAL log_slow_verbosity='full';
```

To make this permanent, configure it in a server option group in an option file, then restart the server:

```
[mariadb]
...
log_output=FILE
slow_query_log
slow_query_log_file=slow-queries.log
long_query_time=5.0
log_slow_verbosity=query_plan,explain,engine
```

It is possible to have EXPLAIN output printed in the slow query log.

Viewing the Slow Query Log

Slow query logs written to file can be viewed with any text editor, or you can use the mariadb-dumpslow tool, to ease the process by summarizing the information.

Queries that you find in the log are key queries to try to optimize by constructing a more efficient query or by making better use of indexes.

For queries that appear in the log that cannot be optimized in the above ways, perhaps because they are simply very large selects, due to slow hardware, or very high lock/cpu/io contention, using shard/clustering/load balancing solutions, better hardware, or stats tables may help to improve these queries.

Slow query logs written to table can be viewed by querying the slow_log table.

Variables Related to the Slow Query Log

- slow_query_log - enable/disable the slow query log. Renamed to log_slow_query from MariaDB 10.11.
- log_output - how the output will be written.
- log_slow_admin_statements. Whether to log `OPTIMIZE`, `ANALYZE`, `ALTER`, and other administrative statements to the slow log. Deprecated from MariaDB 11.0, use log_slow_filter without admin.
- slow_query_log_file - name of the slow query log file. Renamed to log_slow_query_file_name from MariaDB 10.11.0.
- long_query_time - time in seconds/microseconds defining a slow query. Renamed to log_slow_query_time from MariaDB 10.11.0.
- log_queries_not_using_indexes - whether or not to log queries that don't use indexes.
- log_slow_admin_statements - whether or not to log certain admin statements.
- log_slow_disabled_statements - types of statements that should not be logged in the slow query log.
- min_examined_row_limit - minimum rows a query must examine to be slow. Renamed to log_slow_min_examined_row_limit from MariaDB 10.11.0.

- log_slow_rate_limit - permits a fraction of slow queries to be logged.
- log_slow_verbosity - amount of detail in the log.
- log_slow_filter - limit which queries to log.
- log_slow_slave_statements - log slow statements executed by replica thread to the slow log if it is open.

Rotating the Slow Query Log on Unix and Linux

Unix and Linux distributions offer the logrotate utility, which makes it easy to rotate log files. See Rotating Logs on Unix and Linux for more information on how to use this utility to rotate the slow query log.

This page is licensed: CC BY-SA / Gnu FDL

EXPLAIN in the Slow Query Log

Additional Columns in MariaDB 10.1 and Later

Starting from MariaDB 10.1.0, the `EXPLAIN` output in the slow query log includes two additional columns: `r_rows` and `r_filtered`

When a user manually executes the standard `EXPLAIN` statement, these columns are not displayed. Instead, they come from the `ANALYZE` statement, which runs the query and returns actual runtime statistics.

MariaDB uses `ANALYZE` statement internally to generate the `EXPLAIN` output when a query is stored in the slow query log. As a result, the slow query log may contain both estimated and actual execution statistics.

For more information about `r_rows` and `r_filtered` columns, see the ANALYZE Statement page.

Switching it On

EXPLAIN output can be switched on by specifying the `explain` keyword in the log_slow_verbosity system variable. Alternatively, you can set with the `log-slow-verbosity` command line argument.

```
[mysqld]
log-slow-verbosity=query_plan,explain
```

EXPLAIN output will only be recorded if the slow query log is written to a file (and not to a table - see Writing logs into tables). This limitation also applies to other extended statistics that are written into the slow query log.

What it Looks Like

When explain recording is on, slow query log entries look like this:

```
# User@Host: user[user] @ test.domain.com [10.6.184.141]
# Thread_id: 75816 Schema: db QC_hit: No
# Query_time: 420.906319 Lock_time: 0.086885 Rows_sent: 0 Rows_examined: 9734191
# Rows_affected: 0
#
# explain: id select_type table type possible_keys key key_len ref rows r_rows filtered r_filtered Extra
# explain: 1 SIMPLE tab ALL a_timestamp_indx NULL NULL NULL 9330481 9734191.00 100.00 55.55 Using where
#
SET timestamp=1535157447;
delete from tab where a_timestamp < (unix_timestamp(now())-14400)*1000000;
```

`EXPLAIN` lines start with `explain:`.

See Also

- MDEV-407

This page is licensed: CC BY-SA / Gnu FDL

log_slow_always_query_time System Variable

- Description: Queries slower than log_slow_always_query_time are not affected by log_slow_rate_limit or log_slow_min_examined_row_limit. Query will be logged to the slow query log if the execution time of the query is longer than log_slow_query_time and log_slow_always_query_time. The argument will be treated as a decimal value with microsecond precision.
- Command line: `--log-slow-always-query-time=num`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric (double)`
- Default Value: `31536000.000000`
- Range: `0` to `31536000`
- Introduced: MariaDB 11.7

This page is licensed: CC BY-SA / Gnu FDL

SQL Error Log Plugin

The `SQL_ERROR_LOG` plugin collects errors sent to clients in a log file defined by sql_error_log_filename, so that they can later be analyzed. The log file can be rotated if sql_error_log_rotate is set.

Errors are logged as they happen and an error will be logged even if it was handled by a condition handler and was never technically *sent* to the client.

From MariaDB 10.11.5 warnings can also be logged if sql_error_log_warnings is enabled.

Comments are also logged, which can make the log easier to search. But this is only possible if the client does not strip the comments away. For example, the mariadb command-line client only leaves comments when started with the –comments option.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN. For example:

```
INSTALL SONAME 'sql_errlog';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
...
plugin_load_add = sql_errlog
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN. For example:

```
UNINSTALL SONAME 'sql_errlog';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time

the server is restarted.

Logging

The log format until MariaDB 10.10 is:

```
Time User Error_code: Error_message : Query
```

Starting from MariaDB 10.11, the format is:

```
Time User Type Error_code: Error_message : Query
```

Starting from MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3, MariaDB 11.3.2, and MariaDB 11.4.1, when the sql_error_log_with_db_and_thread_info variable is enabled, the log also contains thread id and database name. If there is no database, `NULL` will be displayed.

```
Time Thread_id User Database_name Type Error_code: Error_message : Query
```

Each separated by a space or : as above

Option	Description	Version
Time	Time (YYYY-MM-DD hh-mm-ss)	5.5.22
Thread Id	Thread Id of current thread	10.6.17
User	privilege_user [login_user_name] @ hostname [ip]	5.5.22
Database name	Name of the currently selected database	10.6.17
Type	ERROR or WARNING	10.11.6
Error_code	OS error, MariaDB storage engine code (120-199) or MariaDB internal error code (1000-)	5.5.22
Query	Query text	5.5.22

Formal Specification

The SQL Error Log Plugin collects errors sent to clients. Depending on the version and configuration (`sql_error_log_with_db_and_thread_info`), the log follows a positional space-delimited format.

Modern Template (MariaDB 11.4+ with Full Info): `Time Thread_id User Database_name Type Error_code: Error_message : Query`

Time	**`YYYY-MM-DD hh-mm-ss`**	**Timestamp of the error.**
Thread_id	`Unsigned Integer`	Standardized: Thread ID. Matches the `Thread ID` in all other server logs.
User	`String`	The login user and client host/IP.
Database_name	`String`	The selected database or `NULL`.
Type	`ERROR` or `WARNING`	Severity level of the message.
Error_code	`Integer`	MariaDB or OS error code.
Query	`String`	The literal SQL query text.

Example of Logs

```
2023-10-31 15:54:37 root[root] @ localhost [] ERROR 1146: Table 'test.t_doesnt_exist' doesn't exist :
select * from t_doesnt_exist
2023-10-31 15:54:37 root[root] @ localhost [] ERROR 1064: You have an error in your SQL syntax; check the
manual that corresponds to your MariaDB server version for the right syntax to use near
'syntax_error_query' at line 1 : syntax_error_query
2023-10-31 15:54:37 root[root] @ localhost [] ERROR 1146: Table 'test.temptab' doesn't exist : SELECT `c`
FROM `temptab`
2023-11-01 11:31:15 [monty] @ storm [192.168.0.12] ERROR 1051: Unknown table 'test.t1' : drop table t1
```

With sql_error_log_with_db_and_thread_info enabled (database `test` and thread id `4`):

```
2023-10-31 15:54:37 4 root[root] @ localhost [] `test` ERROR 1146: Table 'test.t_doesnt_exist' doesn't
exist : select * from t_doesnt_exist
```

Example Usage

```
install plugin SQL_ERROR_LOG soname 'sql_errlog';
Query OK, 0 rows affected (0.00 sec)

USE test;

SET sql_mode='STRICT_ALL_TABLES,NO_ENGINE_SUBSTITUTION';
Query OK, 0 rows affected (0.00 sec)

CREATE TABLE foo2 (id INT) ENGINE=WHOOPSIE;
ERROR 1286 (42000): Unknown storage engine 'WHOOPSIE'
\! cat data/sql_errors.log
2013-03-19  9:38:40 msandbox[msandbox] @ localhost [] ERROR 1286: Unknown storage engine 'WHOOPSIE' :
CREATE TABLE foo2 (id int) ENGINE=WHOOPSIE
```

Versions

Version	Status	Introduced
1.1	Stable	MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3
1.0	Stable	MariaDB 10.1.13
1.0	Gamma	MariaDB 10.0.10
1.0	Alpha	MariaDB 5.5.22

System Variables and Options

- sql_error_log
- sql_error_log_filename
- sql_error_log_rate
- sql_error_log_rotate
- sql_error_log_rotations
- sql_error_log_size_limit
- sql_error_log_size_warnings
- sql_error_log_with_db_and_thread_info

This page is licensed: CC BY-SA / Gnu FDL

Transaction Coordinator Log

See Transaction Coordinator Log Overview.

Explains the purpose of the Transaction Coordinator (TC) log (`tc.log`), which maintains consistency for XA transactions that affect multiple storage engines, and how to configure it.

See Heuristic Recovery with the Transaction Coordinator Log.

Describes the process of heuristic recovery using the TC log to resolve “in-doubt” transactions that may occur after a server crash during a 2-phase commit.

This page is licensed: CC BY-SA / Gnu FDL

Transaction Coordinator Log Overview

The transaction coordinator log (tc_log) is used to coordinate transactions that affect multiple XA-capable storage engines. If you have two or more XA-capable storage engines enabled, then a transaction coordinator log must be available.

Types of Transaction Coordinator Logs

There are currently two implementations of the transaction coordinator log:

- Binary log-based transaction coordinator log
- Memory-mapped file-based transaction coordinator log

If the binary log is enabled on a server, then the server will use the binary log-based transaction coordinator log. Otherwise, it will use the memory-mapped file-based transaction coordinator log.

Binary Log-Based Transaction Coordinator Log

This transaction coordinator uses the binary log, which is enabled by the log_bin server option.

Memory-Mapped File-Based Transaction Coordinator Log

This transaction coordinator uses the memory-mapped file defined by the –log-tc server option. The size is defined by the log_tc_size system variable.

Some facts about this log:

- The log consists of a memory-mapped file that is divided into pages of 8KB size.
- The usable size of the first page is smaller because of the log header. There is a PAGE control structure for each page.
- Each page (or rather its PAGE control structure) can be in one of the three states - active, syncing, pool.
- There could be only one page in the active or syncing state, but many in the pool state - pool is a fifo queue.
- The usual lifecycle of a page is pool->active->syncing->pool.
- The “active” page is a page where new xid’s are logged.
- The page stays active as long as the syncing slot is taken.
- The “syncing” page is being synced to disk. no new xid can be added to it.
- When the syncing is done the page is moved to a pool and an active page becomes “syncing”.

The result of such an architecture is a natural “commit grouping” - If commits are coming faster than the system can sync, they do not stall. Instead, all commits that came since the last sync are logged to the same “active” page, and they all are synced with the next - one - sync. Thus, thought individual commits are delayed, throughput is not decreasing.

When an xid is added to an active page, the thread of this xid waits for a page’s condition until the page is synced. When a syncing slot becomes vacant one of these waiters is awakened to take care of syncing. It syncs the page and signals all waiters that the page is synced. The waiters are counted, and a page may never become active

again until waiters==0, which means that is all waiters from the previous sync have noticed that the sync was completed.

Note that a page becomes “dirty” and has to be synced only when a new xid is added into it. Removing a xid from a page does not make it dirty - we don’t sync xid removals to disk.

Monitoring the Memory-Mapped File-Based Transaction Coordinator Log

The memory-mapped transaction coordinator log can be monitored with the following status variables:

- Tc_log_max_pages_used
- Tc_log_page_size
- Tc_log_page_waits

Heuristic Recovery with the Transaction Coordinator Log

One of the main purposes of the transaction coordinator log is in crash recovery. See Heuristic Recovery with the Transaction Coordinator Log for more information about that.

Known Issues

You must enable exactly N storage engines

Prior to MariaDB 10.1.10, if you were using the memory-mapped file-based transaction coordinator log, and then if the server crashed and you changed the number of XA-capable storage engines that it loaded, then you could see errors like the following:

```
2018-11-30 23:08:49 140046048638848 [Note] Recovering after a crash using tc.log
2018-11-30 23:08:49 140046048638848 [ERROR] Recovery failed! You must enable exactly 3 storage engines
that support two-phase commit protocol
2018-11-30 23:08:49 140046048638848 [ERROR] Crash recovery failed. Either correct the problem (if it's,
for example, out of memory error) and restart, or delete tc log and start mysqld with --tc-heuristic-
recover={commit|rollback}
2018-11-30 23:08:49 140046048638848 [ERROR] Can't init tc log
2018-11-30 23:08:49 140046048638848 [ERROR] Aborting
```

To recover from this error, delete the file defined by the –log-tc server option, and then restart the server with the –tc-heuristic-recover option set.

See MDEV-9214 for more information.

Bad magic header in tc log

If you are using the memory-mapped file-based transaction coordinator log, then it is possible to see errors like the following:

```
2018-09-19  4:29:31 0 [Note] Recovering after a crash using tc.log
2018-09-19  4:29:31 0 [ERROR] Bad magic header in tc log
2018-09-19  4:29:31 0 [ERROR] Crash recovery failed. Either correct the problem (if it's, for example,
out of memory error) and restart, or delete tc log and start mysqld with --tc-heuristic-recover=
{commit|rollback}
2018-09-19  4:29:31 0 [ERROR] Can't init tc log
2018-09-19  4:29:31 0 [ERROR] Aborting
```

This means that the header of the memory-mapped file-based transaction coordinator log is corrupt. To recover from this error, delete the file defined by the –log-tc server option, and then restart the server with the –tc-heuristic-recover option set.

This issue is known to occur when using docker. In that case, the problem may be caused by using a MariaDB container version with a data directory from a different MariaDB or MySQL version. Therefore, some potential fixes

are:

- Pinning the docker instance to a specific MariaDB version in the docker compose file, so that it consistently uses the same version.
- Running mariadb-upgrade to ensure that the data directory is upgraded to match the server version.

See this docker issue for more information.

MariaDB Galera Cluster

MariaDB Galera Cluster builds include a built-in plugin called `wsrep`. Prior to MariaDB 10.4.3, this plugin was internally considered an XA-capable storage engine. Consequently, these MariaDB Galera Cluster builds have multiple XA-capable storage engines by default, even if the only "real" storage engine that supports external XA transactions enabled on these builds by default is InnoDB. Therefore, when using one these builds MariaDB would be forced to use a transaction coordinator log by default, which could have performance implications.

For example, MDEV-16509 describes performance problems where MariaDB Galera Cluster actually performs better when the binary log is enabled. It is possible that this is caused by the fact that MariaDB is forced to use the memory-mapped file-based transaction coordinator log in this case, which may not perform as well.

This became a bigger issue in MariaDB 10.1 when the MySQL-wsrep patch that powers MariaDB Galera Cluster was enabled on most MariaDB builds on Linux by default. Consequently, this built-in `wsrep` plugin would exist on those MariaDB builds on Linux by default. Therefore, MariaDB users might pay a performance penalty, even if they never actually intended to use the MariaDB Galera Cluster features included in MariaDB 10.1.

In MariaDB 10.4.3 and later, the built-in `wsrep` plugin has been changed to a replication plugin. Therefore, it is no longer considered an XA-capable storage engine, so it no longer forces MariaDB to use a transaction coordinator log by default.

See MDEV-16442 for more information.

This page is licensed: CC BY-SA / Gnu FDL

Heuristic Recovery with the Transaction Coordinator Log

The transaction coordinator log (tc_log) is used to coordinate transactions that affect multiple XA-capable storage engines. One of the main purposes of this log is in crash recovery.

Modes of Crash Recovery

There are two modes of crash recovery:

- Automatic crash recovery.
- Manual heuristic recovery when –tc-heuristic-recover is set to some value other than `OFF`.

Automatic Crash Recovery

Automatic crash recovery occurs during startup when MariaDB needs to recover from a crash and –tc-heuristic-recover is set to `OFF`, which is the default value.

Automatic Crash Recovery with the Binary Log-Based Transaction Coordinator Log

If MariaDB needs to perform automatic crash recovery and if the binary log is enabled, then the error log will contain messages like this:

```
[Note] Recovering after a crash using cmdb-mariadb-0-bin
[Note] InnoDB: Buffer pool(s) load completed at 190313 11:24:29
[Note] Starting crash recovery...
[Note] Crash recovery finished.
```

When the InnoDB-based Binary Log (`--binary-storage-engine=innodb`) is enabled, the binary log is managed transactionally within InnoDB tablespaces. As a result, no separate log exists between InnoDB table data and the binary log. Hence, two-phase commit (2PC) and independent binary log crash recovery are no longer required.

Automatic Crash Recovery with the Memory-Mapped File-Based Transaction Coordinator Log

If MariaDB needs to perform automatic crash recovery and if the binary log is **not** enabled, then the error log will contain messages like this:

```
[Note] Recovering after a crash using tc.log
[Note] InnoDB: Buffer pool(s) load completed at 190313 11:26:32
[Note] Starting crash recovery...
[Note] Crash recovery finished.
```

Manual Heuristic Recovery

Manual heuristic recovery occurs when –tc-heuristic-recover is set to some value other than `OFF` . This might be needed if the server finds prepared transactions during crash recovery that are not in the transaction coordinator log. For example, the error log might contain an error like this:

```
[ERROR] Found 1 prepared transactions! It means that mysqld was not shut down properly last time and
critical recovery information (last binlog or tc.log file) was manually deleted after a crash. You have
to start mysqld with --tc-heuristic-recover switch to commit or rollback pending transactions.
```

When manual heuristic recovery is initiated, MariaDB will ignore information about transactions in the transaction coordinator log during the recovery process. Prepared transactions that are encountered during the recovery process will either be rolled back or committed, depending on the value of –tc-heuristic-recover.

When manual heuristic recovery is initiated, the error log will contain a message like this:

```
[Note] Heuristic crash recovery mode
```

Manual Heuristic Recovery with the Binary Log-Based Transaction Coordinator Log

If –tc-heuristic-recover is set to some value other than `OFF` and if the binary log is enabled, then MariaDB will ignore information about transactions in the binary log during the recovery process. Prepared transactions that are encountered during the recovery process will either be rolled back or committed, depending on the value of –tc-heuristic-recover.

After the recovery process is complete, MariaDB will create a new empty binary log file, so that the old corrupt ones can be ignored.

Manual Heuristic Recovery with the Memory-Mapped File-Based Transaction Coordinator Log

If –tc-heuristic-recover is set to some value other than `OFF` and if the binary log is **not** enabled, then MariaDB will ignore information about transactions in the memory-mapped file defined by the –log-tc option during the recovery process. Prepared transactions that are encountered during the recovery process will either be rolled back or committed, depending on the value of –tc-heuristic-recover.

This page is licensed: CC BY-SA / Gnu FDL

Writing Logs Into Tables

By default, all logs are disabled or written into files. The general query log and the slow query log can also be written to special tables in the `mysql` database. During the startup, entries will always be written into files.

Note that EXPLAIN output will only be recorded if the slow query log is written to a file and not to a table.

To write logs into tables, the log_output server system variable is used. Allowed values are `FILE`, `TABLE` and `NONE`. It is possible to specify multiple values, separated with commas, to write the logs into both tables and files. `NONE` disables logging and has precedence over the other values.

So, to write logs into tables, one of the following settings can be used:

```
SET GLOBAL log_output = 'TABLE';
SET GLOBAL log_output = 'FILE,TABLE';
```

The general log will be written into the general_log table, and the slow query log will be written into the slow_log table. Only a limited set of operations are supported for those special tables. For example, direct DML statements (like `INSERT`) on those tables will fail with an error similar to the following:

```
ERROR 1556 (HY000): You can't use locks with log tables.
```

To flush data to the tables, use FLUSH TABLES instead of FLUSH LOGS.

To empty the contents of the log tables, TRUNCATE TABLE can be used.

The log tables use the CSV storage engine by default. This allows an external program to read the files if needed: normal CSV files are stored in the `mysql` subdirectory, in the data dir. However that engine is slow because it does not support indexes, so you can convert the tables to MyISAM (but not other storage engines). To do so, first temporarily disable logging:

```
SET GLOBAL general_log = 'OFF';
ALTER TABLE mysql.general_log ENGINE = MyISAM;
ALTER TABLE mysql.slow_log ENGINE = MyISAM;
SET GLOBAL general_log = @old_log_state;
```

CHECK TABLE and CHECKSUM TABLE are supported.

CREATE TABLE is supported. ALTER TABLE, RENAME TABLE and DROP TABLE are supported when logging is disabled, but log tables cannot be partitioned.

Contents of log tables are not logged in the binary log, thus cannot be replicated.

This page is licensed: CC BY-SA / Gnu FDL

Variables and Modes

See Options, System & Status Variables.

A comprehensive, alphabetical reference list of all server command-line options, system variables, and status variables available in MariaDB.

See Server Status Variables.

Documentation for server status variables, which provide information about the server's current state and operation (e.g., `Aborted_connects`, `Uptime`), accessed via `SHOW STATUS`.

See Server System Variables.

Reference for MariaDB Server system variables — how to view and set them, their scope and dynamism, and their effect on connections, caching, logging, and performance.

See NEW_MODE.

Explains the `NEW_MODE` system variable (from MariaDB 11.4), which lets you opt in to new behaviors and optimizations in otherwise stable versions.

See OLD_MODE.

Describes the `OLD_MODE` system variable, used to revert specific behaviors to match older MariaDB or MySQL versions for compatibility purposes during upgrades.

See SQL_MODE.

Complete SQL_MODE reference: set via SET/SET GLOBAL/–sql-mode, view @@SQL_MODE, STRICT_TRANS_TABLES, ANSI_QUOTES, TRADITIONAL, and database emulation.

See SQL_MODE=MSSQL.

SQL_MODE=MSSQL enables Microsoft SQL Server compatibility behaviors in MariaDB.

See SQL_MODE=ORACLE.

SQL_MODE=ORACLE enables Oracle Database compatibility behaviors in MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

Options, System & Status Variables

- -a (–ansii)
- –abort-slave-event-count
- Aborted_clients
- Aborted_connects
- Aborted_connects_preauth
- Access_denied_errors
- Acl_column_grants
- Acl_database_grants
- Acl_function_grants
- Acl_package_body_grants
- Acl_package_spec_grants
- Acl_procedure_grants
- Acl_proxy_users
- Acl_role_grants
- Acl_roles
- Acl_table_grants
- Acl_users
- --allow-suspicious-udfs, allow_suspicious_udfs
- alter_algorithm
- analyze_max_length
- analyze_sample_percentage
- --ansii
- aria_block_size
- aria_checkpoint_interval
- aria_checkpoint_log_activity
- aria_encrypt_tables
- aria_force_start_after_recovery_failures
- aria_group_commit
- aria_group_commit_interval

- aria_log_dir_path
- aria_log_file_size
- aria_log_purge_type
- aria_max_sort_file_size
- aria_page_checksum
- aria_pagecache_age_threshold
- aria_pagecache_blocks_not_flushed
- aria_pagecache_blocks_unused
- aria_pagecache_blocks_used
- aria_pagecache_buffer_size
- aria_pagecache_division_limit
- aria_pagecache_file_hash_size
- aria_pagecache_read_requests
- aria_pagecache_reads
- aria_pagecache_segments
- aria_pagecache_write_requests
- aria_pagecache_writes
- aria_recover
- aria_recover_options
- aria_repair_threads
- aria_sort_buffer_size
- aria_stats_method
- aria_sync_log_dir
- aria_transaction_log_syncs
- aria_used_for_temp_tables
- autocommit
- auto_increment_increment
- auto_increment_offset
- automatic_sp_privileges
- aws_key_management_key_spec
- aws_key_management_log_level
- aws_key_management_master_key_id
- aws_key_management_mock
- aws_key_management_region
- aws_key_management_request_timeout
- aws_key_management_rotate_key
- back_log
- -b, basedir
- big_tables
- bind_address
- binlog_alter_two_phase
- binlog_annotate_row_events
- Binlog_bytes_written
- Binlog_cache_disk_use
- binlog_cache_size

- Binlog_cache_use
- binlog_checksum
- binlog_commit_wait_count
- binlog_commit_wait_usec
- Binlog_commits
- binlog_direct_non_transactional_updates
- Binlog_disk_use
- –binlog-do-db, binlog_do_db
- binlog_expire_logs_seconds
- binlog_file_cache_size
- binlog_format
- Binlog_group_commits
- Binlog_group_commit_trigger_count
- Binlog_group_commit_trigger_lock_wait
- Binlog_group_commit_trigger_timeout
- binlog_gtid_index
- Binlog_gtid_index_hit
- Binlog_gtid_index_miss
- binlog_gtid_index_page_size
- binlog_gtid_index_span_min
- --binlog-ignore-db, binlog_ignore_db
- binlog_large_commit_threshold
- binlog_legacy_event_pos
- binlog_optimize_thread_scheduling
- binlog_row_image
- --binlog-row-event-max-size, binlog_row_event_max_size
- binlog_row_metadata
- Binlog_snapshot_file
- Binlog_snapshot_position
- binlog_space_limit
- Binlog_stmt_cache_disk_use
- Binlog_stmt_cache_use
- binlog_stmt_cache_size
- block_encryption_mode
- --bootstrap
- bulk_insert_buffer_size
- Busy_time
- Bytes_received
- Bytes_sent
- character_set_client
- --character-set-client-handshake
- character_set_collations
- character_set_connection
- character_set_database
- character_set_filesystem

- character_set_results
- -C, character_set_server
- character_set_system
- character_sets_dir
- check_constraint_checks
- -r, –chroot
- collation_connection
- collation_database
- collation_server
- Column_compressions
- column_compression_threshold
- column_compression_zlib_level
- column_compression_zlib_strategy
- column_compression_zlib_wrap
- Column_decompressions
- Com_admin_commands
- Com_alter_db
- Com_alter_db_upgrade
- Com_alter_event
- Com_alter_function
- Com_alter_procedure
- Com_alter_sequence
- Com_alter_server
- Com_alter_table
- Com_alter_tablespace
- Com_alter_user
- Com_analyze
- Com_assign_to_keycache
- Com_backup
- Com_backup_lock
- Com_backup_table
- Com_begin
- Com_binlog
- Com_call_procedure
- Com_change_db
- Com_change_master
- Com_check
- Com_checksum
- Com_commit
- Com_compound_sql
- Com_create_db
- Com_create_event
- Com_create_function
- Com_create_index
- Com_create_package

- Com_create_package_body
- Com_create_procedure
- Com_create_role
- Com_create_sequence
- Com_create_server
- Com_create_table
- Com_create_temporary_table
- Com_create_trigger
- Com_create_udf
- Com_create_user
- Com_create_view
- Com_dealloc_sql
- Com_delete
- Com_delete_multi
- Com_do
- Com_drop_db
- Com_drop_event
- Com_drop_function
- Com_drop_index
- Com_drop_package
- Com_drop_package_body
- Com_drop_procedure
- Com_drop_role
- Com_drop_sequence
- Com_drop_server
- Com_drop_table
- Com_drop_temporary_table
- Com_drop_trigger
- Com_drop_user
- Com_drop_user
- Com_drop_view
- Com_empty_query
- Com_execute_immediate
- Com_execute_sql
- Com_flush
- Com_get_diagnostics
- Com_grant
- Com_grant_role
- Com_ha_close
- Com_ha_open
- Com_ha_read
- Com_help
- Com_insert
- Com_insert_select
- Com_install_plugin

- Com_kill
- Com_load
- Com_load_master_data
- Com_load_master_table
- Com_lock_tables
- Com_multi
- Com_optimize
- Com_preload_keys
- Com_prepare_sql
- Com_purge
- Com_purge_before_date
- Com_release_savepoint
- Com_rename_table
- Com_rename_user
- Com_repair
- Com_replace
- Com_replace_select
- Com_reset
- Com_resignal
- Com_restore_table
- Com_revoke
- Com_revoke_all
- Com_revoke_grant
- Com_rollback
- Com_rollback_to_savepoint
- Com_savepoint
- Com_select
- Com_set_option
- Com_show_authors
- Com_show_binlog_events
- Com_show_binlogs
- Com_show_charsets
- Com_show_client_statistics
- Com_show_collations
- Com_show_column_types
- Com_show_contributors
- Com_show_create_db
- Com_show_create_event
- Com_show_create_func
- Com_show_create_package
- Com_show_create_package_body
- Com_show_create_proc
- Com_show_create_table
- Com_show_create_trigger
- Com_show_create_user

- Com_show_databases
- Com_show_engine_logs
- Com_show_engine_mutex
- Com_show_engine_status
- Com_show_events
- Com_show_errors
- Com_show_explain
- Com_show_fields
- Com_show_function_status
- Com_show_generic
- Com_show_grants
- Com_show_keys
- Com_show_index_statistics
- Com_show_binlog_status
- Com_show_master_status
- Com_show_new_master
- Com_show_open_tables
- Com_show_package_status
- Com_show_package_body_status
- Com_show_plugins
- Com_show_privileges
- Com_show_procedure_status
- Com_show_processlist
- Com_show_profile
- Com_show_profiles
- Com_show_relaylog_events
- Com_show_slave_hosts
- Com_show_slave_status
- Com_show_status
- Com_show_storage_engines
- Com_show_table_statistics
- Com_show_table_status
- Com_show_tables
- Com_show_triggers
- Com_show_user_statistics
- Com_show_variable
- Com_show_warnings
- Com_shutdown
- Com_signal
- Com_slave_start
- Com_slave_stop
- Com_start_all_slaves
- Com_start_slave
- Com_stop_all_slaves
- Com_stop_slave

- Com_stmt_close
- Com_stmt_execute
- Com_stmt_fetch
- Com_stmt_prepare
- Com_stmt_reprepare
- Com_stmt_reset
- Com_stmt_send_long_data
- Com_truncate
- Com_uninstall_plugin
- Com_unlock_tables
- Com_update
- Com_update_multi
- Com_xa_commit
- Com_xa_end
- Com_xa_prepare
- Com_xa_recover
- Com_xa_rollback
- Com_xa_start
- completion_type
- Compression
- concurrent_insert
- connect_class_path
- connect_cond_push
- connect_conv_size
- connect_default_depth
- connect_default_prec
- connect_enable_mongo
- connect_exact_info
- connect_force_bson
- connect_indx_map
- connect_java_wrapper
- connect_json_all_path
- connect_json_grp_size
- connect_json_null
- connect_jvm_path
- connect_timeout
- connect_type_conv
- connect_use_tempfile
- connect_work_size
- connect_xtrace
- Connection_errors_accept
- Connection_errors_internal
- Connection_errors_max_connections
- Connection_errors_peer_address
- Connection_errors_select

- Connection_errors_tcpwrap
- Connections
- –console
- core_file
- Cpu_time
- --cracklib-password-check
- cracklib_password_check-dictionary
- create_tmp_table_binlog_formats
- Created_tmp_disk_tables
- Created_tmp_files
- Created_tmp_tables
- -h, datadir
- date_format
- datetime_format
- deadlock_search_depth_long
- deadlock_search_depth_short
- deadlock_timeout_long
- deadlock_timeout_short
- -#, debug
- –debug-assert-if-crashed-table
- –debug-binlog-fsync-sleep
- –debug-crc-break
- –debug-flush
- –debug-no-sync
- debug_no_thread_alarm
- debug_sync
- –debug-sync-timeout
- --default-character-set
- default_master_connection
- default_password_lifetime
- default_regex_flags
- default_storage_engine
- default_table_type
- default_tmp_storage_engine
- --default-time-zone
- default_week_format
- --defaults-extra-file
- --defaults-file
- delay_key_write
- Delayed_errors
- delayed_insert_limit
- Delayed_insert_threads
- delayed_insert_timeout
- delayed_queue_size
- Delayed_writes

- Delete_scan
- --des-key-file
- disconnect_on_expired_password
- --disconnect-slave-event-count
- --disks
- div_precision_increment
- Empty_queries
- encrypt_binlog
- encrypt_tmp_disk_tables
- encrypt_tmp_files
- encryption_algorithm
- enforce_storage_engine
- engine_condition_pushdown
- eq_range_index_dive_limit
- error_count
- event_scheduler
- Executed_events
- Executed_triggers
- -T, –exit-info
- expensive_subquery_limit
- expire_logs_days
- explicit_defaults_for_timestamp
- --external-locking
- external_user
- extra_max_connections
- extra_port
- Feature_application_time_periods
- Feature_check_constraint
- Feature_custom_aggregate_functions
- Feature_delay_key_write
- Feature_dynamic_columns
- Feature_fulltext
- Feature_gis
- Feature_insert_returning
- Feature_invisible_columns
- Feature_json
- Feature_locale
- Feature_subquery
- Feature_timezone
- Feature_trigger
- Feature_window_functions
- Feature_xml
- --feedback
- feedback_http_proxy
- feedback_send_retry_wait

- feedback_send_timeout
- feedback_server_uid
- feedback_url
- feedback_user_info
- file_key_management_encryption_algorithm
- file_key_management_filekey
- file_key_management_filename
- --flashback
- flush
- Flush_commands
- flush_time
- foreign_key_checks
- ft_boolean_syntax
- ft_max_word_len
- ft_min_word_len
- ft_query_expansion_limit
- ft_stopword_file
- --gdb
- general_log
- general_log_file
- --getopt-prefix-matching
- group_concat_max_len
- gssapi_keytab_path
- gssapi_principal_name
- gssapi_mech_name
- gtid_binlog_pos
- gtid_binlog_state
- gtid_cleanup_batch_size
- gtid_current_pos
- gtid_domain_id
- gtid_ignore_duplicates
- gtid_pos_auto_engines
- gtid_seq_no
- gtid_slave_pos
- gtid_strict_mode
- -h, datadir
- Handler_commit
- Handler_delete
- Handler_discover
- Handler_external_lock
- Handler_icp_attempts
- Handler_icp_match
- Handler_mrr_init
- Handler_mrr_key_refills
- Handler_mrr_rowid_refills

- Handler_prepare
- Handler_read_first
- Handler_read_key
- Handler_read_last
- Handler_read_next
- Handler_read_prev
- Handler_read_retry
- Handler_read_rnd
- Handler_read_rnd_deleted
- Handler_read_rnd_next
- Handler_rollback
- Handler_savepoint
- Handler_savepoint_rollback
- Handler_tmp_delete
- Handler_tmp_update
- Handler_tmp_write
- Handler_update
- Handler_write
- handlersocket_accept_balance
- handlersocket_address
- handlersocket_backlog
- handlersocket_epoll
- handlersocket_plain_secret
- handlersocket_plain_secret_wr
- handlersocket_port
- handlersocket_port_wr
- handlersocket_rcvbuf
- handlersocket_readsize
- handlersocket_sndbuf
- handlersocket_threads
- handlersocket_threads_wr
- handlersocket_timeout
- handlersocket_verbose
- handlersocket_wrlock_timeout
- hashicorp-key-management-cache-timeout
- hashicorp-key-management-cache-version-timeout
- hashicorp-key-management-caching-enabled
- hashicorp-key-management-check-kv-version
- hashicorp-key-management-max-retries
- hashicorp-key-management-timeout
- hashicorp-key-management-token
- hashicorp-key-management-use-cache-on-timeout
- hashicorp-key-management-vault-ca
- hashicorp-key-management-vault-url
- have_compress

- have_crypt
- have_csv
- have_dynamic_loading
- have_geometry
- have_innodb
- have_ndbcluster
- have_openssl
- have_partitioning
- have_profiling
- have_query_cache
- have_rtree_keys
- have_ssl
- have_symlink
- --help
- histogram_size
- histogram_type
- host_cache_size
- hostname
- identity
- idle_readonly_transaction_timeout
- idle_transaction_timeout
- idle_write_transaction_timeout
- ignore_db_dirs
- ignore_builtin_innodb
- in_predicate_conversion_threshold
- in_transaction
- init_connect
- init_file
- --init-rpl-role
- init_slave
- --innodb
- innodb_adaptive_checkpoint
- innodb_adaptive_flushing
- innodb_adaptive_flushing_lwm
- innodb_adaptive_flushing_method
- Innodb_adaptive_hash_cells
- Innodb_adaptive_hash_hash_searches
- Innodb_adaptive_hash_heap_buffers
- innodb_adaptive_hash_index
- innodb_adaptive_hash_index_partitions
- innodb_adaptive_hash_index_parts
- Innodb_adaptive_hash_non_hash_searches
- innodb_adaptive_max_sleep_delay
- innodb_additional_mem_pool_size
- innodb_alter_copy_bulk

- innodb_api_bk_commit_interval
- innodb_api_disable_rowlock
- innodb_api_enable_binlog
- innodb_api_enable_mdl
- innodb_api_trx_level
- Innodb_async_reads_pending
- Innodb_async_reads_queue_size
- Innodb_async_reads_tasks_running
- Innodb_async_reads_total_enqueues
- Innodb_async_reads_total_count
- Innodb_async_reads_wait_slot_sec
- Innodb_async_writes_pending
- Innodb_async_writes_queue_size
- Innodb_async_writes_tasks_running
- Innodb_async_writes_total_enqueues
- Innodb_async_writes_total_count
- Innodb_async_writes_wait_slot_sec
- innodb-auto-lru-dump
- innodb_autoextend_increment
- innodb_autoinc_lock_mode
- Innodb_available_undo_logs
- Innodb_background_log_sync
- innodb_background_scrub_data_check_interval
- innodb_background_scrub_data_compressed
- innodb_background_scrub_data_interval
- innodb_background_scrub_data_uncompressed
- innodb_blocking_buffer_pool_restore
- innodb_buf_dump_status_frequency
- Innodb_buffer_pool_bytes_data
- Innodb_buffer_pool_bytes_dirty
- innodb_buffer_pool_chunk_size
- innodb_buffer_pool_dump_at_shutdown
- innodb_buffer_pool_dump_now
- innodb_buffer_pool_dump_pct
- Innodb_buffer_pool_dump_status
- innodb_buffer_pool_evict
- innodb_buffer_pool_filename
- innodb_buffer_pool_instances
- innodb_buffer_pool_load_abort
- innodb_buffer_pool_load_at_startup
- innodb_buffer_pool_load_now
- Innodb_buffer_pool_load_incomplete
- innodb_buffer_pool_load_pages_abort
- Innodb_buffer_pool_load_status
- Innodb_buffer_pool_pages_data

- Innodb_buffer_pool_pages_dirty
- Innodb_buffer_pool_pages_flushed
- Innodb_buffer_pool_pages_LRU_flushed
- Innodb_buffer_pool_pages_LRU_freed
- Innodb_buffer_pool_pages_free
- Innodb_buffer_pool_pages_made_not_young
- Innodb_buffer_pool_pages_made_young
- Innodb_buffer_pool_pages_misc
- Innodb_buffer_pool_pages_old
- Innodb_buffer_pool_pages_total
- innodb_buffer_pool_populate
- Innodb_buffer_pool_read_ahead
- Innodb_buffer_pool_read_ahead_evicted
- Innodb_buffer_pool_read_ahead_rnd
- Innodb_buffer_pool_read_requests
- Innodb_buffer_pool_reads
- Innodb_buffer_pool_resize_status
- innodb_buffer_pool_restore_at_startup
- innodb_buffer_pool_shm_checksum
- innodb_buffer_pool_shm_key
- innodb_buffer_pool_size
- innodb_buffer_pool_size_auto_min
- innodb_buffer_pool_size_max
- Innodb_buffer_pool_wait_free
- Innodb_buffer_pool_write_requests
- Innodb_buffered_aio_submitted
- innodb_change_buffer_dump
- innodb_change_buffer_max_size
- innodb_change_buffering
- innodb_change_buffering_debug
- Innodb_checkpoint_age
- innodb_checkpoint_age_target
- Innodb_checkpoint_max_age
- Innodb_checkpoint_target_age
- innodb_checksum_algorithm
- innodb_checksums
- innodb_cleaner_lsn_age_factor
- --innodb-cmp
- innodb_cmp_per_index_enabled
- --innodb-cmp-reset
- --innodb-cmpmem
- --innodb-cmpmem-reset
- innodb_commit_concurrency
- innodb_compression_algorithm
- innodb_compression_default

- innodb_compression_failure_threshold_pct
- innodb_compression_level
- innodb_compression_pad_pct_max
- innodb_concurrency_tickets
- innodb_corrupt_table_action
- Innodb_current_row_locks
- innodb_data_file_buffering
- innodb_data_file_path
- innodb_data_file_write_through
- Innodb_data_fsyncs
- innodb_data_home_dir
- Innodb_data_pending_fsyncs
- Innodb_data_pending_reads
- Innodb_data_pending_writes
- Innodb_data_read
- Innodb_data_reads
- Innodb_data_writes
- Innodb_data_written
- Innodb_dblwr_pages_written
- Innodb_dblwr_writes
- innodb_deadlock_detect
- innodb_deadlock_report
- Innodb_deadlocks
- innodb_default_encryption_key_id
- innodb_default_page_encryption_key
- innodb_default_row_format
- innodb_defragment
- Innodb_defragment_compression_failures
- Innodb_defragment_count
- Innodb_defragment_failures
- innodb_defragment_fill_factor
- innodb_defragment_fill_factor_n_recs
- innodb_defragment_frequency
- innodb_defragment_n_pages
- innodb_defragment_stats_accuracy
- innodb_dict_size_limit
- Innodb_dict_tables
- innodb_disable_sort_file_cache
- innodb_disallow_writes
- innodb_doublewrite
- innodb_doublewrite_file
- innodb_empty_free_list_algorithm
- innodb_enable_unsafe_group_commit
- innodb_encrypt_log
- innodb_encrypt_tables

- innodb_encrypt_temporary_tables
- Innodb_encryption_n_merge_blocks_decrypted
- Innodb_encryption_n_merge_blocks_encrypted
- Innodb_encryption_n_rowlog_blocks_decrypted
- Innodb_encryption_n_rowlog_blocks_encrypted
- Innodb_encryption_n_temp_blocks_decrypted
- Innodb_encryption_n_temp_blocks_encrypted
- Innodb_encryption_num_key_requests
- innodb_encryption_rotate_key_age
- Innodb_encryption_rotation_estimated_iops
- innodb_encryption_rotation_iops
- Innodb_encryption_rotation_pages_flushed
- Innodb_encryption_rotation_pages_modified
- Innodb_encryption_rotation_pages_read_from_cache
- Innodb_encryption_rotation_pages_read_from_disk
- innodb_encryption_threads
- innodb_extra_rsegments
- innodb_extra_undoslots
- innodb_fake_changes
- innodb_fast_checksum
- innodb_fast_shutdown
- innodb_fatal_semaphore_wait_threshold
- innodb_file_format
- innodb_file_format_check
- innodb_file_format_max
- --innodb-file-io-threads
- innodb_file_per_table
- innodb_fill_factor
- innodb_flush_log_at_timeout
- innodb_flush_log_at_trx_commit
- innodb_flush_method
- innodb_flush_neighbor_pages
- innodb_flush_neighbors
- innodb_flush_sync
- innodb_flushing_avg_loops
- innodb_force_load_corrupted
- innodb_force_primary_key
- innodb_force_recovery
- innodb_foreground_preflush
- innodb_ft_aux_table
- innodb_ft_cache_size
- innodb_ft_enable_diag_print
- innodb_ft_enable_stopword
- innodb_ft_max_token_size
- innodb_ft_min_token_size

- innodb_ft_num_word_optimize
- innodb_ft_result_cache_limit
- innodb_ft_server_stopword_table
- innodb_ft_sort_pll_degree
- innodb_ft_total_cache_size
- innodb_ft_user_stopword_table
- Innodb_have_atomic_builtins
- Innodb_have_bzip2
- Innodb_have_lz4
- Innodb_have_lzma
- Innodb_have_lzo
- Innodb_have_punch_hole
- Innodb_have_snappy
- Innodb_history_list_length
- innodb_ibuf_accel_rate
- innodb_ibuf_active_contract
- Innodb_ibuf_discarded_delete_marks
- Innodb_ibuf_discarded_deletes
- Innodb_ibuf_discarded_inserts
- Innodb_ibuf_free_list
- innodb_ibuf_max_size
- Innodb_ibuf_merged_delete_marks
- Innodb_ibuf_merged_deletes
- Innodb_ibuf_merged_inserts
- Innodb_ibuf_merges
- Innodb_ibuf_segment_size
- Innodb_ibuf_size
- innodb_idle_flush_pct
- innodb_immediate_scrub_data_uncompressed
- innodb_import_table_from_xtrabackup
- --innodb-index-stats
- Innodb_instant_alter_column
- innodb_instant_alter_column_allowed
- innodb_instrument_semaphores
- innodb_io_capacity
- innodb_io_capacity_max
- innodb_kill_idle_transaction
- innodb_large_prefix
- innodb_lazy_drop_table
- innodb_lock_schedule_algorithm
- innodb_lock_wait_timeout
- --innodb-lock-waits
- innodb_locking_fake_changes
- --innodb-locks
- innodb_locks_unsafe_for_binlog

- innodb_log_arch_dir
- innodb_log_arch_expire_sec
- innodb_log_archive
- innodb_log_block_size
- innodb_log_buffer_size
- innodb_log_checkpoint_now
- innodb_log_checksum_algorithm
- innodb_log_checksums
- innodb_log_compressed_pages
- innodb_log_file_buffering
- innodb_log_file_mmap
- innodb_log_file_size
- innodb_log_file_write_through
- innodb_log_files_in_group
- innodb_log_group_home_dir
- innodb_log_optimize_ddl
- innodb_log_spin_wait_delay
- Innodb_log_waits
- innodb_log_write_ahead_size
- Innodb_log_write_requests
- Innodb_log_writes
- innodb_lru_flush_size
- innodb_lru_scan_depth
- Innodb_lsn_current
- Innodb_lsn_flushed
- Innodb_lsn_last_checkpoint
- Innodb_master_thread_1_second_loops
- Innodb_master_thread_10_second_loops
- Innodb_master_thread_active_loops
- Innodb_master_thread_background_loops
- Innodb_master_thread_idle_loops
- Innodb_master_thread_main_flush_loops
- Innodb_master_thread_sleeps
- innodb_max_bitmap_file_size
- innodb_max_changed_pages
- innodb_max_dirty_pages_pct
- innodb_max_dirty_pages_pct_lwm
- innodb_max_purge_lag
- innodb_max_purge_lag_delay
- innodb_max_purge_lag_wait
- Innodb_max_trx_id
- innodb_max_undo_log_size
- Innodb_mem_adaptive_hash
- Innodb_mem_dictionary
- Innodb_mem_total

- innodb_merge_sort_block_size
- innodb_mirrored_log_groups
- innodb_monitor_disable
- innodb_monitor_enable
- innodb_monitor_reset
- innodb_monitor_reset_all
- innodb_mtflush_threads
- Innodb_mutex_os_waits
- Innodb_mutex_spin_rounds
- Innodb_mutex_spin_waits
- Innodb_num_index_pages_written
- Innodb_num_non_index_pages_written
- Innodb_num_open_files
- Innodb_num_page_compressed_trim_op
- Innodb_num_page_compressed_trim_op_saved
- Innodb_num_pages_encrypted
- Innodb_num_pages_page_compressed
- Innodb_num_pages_page_compression_error
- Innodb_num_pages_page_decompressed
- Innodb_num_pages_page_decrypted
- Innodb_num_pages_page_encryption_error
- innodb_numa_interleave
- innodb_old_blocks_pct
- innodb_old_blocks_time
- Innodb_oldest_view_low_limit_trx_id
- innodb_online_alter_log_max_size
- Innodb_onlineddl_pct_progress
- Innodb_onlineddl_rowlog_pct_used
- Innodb_onlineddl_rowlog_rows
- innodb_open_files
- innodb_optimize_fulltext_only
- Innodb_os_log_fsyncs
- Innodb_os_log_pending_fsyncs
- Innodb_os_log_pending_writes
- Innodb_os_log_written
- innodb_page_cleaners
- Innodb_page_compression_saved
- Innodb_page_compression_trim_sect512
- Innodb_page_compression_trim_sect1024
- Innodb_page_compression_trim_sect2048
- Innodb_page_compression_trim_sect4096
- Innodb_page_compression_trim_sect8192
- Innodb_page_compression_trim_sect16384
- Innodb_page_compression_trim_sect32768
- innodb_page_size

- Innodb_page_size
- Innodb_pages_created
- Innodb_pages_read
- Innodb_pages0_read
- Innodb_pages_written
- innodb-pass-corrupt-table
- innodb_prefix_index_cluster_optimization
- innodb_print_all_deadlocks
- innodb_purge_batch_size
- innodb_purge_rseg_truncate_frequency
- innodb_purge_threads
- Innodb_purge_trx_id
- Innodb_purge_undo_no
- innodb_random_read_ahead
- innodb_read_ahead
- innodb_read_ahead_threshold
- innodb_read_io_threads
- innodb_read_only
- Innodb_read_views_memory
- innodb_recovery_stats
- innodb-recovery-update-relay-log
- innodb_replication_delay
- innodb_rollback_on_timeout
- innodb_rollback_segments
- Innodb_row_lock_current_waits
- Innodb_row_lock_numbers
- Innodb_row_lock_time
- Innodb_row_lock_time_avg
- Innodb_row_lock_time_max
- Innodb_row_lock_waits
- Innodb_rows_deleted
- Innodb_rows_inserted
- Innodb_rows_read
- Innodb_rows_updated
- --innodb-rseg
- Innodb_s_lock_os_waits
- Innodb_s_lock_spin_rounds
- Innodb_s_lock_spin_waits
- innodb_safe_truncate
- innodb_sched_priority_cleaner
- Innodb_scrub_background_page_reorganizations
- Innodb_scrub_background_page_split_failures_missing_index
- Innodb_scrub_background_page_split_failures_out_of_filespace
- Innodb_scrub_background_page_split_failures_underflow
- Innodb_scrub_background_page_split_failures_unknown

- Innodb_scrub_background_page_splits
- innodb_scrub_log
- Innodb_scrub_log
- innodb_scrub_log_interval
- innodb_scrub_log_speed
- Innodb_secondary_index_triggered_cluster_reads
- Innodb_secondary_index_triggered_cluster_reads_avoided
- innodb-show-locks-held
- innodb_show_verbose_locks
- innodb_simulate_comp_failures
- innodb_snapshot_isolation
- innodb_sort_buffer_size
- innodb_spin_wait_delay
- innodb_stats_auto_recalc
- innodb_stats_auto_update
- innodb_stats_include_delete_marked
- innodb_stats_method
- innodb_stats_modified_counter
- innodb-stats-on-metadata
- innodb_stats_persistent
- innodb_stats_persistent_sample_pages
- innodb_stats_sample_pages
- innodb_stats_traditional
- innodb_stats_transient_sample_pages
- innodb_stats_update_need_lock
- --innodb-status-file
- innodb_status_output
- innodb_status_output_locks
- innodb_strict_mode
- innodb_support_xa
- innodb_sync_array_size
- innodb_sync_spin_loops
- --innodb-sys-indexes
- --innodb-sys-stats
- --innodb-sys-tables
- Innodb_system_rows_deleted
- Innodb_system_rows_inserted
- Innodb_system_rows_read
- Innodb_system_rows_updated
- innodb_table_locks
- --innodb-table-stats
- innodb_temp_data_file_path
- innodb_thread_concurrency
- innodb_thread_concurrency_timer_based
- innodb_thread_sleep_delay

- innodb_tmpdir
- innodb_track_changed_pages
- innodb_track_redo_log_now
- --innodb-trx
- innodb_truncate_temporary_tablespace_now
- Innodb_truncated_status_writes
- innodb_undo_directory
- innodb_undo_log_truncate
- innodb_undo_logs
- innodb_undo_tablespaces
- Innodb_undo_truncations
- innodb_use_atomic_writes
- innodb_use_fallocate
- innodb_use_global_flush_log_at_trx_commit
- innodb_use_mtflush
- innodb_use_native_aio
- innodb_use_purge_thread
- innodb_use_stacktrace
- innodb_use_sys_malloc
- innodb_use_sys_stats_table
- innodb_use_trim
- innodb_version
- innodb_write_io_threads
- Innodb_x_lock_os_waits
- Innodb_x_lock_spin_rounds
- Innodb_x_lock_spin_waits
- --insert_id
- --install
- --install-manual
- interactive_timeout
- join_buffer_size
- join_buffer_space_limit
- join_cache_level
- keep_files_on_create
- Key_blocks_not_flushed
- Key_blocks_unused
- Key_blocks_used
- Key_blocks_warm
- key_buffer_size
- key_cache_age_threshold
- key_cache_block_size
- key_cache_division_limit
- key_cache_file_hash_size
- key_cache_segments
- Key_read_requests

- Key_reads
- Key_write_requests
- Key_writes
- -L, language
- large_files_support
- large_page_size
- large_pages
- last_gtid
- last_insert_id
- Last_query_cost
- lc_messages
- lc_messages_dir
- lc_time_names
- legacy_xa_rollback_at_disconnect
- license
- local_infile
- lock_wait_timeout
- locked_in_memory
- -l, log
- --log-basename
- log_bin
- log_bin_basename
- log_bin_compress
- log_bin_compress_min_len
- log_bin_index
- log_bin_trust_function_creators
- --log-bin-trust-routine-creators
- --log-ddl-recovery
- log_disabled_statements
- log_error
- -0, –log-long-format
- log_output
- log_queries_not_using_indexes
- --log-short-format
- log_slave_updates
- log_slow_admin_statements
- log_slow_always_query_time
- log_slow_disabled_statements
- --log-slow-file
- log_slow_filter
- log_slow_min_examined_row_limit
- log_slow_queries
- log_slow_query
- log_slow_query_file
- log_slow_query_time

- log_slow_rate_limit
- log_slow_slave_statements
- --log-slow-time
- log_slow_verbosity
- --log-tc
- log_tc_size
- -W, –log-warnings, log_warnings
- long_query_time
- --log-isam
- low_priority_updates
- lower_case_file_system
- lower_case_table_names
- --master-connect-retry
- Master_gtid_wait_count
- Master_gtid_wait_time
- Master_gtid_wait_timeouts
- --master-host
- master_info_file
- --master-password
- --master-port
- --master-retry-count
- --master-ssl
- --master-ssl-ca
- --master-ssl-capath
- --master-ssl-cert
- --master-ssl-cipher
- --master-ssl-key
- --master-user
- master_verify_checksum
- max_allowed_packet
- --max-binlog-dump-events
- max_binlog_cache_size
- max_binlog_size
- max_binlog_stmt_cache_size
- max_binlog_total_size
- max_connect_errors
- max_connections
- max_delayed_threads
- max_digest_length
- max_error_count
- max_heap_table_size
- max_insert_delayed_threads
- max_join_size
- max_length_for_sort_data
- max_long_data_size

- max_open_cursors
- max_prepared_stmt_count
- max_password_errors
- max_recursive_iterations
- max_relay_log_size
- max_rowid_filter_size
- max_seeks_for_key
- max_session_mem_used
- max_sort_length
- max_sp_recursion_depth
- max_statement_time
- Max_statement_time_exceeded
- max_tmp_session_space_usage
- Max_tmp_space_used
- max_tmp_tables
- max_tmp_total_space_usage
- Max_used_connections_time
- Max_used_connections
- max_user_connections
- max_write_lock_count
- --memlock
- Memory_used
- Memory_used_initial
- metadata_locks_cache_size
- metadata_locks_hash_instances
- metadata_locks_instances
- mhnsw_default_distance
- mhnsw_default_m
- mhnsw_ef_search
- mhnsw_max_cache_size
- min-examined-row-limit
- mroonga_action_on_fulltext_query_error
- mroonga_boolean_mode_syntax_flags
- Mroonga_count_skip
- mroonga_database_path_prefix
- mroonga_default_parser
- mroonga_default_tokenizer
- mroonga_default_wrapper_engine
- mroonga_dry_write
- mroonga_enable_operations_recording
- mroonga_enable_optimization
- Mroonga_fast_order_limit
- mroonga_libgroonga_embedded
- mroonga_libgroonga_support_zlib
- mroonga_libgroonga_support_zstd

- mroonga_libgroonga_version
- mroonga_log_file
- mroonga_log_level
- mroonga_match_escalation_threshold
- mroonga_max_n_records_for_estimate
- mroonga_query_log_file
- mroonga_vector_column_delimiter
- mroonga_version
- mrr_buffer_size
- multi_range_count
- myisam_block_size
- myisam_data_pointer_size
- myisam_max_extra_sort_file_size
- myisam_max_sort_file_size
- myisam_mmap_size
- myisam_recover_options
- myisam_repair_threads
- myisam_sort_buffer_size
- myisam_stats_method
- myisam_use_mmap
- mysql56_temporal_format
- named_pipe
- --ndb-use-copying-alter-table
- net_buffer_length
- net_read_timeout
- net_retry_count
- net_write_timeout
- Not_flushed_delayed_rows
- --new
- --old, old
- old_alter_table
- old_mode
- old_passwords
- --old-style-user-limits
- --one-thread
- Open_files
- open_files_limit
- Open_streams
- Open_table_definitions
- Open_tables
- Opened_files
- Opened_plugin_libraries
- Opened_table_definitions
- Opened_tables
- Opened_views

- optimizer_adjust_secondary_key_costs
- optimizer_extra_pruning_depth
- optimizer_join_limit_pref_ratio
- optimizer_max_sel_arg_weight
- optimizer_max_sel_args
- optimizer_prune_level
- optimizer_search_depth
- optimizer_selectivity_sampling_limit
- optimizer_switch
- optimizer_record_context
- optimizer_trace
- optimizer_trace_max_mem_size
- optimizer_use_condition_selectivity
- oqgraph_allow_create_integer_latch
- Oqgraph_boost_version
- Oqgraph_compat_mode
- Oqgraph_verbose_debug
- pam_debug
- -P, –port, port
- pam_use_cleartext_plugin
- pam_windbind_workaround
- password_reuse_check_interval
- performance_schema
- Performance_schema_accounts_lost
- performance_schema_accounts_size
- Performance_schema_cond_classes_lost
- Performance_schema_cond_instances_lost
- –performance-schema-consumer-events-stages-current
- –performance-schema-consumer-events-stages-history
- –performance-schema-consumer-events-stages-history-long
- –performance-schema-consumer-events-statements-current
- –performance-schema-consumer-events-statements-history
- –performance-schema-consumer-events-statements-history-long
- –performance-schema-consumer-events-waits-current
- –performance-schema-consumer-events-waits-history
- –performance-schema-consumer-events-waits-history-long
- –performance-schema-consumer-global-instrumentation
- –performance-schema-consumer-statements-digest
- –performance-schema-consumer-thread-instrumentation
- Performance_schema_digest_lost
- performance_schema_digests_size
- performance_schema_events_stages_history_long_size
- performance_schema_events_stages_history_size
- performance_schema_events_statements_history_long_size
- performance_schema_events_statements_history_size

- performance_schema_events_transactions_history_long_size
- performance_schema_events_transactions_history_size
- performance_schema_events_waits_history_long_size
- performance_schema_events_waits_history_size
- Performance_schema_file_classes_lost
- Performance_schema_file_handles_lost
- Performance_schema_file_instances_lost
- Performance_schema_hosts_lost
- performance_schema_hosts_size
- Performance_schema_locker_lost
- Performance_schema_index_stat_lost
- performance_schema_max_cond_classes
- performance_schema_max_cond_instances
- performance_schema_max_digest_length
- performance_schema_max_file_classes
- performance_schema_max_file_handles
- performance_schema_max_file_instances
- performance_schema_max_index_stat
- performance_schema_max_memory_classes
- performance_schema_max_metadata_locks
- performance_schema_max_mutex_classes
- performance_schema_max_mutex_instances
- performance_schema_max_prepared_statement_instances
- performance_schema_max_program_instances
- performance_schema_max_sql_text_length
- performance_schema_max_rwlock_classes
- performance_schema_max_rwlock_instances
- performance_schema_max_socket_classes
- performance_schema_max_socket_instances
- performance_schema_max_stage_classes
- performance_schema_max_statement_classes
- performance_schema_max_statement_stack
- performance_schema_max_table_handles
- performance_schema_max_table_instances
- performance_schema_max_table_lock_stat
- performance_schema_max_thread_classes
- performance_schema_max_thread_instances
- Performance_schema_memory_classes_lost
- Performance_schema_metadata_lock_lost
- Performance_schema_mutex_classes_lost
- Performance_schema_mutex_instances_lost
- Performance_schema_nested_statement_lost
- Performance_schema_prepared_statements_lost
- Performance_schema_program_lost
- Performance_schema_rwlock_classes_lost

- Performance_schema_rwlock_instances_lost
- Performance_schema_session_connect_attrs_lost
- Performance_schema_socket_classes_lost
- Performance_schema_socket_instances_lost
- Performance_schema_stage_classes_lost
- Performance_schema_stage_classes_lost
- Performance_schema_statement_classes_lost
- performance_schema_session_connect_attrs_size
- performance_schema_setup_actors_size
- performance_schema_setup_objects_size
- Performance_schema_table_handles_lost
- Performance_schema_table_instances_lost
- Performance_schema_table_lock_stat_lost
- Performance_schema_thread_classes_lost
- Performance_schema_thread_instances_lost
- Performance_schema_schema_users_lost
- performance_schema_users_size
- --pid-file, pid_file
- --plugin-load
- --plugin-load-add
- plugin_dir
- plugin_maturity
- -P, –port, port
- --port-open-timeout
- preload_buffer_size
- Prepared_stmt_count
- profiling
- profiling_history_size
- progress_report_time
- protocol_version
- proxy_protocol_networks
- proxy_user
- pseudo_slave_mode
- pseudo_thread_id
- Qcache_free_blocks
- Qcache_free_memory
- Qcache_hits
- Qcache_inserts
- Qcache_lowmem_prunes
- Qcache_not_cached
- Qcache_queries_in_cache
- Qcache_total_blocks
- Queries
- query_alloc_block_size
- --query-cache-info

- query_cache_limit
- query_cache_min_res_unit
- query_cache_size
- query_cache_strip_comments
- query_cache_type
- query_cache_wlock_invalidate
- query_prealloc_size
- --query-response-time
- --query-response-time-audit
- query_response_time_flush
- query_response_time_range_base
- query_response_time_exec_time_debug
- query_response_time_session_stats
- query_response_time_stats
- Questions
- -r, –chroot
- rand_seed1
- rand_seed2
- range_alloc_block_size
- read_buffer_size
- read_binlog_speed_limit
- read_only
- read_rnd_buffer_size
- --record-buffer
- redirect_url
- relay_log
- relay_log_basename
- relay_log_index
- relay_log_info_file
- relay_log_purge
- relay_log_recovery
- relay_log_space_limit
- --remove
- replicate_annotate_row_events
- replicate_do_db
- replicate_do_table
- replicate_events_marked_for_skip
- replicate_ignore_db
- replicate_ignore_table
- replicate_rewrite_db
- replicate-same-server-id
- replicate_wild_do_table
- replicate_wild_ignore_table
- report_host
- report_password

- report_port
- report_user
- require_secure_transport
- rocksdb_access_hint_on_compaction_start
- rocksdb_advise_random_on_open
- rocksdb_allow_concurrent_memtable_write
- rocksdb_allow_mmap_reads
- rocksdb_allow_mmap_writes
- rocksdb_allow_to_start_after_corruption
- rocksdb_background_sync
- rocksdb_base_background_compactions
- rocksdb_blind_delete_primary_key
- rocksdb_block_cache_size
- rocksdb_block_restart_interval
- Rocksdb_block_cache_add
- Rocksdb_block_cache_add_failures
- Rocksdb_block_bytes_read
- Rocksdb_block_bytes_write
- Rocksdb_block_cache_data_add
- Rocksdb_block_cache_data_bytes_insert
- Rocksdb_block_cache_data_hit
- Rocksdb_block_cache_data_miss
- Rocksdb_block_cache_filter_add
- Rocksdb_block_cache_filter_bytes_evict
- Rocksdb_block_cache_filter_bytes_insert
- Rocksdb_block_cache_filter_hit
- Rocksdb_block_cache_filter_miss
- Rocksdb_block_cache_hit
- Rocksdb_block_cache_index_add
- Rocksdb_block_cache_index_bytes_evict
- Rocksdb_block_cache_index_bytes_insert
- Rocksdb_block_cache_index_hit
- Rocksdb_block_cache_index_miss
- Rocksdb_block_cache_miss
- Rocksdb_block_cachecompressed_hit
- Rocksdb_block_cachecompressed_miss
- rocksdb_block_size
- rocksdb_block_size_deviation
- Rocksdb_bloom_filter_full_positive
- Rocksdb_bloom_filter_full_true_positive
- Rocksdb_bloom_filter_prefix_checked
- Rocksdb_bloom_filter_prefix_useful
- Rocksdb_bloom_filter_useful
- rocksdb_bulk_load
- rocksdb_bulk_load_allow_sk

- rocksdb_bulk_load_allow_unsorted
- rocksdb_bulk_load_size
- rocksdb_bytes_per_sync
- Rocksdb_bytes_read
- Rocksdb_bytes_written
- rocksdb_cache_dump
- rocksdb_cache_high_pri_pool_ratio
- rocksdb_cache_index_and_filter_blocks
- rocksdb_cache_index_and_filter_with_high_priority
- rocksdb_checksums_pct
- rocksdb_collect_sst_properties
- rocksdb_commit_in_the_middle
- rocksdb_commit_time_batch_for_recovery
- rocksdb_compact_cf
- Rocksdb_compact_read_bytes
- Rocksdb_compact_write_bytes
- Rocksdb_compaction_key_drop_new
- Rocksdb_compaction_key_drop_obsolete
- Rocksdb_compaction_key_drop_user
- rocksdb_compaction_readahead_size
- rocksdb_compaction_sequential_deletes
- rocksdb_compaction_sequential_deletes_count_sd
- rocksdb_concurrent_prepare
- rocksdb_compaction_sequential_deletes_file_size
- rocksdb_compaction_sequential_deletes_window
- Rocksdb_covered_secondary_key_lookups
- rocksdb_create_checkpoint
- rocksdb_create_if_missing
- rocksdb_create_missing_column_families
- rocksdb_datadir
- rocksdb_db_write_buffer_size
- rocksdb_deadlock_detect
- rocksdb_deadlock_detect_depth
- rocksdb_debug_manual_compaction_delay
- rocksdb_debug_optimizer_no_zero_cardinality
- rocksdb_debug_ttl_ignore_pk
- rocksdb_debug_ttl_read_filter_ts
- rocksdb_debug_ttl_rec_ts
- rocksdb_debug_ttl_snapshot_ts
- rocksdb_default_cf_options
- rocksdb_delayed_write_rate
- rocksdb_delete_cf
- rocksdb_delete_obsolete_files_period_micros
- rocksdb_enable_2pc
- rocksdb_enable_bulk_load_api

- rocksdb_enable_insert_with_update_caching
- rocksdb_enable_thread_tracking
- rocksdb_enable_ttl
- rocksdb_enable_ttl_read_filtering
- rocksdb_enable_write_thread_adaptive_yield
- rocksdb_error_if_exists
- rocksdb_on_suboptimal_collation
- rocksdb_flush_log_at_trx_commit
- rocksdb_flush_memtable_on_analyze
- Rocksdb_flush_write_bytes
- rocksdb_force_compute_memtable_stats
- rocksdb_force_compute_memtable_stats_cachetime
- rocksdb_force_flush_memtable_and_lzero_now
- rocksdb_force_flush_memtable_now
- rocksdb_force_index_records_in_range
- Rocksdb_get_hit_l0
- Rocksdb_get_hit_l1
- Rocksdb_get_hit_l2_and_up
- Rocksdb_getupdatessince_calls
- rocksdb_git_hash
- rocksdb_hash_index_allow_collision
- rocksdb_ignore_unknown_options
- rocksdb_index_type
- rocksdb_info_log_level
- rocksdb_io_write_timeout
- rocksdb_is_fd_close_on_exec
- Rocksdb_iter_bytes_read
- rocksdb_keep_log_file_num
- Rocksdb_l0_num_files_stall_micros
- Rocksdb_l0_slowdown_micros
- rocksdb_large_prefix
- rocksdb_lock_scanned_rows
- rocksdb_lock_wait_timeout
- rocksdb_log_dir
- rocksdb_log_file_time_to_roll
- rocksdb_manifest_preallocation_size
- rocksdb_manual_compaction_threads
- Rocksdb_manual_compactions_processed
- Rocksdb_manual_compactions_running
- rocksdb_manual_wal_flush
- rocksdb_master_skip_tx_api
- rocksdb_max_background_compactions
- rocksdb_max_latest_deadlocks
- rocksdb_max_background_flushes
- rocksdb_max_background_jobs

- rocksdb_max_log_file_size
- rocksdb_max_manifest_file_size
- rocksdb_max_manual_compactions
- rocksdb_max_open_files
- rocksdb_max_row_locks
- rocksdb_max_subcompactions
- rocksdb_max_total_wal_size
- Rocksdb_memtable_compaction_micros
- Rocksdb_memtable_hit
- Rocksdb_memtable_miss
- Rocksdb_memtable_total
- Rocksdb_memtable_unflushed
- rocksdb_merge_buf_size
- rocksdb_merge_combine_read_size
- rocksdb_merge_tmp_file_removal_delay_ms
- rocksdb_new_table_reader_for_compaction_inputs
- rocksdb_no_block_cache
- Rocksdb_no_file_closes
- Rocksdb_no_file_errors
- Rocksdb_no_file_opens
- Rocksdb_num_iterators
- Rocksdb_number_block_not_compressed
- Rocksdb_db_next
- Rocksdb_db_next_found
- Rocksdb_db_prev
- Rocksdb_db_prev_found
- Rocksdb_db_seek
- Rocksdb_db_seek_found
- Rocksdb_number_deletes_filtered
- Rocksdb_number_keys_read
- Rocksdb_number_keys_updated
- Rocksdb_number_keys_written
- Rocksdb_number_merge_failures
- Rocksdb_number_multiget_bytes_read
- Rocksdb_number_multiget_get
- Rocksdb_number_multiget_keys_read
- Rocksdb_number_reseeks_iteration
- Rocksdb_number_sst_entry_delete
- Rocksdb_number_sst_entry_merge
- Rocksdb_number_sst_entry_other
- Rocksdb_number_sst_entry_put
- Rocksdb_number_sst_entry_singledelete
- Rocksdb_number_superversion_acquires
- Rocksdb_number_superversion_cleanups
- Rocksdb_number_superversion_releases

- rocksdb_override_cf_options
- rocksdb_paranoid_checks
- rocksdb_pause_background_work
- rocksdb_perf_context_level
- rocksdb_persistent_cache_path
- rocksdb_persistent_cache_size_mb
- rocksdb_pin_l0_filter_and_index_blocks_in_cache
- rocksdb_print_snapshot_conflict_queries
- Rocksdb_queries_point
- Rocksdb_queries_range
- rocksdb_rate_limiter_bytes_per_sec
- rocksdb_read_free_rpl_tables
- rocksdb_records_in_range
- rocksdb_remove_mariadb-backup_checkpoint
- rocksdb_reset_stats
- rocksdb_rollback_on_timeout
- Rocksdb_row_lock_deadlocks
- Rocksdb_row_lock_wait_timeouts
- Rocksdb_rows_deleted
- Rocksdb_rows_deleted_blind
- Rocksdb_rows_expired
- Rocksdb_rows_filtered
- Rocksdb_rows_inserted
- Rocksdb_rows_read
- Rocksdb_rows_updated
- rocksdb_seconds_between_stat_computes
- rocksdb_signal_drop_index_thread
- rocksdb_sim_cache_size
- rocksdb_skip_bloom_filter_on_read
- rocksdb_skip_fill_cache
- rocksdb_skip_unique_check_tables
- Rocksdb_snapshot_conflict_errors
- rocksdb_sst_mgr_rate_bytes_per_sec
- Rocksdb_stall_l0_file_count_limit_slowdowns
- Rocksdb_stall_l0_file_count_limit_stops
- Rocksdb_stall_locked_l0_file_count_limit_slowdowns
- Rocksdb_stall_locked_l0_file_count_limit_stops
- Rocksdb_stall_memtable_limit_slowdowns
- Rocksdb_stall_memtable_limit_stops
- Rocksdb_stall_micros
- Rocksdb_stall_pending_compaction_limit_slowdowns
- Rocksdb_stall_pending_compaction_limit_stops
- Rocksdb_stall_total_slowdowns
- Rocksdb_stall_total_stops
- rocksdb_stats_dump_period_sec

- rocksdb_stats_level
- rocksdb_stats_recalc_rate
- rocksdb_store_row_debug_checksums
- rocksdb_strict_collation_check
- rocksdb_strict_collation_exceptions
- rocksdb_supported_compression_types
- Rocksdb_system_rows_deleted
- Rocksdb_system_rows_inserted
- Rocksdb_system_rows_read
- Rocksdb_system_rows_updated
- rocksdb_table_cache_numshardbits
- rocksdb_table_stats_sampling_pct
- rocksdb_tmpdir
- rocksdb_trace_sst_api
- rocksdb_two_write_queues
- rocksdb_unsafe_for_binlog
- rocksdb_update_cf_options
- rocksdb_use_adaptive_mutex
- rocksdb_use_clock_cache
- rocksdb_use_direct_io_for_flush_and_compaction
- rocksdb_use_direct_reads
- rocksdb_use_fsync
- rocksdb_validate_tables
- rocksdb_verify_row_debug_checksums
- Rocksdb_wal_bytes
- rocksdb_wal_bytes_per_sync
- rocksdb_wal_dir
- Rocksdb_wal_group_syncs
- rocksdb_wal_recovery_mode
- rocksdb_wal_size_limit_mb
- Rocksdb_wal_synced
- rocksdb_wal_ttl_seconds
- rocksdb_whole_key_filtering
- rocksdb_write_batch_max_bytes
- rocksdb_write_disable_wal
- rocksdb_write_ignore_missing_column_families
- rocksdb_write_policy
- Rocksdb_write_other
- Rocksdb_write_self
- Rocksdb_write_timedout
- Rocksdb_write_wal
- rowid_merge_buff_size
- Resultset_metadata_skipped
- Rows_read
- Rows_sent

- Rows_tmp_read
- rpl_recovery_rank
- Rpl_semi_sync_master_clients
- rpl_semi_sync_master_enabled
- Rpl_semi_sync_master_net_avg_wait_time
- Rpl_semi_sync_master_net_wait_time
- Rpl_semi_sync_master_net_waits
- Rpl_semi_sync_master_no_times
- Rpl_semi_sync_master_no_tx
- Rpl_semi_sync_master_status
- Rpl_semi_sync_master_timefunc_failures
- rpl_semi_sync_master_timeout
- rpl_semi_sync_master_trace_level
- Rpl_semi_sync_master_tx_avg_wait_time
- Rpl_semi_sync_master_tx_wait_time
- Rpl_semi_sync_master_tx_waits
- rpl_semi_sync_master_wait_no_slave
- rpl_semi_sync_master_wait_point
- Rpl_semi_sync_master_wait_pos_backtraverse
- Rpl_semi_sync_master_wait_sessions
- Rpl_semi_sync_master_yes_tx
- rpl_semi_sync_slave_delay_master
- rpl_semi_sync_slave_enabled
- rpl_semi_sync_slave_kill_conn_timeout
- Rpl_semi_sync_slave_status
- rpl_semi_sync_slave_trace_level
- Rpl_status
- Rpl_transactions_multi_engine
- -s, –symbolic-links
- s3_access_key
- s3_block_size
- s3_bucket
- s3_debug
- s3_host_name
- s3_no_content_type
- s3_pagecache_age_threshold
- s3_pagecache_buffer_size
- s3_pagecache_division_limit
- s3_pagecache_file_hash_size
- s3_port
- s3_protocol_version
- s3_provider
- s3_region
- s3_replicate_alter_as_create_select
- s3_secret_key

- s3_slave_ignore_updates
- s3_ssl_no_verify
- s3_use_http
- --safe-mode
- safe_show_database
- --safe-user-create
- --safemalloc-mem-limit
- secure_auth
- secure_file_priv
- secure_timestamp
- Select_full_join
- Select_full_range_join
- Select_range
- Select_range_check
- Select_scan
- --server-audit
- Server_audit_active
- Server_audit_current_log
- server_audit_events
- server_audit_excl_users
- server_audit_file_path
- Server_audit_last_error
- server_audit_file_rotate_now
- server_audit_file_rotate_size
- server_audit_file_rotations
- server_audit_incl_users
- server_audit_loc_info
- server_audit_logging
- server_audit_mode
- server_audit_output_type
- server_audit_query_limit
- server_audit_syslog_facility
- server_audit_syslog_ident
- server_audit_syslog_info
- server_audit_syslog_priority
- Server_audit_writes_failed
- server_id
- server_uid
- session_track_schema
- session_track_state_change
- session_track_system_variables
- session_track_transaction_info
- -O, –set-variable
- shared_memory
- shared_memory_base_name

- show_old_temporals
- show_slave_auth_info
- --silent-startup
- simple_password_check_digits
- simple_password_check_letters_same_case
- simple_password_check_minimal_length
- simple_password_check_other_characters
- --skip-automatic-sp-privileges
- --skip-bdb
- skip_external_locking
- --skip-grant-tables, skip_grant_tables
- --skip-host-cache
- --skip-innodb
- --skip-innodb-checksums
- --skip-innodb-doublewrite
- --skip-large-pages
- --skip-log-error
- --skip-name-resolve, skip_name_resolve
- --skip-new
- skip_networking
- skip_parallel_replication
- --skip-partition
- skip_replication
- --skip-show-database, skip_show_database
- --skip-slave-start
- --skip-ssl
- --skip-stack-trace
- --skip-symbolic-links
- --skip-symlink
- --skip-thread-priority
- slave_abort_blocking_timeout
- slave_compressed_protocol
- slave_connections_needed_for_purge
- Slave_connections
- slave_ddl_exec_mode
- slave_domain_parallel_threads
- slave_exec_mode
- Slave_heartbeat_period
- slave_load_tmpdir
- slave_max_allowed_packet
- slave_max_statement_time
- slave_net_timeout
- Slave_open_temp_tables
- slave_parallel_max_queued
- slave_parallel_mode

- slave_parallel_threads
- slave_parallel_workers
- Slave_received_heartbeats
- Slave_retried_transactions
- slave_run_triggers_for_rbr
- Slave_running
- slave_skip_errors
- Slave_skipped_errors
- slave_sql_verify_checksum
- slave_transaction_retries
- slave_transaction_retry_errors
- slave_transaction_retry_interval
- slave_type_conversions
- Slaves_connected
- Slaves_running
- Slow_launch_threads
- slow_launch_time
- Slow_queries
- slow_query_log
- slow_query_log_file
- --slow-start-timeout
- socket
- sort_buffer_size
- Sort_merge_passes
- Sort_priority_queue_sorts
- Sort_range
- Sort_rows
- Sort_scan
- Sphinx_error
- Sphinx_time
- Sphinx_total
- Sphinx_total_found
- Sphinx_word_count
- Sphinx_words
- spider_auto_increment_mode
- spider_bgs_first_read
- spider_bgs_mode
- spider_bgs_second_read
- spider_bka_engine
- spider_bka_mode
- spider_block_size
- spider_buffer_size
- spider_bulk_size
- spider_bulk_update_mode
- spider_bulk_update_size

- spider_casual_read
- spider_conn_recycle_mode
- spider_conn_recycle_strict
- spider_conn_wait_timeout
- spider_connect_error_interval
- spider_connect_mutex
- spider_connect_retry_count
- spider_connect_retry_interval
- spider_connect_timeout
- spider_crd_bg_mode
- spider_crd_interval
- spider_crd_mode
- spider_crd_sync
- spider_crd_type
- spider_crd_weight
- spider_delete_all_rows_type
- Spider_direct_aggregate
- Spider_direct_delete
- spider_direct_dup_insert
- spider_direct_order_limit (system variable)
- Spider_direct_order_limit (status variable)
- Spider_direct_update
- spider_dry_access
- spider_error_read_mode
- spider_error_write_mode
- spider_first_read
- spider_force_commit
- spider_general_log
- spider_ignore_comments
- spider_index_hint_pushdown
- spider_init_sql_alloc_size
- spider_internal_limit
- spider_internal_offset
- spider_internal_optimize
- spider_internal_optimize_local
- spider_internal_sql_log_off
- spider_internal_unlock
- spider_internal_xa
- spider_internal_xa_id_type
- spider_internal_xa_snapshot
- spider_load_crd_at_startup
- spider_load_sts_at_startup
- spider_local_lock_table
- spider_lock_exchange
- spider_log_result_error_with_sql

- spider_log_result_errors
- spider_low_mem_read
- spider_max_connections
- spider_max_order
- Spider_mon_table_cache_version
- Spider_mon_table_cache_version_req
- spider_multi_split_read
- spider_net_read_timeout
- spider_net_write_timeout
- Spider_parallel_search
- spider_ping_interval_at_trx_start
- spider_quick_mode
- spider_quick_page_byte
- spider_quick_page_size
- spider_read_only_mode
- spider_remote_access_charset
- spider_remote_autocommit
- spider_remote_default_database
- spider_remote_sql_log_off
- spider_remote_time_zone
- spider_remote_trx_isolation
- spider_remote_wait_timeout
- spider_reset_sql_alloc
- spider_same_server_link
- spider_second_read
- spider_select_column_mode
- spider_selupd_lock_mode
- spider_semi_split_read
- spider_semi_split_read_limit
- spider_semi_table_lock
- spider_semi_table_lock_connection
- spider_semi_trx
- spider_semi_trx_isolation
- spider_skip_default_condition
- spider_skip_parallel_search
- spider_slave_trx_isolation
- spider_split_read
- spider_store_last_crd
- spider_store_last_sts
- spider_strict_group_by
- spider_sts_bg_mode
- spider_sts_interval
- spider_sts_mode
- spider_sts_sync
- spider_support_xa

- spider_sync_autocommit
- spider_sync_sql_mode
- spider_sync_time_zone
- spider_sync_trx_isolation
- spider_table_crd_thread_count
- spider_table_init_error_interval
- spider_table_sts_thread_count
- spider_udf_ct_bulk_insert_interval
- spider_udf_ct_bulk_insert_rows
- spider_udf_ds_bulk_insert_rows
- spider_udf_ds_table_loop_mode
- spider_udf_ds_use_real_table
- spider_udf_table_lock_mutex_count
- spider_udf_table_mon_mutex_count
- spider_use_all_conns_snapshot
- spider_use_cond_other_than_pk_for_update
- spider_use_consistent_snapshot
- spider_use_default_database
- spider_use_flash_logs
- spider_use_handler
- spider_use_pushdown_udf
- spider_use_table_charset
- spider_version
- spider_wait_timeout
- spider_xa_register_mode
- --sporadic-binlog-dump-fail
- sql_auto_is_null
- sql_big_selects
- sql_big_tables
- sql_buffer_result
- --sql-bin-update-same
- sql_error_log_filename
- sql_error_log_rate
- sql_error_log_rotate
- sql_error_log_rotations
- sql_error_log_size_limit
- sql_error_log_warnings
- --sql-if-exists, sql_if_exists
- sql_log_bin
- sql_log_off
- sql_log_update
- sql_low_priority_updates
- sql_max_join_size
- sql_mode
- sql_notes

- sql_quote_show_create
- sql_safe_updates
- sql_select_limit
- sql_slave_skip_counter
- sql_warnings
- --ssl
- Ssl_accept_renegotiates
- Ssl_accepts
- ssl_ca
- Ssl_callback_cache_hits
- ssl_capath
- ssl_cert
- ssl_cipher
- Ssl_cipher
- Ssl_cipher_list
- Ssl_client_connects
- Ssl_connect_renegotiates
- ssl_crl
- ssl_crlpath
- Ssl_ctx_verify_depth
- Ssl_ctx_verify_mode
- Ssl_default_timeout
- Ssl_finished_accepts
- Ssl_finished_connects
- ssl_key
- ssl_passphrase
- Ssl_server_not_after
- Ssl_server_not_before
- Ssl_session_cache_hits
- Ssl_session_cache_misses
- Ssl_session_cache_mode
- Ssl_session_cache_overflows
- Ssl_session_cache_size
- Ssl_session_cache_timeouts
- Ssl_sessions_reused
- Ssl_used_session_cache_entries
- Ssl_verify_depth
- Ssl_verify_mode
- standard_compliant_cte
- --stack-trace
- --standalone
- storage_engine
- stored_program_cache
- strict_password_validation
- Subquery_cache_hit

- Subquery_cache_miss
- -s, –symbolic-links
- sync_binlog
- sync_frm
- sync_master_info
- sync_relay_log
- sync_relay_log_info
- --sync-sys
- Syncs
- --sysdate-is-now
- -T, –exit-info
- system_time_zone
- system_versioning_alter_history
- system_versioning_asof
- system_versioning_innodb_algorithm_simple
- system_versioning_insert_history
- --table-cache
- table_definition_cache
- table_lock_wait_timeout
- Table_locks_immediate
- Table_locks_waited
- table_open_cache
- Table_open_cache_active_instances
- Table_open_cache_hits
- table_open_cache_instances
- Table_open_cache_misses
- Table_open_cache_overflows
- table_type
- --tc-heuristic-recover
- Tc_log_max_pages_used
- Tc_log_page_size
- Tc_log_page_waits
- tcp_keepalive_interval
- tcp_keepalive_probes
- tcp_keepalive_time
- tcp_nodelay
- --temp-pool
- --test-expect-abort
- --test-ignore-wrong-options
- --thread-alarm
- thread_cache_size
- thread_concurrency
- thread_handling
- thread_pool_dedicated_listener
- thread_pool_exact_stats

- thread_pool_idle_timeout
- thread_pool_max_threads
- thread_pool_min_threads
- thread_pool_oversubscribe
- thread_pool_prio_kickup_timer
- thread_pool_priority
- thread_pool_size
- thread_pool_stall_limit
- Threadpool_idle_threads
- Threadpool_threads
- thread_stack
- Threads_cached
- Threads_connected
- Threads_created
- Threads_running
- timed_mutexes
- timestamp
- time-format
- time_zone
- tls_version
- tmp_disk_table_size
- tmp_memory_table_size
- Tmp_space_used
- tmp_table_size
- -t, tmpdir
- transaction_alloc_block_size
- transaction_isolation
- transaction_prealloc_size
- --transaction-read-only
- Transactions_gtid_foreign_engine
- Transactions_multi_engine
- tx_isolation
- tx_read_only
- -u, –user
- unique_checks
- updatable_views_with_limit
- Update_scan
- Uptime
- Uptime_since_flush_status
- -u, –user
- use_stat_tables
- --userstat, userstat
- -v, –verbose
- -V, version
- version_comment

- version_compile_machine
- version_compile_os
- version_malloc_library
- version_source_revision
- version_ssl_library
- -W, log_warnings
- wait_timeout
- warning_count
- wsrep
- wsrep_allowlist
- wsrep_applier_retry_count
- wsrep_applier_thread_count
- wsrep_apply_oooe
- wsrep_apply_oool
- wsrep_auto_increment_control
- wsrep_causal_reads
- wsrep_cert_deps_distance
- wsrep_certification_rules
- wsrep_certify_nonPK
- wsrep_cluster_address
- wsrep_cluster_capabilities
- wsrep_cluster_conf_id
- wsrep_cluster_name
- wsrep_cluster_size
- wsrep_cluster_state_uuid
- wsrep_cluster_status
- wsrep_connected
- wsrep_convert_LOCK_to_trx
- wsrep_data_home_dir
- wsrep_dbug_option
- wsrep_debug
- wsrep_desync
- wsrep_dirty_reads
- wsrep_drupal_282555_workaround
- wsrep_flow_control_paused
- wsrep_flow_control_recv
- wsrep_flow_control_sent
- wsrep_gtid_domain_id
- wsrep_gtid_mode
- wsrep_gtid_seq_no
- wsrep_forced_binlog_format
- wsrep_ignore_apply_errors
- wsrep_last_committed
- wsrep_load_data_splitting
- wsrep_local_bf_aborts

- wsrep_local_cert_failures
- wsrep_local_commits
- wsrep_local_index
- wsrep_local_recv_queue
- wsrep_local_recv_queue_avg
- wsrep_local_replays
- wsrep_local_send_queue
- wsrep_local_send_queue_avg
- wsrep_local_state
- wsrep_local_state_comment
- wsrep_local_state_uuid
- wsrep_log_conflicts
- wsrep_max_ws_rows
- wsrep_max_ws_size
- wsrep_mode
- wsrep_mysql_replication_bundle
- --wsrep-new-cluster
- wsrep_node_address
- wsrep_node_incoming_address
- wsrep_node_name
- wsrep_notify_cmd
- wsrep_on
- wsrep_OSU_method
- wsrep_protocol_version
- wsrep_provider_name
- wsrep_patch_version
- wsrep_provider
- wsrep_provider_options
- wsrep_provider_vendor
- wsrep_provider_version
- wsrep_ready
- wsrep_received
- wsrep_received_bytes
- wsrep_recover
- wsrep_reject_queries
- wsrep_replicate_myisam
- wsrep_replicated
- wsrep_replicated_bytes
- wsrep_retry_autocommit
- wsrep_rollbacker_thread_count
- wsrep_slave_FK_checks
- wsrep_slave_threads
- wsrep_slave_UK_checks
- wsrep_sr_store
- wsrep_sst_auth

- wsrep_sst_donor
- wsrep_sst_donor_rejects_queries
- wsrep_sst_method
- wsrep_sst_receive_address
- wsrep_start_position
- wsrep_status_file
- wsrep_strict_ddl
- wsrep_sync_wait
- wsrep-trx-fragment-size
- wsrep-trx-fragment-unit
- wsrep_thread_count

See Also

- System and Status Variables Added By Major Release

This page is licensed: CC BY-SA / Gnu FDL

Server Status Variables

> **Note:**
>
> **For a full list of server options, system variables and status variables, see this page.**

Most status variables are described on this page, but some are described elsewhere:

- Aria Status Variables
- Galera Status Variables
- InnoDB Status Variables
- Mroonga Status Variables
- MyRocks Status Variables
- Performance Scheme Status Variables
- Replication and Binary Log Status Variables
- S3 Storage Engine Status Variables
- Server_Audit Status Variables
- Sphinx Status Variables
- Spider Status Variables

Use the SHOW STATUS statement to view status variables. This information also can be obtained using the mariadb-admin extended-status command, or by querying the Information Schema GLOBAL_STATUS and SESSION_STATUS tables.

Issuing a FLUSH STATUS will reset many status variables to zero.

List of Server Status Variables

`Aborted_clients`

- Description: Number of aborted client connections. This can be due to the client not calling mysql_close() before exiting, the client sleeping without issuing a request to the server for more seconds than specified by wait_timeout or interactive_timeout, or by the client program ending in the midst of transferring data. The global value can be flushed by FLUSH STATUS.

- Scope: Global
- Data Type: `numeric`

`Aborted_connects`

- Description: Number of failed server connection attempts. This can be due to a client using an incorrect password, a client not having privileges to connect to a database, a connection packet not containing the correct information, or if it takes more than connect_timeout seconds to get a connect packet. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Aborted_connects_preauth`

- Description: Number of connection attempts that were aborted prior to authentication (regardless of whether or not an error occurred).
- Scope: Global
- Data Type: `numeric`

`Access_denied_errors`

- Description: Number of access denied errors. For details on when this is incremented, see Incrementing of the access_denied_errors status variable.
- Scope: Global
- Data Type: `numeric`

`Acl_column_grants`

- Description: Number of column permissions granted (rows in the mysql.columns_priv table).
- Scope: Global
- Data Type: `numeric`

`Acl_database_grants`

- Description: Number of database permissions granted (rows in the mysql.db table).
- Scope: Global
- Data Type: `numeric`

`Acl_function_grants`

- Description: Number of function permissions granted (rows in the mysql.procs_priv table with a routine type of `FUNCTION`).
- Scope: Global
- Data Type: `numeric`

`Acl_package_body_grants`

- Description:
- Scope: Global
- Data Type: `numeric`

`Acl_package_spec_grants`

- Description:

- Scope: Global
- Data Type: `numeric`

`Acl_procedure_grants`

- Description: Number of procedure permissions granted (rows in the mysql.procs_priv table with a routine type of `PROCEDURE`).
- Scope: Global
- Data Type: `numeric`

`Acl_proxy_users`

- Description: Number of proxy permissions granted (rows in the mysql.proxies_priv table).
- Scope: Global
- Data Type: `numeric`

`Acl_role_grants`

- Description: Number of role permissions granted (rows in the mysql.roles_mapping table).
- Scope: Global
- Data Type: `numeric`

`Acl_roles`

- Description: Number of roles (rows in the mysql.user table where `is_role='Y'`).
- Scope: Global
- Data Type: `numeric`

`Acl_table_grants`

- Description: Number of table permissions granted (rows in the mysql.tables_priv table).
- Scope: Global
- Data Type: `numeric`

`Acl_users`

- Description: Number of users (rows in the mysql.user table where `is_role='N'`).
- Scope: Global
- Data Type: `numeric`

`Busy_time`

- Description: Cumulative time in seconds of activity on connections. Part of User Statistics. Requires the userstat system variable to be set in order to be recorded.
- Scope: Global
- Data Type: `numeric`

`Bytes_received`

- Description: Total bytes received from all clients.
- Scope: Global
- Data Type: `numeric`

`Bytes_sent`

- Description: Total bytes sent to all clients.
- Scope: Global, Session
- Data Type: `numeric`

`Com_admin_commands`

- Description: Number of admin commands executed. These include table dumps, change users, binary log dumps, shutdowns, pings and debugs.
- Scope: Global, Session
- Data Type: `numeric`

`Com_alter_db`

- Description: Number of ALTER DATABASE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_alter_db_upgrade`

- Description: Number of ALTER DATABASE … UPGRADE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_alter_event`

- Description: Number of ALTER EVENT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_alter_function`

- Description: Number of ALTER FUNCTION commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_alter_procedure`

- Description: Number of ALTER PROCEDURE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_alter_sequence`

- Description: Number of ALTER SEQUENCE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_alter_server`

- Description: Number of ALTER SERVER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_alter_table`

- Description: Number of ALTER TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_alter_tablespace`

- Description: Number of ALTER TABLESPACE commands executed (unsupported by MariaDB).
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.7.0

`Com_alter_user`

- Description: Number of ALTER USER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_analyze`

- Description: Number of ANALYZE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_assign_to_keycache`

- Description: Number of assign to keycache commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_backup`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.4.1

`Com_backup_lock`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.4.2

`Com_backup_table`

- Description: Removed in MariaDB 5.5. In older versions, Com_backup_table contains the number of BACKUP TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 5.5

`Com_begin`

- Description: Number of BEGIN or START TRANSACTION statements executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_binlog`

- Description: Number of BINLOG commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_call_procedure`

- Description: Number of CALL procedure_name statements executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_change_db`

- Description: Number of USE database_name commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_check`

- Description: Number of CHECK TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_checksum`

- Description: Number of CHECKSUM TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_commit`

- Description: Number of COMMIT commands executed. Differs from Handler_commit, which counts internal commit statements.
- Scope: Global, Session
- Data Type: `numeric`

`Com_compound_sql`

- Description: Number of compound sql statements.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_db`

- Description: Number of CREATE DATABASE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_event`

- Description: Number of CREATE EVENT commands executed. Differs from Executed_events in that it is incremented when the CREATE EVENT is run, and not when the event executes.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_function`

- Description: Number of CREATE FUNCTION commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_index`

- Description: Number of CREATE INDEX commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_package`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_package_body`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_procedure`

- Description: Number of CREATE PROCEDURE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_role`

- Description: Number of CREATE ROLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_sequence`

- Description: Number of CREATE SEQUENCE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_server`

- Description: Number of CREATE SERVER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_table`

- Description: Number of CREATE TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_temporary_table`

- Description: Number of CREATE TEMPORARY TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_trigger`

- Description: Number of CREATE TRIGGER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_udf`

- Description: Number of CREATE UDF commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_user`

- Description: Number of CREATE USER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_create_view`

- Description: Number of CREATE VIEW commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_dealloc_sql`

- Description: Number of DEALLOCATE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_delete`

- Description: Number of DELETE commands executed. Differs from Handler_delete, which counts the number of times rows have been deleted from tables.
- Scope: Global, Session
- Data Type: `numeric`

`Com_delete_multi`

- Description: Number of multi-table DELETE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_do`

- Description: Number of DO commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_db`

- Description: Number of DROP DATABASE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_event`

- Description: Number of DROP EVENT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_function`

- Description: Number of DROP FUNCTION commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_index`

- Description: Number of DROP INDEX commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_package`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_package_body`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_procedure`

- Description: Number of DROP PROCEDURE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_role`

- Description: Number of DROP ROLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_sequence`

- Description: Number of DROP SEQUENCE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_server`

- Description: Number of DROP SERVER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_table`

- Description: Number of DROP TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_temporary_table`

- Description: Number of DROP TEMPORARY TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_trigger`

- Description: Number of DROP TRIGGER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_user`

- Description: Number of DROP USER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_drop_view`

- Description: Number of DROP VIEW commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_empty_query`

- Description: Number of queries to the server that do not produce SQL queries. An SQL query simply returning no results does not increment `Com_empty_query` - see Empty_queries instead. An example of an empty query sent to the server is `mariadb --comments -e '-- sql comment'`
- Scope: Global, Session
- Data Type: `numeric`

`Com_execute_immediate`

- Description: Number of EXECUTE IMMEDIATE statements executed.
- Scope: Global, Session

- Data Type: `numeric`

`Com_execute_sql`

- Description: Number of EXECUTE statements executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_flush`

- Description: Number of FLUSH commands executed. This differs from Flush_commands, which also counts internal server flush requests.
- Scope: Global, Session
- Data Type: `numeric`

`Com_get_diagnostics`

- Description: Number of GET DIAGNOSTICS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_grant`

- Description: Number of GRANT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_grant_role`

- Description: Number of GRANT role commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_ha_close`

- Description: Number of HANDLER table_name CLOSE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_ha_open`

- Description: Number of HANDLER table_name OPEN commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_ha_read`

- Description: Number of HANDLER table_name READ commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_help`

- Description: Number of HELP commands executed.
- Scope: Global, Session

- Data Type: `numeric`

`Com_insert`

- Description: Number of INSERT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_insert_select`

- Description: Number of INSERT … SELECT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_install_plugin`

- Description: Number of INSTALL PLUGIN commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_kill`

- Description: Number of KILL commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_load`

- Description: Number of LOAD commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_load_master_data`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 5.5

`Com_load_master_table`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 5.5

`Com_multi`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_lock_tables`

- Description: Number of [lock-tables|LOCK TABLES]] commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_optimize`

- Description: Number of OPTIMIZE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_preload_keys`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_prepare_sql`

- Description: Number of PREPARE statements executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_purge`

- Description: Number of PURGE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_purge_before_date`

- Description: Number of PURGE BEFORE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_release_savepoint`

- Description: Number of RELEASE SAVEPOINT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_rename_table`

- Description: Number of RENAME TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_rename_user`

- Description: Number of RENAME USER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_repair`

- Description: Number of REPAIR TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_replace`

- Description: Number of REPLACE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_replace_select`

- Description: Number of REPLACE … SELECT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_reset`

- Description: Number of RESET commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_resignal`

- Description: Number of RESIGNAL commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_restore_table`

- Description: Removed in MariaDB 5.5. In older versions, Com_restore_table contains the number of RESTORE TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 5.5

`Com_revoke`

- Description: Number of REVOKE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_revoke_all`

- Description: Number of REVOKE ALL commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_revoke_grant`

- Description: Number of REVOKE role commands executed.
- Scope: Global, Session

- Data Type: `numeric`

`Com_rollback`

- Description: Number of ROLLBACK commands executed. Differs from Handler_rollback, which is the number of transaction rollback requests given to a storage engine.
- Scope: Global, Session
- Data Type: `numeric`

`Com_rollback_to_savepoint`

- Description: Number of ROLLBACK … TO SAVEPOINT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_savepoint`

- Description: Number of SAVEPOINT commands executed. Differs from Handler_savepoint, which is the number of transaction savepoint creation requests.
- Scope: Global, Session
- Data Type: `numeric`

`Com_select`

- Description: Number of SELECT commands executed. Also includes queries that make use of the query cache.
- Scope: Global, Session
- Data Type: `numeric`

`Com_set_option`

- Description: Number of SET OPTION commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_signal`

- Description: Number of SIGNAL statements executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_authors`

- Description: Number of SHOW AUTHORS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_binlog_events`

- Description: Number of SHOW BINLOG EVENTS statements executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_binlogs`

- Description: Number of SHOW BINARY LOGS commands executed.

- Scope: Global, Session
- Data Type: `numeric`

`Com_show_charsets`

- Description: Number of SHOW CHARACTER SET commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_client_statistics`

- Description: Number of SHOW CLIENT STATISTICS commands executed. Removed in MariaDB 10.1.1 when that statement was replaced by the generic SHOW information_schema_table.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.1.1

`Com_show_collations`

- Description: Number of SHOW COLLATION commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_column_types`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 5.5

`Com_show_contributors`

- Description: Number of SHOW CONTRIBUTORS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_create_db`

- Description: Number of SHOW CREATE DATABASE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_create_event`

- Description: Number of SHOW CREATE EVENT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_create_func`

- Description: Number of SHOW CREATE FUNCTION commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_create_package`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_create_package_body`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_create_proc`

- Description: Number of SHOW CREATE PROCEDURE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_create_table`

- Description: Number of SHOW CREATE TABLE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_create_trigger`

- Description: Number of SHOW CREATE TRIGGER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_create_user`

- Description: Number of SHOW CREATE USER commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_databases`

- Description: Number of SHOW DATABASES commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_engine_logs`

- Description: Number of SHOW ENGINE LOGS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_engine_mutex`

- Description: Number of SHOW ENGINE MUTEX commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_engine_status`

- Description: Number of SHOW ENGINE STATUS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_events`

- Description: Number of SHOW EVENTS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_errors`

- Description: Number of SHOW ERRORS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_explain`

- Description: Number of SHOW EXPLAIN commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_fields`

- Description: Number of SHOW COLUMNS or SHOW FIELDS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_function_status`

- Description: Number of SHOW FUNCTION STATUS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_generic`

- Description: Number of generic SHOW commands executed, such as SHOW INDEX_STATISTICS and SHOW TABLE_STATISTICS
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_grants`

- Description: Number of SHOW GRANTS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_keys`

- Description: Number of SHOW INDEX or SHOW KEYS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_index_statistics`

- Description: Number of SHOW INDEX_STATISTICS commands executed. Removed in MariaDB 10.1.1 when that statement was replaced by the generic SHOW information_schema_table.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.1.1

`Com_show_open_tables`

- Description: Number of SHOW OPEN TABLES commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_package_status`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_package_body_status`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_plugins`

- Description: Number of SHOW PLUGINS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_privileges`

- Description: Number of SHOW PRIVILEGES commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_procedure_status`

- Description: Number of SHOW PROCEDURE STATUS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_processlist`

- Description: Number of SHOW PROCESSLIST commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_profile`

- Description: Number of SHOW PROFILE commands executed.
- Scope: Global, Session

- Data Type: `numeric`

`Com_show_profiles`

- Description: Number of SHOW PROFILES commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_relaylog_events`

- Description: Number of SHOW RELAYLOG EVENTS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_status`

- Description: Number of SHOW STATUS commands executed.
- Scope: Global, Session
- Data Type: `numeric` st

`Com_show_storage_engines`

- Description: Number of SHOW STORAGE ENGINES - or `SHOW ENGINES` - commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_table_statistics`

- Description: Number of SHOW TABLE STATISTICS commands executed. Removed in MariaDB 10.1.1 when that statement was replaced by the generic SHOW information_schema_table.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.1.1

`Com_show_table_status`

- Description: Number of SHOW TABLE STATUS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_tables`

- Description: Number of SHOW TABLES commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_triggers`

- Description: Number of SHOW TRIGGERS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_user_statistics`

- Description: Number of SHOW USER STATISTICS commands executed. Removed in MariaDB 10.1.1 when that statement was replaced by the generic SHOW information_schema_table.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.1.1

`Com_show_variable`

- Description: Number of SHOW VARIABLES commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_warnings`

- Description: Number of SHOW WARNINGS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_shutdown`

- Description: Number of SHUTDOWN commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_stmt_close`

- Description: Number of prepared statements closed (deallocated or dropped).
- Scope: Global, Session
- Data Type: `numeric`

`Com_stmt_execute`

- Description: Number of prepared statements executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_stmt_fetch`

- Description: Number of prepared statements fetched.
- Scope: Global, Session
- Data Type: `numeric`

`Com_stmt_prepare`

- Description: Number of prepared statements prepared.
- Scope: Global, Session
- Data Type: `numeric`

`Com_stmt_reprepare`

- Description: Number of prepared statements reprepared.
- Scope: Global, Session

- Data Type: `numeric`

`Com_stmt_reset`

- Description: Number of prepared statements where the data of a prepared statement which was accumulated in chunks by sending long data has been reset.
- Scope: Global, Session
- Data Type: `numeric`

`Com_stmt_send_long_data`

- Description: Number of prepared statements where the parameter data has been sent in chunks (long data).
- Scope: Global, Session
- Data Type: `numeric`

`Com_truncate`

- Description: Number of TRUNCATE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_uninstall_plugin`

- Description: Number of UNINSTALL PLUGIN commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_unlock_tables`

- Description: Number of UNLOCK TABLES commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_update`

- Description: Number of UPDATE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_update_multi`

- Description: Number of multi-table UPDATE commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_xa_commit`

- Description: Number of XA statements committed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_xa_end`

- Description: Number of XA statements ended.
- Scope: Global, Session

- Data Type: `numeric`

`Com_xa_prepare`

- Description: Number of XA statements prepared.
- Scope: Global, Session
- Data Type: `numeric`

`Com_xa_recover`

- Description: Number of XA RECOVER statements executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_xa_rollback`

- Description: Number of XA statements rolled back.
- Scope: Global, Session
- Data Type: `numeric`

`Com_xa_start`

- Description: Number of XA statements started.
- Scope: Global, Session
- Data Type: `numeric`

`Compression`

- Description: Whether client-server traffic is compressed.
- Scope: Session
- Data Type: `boolean`

`Connection_errors_accept`

- Description: Number of errors that occurred during calls to accept() on the listening port. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Connection_errors_internal`

- Description: Number of refused connections due to internal server errors, for example out of memory errors, or failed thread starts. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Connection_errors_max_connections`

- Description: Number of refused connections due to the max_connections limit being reached. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Connection_errors_peer_address`

- Description: Number of errors while searching for the connecting client IP address. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Connection_errors_select`

- Description: Number of errors during calls to select() or poll() on the listening port. The client would not necessarily have been rejected in these cases. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Connection_errors_tcpwrap`

- Description: Number of connections the libwrap library refused. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Connections`

- Description: Number of connection attempts (both successful and unsuccessful)
- Scope: Global
- Data Type: `numeric`

`Cpu_time`

- Description: Total CPU time used. Part of User Statistics. Requires the userstat system variable to be set in order to be recorded.
- Scope: Global, Session
- Data Type: `numeric`

`Created_tmp_disk_tables`

- Description: Number of on-disk temporary tables created.
- Scope: Global, Session
- Data Type: `numeric`

`Created_tmp_files`

- Description: Number of temporary files created. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Created_tmp_tables`

- Description: Number of in-memory temporary tables created.
- Scope: Global
- Data Type: `numeric`

`Delayed_errors`

- Description: Number of errors which occurred while doing INSERT DELAYED. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Delayed_insert_threads`

- Description: Number of INSERT DELAYED threads.
- Scope: Global
- Data Type: `numeric`

`Delayed_writes`

- Description: Number of INSERT DELAYED rows written. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Delete_scan`

- Description: Number of DELETE s that required a full table scan.
- Scope: Global
- Data Type: `numeric`

`Empty_queries`

- Description: Number of queries returning no results. Note this is not the same as Com_empty_query.
- Scope: Global, Session
- Data Type: `numeric`

`Executed_events`

- Description: Number of times events created with CREATE EVENT have executed. This differs from Com_create_event in that it is only incremented when the event has run, not when it executes.
- Scope: Global, Session
- Data Type: `numeric`

`Executed_triggers`

- Description: Number of times triggers created with CREATE TRIGGER have executed. This differs from Com_create_trigger in that it is only incremented when the trigger has run, not when it executes.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_application_time_periods`

- Description: Number of times a table created with periods has been opened.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_check_constraint`

- Description: Number of times constraints were checked. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session

- Data Type: `numeric`

`Feature_custom_aggregate_functions`

- Description: Number of queries which make use of custom aggregate functions.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_delay_key_write`

- Description: Number of tables opened that are using delay_key_write. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_dynamic_columns`

- Description: Number of times the COLUMN_CREATE() function was used.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_fulltext`

- Description: Number of times the MATCH … AGAINST() function was used.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_gis`

- Description: Number of times a table with a any of the geometry columns was opened.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_insert_returning`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.5.0

`Feature_invisible_columns`

- Description: Number of invisible columns in all opened tables.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_json`

- Description: Number of times JSON functionality has been used, such as one of the JSON functions. Does not include the CONNECT engine JSON type, or EXPLAIN/ANALYZE FORMAT=JSON.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_locale`

- Description: Number of times the @@lc_messages variable was assigned into.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_subquery`

- Description: Number of subqueries (excluding subqueries in the FROM clause) used.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_system_versioning`

- Description: Number of times system versioning functionality has been used (opening a table WITH SYSTEM VERSIONING).
- Scope: Global, Session
- Data Type: `numeric`

`Feature_timezone`

- Description: Number of times an explicit timezone (excluding UTC and SYSTEM) was specified.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_trigger`

- Description: Number of triggers loaded.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_window_functions`

- Description: Number of times window functions were used.
- Scope: Global, Session
- Data Type: `numeric`

`Feature_xml`

- Description: Number of times XML functions (EXTRACTVALUE() and UPDATEXML()) were used.
- Scope: Global, Session
- Data Type: `numeric`

`Flush_commands`

- Description: Number of FLUSH statements executed, as well as due to internal server flush requests. This differs from Com_flush, which simply counts FLUSH statements, not internal server flush operations.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.5.1

`Handler_commit`

- Description: Number of internal COMMIT requests. Differs from Com_commit, which counts the number of COMMIT statements executed.

- Scope: Global, Session
- Data Type: `numeric`

`Handler_delete`

- Description: Number of times rows have been deleted from tables. Differs from Com_delete, which counts DELETE statements.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_discover`

- Description: Discovery is when the server asks the NDBCLUSTER storage engine if it knows about a table with a given name. Handler_discover indicates the number of times that tables have been discovered in this way.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_external_lock`

- Description: Incremented for each call to the external_lock() function, which generally occurs at the beginning and end of access to a table instance.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_icp_attempts`

- Description: Number of times pushed index condition was checked. The smaller the ratio of Handler_icp_attempts to Handler_icp_match the better the filtering. See Index Condition Pushdown.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_icp_match`

- Description: Number of times pushed index condition was matched. The smaller the ratio of Handler_icp_attempts to Handler_icp_match the better the filtering. See Index Condition Pushdown.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_mrr_init`

- Description: Counts how many MRR (multi-range read) scans were performed. See Multi Range Read optimization.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_mrr_key_refills`

- Description: Number of times key buffer was refilled (not counting the initial fill). A non-zero value indicates there wasn't enough memory to do key sort-and-sweep passes in one go. See Multi Range Read optimization.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_mrr_rowid_refills`

- Description: Number of times rowid buffer was refilled (not counting the initial fill). A non-zero value indicates there wasn't enough memory to do rowid sort-and-sweep passes in one go. See Multi Range Read optimization.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_prepare`

- Description: Number of two-phase commit prepares.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_read_first`

- Description: Number of requests to read the first row from an index. A high value indicates many full index scans, e.g. `SELECT a FROM table_name` where `a` is an indexed column.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_read_key`

- Description: Number of row read requests based on an index value. A high value indicates indexes are regularly being used, which is usually positive.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_read_last`

- Description: Number of requests to read the last row from an index. ORDER BY DESC results in a last-key request followed by several previous-key requests.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_read_next`

- Description: Number of requests to read the next row from an index (in order). Increments when doing an index scan or querying an index column with a range constraint.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_read_prev`

- Description: Number of requests to read the previous row from an index (in order). Mostly used with ORDER BY DESC.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_read_retry`

- Description: Number of read retrys triggered by semi_consistent_read (InnoDB feature).
- Scope: Global
- Data Type: `numeric`

`Handler_read_rnd`

- Description: Number of requests to read a row based on its position. If this value is high, you may not be using joins that don't use indexes properly, or be doing many full table scans.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_read_rnd_deleted`

- Description: Number of requests to delete a row based on its position.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_read_rnd_next`

- Description: Number of requests to read the next row. A large number of these may indicate many table scans and improperly used indexes.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_rollback`

- Description: Number of transaction rollback requests given to a storage engine. Differs from Com_rollback, which is the number of ROLLBACK commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_savepoint`

- Description: Number of transaction savepoint creation requests. Differs from Com_savepoint which is the number of SAVEPOINT commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_savepoint_rollback`

- Description: Number of requests to rollback to a transaction savepoint.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_tmp_delete`

- Description: Number of requests to delete a row in a temporary table.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_tmp_update`

- Description: Number of requests to update a row to a temporary table.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_tmp_write`

- Description: Number of requests to write a row to a temporary table.

- Scope: Global, Session
- Data Type: `numeric`

`Handler_update`

- Description: Number of requests to update a row in a table. Since MariaDB 5.3, this no longer counts temporary tables - see Handler_tmp_update.
- Scope: Global, Session
- Data Type: `numeric`

`Handler_write`

- Description: Number of requests to write a row to a table. Since MariaDB 5.3, this no longer counts temporary tables - see Handler_tmp_write.
- Scope: Global, Session
- Data Type: `numeric`

`Key_blocks_not_flushed`

- Description: Number of key cache blocks which have been modified but not flushed to disk.
- Scope: Global
- Data Type: `numeric`

`Key_blocks_unused`

- Description: Number of unused key cache blocks.
- Scope: Global
- Data Type: `numeric`

`Key_blocks_used`

- Description: Max number of key cache blocks which have been used simultaneously.
- Scope: Global
- Data Type: `numeric`

`Key_blocks_warm`

- Description: Number of key cache blocks in the warm list.
- Scope: Global
- Data Type: `numeric`

`Key_read_requests`

- Description: Number of key cache block read requests. See Optimizing key_buffer_size.
- Scope: Global
- Data Type: `numeric`

`Key_reads`

- Description: Number of physical index block reads. See Optimizing key_buffer_size.
- Scope: Global
- Data Type: `numeric`

`Key_write_requests`

- Description: Number of requests to write a block to the key cache.
- Scope: Global
- Data Type: `numeric`

`Key_writes`

- Description: Number of key cache block write requests
- Scope: Global
- Data Type: `numeric`

`Last_query_cost`

- Description: The most recent query optimizer query cost calculation. Can not be calculated for complex queries, such as subqueries or UNION. It will be set to 0 for complex queries.
- Scope: Session
- Data Type: `numeric`

`Maria_*`

- Description: When the Maria storage engine was renamed Aria, the Maria variables existing at the time were renamed at the same time. See Aria Server Status Variables.

`Max_memory_used`

- Description: The maximum memory allocation used by the current connection.
- Scope: Session
- Data Type: `numeric`
- Introduced: MariaDB 10.6.21

`Max_statement_time_exceeded`

- Description: Number of queries that exceeded the execution time specified by max_statement_time. See Aborting statements that take longer than a certain time to execute.
- Data Type: `numeric`

`Max_tmp_space_used`

- Description: Maximum temporary space used. See Limiting Size of Created Disk Temporary Files and Tables Overview
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Max_used_connections`

- Description: Max number of connections ever open at the same time. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Max_used_connections_time`

- Description: The time at which the last change of max_used_connections occurred. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `datetime`
- Introduced: MariaDB 11.0.2, MariaDB 11.1.1

`Memory_used`

> **Note:**
>
> `Memory_used` now encompasses `Innodb_buffer_pool_bytes_data` rather than reporting them as separate pools.

- Description: Memory used by the server (Global scope) or the current connection (Session scope), in bytes. Since MariaDB 10.6.16, Global scope includes major buffers (InnoDB buffer pool, MyISAM key buffer, and Aria page cache) that were previously excluded.
- Scope: Global, Session
- Data Type: `numeric`

`Memory_used_initial`

- Description: Amount of memory that was used when the server started to service the user connections.
- Scope: Global
- Data Type: `numeric`

`Not_flushed_delayed_rows`

- Description: Number of INSERT DELAYED rows waiting to be written.
- Scope: Global
- Data Type: `numeric`

`Open_files`

- Description: Number of regular files currently opened by the server. Does not include sockets or pipes, or storage engines using internal functions.
- Scope: Global
- Data Type: `numeric`

`Open_streams`

- Description: Number of currently opened streams, usually log files.
- Scope: Global
- Data Type: `numeric`

`Open_table_definitions`

- Description: Number of currently cached .frm files.
- Scope: Global
- Data Type: `numeric`

`Open_tables`

- Description: Number of currently opened tables, excluding temporary tables.

- Scope: Global, Session
- Data Type: `numeric`

`Opened_files`

- Description: Number of files the server has opened.
- Scope: Global
- Data Type: `numeric`

`Opened_plugin_libraries`

- Description: Number of shared libraries that the server has opened to load plugins.
- Scope: Global
- Data Type: `numeric`

`Opened_table_definitions`

- Description: Number of .frm files that have been cached.
- Scope: Global, Session
- Data Type: `numeric`

`Opened_tables`

- Description: Number of tables the server has opened.
- Scope: Global, Session
- Data Type: `numeric`

`Opened_views`

- Description: Number of views the server has opened.
- Scope: Global, Session
- Data Type: `numeric`

`Prepared_stmt_count`

- Description: Current number of prepared statements.
- Scope: Global
- Data Type: `numeric`

`Qcache_free_blocks`

- Description: Number of free query cache memory blocks.
- Scope: Global
- Data Type: `numeric`

`Qcache_free_memory`

- Description: Amount of free query cache memory.
- Scope: Global
- Data Type: `numeric`

`Qcache_hits`

- Description: Number of requests served by the query cache. The global value can be flushed by FLUSH STATUS.

- Scope: Global
- Data Type: `numeric`

`Qcache_inserts`

- Description: Number of queries ever cached in the query cache. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Qcache_lowmem_prunes`

- Description: Number of pruning operations performed to remove old results to make space for new results in the query cache. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Qcache_not_cached`

- Description: Number of queries that are uncacheable by the query cache, or use SQL_NO_CACHE. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Qcache_queries_in_cache`

- Description: Number of queries currently cached by the query cache.
- Scope: Global
- Data Type: `numeric`

`Qcache_total_blocks`

- Description: Number of blocks used by the query cache.
- Scope: Global
- Data Type: `numeric`

`Queries`

- Description: Number of statements executed by the server, excluding COM_PING and COM_STATISTICS. Differs from Questions in that it also counts statements executed within stored programs.
- Scope: Global, Session
- Data Type: `numeric`

`Query_time`

- Description: Cumulative time in seconds, with microsecond precision, of running queries.
- Scope: Global,Session
- Data Type: `numeric`
- Introduced: MariaDB 11.4

`Questions`

- Description: Number of statements executed by the server, excluding COM_PING, COM_STATISTICS, COM_STMT_PREPARE, COM_STMT_CLOSE, and COM_STMT_RESET statements. Differs from Queries in

that it doesn't count statements executed within stored programs.
- Scope: Global, Session
- Data Type: `numeric`

`Resultset_metadata_skipped`

- Description: Number of times sending the metadata has been skipped. Metadata is not resent if metadata does not change between prepare and execute of prepared statement, or between executes.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.6.0

`Rows_read`

- Description: Number of requests to read a row (excluding temporary tables).
- Scope: Global, Session
- Data Type: `numeric`

`Rows_sent`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Rows_tmp_read`

- Description: Number of requests to read a row in a temporary table.
- Scope: Global, Session
- Data Type: `numeric`

`Select_full_join`

- Description: Number of joins which did not use an index. If not zero, you may need to check table indexes.
- Scope: Global, Session
- Data Type: `numeric`

`Select_full_range_join`

- Description: Number of joins which used a range search of the first table.
- Scope: Global, Session
- Data Type: `numeric`

`Select_range`

- Description: Number of joins which used a range on the first table.
- Scope: Global, Session
- Data Type: `numeric`

`Select_range_check`

- Description: Number of joins without keys that check for key usage after each row. If not zero, you may need to check table indexes.
- Scope: Global, Session
- Data Type: `numeric`

`Select_scan`

- Description: Number of joins which used a full scan of the first table.
- Scope: Global, Session
- Data Type: `numeric`

`Slow_launch_threads`

- Description: Number of threads which took longer than slow_launch_time to create. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Slow_queries`

- Description: Number of queries which took longer than long_query_time to run. The slow query log does not need to be active for this to be recorded.
- Scope: Global, Session
- Data Type: `numeric`

`Sort_merge_passes`

- Description: Number of merge passes performed by the sort algorithm. If too high, you may need to look at improving your query indexes, or increasing the sort_buffer_size.
- Scope: Global, Session
- Data Type: `numeric`

`Sort_priority_queue_sorts`

- Description: The number of times that sorting was done through a priority queue. (The total number of times sorting was done is a sum Sort_range and Sort_scan). See filesort with small LIMIT optimization.
- Scope: Global, Session
- Data Type: `numeric`

`Sort_range`

- Description: Number of sorts which used a range.
- Scope: Global, Session
- Data Type: `numeric`

`Sort_rows`

- Description: Number of rows sorted.
- Scope: Global, Session
- Data Type: `numeric`

`Sort_scan`

- Description: Number of sorts which used a full table scan.
- Scope: Global, Session
- Data Type: `numeric`

`Subquery_cache_hit`

- Description: Counter for all subquery cache hits. The global value can be flushed by FLUSH STATUS.

- Scope: Global, Session
- Data Type: `numeric`

`Subquery_cache_miss`

- Description: Counter for all subquery cache misses. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Syncs`

- Description: Number of times my_sync() has been called, or the number of times the server has had to force data to disk. Covers the binary log, .frm creation (if these operations are configured to sync) and some storage engines (Archive,CSV, Aria), but not XtraDB/InnoDB).
- Scope: Global, Session
- Data Type: `numeric`

`Table_locks_immediate`

- Description: Number of table locks which were completed immediately. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Table_locks_waited`

- Description: Number of table locks which had to wait. Indicates table lock contention. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Table_open_cache_active_instances`

- Description: Number of active instances for open tables cache lookups.
- Scope:
- Data Type: `numeric`

`Table_open_cache_hits`

- Description: Number of hits for open tables cache lookups.
- Scope:
- Data Type: `numeric`

`Table_open_cache_misses`

- Description: Number of misses for open tables cache lookups.
- Scope:
- Data Type: `numeric`

`Table_open_cache_overflows`

- Description: Number of overflows for open tables cache lookups.
- Scope:
- Data Type: `numeric`

`Tc_log_max_pages_used`

- Description: Max number of pages used by the memory-mapped file-based transaction coordinator log. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Tc_log_page_size`

- Description: Page size of the memory-mapped file-based transaction coordinator log.
- Scope: Global
- Data Type: `numeric`

`Tc_log_page_waits`

- Description: Number of times a two-phase commit was forced to wait for a free memory-mapped file-based transaction coordinator log page. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Threads_cached`

- Description: Number of threads cached in the thread cache. This value will be zero if the thread pool is in use.
- Scope: Global
- Data Type: `numeric`

`Threads_connected`

- Description: Number of clients connected to the server. See Handling Too Many Connections. The `Threads_connected` name is inaccurate when the thread pool is in use, since each client connection does not correspond to a dedicated thread in that case.
- Scope: Global
- Data Type: `numeric`

`Threads_created`

- Description: Number of threads created to respond to client connections. If too large, look at increasing thread_cache_size.
- Scope: Global
- Data Type: `numeric`

`Threads_running`

- Description: Number of client connections that are actively running a command, and not just sleeping while waiting to receive the next command to execute. Some internal system threads also count towards this status variable if they would show up in the output of the SHOW PROCESSLIST statement.
  - In MariaDB 10.3.2 and before, a global counter was updated each time a client connection dispatched a command. In these versions, the global and session status variable are always the same value.
  - In MariaDB 10.3.3 and later, the global counter has been removed as a performance improvement. Instead, when the global status variable is queried, it is calculated dynamically by essentially adding up all the running client connections as they would appear in SHOW PROCESSLIST output. A client connection is only considered to be running if its thread COMMAND value is not equal to `Sleep`. When the session status variable is queried, it always returns `1`.

- Scope: Global
- Data Type: `numeric`

`Tmp_space_used`

- Description: Temporary space used. See Limiting Size of Created Disk Temporary Files and Tables Overview
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Update_scan`

- Description: Number of updates that required a full table scan.
- Scope: Global
- Data Type: `numeric`

`Uptime`

- Description: Number of seconds the server has been running.
- Scope: Global
- Data Type: `numeric`

`Uptime_since_flush_status`

- Description: Number of seconds since the last FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

This page is licensed: CC BY-SA / Gnu FDL

Server System Variables

Overview

MariaDB has many system variables that can be changed to suit your needs.

> **Note:**
>
> **For a full list of server options, system variables and status variables, see this page.**

Many of the general system variables are described on this page, but others are described elsewhere:

- Aria System Variables
- CONNECT System Variables
- Galera System Variables
- Global Transaction ID System Variables
- HandlerSocket Plugin System Variables
- InnoDB System Variables
- Mroonga System Variables
- MyRocks System Variables
- MyISAM System Variables
- Performance Schema System Variables
- Replication and Binary Log System Variables

- S3 Storage Engine System Variables
- Server_Audit System Variables
- Spider System Variables
- SQL_ERROR_LOG Plugin System Variables
- SSL System Variables
- Threadpool System Variables
- Vector System Variables

See also the Full list of MariaDB options, system and status variables.

Most of these can be set with command line options and many of them can be changed at runtime.
Variables that can be changed at runtime (and therefore are not read-only) are described as “Dynamic” below, and elsewhere in the documentation.

There are a few ways to see the full list of server system variables:

- While in the mariadb client, run:

```
SHOW VARIABLES;
```

- See SHOW VARIABLES for instructions on using this command.
- From your shell, run mariadbd like so:

```
mariadbd --verbose --help
```

- View the Information Schema GLOBAL_VARIABLES, SESSION_VARIABLES, and SYSTEM_VARIABLES tables.

Setting Server System Variables

There are several ways to set server system variables:

- Specify them on the command line:

```
shell> ./mariadbd-safe --aria_group_commit="hard"
```

- Specify them in your my.cnf file (see Configuring MariaDB with my.cnf for more information):

```
aria_group_commit = "hard"
```

- Set them from the mariadb client using the SET command. Only variables that are dynamic can be set at runtime in this way. Note that variables set in this way will not persist after a restart.

```
SET GLOBAL aria_group_commit="hard";
```

By convention, server variables have usually been specified with an underscore in the configuration files, and a dash on the command line. You can however specify underscores as dashes - they are interchangeable.

Variables that take a numeric size can either be specified in full, or with a suffix for easier readability. Valid suffixes are:

Suffix	Description	Value
K	kilobytes	1024
M	megabytes	10242
G	gigabytes	10243

Suffix	Description	Value
T	terabytes	10244 (from MariaDB 10.3.3)
P	petabytes	10245 (from MariaDB 10.3.3)
E	exabytes	10246 (from MariaDB 10.3.3)

The suffix can be upper or lower-case.

List of Server System Variables

`allow_suspicious_udfs`

- Description: Allows use of user-defined functions consisting of only one symbol `x()` without corresponding `x_init()` or `x_deinit()`. That also means that one can load any function from any library, for example `exit()` from `libc.so`. Not recommended unless you require old UDFs with one symbol that cannot be recompiled. Before MariaDB 10.10, available as an option only.
- Command line: `--allow-suspicious-udfs`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.10

`alter_algorithm`

- Description: The implied `ALGORITHM` for ALTER TABLE if no `ALGORITHM` clause is specified. The deprecated variable old_alter_table is an alias for this. The feature was removed in MariaDB 11.5. See ALGORITHM=DEFAULT.
  - `COPY` corresponds to the pre-MySQL 5.1 approach of creating an intermediate table, copying data one row at a time, and renaming and dropping tables.
  - `INPLACE` requests that the operation be refused if it cannot be done natively inside a the storage engine.
  - `DEFAULT` (the default) chooses `INPLACE` if available, and falls back to `COPY`.
  - `NOCOPY` refuses to copy a table.
  - `INSTANT` refuses an operation that would involve any other than metadata changes.
- Command line: `--alter-algorithm=default`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumerated`
- Default Value: `DEFAULT`
- Valid Values: `DEFAULT`, `COPY`, `INPLACE`, `NOCOPY`, `INSTANT`
- Introduced: MariaDB 10.3.7
- Deprecated: MariaDB 11.5

`analyze_max_length`

- Description: Prevents collection of column statistics for `CHAR`/`VARCHAR` columns that can be analyzed automatically by `ANALYZE TABLE PERSISTENT`. Columns exceeding this threshold in bytes will be skipped unless included explicitly in `FOR COLUMNS(...)`.
- Command line: `--analyze-max-length=val`
- Scope: Global, Session

- Dynamic: Yes
- Data Type: `INT UNSIGNED`
- Default Value: `4294967295 (4G)`
- Range: `32` to `4294967295`
- Introduced: MariaDB 10.6.23, MariaDB 10.11.14, MariaDB 11.4.8, MariaDB 11.8.3, MariaDB 12.0.2, MariaDB 12.1.1, MariaDB Enterprise Server 10.6, MariaDB Enterprise Server 11.4, MariaDB Enterprise Server 11.8

`analyze_sample_percentage`

- Description: Percentage of rows from the table ANALYZE TABLE will sample to collect table statistics. Set to 0 to let MariaDB decide what percentage of rows to sample.
- Command line: `--analyze-sample-percentage=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100.000000`
- Range: `0` to `100`
- Introduced: MariaDB 10.4.3

`autocommit`

- Description: If set to 1, the default, all queries are committed immediately. The LOCK IN SHARE MODE and FOR UPDATE clauses therefore have no effect. If set to 0, they are only committed upon a COMMIT statement or rolled back with a ROLLBACK statement. If autocommit is set to 0, and then changed to 1, all open transactions are immediately committed.
- Command line: `--autocommit[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`

`automatic_sp_privileges`

- Description: When set to 1, the default, when a stored routine is created, the creator is automatically granted permission to ALTER (which includes dropping) and to EXECUTE the routine. If set to 0, the creator is not automatically granted these privileges.
- Command line: `--automatic-sp-privileges`, `--skip-automatic-sp-privileges`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`

`back_log`

- Description: Connections take a small amount of time to start, and this setting determines the number of outstanding connection requests MariaDB can have, or the size of the listen queue for incoming TCP/IP requests. Requests beyond this will be refused. Increase if you expect short bursts of connections. Cannot be set higher than the operating system limit (see the Unix listen() man page). If not set, set to `0`, or the `--autoset-back-log` option is used, will be autoset to the lower of `900` and (50 + max_connections/5).
- Command line: `--back-log=#`

- Scope: Global
- Dynamic: No
- Type: number
- Default Value:
  - The lower of `900` and (50 + max_connections/5)

`basedir`

- Description: Path to the MariaDB installation directory. Other paths are usually resolved relative to this base directory.
- Command line: `--basedir=path` or `-b path`
- Scope: Global
- Dynamic: No
- Type: directory name

`big_tables`

- Description: If this system variable is set to 1, then temporary tables will be saved to disk instead of memory.
  - This system variable's original intention was to allow result sets that were too big for memory-based temporary tables and to avoid the resulting 'table full' errors.
  - This system variable is no longer needed, because the server can automatically convert large memory-based temporary tables into disk-based temporary tables when they exceed the value of the tmp_memory_table_size system variable.
  - To prevent memory-based temporary tables from being used at all, set the tmp_memory_table_size system variable to `0`.
  - In MariaDB 5.5 and earlier, sql_big_tables is a synonym.
  - From MariaDB 10.5, this system variable is deprecated.
- Command line: `--big-tables`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`
- Deprecated: MariaDB 10.5.0
- Removed: MariaDB 12.0

`bind_address`

- Description: By default, the MariaDB server listens for TCP/IP connections on all addresses. You can specify an alternative when the server starts using this option; either a host name, an IPv4 or an IPv6 address, "::" or "" *(all addresses). In some systems, such as Debian and Ubuntu, the bind_address is set to 127.0.0.1, which binds the server to listen on localhost only.* `bind_address` *has always been available as a mariadbd option; from MariaDB 10.3.3 it's also available as a system variable. Before MariaDB 10.6.0 "::" implied listening additionally on IPv4 addresses like ""*. From 10.6.0 onwards it refers to IPv6 stictly. Starting with MariaDB 10.11, a comma-separated list of addresses to bind to can be given. See also Configuring MariaDB for Remote Client Access.
- Command line: `--bind-address=addr`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: (Empty string)

- Valid Values: Host name, IPv4, IPv6, ::, *
- Introduced: MariaDB 10.3.3 (as a system variable)

`block_encryption_mode`

- Description: Default block encryption mode for AES_ENCRYPT() and AES_DECRYPT() functions.
- Command line: `--block-encryption-mode=val`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `aes-128-ecb`
- Valid values: `aes-128-ecb`, `aes-192-ecb`, `aes-256-ecb`, `aes-128-cbc`, `aes-192-cbc`, `aes-256-cbc`, `aes-128-ctr`, `aes-192-ctr`, `aes-256-ctr`
- Introduced: MariaDB 11.2.0

`bulk_insert_buffer_size`

- Description: Size in bytes of the per-thread cache tree used to speed up bulk inserts into MyISAM and Aria tables. A value of 0 disables the cache tree.
- Command line: `--bulk-insert-buffer-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `8388608`
- Range - 32 bit: `0` to `4294967295`
- Range - 64 bit: `0` to `18446744073709547520`

`character_set_client`

- Description: Determines the character set for queries arriving from the client. It can be set per session by the client, although the server can be configured to ignore client requests with the `--skip-character-set-client-handshake` option. If the client does not request a character set or requests a character set that the server does not support, the global value will be used. utf16, utf16le, utf32 and ucs2 cannot be used as client character sets. From MariaDB 10.6, the `utf8` character set (and related collations) was by default an alias for `utf8mb3` rather than the other way around. From MariaDB 13.1, `utf8` is by default an alias for `utf8mb4`; the previous `utf8mb3` default can be restored with the deprecated `UTF8_IS_UTF8MB3` flag of the old_mode system variable.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value:
  - >= MariaDB 11.6: `utf8mb4`
  - <= MariaDB 11.5: `latin1`

`character_set_collations`

- Description: Overrides for character set default collations. Takes a comma-delimited list of character set and collation settings, for example `SET @@character_set_collations = 'utf8mb4=uca1400_ai_ci, latin2=latin2_hungarian_ci';` The new variable will take effect in all cases where a character set is explicitly or implicitly specified without an explicit COLLATE clause, including but not limited to:
  - Column collation

    - Table collation
    - Database collation
    - CHAR(expr USING csname)
    - CONVERT(expr USING csname)
    - CAST(expr AS CHAR CHARACTER SET csname)
    - '' - character string literal
    - _utf8mb3'text' - a character string literal with an introducer
    - _utf8mb3 X'61' - a character string literal with an introducer with hex notation
    - _utf8mb3 0x61 - a character string literal with an introducer with hex hybrid notation
    - @@collation_connection after a SET NAMES without COLLATE
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value:
    - `utf8mb3=utf8mb3_uca1400_ai_ci, ucs2=ucs2_uca1400_ai_ci, utf8mb4=utf8mb4_uca1400_ai_ci, utf16=utf16_uca1400_ai_ci, utf32=utf32_uca1400_ai_ci` (>= MariaDB 11.5)
    - Empty (<= MariaDB 11.4)
- Introduced: MariaDB 11.2

`character_set_connection`

- Description: Character set used for number to string conversion, as well as for literals that don't have a character set introducer. From MariaDB 10.6, the `utf8` character set (and related collations) was by default an alias for `utf8mb3` rather than the other way around. From MariaDB 13.1, `utf8` is by default an alias for `utf8mb4`; the previous `utf8mb3` default can be restored with the deprecated `UTF8_IS_UTF8MB3` flag of the old_mode system variable.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value:
    - >=MariaDB 11.6: `utf8mb4`
    - >= MariaDB 10.6: `utf8mb3`
    - <= MariaDB 10.5: `utf8`

`character_set_database`

- Description: Character set used by the default database and set by the server whenever the default database is changed. If there's no default database, character_set_database contains the same value as character_set_server. This variable is dynamic, but should not be set manually, only by the server.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `utf8mb4` (>= MariaDB 11.6.0), `latin1` (<= MariaDB 11.5)

`character_set_filesystem`

- Description: The character set for the filesystem. Used for converting file names specified as a string literal from character_set_client to character_set_filesystem before opening the file. By default, set to `binary`, so no

conversion takes place. This could be useful for statements such as LOAD_FILE() or LOAD DATA INFILE on system where multi-byte file names are use.

- Command line: `--character-set-filesystem=name`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `binary`

`character_set_results`

- Description: Character set used for results and error messages returned to the client. From MariaDB 10.6, the `utf8` character set (and related collations) was by default an alias for `utf8mb3` rather than the other way around. From MariaDB 13.1, `utf8` is by default an alias for `utf8mb4`; the previous `utf8mb3` default can be restored with the deprecated `UTF8_IS_UTF8MB3` flag of the old_mode system variable.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `utf8mb3` (>= MariaDB 10.6), `utf8` (<= MariaDB 10.5)

`character_set_server`

- Description: Default character set used by the server. See character_set_database for character sets used by the default database. Defaults may be different on some systems, see for example Differences in MariaDB in Debian.
- Command line: `--character-set-server`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `utf8mb4` (>= MariaDB 11.6.0), `latin1` (<= MariaDB 11.5)

`character_set_system`

- Description: Character set used by the server to store identifiers, always set to utf8, or its synonym utf8mb3 starting with MariaDB 10.6. From MariaDB 10.6, the `utf8` character set (and related collations) was by default an alias for `utf8mb3` rather than the other way around. From MariaDB 13.1, `utf8` is by default an alias for `utf8mb4`; the previous `utf8mb3` default can be restored with the deprecated `UTF8_IS_UTF8MB3` flag of the old_mode system variable.
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `utf8mb3` (>= MariaDB 10.6), `utf8` (<= MariaDB 10.5)

`character_sets_dir`

- Description: Directory where the character sets are installed.
- Command line: `--character-sets-dir=path`
- Scope: Global
- Dynamic: No
- Type: directory name

`check_constraint_checks`

- Description: If set to `0`, will disable constraint checks, for example when loading a table that violates some constraints that you plan to fix later.
- Scope: Global, Session
- Dynamic: Yes
- Type: boolean
- Default: ON

`collation_connection`

- Description: Collation used for the connection character set.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`

`collation_database`

- Description: Collation used for the default database. Set by the server if the default database changes, if there is no default database the value from the `collation_server` variable is used. This variable is dynamic, but should not be set manually, only by the server.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`

`collation_server`

- Description: Default collation used by the server. This is set to the default collation for a given character set automatically when character_set_server is changed, but it can also be set manually. Defaults may be different on some systems, see for example Differences in MariaDB in Debian.
- Command line: `--collation-server=name`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `latin1_swedish_ci`

`completion_type`

- Description: The transaction completion type. If set to `NO_CHAIN` or `0` (the default), there is no effect on commits and rollbacks. If set to `CHAIN` or `1`, a COMMIT statement is equivalent to COMMIT AND CHAIN, while a ROLLBACK is equivalent to ROLLBACK AND CHAIN, so a new transaction starts straight away with the same isolation level as transaction that's just finished. If set to `RELEASE` or `2`, a COMMIT statement is equivalent to COMMIT RELEASE, while a ROLLBACK is equivalent to ROLLBACK RELEASE, so the server will disconnect after the transaction completes. Note that the transaction completion type only applies to explicit commits, not implicit commits.
- Command line: `--completion-type=name`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumerated`
- Default Value: `NO_CHAIN`
- Valid Values: `0`, `1`, `2`, `NO_CHAIN`, `CHAIN`, `RELEASE`

`concurrent_insert`

- Description: If set to `AUTO` or `1`, the default, MariaDB allows concurrent INSERTs and SELECTs for MyISAM tables with no free blocks in the data (deleted rows in the middle). If set to `NEVER` or `0`, concurrent inserts are disabled. If set to `ALWAYS` or `2`, concurrent inserts are permitted for all MyISAM tables, even those with holes, in which case new rows are added at the end of a table if the table is being used by another thread. If the –skip-new option is used when starting the server, concurrent_insert is set to `NEVER`. Changing the variable only affects new opened tables. Use FLUSH TABLES If you want it to also affect cached tables. See Concurrent Inserts for more.
- Command line: `--concurrent-insert[=value]`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumerated`
- Default Value: `AUTO`
- Valid Values: `0`, `1`, `2`, `AUTO`, `NEVER`, `ALWAYS`

`connect_timeout`

- Description: Time in seconds that the server waits for a connect packet before returning a 'Bad handshake'. Increasing may help if clients regularly encounter 'Lost connection to MySQL server at 'X', system error: error_number' type-errors.
- Command line: `--connect-timeout=#`
- Scope: Global
- Dynamic: Yes
- Type: numeric
- Default Value: `10`
- Range: `2` to `31536000`

`core_file`

- Description: Write a core-file on crashes. The file name and location are system dependent. On Linux it is usually called `core.${PID}`, and it is usually written to the data directory. However, this can be changed.
  - See Enabling Core Dumps for more information.
  - Previously this system variable existed only as an option, but it was also made into a read-only system variable starting with MariaDB 10.3.9, MariaDB 10.2.17 and MariaDB 10.1.35.
  - On Windows >= MariaDB 10.4.3, this option is set by default.
  - Note that the option accepts no arguments; specifying `--core-file` sets the value to `ON`. It cannot be disabled in the case of Windows >= MariaDB 10.4.3.
- Command line: `--core-file`
- Scope: Global
- Dynamic: No
- Type: boolean
- Default Value:
  - Windows >= MariaDB 10.4.3: `ON`
  - All other systems: `OFF`

`datadir`

- Description: Directory where the data is stored.
- Command line: `--datadir=path` or `-h path`

- Scope: Global
- Dynamic: No
- Type: directory name

`date_format`

- Description: Unused.
- Removed: MariaDB 11.3.0

`datetime_format`

- Description: Unused.
- Removed: MariaDB 11.3.0

`debug/debug_dbug`

- Description: Available in debug builds only (built with -DWITH_DEBUG=1). Used in debugging through the DBUG library to write to a trace file. Just using `--debug` will write a trace of what mariadbd is doing to the default trace file.
- Command line: `-#`, `--debug[=debug_options]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value:
  - = MariaDB 10.5: `d:t:i:o,/tmp/mariadbd.trace` (Unix) or `d:t:i:O,\mariadbd.trace` (Windows)
- Debug Options: See the option flags on the mysql_debug page

`debug_no_thread_alarm`

- Description: Disable system thread alarm calls. Disabling it may be useful in debugging or testing, never do it in production.
- Command line: `--debug-no-thead-alarm=#`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 11.4

`debug_sync`

- Description: Used in debugging to show the interface to the Debug Sync facility. MariaDB needs to be configured with -DENABLE_DEBUG_SYNC=1 for this variable to be available.
- Scope: Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `OFF` or `ON - current signal name`

`default_password_lifetime`

- Description: This defines the global password expiration policy. 0 means automatic password expiration is disabled. If the value is a positive integer N, the passwords must be changed every N day. This behavior can be

overridden using the password expiration options in ALTER USER.

- Command line: `--default-password-lifetime=#`
- Scope: Global
- Dynamic: Yes
- Type: numeric
- Default Value: `0`
- Range: `0` to `4294967295`

`default_regex_flags`

- Description: Introduced to address remaining incompatibilities between PCRE and the old regex library. Accepts a comma-separated list of zero or more of the following values:

Value	Pattern equivalent	Meaning
DOTALL	(?s)	. matches anything including NL
DUPNAMES	(?J)	Allow duplicate names for subpatterns
EXTENDED	(?x)	Ignore white space and comments
EXTRA	(?X)	extra features (e.g. error on unknown escape character)
MULTILINE	(?m)	^ and $ match newlines within data
UNGREEDY	(?U)	Invert greediness of quantifiers

- Command line: `--default-regex-flags=value`
- Scope: Global, Session
- Dynamic: Yes
- Type: enumeration
- Default Value: empty
- Valid Values: `DOTALL`, `DUPNAMES`, `EXTENDED`, `EXTRA`, `MULTILINE`, `UNGREEDY`

`default_storage_engine`

- Description: The default storage engine. The default storage engine must be enabled at server startup, or the server won't start.
- Command line: `--default-storage-engine=name`
- Scope: Global, Session
- Dynamic: Yes
- Type: enumeration
- Default Value: `InnoDB`

`default_table_type`

- Description: A synonym for default_storage_engine. Removed in MariaDB 5.5.
- Command line: `--default-table-type=name`
- Scope: Global, Session
- Dynamic: Yes
- Removed: MariaDB 5.5

`default_tmp_storage_engine`

- Description: Default storage engine that will be used for tables created with CREATE TEMPORARY TABLE where no engine is specified. For internal temporary tables see aria_used_for_temp_tables). The storage engine used must be active or the server will not start. See default_storage_engine for the default for non-temporary tables. Defaults to NULL, in which case the value from default_storage_engine is used. ROCKSDB temporary tables cannot be created. Before MariaDB 10.7, attempting to do so would silently fail, and a MyISAM table would instead be created. From MariaDB 10.7, an error is returned.
- Command line: `--default-tmp-storage-engine=name`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: NULL

`default_week_format`

- Description: Default mode for the WEEK() function. See that page for details on the different modes
- Command line: `--default-week-format=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `7`

`delay_key_write`

- Description: Specifies how MyISAM tables handles CREATE TABLE DELAY_KEY_WRITE. If set to `ON`, the default, any DELAY KEY WRITEs are honored. The key buffer is then flushed only when the table closes, speeding up writes. MyISAM tables should be automatically checked upon startup in this case, and –external locking should not be used, as it can lead to index corruption. If set to `OFF`, DELAY KEY WRITEs are ignored, while if set to `ALL`, all new opened tables are treated as if created with DELAY KEY WRITEs enabled.
- Command line: `--delay-key-write[=name]`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `ON`
- Valid Values: `ON`, `OFF`, `ALL`

`delayed_insert_limit`

- Description: After this many rows have been inserted with INSERT DELAYED, the handler will check for and execute any waiting SELECT statements.
- Command line: `--delayed-insert-limit=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `1` to `4294967295`

`delayed_insert_timeout`

- Description: Time in seconds that the INSERT DELAYED handler will wait for INSERTs before terminating.
- Command line: `--delayed-insert-timeout=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `300`

`delayed_queue_size`

- Description: Number of rows, per table, that can be queued when performing INSERT DELAYED statements. If the queue becomes full, clients attempting to perform INSERT DELAYED's will wait until the queue has room available again.
- Command line: `--delayed-queue-size=#`
- Scope: Global
- Dynamic: Yes
- Type: numeric
- Default Value: `1000`
- Range: `1 to 4294967295`

`disconnect_on_expired_password`

- Description: When a user password has expired (see User Password Expiry), this variable controls how the server handles clients that are not aware of the sandbox mode. If enabled, the client is not permitted to connect, otherwise the server puts the client in a sandbox mode.
- Command line: `--disconnect-on-expired-password[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Type: boolean
- Default Value: `OFF`

`div_precision_increment`

- Description: The precision of the result of the decimal division will be the larger than the precision of the dividend by that number. By default it's `4`, so `SELECT 2/15` would return 0.1333 and `SELECT 2.0/15` would return 0.13333. After setting div_precision_increment to `6`, for example, the same operation would return 0.133333 and 0.1333333 respectively. `div_precision_increment` is taken into account in intermediate calculations. Instead of truncating decimal values after every division, they are instead truncated for comparison purposes only.
- Command line: `--div-precision-increment=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4`
- Range: `0` to `30`

`encrypt_tmp_disk_tables`

- Description: Enables automatic encryption of all internal on-disk temporary tables that are created during query execution if aria_used_for_temp_tables=ON is set. See Data at Rest Encryption and Enabling Encryption for

Internal On-disk Temporary Tables.

- Command line: `--encrypt-tmp-disk-tables[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`encrypt_tmp_files`

- Description: Enables automatic encryption of temporary files, such as those created for filesort operations, binary log file caches, etc. See Data at Rest Encryption.
- Command line: `--encrypt-tmp-files[={0|1}]`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`encryption_algorithm`

- Description: Which encryption algorithm to use for table encryption. `aes_cbc` is the recommended one. See Table and Tablespace Encryption.
- Command line: `--encryption-algorithm=value`
- Scope: Global
- Dynamic: No
- Data Type: `enum`
- Default Value: `none`
- Valid Values: `none`, `aes_ecb`, `aes_cbc`, `aes_ctr`
- Introduced: MariaDB 10.1.3
- Removed: MariaDB 10.1.4

`enforce_storage_engine`

- Description: Force the use of a particular storage engine for new tables. Used to avoid unwanted creation of tables using another engine. For example, setting to InnoDB will prevent any MyISAM tables from being created. If another engine is specified in a CREATE TABLE statement, the outcome depends on whether the `NO_ENGINE_SUBSTITUTION` SQL_MODE has been set or not. If set, the query will fail, while if not set, a warning will be returned and the table created according to the engine specified by this variable. The variable has a session scope but is only modifiable by a user with the SUPER privilege.
- Command line: None
- Scope: Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `none`

`engine_condition_pushdown`

- Description: Deprecated in MariaDB 5.5 and removed and replaced by the optimizer_switch `engine_condition_pushdown={on|off}` flag in MariaDB 10.0. Specifies whether the engine condition pushdown optimization is enabled. Since MariaDB 10.1.1, engine condition pushdown is enabled for all engines that support it.

- Command line: `--engine-condition-pushdown`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 5.5
- Removed: MariaDB 10.0

`eq_range_index_dive_limit`

- Description: Limit used for speeding up queries listed by long nested INs. The optimizer will use existing index statistics instead of doing index dives for equality ranges if the number of equality ranges for the index is larger than or equal to this number. If set to `0` (unlimited), index dives are always used.
- Command line: `--eq-range-index-dive-limit=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `200`
- Range: `0` to `4294967295`

`error_count`

- Description: Read-only variable denoting the number of errors from the most recent statement in the current session that generated errors. See SHOW_ERRORS().
- Scope: Session
- Dynamic: Yes
- Data Type: `numeric`

`event_scheduler`

- Description: Status of the Event Scheduler. Can be set to `ON` or `OFF`, while `DISABLED` means it cannot be set at runtime. Setting the variable will cause a load of events if they were not loaded at startup.
- Command line: `--event-scheduler[=value]`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `OFF`
- Valid Values: `ON` (or `1`), `OFF` (or `0`), `DISABLED`

`expensive_subquery_limit`

- Description: Number of rows to be examined for a query to be considered expensive, that is, maximum number of rows a subquery may examine in order to be executed during optimization and used for constant optimization.
- Command line: `--expensive-subquery-limit=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `0` upwards

`explicit_defaults_for_timestamp`

- Description: This option causes CREATE TABLE to create all TIMESTAMP columns as NULL with the DEFAULT NULL attribute, without this option, TIMESTAMP columns are NOT NULL and have implicit DEFAULT clauses.
- Command line: `--explicit-defaults-for-timestamp=[={0|1}]`
- Scope:
  - Global, Session (>= MariaDB 10.8.4, MariaDB 10.7.5, MariaDB 10.6.9, MariaDB 10.5.17)
  - Global (<= MariaDB 10.8.3, MariaDB 10.7.4, MariaDB 10.6.8, MariaDB 10.5.16)
- Dynamic:
  - Yes (>= MariaDB 10.8.4, MariaDB 10.7.5, MariaDB 10.6.9, MariaDB 10.5.17)
  - No (<= MariaDB 10.8.3, MariaDB 10.7.4, MariaDB 10.6.8, MariaDB 10.5.16)
- Data Type: `boolean`
- Default Value: `ON` (>= MariaDB 10.10), `OFF` (<= MariaDB 10.9)

`external_user`

- Description: External user name set by the plugin used to authenticate the client. `NULL` if native MariaDB authentication is used. For example, from MariaDB 11.6, the Unix socket authentication plugin permits an authentication string, so that the OS and MariaDB user will be different. `external_user` then contains the external OS user. See Authentication Plugin - Unix Socket: Creating Users
- Scope: Session
- Dynamic: No
- Data Type: `string`
- Default Value: `NULL`

`flush`

- Description: Usually, MariaDB writes changes to disk after each SQL statement, and the operating system handles synchronizing (flushing) it to disk. If set to `ON`, the server will synchronize all changes to disk after each statement.
- Command line: `--flush`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`flush_time`

- Description: Interval in seconds that tables are closed to synchronize (flush) data to disk and free up resources. If set to 0, the default, there is no automatic synchronizing tables and closing of tables. This option should not be necessary on systems with sufficient resources.
- Command line: `--flush_time=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`

`foreign_key_checks`

- Description: If set to 1 (the default) foreign key constraints (including ON UPDATE and ON DELETE behavior) InnoDB tables are checked, while if set to 0, they are not checked. `0` is not recommended for normal use, though it can be useful in situations where you know the data is consistent, but want to reload data in a different order from that specified by parent/child relationships. Setting this variable to 1 does not retrospectively check for inconsistencies introduced while set to 0. Setting it to 0 suspends the checks on data only; it does not permit schema changes that would leave a constraint invalid, so a column used in a foreign key still cannot be dropped, renamed, or changed in type, character set, or collation. See Changing the Character Set or Collation of a Foreign Key Column.
- Command line: None
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`

`ft_boolean_syntax`

- Description: List of operators supported by an IN BOOLEAN MODE full-text search. If you wish to change, note that each character must be ASCII and non-alphanumeric, the full string must be 14 characters and the first or second character must be a space (marking the behavior by default). Positions 10, 13 and 14 are reserved for future extensions. Also, no duplicates are permitted except for the phrase quoting characters in positions 11 and 12, which may be the same.
- Command line: `--ft-boolean-syntax=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `+ -><()*:""&|`

`ft_max_word_len`

- Description: Maximum length for a word to be included in the MyISAM full-text index. If this variable is changed, the full-text index must be rebuilt in order for the new value to take effect. The quickest way to do this is by issuing a `REPAIR TABLE table_name QUICK` statement. See innodb_ft_max_token_size for the InnoDB equivalent.
- Command line: `--ft-max-word-len=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `84`
- Minimum Value: `10`

`ft_min_word_len`

- Description: Minimum length for a word to be included in the MyISAM full-text index. If this variable is changed, the full-text index must be rebuilt in order for the new value to take effect. The quickest way to do this is by issuing a `REPAIR TABLE table_name QUICK` statement. See innodb_ft_min_token_size for the InnoDB equivalent.
- Command line: `--ft-min-word-len=#`
- Scope: Global
- Dynamic: No

- Data Type: `numeric`
- Default Value: `4`
- Minimum Value: `1`

`ft_query_expansion_limit`

- Description: For full-text searches, denotes the number of top matches when using `WITH QUERY EXPANSION`.
- Command line: `--ft-query-expansion-limit=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `20`
- Range: `0` to `1000`

`ft_stopword_file`

- Description: File containing a list of stopwords for use in MyISAM full-text searches. Unless an absolute path is specified the file will be looked for in the data directory. The file is not parsed for comments, so all words found become stopwords. By default, a built-in list of words (built from `storage/myisam/ft_static.c file`) is used. Stopwords can be disabled by setting this variable to `''` (an empty string). If this variable is changed, the full-text index must be rebuilt. The quickest way to do this is by issuing a `REPAIR TABLE table_name QUICK` statement. See innodb_ft_server_stopword_table for the InnoDB equivalent.
- Command line: `--ft-stopword-file=file_name`
- Scope: Global
- Dynamic: No
- Data Type: `file name`
- Default Value: `(built-in)`

`general_log`

- Description: If set to 0, the default unless the –general-log option is used, the general query log is disabled, while if set to 1, the general query log is enabled. See log_output for how log files are written. If that variable is set to `NONE`, no logs will be written even if general_query_log is set to `1`.
- Command line: `--general-log`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`

`general_log_file`

- Description: Name of the general query log file. If this is not specified, the name is taken from the log-basename setting or from your system hostname with `.log` as a suffix. If –log-basename is also set, `general_log_file` should be placed after in the config files. Later settings override earlier settings, so `log-basename` will override any earlier log file name settings.
- Command line: `--general-log-file=file_name`
- Scope: Global
- Dynamic: Yes
- Data Type: `file name`
- Default Value: `host_name.log`

`group_concat_max_len`

- Description: Maximum length in bytes of the returned result for the functions GROUP_CONCAT(), JSON_OBJECTAGG and JSON_ARRAYAGG.
- Command line: `--group-concat-max-len=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `1048576` (1M)
- Range: `4` to `1073741824`

.

`have_compress`

- Description: If the zlib compression library is accessible to the server, this will be set to `YES`, otherwise it will be `NO`. The COMPRESS() and UNCOMPRESS() functions will only be available if set to `YES`.
- Scope: Global
- Dynamic: No

`have_crypt`

- Description: If the crypt() system call is available this variable will be set to `YES`, otherwise it will be set to `NO`. If set to `NO`, the ENCRYPT() function cannot be used.
- Scope: Global
- Dynamic: No

`have_csv`

- Description: If the server supports CSV tables, will be set to `YES`, otherwise will be set to `NO`. Removed in MariaDB 10.0, use the Information Schema PLUGINS table or SHOW ENGINES instead.
- Scope: Global
- Dynamic: No
- Removed: MariaDB 10.0

`have_dynamic_loading`

- Description: If the server supports dynamic loading of plugins, will be set to `YES`, otherwise will be set to `NO`.
- Scope: Global
- Dynamic: No

`have_geometry`

- Description: If the server supports spatial data types, will be set to `YES`, otherwise will be set to `NO`.
- Scope: Global
- Dynamic: No

`have_ndbcluster`

- Description: If the server supports NDBCluster.
- Scope: Global
- Dynamic: No

- Removed: MariaDB 10.0

`have_partitioning`

- Description: If the server supports partitioning, will be set to `YES`, unless the `--skip-partition` option is used, in which case will be set to `DISABLED`. Will be set to `NO` otherwise. Removed in MariaDB 10.0 - SHOW PLUGINS should be used instead.
- Scope: Global
- Dynamic: No
- Removed: MariaDB 10.0

`have_profiling`

- Description: If statement profiling is available, will be set to `YES`, otherwise will be set to `NO`. See SHOW PROFILES() and SHOW PROFILE().
- Scope: Global
- Dynamic: No

`have_query_cache`

- Description: If the server supports the query cache, will be set to `YES`, otherwise will be set to `NO`.
- Scope: Global
- Dynamic: No

`have_rtree_keys`

- Description: If RTREE indexes (used for spatial indexes) are available, will be set to `YES`, otherwise will be set to `NO`.
- Scope: Global
- Dynamic: No

`have_symlink`

- Description: This system variable can be used to determine whether the server supports symbolic links (note that it has no meaning on Windows).
  - If symbolic links are supported, then the value will be `YES`.
  - If symbolic links are not supported, then the value will be `NO`.
  - If symbolic links are disabled with the –symbolic-links option and the `skip` option prefix (i.e. –skip-symbolic-links), then the value will be `DISABLED`.
  - Symbolic link support is required for the INDEX DIRECTORY and DATA DIRECTORY table options.
- Scope: Global
- Dynamic: No

`histogram_size`

- Description: Number of bytes used for a histogram, or, from MariaDB 10.7 when histogram_type is set to `JSON_HB`, number of buckets. If set to 0, no histograms are created by ANALYZE.
- Command line: `--histogram-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `254`

- Range: `0` to `255`

`histogram_type`

- Description: Specifies the type of histograms created by ANALYZE..
  - `SINGLE_PREC_HB` - single precision height-balanced.
  - `DOUBLE_PREC_HB` - double precision height-balanced.
  - `JSON_HB` - JSON height-balanced histograms (from MariaDB 10.8)
- Command line: `--histogram-type=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value:
  - `JSON_HB` (>= MariaDB 11.0)
  - `DOUBLE_PREC_HB` (<= MariaDB 10.11, >= MariaDB 10.4.3)
- Valid Values:
  - `SINGLE_PREC_HB`, `DOUBLE_PREC_HB` (<= MariaDB 10.6)
  - `SINGLE_PREC_HB`, `DOUBLE_PREC_HB`, `JSON_HB` (>= MariaDB 10.8)

`host_cache_size`

- Description: Number of host names that will be cached to avoid resolving. Setting to `0` disables the cache. Changing the value while the server is running causes an implicit FLUSH HOSTS, clearing the host cache and truncating the performance_schema.host_cache table. If you are connecting from a lot of different machines you should consider increasing. Some container configs explicitly set `host_cache_size` to `0`, rather than leave it as the default, `128`.
- Command line: `--host-cache-size=#`.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `128`
- Range: `0` to `65536`

`hostname`

- Description: When the server starts, this variable is set to the server host name.
- Scope: Global
- Dynamic: No
- Data Type: `string`

`identity`

- Description: A synonym for last_insert_id variable.

`idle_readonly_transaction_timeout`

- Description: Time in seconds that the server waits for idle read-only transactions before killing the connection. If set to `0`, the default, connections are never killed. See also idle_transaction_timeout, idle_write_transaction_timeout and Transaction Timeouts.
- Scope: Global, Session

- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `31536000`

`idle_transaction_timeout`

- Description: Time in seconds that the server waits for idle transactions before killing the connection. If set to `0`, the default, connections are never killed. See also idle_readonly_transaction_timeout, idle_write_transaction_timeout and Transaction Timeouts.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `31536000`

`idle_write_transaction_timeout`

- Description: Time in seconds that the server waits for idle read-write transactions before killing the connection. If set to `0`, the default, connections are never killed. See also idle_transaction_timeout, idle_readonly_transaction_timeout and Transaction Timeouts. Called `idle_readwrite_transaction_timeout` until MariaDB 10.3.2.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `31536000`

`ignore_db_dirs`

- Description: Tells the server that this directory can never be a database. That means two things - firstly it is ignored by the SHOW DATABASES command and INFORMATION_SCHEMA tables. And secondly, USE, CREATE DATABASE and SELECT statements will return an error if the database from the ignored list specified. Use this option for several times if you need to ignore more than one directory. To make the list empty set the void value to the option as –ignore-db-dir=. If the option or configuration is specified multiple times, viewing this value will list the ignore directories separated by commas.
- Command line: `--ignore-db-dirs=dir`.
- Scope: Global
- Dynamic: No
- Data Type: `string`

`in_predicate_conversion_threshold`

- Description: The minimum number of scalar elements in the value list of an IN predicate that triggers its conversion to an IN subquery. Set to 0 to disable the conversion. See Conversion of Big IN Predicates Into Subqueries.
- Command line: `--in-predicate-conversion-threshold=#`
- Scope: Global, Session
- Dynamic: No
- Data Type: `numeric`

- Default Value: `1000`
- Range: `0` to `4294967295`

`in_transaction`

- Description: Session-only and read-only variable that is set to `1` if a transaction is in progress, `0` if not.
- Command line: No
- Scope: Session
- Dynamic: No
- Data Type: `boolean`
- Default Value: `0`

`init_connect`

- Description: String containing one or more SQL statements, separated by semicolons, that will be executed by the server for each client connecting. If there's a syntax error in the one of the statements, the client will fail to connect. For this reason, the statements are not executed for users with the SUPER privilege or, from MariaDB 10.5.2, the CONNECTION ADMIN privilege, who can then still connect and correct the error. See also init_file.
- Command line: `--init-connect=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`

`init_file`

- Description: Name of a file containing SQL statements that will be executed by the server on startup. Each statement should be on a new line, and end with a semicolon. See also init_connect.
- Command line: `init-file=file_name`
- Scope: Global
- Dynamic: No
- Data Type: `file name`

`insert_id`

- Description: Value to be used for the next statement inserting a new AUTO_INCREMENT value.
- Scope: Session
- Dynamic: Yes
- Data Type: `numeric`

`interactive_timeout`

- Description: Time in seconds that the server waits for an interactive connection (one that connects with the mysql_real_connect() CLIENT_INTERACTIVE option) to become active before closing it. See also wait_timeout.
- Command line: `--interactive-timeout=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `28800`
- Range: (Windows): `1` to `2147483`
- Range: (Other): `1` to `31536000`

`join_buffer_size`

- Description: Minimum size in bytes of the buffer used for queries that cannot use an index, and instead perform a full table scan. Increase to get faster full joins when adding indexes is not possible, although be aware of memory issues, since joins will always allocate the minimum size. Best left low globally and set high in sessions that require large full joins. In 64-bit platforms, Windows truncates values above 4GB to 4GB with a warning.
- Command line: `--join-buffer-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `262144` (256kB)
- Range (non-Windows): `128` to `18446744073709547520`
- Range (Windows): `8228` to `18446744073709547520`

`join_buffer_space_limit`

- Description: Maximum size in bytes of the query buffer, By default 1024_128_10.
- Command line: `--join-buffer-space-limit=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2097152`
- Range: `2048` to `18446744073709551615`

`join_cache_level`

- Description: Controls which of the eight block-based algorithms can be used for join operations.
  - 1 – flat (Block Nested Loop) BNL
  - 2 – incremental BNL
  - 3 – flat Block Nested Loop Hash (BNLH)
  - 4 – incremental BNLH
  - 5 – flat Batch Key Access (BKA)
  - 6 – incremental BKA
  - 7 – flat Batch Key Access Hash (BKAH)
  - 8 – incremental BKAH
- Command line: `--join-cache-level=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2`
- Range: `0` to `8`

`keep_files_on_create`

- Description: If a MyISAM table is created with no DATA DIRECTORY option, the .MYD file is stored in the database directory. When set to `0`, the default, if MariaDB finds another .MYD file in the database directory it will overwrite it. Setting this variable to `1` means that MariaDB will return an error instead, just as it usually does in the same situation outside of the database directory. The same applies for .MYI files and no INDEX DIRECTORY

option. This variable was deprecated in MariaDB 10.8.0, but that deprecation was reverted in MariaDB 13.0 and the variable is no longer deprecated.

- Command line: `--keep-files-on-create=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`large_files_support`

- Description: `ON` if the server if was compiled with large file support, else `OFF` .
- Scope: Global
- Dynamic: No

`large_page_size`

- Description: Indicates the size of memory page if large page support (Linux only) is enabled. The page size is determined from the Hugepagesize setting in `/proc/meminfo`. See large_pages. Deprecated and unused (since multiple page size support was added).
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0` (zero, from MariaDB 10.5.3)
- Deprecated: MariaDB 10.5.3
- Removed: MariaDB 12.0

`large_pages`

- Description: Whether or not large page support is used.
  - This is set with `--large-pages` or disabled with `--skip-large-pages`.
  - Large pages are used for the innodb buffer pool and for online DDL (of size 3* innodb_sort_buffer_size (or 6 when encryption is used)).
  - **Linux**: The `sysctl` variable `kernel.shmmax` must be larger than `llocation`, and the `sysctl` variable `vm.nr_hugepages` must be larger than the usage. The `ulimit` for locked memory must be sufficient to cover the amount used (`ulimit -l` and equivalent in `/etc/security/limits.conf/` or in systemd LimitMEMLOCK). If these operating system controls or insufficient free huge pages are available, the allocation of large pages falls back to conventional memory allocation, and a warning appears in the logs. Only allocations of the default `Hugepagesize` currently occur (see `/proc/meminfo`). The implementation supports multiple page sizes using the Linux built-in huge page feature with the enhancements available in the Linux kernel 3.8 and later.
    - To configure Linux to use huge pages, set the `hugepages` and `hugepagesz` kernel parameters, which can be done via GRUB_CMDLINE_LINUX.
    - To find possible huge page size values, issue this command: `ls -la /sys/devices/system/node/node0/hugepages/`
    - To determine how many huge pages are free, issue this command: `more /sys/devices/system/node/node*/hugepages/hugepages-*kB/free_hugepages`
  - **Windows**: The implementation uses the Windows `GetLargePageMinimum()` syscall to determine the supported page sizes.
    - This works only if the user that runs `mariadbd` has the `SeLockMemoryPrivilege` privilege.

    - **Solaris, FreeBSD, NetBSD**: The implementation uses the `getpagesizes()` syscall to determine the supported page sizes.
        - On Solaris, find possible huge page sizes by calling getpagessizes syscall.
    - **macOS, OpenBSD**: The implementation uses the `getpagesize()` syscall to determine the system's current page size.
- Command line: `--large-pages`, `--skip-large-pages`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`last_insert_id`

- Description: Contains the same value as that returned by LAST_INSERT_ID(). Note that setting this variable doesn't update the value returned by the underlying function.
- Scope: Session
- Dynamic: Yes
- Data Type: `numeric`

`lc_messages`

- Description: This system variable can be specified as a locale name. The language of the associated locale will be used for error messages. See Server Locales for a list of supported locales and their associated languages.
    - This system variable is set to `en_US` by default, which means that error messages are in English by default.
    - If this system variable is set to a valid locale name, but the server can't find an error message file for the language associated with the locale, then the default language will be used instead.
    - This system variable is used along with the lc_messages_dir system variable to construct the path to the error messages file.
    - See Setting the Language for Error Messages for more information.
- Command line: `--lc-messages=name`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `en_us`

`lc_messages_dir`

- Description: This system variable can be specified either as the path to the directory storing the server's error message files or as the path to the directory storing the specific language's error message file. See Server Locales for a list of available locales and their related languages.
    - The server initially tries to interpret the value of this system variable as a path to the directory storing the server's error message files. Therefore, it constructs the path to the language's error message file by concatenating the value of this system variable with the language name of the locale specified by the lc_messages system variable.
    - If the server does not find the error message file for the language, then it tries to interpret the value of this system variable as a direct path to the directory storing the specific language's error message file.
    - See Setting the Language for Error Messages for more information.
- Command line: `--lc-messages-dir=path`

- Scope: Global
- Dynamic: No
- Data Type: `directory name`

`lc_time_names`

- Description: The locale that determines the language used for the date and time functions DAYNAME(), MONTHNAME() and DATE_FORMAT(). Locale names are language and region subtags, for example 'en_ZA' (English - South Africa) or 'es_US: Spanish - United States'. The default is always 'en-US' regardless of the system's locale setting. See server locale for a full list of supported locales.
- Command line: `--lc-time-names=name`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `en_US`

`license`

- Description: Server license, for example `GPL`.
- Scope: Global
- Dynamic: No
- Data Type: `string`

`local_infile`

- Description: If set to `1`, LOCAL is supported for LOAD DATA INFILE statements. If set to `0`, usually for security reasons, attempts to perform a LOAD DATA LOCAL will fail with an error message.
- Command line: `--local-infile=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`lock_wait_timeout`

- Description: Timeout in seconds for attempts to acquire metadata locks. Statements using metadata locks include FLUSH TABLES WITH READ LOCK, LOCK TABLES, HANDLER and DML and DDL operations on tables, stored procedures and functions, and views. The timeout is separate for each attempt, of which there may be multiple in a single statement. `0` means no wait. See WAIT and NOWAIT.
- Command line: `--lock-wait-timeout=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `86400` (1 day)
- Range:
  - `0` to `31536000`

`locked_in_memory`

- Description: Indicates whether –memlock was used to lock mariadbd in memory.

- Command line: `--memlock`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`log`

- Description: Deprecated and removed in MariaDB 10.0, use general_log instead.
- Command line: `-l [filename]` or `--log[=filename]`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `OFF`
- Removed: MariaDB 10.0

`log_disabled_statements`

- Description: If set, the specified type of statements (slave and/or stored procedure statements) will not be logged to the general log. Multiple values are comma-separated, without spaces.
- Command line: `--log-disabled_statements=value`
- Scope: Global, Session
- Dynamic: No
- Data Type: `set`
- Default Value: `sp`
- Valid Values: `slave` and/or `sp`, or empty string for none

`log_error`

- Description: Specifies the name of the error log. If –console is specified later in the configuration (Windows only) or this option isn't specified, errors will be logged to stderr. If no name is provided, errors will still be logged to `hostname.err` in the `datadir` directory by default. If a configuration file sets `--log-error`, one can reset it with `--skip-log-error` (useful to override a system wide configuration file). MariaDB always writes its error log, but the destination is configurable. See error log for details. Note that if –log-basename is also set, `log_error` should be placed after in the config files. Later settings override earlier settings, so `log-basename` will override any earlier log file name settings.
- Command line: `--log-error[=name]`, `--skip-log-error`
- Scope: Global
- Dynamic: No
- Data Type: `file name`
- Default Value: (empty string)

`log_output`

- Description: How the output for the general query log and the slow query log is stored. By default, written to file (`FILE`), it can also be stored in the general_log and slow_log tables in the mysql database (`TABLE`) or not stored at all (`NONE`). More than one option can be chosen at the same time, with `NONE` taking precedence if present. Logs will not be written if logging is not enabled. See Writing logs into tables, and the slow_query_log and general_log server system variables.
- Command line: `--log-output=name`

- Scope: Global
- Dynamic: Yes
- Data Type: `set`
- Default Value: `FILE`
- Valid Values: `TABLE`, `FILE` or `NONE`

`log_queries_not_using_indexes`

- Description: Queries that don't use an index, or that perform a full index scan where the index doesn't limit the number of rows, will be logged to the slow query log (regardless of time taken). The slow query log needs to be enabled for this to have an effect. Mapped to `log_slow_filter='not_using_index'` from MariaDB 10.3.1.
- Command line: `--log-queries-not-using-indexes`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`log_slow_admin_statements`

- Description: Log slow OPTIMIZE, ANALYZE, ALTER and other administrative statements to the slow log if it is open. See also log_slow_disabled_statements and log_slow_filter. Deprecated, use log_slow_filter without `admin`.
- Command line: `--log-slow-admin-statements`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value:
  - `ON`
- Deprecated: MariaDB 11.0.1

`log_slow_disabled_statements`

- Description: If set, the specified type of statements will not be logged to the slow query log. See also log_slow_admin_statements and log_slow_filter.
- Command line: `--log-slow-disabled_statements=value`
- Scope: Global, Session
- Dynamic: No
- Data Type: `set`
- Default Value: `sp`
- Valid Vales: `admin`, `call`, `slave` and/or `sp`

`log_slow_filter`

- Description: Comma-delimited string (without spaces) containing one or more settings for filtering what is logged to the slow query log. If a query matches one of the types listed in the filter, and takes longer than long_query_time, it will be logged (except for 'not_using_index' which is always logged if enabled, regardless of the time). Sets log-slow-admin-statements to ON. See also log_slow_disabled_statements.
  - `admin` log administrative queries (create, optimize, drop etc…)
  - `filesort` logs queries that use a filesort.
  - `filesort_on_disk` logs queries that perform a filesort on disk.

    - `filesort_priority_queue`
    - `full_join` logs queries that perform a join without indexes.
    - `full_scan` logs queries that perform full table scans.
    - `not_using_index` logs queries that don't use an index, or that perform a full index scan where the index doesn't limit the number of rows. Disregards long_query_time, unlike other options. log_queries_not_using_indexes maps to this option. From MariaDB 10.3.1.
    - `query_cache` log queries that are resolved by the query cache.
    - `query_cache_miss` logs queries that are not found in the query cache.
    - `tmp_table` logs queries that create an implicit temporary table.
    - `tmp_table_on_disk` logs queries that create a temporary table on disk.
- Command line: `log-slow-filter=value1[,value2...]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value:
    - `admin`, `filesort`, `filesort_on_disk`, `filesort_priority_queue`, `full_join`, `full_scan`, `query_cache`, `query_cache_miss`, `tmp_table`, `tmp_table_on_disk`
- Valid Values:
    - `admin`, `filesort`, `filesort_on_disk`, `filesort_priority_queue`, `full_join`, `full_scan`, `not_using_index`, `query_cache`, `query_cache_miss`, `tmp_table`, `tmp_table_on_disk`

`log_slow_max_warnings`

- Description: Max numbers of warnings printed to slow query log per statement
- Command line: `log-slow-max-warnings=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10`
- Range: `0` to `1000`
- Introduced: MariaDB 10.6.16, MariaDB 10.10.7, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3

`log_slow_min_examined_row_limit`

- Description: Don't write queries to slow query log that examine fewer rows than the set value. If set to `0`, the default, no row limit is used. `min_examined_row_limit` is an alias. From MariaDB 11.7, queries slower than log_slow_always_query_time will always be logged.
- Command line: `--log-slow-min-examined-row-limit=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0-4294967295`
- Introduced: MariaDB 10.11

`log_slow_queries`

- Description: Deprecated and removed in MariaDB 10.0, use slow_query_log instead.

- Command line: `--log-slow-queries[=name]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Removed: MariaDB 10.0

`log_slow_query`

- Description: If set to 0, the default unless the –slow-query-log option is used, the slow query log is disabled, while if set to 1 (both global and session variables), the slow query log is enabled. Named slow_query_log before MariaDB 10.11.0, which is now an alias.
- Command line: `--slow-query-log`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`
- Introduced: MariaDB 10.11.0
- See also: See log_output to see how log files are written. If that variable is set to `NONE`, no logs will be written even if log_slow_query is set to `1`.

`log_slow_query_file`

- Description: Name of the slow query log file. Before MariaDB 10.11, was named slow_query_log_file. This was named `log_slow_query_file_name` in the MariaDB 10.11.0 preview release. If –log-basename is also set, `log_slow_query_file` should be placed after in the config files. Later settings override earlier settings, so `log-basename` will override any earlier log file name settings.
- Command line: `--log-slow-query-file=file_name`
- Scope: Global
- Dynamic: Yes
- Data Type: `file name`
- Default Value: `host_name-slow.log`
- Introduced: MariaDB 10.11.0

`log_slow_query_time`

- Description: If a query takes longer than this many seconds to execute (microseconds can be specified too), the Slow_queries status variable is incremented and, if enabled, the query is logged to the slow query log. Before MariaDB 10.11, was named long_query_time. Affected by log_slow_rate_limit and log_slow_min_examined_row_limit.
- Command line: `--log-slow-query-time=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10.000000`
- Range: `0` to `31536000`
- Introduced: MariaDB 10.11.0

`log_slow_rate_limit`

- Description: The slow query log will log every this many queries. The default is `1`, or every query, while setting it to `20` would log every 20 queries, or five percent. Aims to reduce I/O usage and excessively large slow query logs. See also Slow Query Log Extended Statistics. From MariaDB 11.7, queries slower than log_slow_always_query_time will always be logged.
- Command line: `log-slow-rate-limit=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` upwards

`log_slow_verbosity`

- Description: Controls information to be added to the slow query log. Options are added in a comma-delimited string. See also Slow Query Log Extended Statistics. log_slow_verbosity is not supported when log_output='TABLE'.
  - `query_plan` logs query execution plan information
  - `innodb` Alias to `engine` (from MariaDB 10.6.15 and MariaDB 10.11.5), previously ignored.
  - `explain` prints EXPLAIN output in the slow query log. See EXPLAIN in the Slow Query Log.
  - `engine` Logs engine statistics (from MariaDB 10.6.15 and MariaDB 10.11.5).
  - `warnings` Print all errors, warnings and notes for the statement to the slow query log. (from MariaDB 10.6.16).
  - `all` Enables all above options (From MariaDB 10.6.16)
  - `full` Enables all above options.
- Command line: `log-slow-verbosity=value1[,value2...]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: (Empty)
- Valid Values:
  - > = MariaDB 10.6.16, MariaDB 10.11.6: (Empty), `query_plan`, `innodb`, `explain`, `engine`, `warnings`, `all`, `full`
  - > = MariaDB 10.6.15, MariaDB 10.11.5: (Empty), `query_plan`, `innodb`, `explain`, `engine`, `full`
  - <= MariaDB 10.6.14, MariaDB 10.11.4: (Empty), `query_plan`, `innodb`, `explain`

`log_tc_size`

- Description: Defines the size in bytes of the memory-mapped file-based transaction coordinator log, which is only used if the binary log is disabled. If you have two or more XA-capable storage engines enabled, then a transaction coordinator log must be available. This size is defined in multiples of 4096. See Transaction Coordinator Log for more information. Also see the –log-tc server option and the –tc-heuristic-recover option.
- Command line: `log-tc-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`

- Default Value: `24576`
- Range: `12288` to `18446744073709551615`

`log_warnings`

- Description: Determines which additional warnings are logged. Setting to `0` disables additional warning logging. Note that this does not prevent all warnings, there is a core set of warnings that will always be written to the error log. The additional warnings are as follows:
  - log_warnings = 0
    - No notes from the event scheduler, only warnings and errors (from MariaDB 10.11.16, 11.4.10, 11.8.6, 12.2.2)
  - log_warnings >= 1
    - Only startup and shutdown event scheduler notes (from MariaDB 10.11.16, 11.4.10, 11.8.6, 12.2.2)
    - Event scheduler information.
    - System signals.
    - Wrong usage of `--user`.
    - Failed `setrlimit()` and `mlockall()`.
    - Changed limits.
    - Wrong values of `lower_case_table_names` and `stack_size`.
    - Wrong values for command line options.
    - Start log position and some master information when starting slaves.
    - Slave reconnects.
    - Killed slaves.
    - Error reading relay logs.
    - Unsafe statements for statement-based replication. If this warning occurs frequently, it is throttled to prevent flooding the log.
    - Disabled plugins that one tried to enable or use.
    - UDF files that didn't include the required init functions.
    - DNS lookup failures.
  - log_warnings >= 2
    - Access denied errors.
    - Connections aborted or closed due to errors or timeouts.
    - Table handler errors.
    - Messages related to the files used to persist replication state:
      - Either the default `master.info` file or the file that is configured by the master_info_file option.
      - Either the default `relay-log.info` file or the file that is configured by the relay_log_info_file system variable.
    - Information about a master's binary log dump thread.
  - log_warnings >= 3
    - All errors and warnings during MyISAM repair and auto recover.
    - Information about old-style language options.
    - Information about progress of InnoDB online DDL.
  - log_warnings >=4
    - Connections aborted due to "Too many connections" errors.
    - Connections closed normally without authentication.
    - Connections aborted due to KILL.

        - Connections closed due to released connections, such as when completion_type is set to `RELEASE`.
        - Could not read packet: (a lot more information)
        - All read/write errors for a connection are logged to the error log.
    - log_warnings >=9
        - Information about initializing plugins.

- Command line: `-W [level]` or `--log-warnings[=level]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2`
- Range: `0` to `4294967295`

`long_query_time`

- Description: If a query takes longer than this many seconds to execute (microseconds can be specified too), the Slow_queries status variable is incremented and, if enabled, the query is logged to the slow query log. From MariaDB 10.11.0, this is an alias for log_slow_query_time.
- Command line: `--long-query-time=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10.000000`
- Range: `0` upwards

`low_priority_updates`

- Description: If set to 1 (0 is the default), for storage engines that use only table-level locking (Aria, MyISAM, MEMORY and MERGE), all INSERTs, UPDATEs, DELETEs and LOCK TABLE WRITEs will wait until there are no more SELECTs or LOCK TABLE READs pending on the relevant tables. Set this to 1 if reads are prioritized over writes.
    - In MariaDB 5.5 and earlier, sql_low_priority_updates is a synonym.
- Command line: `--low-priority-updates`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`

`lower_case_file_system`

- Description: Read-only variable describing whether the file system is case-sensitive. If set to `OFF`, file names are case-sensitive. If set to `ON`, they are not case-sensitive.
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `##`

`lower_case_table_names`

- Description: If set to `0` (the default on Unix-based systems), table names and aliases and database names are compared in a case-sensitive manner. If set to `1` (the default on Windows), names are stored in lowercase and not compared in a case-sensitive manner. If set to `2` (the default on Mac OS X), names are stored as declared but compared in lowercase.
  This system variable's value cannot be changed after the datadir has been initialized. lower_case_table_names is set when a MariaDB instance starts, and it remains constant afterwards.
- Command line: `--lower-case-table-names[=#]`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0` (Unix), `1` (Windows), `2` (Mac OS X)
- Range: `0` to `2`

`max_allowed_packet`

- Description: Maximum size in bytes of a packet or a generated/intermediate string. The packet message buffer is initialized with the value from net_buffer_length, but can grow up to max_allowed_packet bytes. Set as large as the largest BLOB, in multiples of 1024. If this value is changed, it should be changed on the client side as well. See slave_max_allowed_packet for a specific limit for replication purposes.
- Command line: `--max-allowed-packet=#`
- Scope: Global, Session
- Dynamic: Yes (Global), No (Session)
- Data Type: `numeric`
- Default Value:
  - `16777216` (16M)
  - `1073741824` (1GB) (client-side)
- Range: `1024` to `1073741824`

`max_connect_errors`

- Description: Limit to the number of successive failed connects from a host before the host is blocked from making further connections. The count for a host is reset to zero if they successfully connect. To unblock, flush the host cache with a FLUSH HOSTS statement or mariadb-admin flush-hosts. The performance_schema.host_cache table contains the status of the current hosts.
- Command line: `--max-connect-errors=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `1` to `4294967295`

`max_connections`

- Description: The maximum number of simultaneous client connections. See also Handling Too Many Connections. Note that this value affects the number of file descriptors required on the operating system. Minimum was changed from `1` to `10` to avoid possible unexpected results for the user (MDEV-18252). Note that MariaDB always has one reserved connection for a `SUPER` (or `CONNECTION ADMIN` user). Additionally, it can listen on a separate port, so will be available even when the max_connections limit is reached.

- Command line: `--max-connections=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `151`
- Range: `10` to `100000`

> **Note:**
>
> **Systemd thread limit (MDEV-30236)**
> When running MariaDB under systemd, be aware that systemd's default `TasksMax` (≈ 4,915 tasks per service) may prevent reaching high `max_connections` values—even if configured higher. Starting with MariaDB 10.4.33, the systemd unit includes:
>
> ```
> [Service]
> TasksMax=infinity
> ```
>
> This setting removes the artificial cap, allowing `max_connections` to scale per your configuration (subject to OS memory and thread limits).

`max_delayed_threads`

- Description: Limits to the number of INSERT DELAYED threads. Once this limit is reached, the insert is handled as if there was no DELAYED attribute. If set to `0`, DELAYED is ignored entirely. The session value can only be set to `0` or to the same as the global value.
- Command line: `--max-delayed-threads=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `20`
- Range: `0` to `16384`

`max_digest_length`

- Description: Maximum length considered for computing a statement digest, such as used by the Performance Schema and query rewrite plugins. Statements that differ after this many bytes produce the same digest, and are aggregated for statistics purposes. The variable is allocated per session. Increasing will allow longer statements to be distinguished from each other, but increase memory use, while decreasing will reduce memory use, but more statements may become indistinguishable.
- Command line: `--max-digest-length=#`
- Scope: Global,
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1024`
- Range: `0` to `1048576`

`max_error_count`

- Description: Specifies the maximum number of messages stored for display by SHOW ERRORS and SHOW WARNINGS statements.
- Command line: `--max-error-count=#`

- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `64`
- Range: `0` to `65535`

`max_heap_table_size`

- Description: Maximum size in bytes for user-created MEMORY tables. Setting the variable while the server is active has no effect on existing tables unless they are recreated or altered. The smaller of max_heap_table_size and tmp_table_size also limits internal in-memory tables. When the maximum size is reached, any further attempts to insert data will receive a “table … is full” error. Temporary tables created with CREATE TEMPORARY will not be converted to Aria, as occurs with internal temporary tables, but will also receive a table full error.
- Command line: `--max-heap-table-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `16777216`
- Range : `16384` to `4294966272`

`max_insert_delayed_threads`

- Description: Synonym for max_delayed_threads.

`max_join_size`

- Description: Statements will not be performed if they are likely to need to examine more than this number of rows, row combinations or do more disk seeks. Can prevent poorly-formatted queries from taking server resources. Changing this value to anything other the default will reset sql_big_selects to 0. If sql_big_selects is set again, max_join_size will be ignored. This limit is also ignored if the query result is sitting in the query cache. Previously named sql_max_join_size, which is still a synonym.
- Command line: `--max-join-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `18446744073709551615`
- Range: `1` to `18446744073709551615`

`max_length_for_sort_data`

- Description: Used to decide which algorithm to choose when sorting rows. If the total size of the column data, not including columns that are part of the sort, is less than `max_length_for_sort_data`, then we add these to the sort key. This can speed up the sort as we don’t have to re-read the same row again later. Setting the value too high can slow things down as there will be a higher disk activity for doing the sort.
- Command line: `--max-length-for-sort-data=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1024`
- Range: `4` to `8388608`

`max_long_data_size`

- Description: Maximum size for parameter values sent with mysql_stmt_send_long_data(). If not set, will default to the value of max_allowed_packet. Deprecated in MariaDB 5.5 and removed in MariaDB 10.5.0; use max_allowed_packet instead.
- Command line: `--max-long-data-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `16777216` (16M)
- Range: `1024` to `4294967295`
- Deprecated: MariaDB 5.5
- Removed: MariaDB 10.5.0

`max_open_cursors`

- Description: The maximum number of open cursors allowed per session.
- Command line: `--max-open-cursors=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `50`
- Range: `0` to 65536
- Introduced: MariaDB 12.0

`max_password_errors`

- Description: The maximum permitted number of failed connection attempts due to an invalid password before a user is blocked from further connections. FLUSH_PRIVILEGES will permit the user to connect again. This limit is not applicable for users with the SUPER privilege or, from MariaDB 10.5.2, the CONNECTION ADMIN privilege, with a hostname of localhost, 127.0.0.1 or ::1. See also the Information Schema USERS table.
- Command line: `--max-password-errors=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4294967295`
- Range: `1` to `4294967295`

`max_prepared_stmt_count`

- Description: Maximum number of prepared statements on the server. Can help prevent certain forms of denial-of-service attacks. If set to `0`, no prepared statements are permitted on the server.
- Command line: `--max-prepared-stmt-count=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `16382`
- Range: `0` to `4294967295`

`max_recursive_iterations`

- Description: Maximum number of iterations when executing recursive queries, used to prevent infinite loops in recursive CTEs.
- Command line: `--max-recursive-iterations=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1000` (>= MariaDB 10.6.0), `4294967295` (<= MariaDB 10.5)
- Range: `0` to `4294967295`

`max_rowid_filter_size`

- Description: The maximum size of the container of a rowid filter.
- Command line: `--max-rowid-filter-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `131072`
- Range: `1024` to `18446744073709551615`

`max_seeks_for_key`

- Description: The optimizer assumes that the number specified here is the most key seeks required when searching with an index, regardless of the actual index cardinality. If this value is set lower than its default and maximum, indexes will tend to be preferred over table scans.
- Command line: `--max-seeks-for-key=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4294967295`
- Range: `1` to `4294967295`

`max_session_mem_used`

- Description: Amount of memory a single user session is allowed to allocate. This limits the value of the session variable Memory_used.
- Command line: `--max-session-mem-used=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `9223372036854775807` (8192 PB)
- Range: `8192` to `18446744073709551615`

`max_sort_length`

- Description: Maximum size in bytes used for sorting data values - anything exceeding this is ignored. The server uses only the first `max_sort_length` bytes of each value and ignores the rest. Increasing this may require sort_buffer_size to be increased (especially if ER_OUT_OF_SORTMEMORY errors start appearing). From MariaDB 11.7, a warning is generated when max_sort_length is exceeded.

- Command line: `--max-sort-length=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1024`
- Range:
  - `4` to `8388608` (<= MariaDB 10.4.13, MariaDB 10.5.3)
  - `8` to `8388608` (>= MariaDB 10.4.14, MariaDB 10.5.4)
  - `64` to `8388608` (>= MariaDB 10.4.16, MariaDB 10.5.7)

`max_sp_recursion_depth`

- Description: Permitted number of recursive calls for a stored procedure. `0`, the default, no recursion is permitted. Increasing this value increases the thread stack requirements, so you may need to increase thread_stack as well. This limit doesn't apply to stored functions.
- Command line: `--max-sp-recursion-depth[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `255`

`max_statement_time`

- Description: Maximum time in seconds that a query can execute before being aborted. This includes all queries, not just SELECT statements, but excludes statements in stored procedures. If set to 0, no limit is applied. See Aborting statements that take longer than a certain time to execute for details and limitations. Useful when combined with SET STATEMENT for limiting the execution times of individual queries. Replicas are not affected by this variable, however, from MariaDB 10.10, there's slave_max_statement_time that sets the limit to abort queries on a replica.
- Command line: `--max-statement-time[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0.000000`
- Range: `0` to `31536000`

`max_tmp_tables`

- Description: Unused.
- Removed: MariaDB 11.3.0

`max_user_connections`

- Description:
  Maximum simultaneous connections permitted for each user account. When set to `0`, there is no per user limit. Setting it to `-1` stops users without the SUPER privilege or, from MariaDB 10.5.2, the CONNECTION ADMIN privilege, from connecting to the server. The session variable is always read-only, and only privileged users can modify user limits. The session variable defaults to the global `max_user_connections` variable, unless the user's specific MAX_USER_CONNECTIONS resource option is non-zero. When both global variable and the

user resource option are set, the user's MAX_USER_CONNECTIONS is used. Note: This variable does not affect users with the SUPER privilege or, from MariaDB 10.5.2, the CONNECTION ADMIN privilege.

- Command line: `--max-user-connections=#`
- Scope: Global, Session
- Dynamic: Yes, (except when globally set to `0` or `-1`)
- Data Type: `numeric`
- Default Value: `0`
- Range: `-1` to `4294967295`

`max_write_lock_count`

- Description: Read lock requests will be permitted for processing after this many write locks. Applies only to storage engines that use table level locks (thr_lock), so no effect with InnoDB or Archive.
- Command line: `--max-write-lock-count=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `4294967295`
- Range: `1` to `4294967295`

`metadata_locks_cache_size`

- Description: Unused since 10.1.4
- Command line: `--metadata-locks-cache-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1024`
- Range: `1` to `1048576`

`metadata_locks_hash_instances`

- Description: Unused since 10.1.4
- Command line: `--metadata-locks-hash-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `8`
- Range: `1` to `1024`

`metadata_locks_instances`

- Description: Number of fast lanes to create for metadata locks. Can be used to improve DML scalability by eliminating MDL_lock::rwlock load. Use 1 to disable MDL fast lanes. Supported MDL namespaces: BACKUP.
- Command line: `--metadata-locks-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `8`

- Range: `1` to `256`
- Introduced: MariaDB 12.1

`min_examined_row_limit`

- Description: Don't write queries to slow query log that examine fewer rows than the set value. If set to `0`, the default, no row limit is used. From MariaDB 10.11.0, this is an alias for log_slow_min_examined_row_limit.
- Command line: `--min-examined-row-limit=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0-4294967295`

`mrr_buffer_size`

- Description: Size of buffer to use when using multi-range read with range access. See Multi Range Read optimization for more information.
- Command line: `--mrr-buffer-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `262144`
- Range `8192` to `2147483647`

`multi_range_count`

- Description: Ignored. Use mrr_buffer_size instead.
- Command line: `--multi-range-count=#`
- Default Value: `256`
- Removed: MariaDB 10.5.1

`mysql56_temporal_format`

- Description: If set (the default), MariaDB uses the MySQL 5.6 low level formats for TIME, DATETIME and TIMESTAMP instead of the MariaDB 5.3 version. The version MySQL introduced in 5.6 requires more storage, but potentially allows negative dates and has some advantages in replication. There should be no reason to revert to the old MariaDB 5.3 microsecond format. See also MDEV-10723.
- Command line: `--mysql56-temporal-format`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`named_pipe`

- Description: On Windows systems, determines whether connections over named pipes are permitted.
- Command line: `--named-pipe`
- Scope: Global
- Dynamic: No

- Data Type: `boolean`
- Default Value: `OFF`

`net_buffer_length`

- Description: The starting size, in bytes, for the connection and thread buffers for each client thread. The size can grow to max_allowed_packet. This variable's session value is read-only. Can be set to the expected length of client statements if memory is a limitation.
- Command line: `--net-buffer-length=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `16384`
- Range: `1024` to `1048576`

`net_read_timeout`

- Description: Time in seconds the server will wait for a client connection to send more data before aborting the read. See also net_write_timeout and slave_net_timeout
- Command line: `--net-read-timeout=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `30`
- Range: `1` to `31536000`

`net_retry_count`

- Description: Permit this many retries before aborting when attempting to read or write on a communication port. On FreeBSD systems should be set higher as threads are sent internal interrupts..
- Command line: `--net-retry-count=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10`
- Range: `1` to `4294967295`

`net_write_timeout`

- Description: Time in seconds to wait on writing a block to a connection before aborting the write. See also net_read_timeout and slave_net_timeout.
- Command line: `--net-write-timeout=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `60`
- Range: `1` upwards

`new_mode`

- Description: Used to enable new behavior in otherwise stable versions. See NEW Mode. Non-default NEW_MODE options are by design deprecated and will eventually be removed.
- Command line: `--new-mode`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `(empty string)`
- Valid Values: See NEW Mode for the full list.

`note_verbosity`

- Description: Verbosity level for note-warnings given to the user. Options are added in a comma-delimited string, except for `all`, which sets all options. Be aware that if the old sql_notes variable is 0, one will not get any notes. Setting `note_verbosity` to "" is the recommended way to disable notes.
  - `basic` All old notes.
  - `unusable_keys` Give warnings for unusable keys for SELECT, DELETE and UPDATE.
  - `explain` Give warnings for unusable keys for EXPLAIN.
  - `all` Enables all above options. This has to be given alone.
- Command line: `note-verbosity=value1[,value2...]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `basic,explain`
- Valid Values: `basic,explain,unusable_keys` or `all`.
- Introduced: MariaDB 10.6.16

`old`

- Description: Disabled by default, enabling it reverts index hints to those used before MySQL 5.1.17. Enabling may lead to replication errors. Deprecated and replaced by old_mode from MariaDB 10.9.
- Command line: `--old`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 10.9
- Removed: MariaDB 13.0. Use old_mode instead.

`old_alter_table`

- Description: From MariaDB 10.3.7, an alias for alter_algorithm. Prior to that, if set to `1` (`0` is default), MariaDB reverts to the non-optimized, pre-MySQL 5.1, method of processing ALTER TABLE statements. A temporary table is created, the data is copied over, and then the temporary table is renamed to the original.
- Command line: `--old-alter-table`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumerated` (>=MariaDB 10.3.7)

- Default Value: See alter_algorithm
- Valid Values: See alter_algorithm for the full list.
- Deprecated: MariaDB 10.3.7 (superseded by alter_algorithm)
- Removed: MariaDB 11.2.0

`old_mode`

- Description: Used for getting MariaDB to emulate behavior from an old version of MySQL or MariaDB. See OLD Mode. Fully replaces the old variable from MariaDB 10.9. Non-default OLD_MODE options are by design deprecated and will eventually be removed.
- Command line: `--old-mode`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `(empty string)` (>= MariaDB 13.1), `UTF8_IS_UTF8MB3` (MariaDB 10.6 to MariaDB 13.0), `(empty string)` (<= MariaDB 10.5)
- Valid Values: See OLD Mode for the full list.

`old_passwords`

- Description: If set to `1` (`0` is default), MariaDB reverts to using the mysql_old_password authentication plugin by default for newly created users and passwords, instead of the mysql_native_password authentication plugin.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`open_files_limit`

- Description: The number of file descriptors available to MariaDB. If you are getting the `Too many open files` error, then you should increase this limit. If set to 0, then MariaDB will calculate a limit based on the following:

MAX(max_connections*5, max_connections +table_open_cache*2)

MariaDB sets the limit with setrlimit. MariaDB cannot set this to exceed the hard limit imposed by the operating system. Therefore, you may also need to change the hard limit. There are a few ways to do so.

- If you are using mariadbd_safe to start `mariadbd`, then see the instructions at mariadbd_safe: Configuring the Open Files Limit.
- If you are using systemd to start `mariadbd`, then see the instructions at systemd: Configuring the Open Files Limit.
- Otherwise, you can change the hard limit for the `mysql` user account by modifying /etc/security/limits.conf. See Configuring Linux for MariaDB: Configuring the Open Files Limit for more details.
- Command line: `--open-files-limit=count`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: Autosized (see description)
- Range: `0` to `4294967295`

`optimizer_extra_pruning_depth`

- Description:If the optimizer needs to enumerate a join prefix of this size or larger, then it will try aggressively prune away the search space.
- Command line: `--optimizer-extra-pruning-depth[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `8`
- Range: `0` to `62`
- Introduced: MariaDB 10.10.1

optimizer_join_limit_pref_ratio

- Description:Controls the optimizer_join_limit_pref_ratio optimization.
- Command line: `--optimizer-join-limit-pref-ratio[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0` (Disable)
- Range: `0` to `4294967295`
- Introduced: MariaDB 10.6.20, MariaDB 10.11.10, MariaDB 11.2.6, MariaDB 11.4.4, MariaDB 11.6.2

`optimizer_max_sel_arg_weight`

- Description: This is an actively enforced maximum effective SEL_ARG tree weight limit. A SEL_ARG weight is the number of effective “ranges” hanging off this root (that is, merged tree elements are “unmerged” to count the weight). During range analysis, looking for possible index merges, SEL_ARG graphs related to key ranges in query conditions are being processed. Graphs exceeding this limit will stop keys being ‘and’ed and ’or’ed together to form a new larger SEL_ARG graph. After each ’and’ or ‘or’ process, this maximum weight limit is enforced. It enforces this limit by pruning the key part being used. This key part pruning can be used to limit/disable index merge SEL_ARG graph construction on overly long query conditions.
- Command line: `--optimizer-max-sel-arg-weight=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `32000`
- Range: `0` to `18446744073709551615`
- Introduced: MariaDB 10.5.9

`optimizer_max_sel_args`

- Description: The maximum number of SEL_ARG objects created when optimizing a range. If more objects would be needed, range scans will not be used by the optimizer.
- Command line: `--optimizer-max-sel-args=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `16000`
- Range: `0` to `4294967295`

- Introduced: MariaDB 10.6.16, MariaDB 10.10.7, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3

`optimizer_prune_level`

- Description:Controls the heuristic(s) applied during query optimization to prune less-promising partial plans from the optimizer search space.
  - `0` : heuristics are disabled and an exhaustive search is performed
  - `1` : the optimizer will use heuristics to prune less-promising partial plans from the optimizer search space
  - `2` : tables using EQ_REF will be joined together as 'one entity' and the different combinations of these tables will not be considered (from MariaDB 10.10)
- Command line: `--optimizer-prune-level[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2` (>= MariaDB 10.10), `1` (<= MariaDB 10.9)

`optimizer_search_depth`

- Description: Maximum search depth by the query optimizer. Smaller values lead to less time spent on execution plans, but potentially less optimal results. If set to `0`, MariaDB will automatically choose a reasonable value. Since the better results from more optimal planning usually offset the longer time spent on planning, this is set as high as possible by default. `63` is a valid value, but its effects (switching to the original find_best search) are deprecated.
- Command line: `--optimizer-search-depth[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `62`
- Range: `0` to `63`

`optimizer_selectivity_sampling_limit`

- Description: Controls number of record samples to check condition selectivity. Only used if `[optimizer_use_condition_selectivity](#pg-mization-and-tuning-system-variables-server-system-variables-e441845d) > 4.`
- Command line: `optimizer-selectivity-sampling-limit[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `10` upwards

`optimizer_switch`

- Description: A series of flags for controlling the query optimizer. See Optimizer Switch for defaults, and a comparison to MySQL.
- Command line: `--optimizer-switch=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`

- Valid Values:
  - `condition_pushdown_for_derived={on|off}`
  - `condition_pushdown_for_subquery={on|off}`
  - `condition_pushdown_from_having={on|off}`
  - `cset_narrowing={on|off}` - see Charset Narrowing Optimization (>= MariaDB 10.6.16, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3 and MariaDB 11.2.2)
  - `default` - set all optimizations to their default values.
  - `derived_merge={on|off}` - see Derived table merge optimization
  - `derived_with_keys={on|off}` - see Derived table with key optimization
  - `duplicateweedout={on|off}`. From MariaDB 12.0.
  - `engine_condition_pushdown={on|off}`. Deprecated in MariaDB 10.1.1 as engine condition pushdown is now automatically enabled for all engines that support it.
  - `exists_to_in={on|off}` - see EXISTS-to-IN optimization
  - `extended_keys={on|off}` - see Extended Keys
  - `firstmatch={on|off}` - see First Match Strategy
  - `hash_join_cardinality={on|off}` - see hash_join_cardinality-optimizer_switch-flag (>= MariaDB 11.0.2, MariaDB 10.11.3, MariaDB 10.6.13)
  - `index_condition_pushdown={on|off}` - see Index Condition Pushdown
  - `index_merge={on|off}`
  - `index_merge_intersection={on|off}`
  - `index_merge_sort_intersection={on|off}` - more details
  - `index_merge_sort_union={on|off}`
  - `index_merge_union={on|off}`
  - `in_to_exists={on|off}` - see IN-TO-EXISTS transformation
  - `join_cache_bka={on|off}` - see Block-Based Join Algorithms
  - `join_cache_hashed={on|off}` - see Block-Based Join Algorithms
  - `join_cache_incremental={on|off}` - see Block-Based Join Algorithms
  - `loosescan={on|off}` - see LooseScan strategy
  - `materialization={on|off}` - Semi-join and non semi-join materialization.
  - `mrr={on|off}` - see Multi Range Read optimization
  - `mrr_cost_based={on|off}` - see Multi Range Read optimization
  - `mrr_sort_keys={on|off}` - see Multi Range Read optimization
  - `not_null_range_scan={on|off}` - see not_null_range_scan optimization (>= MariaDB 10.5.0)
  - `optimize_join_buffer_size={on|off}` - see Block-Based Join Algorithms
  - `orderby_uses_equalities={on|off}` - if not set, the optimizer ignores equality propagation. See MDEV-8989.
  - `outer_join_with_cache={on|off}` - see Block-Based Join Algorithms
  - `partial_match_rowid_merge={on|off}` - see Non-semi-join subquery optimizations
  - `partial_match_table_scan={on|off}` - see Non-semi-join subquery optimizations
  - `reorder_outer_joins` - see reorder_outer_joins (MariaDB 12.3)
  - `rowid_filter={on|off}` - see Rowid Filtering Optimization
  - `sargable_casefold={on|off}` (>= MariaDB 11.3.0)
  - `semijoin={on|off}` - see Semi-join subquery optimizations
  - `semijoin_with_cache={on|off}` - see Block-Based Join Algorithms
  - `split_materialized={on|off}`

- `subquery_cache={on|off}` - see subquery cache.
- `table_elimination={on|off}` - see Table Elimination User Interface

`optimizer_record_context`

- Description: Controls storing of optimizer context of all tables that are referenced in a query.
- Command line: `--optimizer-record-context{=0|1}`
- Scope: Session
- Dynamic: Yes
- Data Type: boolean
- Default Value: `OFF`
- Introduced: MariaDB 12.1

`optimizer_trace`

- Description: Controls tracing of the optimizer: optimizer_trace=option=val[,option=val…], where option is one of {enabled} and val is one of {on, off, default}
- Command line: `--optimizer-trace=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `enabled=off`
- Valid Values: `enabled={on|off|default}`

`optimizer_trace_max_mem_size`

- Description: Limits the memory used while tracing a query by specifying the maximum allowed cumulated size, in bytes, of stored optimizer traces.
- Command line: `--optimizer-trace-max-mem-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1048576`
- Range: `1` to `18446744073709551615`

`optimizer_use_condition_selectivity`

- Description: Controls which statistics can be used by the optimizer when looking for the best query execution plan. In most cases, the default value, `4` will be suitable. However, if you are hitting some of the rare cases where this does not work well (see MDEV-23707), you can usually work around this by setting this variable to `1`.
  - `1` Use selectivity of predicates as in MariaDB 5.5.
  - `2` Use selectivity of all range predicates supported by indexes.
  - `3` Use selectivity of all range predicates estimated without histogram.
  - `4` Use selectivity of all range predicates estimated with histogram.
  - `5` Additionally use selectivity of certain non-range predicates calculated on record sample.
- Command line: `--optimizer-use-condition-selectivity=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`

- Default Value: `4`
- Range: `1` to `5`

`pid_file`

- Description: Full path of the process ID file. If –log-basename is also set, `pid_file` should be placed after in the config files. Later settings override earlier settings, so `log-basename` will override any earlier log file name settings.
- Command line: `--pid-file=file_name`
- Scope: Global
- Dynamic: No
- Data Type: `file name`

`plugin_dir`

- Description: Path to the plugin directory. For security reasons, either make sure this directory can only be read by the server, or set secure_file_priv.
- Command line: `--plugin-dir=path`
- Scope: Global
- Dynamic: No
- Data Type: `directory name`
- Default Value: `BASEDIR/lib/plugin`

`plugin_maturity`

- Description: The lowest acceptable plugin maturity. MariaDB will not load plugins less mature than the specified level.
- Command line: `--plugin-maturity=level`
- Scope: Global
- Dynamic: No
- Type: enum
- Default Value: One less than the server maturity
- Valid Values: `unknown`, `experimental`, `alpha`, `beta`, `gamma`, `stable`

`port`

- Description: Port to listen for TCP/IP connections. If set to `0`, will default to, in order of preference, my.cnf, the `MYSQL_TCP_PORT` environment variable, /etc/services, built-in default (3306).
- Command line: `--port=#`, `-P`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `3306`
- Range: `0` to `65535`

`preload_buffer_size`

- Description: Size in bytes of the buffer allocated when indexes are preloaded.
- Command line: `--preload-buffer-size=#`
- Scope: Global, Session

- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `32768`
- Range: `1024` to `1073741824`

`profiling`

- Description: If set to `1` (`0` is default), statement profiling will be enabled. See SHOW PROFILES() and SHOW PROFILE().
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`profiling_history_size`

- Description: Number of statements about which profiling information is maintained. If set to `0`, no profiles are stored. See SHOW PROFILES.
- Command line: `--profiling-history-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `15`
- Range: `0` to `100`

`progress_report_time`

- Description: Time in seconds between sending progress reports to the client for time-consuming statements. If set to `0`, progress reporting will be disabled.
- Command line: `--progress-report-time=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `5`
- Range: `0` to `4294967295`

`protocol_version`

- Description: The version of the client/server protocol used by the MariaDB server.
- Command line: None
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `10`
- Range: `0` to `4294967295`

`proxy_protocol_networks`

- Description: Enable proxy protocol for these source networks. The syntax is a comma separated list of IPv4 and IPv6 networks. If the network doesn't contain a mask, it is considered to be a single host. "*" represents all

networks and must be the only directive on the line. String “localhost” represents non-TCP local connections (Unix domain socket, Windows named pipe or shared memory). See Proxy Protocol Support.

- Command line: `--proxy-protocol-networks=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: (empty)

`proxy_user`

- Description: Set to the proxy user account name if the current client is a proxy, else `NULL`.
- Scope: Session
- Dynamic: No
- Data Type: `string`

`pseudo_slave_mode`

- Description: For internal use by the server.
- Scope: Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `OFF`

`pseudo_thread_id`

- Description: For internal use only.
- Scope: Session
- Dynamic: Yes
- Data Type: `numeric`

`query_alloc_block_size`

- Description: Size in bytes of the extra blocks allocated during query parsing and execution (after query_prealloc_size is used up).
- Command line: `--query-alloc-block-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `BIGINT UNSIGNED`
- Default Value:
  - >= MariaDB 11.4.5, MariaDB 10.11.11, MariaDB 10.6.22: `32768`
  - <= MariaDB 11.4.4, MariaDB 10.11.10, MariaDB 10.6.21: `16384`
- Range: `1024` to `4294967295`
- Block size: `1024`

`query_cache_limit`

- Description: Size in bytes for which results larger than this are not stored in the query cache.
- Command line: `--query-cache-limit=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `1048576` (1MB)
- Range: `0` to `4294967295`

`query_cache_min_res_unit`

- Description: Minimum size in bytes of the blocks allocated for query cache results.
- Command line: `--query-cache-min-res-unit=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4096` (4KB)
- Range - 32 bit: `1024` to `4294967295`
- Range - 64 bit: `1024` to `18446744073709547520`

`query_cache_size`

- Description: Size in bytes available to the query cache. About 40KB is needed for query cache structures, so setting a size lower than this will result in a warning. `0`, the default before MariaDB 10.1.7, effectively disables the query cache.

Warning: Starting from MariaDB 10.1.7, query_cache_type is automatically set to ON if the server is started with the query_cache_size set to a non-zero (and non-default) value. This will happen even if query_cache_type is explicitly set to OFF in the configuration.

- Command line: `--query-cache-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1M` (although frequently given a default value in some setups)
- Valid Values: `0` upwards in units of 1024.

`query_cache_strip_comments`

- Description: If set to `1` (`0` is default), the server will strip any comments from the query before searching to see if it exists in the query cache. Multiple space, line feeds, tab and other white space characters will also be removed.
- Command line: `query-cache-strip-comments`
- Scope: Session, Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`query_cache_type`

- Description: If set to `0`, the query cache is disabled (although a buffer of query_cache_size bytes is still allocated). If set to `1` all SELECT queries will be cached unless SQL_NO_CACHE is specified. If set to `2` (or `DEMAND`), only queries with the SQL CACHE clause will be cached. Note that if the server is started with the query cache disabled, it cannot be enabled at runtime.

Warning: Starting from MariaDB 10.1.7, query_cache_type is automatically set to ON if the server is started with the query_cache_size set to a non-zero (and non-default) value. This will happen even if query_cache_type is explicitly

set to OFF in the configuration.

- Command line: `--query-cache-type=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `OFF`
- Valid Values: `0` or `OFF`, `1` or `ON`, `2` or `DEMAND`

`query_cache_wlock_invalidate`

- Description: If set to `0`, the default, results present in the query cache will be returned even if there's a write lock on the table. If set to `1`, the client will first have to wait for the lock to be released.
- Command line: `--query-cache-wlock-invalidate`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`query_prealloc_size`

- Description: Size in bytes of the persistent buffer for query parsing and execution, allocated on connect and freed on disconnect. Increasing may be useful if complex queries are being run, as this will reduce the need for more memory allocations during query operation. See also query_alloc_block_size.
- Command line: `--query-prealloc-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - >= MariaDB 11.4.5, MariaDB 10.11.11, MariaDB 10.6.22: `32768`
  - <= MariaDB 11.4.4, MariaDB 10.11.10, MariaDB 10.6.21: `24576`
- Range: `1024` to `4294967295`

`rand_seed1`

- Description: `rand_seed1` and `rand_seed2` facilitate replication of the RAND() function. The master passes the value of these to the slaves so that the random number generator is seeded in the same way, and generates the same value, on the slave as on the master.
- Command line: None
- Scope: Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: Varies
- Range: `0` to `18446744073709551615`

`rand_seed2`

- Description: See rand_seed1.

`range_alloc_block_size`

- Description: Size in bytes of blocks allocated during range optimization. The unit size in 1024.
- Command line: `--range-alloc-block-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4096`
- Range - 32 bit: `4096` to `4294967295`
- Range - 64 bit: `4096` to `18446744073709547520`

`read_buffer_size`

- Description: Each thread performing a sequential scan (for MyISAM, Aria and MERGE tables) allocates a buffer of this size in bytes for each table scanned. Increase if you perform many sequential scans. If not in a multiple of 4KB, will be rounded down to the nearest multiple. Also used in ORDER BY's for caching indexes in a temporary file (not temporary table), for caching results of nested queries, for bulk inserts into partitions, and to determine the memory block size of MEMORY tables.
- Command line: `--read-buffer-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `131072`
- Range: `8192` to `2147479552`

`read_only`

> **CURRENT**
>
> - Description: Do not allow changes to non-temporary tables. Options are: `OFF` — changes allowed; `ON` — Disallow changes for users without the `READ ONLY ADMIN` privilege; `NO_LOCK` — Additionally disallows `LOCK TABLES` and `SELECT ... IN SHARE MODE`; `NO_LOCK_NO_ADMIN` — Disallows also for users with `READ_ONLY ADMIN` privilege. Replication (slave) threads are not affected by this option.
> - Command line: `--read-only`
> - Scope: Global
> - Dynamic: Yes
> - Data Type: `enum`
> - Default Value: `OFF`
> - Valid values: `OFF`, `ON`, `NO_LOCK`, `NO_LOCK_NO_ADMIN`
>
> **< 12.0**
>
> - Description: When set to `1` (`0` is default), no updates are permitted except from users with the SUPER privilege or, from MariaDB 10.5.2, the READ ONLY ADMIN privilege, or replica servers updating from a primary. The `read_only` variable is useful for replica servers to ensure no updates are accidentally made outside of what are performed on the primary. Inserting rows to log tables, updates to temporary tables and OPTIMIZE TABLE or ANALYZE TABLE statements are excluded from this limitation. If `read_only` is set to `1`, then the SET PASSWORD statement is limited only to users with the SUPER privilege (<= MariaDB 10.5.1) or READ ONLY ADMIN privilege (>= MariaDB 10.5.2). Attempting to set this variable to `1` will fail if the current session has table locks or transactions pending, while if other sessions hold table locks, the statement will wait until these locks are released before completing. While the attempt to set `read_only` is waiting, other requests

> for table locks or transactions will also wait until `read_only` has been set. See Read-Only Replicas for more. From MariaDB 10.5.2, the READ_ONLY ADMIN privilege will allow users granted that privilege to perform writes, even if the `read_only` variable is set. In earlier versions, and until MariaDB 10.11.0, users with the SUPER can perform writes while this variable is set.
> - Command line: `--read-only`
> - Scope: Global
> - Dynamic: Yes
> - Data Type: `boolean`
> - Default Value: `OFF`

`read_rnd_buffer_size`

- Description: Size in bytes of the buffer used when reading rows from a MyISAM table in sorted order after a key sort. Larger values improve ORDER BY performance, although rather increase the size by SESSION where the need arises to avoid excessive memory use.
- Command line: `--read-rnd-buffer-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `262144`
- Range: `8200` to `2147483647`

`redirect_url`

- Description: URL of another server to redirect clients to. Format should be `{mysql,mariadb}://host [:port]`. Empty string means no redirection. For example, `set global redirect_url="mysql://mariadb.org:12345"`. See Connection Redirection Mechanism in the MariaDB Client/Server Protocol.
- Command line: `--redirect_url=val`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: Empty
- Introduced: MariaDB 11.3.0

`require_secure_transport`

- Description: When this option is enabled, connections attempted using insecure transport will be rejected. Secure transports are SSL/TLS, Unix sockets or named pipes. Note that per-account requirements take precedence.
- Command line: `--require-secure-transport[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.5.2

`rowid_merge_buff_size`

- Description: The maximum size in bytes of the memory available to the Rowid-merge strategy. See Non-semi-join subquery optimizations for more information.
- Command line: `--rowid-merge-buff-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `8388608`
- Range: `0` to `2147483647`

`rpl_recovery_rank`

- Description: Unused.
- Removed: MariaDB 10.1.2

`safe_show_database`

- Description: This variable was removed in MariaDB 5.5 and has been replaced by the more flexible SHOW DATABASES privilege.
- Command line: `--safe-show-database` (until MySQL 4.1.1)
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Removed: MariaDB 5.5

`secure_auth`

- Description: Connections will be blocked if they use the mysql_old_password authentication plugin. The server will also fail to start if the privilege tables are in the old, pre-MySQL 4.1 format. `secure_auth=0` was deprecated in MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3.
- Command line: `--secure-auth`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Removed: MariaDB 13.0. The `secure_auth` system variable and the `--secure-auth` server option were removed. The protection they provided (blocking connections that use the mysql_old_password authentication plugin, and refusing to start when the privilege tables are in the old, pre-MySQL 4.1 format) is now always enforced and can no longer be disabled.

`secure_file_priv`

- Description: LOAD DATA, SELECT … INTO and LOAD FILE() will only work with files in the specified path. If not set, the default, or set to empty string, the statements will work with any files that can be accessed.
- Command line: `--secure-file-priv=path`
- Scope: Global
- Dynamic: No
- Data Type: `path name`
- Default Value: None

`secure_timestamp`

- Description: Restricts direct setting of a session timestamp. Possible levels are:
  - YES - timestamp cannot deviate from the system clock. Intended to prevent tampering with system versioning history. Should not be used on replicas, as when a value based on the timestamp is inserted in statement mode, discrepancies can occur.
  - REPLICATION - replication thread can adjust timestamp to match the primary's
  - SUPER - a user with this privilege and a replication thread can adjust timestamp
  - NO - historical behavior, anyone can modify session timestamp
- Command line: `--secure-timestamp=value`
- Scope: Global
- Dynamic: No
- Data Type: `enum`
- Default Value: `NO`

`server_uid`

- Description: Automatically calculated server unique id hash. Added to the error log to allow one to verify if error reports are from the same server. UID is a base64-encoded SHA1 hash of the MAC address of one of the interfaces, and the tcp port that the server is listening on.
- Command line: None
- Scope: Global
- Dynamic: No
- Data Type: `varchar`
- Default Value: None
- Introduced: MariaDB 10.5.26, MariaDB 10.6.19, MariaDB 10.11.9, MariaDB 11.1.6, MariaDB 11.2.5, MariaDB 11.4.3, MariaDB 11.5.2, MariaDB 11.6.1

`session_track_schema`

- Description: Whether to track changes to the default schema within the current session.
- Command line: `--session-track-schema={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`session_track_state_change`

- Description: Whether to track changes to the session state.
- Command line: `--session-track-state-change={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`session_track_system_variables`

- Description: Comma-separated list of session system variables for which to track changes. For compatibility with MySQL defaults, this variable should be set to "autocommit, character_set_client, character_set_connection,

character_set_results, time_zone". The `*` character tracks all session variables.

- Command line: `--session-track-system-variables=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value:
  - > = MariaDB 11.3: `autocommit,character_set_client,character_set_connection,character_set_results,redirect_url,time_zone`
  - <= MariaDB 11.2: `autocommit, character_set_client, character_set_connection, character_set_results, time_zone`

`session_track_transaction_info`

- Description: Track changes to the transaction attributes. OFF to disable; STATE to track just transaction state (Is there an active transaction? Does it have any data? etc.); CHARACTERISTICS to track transaction state and report all statements needed to start a transaction with the same characteristics (isolation level, read only/read write,snapshot - but not any work done / data modified within the transaction).
- Command line: `--session-track-transaction-info=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `OFF`
- Valid Values: `OFF`, `STATE`, `CHARACTERISTICS`

`shared_memory`

- Description: Windows only, determines whether the server permits shared memory connections. See also shared_memory_base_name.
- Scope: Global
- Dynamic: No

`shared_memory_base_name`

- Description: Windows only, specifies the name of the shared memory to use for shared memory connection. Mainly used when running more than one instance on the same physical machine. By default the name is `MYSQL` and is case sensitive. See also shared_memory.
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `MYSQL`

`shutdown_wait_for_slaves`

- Description: When `ON, SHUTDOWN` command runs with implicit `WAIT FOR ALL SLAVES` option. That is, when running `SHUTDOWN`, before killing the binary log dump threads, the server will first kill all client threads and send all binary log events to all connected replicas.
- Scope: Global
- Dynamic: No

- Data Type: `Boolean`
- Default Value: `OFF`

`skip_external_locking`

- Description: If this system variable is set, then some kinds of external table locks will be disabled for some storage engines.
  - If this system variable is set, then the MyISAM storage engine will not use file-based locks. Otherwise, it will use the fcntl() function with the `F_SETLK` option to get file-based locks on Unix, and it will use the LockFileEx() function to get file-based locks on Windows.
  - If this system variable is set, then the Aria storage engine will not lock a table when it decrements the table's in-file counter that keeps track of how many connections currently have the table open. See MDEV-19393 for more information.
  - Note that command line option name is the opposite of the variable name, and the value is the opposite too. `--external-locking=1` means `@@skip_external_locking=0`, and vice versa.
- Command line: `--external-locking`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `1` (for the variable, that is `0` for the command line option)

`skip_grant_tables`

- Description: Start without grant tables. This gives all users FULL ACCESS to all tables. Before MariaDB 10.10, available as an option only. Use mariadb-admin flush-privileges, mariadb-admin reload or FLUSH PRIVILEGES to resume using the grant tables.
- Command line: `--skip-grant-tables`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.10

`skip_name_resolve`

- Description: If set to `ON` (`OFF` is the default), only IP addresses are used for connections. Host names are not resolved. All host values in the GRANT tables must be IP addresses (or localhost). Some container configs explicitly set `skip_name_resolve` to `ON`, rather than leave it as the default, `OFF`.
- Command line: `--skip-name-resolve`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`skip_networking`

- Description: If set to 1, (0 is the default), the server does not listen for TCP/IP connections. All interaction with the server will be through socket files (Unix) or named pipes or shared memory (Windows). It's recommended to use this option if only local clients are permitted to connect to the server.
- Command line: `--skip-networking`

- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `0`

`skip_show_database`

- Description: If set to 1, (0 is the default), only users with the SHOW DATABASES privilege can use the SHOW DATABASES statement to see all database names.
- Command line: `--skip-show-database`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `0`

`slow_launch_time`

- Description: Time in seconds. If a thread takes longer than this to launch, the `slow_launch_threads` server status variable is incremented.
- Command line: `--slow-launch-time=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2`

`slow_query_log`

- Description: If set to 0, the default unless the –slow-query-log option is used, the slow query log is disabled, while if set to 1 (both global and session variables), the slow query log is enabled. From MariaDB 10.11.0, an alias for log_slow_query.
- Command line: `--slow-query-log`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Data Type: `boolean`
- Default Value: `0`
- See also: See log_output to see how log files are written. If that variable is set to `NONE`, no logs will be written even if slow_query_log is set to `1`.

`slow_query_log_file`

- Description: Name of the slow query log file. From MariaDB 10.11, an alias for log_slow_query_file. If –log-basename is also set, `slow_query_log_file` should be placed after in the config files. Later settings override earlier settings, so `log-basename` will override any earlier log file name settings.
- Command line: `--slow-query-log-file=file_name`
- Scope: Global
- Dynamic: Yes
- Data Type: `file name`
- Default Value: `host_name-slow.log`

`socket`

- Description: On Unix-like systems, this is the name of the socket file used for local client connections, by default `/tmp/mysql.sock`, often changed by the distribution, for example `/var/lib/mysql/mysql.sock`. On Windows, this is the name of the named pipe used for local client connections, by default `MySQL`. On Windows, this is not case-sensitive.
- Command line: `--socket=name`
- Scope: Global
- Dynamic: No
- Data Type: `file name`
- Default Value: `/tmp/mysql.sock` (Unix), `MySQL` (Windows)

`sort_buffer_size`

- Description: Each session performing a sort allocates a buffer with this amount of memory. Not specific to any storage engine. If the status variable sort_merge_passes is too high, you may need to look at improving your query indexes or increasing this. Consider reducing where there are many small sorts, such as OLTP, and increasing where needed by session. 16k is a suggested minimum.
- Command line: `--sort-buffer-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `number`
- Default Value: `2M (2097152)` (some distributions increase the default)

`sql_auto_is_null`

- Description: If set to 1, the query `SELECT * FROM table_name WHERE auto_increment_column IS NULL` will return an auto-increment that has just been successfully inserted, the same as the LAST_INSERT_ID() function. Some ODBC programs make use of this IS NULL comparison.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`

`sql_big_selects`

- Description: If set to 0, MariaDB will not perform large SELECTs. See max_join_size for details. If max_join_size is set to anything but DEFAULT, sql_big_selects is automatically set to 0. If sql_big_selects is again set, max_join_size will be ignored.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`

`sql_big_tables`

- Description: Old variable, which if set to 1, allows large result sets by saving all temporary sets to disk, avoiding 'table full' errors. No longer needed, as the server now handles this automatically.
  - This is a synonym for big_tables.
- Command line: `--sql-big-tables`
- Scope: Global, Session

- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`
- Removed: MariaDB 10.0

`sql_buffer_result`

- Description: If set to 1 (0 is default), results from SELECT statements are always placed into temporary tables. This can help the server when it takes a long time to send the results to the client by allowing the table locks to be freed early.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`

`sql_if_exists`

- Description: If set to 1, adds an implicit IF EXISTS to ALTER, RENAME and DROP of TABLES, VIEWS, FUNCTIONS and PACKAGES. This variable is mainly used in replication to tag DDLs that can be ignored on the slave if the target table doesn't exist.
- Command line: `--sql-if-exists[={0|1}]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.5.2

`sql_log_off`

- Description: If set to 1 (0 is the default), no logging to the general query log is done for the client. Only clients with the SUPER privilege can update this variable.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`

`sql_log_update`

- Description: Removed. Use sql_log_bin instead.
- Removed: MariaDB/MySQL 5.5

`sql_low_priority_updates`

- Description: If set to 1 (0 is the default), for storage engines that use only table-level locking (Aria, MyISAM, MEMORY and MERGE), all INSERTs, UPDATEs, DELETEs and LOCK TABLE WRITEs will wait until there are no more SELECTs or LOCK TABLE READs pending on the relevant tables. Set this to 1 if reads are prioritized over writes.
  - This is a synonym for low_priority_updates.
- Command line: `--sql-low-priority-updates`
- Scope: Global, Session
- Dynamic: Yes

- Data Type: `boolean`
- Default Value: `0`
- Removed: MariaDB 10.0

`sql_max_join_size`

- Description: Synonym for max_join_size, the preferred name.
- Deprecated: MariaDB 5.5
- Removed: MariaDB 10.0

`sql_mode`

- Description: Sets the SQL Mode. Multiple modes can be set, separated by a comma.
- Command line: `--sql-mode=value[,value[,value...]]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value:
  - `STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION`
- Valid Values: See SQL Mode for the full list.

`sql_notes`

- Description: If set to 1, the default, warning_count is incremented each time a Note warning is encountered. If set to 0, Note warnings are not recorded. mariadb-dump has outputs to set this variable to 0 so that no unnecessary increments occur when data is reloaded. See also note_verbosity, which defines which notes should be given. The recommended way, as of MariaDB 10.6.16, to disable notes is to set `note_verbosity` to ““.
- Command line: None
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`

`sql_quote_show_create`

- Description: If set to 1, the default, the server will quote identifiers for SHOW CREATE DATABASE, SHOW CREATE TABLE and SHOW CREATE VIEW statements. Quoting is disabled if set to 0. Enable to ensure replication works when identifiers require quoting.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`

`sql_safe_updates`

- Description: If set to 1, UPDATEs and DELETEs must be executed by using an index (simply mentioning an indexed column in a WHERE clause is not enough, optimizer must actually use it) or they must mention an indexed column and specify a LIMIT clause. Otherwise a statement will be aborted. Prevents the common mistake of accidentally deleting or updating every row in a table. Until MariaDB 10.3.11, could not be set as a command-line option or in my.cnf.

- Command line: `--sql-safe-updates[={0|1}]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`sql_select_limit`

- Description: Maximum number of rows that can be returned from a SELECT query. Default is the maximum number of rows permitted per table by the server, usually 232-1 or 264-1. Can be restored to the default value after being changed by assigning it a value of DEFAULT. If a SELECT has a LIMIT clause, the LIMIT takes precedence over the value of the variable.
- Command line: None
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `18446744073709551615`

`sql_warnings`

- Description: If set to 1, single-row INSERTs will produce a string containing warning information if a warning occurs.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF (0)`

`storage_engine`

- Description: See default_storage_engine.
- Deprecated: MariaDB 5.5
- Removed: MariaDB 12.0

`standard_compliant_cte`

- Description: Allow only standard-compliant common table expressions. Prior to MariaDB 10.2.4, this variable was named `standards_compliant_cte`.
- Command line: `--standard-compliant-cte={0|1}`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`stored_program_cache`

- Description: Limit to the number of stored routines held in the stored procedures and stored functions caches. Each time a stored routine is executed, this limit is first checked, and if the number held in the cache exceeds this, that cache is flushed and memory freed.
- Command line: `--stored-program-cache=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `256`
- Range: `256` to `524288`

`strict_password_validation`

- Description: When password validation plugins are enabled, reject passwords that cannot be validated (passwords specified as a hash). This excludes direct updates to the privilege tables.
- Command line: `--strict-password-validation`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`sync_frm`

- Description: If set to 1, the default, each time a non-temporary table is created, its .frm definition file is synced to disk. Fractionally slower, but safer in case of a crash.
- Command line: `--sync-frm`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `TRUE`

`system_time_zone`

- Description: The system time zone is determined when the server starts. The system time zone is usually read from the operating system's environment but can be overridden by setting the 'TZ' environment variable before starting the server. See Time Zones: System Time Zone for the various ways to change the system time zone. This variable is not the same as the time_zone system variable, which is the variable that actually controls a session's active time zone. The system time zone is used for a session when `time_zone` is set to the special value `SYSTEM`.
- Scope: Global
- Dynamic: No
- Data Type: `string`

`table_definition_cache`

- Description: Number of table definitions that can be cached. Table definitions are taken from the .frm files, and if there are a large number of tables increasing the cache size can speed up table opening. Unlike the table_open_cache, as the table_definition_cache doesn't use file descriptors and is much smaller.
- Command line: `--table-definition-cache=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `400`
- Range: `400` to `2097152`

`table_lock_wait_timeout`

- Description: Unused, and removed.

- Command line: `--table-lock-wait-timeout=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `50`
- Range: `1` to `1073741824`
- Removed: MariaDB 5.5

`table_open_cache`

- Description: Maximum number of open tables cached in one table cache instance. See Optimizing table_open_cache for suggestions on optimizing. Increasing table_open_cache increases the number of file descriptors required.
- Command line: `--table-open-cache=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `2000`
- Range:
  - `1` to `1048576` (1024K)

`table_open_cache_instances`

- Description: This system variable specifies the maximum number of table cache instances. MariaDB Server initially creates just a single instance. However, whenever it detects contention on the existing instances, it will automatically create a new instance. When the number of instances has been increased due to contention, it does not decrease again. The default value of this system variable is `8`, which is expected to handle up to 100 CPU cores. If your system is larger than this, then you may benefit from increasing the value of this system variable.
  - Depending on the ratio of actual available file handles, and table_open_cache size, the max. instance count may be auto adjusted to a lower value on server startup.
  - The implementation and behavior of this feature is different than the same feature in MySQL 5.6.
  - See Optimizing table_open_cache: Automatic Creation of New Table Open Cache Instances for more information.
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `8` (>= MariaDB 10.2.2)
- Range: `1` to `64`

`table_type`

- Description: Removed and replaced by storage_engine. Use default_storage_engine instead.

`tcp_keepalive_interval`

- Description: The interval, in seconds, between when successive keep-alive packets are sent if no acknowledgement is received. If set to 0, the system dependent default is used.
- Command line: `--tcp-keepalive-interval=#`
- Scope: Global

- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `2147483`

`tcp_keepalive_probes`

- Description: The number of unacknowledged probes to send before considering the connection dead and notifying the application layer. If set to 0, a system dependent default is used.
- Command line: `--tcp-keepalive-probes=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `2147483`

`tcp_keepalive_time`

- Description: Timeout, in seconds, with no activity until the first TCP keep-alive packet is sent. If set to 0, a system dependent default is used.
- Command line: `--tcp-keepalive-time=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `2147483`

`tcp_nodelay`

- Description: Set the TCP_NODELAY option (disable Nagle's algorithm) on socket.
- Command line: `--tcp-nodelay={0|1}`
- Scope: Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`

`thread_cache_size`

- Description: Number of threads server caches for reuse. If this limit hasn't been reached, when a client disconnects, its threads are put into the cache and reused where possible. In MariaDB 10, MariaDB 5.5, and newer, the threads are freed after 5 minutes of idle time. Normally this setting has little effect, as the other aspects of the thread implementation are more important, but increasing it can help servers with high volumes of connections per second so that most can use a cached, rather than a new, thread. The cache miss rate can be calculated as the server status variables threads_created/connections. If the thread pool is active, `thread_cache_size` is ignored. If `thread_cache_size` is set to greater than the value of max_connections, `thread_cache_size` will be set to the max_connections value.
- Command line: `--thread-cache-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`

- Default Value: `256` (adjusted if thread pool is active)
- Range: `0` to `16384`

`thread_concurrency`

- Description: Allows applications to give the system a hint about the desired number of threads. Specific to Solaris only, invokes thr_setconcurrency(). Deprecated and has no effect from MariaDB 5.5.
- Command line: `--thread-concurrency=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `10`
- Range: `1` to `512`
- Deprecated: MariaDB 5.5
- Removed: MariaDB 10.5.1

`thread_stack`

- Description: Stack size for each thread. If set too small, limits recursion depth of stored procedures and complexity of SQL statements the server can handle in memory. Also affects limits in the crash-me test.
- Command line: `--thread-stack=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value:
  - `299008`
- Range: `131072` to `18446744073709551615`

`time_format`

- Description: Unused.
- Removed: MariaDB 11.3.0

`time_zone`

- Description: The global value determines the default time zone for sessions that connect. The session value determines the session's active time zone. When it is set to `SYSTEM`, the session's time zone is determined by the system_time_zone system variable.
- Command line: `--default-time-zone=string`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `SYSTEM`

`timed_mutexes`

- Description: Determines whether InnoDB mutexes are timed. `OFF`, the default, disables mutex timing, while `ON` enables it. See also SHOW ENGINE for more on mutex statistics. Deprecated and has no effect.
- Command line: `--timed-mutexes`
- Scope: Global

- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Deprecated: MariaDB 5.5.39
- Removed: MariaDB 10.5.1

`timestamp`

- Description: Sets the time for the client. This will affect the result returned by the NOW() function, not the SYSDATE() function, unless the server is started with the –sysdate-is-now option, in which case SYSDATE becomes an alias of NOW, and will also be affected. Also used to get the original timestamp when restoring rows from the binary log.
- Scope: Session
- Dynamic: Yes
- Valid Values: `timestamp_value` (Unix epoch timestamp, not MariaDB timestamp), `DEFAULT`

`tmp_disk_table_size`

- Description: Max size for data for an internal temporary on-disk MyISAM or Aria table. These tables are created as part of complex queries when the result doesn't fit into the memory engine. You can set this variable if you want to limit the size of temporary tables created in your temporary directory tmpdir.
- Command line: `--tmp-disk-table-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `18446744073709551615` (max unsigned integer, no limit)
- Range: `1024` to `18446744073709551615`

`tmp_memory_table_size`

- Description: An alias for tmp_table_size.
- Command line: `--tmp-memory-table-size=#`

`tmp_table_size`

- Description: The largest size for temporary tables in memory (not MEMORY tables) although if max_heap_table_size is smaller the lower limit will apply. You can see if it's necessary to increase by comparing the status variables `Created_tmp_disk_tables` and `Created_tmp_tables` to see how many temporary tables out of the total created needed to be converted to disk. Often complex GROUP BY queries are responsible for exceeding the limit. Defaults may be different on some systems, see for example Differences in MariaDB in Debian. tmp_memory_table_size is an alias.
- Command line: `--tmp-table-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `16777216` (16MB)
- Range:
  - `1024` to `4294967295` (< MariaDB 10.5)
  - `0` to `4294967295` (>= MariaDB 10.5.0)

`tmpdir`

- Description: Directory for storing temporary tables and files. Can specify a list (separated by semicolons in Windows, and colons in Unix) that will then be used in round-robin fashion. This can be used for load balancing across several disks. Note that if the server is a replication replica, and slave_load_tmpdir, which overrides `tmpdir` for replicas, is not set, you should not set `tmpdir` to a directory that is cleared when the machine restarts, or else replication may fail.
- Command line: `--tmpdir=path` or `-t path`
- Scope: Global
- Dynamic: No
- Type: directory name/s
- Default:
  - `$TMPDIR` (environment variable) if set
  - otherwise `$TEMP` if set and on Windows
  - otherwise `$TMP` if set and on Windows
  - otherwise, `P_tmpdir` (`"/tmp"`) or `C:\TEMP` (unless overridden during build time)

`transaction_alloc_block_size`

- Description: Size in bytes to increase the memory pool available to each transaction when the available pool is not large enough. See transaction_prealloc_size.
- Command line: `--transaction-alloc-block-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Type: numeric
- Default Value: `8192`
- Range: `1024` to `134217728` (128M)
- Block Size: `1024`

`transaction_isolation`

- Description: The transaction isolation level. See also SET TRANSACTION ISOLATION LEVEL. Introduced in MariaDB 11.1.1 to replace the tx_isolation system variable and align the option and the system variable name.
- Command line: `--transaction-isolation=name`
- Scope: Global, Session
- Dynamic: Yes
- Type: enumeration
- Default Value: `REPEATABLE-READ`
- Valid Values: `READ-UNCOMMITTED`, `READ-COMMITTED`, `REPEATABLE-READ`, `SERIALIZABLE`
- Introduced: MariaDB 11.1.1

`transaction_prealloc_size`

- Description: Initial size of a memory pool available to each transaction for various memory allocations. If the memory pool is not large enough for an allocation, it is increased by transaction_alloc_block_size bytes, and truncated back to transaction_prealloc_size bytes when the transaction is completed. If set large enough to contain all statements in a transaction, extra malloc() calls are avoided.
- Command line: `--transaction-prealloc-size=#`
- Scope: Global, Session
- Dynamic: Yes

- Type: numeric
- Default Value: `4096`
- Range: `1024` to `134217728` (128M)
- Block Size: `1024`

`transaction_read_only`

- Description: Default transaction access mode. If set to `OFF`, the default, access is read/write. If set to `ON`, access is read-only. The `SET TRANSACTION` statement can also change the value of this variable. See SET TRANSACTION and START TRANSACTION.
- Command line: None
- Scope: Global, Session
- Dynamic: Yes
- Type: boolean
- Default Value: `OFF`
- Introduced: MariaDB 11.1

`tx_isolation`

- Description: The transaction isolation level. Setting this session variable via `set @@tx_isolation=` will take effect for only the subsequent transaction in the current session, much like SET TRANSACTION ISOLATION LEVEL. To set for a session, use `SET SESSION tx_isolation` or `SET @@session.tx_isolation`. See MDEV-31751. See also SET TRANSACTION ISOLATION LEVEL. In MariaDB 11.1, this system variable is deprecated and replaced by transaction_isolation.
- Command line: `--transaction-isolation=name`
- Scope: Global, Session
- Dynamic: Yes
- Type: enumeration
- Default Value: `REPEATABLE-READ`
- Valid Values: `READ-UNCOMMITTED`, `READ-COMMITTED`, `REPEATABLE-READ`, `SERIALIZABLE`
- Deprecated: MariaDB 11.1

`tx_read_only`

- Description: Default transaction access mode. If set to `OFF`, the default, access is read/write. If set to `ON`, access is read-only. The `SET TRANSACTION` statement can also change the value of this variable. See SET TRANSACTION and START TRANSACTION. In MariaDB 11.1, this system variable is deprecated and replaced by transaction_read_only.
- Command line: `--transaction-read-only=#`
- Scope: Global, Session
- Dynamic: Yes
- Type: boolean
- Default Value: `OFF`
- Deprecated: MariaDB 11.1

`unique_checks`

- Description: If set to 0, storage engines can (but are not required to) assume that duplicate keys are not present in input data. If set to 0, inserting duplicates into a `UNIQUE` index can succeed, causing the table to become corrupted. Set to 0 to speed up imports of large tables to InnoDB.

- Scope: Global, Session
- Dynamic: Yes
- Type: boolean
- Default Value: `1`

`updatable_views_with_limit`

- Description: Determines whether view updates can be made with an UPDATE or DELETE statement with a LIMIT clause if the view does not contain all primary or not null unique key columns from the underlying table. `0` prohibits this, while `1` permits it while issuing a warning (the default).
- Command line: `--updatable-views-with-limit=#`
- Scope: Global, Session
- Dynamic: Yes
- Type: boolean
- Default Value: `1`

`use_stat_tables`

- Description: Controls the use of engine-independent table statistics.
  - `never`: The optimizer will not use data from statistics tables.
  - `complementary`: The optimizer uses data from statistics tables if the same kind of data is not provided by the storage engine.
  - `preferably`: Prefer the data from statistics tables, if it's not available there, use the data from the storage engine.
  - `complementary_for_queries`: Same as `complementary`, but for queries only (to avoid needlessly collecting for ANALYZE TABLE).
  - `preferably_for_queries`: Same as `preferably`, but for queries only (to avoid needlessly collecting for ANALYZE TABLE).
- Command line: `--use-stat-tables=mode`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `preferably_for_queries`

`validate_config`

- Description: Validates the server configuration (from configuration files and command line). Exits with exit code `0` on success, or non-zero on failure, without actually starting the server. Example:

  ```
  mariadbd --defaults-file=/etc/my.cnf --validate-config
  ```

  > `-help` or `-version` take precedence. If used with this option, the configuration check is **not** performed.

- Exit codes:
  - `0`: valid configuration
  - `2`: unknown option
  - `7`: unknown variable
- Command line: `--validate-config`

- Scope: Global
- Dynamic: No
- Data Type: numeric
- Introduced: MariaDB 13.0

`version`

- Description: Server version number. It may also include a suffix with configuration or build information. `-debug` indicates debugging support was enabled on the server, and `-log` indicates at least one of the binary log, general log or slow query log are enabled, for example `10.0.1-MariaDB-mariadb1precise-log`. Can be set at startup in order to fake the server version.
- Command line: `-V`, `--version[=name]`
- Scope: Global
- Dynamic: No
- Data Type: string

`version_comment`

- Description: Value of the COMPILATION_COMMENT option specified by CMake when building MariaDB, for example `mariadb.org binary distribution`.
- Scope: Global
- Dynamic: No
- Type: string

`version_compile_machine`

- Description: The machine type or architecture MariaDB was built on, for example `i686`.
- Scope: Global
- Dynamic: No
- Type: string

`version_compile_os`

- Description: Operating system that MariaDB was built on, for example `debian-linux-gnu`.
- Scope: Global
- Dynamic: No
- Type: string

`version_malloc_library`

- Description: Version of the used malloc library.
- Command line: None
- Scope: Global
- Dynamic: No
- Type: string

`version_source_revision`

- Description: Source control revision id for MariaDB source code, enabling one to see exactly which version of the source was used for a build.
- Command line: None

- Scope: Global
- Dynamic: No
- Type: string

`wait_timeout`

- Description: Time in seconds that the server waits for a connection to become active before closing it. The session value is initialized when a thread starts up from either the global value, if the connection is non-interactive, or from the interactive_timeout value, if the connection is interactive.
- Command line: `--wait-timeout=#`
- Scope: Global, Session
- Dynamic: Yes
- Type: numeric
- Default Value: `28800`
- Range: (Windows): `1` to `2147483`
- Range: (Other): `1` to `31536000`

`warning_count`

- Description: Read-only variable indicating the number of warnings, errors and notes resulting from the most recent statement that generated messages. See SHOW WARNINGS for more. Note warnings will only be recorded if sql_notes is true (the default).
- Scope: Session
- Dynamic: No
- Type: numeric

This page is licensed: CC BY-SA / Gnu FDL

NEW_MODE

The `NEW_MODE` system variable and command line switch were introduced in MariaDB 11.4 to provide a way to enable new behavior in otherwise stable versions. Specifying a flag in the `NEW_MODE` variable enables the corresponding new behavior; otherwise, the old (stable) behavior is used. This can be useful to preserve execution plans in stable versions that may change when the new behavior is active.

A sample usage scenario is:

- a fix (new behavior) is pushed into stable releases (for example, MariaDB 11.4 and MariaDB 11.8). It becomes available in `NEW_MODE` but isn't enabled by default.
- in MariaDB 12.1, the fix (new behavior) is enabled without the switch. There's no way to turn it off.
  - `NEW_MODE` does not list the fix as something that can be turned on.
  - if you specify `fix_X` that is no longer switchable, a warning is printed. However, if you specify `fix_that_never_existed`, an error is produced.

Syntax

You can set `NEW_MODE` from the command line using the `--new-mode` option, or by setting the new_mode system variable.

```
SET [GLOBAL|SESSION] new_mode = 'fix_1[,fix_2]...';
```

The session value only affects the current client and can be changed by the client when required. Setting the global value requires the SUPER privilege, and the change will affect any clients that connect from that point forward.

Example

```
SET GLOBAL new_mode = 'FIX_DISK_TMPTABLE_COSTS';
...
SET new_mode = 'FIX_DISK_TMPTABLE_COSTS,FIX_INDEX_STATS_FOR_ALL_NULLS';
...
SET SESSION new_mode = 'FIX_INDEX_STATS_FOR_ALL_NULLS';

SET new_mode = '';
```

Available NEW_MODE Flags

1. `FIX_DISK_TMPTABLE_COSTS`
2. `FIX_INDEX_STATS_FOR_ALL_NULLS`

FIX_DISK_TMPTABLE_COSTS

From MariaDB 11.8.4 to MariaDB 12.0. Starting from MariaDB 12.1.2, this behavior is enabled by default.

This flag improves the cost computation for using temporary tables in certain cases, including semi-join subquery materialization (MDEV-37723).

FIX_INDEX_STATS_FOR_ALL_NULLS

From MariaDB 11.4.9 to MariaDB 12.0.

This flag improves the selection of execution plans when indexed columns contain only NULL values (MDEV-36761). Starting from MariaDB 12.1.2, this behavior is enabled by default.

For proper application of the fix, engine-independent statistics must be collected for tables having columns with only NULL values:

```
ANALYZE TABLE table_name PERSISTENT FOR ALL;
```

or at least for indexed columns with only NULL values:

```
ANALYZE TABLE table_name PERSISTENT FOR COLUMNS (b) INDEXES (key_b);
```

This page is licensed: CC BY-SA / Gnu FDL

OLD_MODE

The old_mode system variable was introduced to replace the old variable with a new one with better granularity.

MariaDB supports several different modes which allow you to tune it to suit your needs.

The most important ways for doing this are with SQL_MODE and `OLD_MODE`.

SQL_MODE is used for getting MariaDB to emulate behavior from other SQL servers, while `OLD_MODE` is used for emulating behavior from older MariaDB or MySQL versions.

`OLD_MODE` is a string with different options separated by commas (‘`,`’) without spaces. The options are case insensitive.

Normally `OLD_MODE` should be empty. It’s mainly used to get old behavior when switching to MariaDB or to a new major version of MariaDB, until you have time to fix your application.

Between major versions of MariaDB various options supported by `OLD_MODE` may be removed. This is intentional as we assume that the application will be fixed to conform with the new MariaDB behavior between releases.

In other words, `OLD_MODE` options are by design deprecated from the day they were added and will eventually be removed as any other deprecated feature.

You can check the variable's local and global value with:

```
SELECT @@OLD_MODE, @@GLOBAL.OLD_MODE;
```

You can set the `OLD_MODE` either from the command line (option `--old-mode`) or by setting the old_mode system variable.

Non-default old mode features are deprecated by design, and a warning is issued when set.

Modes

The different values of `OLD_MODE` are:

2_DIGIT_YEAR

From MariaDB 13.0, restores support for the two-digit `YEAR(2)` data type, which is otherwise no longer accepted. Like other `OLD_MODE` flags, it is deprecated by design and intended only as a temporary migration aid.

COMPAT_5_1_CHECKSUM

From MariaDB 10.9, the –old option is deprecated. This option allows behaviour of the –old option for enabling the old-style checksum for `CHECKSUM TABLE` that MySQL 5.1 supports

IGNORE_INDEX_ONLY_FOR_JOIN

From MariaDB 10.9, the –old option is deprecated. This option allows behaviour of the –old option for disabling the index only for joins, but allow it for ORDER BY.

LOCK_ALTER_TABLE_COPY

From MariaDB 11.2. The non-locking copy ALTER introduced in MDEV-16329 should be beneficial in the vast majority of cases, but scenarios can exist which significantly impact performance. For example, RBR on tables without a primary key. When non-locking ALTER is performed on such a table, and DML affecting a large number of records is run in parallel, the ALTER can become extremely slow, and further DML can also be affected. If there is a chance of such scenarios (and no possibility of improving the schema by immediately adding primary keys), ALTER should be performed with the explicit LOCK=SHARED clause. If this is also impossible, then LOCK_ALTER_TABLE_COPY flag should be added to the old_mode variable until the schema can be improved.

NO_DUP_KEY_WARNINGS_WITH_IGNORE

Don't print duplicate key warnings when using INSERT IGNORE.

NO_NULL_COLLATION_IDS

A compatibility setting to support connectors (in particular MySQL Connector/NET) that give an exception when collation ids returned by SHOW COLLATION are NULL. It is automatically set when a MySQL Connector/NET connection is determined. From MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3.

NO_PROGRESS_INFO

Don't show progress information in SHOW PROCESSLIST.

OLD_FLUSH_STATUS

From MariaDB 11.5, restores the pre-MariaDB 11.5 behavior of FLUSH STATUS.

SESSION_USER_IS_USER

From MariaDB 11.7, restores the pre-MariaDB 11.7 behavior of SESSION_USER.

UTF8_IS_UTF8MB3

From MariaDB 10.6.1, the main name of the previous 3-byte `utf8` character set was changed to `utf8mb3`. When this flag is set, `utf8` is an alias for `utf8mb3`; when it is not set, `utf8` is an alias for `utf8mb4`.

> **Warning:**
>
> From MariaDB 13.1, `UTF8_IS_UTF8MB3` is no longer set by default. The default `old_mode` is now empty, so `utf8` is an alias for `utf8mb4` by default. The flag is also deprecated from MariaDB 13.1, and setting it raises a deprecation warning.

ZERO_DATE_TIME_CAST

When a TIME value is cast to a DATETIME, the date part will be `0000-00-00`, not CURRENT_DATE (as dictated by the SQL standard).

OLD_MODE and Stored Programs

In contrast to SQL_MODE, stored programs use the current user's `OLD_MODE` value.

Changes to `OLD_MODE` are not sent to replicas.

Examples

This example shows how to get a readable list of enabled OLD_MODE flags:

```
SELECT REPLACE(@@OLD_MODE, ',', '\n');
+--------------------------------------------------+
| REPLACE(@@OLD_MODE, ',', '\n')                   |
+--------------------------------------------------+
| NO_DUP_KEY_WARNINGS_WITH_IGNORE                  |
| NO_PROGRESS_INFO                                 |
+--------------------------------------------------+
```

Adding a new flag:

```
SET @@OLD_MODE = CONCAT(@@OLD_MODE, ',NO_PROGRESS_INFO');
```

If the specified flag is already ON, the above example has no effect but does not produce an error.

How to unset a flag:

```
SET @@OLD_MODE = REPLACE(@@OLD_MODE, 'NO_PROGRESS_INFO', '');
```

How to check if a flag is set:

```
SELECT @@OLD_MODE LIKE '%NO_PROGRESS_INFO';
+-----------------------------------+
| @@OLD_MODE LIKE '%NO_PROGESS_INFO' |
+-----------------------------------+
|                                 1 |
+-----------------------------------+
```

From MariaDB 11.3:

```
SET @@OLD_MODE = CONCAT(@@OLD_MODE, ',NO_PROGRESS_INFO');
Query OK, 0 rows affected, 1 warning (0.000 sec)

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------------+
| Level   | Code | Message                                                                     |
+---------+------+-----------------------------------------------------------------------------+
| Warning | 1287 | 'NO_PROGRESS_INFO' is deprecated and will be removed in a future release |
+---------+------+-----------------------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

SQL_MODE

MariaDB supports several different modes which allow you to tune it to suit your needs.

The most important ways for doing this are using `SQL_MODE` (controlled by the sql_mode system variable) and OLD_MODE (the old_mode system variable). `SQL_MODE` is used for getting MariaDB to emulate behavior from other SQL servers, while OLD_MODE is used for emulating behavior from older MariaDB or MySQL versions.

`SQL_MODE` is a string with different options separated by commas (‘`,`’) without spaces. The options are case insensitive.

You can check the local and global value of it with:

```
SELECT @@SQL_MODE, @@GLOBAL.SQL_MODE;
```

Setting SQL_MODE

Defaults

From version	Default sql_mode setting
MariaDB 10.2.4	STRICT_TRANS_TABLES, ERROR_FOR_DIVISION_BY_ZERO , NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION
MariaDB 10.1.7	NO_ENGINE_SUBSTITUTION, NO_AUTO_CREATE_USER
<= MariaDB 10.1.6	No value

You can set the `SQL_MODE` either from the command line (the `--sql-mode` option) or by setting the sql_mode system variable.

```
SET sql_mode = 'modes';
SET GLOBAL sql_mode = 'modes';
```

The session value only affects the current client, and can be changed by the client when required. To set the global value, the SUPER privilege is required, and the change affects any clients that connect from that point on.

SQL_MODE Values

The different `SQL_MODE` values are:

ALLOW_INVALID_DATES

Allow any day between 1-31 in the day part. This is convenient when you want to read in all (including wrong data) into the database and then manipulate it there.

Note, that MariaDB assumes that table content matches the current setting of this mode. If you enable it, insert some invalid dates into the table and then disable it for a session, the result will be not well defined — some queries

might return invalid dates, while other will not.

ANSI

Changes the SQL syntax to be closer to ANSI SQL.

Sets: REAL_AS_FLOAT, PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE.

It also adds a restriction: an error will be returned if a subquery uses an aggregating function with a reference to a column from an outer query in a way that cannot be resolved.

If set, SHOW CREATE TABLE output will not display MariaDB-specific table attributes.

ANSI_QUOTES

Changes the double-quote character (`"`) to be treated as a backtick (`` ` ``), the identifier quote character. This can break old MariaDB applications which assume that `"` is used as a string quote character.

DB2

Same as: PIPES_AS_CONCAT, ANSI_QUOTES , IGNORE_SPACE, DB2, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS

If set, SHOW CREATE TABLE output will not display MariaDB-specific table attributes.

EMPTY_STRING_IS_NULL

Oracle-compatibility option that translates Item_string created in the parser to `Item_null`, and translates binding an empty string as prepared statement parameters to binding `NULL`. For example, `SELECT '' IS NULL` returns TRUE, `INSERT INTO t1 VALUES ('')` inserts `NULL`.

ERROR_FOR_DIVISION_BY_ZERO

If not set, division by zero returns NULL. If set returns an error if one tries to update a column with 1/0 and returns a warning as well. Also see MDEV-8319. Default since MariaDB 10.2.4.

HIGH_NOT_PRECEDENCE

Compatibility option for MySQL 5.0.1 and before; This changes `NOT a BETWEEN b AND c` to be parsed as `(NOT a) BETWEEN b AND c`

IGNORE_BAD_TABLE_OPTIONS

If this is set generate a warning (not an error) for wrong table option in CREATE TABLE. Also, since 10.0.13, do not comment out these wrong table options in SHOW CREATE TABLE.

IGNORE_SPACE

Allow one to have spaces (including tab characters and new line characters) between function name and '('. The drawback is that this causes built in functions to become reserved words.

MAXDB

Same as: PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, MAXDB, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS, NO_AUTO_CREATE_USER.

Also has the effect of silently converting TIMESTAMP fields into DATETIME fields when created or modified.

If set, SHOW CREATE TABLE output will not display MariaDB-specific table attributes.

MSSQL

Additionally implies the following: PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS.

Additionally from MariaDB 10.4.5, implements a limited subset of Microsoft SQL Server's language. See SQL_MODE=MSSQL for more.

If set, SHOW CREATE TABLE output will not display MariaDB-specific table attributes.

MYSQL323

Same as: NO_FIELD_OPTIONS, HIGH_NOT_PRECEDENCE.

MYSQL40

Same as: NO_FIELD_OPTIONS, HIGH_NOT_PRECEDENCE.

NO_AUTO_CREATE_USER

Don't automatically create users with `GRANT` unless authentication information is specified. If none is provided, will produce a 1133 error: "Can't find any matching row in the user table". Default since MariaDB 10.1.7.

NO_AUTO_VALUE_ON_ZERO

If set, don't generate an AUTO_INCREMENT on INSERT of zero in an `AUTO_INCREMENT` column, or when adding an AUTO_INCREMENT attribute with the ALTER TABLE statement. Normally both `zero` and `NULL` generate new `AUTO_INCREMENT` values.

NO_BACKSLASH_ESCAPES

Disables using the backslash character `\` as an escape character within strings, making it equivalent to an ordinary character.

NO_DIR_IN_CREATE

Ignore all INDEX DIRECTORY and DATA DIRECTORY directives when creating a table. Can be useful on replica servers.

NO_ENGINE_SUBSTITUTION

If not set, if the available storage engine specified by a CREATE TABLE or ALTER TABLE is not available, a warning is given and the default storage engine is used instead. If set, generate a 1286 error when creating a table if the specified storage engine is not available. See also enforce_storage_engine. Default since MariaDB 10.1.7.

NO_FIELD_OPTIONS

Remove MariaDB-specific column options from the output of SHOW CREATE TABLE. This is also used by the portability mode of mariadb-dump.

NO_KEY_OPTIONS

Remove MariaDB-specific index options from the output of SHOW CREATE TABLE. This is also used by the portability mode of mariadb-dump.

NO_TABLE_OPTIONS

Remove MariaDB-specific table options from the output of SHOW CREATE TABLE. This is also used by the portability mode of mariadb-dump.

NO_UNSIGNED_SUBTRACTION

When enabled, subtraction results are signed even if the operands are unsigned.

NO_ZERO_DATE

Don't allow '0000-00-00' as a valid date in strict mode (produce a 1525 error). Zero dates can be inserted with IGNORE. If not in strict mode, a warning is generated.

NO_ZERO_IN_DATE

Don't allow dates where the year is not zero but the month or day parts of the date *are* zero (produce a 1525 error). For example, with this set, '0000-00-00' is allowed, but '1970-00-10' or '1929-01-00' are not. If the ignore option is used, MariaDB will insert '0000-00-00' for those types of dates. If not in strict mode, a warning is generated instead.

ONLY_FULL_GROUP_BY

For SELECT … GROUP BY queries, disallow SELECTing columns which are not referred to in the GROUP BY clause, unless they are passed to an aggregate function like COUNT() or MAX(). Produce a 1055 error.

ORACLE

In all versions of MariaDB up to MariaDB 10.2, this sets `sql_mode` that is equivalent to: PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS, NO_AUTO_CREATE_USER

From MariaDB 10.3, this mode also sets SIMULTANEOUS_ASSIGNMENT and configures the server to understand a large subset of Oracle's PL/SQL language instead of MariaDB's traditional syntax for stored routines. See SQL_MODE=ORACLE From MariaDB 10.3.

If set, SHOW CREATE TABLE output will not display MariaDB-specific table attributes.

PAD_CHAR_TO_FULL_LENGTH

Trailing spaces in CHAR columns are by default trimmed upon retrieval. With PAD_CHAR_TO_FULL_LENGTH enabled, no trimming occurs. Does not apply to VARCHARs.

PIPES_AS_CONCAT

Allows using the pipe character (ASCII 124) as string concatenation operator. This means that `"A" || "B"` can be used in place of `CONCAT("A", "B")`.

POSTGRESQL

Same as: PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, POSTGRESQL, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS.

If set, SHOW CREATE TABLE output will not display MariaDB-specific table attributes.

REAL_AS_FLOAT

`REAL` is a synonym for FLOAT rather than DOUBLE.

SIMULTANEOUS_ASSIGNMENT

Setting this makes the SET part of the UPDATE statement evaluate all assignments simultaneously, not left-to-right. From MariaDB 10.3.5.

STRICT_ALL_TABLES

Strict mode. Statements with invalid or missing data are aborted and rolled back. For a non-transactional storage engine with a statement affecting multiple rows, this may mean a partial insert or update if the error is found in a row beyond the first.

STRICT_TRANS_TABLES

Strict mode. Statements with invalid or missing data are aborted and rolled back, except that for non-transactional storage engines and statements affecting multiple rows where the invalid or missing data is not the first row, MariaDB will convert the invalid value to the closest valid value, or, if a value is missing, insert the column default value. Default since MariaDB 10.2.4.

TIME_ROUND_FRACTIONAL

With this mode unset, MariaDB truncates fractional seconds when changing precision to smaller. When set, MariaDB will round when converting to TIME, DATETIME and TIMESTAMP, and truncate when converting to DATE. Since MariaDB 10.4.1

TRADITIONAL

Makes MariaDB work like a traditional SQL server. Same as: STRICT_TRANS_TABLES, STRICT_ALL_TABLES, NO_ZERO_IN_DATE, NO_ZERO_DATE, ERROR_FOR_DIVISION_BY_ZERO, NO_ENGINE_SUBSTITUTION, NO_AUTO_CREATE_USER.

Strict Mode

A mode where at least one of `STRICT_TRANS_TABLES` or `STRICT_ALL_TABLES` is enabled is called *strict mode*.

With strict mode set (default from MariaDB 10.2.4), statements that modify tables (either transactional for `STRICT_TRANS_TABLES` or all for `STRICT_ALL_TABLES`) will fail, and an error will be returned instead. The IGNORE keyword can be used when strict mode is set to convert the error to a warning.

With strict mode not set (default in version <= MariaDB 10.2.3), MariaDB will automatically adjust invalid values, for example, truncating strings that are too long, or adjusting numeric values that are out of range, and produce a warning.

Statements that don't modify data will return a warning when adjusted regardless of mode.

SQL_MODE and Stored Programs

Stored programs and views always use the SQL_MODE that was active when they were created. This means that users can safely change session or global SQL_MODE; the stored programs they use will still work as usual.

It is possible to change session SQL_MODE within a stored program. In this case, the new SQL_MODE will be in effect only in the body of the current stored program. If it calls some stored procedures, they will not be affected by the change.

Some Information Schema tables (such as ROUTINES) and SHOW CREATE statements such as SHOW CREATE PROCEDURE show the SQL_MODE used by the stored programs.

Examples

This example shows how to get a readable list of enabled SQL_MODE flags:

```
SELECT REPLACE(@@SQL_MODE, ',', '\n');
+--------------------------------------------------------------------------+
| REPLACE(@@SQL_MODE, ',', '\n')                                           |
+--------------------------------------------------------------------------+
| STRICT_TRANS_TABLES
NO_ZERO_IN_DATE
NO_ZERO_DATE
NO_ENGINE_SUBSTITUTION |
+--------------------------------------------------------------------------+
```

Adding a new flag:

```
SET @@SQL_MODE = CONCAT(@@SQL_MODE, ',NO_ENGINE_SUBSTITUTION');
```

If the specified flag is already ON, the above example has no effect but does not produce an error.

How to unset a flag:

```
SET @@SQL_MODE = REPLACE(@@SQL_MODE, 'NO_ENGINE_SUBSTITUTION', '');
```

How to check if a flag is set:

```
SELECT @@SQL_MODE LIKE '%NO_ZERO_DATE%';
+-----------------------------------+
| @@SQL_MODE LIKE '%NO_ZERO_DATE%' |
+-----------------------------------+
|                                 1 |
+-----------------------------------+
```

Without and with strict mode:

```
CREATE TABLE strict (s CHAR(5), n TINYINT);

INSERT INTO strict VALUES ('MariaDB', '128');
Query OK, 1 row affected, 2 warnings (0.14 sec)

SHOW WARNINGS;
+---------+------+--------------------------------------------+
| Level   | Code | Message                                    |
+---------+------+--------------------------------------------+
| Warning | 1265 | Data truncated for column 's' at row 1     |
| Warning | 1264 | Out of range value for column 'n' at row 1 |
+---------+------+--------------------------------------------+
2 rows in set (0.00 sec)

SELECT * FROM strict;
+-------+------+
| s     | n    |
+-------+------+
| Maria |  127 |
+-------+------+

SET sql_mode='STRICT_TRANS_TABLES';

INSERT INTO strict VALUES ('MariaDB', '128');
ERROR 1406 (22001): Data too long for column 's' at row 1
```

Overriding strict mode with the IGNORE keyword:

```
INSERT IGNORE INTO strict VALUES ('MariaDB', '128');
Query OK, 1 row affected, 2 warnings (0.15 sec)
```

See Also

- SQL_MODE=MSSQL
- SQL_MODE=ORACLE

This page is licensed: CC BY-SA / Gnu FDL

Security

See Securing MariaDB.

Overview of security best practices for MariaDB, covering privilege separation, mandatory access control (SELinux), and vulnerability tracking.

See Encryption.

Enhance MariaDB Server security with encryption. This section covers data-at-rest and in-transit encryption, helping you protect sensitive information and meet compliance requirements.

See User Account Management.

Master user account management in MariaDB Server. This section covers creating, modifying, and revoking user privileges to ensure secure and controlled access to your databases.

See Authentication with Enterprise Server.

Learn about authentication with MariaDB Enterprise Server. This section covers advanced authentication methods & plugins to enhance security and integrate with enterprise identity management systems.

See Limiting Size of Created Disk Temporary Files and Tables.

The ability to limit the size of created disk temporary files and tables was introduced in MariaDB 11.5.

See Security Vulnerabilities (CVE) Fixed in MariaDB.

Lists of Security Vulnerabilities (CVE) fixed in MariaDB products.

This page is licensed: CC BY-SA / Gnu FDL

Securing MariaDB

See Running MariaDB as root.

Understand the implications of running MariaDB Server as root. This section highlights security risks and provides guidance on configuring MariaDB Server to operate with less privileged user accounts.

See MariaDB on SELinux.

Secure MariaDB Server with SELinux. This section guides you through configuring SELinux policies to enhance the security posture of your MariaDB deployments on Linux systems.

See Securing MariaDB Logs.

Learn how to harden MariaDB log files by implementing at-rest encryption, TLS for transit, strict OS permissions, and automated rotation to ensure data integrity and regulatory compliance.

See Security Vulnerabilities (CVE) Fixed in MariaDB.

This is the master list of CVEs fixed across all versions of MariaDB.

See Security Vulnerabilities (CVE) Fixed in Oracle MySQL That Did Not Exist in MariaDB.

This page lists all CVEs that were fixed in MySQL and mentioned in Oracle CPU Advisories, but that — to the best of our knowledge — were never present in MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

Running MariaDB as root

MariaDB should never be run as the system root user (this is unrelated to the MariaDB root user). If it is, any user with the `FILE` privilege can create or modify any files on the server as root.

If you attempt to run the server (`mariadbd`) as root, MariaDB normally returns the error **Fatal error: Please read "Security" section of the manual to find out how to run mariadbd as root!**. If you need to override this restriction, start `mariadbd` with the user=root option.

Better practice, and the default in most situations, is to use a separate user, exclusively used for MariaDB. In most distributions, this user is called `mysql`.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB on SELinux

Security-Enhanced Linux (SELinux) is a Linux kernel module that provides a framework for configuring mandatory access control (MAC) system for many resources on the system. It is enabled by default on some Linux distributions, including RHEL, CentOS, Fedora, and other similar Linux distribution. SELinux prevents programs from accessing files, directories or ports unless it is configured to access those resources.

Verifying Whether SELinux Is Enabled

To verify whether SELinux is enabled, execute the getenforce command. For example:

```
getenforce
```

Temporarily Putting mysqld Into Permissive Mode

When you are troubleshooting issues that you think SELinux might be causing, it can help to temporarily put `mysqld_t` into permissive mode. This can be done by executing the semanage command. For example:

```
sudo semanage permissive -a mysqld_t
```

If that solved the problem, it means that the current SELinux policy is the culprit. You need to adjust the SELinux policy or labels for MariaDB.

Configuring a MariaDB Server SELinux Policy

MariaDB Server should work with your default distribution policy (which is usually part of the `selinux-policy` or `selinux-policy-targeted` system package). If you use `mysqld_safe`, you will need an additional policy file, `mariadb.pp`, which is installed together with the MariaDB Server. It will be loaded automatically if you have `/usr/sbin/semodule` installed, but you can load it manually anytime with

```
/usr/sbin/semodule -i /usr/share/mariadb/policy/selinux/mariadb.pp
```

Note that this policy file extends, rather than replaces the system policy.

Setting File Contexts

SELinux uses file contexts as a way to determine who should be able to access that file.

File contexts are managed with the semanage fcontext and restorecon commands.

On many systems, the semanage utility is installed by the `policycoreutils-python` package, and the restorecon utility is installed by the `policycoreutils` package. You can install these with the following command:

```
sudo yum install policycoreutils policycoreutils-python
```

A file or directory's current context can be checked by executing `ls` with the `--context` or `--scontext` options.

Setting the File Context for the Data Directory

If you use a custom directory for datadir, then you may need to set the file context for that directory. The SELinux file context for MariaDB data files is `mysqld_db_t`. You can determine if this file context is present on your system and which files or directories it is associated with by executing the following command:

```
sudo semanage fcontext --list | grep mysqld_db_t
```

If you would like to set the file context for your custom directory for your datadir, then that can be done by executing the semanage fcontext and restorecon commands. For example:

```
sudo semanage fcontext -a -t mysqld_db_t "/mariadb/data(/.*)?"
sudo restorecon -Rv /mariadb/data
```

If you would like to check the current file context, you can do so by executing `ls` with the `--context` or `--scontext` options. For example:

```
ls --directory --scontext /mariadb/data
```

Setting the File Context for Log Files

If you use a custom directory for log files, then you may need to set the file context for that directory. The SELinux file context for MariaDB log files is `mysqld_log_t`. You can determine if this file context is present on your system and which files or directories it is associated with by executing the following command:

```
sudo semanage fcontext --list | grep mysqld_log_t
```

If you would like to set the file context for your custom directory for log files, then that can be done by executing the semanage fcontext and restorecon commands. For example:

```
sudo semanage fcontext -a -t mysqld_log_t "/var/log/mysql(/.*)?"
sudo restorecon -Rv /var/log/mysql
```

If you would like to check the current file context, you can do so by executing `ls` with the `--context` or `--scontext` options. For example:

```
ls --directory --scontext /var/log/mysql
```

Setting the File Context for Option Files

If you use a custom directory for option files, then you may need to set the file context for that directory. The SELinux file context for MariaDB option files is `mysqld_etc_t`. You can determine if this file context is present on your system and which files or directories it is associated with by executing the following command:

```
sudo semanage fcontext --list | grep mysqld_etc_t
```

If you would like to set the file context for your custom directory for option files, then that can be done by executing the semanage fcontext and restorecon commands. For example:

```
sudo semanage fcontext -a -t mysqld_etc_t "/etc/mariadb(/.*)?"
sudo restorecon -Rv /etc/mariadb
```

If you would like to check the current file context, you can do so by executing `ls` with the `--context` or `--scontext` options. For example:

```
ls --directory --scontext /etc/mariadb
```

Setting the Unix Socket

A custom location for the socket means it needs to have the right file context of `mysqld_var_run_t` for permitted application to connect to the socket.

```
sudo semanage fcontext -a -t mysqld_var_run_t "/mariadb/run/mariadb.sock"
```

A newly created socket with get the right context.

Allowing Access to the Tmpfs File Context

If you wanted to mount your tmpdir on a `tmpfs` file system or wanted to use a `tmpfs` file system on `/run/shm`, then you might need to allow `mysqld_t` to have access to a couple tmpfs-related file contexts. For example:

```
cd /usr/share/mysql/policy/selinux/
tee ./mysqld_tmpfs.te <<EOF
module mysqld_tmpfs 1.0;

require {
type tmpfs_t;
type mysqld_t;
class dir { write search read remove_name open getattr add_name };
class file { write getattr read lock create unlink open };
}

allow mysqld_t tmpfs_t:dir { write search read remove_name open getattr add_name };

allow mysqld_t tmpfs_t:file { write getattr read lock create unlink open }
EOF
sudo checkmodule -M -m mysqld_tmpfs.te -o mysqld_tmpfs.mod
sudo semodule_package -m mysqld_tmpfs.mod -o mysqld_tmpfs.pp
sudo semodule -i mysqld_tmpfs.pp
```

Troubleshooting SELinux Issues

You might need to troubleshoot SELinux-related issues in cases, such as:

- MariaDB is using a non-default port.
- MariaDB is reading from or writing to some files (datadir, log files, option files, etc.) located at non-default paths.
- MariaDB is using a plugin that requires access to resources that default installations do not use.

File System Permission Errors

If the file system permissions for some MariaDB directory look fine, but the MariaDB error log still has errors that look similar to the following:

```
130321 11:50:51 mysqld_safe Starting mysqld daemon with databases from /datadir
...
2013-03-21 11:50:52 2119 [Warning] Can't create test file /datadir/
2013-03-21 11:50:52 2119 [Warning] Can't create test file /datadir/
...
2013-03-21 11:50:52 2119 [ERROR] /usr/sbin/mysqld: Can't create/write to file
    '/datadir/boxy.pid' (Errcode: 13 - Permission denied)
2013-03-21 11:50:52 2119 [ERROR] Can't start server: can't create PID file:
    Permission denied
130321 11:50:52 mysqld_safe mysqld from pid file /datadir/boxy.pid ended
```

Then check SELinux's `/var/log/audit/audit.log` for log entries that look similar to the following:

```
type=AVC msg=audit(1363866652.030:24): avc:  denied  { write } for  pid=2119
    comm="mysqld" name="datadir" dev=dm-0 ino=394
    scontext=unconfined_u:system_r:mysqld_t:s0
    tcontext=unconfined_u:object_r:default_t:s0 tclass=dir
```

If you see any entries that look similar to this, then you most likely need to adjust the file contexts for some files or directories. See Setting File Contexts for more information on how to do that.

SELinux and MariaDB On a Different Port

TCP and UDP ports are enabled for permission to bind too. If you are using a different port, or some Galera ports, configure SELinux to be able to use those ports:

```
sudo semanage port -a -t mysqld_port_t -p tcp 3307
```

Generating SELinux Policies with audit2allow

In some cases, a MariaDB system might need non-standard policies. It is possible to create these policies from the SELinux audit log using the audit2allow utility. The semanage and semodule utilities will also be needed.

On many systems, the audit2allow and semanage utilities are installed by the `policycoreutils-python` package, and the semodule utility is installed by the `policycoreutils` package. You can install these with the following command:

```
sudo yum install policycoreutils policycoreutils-python
```

The following process can be used to generate a policy from the audit log:

- Remove dontaudits from the policy:

```
sudo semodule -DB
```

- Temporarily put `mysqld_t` into permissive mode. For example:

```
sudo semanage permissive -a mysqld_t
```

- Start MariaDB.
- Do whatever was causing SELinux errors.
- Use the generated audit log to create a policy:

```
sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local
sudo semodule -i mariadb_local.pp
```

- Pull `mysqld_t` out of permissive mode. For example:

```
sudo semanage permissive -d mysqld_t
```

- Restore dontaudits for the policy:

```
sudo setmodule -B
```

The same procedure can be used if MariaDB starts but SELinux prevents it from functioning correctly. For example, SELinux may prevent PAM plugin from authenticating users. The solution is the same — enable auditing, switch to permissive, do, whatever SELinux didn't allow you to, create a policy from the audit log.

When you discover any needed SELinux permissions, please report the needed permissions to your operating system bug tracking so all users can benefit from your work (e.g. Red Hat Bugzilla).

This page is licensed: CC BY-SA / Gnu FDL

Encryption

MariaDB's security architecture distinguishes between data moving across the network (Data-in-Transit) and data stored on disk (Data-at-Rest).

See TLS and Cryptography Libraries.

MariaDB links to cryptography libraries (OpenSSL, wolfSSL, GnuTLS, Schannel) either statically or dynamically. How to verify the active library and version.

See Data-in-Transit Encryption.

Data-in-Transit Encryption

- Protects credentials and query results from "man-in-the-middle" attacks during client-server communication.
- Uses the TLS protocol. It handles the handshake, identity verification, and encryption of the network stream.
- Utilizes Asymmetric Key Pairs (Public/Private keys) and Certificates (PEM/CRT files) managed by libraries like OpenSSL.
- Defined in the `[mariadb]` section using `ssl_cert`, `ssl_key`, and `ssl_ca`.

See Data-at-Rest Encryption.

Data-at-Rest Encryption

- Protects physical data files (InnoDB/Aria tables, Redo logs, and Binary logs) if the storage media or backups are stolen.
- Uses Symmetric Encryption (typically AES) managed by specialized Key Management Plugins.
- Uses Symmetric Keys identified by a `Key ID`. These are fetched from a local file, AWS KMS, or HashiCorp Vault.
- Enabled via variables like `innodb_encrypt_tables` and requires a specific plugin (e.g., `file_key_management`) to be loaded.

This page is licensed: CC BY-SA / Gnu FDL

TLS and Cryptography Libraries

> These libraries provide the cryptographic foundations for both network (or data-in-transit) encryption (TLS) and data-at-rest encryption (TDE).

When MariaDB Server is compiled with TLS[26] and cryptography support, it is usually either statically linked with MariaDB's bundled TLS and cryptography library or dynamically linked with the system's OpenSSL library. MariaDB's bundled TLS library is either wolfSSL or yaSSL, depending on the server version.

When a MariaDB client or client library is compiled with TLS and cryptography support, it is usually either statically linked with MariaDB's bundled TLS and cryptography library or dynamically linked with the system's TLS and cryptography library, which can be OpenSSL, GnuTLS, or Schannel.

Checking Dynamically vs. Statically Linked

Dynamically linking MariaDB to the system's TLS and cryptography library can often be beneficial, since this allows you to fix bugs in the system's TLS and cryptography library independently of MariaDB. For example, when information on the Heartbleed Bug in OpenSSL was released in 2014, the bug could be mitigated by simply updating your system to use a fixed version of the OpenSSL library, then restarting the MariaDB Server.

LINUX

You can verify that `mariadbd` is dynamically linked to the OpenSSL shared library on your system with this ldd command:

```
ldd $(which mariadbd) | grep -E '(libssl|libcrypto)'
        libssl.so.10 => /lib64/libssl.so.10 (0x00007f8736386000)
        libcrypto.so.10 => /lib64/libcrypto.so.10 (0x00007f8735f25000)
```

MACOS

You can verify that `mariadbd` is dynamically linked to the OpenSSL shared library on your system with this command:

```
otool -L $(which mariadbd) | grep -E '(libssl|libcrypto)'
    /opt/homebrew/opt/openssl@3/lib/libssl.3.dylib (compatibility version 3.0.0, current version 3.0.0)
    /opt/homebrew/opt/openssl@3/lib/libcrypto.3.dylib (compatibility version 3.0.0, current version
3.0.0)
```

If the command does not return any results, either your `mariadbd` is statically linked to the TLS and cryptography library on your system or your `mariadbd` is not built with TLS and cryptography support at all.

Checking If the Server Uses OpenSSL

If you aren't sure whether your server is linked with OpenSSL or the bundled TLS library, then you can check the value of the have_openssl system variable. For example:

```
SHOW GLOBAL VARIABLES LIKE 'have_openssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_openssl  | YES   |
+---------------+-------+
```

Checking the Server's OpenSSL Version

If you want to see what version of OpenSSL your server is using, check the value of the version_ssl_library system variable:

```
SHOW GLOBAL VARIABLES LIKE 'version_ssl_library';
+--------------------+--------------------------------+
| Variable_name      | Value                          |
+--------------------+--------------------------------+
| version_ssl_library | OpenSSL 1.0.1e-fips 11 Feb 2013 |
+--------------------+--------------------------------+
```

Note that the version returned by this system variable does not necessarily correspond to the exact version of the OpenSSL package installed on the system. OpenSSL shared libraries tend to contain interfaces for multiple versions at once, to allow for backward compatibility. Therefore, if the OpenSSL package installed on the system is newer than the OpenSSL version that the MariaDB Server binary was built with, the MariaDB Server binary might use one of the interfaces for an older version. See MDEV-15848 for more information:

```
$ cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.5 (Maipo)
$ rpm -q openssl
openssl-1.0.2k-12.el7.x86_64
$ mariadb -u root --batch --execute="SHOW GLOBAL VARIABLES LIKE 'version_ssl_library';"
Variable_name   Value
version_ssl_library     OpenSSL 1.0.1e-fips 11 Feb 2013
$ ldd $(which mariadbd) | grep libcrypto
        libcrypto.so.10 => /lib64/libcrypto.so.10 (0x00007f3dd3482000)
$ readelf -a /lib64/libcrypto.so.10 | grep SSLeay_version
  1374: 000000000006f5d0    21 FUNC    GLOBAL DEFAULT   13 SSLeay_version@libcrypto.so.10
  1375: 000000000006f5f0    21 FUNC    GLOBAL DEFAULT   13 SSLeay_version@OPENSSL_1.0.1
  1377: 000000000006f580    70 FUNC    GLOBAL DEFAULT   13 SSLeay_version@@OPENSSL_1.0.2
```

FIPS Certification

Federal Information Processing Standards (FIPS) are standards published by the U.S. federal government that are used to establish requirements for various aspects of computer systems. FIPS 140-2 is a set of standards for security requirements for cryptographic modules.

This standard is relevant when discussing the TLS and cryptography libraries used by MariaDB. Some of these libraries have been certified to meet the standards set by FIPS 140-2.

FIPS Certification by OpenSSL

The OpenSSL library has a special FIPS mode that has been certified to meet the FIPS 140-2 standard. In FIPS mode, only algorithms and key sizes that meet the FIPS 140-2 standard are enabled by the library.

MariaDB does not yet support enabling FIPS mode within the database server. See MDEV-20260 for more information. Therefore, if you would like to use OpenSSL's FIPS mode with MariaDB, then you would either need to enable FIPS mode at the kernel level or enable it via the OpenSSL configuration file, system-wide or only for the MariaDB process.. See the following resources for more information on how to do that:

- Red Hat Enterprise Linux 7: Security Guide: Chapter 8. Federal Standards and Regulations
- Ubuntu Security Certifications Documentation: FIPS for Ubuntu 16.04 and 18.04
- OpenSSL 1.0.2, configuration file method
- OpenSSL 3.0 configuration file method

FIPS Certification by wolfSSL

The standard version of the wolfSSL library has not been certified to meet the FIPS 140-2 standard, but a special "FIPS-ready" version has been certified. Unfortunately, the "FIPS-ready" version of wolfSSL uses a license that is incompatible with MariaDB's license, so it cannot be used with MariaDB.

FIPS Certification by yaSSL

The yaSSL library has not been certified to meet the FIPS 140-2 standard.

Libraries Used by Each Platform and Package

MariaDB Server

MariaDB Server on Windows

MariaDB Server is statically linked with the bundled wolfSSL library in MSI and ZIP packages on Windows.

MariaDB Server on Linux

MariaDB Server in Binary Tarballs

MariaDB Server is statically linked with the bundled wolfSSL library in binary tarballs on Linux.

MariaDB Server in DEB Packages

MariaDB Server is dynamically linked with the system's OpenSSL library in .deb packages.

See Differences in MariaDB in Debian (and Ubuntu) for more information.

MariaDB Server in RPM Packages

MariaDB Server is dynamically linked with the system's OpenSSL library in .rpm packages.

MariaDB Clients and Utilities

MariaDB Connector/C has been included with MariaDB Server, and the bundled and the clients and utilities are linked with it. On some platforms, MariaDB Connector/C and these clients and utilities may use a different TLS library than the one used by MariaDB Server and libmysqlclient.

MariaDB Clients and Utilities on Windows

MariaDB's clients and utilities and MariaDB Connector/C are dynamically linked with the system's Schannel libraries in MSI and ZIP packages on Windows. libmysqlclient is still statically linked with the bundled wolfSSL library.

MariaDB Clients and Utilities on Linux

MariaDB Clients and Utilities in Binary Tarballs

MariaDB's clients and utilities and MariaDB Connector/C are statically linked with the GnuTLS library in binary tarballs on Linux. libmysqlclient is still statically linked with the bundled wolfSSL library.

MariaDB Clients and Utilities in DEB Packages

MariaDB's clients and utilities, libmysqlclient, and MariaDB Connector/C are dynamically linked with the system's OpenSSL library in .deb packages.

See Differences in MariaDB in Debian (and Ubuntu) for more information.

MariaDB Clients and Utilities in RPM Packages

MariaDB's clients and utilities, libmysqlclient, and MariaDB Connector/C are dynamically linked with the system's OpenSSL library in .rpm packages.

Updating Dynamically Linked OpenSSL Libraries on Linux

When the MariaDB Server or clients and utilities are dynamically linked to the system's OpenSSL library, it makes it very easy to update the libraries. The information below will show how to update these libraries for each platform.

Updating Dynamically Linked OpenSSL Libraries with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to update the libraries using yum or dnf. Starting with RHEL 8 and Fedora 22, `yum` has been replaced by `dnf`, which is the next major version of `yum`. However, `yum` commands still work on many systems that use `dnf`. Update the package by executing the following command:

```
sudo yum update openssl
```

Then restart MariaDB server and any clients or applications that use the library.

Updating Dynamically Linked OpenSSL Libraries with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to recommended to update the libraries using apt-get. First update the package cache by executing the following command:

```
sudo apt update
```

Then update the package by executing the following command:

```
sudo apt-get update openssl
```

Then restart MariaDB server and any clients or applications that use the library.

Updating Dynamically Linked OpenSSL Libraries with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to update the libraries using zypper. Update the package by executing the following command:

```
sudo zypper update openssl
```

Then restart MariaDB server and any clients or applications that use the library.

This page is licensed: CC BY-SA / Gnu FDL

Data-in-Transit Encryption

Protect your data as it moves across the network with Data-in-Transit Encryption. By leveraging the TLS (Transport Layer Security) protocol, MariaDB ensures that credentials, queries, and result sets are encrypted between the client and server. This prevents “man-in-the-middle” attacks and unauthorized eavesdropping on sensitive information.

This guide covers the essentials of securing your network traffic—from configuring SSL certificates and private keys to enforcing secure connections for all users. Secure your communications layer to maintain data integrity and confidentiality in any environment.

See Secure Connections Overview.

Conceptual overview of data-in-transit encryption in MariaDB, discussing supported TLS libraries (OpenSSL, wolfSSL), protocol versions (`tls_version`), and certificate verification.

See Securing Connections for Client and Server.

Complete MariaDB security guide. Complete resource for user management, access control, SSL/TLS encryption, and audit policies with comprehensive examples.

See Enabling TLS on MariaDB Server.

Secure, automatic SSL in MariaDB 11.4+ which is enabled by default and does not require any configuration

See Enabling TLS on MariaDB Server.

Step-by-step instructions for manual configuration of SSL in MariaDB via system variables like `ssl_cert` , `ssl_key` , and `ssl_ca` in the configuration file.

See Replication with Secure Connections.

A guide to securing replication traffic between primary and replica servers, covering the use of `CHANGE MASTER TO` options (e.g., `MASTER_SSL`) and mutual authentication.

See Certificate Creation with OpenSSL.

Complete OpenSSL TLS certificate guide: generate CA key/cert and server key/CSR, sign X509 with openssl x509 -CA/-CAkey, and verify certificates.

See SSL/TLS System Variables.

Reference list of system variables related to TLS configuration, such as `ssl_cipher`, `ssl_crl`, and `have_ssl`, used to manage and monitor encryption settings.

See Using TLSv1.3.

This page is licensed: CC BY-SA / Gnu FDL

Secure Connections Overview

Prior to MariaDB 11.4, by default, MariaDB transmits data between the server and clients without encrypting it. This is generally acceptable when the server and client run on the same host or in networks where security is guaranteed through other means. However, in cases where the server and client exist on separate networks or they are in a high-risk network, the lack of encryption does introduce security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

To mitigate this concern, MariaDB allows you to encrypt data in transit between the server and clients using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix `ssl_`, but internally, MariaDB only supports its secure successors.

Checking MariaDB Server for TLS Support

In order for MariaDB Server to use TLS, it needs to be compiled with TLS support. All MariaDB packages distributed by MariaDB Foundation and MariaDB Corporation are compiled with TLS support.

If you aren't sure whether your MariaDB Server binary was compiled with TLS support, then you can check the value of the have_ssl system variable. For example:

```
SHOW GLOBAL VARIABLES LIKE 'have_ssl';
+---------------+----------+
| Variable_name | Value    |
+---------------+----------+
| have_ssl      | DISABLED |
+---------------+----------+
```

The possible values are:

- If it is `DISABLED`, then the server was compiled with TLS support, but TLS is not enabled.
- If it is `YES`, then the server was compiled with TLS support, and TLS is enabled.
- If it is `NO`, then the server was not compiled with TLS support.

TLS Libraries

When MariaDB is compiled with TLS and cryptography support, it is usually either statically linked with MariaDB's bundled TLS and cryptography library, which might be wolfSSL or yaSSL, or dynamically linked with the system's TLS and cryptography library, which might be OpenSSL, GnuTLS, or Schannel.

See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

TLS Protocol Versions

There are 4 versions of the TLS protocol:

- `TLSv1.0`
- `TLSv1.1`
- `TLSv1.2`
- `TLSv1.3`

Enabling Specific TLS Protocol Versions

In some cases, it might make sense to only enable specific TLS protocol versions. For example, it would make sense if your organization has to comply with a specific security standard. It would also make sense if a vulnerability is found in a specific TLS protocol version, and you would like to ensure that your server does not use the vulnerable protocol version.

The PCI DSS v3.2 recommends using a minimum protocol version of TLSv1.2.

On the **server** side, users can enable specific TLS protocol versions by setting the tls_version system variable. This system variable accepts a comma-separated list of TLS protocol versions. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted. This system variable can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
tls_version = TLSv1.2,TLSv1.3
```

You can check which TLS protocol versions are enabled on a server by executing SHOW GLOBAL VARIABLES. For example:

```
SHOW GLOBAL VARIABLES LIKE 'tls_version';
```

On the **client** side, users can enable specific TLS protocol versions by setting the `--tls-version` option. This option accepts a comma-separated list of TLS protocol versions. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted. For example, to specify this option in a relevant client option group in an option file, you could set the following:

```
[client-mariadb]
tls_version = TLSv1.2,TLSv1.3
```

Or if you wanted to specify it on the command-line with the mariadb client, then you could execute something like this:

```
$ mariadb -u myuser -p -h myserver.mydomain.com \
   --ssl \
   --tls-version="TLSv1.2,TLSv1.3"
```

TLS Protocol Version Support

The TLS protocol versions that are supported depend on the underlying TLS library used by the specific MariaDB binary.

TLS Library	Supported TLS Protocol Versions
openSSL	TLSv1, TLSv1.1, TLSv1.2, TLSv1.3
wolfSSL	TLSv1, TLSv1.1, TLSv1.2, TLSv1.3
yaSSL	TLSv1, TLSv1.1
Schannel	TLSv1, TLSv1.1, TLSv1.2
GnuTLS	TLSv1, TLSv1.1, TLSv1.2, TLSv1.3

See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used by the server and by clients on each platform.

TLS Protocol Version Support in OpenSSL

MariaDB binaries built with the OpenSSL library (OpenSSL 1.0.1 or later) support TLSv1.1 and TLSv1.2.

MariaDB binaries built with the OpenSSL library (OpenSSL 1.1.1 or later) support TLSv1.3.

If your MariaDB Server binary is built with OpenSSL, then you can set the ssl_cipher system variable to values like `SSLv3` or `TLSv1.2` to allow all SSLv3.0 or all TLSv1.2 ciphers. However, this does not necessarily limit the protocol version to TLSv1.2. See MDEV-14101 for more information about that.

Note that the `TLSv1.3` ciphers cannot be excluded when using OpenSSL, even by using the ssl_cipher system variable. See Using TLSv1.3 for details.

SSLv3.0 is known to be vulnerable to the POODLE attack, so it should not be used. SSLv2.0 and SSLv3.0 are disabled for MariaDB Server binaries linked with OpenSSL.

TLS Protocol Version Support in wolfSSL

MariaDB binaries built with the bundled wolfSSL library support TLSv1.0, TLSv1.1, TLSv1.2, and TLSv1.3.

TLS Protocol Version Support in yaSSL

MariaDB binaries built with the bundled yaSSL library support SSLv3.0, TLSv1.0, and TLSv1.1.

SSLv3.0 is known to be vulnerable to the POODLE attack, so it should not be used. SSLv2.0 and SSLv3.0 are disabled for MariaDB Server binaries linked with yaSSL.

TLS Protocol Version Support in Schannel

MariaDB binaries built with the Schannel library support different versions of TLS on different versions of Windows. See the Protocols in TLS/SSL (Schannel SSP) documentation from Microsoft to determine which versions of TLS are supported on each version of Windows.

TLS Protocol Version Support in GnuTLS

MariaDB binaries built with the GnuTLS library support TLSv1.0, TLSv1.1, TLSv1.2, and TLSv1.3.

Enabling TLS

See Securing Connections for Client and Server for information on how to enable TLS on the client and server.

Certificate Verification

Certificate verification is how TLS authenticates its connections by verifying that it is talking to who it says it is. There are multiple components to this verification process:

- Was the certificate signed by a trusted Certificate Authority (CA)?
- Is the certificate expired?
- Is the certificate on my Certificate Revocation List (CRL)?
- Does the certificate belong to who I believe that I'm communicating with?

Certificate Authorities (CAs)

Certificate Authorities (CAs) are entities that you trust to sign TLS certificates. Your organization might have its own internal CA, or it might use trusted third-party CAs.

CAs are specified on the server and client by using the ssl_ca and ssl_capath options.

The ssl_ca option defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs). This option requires that you use the absolute path, not a relative path.

The ssl_capath option defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA). This option requires that you use the absolute path, not a relative path. The ssl_capath option is only supported if the server or client was built with OpenSSL, wolfSSL, or yaSSL. If the client was built with GnuTLS or Schannel, then the ssl_capath option is not supported.

See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

The directory specified by ssl_capath needs to be run through the openssl rehash command. For example, if the following is configured:

```
ssl_capath=/etc/my.cnf.d/certificates/ca/
```

Then execute the following command:

```
openssl rehash /etc/my.cnf.d/certificates/ca/
```

Requiring a Specific Certificate Authority (CA)

The server can require a specific Certificate Authority (CA) for a client if the client's user account has been defined with `REQUIRE ISSUER`. See Securing Connections for Client and Server: Requiring TLS for more information.

Certificate Revocation Lists (CRLs)

Certificate Revocation Lists (CRLs) are lists of certificates that have been revoked by the Certificate Authority (CA) before they were due to expire.

CRLs are specified on the server and client by using the ssl_crl and ssl_crlpath options.

The ssl_crl option defines a path to a PEM file that should contain one or more X509 revoked certificates. This option requires that you use the absolute path, not a relative path. For servers, the ssl_crl option is only valid if the server was built with OpenSSL. If the server was built with wolfSSL or yaSSL, then the ssl_crl option is not supported. For clients, the ssl_crl option is only valid if the client was built with OpenSSL or Schannel. Likewise, if the client was built with GnuTLS, wolfSSL or yaSSL, then the ssl_crl option is not supported.

The ssl_crlpath option defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate. This option requires that you use the absolute path, not a relative path. The ssl_crlpath option is only supported if the server or client was built with OpenSSL. If the server was built with wolfSSL or yaSSL, then the ssl_crlpath option is not supported. Likewise, if the client was built with GnuTLS, Schannel, wolfSSL, or yaSSL, then the ssl_crlpath option is not supported.

See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

The directory specified by ssl_crlpath needs to be run through the openssl rehash command. For example, if the following is configured:

```
ssl_crlpath=/etc/my.cnf.d/certificates/crl/
```

Then execute the following command:

```
openssl rehash /etc/my.cnf.d/certificates/crl/
```

Server Certificate Verification

Normally the TLS implementation performs numerous checks to verify whether the certificate is valid. It should be within its validity period, not revoked, signed by a trusted certificate authority, belong to the host that uses it.

Clients and utilities enable all these checks by default. Last two checks can be disabled with an option, such as `disable-ssl-verify-server-cert`. Before MariaDB 11.3, all server certificate verification checks were disabled by default, and could be enabled with the `ssl-verify-server-cert` option.

To verify whether the server's certificate belong to server's host, clients and utilities will check the **Common Name (CN)** attribute located in the **Subject** field of the certificate against the server's host name and IP address. If the **Common Name (CN)** matches either of those, then the certificate is verified.

Server Certificate Verification with Subject Alternative Names (SANs)

The **Subject Alternative Name (SAN)** field, which is an X.509v3 extension, can also be used for server certificate verification, if it is present in the server certificate. This field is also sometimes called **subjectAltName**. When using a client or utility that supports server certificate verification with **subjectAltName** fields, if the server certificate contains any **subjectAltName** fields, then those fields will also be checked against the server's host name and IP address.

Whether server certificate verification with **subjectAltName** fields is supported depends on the underlying TLS library used by the client or utility.

See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

SAN Support with OpenSSL, wolfSSL, and yaSSL

For clients and utilities built with OpenSSL (OpenSSL 1.0.2 or later), support for server certificate verification with **subjectAltName** fields that contain the server's **host name** was added.

For clients and utilities built with OpenSSL (OpenSSL 1.0.2 or later), support for server certificate verification with **subjectAltName** fields that contain the server's **IP address** was added.

This support also applies to other TLS libraries that use OpenSSL's API. In OpenSSL's API, server certificate verification with **subjectAltName** fields depends on the X509_check_host and X509_check_ip functions. These functions are supported in the following TLS libraries:

- OpenSSL 1.0.2 or later
- wolfSSL

And they are **not** supported in the following TLS libraries:

- yaSSL

MariaDB's RPM packages were built with OpenSSL 1.0.1 on RHEL 7 and CentOS 7, even after OpenSSL 1.0.2 became available on those distributions. As a side effect, the clients and utilities bundled in these packages did not support server certificate verification with the **subjectAltName** field, even if the packages were installed on a system that had OpenSSL 1.0.2 installed. MariaDB's RPM packages on RHEL 7 and CentOS 7 are built with OpenSSL 1.0.2.

SAN Support with Schannel

For clients and utilities linked with Schannel, support for server certificate verification with **subjectAltName** fields was added in MariaDB Connector/C 3.0.2. See CONC-250 for more information.

SAN Support with GnuTLS

For clients and utilities linked with GnuTLS, support for server certificate verification with **subjectAltName** fields was added in MariaDB Connector/C 3.0.0. See CONC-250 for more information.

Client Certificate Verification

The server verifies a client certificate by checking the client's known `SUBJECT` against the **Subject** attribute in the client's certificate. This is only done for user accounts that have been defined with `REQUIRE SUBJECT`. See Securing Connections for Client and Server: Requiring TLS for more information.

See Also

- Mission Impossible: Zero-Configuration SSL (mariadb.org)
- Securing Connections for Client and Server
- Using TLSv1.3
- Certificate Creation with OpenSSL
- Replication with Secure Connections
- Securing Communications in Galera Cluster
- SSL/TLS System Variables
- Data-at-Rest Encryption
- Cyberciti tutorial: How to set up MariaDB SSL and secure connections from clients

This page is licensed: CC BY-SA / Gnu FDL

Securing Connections for Client and Server

Starting from 11.4 MariaDB, it encrypts the transmitted data between the server and clients by default unless the server and client run on the same host.

Before that the default behavior was to transmit the data unencrypted over the network introducing a security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

The data in transit are encrypted (by default or if enabled manually) using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix `ssl_`, but internally, MariaDB only supports its secure successors.

Enabling TLS

Enabling TLS for MariaDB Server

> **CURRENT**
>
> MariaDB enables TLS automatically. Certificates are generated on startup and only stored in memory. Certificate verification is enabled by default on the client side and certificates are verified if the authentication plugin itself is MitM safe (`mysql_native_password`, `ed25519`, `parsec`).
>
> **< 11.4**
>
> In order to enable TLS in a MariaDB server, you need to generate TLS certificates and configure the server to use them.
>
> To do that there are a number of system variables that you need to set, such as:
>
> - You need to set the path to the server's X509 certificate by setting the ssl_cert system variable.
> - You need to set the path to the server's private key by setting the ssl_key system variable.

- Unless you use a certificate which was signed by a trusted certificate authority (CA) you need to set the path to the certificate authority chain that can verify the server's certificate by setting either the ssl_ca or the ssl_capath system variables.
- If you want to restrict the server to certain ciphers, then you also need to set the ssl_cipher system variable.

For example, to set these variables for the server, add the system variables to a relevant server option group in an option file:

```
[mariadb]
...
ssl_cert = /etc/my.cnf.d/certificates/server-cert.pem
ssl_key = /etc/my.cnf.d/certificates/server-key.pem
ssl_ca = /etc/my.cnf.d/certificates/ca.pem
```

And then restart the server to make the changes persistent.

Once the server is back up, you can check that TLS is enabled by checking the value of the have_ssl system variable. For example:

```
SHOW VARIABLES LIKE 'have_ssl';

+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_ssl      | YES   |
+---------------+-------+
```

Reloading the Server's Certificates and Keys Dynamically

The `FLUSH SSL` command can be used to dynamically reinitialize the server's TLS context.

See FLUSH SSL for more information.

Enabling TLS for MariaDB Clients

Different clients and utilities may use different methods to enable TLS. You can let the client to use TLS without specifying client-side certificate — this is called a **one-way TLS** below — to have the connection encrypted. Or you can additionally provide client-side certificate — this is **two-way TLS** — which will allow the server to do the certificate based client authentication.

Enabling One-Way TLS for MariaDB Clients

One-way TLS means that only the server provides a private key and an X509 certificate. When TLS is used without a client certificate, it is called "one-way" TLS, because only the server can be authenticated, so certificate based authentication is only possible in one direction. However, encryption is still possible in both directions. Server certificate verification means that the client verifies that the certificate belongs to the server.

MariaDB starting with 11.4

Starting from MariaDB 11.4 (Connector/C version 3.4) this mode is enabled by default. Connector/C will enable TLS automatically on all non-local connections and will require a verified server certificate to prevent man-in-the-middle attacks.

To enable one-way TLS manually (for older clients or if you want verification with CA certificate) you can specify these options in a relevant client option group in an option file:

```
[client-mariadb]
ssl_ca = /etc/my.cnf.d/certificates/ca.pem
ssl-verify-server-cert
```

Or if you wanted to specify them on the command-line with the mariadb client, then you could execute something like this:

```
$ mariadb -u myuser -p -h myserver.mydomain.com \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --ssl-verify-server-cert
```

You can disable server certificate verification by configuring this:

```
[client-mariadb]
disable-ssl-verify-server-cert
```

(or a similar command line option), but it creates a risk of a man-in-the-middle attack where the connection can be eavesdropped or manipulated even if it is being encrypted.

Enabling Two-Way TLS for MariaDB Clients

Two-way TLS means that both the client and server provide a private key and an X509 certificate. It is called "two-way" TLS because both the client and server can be authenticated.

For example, to specify these options in a relevant client option group in an option file, you could set the following:

```
[client-mariadb]
ssl_cert = /etc/my.cnf.d/certificates/client-cert.pem
ssl_key = /etc/my.cnf.d/certificates/client-key.pem
ssl_ca = /etc/my.cnf.d/certificates/ca.pem
ssl-verify-server-cert
```

Or if you wanted to specify them on the command-line with the mariadb client, then you could execute something like this:

```
$ mariadb -u myuser -p -h myserver.mydomain.com \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --ssl-verify-server-cert
```

Two-way SSL is required for an account if the `REQUIRE X509`, `REQUIRE SUBJECT`, and/or `REQUIRE ISSUER` clauses are specified for the account.

Enabling TLS for MariaDB Connector/C Clients

See the documentation on MariaDB Connector/C's TLS Options for information on how to enable TLS for clients that use MariaDB Connector/C.

Enabling TLS for MariaDB Connector/ODBC Clients

See the documentation on MariaDB Connector/ODBC's TLS-Related Connection Parameters for information on how to enable TLS for clients that use MariaDB Connector/ODBC.

Enabling TLS for MariaDB Connector/J Clients

See the documentation on Using TLS/SSL with MariaDB Connector/J for information on how to enable TLS for clients that use MariaDB Connector/J.

Verifying that a Connection is Using TLS

You can verify that a connection is using TLS by checking the connection's Ssl_version status variable. It will either show the TLS version used, or be empty when no encryption is in effect. For example:

```
SHOW SESSION STATUS LIKE 'Ssl_version';
+---------------+-------------------------+
| Variable_name | Value                   |
+---------------+-------------------------+
| Ssl_version   | TLSv1.3                 |
+---------------+-------------------------+
1 row in set (0.00 sec)
```

Requiring TLS for Specific User Accounts

You can set certain TLS-related restrictions for specific user accounts. For instance, you might use this with user accounts that require access to sensitive data while sending it across networks that you do not control. These restrictions can be enabled for a user account with the CREATE USER, ALTER USER, or GRANT statements. For example:

- A user account must connect via TLS if the user account is defined with the `REQUIRE SSL` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE SSL;
```

- A user account must connect via TLS with a specific cipher if the user account is defined with the `REQUIRE CIPHER` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE CIPHER 'ECDH-RSA-AES256-SHA384';
```

- A user account must connect via TLS with a valid client certificate if the user account is defined with the `REQUIRE X509` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE X509;
```

- A user account must connect via TLS with a specific client certificate if the user account is defined with the `REQUIRE SUBJECT` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=Portland';
```

- A user account must connect via TLS with a client certificate that must be signed by a specific certificate authority if the user account is defined with the `REQUIRE ISSUER` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=Portland'
   AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter
Parker/emailAddress=p.parker@marvel.com';
```

Requiring TLS for Specific User Accounts from Specific Hosts

A user account can have different definitions depending on what host the user account is logging in from. Therefore, it is possible to have different TLS requirements for the same username for different hosts. For example:

```
CREATE USER 'alice'@'localhost'
   REQUIRE NONE;

CREATE USER 'alice'@'%'
   REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=Portland'
   AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter
Parker/emailAddress=p.parker@marvel.com'
   AND CIPHER 'ECDHE-ECDSA-AES256-SHA384';
```

In the above example, the `alice` user account does not require TLS when logging in from localhost. However, when the `alice` user account logs in from any other host, they must use TLS with the given cipher, and they must provide a valid client certificate with the given subject that must have been signed by the given issuer.

This page is licensed: CC BY-SA / Gnu FDL

Zero-Configuration SSL

> **Note:**
>
> This feature is available from MariaDB 11.4.

Zero-Configuration SSL (TLS)

Zero-Configuration SSL is a feature that makes encrypted connections the default standard without requiring manual certificate management. This allows for secure, MitM[27]-resistant connections "out of the box", by automating server certificate generation and client-side verification.

Overview

Traditionally, enabling SSL/TLS required manual steps: generating private keys, creating Certificate Signing Requests (CSRs), and managing a Certificate Authority (CA). Because of this complexity, many deployments remained unencrypted.

Zero-Configuration SSL eliminates these barriers:

1. Automatic Server Setup: The server automatically generates a self-signed certificate if no certificate is configured.
2. Zero-trust validation: Clients verify the server's identity using the existing account password as a shared secret, rather than relying on a trusted CA.

Requirements and Defaults

- MariaDB Version: Server and Client must be version 11.4 or higher.
- Default Behavior: SSL is enabled and verified by default. The client has `--ssl-verify-server-cert` enabled by default.
- Supported Authentication Plugins: This feature works with mysql_native_password, ed25519, and parsec (that are MitM-proof even without SSL).

Configuration Options

While designed to work automatically, the following options allow for manual control:

Option	Description
`--disable-ssl`	Disables SSL entirely (not recommended).
`--skip-ssl-verify-server-cert`	A verified certificate is not required, the client will accept unverified certificates or even a completely unencrypted connection. This is not MitM-safe, even if the server uses SSL.

Limitations

Authentication plugins such as PAM, GSSAPI, and `cached_sha2_plugin` are not MitM-proof without SSL, they require SSL to be secure and thus cannot be used to verify SSL itself. One should use traditional CA-based verification with these plugins.

Verification

To verify that Zero-Configuration SSL is being used, connect via TCP/IP (not via UNIX socket, because in that case the certificate is automatically trusted), and look at the status output:

```
mariadb -h 127.0.0.1 -u <username> -p
...
MariaDB [(none)]> status
--------------
mariadb from 12.2.2-MariaDB, client 15.2 for osx10.20 (arm64) using  EditLine wrapper
...
Current user:       stefan@localhost
SSL:            Cipher in use is TLS_AES_256_GCM_SHA384, cert is OK
...
Connection:     127.0.0.1 via TCP/IP
...
TCP port:       3306
...
```

Check if the server has automatically generated its own certificates by querying the session variables:

```
MariaDB [(none)]> SHOW SESSION STATUS LIKE 'Ssl_cipher'; SHOW SESSION STATUS LIKE 'Ssl_version';
+---------------+------------------------+
| Variable_name | Value                  |
+---------------+------------------------+
| Ssl_cipher    | TLS_AES_256_GCM_SHA384 |
+---------------+------------------------+
...
+---------------+---------+
| Variable_name | Value   |
+---------------+---------+
| Ssl_version   | TLSv1.3 |
+---------------+---------+
```

Run the following to see the server's certificate details:

```
$ openssl s_client -starttls mysql 127.0.0.1:3306
Connecting to 127.0.0.1
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN=MariaDB Server
verify error:num=18:self-signed certificate
verify return:1
depth=0 CN=MariaDB Server
verify return:1
---
Certificate chain
 0 s:CN=MariaDB Server
   i:CN=MariaDB Server
   a:PKEY: RSA, 4096 (bit); sigalg: sha256WithRSAEncryption
   v:NotBefore: Mar  2 16:19:15 2026 GMT; NotAfter: Feb 28 16:19:15 2036 GMT
...
```

How it Works

The core mechanism of Zero-Configuration SSL is the use of the user password as a shared secret to verify the server's identity without a Certificate Authority (CA).

1. Connection Establishment: The client connects and establishes a TLS connection using the server's self-signed certificate.
2. Fingerprint Calculation: The client calculates the SHA256 fingerprint of the server's certificate.
3. Signature Generation:
    - The client creates a signature based on the certificate fingerprint and the user's password.

- The server performs the same calculation.

4. Verification: By comparing these values, the client confirms it is communicating with the intended server. This prevents Man-in-the-Middle (MitM) attacks because an attacker would present a different certificate, resulting in a fingerprint mismatch that they cannot resolve without knowing the user's password.

This page is licensed: CC BY-SA / Gnu FDL

Replication with Secure Connections

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Overview

By default, MariaDB replicates data between primaries and replicas without encrypting it. This is generally acceptable when the primary and replica run are in networks where security is guaranteed through other means. However, in cases where the primary and replica exist on separate networks or they are in a high-risk network, the lack of encryption does introduce security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

To mitigate this concern, MariaDB allows you to encrypt replicated data in transit between primaries and replicas using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix `ssl_`, but internally, MariaDB only supports its secure successors.

In order to secure connections between the primary and replica, you need to ensure that both servers were compiled with TLS support. See Secure Connections Overview to determine how to check whether a server was compiled with TLS support.

You also need an X509 certificate, a private key, and the Certificate Authority (CA) chain to verify the X509 certificate for the primary. If you want to use two-way TLS, then you will also an X509 certificate, a private key, and the Certificate Authority (CA) chain to verify the X509 certificate for the replica. If you want to use self-signed certificates that are created with OpenSSL, then see Certificate Creation with OpenSSL for information on how to create those.

Securing Replication Traffic

In order to secure replication traffic, you will need to ensure that TLS is enabled on the primary. If you want to use two-way TLS, then you will also need to ensure that TLS is enabled on the replica. See Securing Connections for Client and Server for information on how to do that.

For example, to set the TLS system variables for each server, add them to a relevant server option group in an option file on each server:

```
[mariadb]
ssl_cert = /etc/my.cnf.d/certificates/server-cert.pem
ssl_key = /etc/my.cnf.d/certificates/server-key.pem
ssl_ca = /etc/my.cnf.d/certificates/ca.pem
```

And then restart the server to make the changes persistent.

At this point, you can reconfigure the replicas to use TLS to encrypt replicated data in transit. There are two methods available to do this:

- Executing the CHANGE MASTER statement to set the relevant TLS options.
- Setting TLS client options in an option file.

Enabling TLS on a Replica

TLS can be enabled on a replication replica by issuing the CHANGE MASTER statement, while setting a number of options. Which options you set depends on whether you want one-way TLS or two-way TLS, and whether or not you want to verify the server certificate.

Enabling Two-Way TLS

Two-way TLS means that both the client and server provide a private key and an X509 certificate. It is called "two-way" TLS because both the client and server can be authenticated. In this case, the "client" is the replica. To configure two-way TLS, you would need to set the following options:

- You need to set the path to the server's certificate by setting the MASTER_SSL_CERT option.
- You need to set the path to the server's private key by setting the MASTER_SSL_KEY option.
- You need to set the path to the certificate authority (CA) chain that can verify the server's certificate by setting either the MASTER_SSL_CA or the MASTER_SSL_CAPATH options.
- If you want server certificate verification, then you also need to set the MASTER_SSL_VERIFY_SERVER_CERT option (enabled by default from MariaDB 11.3).
- If you want to restrict the server to certain ciphers, then you also need to set the MASTER_SSL_CIPHER option.

If the replica threads are currently running, you first need to stop them by executing the STOP SLAVE statement. For example:

```
STOP SLAVE;
```

Then, execute the CHANGE MASTER statement to configure the replica to use TLS. For example:

```
CHANGE MASTER TO
   MASTER_SSL_CERT = '/path/to/client-cert.pem',
   MASTER_SSL_KEY = '/path/to/client-key.pem',
   MASTER_SSL_CA = '/path/to/ca/ca.pem',
   MASTER_SSL_VERIFY_SERVER_CERT=1;
```

At this point, you can start replication by executing the START SLAVE statement. For example:

```
START SLAVE;
```

The replica now uses TLS to encrypt data in transit as it replicates it from the primary.

Enabling One-Way TLS

Enabling One-Way TLS With Server Certificate Verification

One-way TLS means that only the server provides a private key and an X509 certificate. When TLS is used without a client certificate, it is called "one-way" TLS, because only the server can be authenticated, so authentication is only possible in one direction. However, encryption is still possible in both directions. Server certificate verification means that the client verifies that the certificate belongs to the server. In this case, the "client" is the replica. This mode is enabled by default starting from MariaDB 11.3. To configure one-way TLS in earlier versions, you would need to set the following options:

- You need to set the path to the certificate authority (CA) chain that can verify the server's certificate by setting either the MASTER_SSL_CA or the MASTER_SSL_CAPATH options.
- You need to set the MASTER_SSL_VERIFY_SERVER_CERT option.

- If you want to restrict the server to certain ciphers, then you also need to set the MASTER_SSL_CIPHER option.

If the replica threads are currently running, you first need to stop them by executing the STOP SLAVE statement. For example:

```
STOP SLAVE;
```

Then, execute the CHANGE MASTER statement to configure the replica to use TLS. For example:

```
CHANGE MASTER TO
   MASTER_SSL_CA = '/path/to/ca/ca.pem',
   MASTER_SSL_VERIFY_SERVER_CERT=1;
```

At this point, you can start replication by executing the START SLAVE statement. For example:

```
START SLAVE;
```

The replica now uses TLS to encrypt data in transit as it replicates it from the primary.

Enabling One-Way TLS Without Server Certificate Verification

One-way TLS means that only the server provides a private key and an X509 certificate. When TLS is used without a client certificate, it is called “one-way” TLS, because only the server can be authenticated, so authentication is only possible in one direction. However, encryption is still possible in both directions. In this case, the “client” is the replica. To configure two-way TLS without server certificate verification, you would need to set the following options:

- You need to configure the replica to use TLS by setting the MASTER_SSL option.
- If you want to restrict the server to certain ciphers, then you also need to set the MASTER_SSL_CIPHER option.
- Starting from MariaDB 11.3 you need to disable the MASTER_SSL_VERIFY_SERVER_CERT option.

If the replica threads are currently running, you first need to stop them by issuing the STOP REPLICA statement:

```
STOP REPLICA;
```

Then, execute the CHANGE MASTER statement to configure the replica to use TLS. For example:

```
CHANGE MASTER TO
   MASTER_SSL=1, MASTER_SSL_VERIFY_SERVER_CERT=0;
```

Now start replication by issuing the START REPLICA statement:

```
START REPLICA;
```

The replica now uses TLS to encrypt data in transit as it replicates from the primary.

Setting TLS Client Options in an Option File

In cases where you don’t mind restarting the server or you are setting the server up from scratch for the first time, you may find it more convenient to configure TLS options for replication through an option file. This is done the same way as it is for other clients. The specific options that you would need to set would depend on whether you want one-way TLS or two-way TLS, and whether you want to verify the server certificate. See Securing Connections for Client and Server: Enabling TLS for MariaDB Clients for more information.

To enable two-way TLS with server certificate verification, you can specify the following options in a relevant client option group in an option file:

```
[client-mariadb]
ssl_cert = /etc/my.cnf.d/certificates/client-cert.pem
ssl_key = /etc/my.cnf.d/certificates/client-key.pem
ssl_ca = /etc/my.cnf.d/certificates/ca.pem
ssl-verify-server-cert
```

Before restarting the server, you may also want to set the –skip-slave-start option in a server option group in an option file. This option prevents the replica threads from restarting automatically when the server starts. Instead, they will have to be restarted manually.

After these changes have been made, you can restart the server.

Once the server is back online, set the MASTER_SSL option by executing the CHANGE MASTER statement. This enables TLS. For example:

```
CHANGE MASTER TO MASTER_SSL=1;
```

The certificate and keys are read from the option file. At this point, you can start replication by issuing the START REPLICA statement:

```
START REPLICA;
```

This page is licensed: CC BY-SA / Gnu FDL

Enabling TLS on MariaDB Server

Overview

MariaDB Server supports data-in-transit encryption, which secures data transmitted over the network. The server and the clients encrypt data using the Transport Layer Security (TLS) protocol, which is a newer version of the Secure Socket Layer (SSL) protocol.

TLS must be manually enabled on the server, which is what this page describes.

Enabling TLS

> **Step 1**
>
> **Acquire an X509 certificate and a private key for the server.**
>
> If it is a test or development server, self-signed certificates and keys should be sufficient.

Step 2

Determine which system variables and you need to configure.

Mandatory system variables and options for TLS include:

System Variable/Option	Description
ssl_cert	X509 cert in PEM format
ssl_key	X509 key in PEM format
ssl_ca	JCA file in PEM format

Other useful system variables and options for TLS include:

System Variable/Option	Description
require_secure_transport	When this option is enabled, connections attempted using insecure transport are rejected. Secure transports are SSL/TLS, Unix sockets, and named pipes.
ssl_capath	CA directory
ssl_cipher	SSL cipher to use
ssl_crl	CRL file in PEM format
ssl_crlpath	CRL directory
tls_version	TLS protocol version for secure connections

Step 3

Choose a configuration file in which to configure your system variables and options.

It is not recommended to make changes to one of the bundled configuration files. Instead, we recommend to create a custom configuration file in one of the directories included by the main configuration file (for instance, my.cnf). Configuration files in included directories are read in alphabetical order. If you want your custom configuration file to override the bundled configuration files, it is a good idea to prefix the custom configuration file's name with a string that will be sorted last, such as `z-`. Examples:

- RHEL, CentOS, Rocky Linux, and SLES: `/etc/my.cnf.d/z-custom-my.cnf`
- Debian and Ubuntu: `/etc/mysql/mariadb.conf.d/z-custom-my.cnf`
- macOS: `/opt/homebrew/etc/z-custom-my.cnf`

Step 4

Set your system variables and options in the configuration file.

They need to be set in a group that is read by MariaDB Server, such as `[mariadb]`, `[server]`, or `[client-server]`:

```
[mariadb]
ssl_cert = /certs/server-cert.pem
ssl_key = /certs/server-key.pem
ssl_ca = /certs/ca-cert.pem
```

Step 5

Restart the server.

On most Linux systems, run this command:

```
sudo systemctl restart mariadb
```

On macOS, run this command:

```
brew services restart mariadb
```

Step 6

Connect to the server.

Start a client like mariadb (see this page for connection options):

```
$ sudo mariadb
```

Step 7

Verify that TLS is enabled.

```
SHOW GLOBAL VARIABLES LIKE 'have_ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_ssl      | YES   |
+---------------+-------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Certificate Creation with OpenSSL

Warning:

The instructions below generate version 1 certificates only. These work fine with servers and clients using OpenSSL, but fail if WolfSSL is used instead, as is the case for our Windows MSI packages and our binary tarballs for Linux.

WolfSSL requires version 3 certificates instead when using TLS v1.2 or higher, and so won't work with certificates generated as shown here when using two-way TLS with explicit client certificates.

Generating version 3 certificates requires a few more steps. See also: MDEV-25701

In order to secure communications with the MariaDB Server using TLS, you need to create a private key and an X509 certificate for the server. You may also want to create additional private keys and X509 certificates for any clients that need to connect to the server with TLS. This guide covers how to create a private key and a self-signed X509 certificate with OpenSSL.

Certificate Creation

The OpenSSL library provides a command-line tool called openssl, which can be used for performing various tasks with the library, such as generating private keys, creating X509 certificate requests, signing X509 certificates as a Certificate Authority (CA), and verifying X509 certificates.

Creating a Certificate Authority Private Key and Certificate

The Certificate Authority (CA) is typically an organization (such as Let's Encrypt) that signs the X509 certificate and validates ownership of the domain. However, when you would like to use self-signed certificates, you need to create the private key and certificate for the CA yourself, and then you can use them to sign your own X509 certificates.

To start, generate a private key for the CA using the openssl genrsa command. For example:

```
# openssl genrsa 2048 > ca-key.pem
```

After that, you can use the private key to generate the X509 certificate for the CA using the openssl req command. For example:

```
# openssl req -new -x509 -nodes -days 365000 \
      -key ca-key.pem -out ca.pem
```

The above commands create two files in the working directory: The `ca-key.pem` private key and the `ca.pem` X509 certificate are both are used by the CA to create self-signed X509 certificates below.

Creating a Private Key and a Self-Signed Certificate

Once you have the CA's private key and X509 certificate, you can create the self-signed X509 certificates to use for the MariaDB Server, client, replication and other purposes.

To start, generate a private key and create a certificate request using the openssl req command. For example:

```
# openssl req -newkey rsa:2048 -days 365000 \
      -nodes -keyout server-key.pem -out server-req.pem
```

After that, process the key to remove the passphrase using the openssl rsa command. For example:

```
# openssl rsa -in server-key.pem -out server-key.pem
```

Lastly, using the certificate request and the CA's private key and X509 certificate, you can generate a self-signed X509 certificate from the certificate request using the openssl x509 command. For example:

```
# openssl x509 -req -in server-req.pem -days 365000 \
      -CA ca.pem -CAkey ca-key.pem -set_serial 01 \
      -out server-cert.pem
```

This creates a `server-cert.pem` file, which is the self-signed X509 certificate.

Certificate Verification

Once you have created the CA's X509 certificate and a self-signed X509 certificate, you can verify that the X509 certificate was correctly generated using the openssl verify command. For example:

```
# openssl verify -CAfile ca.pem server-cert.pem
server-cert.pem: OK
```

You can add as many X509 certificates to check against the CA's X509 certificate as you want to verify. A value of `OK` indicates that you can use it was correctly generated and is ready for use with MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

SSL/TLS System Variables

The system variables listed on this page relate to encrypting data during transfer between servers and clients using the Transport Layer Security (TLS) protocol. Often, the term Secure Sockets Layer (SSL) is used interchangeably with TLS, although strictly speaking the SSL protocol is the predecessor of TLS and is no longer considered secure.

For compatibility reasons, the TLS system variables in MariaDB still use the `ssl_` prefix, but MariaDB only supports its more secure successors. For more information on SSL/TLS in MariaDB, see Secure Connections Overview.

Variables

`have_openssl`

- Description: This variable shows whether the server is linked with OpenSSL rather than MariaDB's bundled TLS library, which might be wolfSSL or yaSSL.
  - I this system variable shows `YES`, the server is linked with OpenSSL.
  - See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
- Scope: Global
- Dynamic: No

`have_ssl`

- Description: This variable shows whether the server supports using TLS to secure connections.
  - If the value is `YES`, then the server supports TLS, and TLS is enabled.
  - If the value is `DISABLED`, then the server supports TLS, but TLS is not enabled.
  - If the value is `NO`, then the server was not compiled with TLS support, so TLS cannot be enabled.
  - When TLS is supported, check the have_openssl system variable to determine whether the server is using OpenSSL or MariaDB's bundled TLS library. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
- Scope: Global
- Dynamic: No

`ssl_ca`

- Description: Defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This system variable requires that you use the absolute path, not a relative path. This system variable implies the ssl option.
  - See Secure Connections Overview: Certificate Authorities (CAs) for more information.
- Command line: `--ssl-ca=file_name`
- Scope: Global
- Dynamic: No
- Data Type: `file name`

`ssl_capath`

- Description: Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This system variable requires that you use the absolute path, not a relative path. The directory specified by this variable needs to be run through the openssl rehash command. This system variable implies the ssl option.
  - See Secure Connections Overview: Certificate Authorities (CAs) for more information.
- Command line: `--ssl-capath=directory_name`
- Scope: Global
- Dynamic: No
- Data Type: `directory name`

`ssl_cert`

- Description: Defines a path to the X509 certificate file to use for TLS. This system variable requires that you use the absolute path, not a relative path. This system variable implies the ssl option.

- Command line: `--ssl-cert=name`
- Scope: Global
- Dynamic: No
- Data Type: `file name`
- Default Value: None

`ssl_cipher`

- Description: List of permitted ciphers or cipher suites to use for TLS. Besides cipher names, if MariaDB was compiled with OpenSSL, this variable could be set to `SSLv3` or `TLSv1.2` to allow all SSLv3 or all `TLSv1.2` ciphers. `TLSv1.3` ciphers can be used, too – see Using TLSv1.3 for details. This system variable implies the ssl option.
- Command line: `--ssl-cipher=name`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: None

`ssl_crl`

- Description: Defines a path to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This system variable requires that you use the absolute path, not a relative path.
  - See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information.
  - This variable is only valid if the server was built with OpenSSL. If the server was built with wolfSSL or yaSSL, then this variable is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
- Command line: `--ssl-crl=name`
- Scope: Global
- Dynamic: No
- Data Type: `file name`
- Default Value: None

`ssl_crlpath`

- Description: Defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This system variable requires that you use the absolute path, not a relative path. The directory specified by this variable needs to be run through the openssl rehash command.
  - See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information.
  - This variable is only supported if the server was built with OpenSSL. If the server was built with wolfSSL or yaSSL, then this variable is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
- Command line: `--ssl-crlpath=name`
- Scope: Global
- Dynamic: No
- Data Type: `directory name`
- Default Value: None

`ssl_key`

- Description: Defines a path to a private key file to use for TLS. This system variable requires that you use the absolute path, not a relative path. This system variable implies the ssl option.
- Command line: `--ssl-key=name`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: None

`ssl_passphrase`

- Description: SSL certificate key passphrase. Works similarly to the `--passout/--passin` openssl command-line parameters. The pass phrase value can be formatted as follows:
  - `pass:` *`password`* — Provide the actual *password*.
  - `env:` *`var`* — Obtain the password from the environment variable *var*.
  - `file:` *`filename`* — Read the password from the specified file *filename*. Only the first line, up to the newline character, is read from the stream.

  If `ssl_passphrase` is set, SHOW VARIABLE shows one of `file:`, `env:`, or `pass:`, and doesn't reveal sensitive data.
- Command line: `--ssl-passphrase=val`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: None
- Introduced: MariaDB 12.0

`tls_version`

- Description: This system variable accepts a comma-separated list (with no whitespaces) of TLS protocol versions. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted.
  - See Secure Connections Overview: TLS Protocol Versions for more information.
- Command line: `--tls-version=value`
- Scope: Global
- Dynamic: No
- Data Type: `enumerated`
- Default Value:
  - TLSv1.1,TLSv1.2,TLSv1.3 (<= MariaDB 10.6.15, MariaDB 10.11.5, MariaDB 11.0.3)
  - TLSv1.2,TLSv1.3 (>= MariaDB 10.6.16, MariaDB 10.11.6, MariaDB 11.0.4 and later versions)
- Valid Values: TLSv1.0,TLSv1.1,TLSv1.2,TLSv1.3
- Introduced: MariaDB 10.4.6

`version_ssl_library`

- Description: The version of the TLS library that is being used. Note that the version returned by this system variable does not always necessarily correspond to the exact version of the OpenSSL package installed on the

system. OpenSSL shared libraries tend to contain interfaces for multiple versions at once to allow for backward compatibility. Therefore, if the OpenSSL package installed on the system is newer than the OpenSSL version that the MariaDB server binary was built with, then the MariaDB server binary might use one of the interfaces for an older version.
    - See TLS and Cryptography Libraries Used by MariaDB: Checking the Server's OpenSSL Version for more information.

- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: None

See Also

- Secure Connections Overview
- System Variables for a complete list of system variables and instructions on setting them.
- Full list of MariaDB options, system and status variables

This page is licensed: CC BY-SA / Gnu FDL

Using TLSv1.3

OpenSSL 1.1.1 introduced support for TLSv1.3. TLSv1.3 is a major rewrite of the TLS protocol. (Some even argued it should've been called TLSv2.0.) Among other things,

- it introduces a new set of cipher suites that only work with TLSv1.3,
- and TLSv1.3 does not support cipher suites from previous TLS protocol versions.

> **Note:**
>
> OpenSSL 1.1.1 is primarily (if not exclusively) supported by Red Hat 8.

This incompatible change had a non-obvious consequence. If you specified particular cipher suites to disable old and obsolete TLS protocol version, you might have inadvertently prevented TLSv1.3 from working, if the TLSv1.3 cipher suites were not added to their cipher list. After upgrading to OpenSSL 1.1.1, this might give the impression that you are using TLSv1.3, when their existing cipher suite configuration might prevent it.

To avoid this problem, OpenSSL developers decided that TLSv1.3 cipher suites should not be affected by the normal cipher-selecting API. This means that the ssl_cipher system variable had no effect on the TLSv1.3 cipher suites. This has been fixed in MariaDB, though – you can use the `ssl_cipher` variable just like before that change.

> **Warning:**
>
> You cannot specify both TLSv1.2 and TLSv1.3 at the same time when MariaDB is built against OpenSSL 1.1.1.

See this OpenSSL blog post and GitHub issue for more information.

See Also

- Secure Connections Overview

This page is licensed: CC BY-SA / Gnu FDL

Post-Quantum Cryptography in MariaDB TLS

Overview

Quantum computers use quantum mechanics to solve mathematical problems much faster than traditional computers. Their capacity to crack the public-key cryptographic techniques that currently support TLS handshakes, such as RSA and Elliptic Curve Cryptography (ECC), is especially concerning.

This poses a threat known as **harvest now, decrypt later**: an attacker can intercept and store encrypted database communication now, then decrypt it later when sufficiently robust quantum computers are made accessible. Thus, even current threats pose a risk to long-lived sensitive data, such as financial records, intellectual property, and personally identifiable information (PII).

The **National Institute of Standards and Technology** (NIST) in the United States has standardized two post-quantum cryptography algorithms to address this:

- **ML-KEM** (Module Lattice Key Encapsulation Mechanism) (FIPS 203): A key encapsulation technique for quantum-resistant key exchange during TLS handshakes.
- **ML-DSA** (Module Lattice Digital Signature Algorithm) (FIPS 204): A digital signature scheme for quantum-resistant server authentication.
- **Hybrid mode**: Use of both classical and post-quantum algorithms together during the transition period. The session key is secure if either algorithm remains unbroken.

MariaDB automatically negotiates ML-KEM for TLS 1.3 key exchange when linked against **OpenSSL 3.5 or later**, requiring no further configuration. Because of this, MariaDB connections are inherently quantum-resistant in transit.

Requirements

Component	Version Requirement
MariaDB Server	10.8.1 (source), backported to 10.5.17 (source) as well as to 10.6.9 and 10.7.5 (source). (regarding the sources: MariaDB support for OpenSSL 3.0 guarantees support for 3.5.0 and later too due to API and ABI compatibility)
OpenSSL	3.5.0 or later

How It Works

MariaDB assigns the system OpenSSL library to handle TLS operations. As a TLS 1.3 key exchange group, OpenSSL 3.5 included native support for ML-KEM. A MariaDB server will automatically publish ML-KEM key exchange groups during the TLS 1.3 handshake if OpenSSL 3.5 or later is installed.

There is no need to modify any server system variables. During this time of transition, **hybrid mode** is used. This indicates that ML-KEM and a traditional method like classical Elliptic Curve (X25519) are used for key exchange. This hybrid classical and post-quantum approach is reflected in the negotiated group that can be seen in the handshake output: `X25519MLKEM768`.

This ensures backward compatibility. Clients that do not support ML-KEM continue to use traditional key exchange to establish connections.

Verifying Quantum-Resistant TLS

By connecting to the OpenSSL client and examining the negotiated TLS group, you can verify that a MariaDB connection uses ML-KEM.

```
$ openssl s_client -starttls mysql localhost:3306
```

Output

Look for the Negotiated TLS1.3 group in the output:

```
Peer signing digest: SHA256
Peer signature type: rsa_pss_rsae_sha256
Negotiated TLS1.3 group: X25519MLKEM768
```

A value of `X25519MLKEM768` indicates that the hybrid classical and ML-KEM key exchange is operational. During transit, the connection is quantum-resistant.

Compatibility

MariaDB automatically returns to traditional TLS 1.3 key exchange when a connecting client does not support post-quantum key exchange groups. Current clients and applications do not require any setup adjustments. Connections that are limited to classical methods remain unchanged.

Data-in-Transit and Data-at-Rest

Post quantum cryptography secures data while it is transmitted across a network. Data at rest should also be protected with quantum-resistant encryption to ensure comprehensive quantum resistance. For this reason, AES-256 is advised since it offers enough key length to survive known quantum attacks. See Data-at-Rest-Encryption for information on how to enable encryption for table data and tablespaces in MariaDB.

Performance Considerations

- **Handshake size**: Post-quantum algorithms may produce slightly larger handshakes.
- **Performance impact**: Session reuse reduces overhead.
- **Workload suitability**: Appropriate for typical database workloads.

See Also

- TLS and Cryptography Libraries used by MariaDB
- Data-at-Rest-Encryption
- Securing Connections for Client and Server
- NIST FIPS 203 (ML-KEM)
- NIST FIPS 204 (ML-DSA)

This page is licensed: CC BY-SA / Gnu FDL

Data-at-Rest Encryption

See Data-at-Rest Encryption (TDE) Fundamentals.

The fundamentals of MariaDB data-at-rest encryption (Transparent Data Encryption, TDE), which encrypts files on storage.

See Managing Binary Log Encryption.

How to encrypt binary logs and relay logs so on-disk data modifications are only accessible through the server.

See Aria Encryption.

Learn about Aria encryption in MariaDB Server for data at rest. This section details how to encrypt Aria tablespaces, providing enhanced security for your stored data.

See InnoDB Encryption.

Learn about InnoDB encryption for data at rest. This section details how to encrypt InnoDB tablespaces, ensuring strong data security and compliance for your mission-critical applications.

See Key Management and Encryption Plugins.

Explore key management and encryption plugins for MariaDB Server. This section details how to manage encryption keys and leverage plugins for robust data-at-rest protection.

This page is licensed: CC BY-SA / Gnu FDL

Data-at-Rest Encryption (TDE) Fundamentals

MariaDB Data-at-Rest Encryption, also known as Transparent Data Encryption (TDE), protects your data by encrypting files directly on the storage media. This ensures that if physical disks or backup files are stolen, the data remains unreadable without the corresponding encryption keys.

Security Context

When Encryption Protects Your Data

- Physical Theft: Unauthorized access to hard drives or storage arrays.
- Backup Exposure: If database backups are intercepted or accessed by unauthorized parties.
- Service Provider Access: Preventing infrastructure administrators in cloud or managed environments from viewing sensitive data.

When Encryption is No Help

- Authorized Access: Users with valid SQL credentials will still see decrypted data; TDE does not replace a robust User Account Management strategy.
- Application Vulnerabilities: TDE does not prevent SQL injection or application-level data breaches.
- Data-in-Transit: This feature only secures stored data. Use TLS/SSL to secure data moving across the network.

Architecture and Lifecycle

MariaDB performs encryption at the I/O layer. This process is “transparent” because applications and queries do not need to be modified to interact with encrypted data.

Once a Key Management plugin is configured, encryption occurs automatically whenever MariaDB writes pages to disk, and decryption occurs when data is read back into memory.

- Write Path: When MariaDB flushes data from the buffer pool to the disk, the data is encrypted.
- Read Path: When MariaDB reads data from the disk into memory, it is decrypted.
- Memory: Data stored in RAM (such as the buffer pool) remains in a decrypted state while in use.

Storage Engine Support

MariaDB provides flexible control over what is encrypted, though support and requirements vary by component:

- InnoDB: Fully supported. You can encrypt all tablespaces, individual tables, and the InnoDB redo log. For details, see InnoDB Encryption.
- Aria: Supported only for tables created with `ROW_FORMAT=PAGE` (the default). For details, see Aria Encryption.
- Binary Logs: MariaDB can encrypt binary logs and relay logs to protect the replication pipeline. See Managing Binary Log Encryption.
- Temporary Files: Internal on-disk temporary files can be encrypted by setting `encrypt_tmp_files=ON`.

> **Note:**
>
> **Enterprise Server** provides encryption for **Galera Cluster** via GCache.
> How GCache encryption is enabled and disabled is detailed on this page.

Limitations

The following elements are not encrypted by the MariaDB server:

- Metadata: Information in `.frm` files and the system data dictionary.
- Specific Logs: The MariaDB Error Log, General Query Log, and Slow Query Log.
- Aria Control Log: While Aria tables can be encrypted, the Aria storage engine log is not currently encrypted.
- Utilities: Tools like `mariadb-binlog` require the `--read-from-remote-server` flag to read encrypted content.

Key Management

Encryption requires a key management and encryption plugin. These plugins are responsible for managing the 32-bit integer key identifiers and performing the cryptographic operations. You must install and configure a Key Management Plugin before enabling encryption options for storage engines.

> **Note:**
>
> For the rest of this page, we assume you're using the File Key Management Encryption Plugin. If you're using a different one, adjust the configuration and SQL statements accordingly.

Enabling Data-at-Rest Encryption

Enabling data-at-rest encryption encompasses these steps:

1. Creating a key file.
2. Installing and enabling a key management plugin.
3. Enabling encryption for Aria tables, InnoDB tables, or both.
4. Verifying encryption is turned on.

> **Note:**
>
> The path used in these instructions (`/etc/mysql/encryption/`) is common for most Linux systems. Adjust if your operating system uses a different path.

> **Note:**
>
> MariaDB server must be restarted several times. On most Linux systems, the command for that is `sudo systemctl restart mariadb` . Consult your operating system documentation to find out what your system uses.
>
> If you're getting errors, particularly if MariaDB fails to start, for example due to a configuration issue, inspect the error log. On many modern Linux versions, this can be done using `journalctl` . Example:
>
> ```
> sudo journalctl -u mariadb -n 1000 --no-pager | grep ERROR
> ```

Step 1

Create the key file.

In this step, we create a key file that fulfills basic requirements. It contains only one unencrypted key, which is good enough to perform the rest of the steps. It is highly recommended, though, to use multiple keys, and to encrypt the key files. See Creating the Key File.

CURRENT ENTERPRISE SERVER

Run these commands to create an `encryption` folder, and a 32 byte (256 bit) long key file within that folder.

```
mkdir -p /etc/mysql/encryption
echo $(echo -n "1;1;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile.txt
```

ENTERPRISE SERVER < 11.8 & COMMUNITY SERVER

Run these commands to create an `encryption` folder, and a 32 byte (256 bit) long key file within that folder.

```
mkdir -p /etc/mysql/encryption
echo $(echo -n "1;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile.txt
```

Step 2

Install and enable a key management plugin.

Add the following to your configuration file (for instance, `my.cnf`), then restart the server so the changes take effect:

```
[mariadb]
plugin_load_add = file_key_management
file_key_management_filename = /etc/mysql/encryption/keyfile.txt
# The following line is optional but highly recommended.
# Uncomment it to enable usage of an encrypted key file.
# file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
file_key_management_encryption_algorithm = AES_CTR
```

For details about file name, file key, and encryption algorithm, see File Key Management Encryption Plugin.

Step 3

Enable encryption.

You can encrypt a number of database objects by setting the respective variables:

- InnoDB user tables – innodb_encrypt_tables
- InnoDB redo log – innodb_encrypt_log
- InnoDB temporary files – innodb_encrypt_temporary_tables
  MariaDB creates temporary files on disk, for instance, files used for file sorts. It is recommended to encrypt those, too.
- Aria user tables – aria_encrypt_tables
- Aria temporary tables – encrypt_tmp_disk_tables

To configure global encryption for all of those objects, add this to your configuration file (for instance, `my.cnf`), then restart the server:

```
[mariadb]
innodb_encrypt_tables = ON
innodb_encrypt_log = ON
innodb_encrypt_temporary_tables = ON
aria_encrypt_tables = ON
encrypt_tmp_disk_tables = ON
```

Alternatively, set it by running the following statements. Remember, though, that the settings do not persist across server restarts. Also note that some of the variables cannot be set at runtime – they have to be set in a configuration file:

```
SET GLOBAL innodb_encrypt_tables = ON;   -- for InnoDB tables
SET GLOBAL aria_encrypt_tables = ON;     -- for Aria tables
SET GLOBAL encrypt_tmp_disk_tables = ON; -- for Aria temporary tables
```

> **Note:**
>
> Particularly InnoDB has more encryption options. You can fine-tune the encryption, for instance, configure encryption threads. Those details are covered on this page.

Step 4

Verify encryption is turned on.

Make sure that the global encryption you configured is turned on, by issuing this statement:

```
SELECT
 'InnoDB Tables' AS Component, @@innodb_encrypt_tables AS Active
 UNION SELECT 'InnoDB Logs', @@innodb_encrypt_log
 UNION SELECT 'InnoDB Temp', @@innodb_encrypt_temporary_tables
 UNION SELECT 'Aria System', @@aria_encrypt_tables
 UNION SELECT 'Aria Temp', @@encrypt_tmp_disk_tables;
```

If encryption of all database objects is successfully enabled, this is the output of the query:

```
+---------------+--------+
| Component     | Active |
+---------------+--------+
| InnoDB Tables | ON     |
| InnoDB Logs   | 1      |
| InnoDB Temp   | 1      |
| Aria System   | 1      |
| Aria Temp     | 1      |
+---------------+--------+
```

Disabling Data-at-Rest Encryption

If you determine that encryption is no longer necessary, you can revert the system to an unencrypted state.

Prerequisites

- Encryption Status: MariaDB Enterprise Server must currently have data-at-rest encryption enabled and active.
- Key Management Access: You must have the original key management plugin active and the correct keys loaded to facilitate the decryption of the data.
- Sufficient Disk Space: Ensure adequate free space is available to accommodate the rewritten, unencrypted data files.

Step 1

Decrypt tables.

Disable global encryption at the storage engine level by issuing these statements. Note that some variables cannot be turned off at runtime – they have to be removed from the server configuration (for instance, `my.cnf`), and require a server restart:

```
SET GLOBAL innodb_encrypt_tables = OFF;   -- InnoDB tables
SET GLOBAL aria_encrypt_tables = OFF;     -- Aria tables
SET GLOBAL encrypt_tmp_disk_tables = OFF; -- Aria temporary tables
```

> **Note:**
>
> Any per-table encryption for InnoDB tables explicitly created with `ENCRYPTED=YES` must be manually decrypted using `ALTER TABLE table_name ENCRYPTED=NO`.

Step 2

Disable log and temp encryption.

Update your configuration file (`my.cnf`) to stop encrypting Aria and InnoDB tables, as well as new logs and temporary files:

```
[mariadb]
innodb_encrypt_tables = OFF
innodb_encrypt_log = OFF
innodb_encrypt_temporary_tables = OFF
aria_encrypt_tables = OFF
encrypt_tmp_disk_tables = OFF
```

Step 3

Monitor decryption progress.

You must wait for the background threads to finish decrypting data pages before removing the keys. Monitor the status for InnoDB tables via the Information Schema (there's no built-in way to monitor the status for Aria tables):

```
SELECT NAME, ENCRYPTION_SCHEME, ROTATING_OR_FLUSHING
FROM INFORMATION_SCHEMA.INNODB_TABLESPACES_ENCRYPTION
WHERE ENCRYPTION_SCHEME != 0;
```

> **Warning:**
>
> A restore from an encrypted backup isn't possible after removing the keys.

Step 4

Remove the key management plugin.

Only after all tables and logs are confirmed as unencrypted should you uninstall the encryption plugin and remove its configuration from your `my.cnf` file. For instance, if you're using the File Key Management Encryption Plugin, uninstall it:

```
UNINSTALL SONAME 'file_key_management';
```

Remove its configuration, then restart the server:

```
[mariadb]
# Remove or comment out the following line
plugin_load_add = file_key_management
```

Step 5

Verify encryption is turned off.

Use the same SQL statement shown above to verify encryption is turned off.

TDE in Action: Real-World Behavior

Transparent Data Encryption (TDE) is designed to protect data "at rest" while remaining completely invisible to standard SQL operations. With encryption turned on, MariaDB applies encryption to all new tables automatically.

Automatic Encryption (InnoDB & Aria)

For the subsequent instructions, we use these tables. With encryption turned on, note that they're encrypted without using any special syntax:

```
-- Create an InnoDB table
CREATE TABLE sensitive_accounts_innodb (
    id INT PRIMARY KEY,
    account_name VARCHAR(100)
) ENGINE=InnoDB;

-- Create an Aria table
-- Note that the ROW_FORMAT must be PAGE; otherwise, encryption isn't possible
CREATE TABLE sensitive_accounts_aria (
    id INT PRIMARY KEY,
    account_name VARCHAR(100)
) ENGINE=Aria ROW_FORMAT=PAGE;
```

Verifying Encryption Status

Because InnoDB and Aria handle data differently, you use two different methods to verify their encryption status.

For InnoDB

Check the `information_schema` to see the tablespace encryption scheme. If it's set to `1`, the table is encrypted.

```
SELECT NAME, ENCRYPTION_SCHEME
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME LIKE '%sensitive_accounts_innodb%';
```

```
+--------------------------------+-------------------+
| NAME                           | ENCRYPTION_SCHEME |
+--------------------------------+-------------------+
| test/sensitive_accounts_innodb |                 1 |
+--------------------------------+-------------------+
```

For Aria

There's no built-in way to check the encryption status of Aria tables, but you can check the contents of their `.MAD` files (the Aria data files) with commands like `cat`:

```
sudo cat /var/lib/mysql/test/sensitive_accounts_aria.MAD | more
```

If you see unintelligible output rather than something resembling data, the Aria table is encrypted.

Transparent Data Access

Whether the table is InnoDB or Aria, the encryption is **transparent**. Standard SQL statements work without modification, as the engine decrypts the data in memory (the Buffer Pool for InnoDB or Page Cache for Aria).

```
-- Standard inserts work for both
INSERT INTO sensitive_accounts_innodb VALUES (1, 'Customer InnoDB');
INSERT INTO sensitive_accounts_aria VALUES (1, 'Customer Aria');

-- Standard selects return plain text
SELECT * FROM sensitive_accounts_innodb;
SELECT * FROM sensitive_accounts_aria;
```

Manual Control: Disabling Encryption

If you need to exempt a specific table from the global encryption policy, use the `ENCRYPTED` attribute.

> **Note:**
>
> This is only possible for InnoDB tables.

Decrypting the Tables

Decrypting tables is done with an `ALTER TABLE` statement. For InnoDB tables, you can do this while global InnoDB table encryption is turned on.

> **Note:**
>
> For Aria tables, you can only do this when global Aria table encryption is turned off.

```
-- Disable encryption for the InnoDB table
ALTER TABLE sensitive_accounts_innodb ENCRYPTED=NO;
SELECT NAME, ENCRYPTION_SCHEME  FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
       WHERE NAME LIKE '%sensitive_accounts_innodb%';
+--------------------------------+-------------------+
| NAME                           | ENCRYPTION_SCHEME |
+--------------------------------+-------------------+
| test/sensitive_accounts_innodb |                 0 |
+--------------------------------+-------------------+

-- Disable encryption for the Aria table
-- This is only possible after disabling global Aria encryption
ALTER TABLE sensitive_accounts_aria ENGINE=Aria, ALGORITHM=COPY;
```

Reencrypting the Tables

```
ALTER TABLE sensitive_accounts_innodb ENCRYPTED=YES;
SELECT NAME, ENCRYPTION_SCHEME  FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
       WHERE NAME LIKE '%sensitive_accounts_innodb%';
+--------------------------------+-------------------+
| NAME                           | ENCRYPTION_SCHEME |
+--------------------------------+-------------------+
| test/sensitive_accounts_innodb |                 1 |
+--------------------------------+-------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Managing Binary Log Encryption

MariaDB can encrypt binary logs and relay logs to ensure that data modifications stored on disk are only accessible through the database server.

Prerequisites

Binary log and relay log encryption requires a configured Key Management and Encryption Plugin (such as File Key Management or AWS KMS). MariaDB specifically uses encryption key ID 1 for binary logs.

Key management plugin encryption looks like this (depending on the plugin used, and on the operating system the plugin is installed on):

```
[mariadb]
# File Key Management
plugin_load_add = file_key_management
file_key_management_filename = /etc/mysql/encryption/keyfile.enc
file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
file_key_management_encryption_algorithm = AES_CTR
```

Enabling Encryption

Binary log encryption is controlled by the `encrypt_binlog` system variable.

> **Step 1**
>
> **Stop the MariaDB server.**

> **Step 2**
>
> **Edit the configuration file.**
>
> Add the following to your configuration file (e.g., `my.cnf`):
>
> ```
> [mariadb]
> encrypt_binlog = ON
> ```

> **Step 3**
>
> **Start the server.**

From this point forward, all new binary logs are encrypted. To remove old, unencrypted logs, use `RESET MASTER` or `PURGE BINARY LOGS`.

Key Rotation

If your key management plugin supports rotation (check here), the server can use new key versions for new binary logs. However, unlike InnoDB, binary logs do not have a background re-encryption mechanism. Existing binary log files remain encrypted with the key version used at the time of their creation.

Reading Encrypted Binary Logs

Because encrypted logs cannot be read directly by standard text editors, you must use the `mariadb-binlog` utility.

Method 1: Remote Decryption (Recommended)

The most secure method is to have the server stream the decrypted events. This avoids the need to distribute encryption keys to administrative workstations.

```
mariadb-binlog --read-from-remote-server --user=root -p binlog.000012 > decrypted.sql
```

Method 2: Local Decryption

You can decrypt logs locally if the `mariadb-binlog` utility has access to the same key management plugin and key material used by the server.

```
mariadb-binlog /var/lib/mysql/binlog.000012 > decrypted.sql
```

Disabling Encryption

Disabling encryption stops the server from encrypting future logs. It does not decrypt existing files.

> **Step 1**
>
> **Stop the MariaDB server.**

> **Step 2**
>
> **Edit the configuration file.**
>
> Add the following to your configuration file (e.g., `my.cnf`):
>
> ```
> [mariadb]
> encrypt_binlog = OFF
> ```

> **Step 3**
>
> **Start the server.**

> **Warning:**
>
> Important: Keep your key management plugin loaded as long as you have encrypted binary logs on disk that you may need to read. If the plugin is removed, the server and `mariadb-binlog` will be unable to decrypt those older files.

Understanding Binlog Encryption

Behind the Scenes

When starting with binary log encryption, MariaDB Server logs a `Format_descriptor_log_event` and a `START_ENCRYPTION_EVENT`, then encrypts all subsequent events for the binary log.

Each event header and footer are created and processed to produce encrypted blocks. These encrypted blocks are produced before transactions are committed and before the events are flushed to the binary log. As such, they exist in an encrypted state in memory buffers and in the `IO_CACHE` files for user connections.

Effects of Data-at-Rest Encryption on Replication

When using encrypted binary logs with replication, you can have different encryption keys on the master and the replica. The master decrypts encrypted binary log events as it reads them from disk, and before its binary log dump thread sends them to the replica, so the replica actually receives the unencrypted binary log events.

To ensure that binary log events are encrypted as they are transmitted between the master and replica, use TLS with the replication connection.

Effects of Data-at-Rest Encryption on mariadb-binlog

mariadb-binlog does not have the ability to decrypt encrypted binary logs on its own (see MDEV-8813). In order to use `mariadb-binlog` with encrypted binary logs, use the –read-from-remote-server command-line option, so that the server can decrypt the binary logs for `mariadb-binlog`.

This page is licensed: CC BY-SA / Gnu FDL

Aria Encryption

See Aria: Encryption Overview.

Introduction to encrypting Aria tables, covering the necessary system variables (aria_encrypt_tables, encrypt_tmp_disk_tables) and how to verify encryption status by inspecting data files.

See Aria: Enabling Encryption.

Step-by-step guide to enabling encryption for user-created and internal temporary Aria tables, including the requirement to manually rebuild existing tables using ALTER TABLE.

See Aria: Encryption Keys.

Details how Aria manages encryption keys (using ID 1 for user tables and ID 2 for temporary tables) and notes limitations regarding key rotation and per-table key assignment.

See Aria: Disabling Encryption.

Instructions for safely disabling encryption on Aria tables, emphasizing the need to rebuild tables to an unencrypted state before removing key management plugins.

This page is licensed: CC BY-SA / Gnu FDL

Aria: Encryption Overview

MariaDB can encrypt data in tables that use the Aria storage engine. This includes both user-created tables and internal on-disk temporary tables that use the Aria storage engine. This ensures that your Aria data is only accessible through MariaDB.

For encryption with the InnoDB and XtraDB storage engines, see Encrypting Data for InnoDB/XtraDB.

Basic Configuration

In order to enable encryption for tables using the Aria storage engine, there are a couple server system variables that you need to set and configure. Most users will want to set aria_encrypt_tables and encrypt_tmp_disk_tables.

Users of data-at-rest encryption will also need to have a key management and encryption plugin configured. Some examples are File Key Management Plugin and AWS Key Management Plugin.

```
[mariadb]
# File Key Management
plugin_load_add = file_key_management
file_key_management_filename = /etc/mysql/encryption/keyfile.enc
file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
file_key_management_encryption_algorithm = AES_CTR

# Aria Encryption
aria_encrypt_tables=ON
encrypt_tmp_disk_tables=ON
```

Determining Whether a Table is Encrypted

The InnoDB storage engine has the information_schema.INNODB_TABLESPACES_ENCRYPTION table that can be used to get information about which tables are encrypted. Aria does not currently have anything like that (see MDEV-17324 about that).

To determine whether an Aria table is encrypted, you currently have to search the data file for some plain text that you know is in the data.

For example, let's say that we have the following table:

```
SELECT * FROM db1.aria_tab LIMIT 1;
+----+------+
| id | str  |
+----+------+
|  1 | str1 |
+----+------+
1 row IN SET (0.00 sec
```

Then, we could search the data file that belongs to `db1.aria_tab` for `str1` using a command-line tool, such as strings:

```
$ sudo strings /var/lib/mysql/db1/aria_tab.MAD | grep "str1"
str1
```

If you can find the plain text of the string, then you know that the table is not encrypted.

Encryption and the Aria Log

Only Aria tables are currently encrypted. The Aria log is not yet encrypted. See MDEV-8587.

This page is licensed: CC BY-SA / Gnu FDL

Aria: Enabling Encryption

To enable data-at-rest encryption for tables using the Aria storage engine, configure the server to use an Encryption Key Management plugin. Once this is done, you can enable encryption by setting the relevant system variables.

Encrypting User-Created Tables

For user-created tables, enable encryption by setting the aria_encrypt_tables system variable to `ON`, then restart the server:

```
[mariadb]
aria_encrypt_tables = ON
```

Alternatively, set the variable with an SQL statement. This doesn't require a server restart, but the setting is lost on server restart:

```
SET GLOBAL aria_encrypt_tables=ON
```

Once this is set, Aria enables encryption on all newly created tables.

> **Note:**
>
> **Encryption only works if the ROW_FORMAT table option set to `PAGE`.**
>
> Aria does not support encryption of tables where the `ROW_FORMAT` table option is set to `FIXED` or `DYNAMIC`.

> **Note:**
>
> Aria does not support the ENCRYPTED table option (see MDEV-18049 about that).

> **Note:**
>
> Encryption for Aria can only be enabled globally using the aria_encrypt_tables system variable.

Encrypting Existing Tables

In cases where you have existing Aria tables that you would like to encrypt, the process is a little more complicated. Unlike InnoDB, Aria does not utilize background encryption threads to automatically perform encryption changes (see MDEV-18971 about that). Therefore, to encrypt existing tables, you need to identify each table that needs to be encrypted, and then you need to manually rebuild each table.

First, set the `aria_encrypt_tables` system variable to encrypt new tables.

```
SET GLOBAL aria_encrypt_tables=ON
```

Identify Aria tables that have the `ROW_FORMAT` table option set to `PAGE`.

```
SELECT TABLE_SCHEMA, TABLE_NAME
FROM information_schema.TABLES
WHERE ENGINE='Aria'
  AND ROW_FORMAT='PAGE'
  AND TABLE_SCHEMA != 'information_schema';
```

For each table in the result set, issue an `ALTER TABLE` statement to rebuild the table.

```
ALTER TABLE aria_table ENGINE=Aria ROW_FORMAT=PAGE;
```

This statement causes Aria to rebuild the table using the `ROW_FORMAT` table option. Since you enabled encryption, Aria also encrypts the table in the process.

Encrypting Internal Temporary Tables on Disk

During the execution of queries, MariaDB routinely creates internal temporary tables. These internal temporary tables initially use the MEMORY storage engine, which is entirely stored in memory. When the table size exceeds the allocation defined by the max_heap_table_size system variable, MariaDB writes the data to disk using another storage engine. If you have the aria_used_for_temp_tables set to `ON`, MariaDB uses Aria in writing the internal temporary tables to disk.

Encryption for internal temporary tables is handled separately from encryption for user-created tables. To enable encryption for these tables, set the encrypt_tmp_disk_tables system variable to `ON`. Once set, all internal temporary tables that are written to disk using Aria are automatically encrypted.

Manually Encrypting Tables

Currently, Aria does not support manually encrypting tables through the ENCRYPTED and ENCRYPTION_KEY_ID table options. For more information, see MDEV-18049.

In cases where you want to encrypt tables manually or set the specific encryption key, use InnoDB.

This page is licensed: CC BY-SA / Gnu FDL

Aria: Encryption Keys

As with other storage engines that support data-at-rest encryption, Aria relies on an Encryption Key Management plugin to handle its encryption keys. Where the support is available, Aria can use multiple keys.

Encryption Keys

MariaDB keeps track of each encryption key internally using a 32-bit integer, which serves as the key identifier. Unlike InnoDB, Aria does not support the ENCRYPTION_KEY_ID table option (for more information, see MDEV-18049), which allows the user to specify the encryption key to use. Instead, Aria defaults to specific encryption keys provided by the Encryption Key Management plugin.

- When working with user-created tables, Aria encrypts them to disk using the ID 1 key.
- When working with internal temporary tables written to disk, Aria encrypts them to disk using the ID 2 key, unless there is no ID 2 key, then it falls back on the ID 1 key.

Key Rotation

Some key management and encryption plugins allow you to automatically rotate and version your encryption keys. If a plugin support key rotation, and if it rotates the encryption keys, then InnoDB's background encryption threads can re-encrypt InnoDB pages that use the old key version with the new key version. However, Aria does **not** have a similar mechanism, which means that the tables remain encrypted with the older key version. For more information, see MDEV-18971.

In order for key rotation to work, both the backend key management service (KMS) and the corresponding key management and encryption plugin have to support key rotation. See Encryption Key Management: Support for Key Rotation in Encryption Plugins to determine which plugins currently support key rotation.

This page is licensed: CC BY-SA / Gnu FDL

Aria: Disabling Encryption

Overview

This guide provides instructions for safely disabling encryption for the Aria storage engine, including user-created tables and internal on-disk temporary tables.

To fully disable encryption, you must set the relevant system variables and then rebuild each table to an unencrypted state.

> **Warning:**
>
> **Important Safety Precautions**
>
> - Do not remove your Encryption Key Management plugin yet. MariaDB must be able to read the existing encrypted data to rewrite it in an unencrypted state.
> - Maintain configurations. Keep your key management settings in place until the very last step of the process.

Step 1

Disable encryption for new data.

First, prevent MariaDB from encrypting any new data by updating the global system variables.

User-Created Tables

Set `aria_encrypt_tables` to `OFF`. This ensures that any new Aria tables created or existing tables rebuilt from this point forward will be unencrypted.

```
SET GLOBAL aria_encrypt_tables = OFF;
```

To make this change persistent across server restarts, add the following to your MariaDB configuration file (for instance, `my.cnf`):

```
[mysqld]
aria_encrypt_tables = OFF
```

Internal On-Disk Temporary Tables

MariaDB creates internal temporary tables using the Aria engine when `aria_used_for_temp_tables` is set to `ON`. To ensure these are no longer encrypted, set:

```
SET GLOBAL encrypt_tmp_disk_tables = OFF;
```

Step 2

Identify encrypted Aria tables.

Aria does not use background encryption threads (unlike InnoDB). Therefore, tables already on disk will remain encrypted until you manually rebuild them.

Run the following query to identify which tables need to be rebuilt:

```
SELECT TABLE_SCHEMA, TABLE_NAME
FROM information_schema.TABLES
WHERE ENGINE = 'Aria'
  AND ROW_FORMAT = 'PAGE'
  AND TABLE_SCHEMA NOT IN ('information_schema', 'performance_schema', 'sys')
  AND (CREATE_OPTIONS LIKE '%`encrypted`=yes%' OR CREATE_OPTIONS LIKE '%`encrypted`=1%');
```

> **Step 3**
>
> **Rebuild tables to decrypt.**
>
> To decrypt the tables identified in the previous step, you must rebuild them using an `ALTER TABLE` statement. This causes MariaDB to rewrite the data (`.MAD`) and index (`.MAI`) files in an unencrypted format.
>
> **Option A: Manual Rebuild (Single Table)**
>
> Use the following statement to rebuild a specific table:
>
> ```
> ALTER TABLE db_name.table_name ENGINE=Aria, ALGORITHM=COPY;
> ```
>
> **Option B: Generate Rebuild Statements (Bulk)**
>
> You can generate the DDL for all encrypted Aria tables at once using this query:
>
> ```
> SELECT CONCAT('ALTER TABLE `', table_schema, '`.`', table_name,
>              '` ENGINE=Aria, ALGORITHM=COPY;') AS ddl
> FROM information_schema.tables
> WHERE ENGINE='Aria'
>   AND TABLE_SCHEMA NOT IN ('mysql','information_schema','performance_schema','sys')
>   AND (CREATE_OPTIONS LIKE '%`encrypted`=yes%' OR CREATE_OPTIONS LIKE '%`encrypted`=1%');
> ```

> **Step 4**
>
> **Verify and clean up.**
>
> - Verify: Run the query from Step 2 again to ensure no encrypted Aria tables remain.
> - Remove Plugins (optional): Once all Aria tables (and any InnoDB tables or binary logs) are decrypted, you can safely remove the Encryption Key Management plugin from your configuration file and restart the server.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Encryption

See InnoDB: Encryption Overview.

Introduction to InnoDB's encryption architecture, explaining how data is encrypted/decrypted during disk I/O, the role of the buffer pool (where data is unencrypted), and how to verify encryption stat

See InnoDB: Enabling Encryption.

Step-by-step guide to enabling encryption for InnoDB, covering the configuration of innodb_encrypt_tables for automatic encryption and the use of ENCRYPTED=YES table options for per-table encryption.

See InnoDB: Encryption Keys.

How InnoDB manages encryption keys using 32-bit integer IDs, including the default key ID (innodb_default_encryption_key_id), assigning specific keys to tables, and the process of key rotation.

See InnoDB: Disabling Encryption.

Instructions for safely disabling encryption on InnoDB tables, emphasizing the critical need to decrypt all tablespaces and redo logs using background threads or ALTER TABLE.

See InnoDB: Background Encryption Threads.

Details the operation of background threads (configured via innodb_encryption_threads) which handle key rotation, and the encryption/decryption of tablespaces when global settings.

See InnoDB: Encryption Troubleshooting.

Solutions for common issues such as Error 1005 (Wrong create options) when configuring encryption, and handling cases where encryption key IDs are set for unencrypted tables.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB: Encryption Overview

MariaDB supports data-at-rest encryption for tables using the InnoDB storage engines. When enabled, the server encrypts data when it writes it to and decrypts data when it reads it from the file system. You can configure InnoDB encryption to automatically have all new InnoDB tables automatically encrypted, or specify encrypt per table.

For encrypting data with the Aria storage engine, see Encrypting Data for Aria.

InnoDB Encryption and Decryption Behavior

When data-at-rest encryption is enabled for InnoDB, encryption and decryption occur at specific points during disk I/O operations.

When is InnoDB data encrypted?

When InnoDB pages are written to disk, they are automatically encrypted.

When is InnoDB data decrypted?

InnoDB pages are decrypted before they are read from disk and stored in the InnoDB buffer pool. A page remains decrypted in memory while it is in the buffer pool. As a result, decrypted memory pages may include data from rows, columns, or even tables that the current query does not directly access.

Basic Configuration

Using data-at-rest encryption requires that you first configure an Encryption Key Management plugin, such as the file_key_management or aws_key_management plugins.
MariaDB uses this plugin to store, retrieve and manage the various keys it uses when encrypting data to and decrypting data from the file system.

Once you have the plugin configured, you need to set a few additional system variables to enable encryption on InnoDB tables, including innodb_encrypt_tables, innodb_encrypt_log, innodb_encryption_threads, innodb_encrypt_temporary_tables and innodb_encryption_rotate_key_age.

```
[mariadb]
...

# File Key Management
plugin_load_add = file_key_management
file_key_management_filename = /etc/mysql/encryption/keyfile.enc
file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
file_key_management_encryption_algorithm = AES_CTR

# InnoDB Encryption
innodb_encrypt_tables = ON
innodb_encrypt_temporary_tables = ON
innodb_encrypt_log = ON
innodb_encryption_threads = 4
innodb_encryption_rotate_key_age = 1
```

For more information on system variables for encryption and other features, see the InnoDB system variables page.

Creating Encrypted Tables

To create encrypted tables, specify the table options `ENCRYPTED=YES` and `ENCRYPTION_KEY_ID=` with a corresponding key id;

```
CREATE TABLE t (i INT PRIMARY KEY) ENGINE=InnoDB ENCRYPTED=YES ENCRYPTION_KEY_ID=2;
```

Finding Encrypted Tables

When using data-at-rest encryption with the InnoDB storage engine, it is not necessary that you encrypt every table in your database. You can check which tables are encrypted and which are not by querying the INNODB_TABLESPACES_ENCRYPTION table in the Information Schema. This table provides information on which tablespaces are encrypted, which encryption key each tablespace is encrypted with, and whether the background encryption threads are currently working on the tablespace. Since the system tablespace can also contain tables, it can be helpful to join the `INNODB_TABLESPACES_ENCRYPTION` table with the INNODB_SYS_TABLES table to find out the encryption status of each specific table, rather than each tablespace. For example:

```
SELECT st.SPACE, st.NAME, te.ENCRYPTION_SCHEME, te.ROTATING_OR_FLUSHING
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION te
JOIN information_schema.INNODB_SYS_TABLES st
   ON te.SPACE = st.SPACE \G
*************************** 1. row ***************************
               SPACE: 0
                NAME: SYS_DATAFILES
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 2. row ***************************
               SPACE: 0
                NAME: SYS_FOREIGN
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 3. row ***************************
               SPACE: 0
                NAME: SYS_FOREIGN_COLS
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 4. row ***************************
               SPACE: 0
                NAME: SYS_TABLESPACES
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 5. row ***************************
               SPACE: 0
                NAME: SYS_VIRTUAL
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 6. row ***************************
               SPACE: 0
                NAME: db1/default_encrypted_tab1
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 7. row ***************************
               SPACE: 416
                NAME: db1/default_encrypted_tab2
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 8. row ***************************
               SPACE: 402
                NAME: db1/tab
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 9. row ***************************
               SPACE: 185
                NAME: db1/tab1
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 10. row ***************************
               SPACE: 184
                NAME: db1/tab2
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 11. row ***************************
               SPACE: 414
                NAME: db1/testgb2
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 12. row ***************************
               SPACE: 4
                NAME: mysql/gtid_slave_pos
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 13. row ***************************
               SPACE: 2
                NAME: mysql/innodb_index_stats
   ENCRYPTION_SCHEME: 1
```

```
ROTATING_OR_FLUSHING: 0
*************************** 14. row ***************************
               SPACE: 1
                NAME: mysql/innodb_table_stats
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
*************************** 15. row ***************************
               SPACE: 3
                NAME: mysql/transaction_registry
   ENCRYPTION_SCHEME: 1
ROTATING_OR_FLUSHING: 0
15 rows in set (0.000 sec)
```

Redo Logs

Using data-at-rest encryption with InnoDB, the innodb_encrypt_tables system variable only encrypts the InnoDB tablespaces. In order to also encrypt the InnoDB Redo Logs, you also need to set the innodb_encrypt_log system variable.

Where the encryption key management plugin supports key rotation, the InnoDB Redo Log can also rotate encryption keys. In previous releases, the Redo Log can only use the first encryption key.

See Also

- Data at Rest Encryption
- Encryption Key Management
- Information Schema INNODB_TABLESPACES_ENCRYPTION table

This page is licensed: CC BY-SA / Gnu FDL

InnoDB: Enabling Encryption

In order to enable data-at-rest encryption for tables using the InnoDB storage engines, you first need to configure the server to use an Encryption Key Management plugin. Once this is done, you can enable encryption by setting the innodb_encrypt_tables system variable to encrypt the InnoDB system and file tablespaces and setting the innodb_encrypt_log system variable to encrypt the InnoDB Redo Log.

Setting these system variables enables the encryption feature for InnoDB tables on your server. To use the feature, you need to use the ENCRYPTION_KEY_ID table option to set what encryption key you want to use and set the ENCRYPTED table option to enable encryption.

When encrypting any InnoDB tables, the best practice is also enable encryption for the Redo Log. If you have encrypted InnoDB tables and have not encrypted the Redo Log, data written to an encrypted table may be found unencrypted in the Redo Log.

Enabling Encryption for Automatically Encrypted Tablespaces

The innodb_encrypt_tables system variable controls the configuration of automatic encryption of InnoDB tables. It has the following possible values:

Option	Description
OFF	Disables table encryption.
ON	Enables table encryption, but allows unencrypted tables to be created.
FORCE	Enables table encryption, and doesn't allow unencrypted tables to be created.

When innodb_encrypt_tables is set to `ON`, InnoDB tables are automatically encrypted by default. For example, the following statements create an encrypted table and confirm that it is encrypted:

```
SET GLOBAL innodb_encryption_threads=4;

SET GLOBAL innodb_encrypt_tables=ON;

SET SESSION innodb_default_encryption_key_id=100;

CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
);

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+-------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+-------------------+----------------+
| db1/tab1 |                 1 |            100 |
+----------+-------------------+----------------+
```

When innodb_encrypt_tables is set to `ON`, an unencrypted InnoDB table can be created by setting the ENCRYPTED table option to `NO` for the table. For example, the following statements create an unencrypted table and confirm that it is not encrypted:

```
SET GLOBAL innodb_encryption_threads=4;

SET GLOBAL innodb_encrypt_tables=ON;

SET SESSION innodb_default_encryption_key_id=100;

CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTED=NO;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+-------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+-------------------+----------------+
| db1/tab1 |                 0 |            100 |
+----------+-------------------+----------------+
```

When innodb_encrypt_tables is set to `FORCE`, InnoDB tables are automatically encrypted by default, and unencrypted InnoDB tables can **not** be created. In this scenario, if you set the ENCRYPTED table option to `NO` for a table, then you will encounter an error. For example:

```
SET GLOBAL innodb_encryption_threads=4;

SET GLOBAL innodb_encrypt_tables='FORCE';

SET SESSION innodb_default_encryption_key_id=100;

CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTED=NO;
ERROR 1005 (HY000): Can't create table `db1`.`tab1` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------+
| Level   | Code | Message                                                               |
+---------+------+-----------------------------------------------------------------------+
| Warning |  140 | InnoDB: ENCRYPTED=NO implies ENCRYPTION_KEY_ID=1                      |
| Warning |  140 | InnoDB: ENCRYPTED=NO cannot be used with innodb_encrypt_tables=FORCE |
| Error   | 1005 | Can't create table `db1`.`tab1` (errno: 140 "Wrong create options")  |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB      |
+---------+------+-----------------------------------------------------------------------+
4 rows in set (0.00 sec)
```

When innodb_encrypt_tables is set to `ON` or `FORCE`, then you must ensure that innodb_encryption_threads is set to a non-zero value, so that InnoDB can perform any necessary encryption operations in the background. See background operations for more information about that. innodb_encryption_rotate_key_age must also be set to a non-zero value for the initial encryption operations to happen in the background. See disabling key rotations for more information about that.

Enabling Encryption for Manually Encrypted Tablespaces

If you do not want to automatically encrypt every InnoDB table, then it is possible to manually enable encryption for just the subset of InnoDB tables that you would like to encrypt. MariaDB provides the ENCRYPTED and ENCRYPTION_KEY_ID table options that can be used to manually enable encryption for specific InnoDB tables. These table options can be used with CREATE TABLE and ALTER TABLE statements. These table options can only be used with InnoDB tables that have their own InnoDB's file-per-table tablespaces, meaning that tables that were created with innodb_file_per_table=ON set.

Table Option	Value	Description
ENCRYPTED	YES / NO / DEFAULT	Defines whether or not to encrypt the table. `DEFAULT` means that the decision is based on the global innodb_encrypt_tables setting.
ENCRYPTION_KEY_ID	32-bit integer	Defines the identifier for the encryption key to use

You can manually enable or disable encryption for a table by using the ENCRYPTED table option. If you only need to protect a subset of InnoDB tables with encryption, then it can be a good idea to manually encrypt each table that needs the extra protection, rather than encrypting all InnoDB tables globally with innodb_encrypt_tables. This allows you to balance security with speed, as it means the encryption and decryption performance overhead only applies to those tables that require the additional security.

If a manually encrypted InnoDB table contains a FULLTEXT INDEX, then the internal table for the full-text index will not also be manually encrypted. To encrypt internal tables for InnoDB full-text indexes, you must enable automatic InnoDB encryption by setting innodb_encrypt_tables to `ON` or `FORCE`.

You can also manually specify a encryption key for a table by using the ENCRYPTION_KEY_ID table option. This allows you to use different encryption keys for different tables. For example, you might create a table using a statement like this:

```
CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTED=YES ENCRYPTION_KEY_ID=100;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+------------------+----------------+
| db1/tab1 |                1 |            100 |
+----------+------------------+----------------+
```

If the ENCRYPTION_KEY_ID table option is not specified, then the table will be encrypted with the key identified by the innodb_default_encryption_key_id system variable. For example, you might create a table using a statement like this:

```
SET SESSION innodb_default_encryption_key_id=100;

CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTED=YES;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+------------------+----------------+
| db1/tab1 |                1 |            100 |
+----------+------------------+----------------+
```

In the event that you have an existing table and you want to manually enable encryption for that table, then you can do the same with an ALTER TABLE statement. For example:

```
CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTED=NO;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+------------------+----------------+
| db1/tab1 |                0 |            100 |
+----------+------------------+----------------+

ALTER TABLE tab1
   ENCRYPTED=YES ENCRYPTION_KEY_ID=100;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+------------------+----------------+
| db1/tab1 |                1 |            100 |
+----------+------------------+----------------+
```

InnoDB does not permit manual encryption changes to tables in the system tablespace using ALTER TABLE. Encryption of the system tablespace can only be configured by setting the value of the innodb_encrypt_tables

system variable. This means that when you want to encrypt or decrypt the system tablespace, you must also set a non-zero value for the innodb_encryption_threads system variable, and you must also set the innodb_system_rotate_key_age system variable to `1` to ensure that the system tablespace is properly encrypted or decrypted by the background threads. See MDEV-14398 for more information.

Enabling Encryption for Temporary Tablespaces

The innodb_encrypt_temporary_tables system variable controls the configuration of encryption for the temporary tablespace. It has the following possible values:

Option	Description
OFF	Disables temporary table encryption.
ON	Enables temporary table encryption.

This system variable can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
innodb_encrypt_temporary_tables=ON
```

Enabling Encryption for the Redo Log

InnoDB uses the Redo Log in crash recovery. By default, these events are written to file in an unencrypted state. In configuring MariaDB for data-at-rest encryption, ensure that you also enable encryption for the Redo Log.

To encrypt the Redo Log, first stop the server process. Then, set the innodb_encrypt_log to `ON` in a relevant server option group in an option file. For example:

```
[mariadb]
innodb_encrypt_log = ON
```

Then, start MariaDB. When the server starts back up, it checks to recover InnoDB in the event of a crash. Once it is back online, it begins writing encrypted data to the Redo Log.

Key rotation for the Redo Log is supported. See InnoDB Encryption Keys: Key Rotation for more information.

See Also

- Disabling InnoDB encryption

This page is licensed: CC BY-SA / Gnu FDL

InnoDB: Encryption Keys

InnoDB uses encryption key management plugins to support the use of multiple encryption keys.

Encryption Keys

Each encryption key has a 32-bit integer that serves as a key identifier.

The default key is set using the innodb_default_encryption_key_id system variable.

Encryption keys can also be specified with the ENCRYPTION_KEY_ID table option for tables that use file-per-table tablespaces.

InnoDB encrypts the temporary tablespace using the encryption key with the ID `1`.

InnoDB encrypts the Redo Log using the encryption key with the ID `1`.

Keys with Manually Encrypted Tablespaces

With tables that use manually enabled encryption, one way to set the specific encryption key for the table is to use the ENCRYPTION_KEY_ID table option. For example:

```
CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTED=YES ENCRYPTION_KEY_ID=100;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+------------------+----------------+
| db1/tab1 |                1 |            100 |
+----------+------------------+----------------+
```

If the ENCRYPTION_KEY_ID table option is not set for a table that uses manually enabled encryption, then it will inherit the value from the innodb_default_encryption_key_id system variable. For example:

```
SET SESSION innodb_default_encryption_key_id=100;

CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTED=YES;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+------------------+----------------+
| db1/tab1 |                1 |            100 |
+----------+------------------+----------------+
```

Keys with Automatically Encrypted Tablespaces

With tables that use automatically enabled encryption, one way to set the specific encryption key for the table is to use the innodb_default_encryption_key_id system variable. For example:

```
SET GLOBAL innodb_encryption_threads=4;

SET GLOBAL innodb_encrypt_tables=ON;

SET SESSION innodb_default_encryption_key_id=100;

CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
);

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+------------------+----------------+
| db1/tab1 |                1 |            100 |
+----------+------------------+----------------+
```

InnoDB tables that are part of the system tablespace can only be encrypted using the encryption key set by the innodb_default_encryption_key_id system variable.

If the table is in a file-per-table tablespace, and if innodb_encrypt_tables is set to `ON` or `FORCE`, and if innodb_encryption_threads is set to a value greater than `0`, then you can also set the specific encryption key for the table by using the ENCRYPTION_KEY_ID table option. For example:

```
SET GLOBAL innodb_encryption_threads=4;

SET GLOBAL innodb_encrypt_tables=ON;

CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTION_KEY_ID=100;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
    -> FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
    -> WHERE NAME='db1/tab1';
+----------+------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+------------------+----------------+
| db1/tab1 |                1 |            100 |
+----------+------------------+----------------+
```

However, if innodb_encrypt_tables is set to `OFF` or if innodb_encryption_threads is set to `0`, then this will not work. See InnoDB Encryption Troubleshooting: Setting Encryption Key ID For an Unencrypted Table for more information.

Key Rotation

Some key management and encryption plugins allow you to automatically rotate and version your encryption keys. If a plugin support key rotation, and if it rotates the encryption keys, then InnoDB's background encryption threads can re-encrypt InnoDB pages that use the old key version with the new key version.

You can set the maximum age for an encryption key using the innodb_encryption_rotate_key_age system variable. When this variable is set to a non-zero value, background encryption threads constantly check pages to determine if any page is encrypted with a key version that's too old. When the key version is too old, any page encrypted with the older version of the key is automatically re-encrypted in the background to use a more current version of the key. Bear in mind, this constant checking can sometimes result in high CPU usage.

Key rotation for the InnoDB Redo Log is only supported in MariaDB 10.4.0 and later. For more information, see MDEV-12041.

In order for key rotation to work, both the backend key management service (KMS) and the corresponding key management and encryption plugin have to support key rotation. See Encryption Key Management: Support for Key Rotation in Encryption Plugins to determine which plugins currently support key rotation.

Disabling Background Key Rotation Operations

In the event that you encounter issues with background key encryption, you can disable it by setting the innodb_encryption_rotate_key_age system variable to `0`. You may find this useful when the constant key version checks lead to excessive CPU usage. It's also useful in cases where your encryption key management plugin does not support key rotation, (such as with the file_key_management plugin). For more information, see MDEV-14180.

There are, however, issues that can arise when the background key rotation is disabled.

Pending Encryption Operations

When updating the value on the innodb_encrypt_tables system variable, InnoDB internally treats the subsequent background operations to encrypt and decrypt tablespaces as background key rotations. See MDEV-14398 for more information.

You can check the status of background encryption operations by querying the INNODB_TABLESPACES_ENCRYPTION table in the information_schema database.

See InnoDB Background Encryption Threads: Checking the Status of Background Operations for some example queries.

For more information, see MDEV-14398.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB: Disabling Encryption

Overview

Decryption for InnoDB is more complex than Aria because it involves background threads and the Redo Log.

This guide covers the process for safely decrypting InnoDB tablespaces, the system tablespace, and the Redo Log.

> **Warning:**
>
> **Important Safety Precautions**
>
> - Do Not Remove Plugins Early: Keep your Key Management plugin configured and loaded until the very end. If you remove the keys before decryption is complete, the encrypted data will become permanently inaccessible.
> - Order Matters: You must decrypt all tablespaces and the Redo Log before disabling the encryption plugin.

Step 1

Disable automatic encryption.

To stop the server from encrypting new tables and to begin the background decryption process for “automatically” encrypted tables (those where `ENCRYPTED=DEFAULT`), update the global system variables.

```
-- Disable encryption for new tables and the system tablespace
SET GLOBAL innodb_encrypt_tables = OFF;

-- Enable encryption threads to perform the decryption work
SET GLOBAL innodb_encryption_threads = 4;

-- Force rotation to unencrypted state by setting age to 1
SET GLOBAL innodb_encryption_rotate_key_age = 1;
```

To make these changes persistent, update your MariaDB configuration file:

```
[mariadb]
innodb_encrypt_tables = OFF
innodb_encryption_threads = 4
innodb_encryption_rotate_key_age = 1
```

Step 2

Decrypt manually encrypted tables.

Tables created with the explicit option `ENCRYPTED=YES` are not always automatically decrypted by background threads. You must manually issue an `ALTER TABLE` statement for these.

Identify Manually Encrypted Tables

Run this query to find tables that require manual decryption:

```
SELECT TABLE_SCHEMA, TABLE_NAME
FROM information_schema.TABLES
WHERE ENGINE='InnoDB'
  AND (CREATE_OPTIONS LIKE '%ENCRYPTED=YES%' OR CREATE_OPTIONS LIKE '%ENCRYPTION="Y"%');
```

Perform Decryption

For each table identified, run:

```
ALTER TABLE db_name.table_name ENCRYPTED=NO;
```

Step 3

Decrypt the redo log.

The Redo Log must be decrypted separately. This requires a server restart.

The Redo Log is decrypted by ensuring the server can read the existing logs at startup and then configuring it to write new logs in plaintext.

1. Ensure keys are active: Before attempting to disable Redo Log encryption, verify that your key management plugin is fully functional. MariaDB must be able to decrypt the current `ib_logfile` members during the startup/recovery phase.
2. Update Configuration: Change the `innodb_encrypt_log` system variable to `OFF` in your server option file (for instance, `my.cnf`):

   ```
   [mariadb]
   # Ensure the plugin remains loaded for this restart!
   innodb_encrypt_log = OFF
   ```

3. Restart the server – perform a clean restart:

   ```
   sudo systemctl restart mariadb
   ```

   During this restart, MariaDB uses the existing keys to read the encrypted Redo Logs, completes any necessary crash recovery, and then begins writing new Redo Log events in plaintext.

4. Verification: Confirm the status by running:

   ```
   SHOW GLOBAL VARIABLES LIKE 'innodb_encrypt_log';
   ```

   The value should now be `OFF`.

Step 4

Monitor and verify decryption status.

Before removing your encryption keys, you must verify that no tablespaces remain encrypted.

Check Background Progress

Monitor the `INNODB_TABLESPACES_ENCRYPTION` table. Decryption is complete when the count reaches `0`.

```
SELECT COUNT(*) AS "Encrypted_Tablespaces"
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE ENCRYPTION_SCHEME != 0 OR ROTATING_OR_FLUSHING != 0;
```

Verify Individual Tables

Ensure that no tables show encryption in their creation options:

```
SELECT TABLE_NAME, CREATE_OPTIONS
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'your_database_name';
```

Step 5

Clean up.

Once the count of encrypted tablespaces is `0` and the Redo Log has been rotated (after restart), you can safely:

1. Remove the Encryption Key Management plugin settings from your configuration file.
2. Restart the MariaDB Server one final time.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB: Background Encryption Threads

InnoDB performs some encryption and decryption operations with background encryption threads. The innodb_encryption_threads system variable controls the number of threads that the storage engine uses for encryption-related background operations, including encrypting and decrypting pages after key rotations or configuration changes, and scrubbing data to permanently delete it.

Background Operations

InnoDB performs the following encryption and decryption operations using background encryption threads:

- When rotating encryption keys, InnoDB's background encryption threads re-encrypt pages that use key versions older than innodb_encryption_rotate_key_age to the new key version.
- When changing the innodb_encrypt_tables system variable to `FORCE`, InnoDB's background encryption threads encrypt the system tablespace and any file-per-table tablespaces that have the ENCRYPTED table option set to `DEFAULT`.
- When changing the innodb_encrypt_tables system variable to `OFF`, InnoDB's background encryption threads decrypt the system tablespace and any file-per-table tablespacs that have the `ENCRYPTED` table option set to `DEFAULT`.

The innodb_encryption_rotation_iops system variable can be used to configure how many I/O operations you want to allow for the operations performed by InnoDB's background encryption threads.

Whenever you change the value on the innodb_encrypt_tables system variable, InnoDB's background encryption threads perform the necessary encryption or decryption operations. Because of this, you must have a non-zero value set for the innodb_encryption_threads system variable. InnoDB also considers these operations to be key

rotations internally. Because of this, you must have a non-zero value set for the innodb_encryption_rotate_key_age system variable. For more information, see disabling key rotations.

Non-Background Operations

InnoDB performs the following encryption and decryption operations **without** using background encryption threads:

- When a file-per-table tablespaces and using ALTER TABLE to manually set the ENCRYPTED table option to `YES`, InnoDB does not use background threads to encrypt the tablespaces.
- Similarly, when using `file-per-table` tablespaces and using ALTER TABLE to manually set the ENCRYPTED table option to `NO`, InnoDB does not use background threads to decrypt the tablespaces.

In these cases, InnoDB performs the encryption or decryption operation using the server thread for the client connection that executes the statement. This means that you can update encryption on file-per-table tablespaces with an ALTER TABLE statement, even when the innodb_encryption_threads and/or the innodb_rotate_key_age system variables are set to `0`.

InnoDB does not permit manual encryption changes to tables in the system tablespace using ALTER TABLE. Encryption of the system tablespace can only be configured by setting the value of the innodb_encrypt_tables system variable. This means that when you want to encrypt or decrypt the system tablespace, you must also set a non-zero value for the innodb_encryption_threads system variable, and you must also set the innodb_system_rotate_key_age system variable to `1` to ensure that the system tablespace is properly encrypted or decrypted by the background threads. See MDEV-14398 for more information.

Checking the Status of Background Operations

InnoDB records the status of background encryption operations in the INNODB_TABLESPACES_ENCRYPTION table in the information_schema database.

For example, to see which InnoDB tablespaces are currently being decrypted or encrypted on by background encryption, you can check which InnoDB tablespaces have the `ROTATING_OR_FLUSHING` column set to `1`:

```
SELECT SPACE, NAME
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE ROTATING_OR_FLUSHING = 1;
```

And to see how many InnoDB tablespaces are currently being decrypted or encrypted by background encryption threads, you can call the COUNT() aggregate function.

```
SELECT COUNT(*) AS 'encrypting'
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE ROTATING_OR_FLUSHING = 1;
```

And to see how many InnoDB tablespaces are currently being decrypted or encrypted by background encryption threads, while comparing that to the total number of InnoDB tablespaces and the total number of encrypted InnoDB tablespaces, you can join the table with the INNODB_SYS_TABLESPACES table in the information_schema database:

```
/* information_schema.INNODB_TABLESPACES_ENCRYPTION does not always have rows for all tablespaces,
  so let's join it with information_schema.INNODB_SYS_TABLESPACES */
WITH tablespace_ids AS (
   SELECT SPACE
   FROM information_schema.INNODB_SYS_TABLESPACES ist
   UNION
   /* information_schema.INNODB_SYS_TABLESPACES doesn't have a row for the system tablespace (MDEV-20802)
*/
   SELECT 0 AS SPACE
)
SELECT NOW() AS 'time',
   'tablespaces', COUNT(*) AS 'tablespaces',
   'encrypted', SUM(IF(ite.ENCRYPTION_SCHEME IS NOT NULL, ite.ENCRYPTION_SCHEME, 0)) AS 'encrypted',
   'encrypting', SUM(IF(ite.ROTATING_OR_FLUSHING IS NOT NULL, ite.ROTATING_OR_FLUSHING, 0)) AS
'encrypting'
FROM tablespace_ids
LEFT JOIN information_schema.INNODB_TABLESPACES_ENCRYPTION ite
   ON tablespace_ids.SPACE = ite.SPACE
```

This page is licensed: CC BY-SA / Gnu FDL

InnoDB: Encryption Troubleshooting

Wrong Create Options

With InnoDB tables using encryption, there are several cases where a CREATE TABLE or ALTER TABLE statement can throw Error 1005, due to the InnoDB error 140, `Wrong create options`:

```
CREATE TABLE `test`.`table1` ( `id` INT(4) PRIMARY KEY , `name` VARCHAR(50));
ERROR 1005 (HY000): Can't create table `test`.`table1` (errno: 140 "Wrong create options")
```

When this occurs, you can usually get more information about the cause of the error by following it with a SHOW WARNINGS statement.

This error is known to occur in the following cases:

- Encrypting a table by setting the ENCRYPTED table option to `YES` when the innodb_file_per_table is set to `OFF`.In this case, SHOW WARNINGS would return the following:

```
SHOW WARNINGS;
+---------+------+-------------------------------------------------------------------------+
| Level   | Code | Message                                                                 |
+---------+------+-------------------------------------------------------------------------+
| Warning |  140 | InnoDB: ENCRYPTED requires innodb_file_per_table                        |
| Error   | 1005 | Can't create table `db1`.`tab3` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB     |
+---------+------+-------------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

- Encrypting a table by setting the ENCRYPTED table option to `YES`, and the innodb_default_encryption_key_id system variable or the ENCRYPTION_KEY_ID table option refers to a non-existent key identifier. In this case, SHOW WARNINGS would return the following:

```
SHOW WARNINGS;
+---------+------+-------------------------------------------------------------------------+
| Level   | Code | Message                                                                 |
+---------+------+-------------------------------------------------------------------------+
| Warning |  140 | InnoDB: ENCRYPTION_KEY_ID 500 not available                             |
| Error   | 1005 | Can't create table `db1`.`tab3` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB     |
+---------+------+-------------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

- In some versions, this could happen while creating a table with the ENCRYPTED table option set to `DEFAULT` while the innodb_encrypt_tables system variable is set to `OFF`, and the innodb_default_encryption_key_id system variable or the ENCRYPTION_KEY_ID table option are not set to `1`. In this case, SHOW WARNINGS would return the following:

```
SHOW WARNINGS;
+---------+------+----------------------------------------------------------------------+
| Level   | Code | Message                                                              |
+---------+------+----------------------------------------------------------------------+
| Warning |  140 | InnoDB: innodb_encrypt_tables=OFF only allows ENCRYPTION_KEY_ID=1    |
| Error   | 1005 | Can't create table `db1`.`tab3` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB      |
+---------+------+----------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

Creating a table with the ENCRYPTED table option set to `DEFAULT` while the innodb_encrypt_tables system variable is set to `OFF`, and the innodb_default_encryption_key_id system variable or the ENCRYPTION_KEY_ID table option are **not** set to `1` no longer fail, and it no longer throws a warning.

For more information, see MDEV-18601.

Setting Encryption Key ID For an Unencrypted Table

If you set the ENCRYPTION_KEY_ID table option for a table that is unencrypted because the innodb_encrypt_tables system variable is set to `OFF` and the ENCRYPTED table option set to `DEFAULT`, then this encryption key ID will be saved in the table's `.frm` file, but the encryption key will not be saved to the table's `.ibd` file.

As a side effect, with the current encryption design, if the innodb_encrypt_tables system variable is later set to `ON`, and InnoDB goes to encrypt the table, then the InnoDB background encryption threads will not read this encryption key ID from the `.frm` file. Instead, the threads may encrypt the table with the encryption key with ID `1`, which is internally considered the default encryption key when no key is specified. For example:

```
SET GLOBAL innodb_encrypt_tables=OFF;

CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTION_KEY_ID=100;

SET GLOBAL innodb_encrypt_tables=ON;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME='db1/tab1';
+----------+-------------------+----------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+-------------------+----------------+
| db1/tab1 |                 1 |              1 |
+----------+-------------------+----------------+
```

A similar problem is that, if you set the ENCRYPTION_KEY_ID table option for a table that is unencrypted because the ENCRYPTED table option is set to `NO`, then this encryption key ID will be saved in the table's `.frm` file, but the encryption key will not be saved to the table's `.ibd` file.

Recent versions of MariaDB will throw warnings in the case where the ENCRYPTED table option is set to `NO`, but they will allow the operation to succeed. For example:

```
CREATE TABLE tab1 (
   id INT PRIMARY KEY,
   str VARCHAR(50)
) ENCRYPTED=NO ENCRYPTION_KEY_ID=100;
Query OK, 0 rows affected, 1 warning (0.01 sec)

SHOW WARNINGS;
+---------+------+-----------------------------------------------------+
| Level   | Code | Message                                             |
+---------+------+-----------------------------------------------------+
| Warning |  140 | InnoDB: ENCRYPTED=NO implies ENCRYPTION_KEY_ID=1 |
+---------+------+-----------------------------------------------------+
1 row in set (0.00 sec)
```

However, in this case, if you change the ENCRYPTED table option to `YES` or `DEFAULT` with ALTER TABLE, then it will actually use the proper key. For example:

```
SET GLOBAL innodb_encrypt_tables=ON;

ALTER TABLE tab1 ENCRYPTED=DEFAULT;

SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME = 'db1/tab1';
+----------+------------------+---------------+
| NAME     | ENCRYPTION_SCHEME | CURRENT_KEY_ID |
+----------+------------------+---------------+
| db1/tab1 |                1 |            100 |
+----------+------------------+---------------+
```

For more information, see MDEV-17230, MDEV-18601, and MDEV-19086.

Tablespaces Created on MySQL 5.1.47 or Earlier

MariaDB's data-at-rest encryption implementation reused previously unused fields in InnoDB's buffer pool pages to identify the encryption key version and the post-encryption checksum. Prior to MySQL 5.1.48, these unused fields were not initialized in memory due to performance concerns. These fields still had zero values most of the time, but since they were not explicitly initialized, that means that these fields could have occasionally had non-zero values that could have been written into InnoDB's tablespace files. If MariaDB were to encounter an unencrypted page from a tablespace file that was created on an early version of MySQL that also had non-zero values in these fields, then it would mistakenly think that the page was encrypted.

The fix for MDEV-12112 changed the way that MariaDB distinguishes between encrypted and unencrypted pages, so that it is less likely to mistake an unencrypted page for an encrypted page.

If innodb_checksum_algorithm is set to `full_crc32` or `strict_full_crc32`, and if the table does not use ROW_FORMAT=COMPRESSED, then data files are guaranteed to be zero-initialized.

For more information, see MDEV-18097.

Spatial Indexes

Support for encrypting spatial indexes. To enable, set the innodb_checksum_algorithm to `full_crc32` or to `strict_full_crc32`. Note that MariaDB only encrypts spatial indexes when the ROW_FORMAT table option is **not** set to COMPRESSED.

For more information, see MDEV-12026.

Managing CPU Usage During InnoDB Encryption Operations

When encryption threads are enabled for InnoDB tables, the system initiates background threads to encrypt existing tablespace pages. During this initial process, you may see significant spikes in CPU and I/O usage. This can occur due to:

- **Periodic key rotation checks** that trigger re-encryption work.
- **Large datasets** being encrypted, especially if many tables are enabled at once.
- **Clustered environments** such as MariaDB Galera Cluster, where each node performs encryption operations independently.

CPU usage can increase under several conditions. Common contributing factors are included below.

Causes of CPU Spikes

CPU usage increases during table encryption due to the following factors:

- **CPU usage spikes after enabling encryption**: This is expected because InnoDB must begin encrypting existing tablespace pages in the background once encryption is turned on. The initial workload is heavy.
- **Increased disk I/O activity**: The process involves reading unencrypted pages, encrypting them, and writing them back as encrypted. This read–encrypt–write cycle generates both CPU load (for encryption) and disk I/O load (for page movement).
- **Background threads encrypting tablespace pages:** InnoDB background threads actively encrypt tablespace pages that were encrypted before encryption was enabled. If too many threads are active or configured aggressively, they compete for CPU resources, causing spikes.
- **Performance degradation during initial encryption**: The initial stage of encrypting large tablespaces is the most resource-intensive phase. Once completed, ongoing CPU usage stabilizes, but during the initial encryption, performance can degrade significantly.

This behavior is normal during the one-time setup phase when InnoDB begins encrypting existing tablespace pages. However, the impact can be reduced through:

- **Tuning** system variables such as `innodb_encryption_threads`, `innodb_encryption_rotation_iops`, and `innodb_encryption_rotate_key_age`.
- **Monitoring** background encryption activity to check progress and identify bottlenecks.
- **Phased rollout** of encryption, enabling it for a subset of tables at a time rather than all tables at once. See the Operational Best Practices section.

Configuration Options

Monitor Background Encryption Activity

MariaDB performs encryption in background threads. This helps determine if CPU usage is due to ongoing table encryption or key rotation.

Before changing configuration, verify whether background encryption is currently running by executing the following command:

```
SHOW ENGINE INNODB STATUS
```

You can review the encryption activity output.

Additionally, you can check the essential encryption-related status variables, as detailed in InnoDB: Background Encryption Threads.

Disable Key Rotation Checks

The `innodb_encryption_rotate_key_age` system variable manages how often InnoDB checks tablespace pages to determine whether they need to be re-encrypted with a new key.

Setting this variable to `0` disables automatic key rotation checks if you do not require periodic re-encryption of old pages:

```
SET GLOBAL innodb_encryption_rotate_key_age=0;
```

Once disabled, key rotation checks can significantly reduce CPU usage. See innodb_encryption_rotate_key_age.

Limit Background Encryption Threads

The `innodb_encryption_threads` system variable controls the number of background encryption threads.

Lowering this value can prevent CPU contention at the cost of slower encryption progress.

```
SET GLOBAL innodb_encryption_threads=2;
```

You can maintain the value based on available CPU capacity and workload requirements. See innodb_encryption_threads.

Reduce Encryption Rotation IOPS

The `innodb_encryption_rotation_iops` system variable defines the maximum number of I/O operations per second that InnoDB background threads can use for page encryption and key rotation tasks.

Effect of Lowering the Value:

- **Reduces CPU usage:** fewer encryption operations are performed per second, lowering the load on CPU resources.
- **Lowers disk pressure:** decreases the number of read/write operations, reducing contention on storage I/O.

```
SET GLOBAL innodb_encryption_rotation_iops=50;
```

Setting a lower value (e.g., 25 or 50) limits the threads, making them read and write data more slowly, reducing both I/O and the associated CPU overhead. See innodb_encryption_rotation_iops.

Operational Best Practices

In addition to configuration changes, you can also manage encryption more effectively by applying the following operational practices:

- Encrypt table in batches: Instead of enabling encryption for all tables at once, apply it in stages. This allow background encryption to complete for one set of tables and then proceed with more. With this approach, peak CPU and I/O load can be reduced.
- Galera Cluster setup considerations:
  - Encryption operations are performed independently on each node
  - Use an odd number of nodes (3, 5, 7) to maintain quorum and prevent split-brain scenarios. See Weighted Quorum for details.

See Also

- InnoDB Data-at-Rest Encryption
- InnoDB Background Encryption Threads
- InnoDB System Variables

- Galera Cluster

This page is licensed: CC BY-SA / Gnu FDL

Key Management and Encryption Plugins

See Encryption Key Management.

Overview of key management in MariaDB, discussing the need for plugins to manage encryption keys, support for multiple keys (ID 1 for system, ID 2 for temp), and key rotation capabilities.

See File Key Management Encryption Plugin.

Details the File Key Management plugin, which reads encryption keys from a plain-text (or encrypted) file, serving as a simple solution or reference implementation for data-at-rest encryption.

See AWS Key Management Encryption Plugin.

Introduction to the AWS Key Management plugin, which uses Amazon KMS to generate and store master keys, decrypting them at startup to enable data-at-rest encryption with key rotation support.

See Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Advanced Usage.

Step-by-step tutorial for setting up the AWS KMS plugin, covering the creation of a Customer Master Key (CMK) in AWS, configuring IAM roles for EC2, and installing the plugin from source.

See Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Setup Guide.

Advanced configuration guide for the AWS KMS plugin, detailing how to secure key access using IAM policies, restrict usage by IP address, and implement Multi-Factor Authentication (MFA).

See Hashicorp Key Management Plugin.

Guide to using the HashiCorp Key Management plugin, which integrates MariaDB with HashiCorp Vault for centralized, secure key storage and lifecycle management.

See Uninstall Key Management Plugins.

Final step of removing key management plugins from the configuration once all data and logs have been confirmed as unencrypted.

This page is licensed: CC BY-SA / Gnu FDL

Encryption Key Management

MariaDB's data-at-rest encryption requires the use of a key management and encryption plugin. These plugins are responsible both for the management of encryption keys and for the actual encryption and decryption of data.

MariaDB supports the use of multiple encryption keys. Each encryption key uses a 32-bit integer as a key identifier. If the specific plugin supports key rotation, then encryption keys can also be rotated, which creates a new version of the encryption key.

Supported Key Management Plugins list

Plugin	Status	Key Rotation Support	Notes
File-based key management	Supported	Supported from MariaDB Enterprise Server 11.8	Simple but lacks rotation
HashiCorp Vault plugin	Supported, Recommended	Yes	Best for secure, scalable deployments

Choosing an Encryption Key Management Solution

How MariaDB manages encryption keys depends on which encryption key management solution you choose. Currently, MariaDB has three options:

File Key Management Plugin

The File Key Management plugin that ships with MariaDB is a basic key management and encryption plugin that reads keys from a plain-text file. It can also serve as an example and as a starting point when developing a key management plugin.

For more information, see File Key Management Plugin.

Hashicorp Key Management Plugin

Integrates MariaDB encryption with Vault for secure key management.

For more information, refer to the Hashicorp Key Management Plugin.

AWS Key Management Plugin

The AWS Key Management plugin is a key management and encryption plugin that uses the Amazon Web Services (AWS) Key Management Service (KMS). The AWS Key Management plugin depends on the AWS SDK for C++, which uses the Apache License, Version 2.0. The license is not compatible with MariaDB Server's GPL 2.0 license, so we are not able to distribute packages that contain the AWS Key Management plugin. Therefore, the only way to currently obtain the plugin is to install it from the source.

For more information, see AWS Key Management Plugin.

Using Multiple Encryption Keys

Key management and encryption plugins support using multiple encryption keys. Each encryption key can be defined with a different 32-bit integer as a key identifier.

The support for multiple keys opens up some potential use cases. For example, let's say that a hypothetical key management and encryption plugin is configured to provide two encryption keys. One encryption key might be intended for "low security" tables. It could use short keys, which might not be rotated, and data could be encrypted with a fast encryption algorithm. Another encryption key might be intended for "high security" tables. It could use long keys, which are rotated often, and data could be encrypted with a slower but more secure encryption algorithm. The user would specify the identifier of the key that they want to use for different tables, only using high-level security where it's needed.

There are two encryption key identifiers that have special meanings in MariaDB. An encryption key `1` is intended for encrypting system data, such as InnoDB redo logs, binary logs, and so on. It must always exist when data-at-rest encryption is enabled. An encryption key `2` is intended for encrypting temporary data, such as temporary files and Aria internal temporary tables. It is optional. If it doesn't exist, then MariaDB uses an encryption key `1` for these purposes instead. (InnoDB temporary tables, encrypted with `innodb_encrypt_temporary_tables`, always use key `1`, as they share the redo-log encryption key.)

When encrypting InnoDB tables, the key that is used to encrypt tables can be changed.

When encrypting Aria tables, the key that is used to encrypt tables cannot currently be changed.

Key Rotation

Encryption key rotation is optional in MariaDB Server. Key rotation is only supported if the backend key management service (KMS) supports key rotation and if the corresponding key management and encryption plugin for MariaDB also supports key rotation. When a key management and encryption plugin supports key rotation, users can opt to rotate one or more encryption keys, which creates a new version of each rotated encryption key.

Key rotation allows users to improve data security in the following ways:

- If the server is configured to automatically re-encrypt table data with the newer version of the encryption key after the key is rotated, then that prevents an encryption key from being used for long periods of time.
- If the server is configured to simultaneously encrypt table data with multiple versions of the encryption key after the key is rotated, then that prevents all data from being leaked if a single encryption key version is compromised.

The InnoDB storage engine has background encryption threads that can automatically re-encrypt pages when key rotations occur.

The Aria storage engine does not currently have a similar mechanism to re-encrypt pages in the background when key rotations occur.

Support for Key Rotation in Encryption Plugins

Encryption Plugins with Key Rotation Support

- The AWS Key Management Service (KMS) supports encryption key rotation, and the corresponding AWS Key Management Plugin also supports encryption key rotation.
- **HashiCorp Key Management Plugin**: The HashiCorp Key Management Plugin integrates MariaDB with Hashicorp Key Management Plugin for centralized encryption key storage and lifecycle management environments.

Encryption Plugins without Key Rotation Support

- The File Key Management Plugin does not support encryption key rotation because it does not use a backend key management service (KMS).

Encryption Plugin API

New key management and encryption plugins can be developed using the encryption plugin API.

This page is licensed: CC BY-SA / Gnu FDL

File Key Management Encryption Plugin

MariaDB's data-at-rest encryption requires the use of a key management and encryption plugin. These plugins are responsible both for the management of encryption keys and for the actual encryption and decryption of data.

MariaDB supports the use of multiple encryption keys. Each encryption key uses a 32-bit integer as a key identifier. If the specific plugin supports key rotation, then encryption keys can also be rotated, which creates a new version of the encryption key.

The File Key Management plugin that ships with MariaDB is a key management and encryption plugin that reads encryption keys from a plaintext file.

Overview

The File Key Management plugin is the key management and encryption plugin for users who want to use data-at-rest encryption. Some of the plugin's primary features are:

- It reads encryption keys from a plain-text key file.
- As an extra protection mechanism, the plain-text key file can be encrypted.
- It supports multiple encryption keys.
- It supports key rotation with MariaDB Enterprise Server from MariaDB Enterprise Server 11.8.
- It supports two different algorithms for encrypting data.

It can also serve as an example and as a starting point when developing a key management and encryption plugin with the encryption plugin API.

Installing the File Key Management Plugin's Package

The File Key Management plugin is included in MariaDB packages as the `file_key_management.so` or `file_key_management.dll` shared library. The shared library is in the main server package, so no additional package installations are necessary. The plug-in must be installed into MariaDB however as follows.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. The plugin can be installed by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
plugin_load_add = file_key_management
```

Uninstalling the Plugin

Before you uninstall the plugin, you should ensure that data-at-rest encryption is completely disabled, and that MariaDB no longer needs the plugin to decrypt tables or other files.

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN. For example:

```
UNINSTALL SONAME 'file_key_management';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Creating the Key File

In order to encrypt your tables with encryption keys using the File Key Management plugin, create the file containing the encryption keys. File name and location don't matter; see in the following what configuration is needed.

For each encryption key, the file contains these options, separated by a semicolon:

- The key identifier (format: 32-bit integer)
- The key version (optional, and only available as of Enterprise Server 11.8)
- The key itself (format: hex-encoded)

Entries look like this:

> **CURRENT ENTERPRISE SERVER**
>
> ```
> <encryption_key_id>;<encryption_key_version>;<hex-encoded_encryption_key>
> ```
>
> **COMMUNITY SERVER & ENTERPRISE SERVER < 11.8**
>
> ```
> <encryption_key_id>;<hex-encoded_encryption_key>
> ```

You can also optionally encrypt the key file to make it less accessible from the file system. That is explained further in the section below.

The File Key Management plugin uses Advanced Encryption Standard (AES) to encrypt data. It supports 128-bit, 192-bit, and 256-bit encryption keys, just like the plugin does.

Generate random encryption keys using the openssl rand command. To create a random 256-bit (32-byte) encryption key, run the following command:

```
$ openssl rand -hex 32
a7addd9adea9978fda19f21e6be987880e68ac92632ca052e5bb42b1a506939a
```

The key file needs to have a key identifier for each encryption key added to the beginning of each line. Key identifiers do not have to be contiguous. For example, to append three new encryption keys to a new key file, issue these commands:

CURRENT ENTERPRISE SERVER

```
mkdir -p /etc/mysql/encryption
echo $(echo -n "1;1;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile.txt
echo $(echo -n "2;1;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile.txt
echo $(echo -n "100;2;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile.txt
```

COMMUNITY SERVER(ALL) & ENTERPRISE SERVER < 11.8

```
mkdir -p /etc/mysql/encryption
echo $(echo -n "1;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile.txt
echo $(echo -n "2;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile.txt
echo $(echo -n "100;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile.txt
```

The resulting key file looks like this:

CURRENT ENTERPRISE SERVER

```
1;1;a7addd9adea9978fda19f21e6be987880e68ac92632ca052e5bb42b1a506939a
2;1;49c16acc2dffe616710c9ba9a10b94944a737de1beccb52dc1560abfdd67388b
100;2;8db1ee74580e7e93ab8cf157f02656d356c2f437d548d5bf16bf2a56932954a3
```

COMMUNITY SERVER & ENTERPRISE SERVER < 11.8

```
1;a7addd9adea9978fda19f21e6be987880e68ac92632ca052e5bb42b1a506939a
2;49c16acc2dffe616710c9ba9a10b94944a737de1beccb52dc1560abfdd67388b
100;8db1ee74580e7e93ab8cf157f02656d356c2f437d548d5bf16bf2a56932954a3
```

The key identifiers give you a way to reference the encryption keys from MariaDB. In the example above, you could reference these encryption keys using the key identifiers `1`, `2` or `100` with the ENCRYPTION_KEY_ID table option or with system variables such as innodb_default_encryption_key_id. You do not necessarily need multiple encryption keys; an encryption key with the key identifier `1` is the only mandatory one.

If the key file is left unencrypted, the File Key Management plugin only requires the file_key_management_filename system variable to be configured.

This system variable can be specified as a command-line argument to `mariadbd`, or it can be specified in a server option group of an option file, like this:

```
[mariadb]
loose_file_key_management_filename = /etc/mysql/encryption/keyfile
```

> **Note:**
>
> To avoid startup failures, the loose option prefix can be specified. No error is produced in case the plugin hasn't been installed yet.

Encrypting the Key File

> **Note:**
>
> This step is optional, but highly recommended.

If you use an unencrypted key file, keys are stored in plain text on your system, posing a security risk. It's recommended to encrypt the key file, with these hints in mind:

- MariaDB only supports the Cipher Block Chaining (CBC) mode of Advanced Encryption Standard (AES).
- The encryption key size can be 128-bits, 192-bits, or 256-bits.
- The encryption key is created from the SHA-1 hash of the encryption password.
- The encryption password has a maximum length of 256 characters.

Generate a random encryption password using the openssl rand command. To create a random 256 character encryption password, execute the following:

```
$ sudo openssl rand -hex 128 > /etc/mysql/encryption/keyfile.key
```

Encrypt the key file using the openssl enc command. To encrypt the key file with the encryption password created in the previous step, execute for example one of the following commands:

CURRENT (>= 12.0.1)

```
$ sudo openssl enc -aes-256-cbc -md sha256 -pbkdf2 \
   -pass file:/etc/mysql/encryption/keyfile.key \
   -in /etc/mysql/encryption/keyfile.txt \
   -out /etc/mysql/encryption/keys.enc

$ sudo openssl enc -aes-256-cbc -md sha256 -iter 20000 \
   -pass file:/etc/mysql/encryption/keyfile.key \
   -in /etc/mysql/encryption/keyfile.txt \
   -out /etc/mysql/encryption/keys.enc
```

> **Note:**
>
> Use the `-iter` (iterations) option with `-pbkdf2` to specify more iterations than the OpenSSL default of `10k`.

> **Warning:**
>
> To use `-pbkdf2` effectively, specify the iteration count on the MariaDB Server side. Without this, key decryption will fail. Use the option `--file_key_management_use_pbkdf2=` *`number_of_iterations`* to set it in MariaDB Server.

< 12.0.1

```
$ sudo openssl enc -aes-256-cbc -md sha1 \
   -pass file:/etc/mysql/encryption/keyfile.key \
   -in /etc/mysql/encryption/keyfile.txt \
   -out /etc/mysql/encryption/keys.enc
```

The resulting `keys.enc` file is the encrypted version of `keys.txt` file. Delete the unencrypted key file.

Having the key file encrypted requires both the file_key_management_filename and the file_key_management_filekey system variables to be configured.

The `file_key_management_filekey` variable can be provided in two forms:

- As a plain-text encryption password. This is not recommended, since the plain-text encryption password would be visible in the output of the SHOW VARIABLES statement.
- Prefixed with `FILE:`, it can be a path to a file that contains the plain-text encryption password.

> **Note:**
>
> `FILE:` must be in uppercase in MariaDB. If you use lowercase instead (like OpenSSL does), errors occur (for instance, `mariadbd: Cannot decrypt ../key.enc. Wrong key?`).

These variables can be specified as command-line arguments to `mariadbd`, or they can be specified in a relevant server option group in an option file:

CURRENT (>= 12.0.1)

```
[mariadb]
...
loose_file_key_management_filename = /etc/mysql/encryption/keyfile.enc
loose_file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
loose_file_key_management_digest = sha256
loose_file_key_management_use_pbkdf2 = 20000
```

< 12.0.1

```
[mariadb]
...
loose_file_key_management_filename = /etc/mysql/encryption/keyfile.enc
loose_file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
```

> **Note:**
>
> To avoid startup failures, the loose option prefix can be specified. No error is produced in case the plugin hasn't been installed yet.

Choosing an Encryption Algorithm

The File Key Management plugin currently supports two encryption algorithms for encrypting data: `AES_CBC` and `AES_CTR`. Both of these algorithms use Advanced Encryption Standard (AES) in different modes. AES uses 128-bit blocks, and supports 128-bit, 192-bit, and 256-bit keys. The modes are:

- The `AES_CBC` mode uses AES in the Cipher Block Chaining (CBC) mode.
- The `AES_CTR` mode uses AES in two slightly different modes in different contexts. When encrypting tablespace pages (such as pages in InnoDB, XtraDB, and Aria tables), it uses AES in the Counter (CTR) mode. When encrypting temporary files (where the cipher text is allowed to be larger than the plain text), it uses AES in the authenticated Galois/Counter Mode (GCM).

The recommended algorithm is `AES_CTR`, but this algorithm is only available when MariaDB is built with OpenSSL. If the server is built with wolfSSL then this algorithm is not available. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

Configuring the Encryption Algorithm

The encryption algorithm can be configured by setting the file_key_management_encryption_algorithm system variable.

This system variable can be set to one of the following values:

System Variable Value	Description
AES_CBC	Data is encrypted using AES in the Cipher Block Chaining (CBC) mode. This is the default value.
AES_CTR	Data is encrypted using AES either in the Counter (CTR) mode or in the authenticated Galois/Counter Mode (GCM) mode, depending on context. This is only supported in some builds. See the previous section for more information.

This system variable can be specified as command-line arguments to `mariadbd`, or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
...
loose_file_key_management_encryption_algorithm = AES_CTR
```

Note that the loose option prefix is specified. This option prefix is used in case the plugin hasn't been installed yet.

Note that this variable does not affect the algorithm that MariaDB uses to decrypt the key file. This variable only affects the encryption algorithm that MariaDB uses to encrypt and decrypt data. The only algorithm that MariaDB currently supports to encrypt the key file is Cipher Block Chaining (CBC) mode of Advanced Encryption Standard (AES).

Using the File Key Management Plugin

Once the File Key Management Plugin is enabled, you can use it by creating an encrypted table:

```
CREATE TABLE t (i INT) ENGINE=InnoDB ENCRYPTED=YES
```

Now, table `t` will be encrypted using the encryption key from the key file. For more information on how to use encryption, see Data at Rest Encryption.

> **Note:**
>
> The following Information Schema table is available from MariaDB 11.8.

The Information Schema FILE_KEY_MANAGEMENT_KEYS table provides information about files stored in tablespaces, such as those used by the InnoDB storage engine.

Using Multiple Encryption Keys

The File Key Management Plugin supports using multiple encryption keys. Each encryption key can be defined with a different 32-bit integer as a key identifier.

When encrypting InnoDB tables, the key that is used to encrypt tables can be changed. When encrypting Aria tables, the key that is used to encrypt tables cannot currently be changed.

Key Rotation

> **CURRENT ENTERPRISE SERVER**
>
> The plugin supports key rotation and allows an optional key version in the key file. The keys can be rotated using the `FLUSH FILE_KEY_MANAGEMENT_KEYS` statement, without needing to restart the server. The plugin remains compatible with old key file format, and when version is not specified, it defaults to version 1.

Generation of New Key Versions

How would new key versions be generated?

The plugin doesn't generate any encryption keys itself, and it doesn't have a backend KMS[28] to generate encryption keys for it either.

Any encryption keys need to be generated by the user with external tools, such as OpenSSL. For example:

```
-- Generate a new 256-bit key
openssl rand -hex 32
```

Keys need to be manually saved to the key file. See this section.

The format of the key file is simplistic. It stores encryption keys in a plain-text file.

< 11.8

The File Key Management plugin does not support key rotation.

Versions

Version	Status	Introduced
1.0	Stable	From MariaDB 10.1.18

System Variables

`file_key_management_encryption_algorithm`

- This system variable is used to determine which algorithm the plugin will use to encrypt data.
  - The recommended algorithm is `AES_CTR`, but this algorithm is only available when MariaDB is built with OpenSSL. If the server is built with wolfSSL then this algorithm is not available. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
- Command line: `--file-key-management-encryption-algorithm=` *`value`*
- Scope: Global
- Dynamic: No
- Data Type: `enumerated`
- Default Value: `AES_CBC`
- Valid Values: `AES_CBC`, `AES_CTR`

`file_key_management_digest`

- Specifies the digest function to use in key derivation of the key used for decrypting the key file.
- Command line: `--file-key-management-digest=` *`value`*
- Scope: Global
- Dynamic: No
- Data type: enumerated
- Default value: sha1
- Valid values: sha1, sha224, sha256, sha384, sha512
- Available from MariaDB 12.0.1

`file_key_management_filekey`

- Used to determine the encryption password that is used to decrypt the key file defined by file_key_management_filename.
  - If this system variable's value is prefixed with `FILE:`, then it is interpreted as a path to a file that contains the plain-text encryption password.
  - If this system variable's value is not prefixed with `FILE:`, then it is interpreted as the plain-text encryption password. However, this is not recommended.
  - The encryption password has a max length of 256 characters.
  - The only algorithm that MariaDB currently supports when decrypting the key file is Cipher Block Chaining (CBC) mode of Advanced Encryption Standard (AES). The encryption key size can be 128-bits, 192-bits, or 256-bits. The encryption key is calculated by taking a SHA-1 hash of the encryption password. Instead of SHA-1, SHA-2 can be used as well.
- Command line: `--file-key-management-filekey=` *`value`*
- Scope: Global
- Dynamic: No
- Data type: `string`
- Default value: (empty)

`file_key_management_filename`

- Used to determine the path to the file that contains the encryption keys. If file_key_management_filekey is set, this file can be encrypted with Cipher Block Chaining (CBC) mode of Advanced Encryption Standard (AES).
- Command line: `--file-key-management-filename=` *`value`*
- Scope: Global
- Dynamic: No
- Data type: `string`
- Default value: (empty)

`file_key_management_use_pbkdf2`

- Specifies whether pbkdf2 is used in key derivation, and if so, how many iterations.
- Command line: `--file-key-management-use-pbkdf2=` *`number`*
- Scope: Global
- Dynamic: No
- Data type: `numeric`
- Default value: 0
- Valid values: integers ≥ 0 (reasonable value: 600000 for sha256, less for sha512)
- Available from MariaDB 12.0.1

Options

`file_key_management`

- Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

        - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
    - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--file-key-management=value`
- Data type: `enumerated`
- Default value: `ON`
- Valid values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

AWS Key Management Encryption Plugin

Due to license incompatibilities between the MariaDB server source code and the Amazon AWS C++ SDK, we can only distribute the plugin in source code form, and not as ready-to-use binaries. See Installing the Plugin's Package for details.

MariaDB's data-at-rest encryption requires the use of a key management and encryption plugin. These plugins are responsible both for the management of encryption keys and for the actual encryption and decryption of data.

MariaDB supports the use of multiple encryption keys. Each encryption key uses a 32-bit integer as a key identifier. If the specific plugin supports key rotation, then encryption keys can also be rotated, which creates a new version of the encryption key.

The AWS Key Management plugin is a key management and encryption plugin that uses the Amazon Web Services (AWS) Key Management Service (KMS).

Overview

The AWS Key Management plugin uses the Amazon Web Services (AWS) Key Management Service (KMS) to generate and store AES keys on disk, in encrypted form, using the Customer Master Key (CMK) kept in AWS KMS. When MariaDB Server starts, the plugin will decrypt the encrypted keys, using the AWS KMS "Decrypt" API function. MariaDB data will then be encrypted and decrypted using the AES key. It supports multiple encryption keys. It supports key rotation.

Tutorials

Tutorials related to the AWS Key Management plugin can be found at the following pages:

- Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Setup Guide
- Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Advanced Usage

Preparation

- Before you use the plugin, you need to create a Customer Master Key (CMK). Create a key using the AWS Console as described in the AMS KMS developer guide.
- The easiest way to give the AWS key management plugin access to the key is to create an IAM Role with access to the key, and to apply that IAM Role to an EC2 instance where MariaDB Server runs.
- Make sure that MariaDB Server runs under the correct AWS identity that has access to the above key. For example, you can store the AWS credentials in an AWS credentials file for the user who runs `mysqld`. More information about the credentials file can be found in the AWS CLI Getting Started Guide.

Installing the Plugin's Package

The AWS Key Management plugin depends on the AWS SDK for C++, which uses the Apache License, Version 2.0. This license is not compatible with MariaDB Server's GPL 2.0 license, so we are not able to distribute packages that contain the AWS Key Management plugin. Therefore, the only way to currently obtain the plugin is to install it from source.

Installing from Source

When compiling MariaDB from source, the AWS Key Management plugin is built by default, on systems that support it.

Compilation is controlled by the following `cmake` arguments:

- `-DPLUGIN_AWS_KEY_MANAGEMENT=DYNAMIC` to build a loadable plugin library
- `-DAWS_SDK_EXTERNAL_PROJECT=ON` to download the AWS C++ SDK code
- `-DNOT_FOR_DISTRIBUTION=ON` to confirm that you know to not distribute the resulting binaries

The plugin uses AWS C++ SDK, which introduces the following restrictions:

- The plugin can only be built on Windows, Linux, and macOS.
- The plugin requires that one of the following compilers is used: `gcc` 4.8 or later, `clang` 3.3 or later, Visual Studio 2013 or later.
- On Unix, the `libcurl` development package (e.g. `libcurl3-dev` on Debian Jessie), `uuid` development package, and `openssl` need to be installed.
- You may need to use a newer version of `cmake` than is provided by default in your OS.

You do not need to download/install the AWS C++ SDK yourself; the correct version of the SDK GitHub repository will be cloned into the build directory at build time, and only the libraries for AWS components actually needed by the key management plugin will be built, which takes much less time than building the full AWS C++ SDK.

Building on Linux

With all prerequisites installed, the actual build process pretty much comes down to:

```
# clone the MariaDB Server source code repository
git clone https://github.com/MariaDB/server.git
cd server

# prepare the build
mkdir _build
cd _build
cmake .. -DNOT_FOR_DISTRIBUTION=ON \
  -DPLUGIN_AWS_KEY_MANAGEMENT=DYNAMIC \
  -DAWS_SDK_EXTERNAL_PROJECT=1

# build the plugin only
cd plugin/aws_key_management
make
```

Cmake will print the following warning as part of its output. Please take it seriously and do not distribute the `aws_key_management.so` file to any third parties: **You have linked MariaDB with Apache 2.0 libraries! You may not distribute the resulting binary. If you do, you will put yourself into a legal problem with the Free Software Foundation.**

After `make` succeeded you can copy the created `aws_key_management.so` plugin library file to the plugin directory of your actual MariaDB Server machines installation, e.g. `/usr/lib64/mysql/plugin` on RedHat/Fedora based systems or `/usr/lib/mysql/plugin` on Debian-based systems.

Installing the Plugin

Even after the package that contains the plugin's shared library is installed on the operating system, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN. For example:

```
INSTALL SONAME 'aws_key_management';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
plugin_load_add = aws_key_management
```

Uninstalling the Plugin

Before you uninstall the plugin, you should ensure that data-at-rest encryption is completely disabled, and that MariaDB no longer needs the plugin to decrypt tables or other files.

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN. For example:

```
UNINSTALL SONAME 'aws_key_management';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Configuring the AWS Key Management Plugin

To enable the AWS Key Management plugin, you also need to set the plugin's system variables. The aws_key_management_master_key_id system variable is the primary one to set. These system variables can be specified as command-line arguments to mysqld or they can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
...
aws_key_management_master_key_id=alias/<your key's alias>
```

Once you've updated the configuration file, restart the MariaDB server to apply the changes and make the key management and encryption plugin available for use.

Using the AWS Key Management Plugin

Once the AWS Key Management Plugin is enabled, you can use it by creating an encrypted table:

```
CREATE TABLE t (i INT) ENGINE=InnoDB ENCRYPTED=YES
```

Now, table `t` will be encrypted using the encryption key generated by AWS.

For more information on how to use encryption, see data-at-rest encryption.

Using Multiple Encryption Keys

The AWS Key Management Plugin supports using multiple encryption keys. Each encryption key can be defined with a different 32-bit integer as a key identifier. If a previously unused identifier is used, then the plugin will

automatically generate a new key.

When encrypting InnoDB tables, the key that is used to encrypt tables can be changed.

When encrypting Aria tables, the key that is used to encrypt tables cannot currently be changed.

Key Rotation

The AWS Key Management plugin does support key rotation. To rotate a key, set the aws_key_management_rotate_key system variable. For example, to rotate key with ID 2:

```
SET GLOBAL aws_key_management_rotate_key=2;
```

Or to rotate all keys, set the value to -1:

```
SET GLOBAL aws_key_management_rotate_key=-1;
```

Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.2.6, MariaDB 10.1.24
1.0	Beta	MariaDB 10.1.18
1.0	Experimental	MariaDB 10.1.13

System Variables

`aws_key_management_key_spec`

- Description: Encryption algorithm used to create new keys
- Commandline: `--aws-key-management-key-spec=value`
- Scope: Global
- Dynamic: No
- Data Type: `enumerated`
- Default Value: `AES_128`
- Valid Values: `AES_128`, `AES_256`

`aws_key_management_log_level`

- Description: Dump log of the AWS SDK to MariaDB error log. Permitted values, in increasing verbosity, are Off (default), Fatal, Error, Warn, Info, Debug, and Trace.
- Commandline: `--aws-key-management-log-level=value`
- Scope: Global
- Dynamic: No
- Data Type: `enumerated`
- Default Value: `Off`
- Valid Values: `Off`, `Fatal`, `Warn`, `Info`, `Debug` and `Trace`

`aws_key_management_master_key_id`

- Description: AWS KMS Customer Master Key ID (ARN or alias prefixed by alias/) for the master encryption key. Used to create new data keys. If not set, no new data keys will be created.
- Commandline: `--aws-key-management-master-key-id=value`
- Scope: Global

- Dynamic: No
- Data Type: `string`
- Default Value:

`aws_key_management_mock`

- Description: Mock AWS KMS calls (for testing). Must be enabled at compile-time.
- Commandline: `--aws-key-management-mock`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Valid Values: `OFF`, `ON`

`aws_key_management_region`

- Description: AWS region name, e.g us-east-1 . Default is SDK default, which is us-east-1.
- Commandline: `--aws-key-management-region=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `'us-east-1'`

`aws_key_management_request_timeout`

- Description: Timeout in milliseconds for create HTTPS connection or execute AWS request. Specify 0 to use SDK default.
- Commandline: `--aws-key-management-request-timeout=value`
- Scope: Global
- Dynamic: No
- Data Type: `integer`
- Default Value: 0

`aws_key_management_rotate_key`

- Description: Set this variable to a data key ID to perform rotation of the key to the master key given in `aws_key_management_master_key_id`. Specify -1 to rotate all keys.
- Commandline: `--aws-key-management-rotate-key=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `integer`
- Default Value:

Options

`aws_key_management`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.

        - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
        - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
        - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
    - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.

- Commandline: `--aws-key-management=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Advanced Usage

This document assumes you've already set up an Amazon Web Services (AWS) account, created a master key in the Key Management Service (KMS), and have done the basic work to set up the MariaDB AWS KMS plugin. These steps are all described in Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Setup Guide.

Ultimately, keeping all the credentials required to read the key on a single host means that a user who has gained access to the host has enough information to read the encrypted files in the datadir, read the encrypted keys from the datadir, interact with AWS KMS to decrypt the encrypted keys, and then used the decrypted keys to decrypt the encrypted data.

Theoretically, a superuser can read the memory of the MariaDB server process to read the decrypted keys or restart MariaDB with password authentication disabled in order to dump data, or add new users to MariaDB in order to allow a user to connect and dump the data. Resolving these issues is beyond the scope of this document. A user who gains root access to your operating system or root access to your MariaDB server will have the ability to decrypt your data. Plan accordingly.

Managing AWS Credentials

Putting the AWS credentials in a file inside the MariaDB home directory is not ideal. By default, any user with the FILE privilege can read any files that the MariaDB server has permission to read, which would include the credentials file. To protect against this, you should set `secure_file_priv` to restrict the location the server will allow a user to read from when executing `LOAD DATA INFILE` or the `LOAD_FILE()` function.

But putting them in other locations requires passing additional data to the server, which in the case of CentOS 7 requires customizing the systemd startup procedure. This is most easily done by creating a "drop-in" file in /etc/systemd/system/mariadb.service.d/. The file should end in ".conf" but can otherwise be named whatever you like. After making any changes to systemd files, execute `systemctl daemon-reload` and then start (or restart) the service as usual.

You can place the credentials file in a location of your choosing and then refer to that file by setting the `AWS_CREDENTIAL_PROFILES_FILE` environment variable in the drop-in file:

```
[Service]
Environment=AWS_CREDENTIAL_PROFILES_FILE=/etc/aws-kms-credentials
```

The credentials file will need to be readable by the "mysql" user, but it does not need to be readable by any other user.

AWS credentials can also be put directly into a “drop-in” systemd file that will be read when starting the MariaDB service:

```
# cat /etc/systemd/system/mariadb.service.d/aws-kms.conf
[Service]
Environment=AWS_ACCESS_KEY_ID=AKIAIRSG2XYZATCJLZ4A
Environment=AWS_SECRET_ACCESS_KEY=ux91LZIxCp4ZXabcdefgIViQNtTan42QAmJqJVqV
```

However, any OS user can read this information from systemd, which could be considered a security risk. Another solution is to put the credentials in a separate file that is only readable by root and then refer to that file using an `EnvironmentFile` directive in a drop-in systemd file.

```
# cat /etc/systemd/system/mariadb.service.d/aws-kms.env
AWS_ACCESS_KEY_ID=AKIAIRSG2XYZATCJLZ4A
AWS_SECRET_ACCESS_KEY=ux91LZIxCp4ZXabcdefgIViQNtTan42QAmJqJVqV

# chown root /etc/systemd/system/mariadb.service.d/aws-kms.env
# chmod 600 /etc/systemd/system/mariadb.service.d/aws-kms.env

# cat /etc/systemd/system/mariadb.service.d/aws-kms.conf
[Service]
EnvironmentFile=/etc/systemd/system/mariadb.service.d/aws-kms.env
```

That has the advantage the credentials can only be read directly by root. systemd adds those variables to the environment of the MariaDB server when starting it, and MariaDB can use the credentials to interact with AWS. Note, though, that any process running as the “mysql” user can still read the credentials from the proc filesystem on Linux.

```
$ whoami
mysql
$ cat /proc/$(pgrep mysqld)/environ | tr '\0' '\n' | grep AWS
AWS_ACCESS_KEY_ID=AKIAIRSG2XYZATCJLZ4A
AWS_SECRET_ACCESS_KEY=ux91LZIxCp4ZXabcdefgIViQNtTan42QAmJqJVqV
```

AWS KMS Key Policy

AWS KMS allows flexible, user-editable key policy. This offers fine-grained control over which users can operate on keys. The possibilities range from simply restricting which IP addresses are allowed to perform operations on the key, to requiring a MFA (Multi-Factor Authentication) device to use the key, to enforcing separation of responsibilities by creating an additional user with limited privileges to enable and disable the key. All 3 of these options will be outlined in this section.

For more details about customizing the Key Policy for your master keys, please consult the AWS Key Management Service Key Policy documentation.

Source IP Restrictions

A simple, common-sense restriction to put in place is to restrict the range of IP addresses that are allowed to use your master key. This way, even if someone obtains API credentials, they’ll be unable to use them to decrypt your encryptions keys from a different host.

To restrict API access from only a specific IP address or range of IP addresses, you’ll need to manually edit the key policy.

1. Load the IAM console at.
2. Click “Encryption Keys” in the left-hand sidebar.
3. Click the name of your encryption key to view its details.
4. Click the link labeled “Switch to policy view”, to the right of the heading of the “Key Policy” section.

5. Locate the section that contains `"Sid": "Allow use of the key"`.
6. Add this text below the `"Sid"` line:

```
"Condition": {
 "IpAddress": {
 "aws:SourceIp": [
 "10.1.2.3/32"
 ]
 }
 },
```

… replacing `10.1.2.3/32` above with an IP address or range of IP addresses in CIDR format. For example, a single address would be `192.168.12.34/32`, while a range of addresses might be `192.168.0.0/24`.

1. Click “Save Changes”.
2. Click “Proceed” if prompted with a warning about using the default view in the future.

Access to the API will now be restricted to requests coming from the IP address or range of IP addresses specified in the policy.

Using a Multi-Factor Authentication (MFA) Device

One approach is to modify the key policy for the master key so that MFA (Multi-Factor Authentication) is required in order to use the key. This is achieved with a wrapper that handles prompting the user for an MFA token, acquires temporary, limited-privilege credentials from the AWS Security Token Service (STS), and puts those credentials into the environment of the MariaDB server process. The credentials can expire after as little as 15 minutes.

To require an MFA token for users of the key, you’ll need to manually edit the key policy.

1. Load the IAM console at.
2. Click “Encryption Keys” in the left-hand sidebar.
3. Click the name of your encryption key to view its details.
4. Click the link labeled “Switch to policy view”, to the right of the heading of the “Key Policy” section.
5. Locate the section that contains `"Sid": "Allow use of the key"`.
6. Add this text below the `"Sid"` line:

```
"Condition": {
 "Bool": {
 "aws:MultiFactorAuthPresent": "True"
 }
 },
```

1. Click “Save Changes”.
2. Click “Proceed” if prompted with a warning about using the default view in the future.

Now, add an MFA device for your user. You’ll need to have a hardware MFA device or an application such as Google Authenticator installed on your smartphone.

1. Click “Users” in the left-hand sidebar.
2. Click the name of your user.
3. Click the “Security Credentials” tab.
4. In the “Sign-In Credentials” section, click the “Manage MFA Device” button.
5. Complete the steps to activate your MFA device.
6. Copy the ARN for your MFA device. You will need to use this when configuring the wrapper program.

Now, set up the wrapper program.

1. Copy the iam-kms-wrapper file to /usr/local/bin/, and ensure that it is executable.
2. Create a drop-in systemd config file in `/etc/systemd/system/mariadb.service.d/`:

```
[Service]
EnvironmentFile=/etc/systemd/system/mariadb.service.d/aws-kms.env
ExecStart=
ExecStart=/usr/local/bin/iam-kms-wrapper --config=/etc/my.cnf.d/iam-kms-wrapper.config /usr/sbin/mysqld
$MYSQLD_OPTS
```

1. Execute `systemctl daemon-reload`.
2. Create a file at `/etc/my.cnf.d/iam-kms-wrapper.config` with these contents, using the ARN for your MFA device as the value for `kms_mfa_id`:

```
[kms]
kms_mfa_id = arn:aws:iam::551888187628:mfa/MDBEnc
kms_mfa_socket = /tmp/kms_mfa_socket
```

When you start the MariaDB service now, the wrapper will temporarily create a socket file at the location given by the `kms_mfa_socket` option. The wrapper will read the MFA code from the socket and will use it to authenticate to KMS. To give the MFA code, simply write the digits to the socket file using `echo`: `echo 111676 > /tmp/kms_mfa_socket`.

The `systemctl` command will block until MariaDB starts, so you will need to write the code to the socket file via a separate terminal.

Note that the temporary credentials put into the environment of the MariaDB process will expire after a period of time defined by the request to the AWS Security Token Service (STS). In the example below, they will expire after 900 seconds. After that time, MariaDB may be unable to generate new encrypted data keys, which means that, for example, an attempt to create a table with a previously-unused key ID would fail.

Wrapper Program Example

Here's an example wrapper program written in go. Build this into an executable named `iam-kms-wrapper` and use it as instructed above. This could of course be written in any language for which an appropriate AWS SDK exists, but go has the benefit of compiling to a static binary, which means you do not have to worry about interpreter versions or installing complex dependencies on the host that runs your MariaDB server.

```
package main

import (
    "syscall"
    "os"
    "log"
    "flag"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/aws/awserr"
    "github.com/aws/aws-sdk-go/service/sts"
    "github.com/robfig/config"
)

func main() {

    config_file_p := flag.String("config", "", "location of the config file")
    flag.Parse()

    if flag.NArg() < 1 {
        log.Fatal("Command to wrap must be given as first command-line argument")
    }

    cmd := flag.Arg(0)
    args := flag.Args()[0:]

    conf, err := config.ReadDefault(*config_file_p)
    if err != nil {
        log.Fatal(err)
    }

    kms_mfa_id, err := conf.String("kms","kms_mfa_id")
    mfa_socket_file, err := conf.String("kms","kms_mfa_socket")

    sess := session.New()
    svc := sts.New(sess)

    syscall.Umask(0044)

    log.Printf("Reading MFA token from %s\n",mfa_socket_file)

    if err := syscall.Mknod(mfa_socket_file, syscall.S_IFIFO|uint32(os.FileMode(0622)), 0); err != nil {
        log.Fatal(err)
    }

    file, err := os.Open(mfa_socket_file)
    if err != nil {
        log.Fatal(err)
    }

    token := make([]byte, 6)

    if _, err := file.Read(token); err != nil {
        log.Fatal(err)
    }

    file.Close()

    if err := syscall.Unlink(mfa_socket_file); err != nil {
        log.Fatal(err)
    }

    mfa_token := string(token)

    token_params := &sts.GetSessionTokenInput{
        DurationSeconds: aws.Int64(900),
        SerialNumber:    aws.String(kms_mfa_id),
        TokenCode:       aws.String(mfa_token),
```

```
    }

    resp, err := svc.GetSessionToken(token_params)
    if err != nil {
        if awsErr, ok := err.(awserr.Error); ok {
            // Prints out full error message, including original error if there was one.
            log.Fatal("Error:", awsErr.Error())
        } else {
            log.Fatal("Error:", err.Error())
        }
    }

    creds := resp.Credentials

    os.Setenv("AWS_ACCESS_KEY_ID",*creds.AccessKeyId)
    os.Setenv("AWS_SECRET_ACCESS_KEY",*creds.SecretAccessKey)
    os.Setenv("AWS_SESSION_TOKEN",*creds.SessionToken)

    execErr := syscall.Exec(cmd, args, os.Environ())
    if execErr != nil {
        panic(execErr)
    }
}
```

Disabling Keys When not Needed

Another possibility is to use the API to disable access to the master key and enable it only when a trusted administrator knows that the MariaDB service needs to be started. A specialized tool on a separate host could be used to enable the key for a very short period of time while the service starts and then quickly disable the key.

To do this, you can create an extra IAM User that can only use the kms:EnableKey and kms:DisableKey API endpoints for your key. This user will not be able to encrypt or decrypt any data using the key.

First, create a new user.

1. Load the IAM console at.
2. Click “Users” in the left-hand sidebar.
3. Click “Create New Users”.
4. Enter a new user name. (The examples will use “MDBEncAdmin”.)
5. Click “Show User Security Credentials”.
6. Copy the credentials and put them in a `credentials` file with this structure:

```
[MDBEncAdmin]
aws_access_key_id=AKIAJMPPNO7EBKABCDEF
aws_secret_access_key=pVdGwbuK5/jG64aBK1oEJOXRlkdM0aAylgabCDef
```

1. Click “Close”.
2. Click “Close” again if prompted.
3. Click the name of your new user to open the details view.
4. Copy the “User ARN” value for your user (for example “arn:aws:iam::551888181234:user/MDBEncAdmin”). You will need this for the next step.

Now, give the new user permission to perform API operations on your key.

1. Click “Encryption Keys” in the left-hand sidebar.
2. Click the name of your key to open the details view.
3. Click “Switch to policy view” if it is not already open. (The “policy view” is a large text field that contains JSON describing the key policy.)
4. Create a new item in the `Statement` array with this structure:

```
{
 "Sid": "Allow Enable and Disable of the key",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::551888181234:user/MDBEncAdmin"
 },
 "Action": [
 "kms:EnableKey",
 "kms:DisableKey"
 ]
 },
```

…so that your Key Policy looks like this:

```
{
 "Version": "2012-10-17",
 "Id": "key-consolepolicy-2",
 "Statement": [
 {
 "Sid": "Allow Enable and Disable of the key",
 "Effect": "Allow",
 "Principal": {
 ...
```

1. Click “Save Changes”.

You’ve now added a new IAM user and you’ve given that user privileges to enable and disable your key. This user will be able to perform those operations using the AWS CLI or via a script of your own design using the AWS API. For example, using the AWS CLI:

```
$ cat ~/.aws/credentials
[MDBEncAdmin]
aws_access_key_id=AKIAJMPPNO7EBKABCDEF
aws_secret_access_key=pVdGwbuK5/jG64aBK1oEJOXRlkdM0aAylgabCDef

$ AWS_PROFILE=MDBEncAdmin aws --region us-east-1 kms disable-key --key-id arn:aws:kms:us-east-
1:551888181234:key/abcdf8f6-084b-4cff-99ca-abcdef6c7907c
```

In order for MariaDB to start, this new user will have to enable the master key, then the DBA can start MariaDB, and this user can once again disable the master key after the service has started. Note that in this workflow, MariaDB will be unable to create new encryption keys, such as would be done when a user creates a table that refers to a non-existent key ID. The AWS KMS plugin will encounter an error if it tries to generate a new encryption key while the master key is disabled. In that scenario, the key administrator would have to enable the key before the operation could succeed. Here’s what you should expect to see in the journal if MariaDB tries to interact with a disabled master key:

```
[ERROR] AWS KMS plugin : GenerateDataKeyWithoutPlaintext failed : DisabledException - Unable to parse
ExceptionName: DisabledException Message: arn:aws:kms:us-east-1:551888181234:key/abcdf8f6-084b-4cff-99ca-
abcdef6c7907c is disabled.
```

Adding MFA

It’s also possible to add MFA to this technique so that the user that enables & disables the master key has to authenticate using an MFA device. Adapt the instructions in the MFA section above to add MFA to the policy section for the user with EnableKey and DisableKeys privileges, add an MFA device for that user, use Security Token Service (STS) to get temporary security credentials, and then use those credentials to make the API calls. Here’s an example Python script that follows that workflow:

```
#!/usr/bin/env python

import boto3
import sys

# Command-line argument processing should be more robust than this...
action= sys.argv[1]
mfa_token= sys.argv[2]

# These should perhaps go into a config file instead of here
mfa_serial= 'arn:aws:iam::551888181234:mfa/MDBEncAdmin'
key_id= 'arn:aws:kms:us-east-1:551888181234:key/abcdf8f6-084b-4cff-99ca-abcdef6c7907c'

# Make the connection to the Security Token Service to get temporary credentials
token_client= boto3.client('sts')
token_response= token_client.get_session_token(
    DurationSeconds= 900,
    SerialNumber= mfa_serial,
    TokenCode= mfa_token
)

cred= token_response['Credentials']

# Start new session using temporary, MFA-authenticated credentials
kms_session= boto3.session.Session(
    aws_access_key_id= cred['AccessKeyId'],
    aws_secret_access_key= cred['SecretAccessKey'],
    aws_session_token= cred['SessionToken'],
    region_name= key_id.split(':')[3]
)

# Start KMS client and execute operation against key
kms_client= kms_session.client('kms')
if action == 'enable' or action == 'e':
    action_f= kms_client.enable_key
elif action == 'disable' or action == 'd':
    action_f= kms_client.disable_key
else:
    raise Exception('Action must be either "disable" or "enable"')

action_f(KeyId=key_id)
```

```
$ AWS_PROFILE=MDBEncAdmin python kms-manage-key disable 575290
$ AWS_PROFILE=MDBEncAdmin python kms-manage-key enable 799870
```

Logging and Auditing

CloudTrail

Amazon's CloudTrail service creates JSON-formatted text log files for every API interaction. Enabling CloudTrail requires S3, which incurs additional fees.

First, enable CloudTrail and add a trail.

1. Load the CloudTrail console at.
2. If you've never used CloudTrail before, click "Get Started Now".
3. Enter a value for "trail name". This example uses "mariadb-encryption-key".
4. Create a new S3 bucket, using a globally unique name, or use an existing S3 bucket, according to your needs.
5. Click "Turn On".

If you navigate to the S3 bucket you created, you should find log files that contain JSON-formatted descriptions of your API interactions.

CloudWatch

Amazon's CloudWatch service allows you to create alarms and event rules that monitor log information.

First, send your CloudTrail logs to CloudWatch.

1. Load the CloudTrail console at.
2. Click "Trails" in the left-hand navigation sidebar.
3. Click the name of your trail to open the Configuration view.
4. In the "CloudWatch Logs" section, click "Configure".
5. Click "Continue".
6. Click "Allow".

Now, set up an SNS topic to facilitate email notifications.

1. Open.
2. Make sure the region in the console (look in the upper-right corner) is the same as the region where you created your key!
3. Click "Get Started" is prompted.
4. Click "Events" in the left-hand sidebar.
5. Click "Create new topic".
6. Enter a Topic name of your choosing.
7. Enter a Display name of your choosing.
8. Click "Create topic".
9. Click the ARN of your new SNS topic.
10. Click "Create Subscription".
11. Select "Email" from the Protocol dropdown.
12. Enter the desired notification email address in the Endpoint field.
13. Wait for the confirmation email to show up and follow the instructions.

Now, configure CloudWatch and create an Event Rule.

1. Open.
2. Make sure the region in the console (look in the upper-right corner) is the same as the region where you created your key and your SNS topic!
3. Click "Events" in the left-hand sidebar.
4. Click "Create rule".
5. Choose "AWS API call" from the "Select event source" dropdown.
6. Choose "KMS" from the "Service name" dropdown.
7. Decide which operations should trigger the event. (You can eep "Any operation" selected for simplicity.)
8. Click "Add target".
9. Select "SNS target" from the dropdown.
10. Select the SNS topic you created in the previous steps.
11. Click "Configure details".
12. Enter a Name and Description of your choosing.
13. Click "Create rule".

You should now get emails any time someone executes API calls on the KMS service in the region where you've created the CloudWatch Event rule. That means you should get email any time the key is enabled or disabled, and any time the AWS KMS plugin generates new keys or decrypts the keys stored on disk on the MariaDB server.

You may also wish to create an event rule (or an additional event) that matches only when an unauthorized user tries to access the key. You might accomplish that by manually editing the Event selector of the rule to look something like this:

```
{
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "kms.amazonaws.com"
    ],
    "errorCode": [
      "AccessDenied",
      "UnauthorizedOperation"
    ]
  }
}
```

The emails are formatted as JSON. Further customization of the CloudWatch email workflow is beyond the scope of this document.

There are many other workflows available using CloudWatch, including workflows with alarms and dashboards. Those are beyond the scope of this document.

This page is licensed: CC BY-SA / Gnu FDL

Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Setup Guide

Overview

MariaDB contains a robust, full instance, at-rest encryption. This feature uses a flexible plugin interface to allow actual encryption to be done using a key management approach that meets the customer's needs. MariaDB Server includes a plugin that uses the Amazon Web Services (AWS) Key Management Service (KMS) to facilitate separation of responsibilities and remote logging & auditing of key access requests.

Rather than storing the encryption key in a local file, this plugin keeps the master key in AWS KMS. When you first start MariaDB, the AWS KMS plugin will connect to the AWS Key Management Service and ask it to generate a new key. MariaDB will store that key on-disk in an encrypted form. The key stored on-disk cannot be used to decrypt the data; rather, on each startup, MariaDB connects to AWS KMS and has the service decrypt the locally-stored key(s). The decrypted key is stored in-memory as long as the MariaDB server process is running, and that in-memory decrypted key is used to encrypt the local data.

This guide is based on CentOS 7, using systemd with SELinux enabled. Some steps will differ if you use other operating systems or configurations.

Installing the Plugin's Package

The AWS Key Management plugin depends on the AWS SDK for C++, which uses the Apache License, Version 2.0. This license is not compatible with MariaDB Server's GPL 2.0 license, so we are not able to distribute packages that contain the AWS Key Management plugin. Therefore, the only way to currently obtain the plugin is to install it from source.

Installing from Source

When compiling MariaDB from source, the AWS Key Management plugin is built by default on systems that support it.

Compilation is controlled by the `-DPLUGIN_AWS_KEY_MANAGEMENT=DYNAMIC -DAWS_SDK_EXTERNAL_PROJECT=1` cmake arguments.

The plugin uses AWS C++ SDK, which introduces the following restrictions:

- The plugin can only be built on Windows, Linux, and macOS.
- The plugin requires that one of the following compilers is used: `gcc` 4.8 or later, `clang` 3.3 or later, or Visual Studio 2013 or later.
- On Unix, the `libcurl` development package (e.g., `libcurl3-dev` on Debian Jessie), the `uuid` development package, and `openssl` need to be installed.
- You may need to use a newer version of `cmake` than is provided by default in your OS.

Installing the Plugin

Even after the package that contains the plugin's shared library is installed on the operating system, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN. For example:

```
INSTALL SONAME 'aws_key_management';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld, or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
...
plugin_load_add = aws_key_management
```

Sign up for Amazon Web Services

If you already have an AWS account, you can skip this section.

1. Load.
2. Click "Create a Free Account" and complete the steps.
3. You'll need to enter credit card information. Charges related only to your use of the AWS KMS service should be limited to about $1/month for the single master key we will create. If you use other services, additional charges may apply. Consult AWS Cloud Pricing Principles for more information about the pricing of AWS services.
4. You'll need to complete the AWS identity verification process.

Create an IAM User and/or Role

After creating an account or logging in to an existing account, follow these steps to create an IAM User or Role with restricted privileges that will use (but not administer) your master encryption key.

If you intend to run MariaDB Server on an EC2 instance, you should create a Role (or modify an existing Role already attached to your instance). If you intend to run MariaDB Server outside of AWS, you may want to create a User.

Creating an IAM Role

1. Load the Identity and Access Management Console at.
2. Click "Roles" in the left-hand sidebar
3. Click "Create new role"
4. Select "AWS Service Role"

5. Click the “Select” button next to “Amazon EC2 / Allows EC2 instances to call AWS services on your behalf.”
6. Do not select any policies on the “Attach Policy” screen. Click “Next Step”
7. Click “Next Step”
8. Give your Role a “Role name”
9. Click “Create role”

Creating an IAM User

1. Load the Identity and Access Management Console at.
2. Click “Users” in the left-hand sidebar.
3. Click the “Create New Users” button
4. Enter a single User Name of your choosing. We’ll use “MDBEnc” for this demonstration. Keep the “Generate an access key for each user” box checked.
5. Click “Create”.
6. Click “Show User Security Credentials”.
7. Copy the Access Key ID and Secret Access Key. Optionally, you can click “Download Credentials”. We will need these in order for local programs to interact with AWS using its API.
8. Create a file on your computer to hold the credentials for this user. We’ll use this file later. It should have this structure:

```
[default]
aws_access_key_id = AKIAIG6IZ6TKF52FVV5A
aws_secret_access_key = o7CEf7KhZfsVF9cS0a2roqqZNmuzXtIR869zpSBT
```

1. Click “Close”. If prompted because you did not Download Credentials, ensure that you’ve saved them somewhere, and click “Close”.

Create a Master Encryption Key

Now, we’ll create a master encryption key. This key can *never* be retrieved by *any* application or user. This key is used remotely to encrypt (and decrypt) the actual encryption keys that will be used by MariaDB. If this key is deleted or you lose access to it, you will be unable to use the contents of your MariaDB data directory.

1. Click “Encryption Keys” in the left-hand sidebar.
2. Click the “Get Started Now” button.
3. Use the “Filter” dropdown to choose the region where you’d like to create your master key.
4. Click the “Create Key” button.
5. Enter an Alias and Description of your choosing.
6. Click “Next Step”.
7. Do not check the box to make your IAM Role or IAM User a Key Administrator.
8. Click “Next Step” again.
9. Check the boxes to give your IAM Role and/or IAM User permissions to use this key.
10. Click “Next Step”.
11. Click “Finish”.

You should now see your key listed in the console:

You’ll use the “Alias” you provided when you configure MariaDB later.

We now have a Customer Master Key and an IAM user that has privileges to access it using access credentials. This is enough to begin using the AWS KMS plugin.

Configure AWS Credentials

There are a number of ways to give the IAM credentials to the AWS KMS plugin. The plugin supports reading credentials from all standard locations used across the various AWS API clients.

The easiest approach is to run MariaDB Server in an EC2 instance that has an IAM Role with User access to the CMK you wish to use. You can give key access privileges to a Role already attached to your EC2 instance, or you can create a new IAM Role and attach it to an already-running EC2 instance. If you've done that, no further credentials management is required, and you do not need to create a `credentials` file.

If you're not running MariaDB Server on an EC2 instance, you can also place the credentials in the MariaDB data directory. The AWS API client looks for a `credentials` file in the `.aws` subdirectory of the home directory of the user running the client process. In the case of MariaDB, its home directory is its `datadir`.

1. Create a `credentials` file that MariaDB can read. Use the region you selected when creating the key. Master keys cannot be used across regions. For example:

```
$ cat /var/lib/mysql/.aws/credentials
[default]
aws_access_key_id = AKIAIG6IZ6TKF52FVV5A
aws_secret_access_key = o7CEf7KhZfsVF9cS0a2roqqZNmuzXtIR869zpSBT
region = us-east-1
```

1. Change the permissions of the file so that it is owned by, and can only be read by, the `mysql` user:

```
chown mysql /var/lib/mysql/.aws/credentials
chmod 600 /var/lib/mysql/.aws/credentials
```

Configure MariaDB

1. Create a new option file to tell MariaDB to enable encryption functionality and to use the AWS KMS plugin. Create a new file under `/etc/my.cnf.d/` (or wherever your OS may have you create such files) with contents like this:

```
[mariadb]
plugin_load_add = aws_key_management
aws-key-management = FORCE_PLUS_PERMANENT
aws-key-management-master-key-id = alias/mariadb-encryption
aws-key-management-region = us-east-1
!include /etc/my.cnf.d/enable_encryption.preset
```

1. Append the "Alias" value you copied above to `alias/` to use as the value for the `aws-key-management-master-key-id` option.

Note that you **must** include `aws-key-management-region` in your .cnf file if you are not using the us-east-1 region.

Now, you have told MariaDB to use the AWS KMS plugin, and you've put credentials for the plugin in a location where the plugin will find them. The /etc/my.cnf.d/enable_encryption.preset file contains a set of options that enable all available encryption functionality.

When you start MariaDB, the AWS KMS plugin will connect to the AWS Key Management Service and ask it to generate a new key. MariaDB will store that key on-disk in an encrypted form. The key stored on-disk cannot be used to decrypt the data; rather, on each startup, MariaDB must connect to AWS KMS and have the service decrypt the locally-stored key. The decrypted version is stored in-memory as long as the MariaDB server process is running, and that in-memory decrypted key is used to encrypt the local data.

SELinux and Outbound Connections from MariaDB

Because MariaDB needs to connect to the AWS KMS service, you must ensure that the host has outbound network connectivity over port 443 to AWS, and you must ensure that local policies allow the MariaDB server process to make those outbound connections. By default, SELinux restricts MariaDB from making such connections.

The most simple way to cause SELinux to allow outbound HTTPS connections from MariaDB is to enable to mysql_connect_any boolean, like this:

```
setsebool -P mysql_connect_any 1
```

There are more complex alternatives that have a more granular effect, but those are beyond the scope of this document.

Start MariaDB

Start MariaDB using the `systemctl` tool:

```
systemctl start mariadb
```

If you do not use systemd, you may have to start MariaDB using some other mechanism.

You should see journal output similar to this:

```
# journalctl --no-pager -o cat -u mariadb.service

[Note] /usr/sbin/mysqld (mysqld 10.1.9-MariaDB-enterprise-log) starting as process 19831 ...
[Note] AWS KMS plugin: generated encrypted datakey for key id=1, version=1
[Note] AWS KMS plugin: loaded key 1, version 1, key length 128 bit
[Note] InnoDB: Using mutexes to ref count buffer pool pages
[Note] InnoDB: The InnoDB memory heap is disabled
[Note] InnoDB: Mutexes and rw_locks use GCC atomic builtins
[Note] InnoDB: Memory barrier is not used
[Note] InnoDB: Compressed tables use zlib 1.2.7
[Note] InnoDB: Using CPU crc32 instructions
[Note] InnoDB: Initializing buffer pool, size = 2.0G
[Note] InnoDB: Completed initialization of buffer pool
[Note] InnoDB: Highest supported file format is Barracuda.
[Note] InnoDB: 128 rollback segment(s) are active.
[Note] InnoDB: Waiting for purge to start
[Note] InnoDB:  Percona XtraDB (http://www.percona.com) 5.6.26-74.0 started; log sequence number 1616819
[Note] InnoDB: Dumping buffer pool(s) not yet started
[Note] Plugin 'FEEDBACK' is disabled.
[Note] AWS KMS plugin: generated encrypted datakey for key id=2, version=1
[Note] AWS KMS plugin: loaded key 2, version 1, key length 128 bit
[Note] Using encryption key id 2 for temporary files
[Note] Server socket created on IP: '::'.
[Note] Reading of all Master_info entries succeeded
[Note] Added new Master_info '' to hash table
[Note] /usr/sbin/mysqld: ready for connections.
```

Note the several lines of output that refer explicitly to the “AWS KMS plugin”. You can see that the plugin generates a “datakey”, loads that data key, and then later generates and loads a second data key. The 2nd data key is used to encrypt temporary files and temporary tables.

You can see the encrypted keys stored on-disk in the datadir:

```
# ls -l /var/lib/mysql/aws*
-rw-rw----. 1 mysql mysql 188 Feb 25 18:55 /var/lib/mysql/aws-kms-key.1.1
-rw-rw----. 1 mysql mysql 188 Feb 25 18:55 /var/lib/mysql/aws-kms-key.2.1
```

Note that those keys are not useful alone. They are encrypted. When MariaDB starts up, the AWS KMS plugin decrypts those keys by interacting with AWS KMS.

For maximum security, you should start from an empty datadir and run mariadb-install-db after configuring encryption. Then you should re-import your data so that it is fully encrypted. Use `sudo` to run `mariadb-install-db` so that it finds your credentials file:

```
# sudo -u mysql mariadb-install-db
Installing MariaDB/MySQL system tables in '/var/lib/mysql' ...
2016-02-25 23:16:06 139731553998976 [Note] /usr/sbin/mysqld (mysqld 10.1.11-MariaDB-enterprise-log)
starting as process 39551 ...
2016-02-25 23:16:07 139731553998976 [Note] AWS KMS plugin: generated encrypted datakey for key id=1,
version=1
2016-02-25 23:16:07 139731553998976 [Note] AWS KMS plugin: loaded key 1, version 1, key length 128 bit
...
```

Create Encrypted Tables

With `innodb-encrypt-tables=ON`, new InnoDB tables will be encrypted by default, using the key ID set in `innodb_default_encryption_key_id` (default 1). With `innodb-encrypt-tables=FORCE` enabled, it is not possible to manually bypass encryption when creating a table.

You can cause the AWS KMS plugin to create new encryption keys at-will by specifying a new ENCRYPTION_KEY_ID when creating a table:

```
MariaDB [test]> CREATE TABLE t1 (id serial, v VARCHAR(32)) ENCRYPTION_KEY_ID=3;
Query OK, 0 rows affected (0.91 sec)
```

```
[Note] AWS KMS plugin: generated encrypted datakey for key id=3, version=1
[Note] AWS KMS plugin: loaded key 3, version 1, key length 128 bit
```

```
# ls -l /var/lib/mysql/aws*
-rw-rw----. 1 mysql mysql 188 Feb 25 18:55 /var/lib/mysql/aws-kms-key.1.1
-rw-rw----. 1 mysql mysql 188 Feb 25 18:55 /var/lib/mysql/aws-kms-key.2.1
-rw-rw----. 1 mysql mysql 188 Feb 25 19:10 /var/lib/mysql/aws-kms-key.3.1
```

Read more about encrypting data in the Data at Rest Encryption section of the MariaDB Documentation.

AWS KMS Plugin Option Reference

- `aws_key_management_master_key_id`: AWS KMS Customer Master Key ID (ARN or alias prefixed by `alias/`) for master encryption key. Used to create new data keys. If not set, no new data keys will be created.
- `aws_key_management_rotate_key`: Set this variable to a data key ID to perform rotation of the key to the master key given in `aws_key_management_master_key_id`. Specify -1 to rotate all keys.
- `aws_key_management_key_spec`: Encryption algorithm used to create new keys. Allowed values are AES_128 (default) or AES_256.
- `aws_key_management_log_level`: Logging for AWS API. Allowed values, in increasing verbosity, are “Off” (default), “Fatal”, “Error”, “Warn”, “Info”, “Debug”, and “Trace”.

Next Steps

For more information about advanced usage, including strategies to manage credentials, enforce separation of responsibilities, and even require 2-factor authentication to start your MariaDB server, please review Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Advanced Usage.

This page is licensed: CC BY-SA / Gnu FDL

Hashicorp Key Management Plugin

> **Note:**
>
> **Key Rotation and Cache Flushing**
>
> As of MariaDB 12.3, you can manually rotate keys and flush the cache without restarting the server. See Key Rotation and Cache Flushing for details.

> **Note:**
>
> The configured Vault token requires specific read-only permissions. See Required Vault Token Permissions for details to prevent authorization failures during plugin initialization.

The Hashicorp Key Management Plugin is used to implement encryption using keys stored in the Hashicorp Vault KMS. For more information, see Hashicorp Vault and MariaDB, and for how to install Vault, see Install Vault, as well as MySQL/MariaDB Database Secrets Engine.

The current version of this plugin implements the following features:

- Authentication is done using Hashicorp Vault's token authentication method.
- If additional client authentication is required, then the path to the CA authentication bundle file may be passed as a plugin parameter;
- The creation of the keys and their management is carried out using the Hashicorp Vault KMS and its tools.
- The plugin uses libcurl (https) as an interface to the HashiCorp Vault server.
- JSON parsing is performed through the JSON service (through the include/mysql/service_json.h).
- HashiCorp Vault 1.2.4 was used for development and testing.
- As of MariaDB 10.6.24, the plugin is configured to use cached keys for all communication errors, not just for timeouts. This ensures continuous operation when the Vault server is temporarily unreachable.
- As of MariaDB 10.6.24, the default setting for cache usage on error is `ON`.

Since we require support for key versioning, the key-value storage must be configured in Hashicorp Vault as a key-value storage that uses the interface of the second version. For example, you can create it as follows:

```
~$ vault secrets enable -path /test -version=2 kv
```

Key names must correspond to their numerical identifiers. Key identifiers themselves, their possible values, and rules of use are described in more detail in the MariaDB main documentation.

From the point of view of the key-value storage (in terms of Hashicorp Vault), the key is a secret containing one key-value pair with the name "data" and a value representing a binary string containing the key value, for example:

```
~$ vault kv get /test/1

====== Metadata ======
Key              Value
---              -----
created_time     2019-12-14T14:19:19.42432951Z
deletion_time    n/a
destroyed        false
version          1

==== Data ====
Key     Value
---     -----
data    0123456789ABCDEF0123456789ABCDEF
```

Key values are strings containing binary data. MariaDB currently uses the AES algorithm with 256-bit keys as the default encryption method. In this case, the keys that will be stored in the Hashicorp Vault should be 32-byte strings. Most likely, you will use some utilities for creating and administering keys designed to work with Hashicorp Vault. But in the simplest case, keys can be created from the command line through the vault utility, for example, as follows:

```
~$ vault kv put /test/1 data="0123456789ABCDEF0123456789ABCDEF"
```

If you use default encryption (AES), you should ensure that the key length is 32 bytes, as it may fail to use InnoDB as a data storage.

The plugin currently does not unseal Hashicorp Vault on its own; you must do this in advance and on your own.

To use Hashicorp Vault KMS, the plugin must be preloaded and activated on the server. Most of its parameters should not be changed during plugin operation and therefore must be preconfigured as part of the server configuration through configuration file or command line options:

```
--plugin-load-add=hashicorp_key_management.so
--loose-hashicorp-key-management
--loose-hashicorp-key-management-vault-url="$VAULT_ADDR/v1/test"
--loose-hashicorp-key-management-token="$VAULT_TOKEN"
```

Options

The plugin supports the following parameters, which must be set in advance and cannot be changed during server operation:

`hashicorp-key-management-vault-url`

- Description: HTTP[s] URL that is used to connect to the Hashicorp Vault server. It must include the name of the scheme (`https://` for a secure connection) and, according to the API rules for storage of the key-value type in Hashicorp Vault, after the server address, the path must begin with the “/v1/” string (as prefix), for example: `https://127.0.0.1:8200/v1/my_secrets`. By default, the path is not set; therefore, you must replace it with the correct path to your secrets.
- Command line: `--[loose-]hashicorp-key-management-vault-url="<url>"`

`hashicorp-key-management-token`

- Description: An Authentication token that is passed to the Hashicorp Vault in the request header. By default, this parameter contains an empty string, so you must specify the correct value for it, otherwise the Hashicorp Vault server will refuse authorization. Alternatively, you can define an environment variable `VAULT_TOKEN` and store the token there.
- Command line: `--[loose-]hashicorp-key-management-token="<token>"`

`hashicorp-key-management-vault-ca`

- Description: Path to the Certificate Authority (CA) bundle (is a file that contains root and intermediate certificates). By default, this parameter contains an empty string, which means no CA bundle.
- Command line: `--[loose-]hashicorp-key-management-vault-ca="<path>"`

`hashicorp-key-management-timeout`

- Description: Set the duration (in seconds) for the Hashicorp Vault server connection timeout. The default value is 15 seconds. The allowed range is from 1 to 86400 seconds. The user can also specify a zero value, which means the default timeout value set by the libcurl library (currently 300 seconds).
- Command line: `--[loose-]hashicorp-key-management-timeout=<timeout>`

`hashicorp-key-management-max-retries`

- Description: Number of server request retries in case of timeout. Default is three retries.
- Command line: `----[loose-]hashicorp-key-management-max-retries=<retries>`

`hashicorp-key-management-caching-enabled`

- Description: Enable key caching (storing key values received from the Hashicorp Vault server in the local memory). By default, caching is enabled.
- Command line: `--[loose-]hashicorp-key-management-caching-enabled="on"|"off"`

`hashicorp-key-management-use-cache-on-timeout`

- Description: This parameter instructs the plugin to use the key values or version numbers taken from the cache in the event of a timeout when accessing the vault server. By default, this option is disabled. Please note that key values or version numbers will be read from the cache when the timeout expires, only after the number of attempts to read them from the storage server that specified by the –[loose-]hashicorp-key-management-max-retries parameter has been exhausted.
- Command line: `--[loose-]hashicorp-key-management-use-cache-on-timeout="on"|"off"`
- Deprecated in MariaDB 10.11.16

`hashicorp-key-management-cache-timeout`

- Description: The time (in milliseconds) after which the value of the key stored in the cache becomes invalid and an attempt to read this data causes a new request to be sent to the vault server. By default, cache entries become invalid after 60,000 milliseconds (after one minute). If the value of this parameter is zero, then the keys will always be considered invalid, but they still can be used if the vault server is unavailable and the corresponding cache operating mode (`--[loose-]hashicorp-key-management-use-cache-on-timeout="on"`) is enabled.
- As of MariaDB 10.6.24, the default value is 1 year (specified in milliseconds).
- Command line: `--[loose-]hashicorp-key-management-cache-timeout=<timeout>`
- Deprecated in MariaDB 10.11.16

`hashicorp-key-management-cache-version-timeout`

- Description: The time (in milliseconds) after which the information about the latest version number of the key (which is stored in the cache) becomes invalid and an attempt to read this information causes a new request to be sent to the vault server. If the value of this parameter is zero, then information about the latest key version numbers is always considered invalid, unless there is no communication with the vault server, and use of the cache is allowed when the server is unavailable. By default, this parameter is zero, that is, the latest version numbers for the keys stored in the cache are considered always invalid, except when the vault server is unavailable and use of the cache is allowed on server failures.
- Command line: `--[loose-]hashicorp-key-management-cache-version-timeout=<timeout>`

`hashicorp-key-management-check-kv-version`

- Description: This parameter enables (“on”, this is the default value) or disables (“off”) checking the kv storage version during plugin initialization. The plugin requires storage version 2 or later in order for it to work properly.
- When this option is enabled, the configured Vault token must also have read access to the `sys/mounts/my_vault/tune` endpoint, allowing the plugin to determine the kv storage version. See Required Vault Token Permissions for details.
- Command line: `--[loose-]hashicorp-key-management-check-kv-version="on"|"off"`

Required Vault Token Permissions

The token provided through `hashicorp-key-management-token` must have the following Vault access privileges.

Given a `hashicorp-key-management-vault-url` of `http://vault-server/v1/my_vault`, the token requires:

Value Path	Access Required	Condition
`my_vault/data`	read	Always required
`sys/mounts/my_vault/tune`	read	Required unless `hashicorp-key-management-check-kv-version` is set to `off`

Note: During plugin initialization, the kv storage version is verified using the `sys/mounts/my_vault/tune` path. Most configurations require this permission because `hashicorp-key-management-check-kv-version` is set to `on` by default.

Example Minimal Vault Policy for Read-Only Access

The following example provides a minimal Vault policy that grants the required privileges:

```
path "my_vault/data/*" {
  capabilities = ["read"]
}

path "sys/mounts/my_vault/tune" {
  capabilities = ["read"]
}
```

Replace `my_vault` with the path segment of your `hashicorp-key-management-vault-url` configuration.

This ensures that the plugin does not require additional permissions to access encryption keys or verify the kv storage version.

Key Rotation and Cache Flushing

> **Note:**
>
> This functionality is available from MariaDB 12.3.

The HashiCorp Key Management plugin supports key versioning provided by the HashiCorp Vault Server. In previous versions, rotating keys required a server restart to clear the internal cache. As of MariaDB 12.3, you can flush the plugin cache manually while the server is running.

Flushing the Cache

To rotate keys, you must flush the cached keys using the `FLUSH` command. This clears the local cache, forcing the server to re-fetch the latest key versions from the HashiCorp Vault server upon the next access.

Executing this command requires the `RELOAD` privilege.

```
FLUSH HASHICORP_KEY_MANAGEMENT_CACHE;
```

Verifying Key Versions

To view the current Key IDs and Key Versions stored in the latest version cache, you can query the Information Schema table or use the `SHOW` command.

See Information Schema HASHICORP_KEY_MANAGEMENT_CACHE for table details.

Using the SHOW command:

```
SHOW HASHICORP_KEY_MANAGEMENT_CACHE;
```

See Also

- HashiCorp Vault and MariaDB

This page is licensed: CC BY-SA / Gnu FDL

Uninstall Key Management Plugins

Once all data and logs have been decrypted, you can safely remove key management plugins, if desired. For example, if using the file key management plugin:

```
# Comment out or remove the following lines
plugin_load_add = file_key_management
file_key_management_filename = /etc/mysql/encryption/keyfile
```

Restart the server after editing the configuration.

- Ensure that all tables no longer report encryption in `CREATE_OPTIONS`.
- Confirm that redo logs and binary logs are unencrypted by checking server variables and log headers.
- Confirm that no key management plugin is loaded:

```
SHOW PLUGINS;
```

This page is licensed: CC BY-SA / Gnu FDL

User Account Management

See Roles.

Manage roles in MariaDB Server for streamlined user access control. This section explains how to create, assign, and manage roles to simplify privilege management and enhance security.

See Account Locking.

Explains how to lock and unlock user accounts using CREATE USER and ALTER USER statements to prevent login access without deleting the account.

See Authentication.

Details the authentication changes introduced in MariaDB 10.4, including multiple authentication plugins per user, the mysql.global_priv table, and the default unix_socket authentication for root.

See Incrementing of the access_denied_errors Status Variable.

Describes the conditions that trigger the access_denied_errors status variable, such as failed logins, invalid privileges, or missing SSL requirements, aiding in security monitoring.

See User Password Expiry.

Guide to configuring password expiration policies, including setting global lifetimes via default_password_lifetime or per-user limits, and handling expired password connections.

See Catalogs.

Introduction to Catalogs, a multi-tenancy feature for isolating database objects and users, planned for future MariaDB releases.

This page is licensed: CC BY-SA / Gnu FDL

Roles

See Roles Overview.

Overview of Role-Based Access Control (RBAC) in MariaDB, which simplifies privilege management by allowing permissions to be grouped into roles and assigned to users.

See System Users, Roles, Privileges.

Details default system users like mariadb.sys and the PUBLIC role, including their creation, purpose, and special management behaviors.

See CREATE ROLE.

Create new roles to simplify privilege management. Learn how to define a role that can be assigned to multiple users or other roles.

See DROP ROLE.

Remove a role from the system. Learn the syntax to delete defined roles and revoke them from any users or roles that currently hold them.

See CURRENT_ROLE.

Returns the current role. The current role can be set with SET ROLE or SET DEFAULT ROLE.

See SET ROLE.

Sets the current role for the session. Learn how to enable none, or a specific role to change your current privileges dynamically.

See SET DEFAULT ROLE.

Define the default role for a user. Learn how to configure which role is automatically active when a user connects to the server.

See GRANT.

Complete privilege management guide for MariaDB. Complete GRANT syntax for database, table, and column permissions with roles with comprehensive examples and.

See REVOKE.

Remove privileges or roles. Learn how to withdraw previously granted permissions from users or roles to restrict access and secure the database.

See mysql.roles_mapping Table.

The mysql.roles_mapping table manages role assignments, linking user accounts to the roles they have been granted.

See Information Schema APPLICABLE_ROLES Table.

The Information Schema APPLICABLE_ROLES table shows the role authorizations that the current user may use, detailing grantable and default status.

See Information Schema ENABLED_ROLES Table.

The Information Schema ENABLED_ROLES table lists all roles that are currently enabled for the current session, including nested roles.

This page is licensed: CC BY-SA / Gnu FDL

Roles Overview

Description

A role bundles a number of privileges together. It assists larger organizations where, typically, a number of users would have the same privileges, and, previously, the only way to change the privileges for a group of users was by changing each user's privileges individually.

Alternatively, multiple external users could have been assigned the same user, and there would have been no way to see which actual user was responsible for which action.

With roles, managing this is easy. For example, there could be a number of users assigned to a journalist role, with identical privileges. Changing the privileges for all the journalists is a matter of simply changing the role's privileges, while the individual user is still linked with any changes that take place.

Roles are created with the CREATE ROLE statement, and dropped with the DROP ROLE statement. Roles are then assigned to a user with an extension to the GRANT statement, while privileges are assigned to a role in the regular way with GRANT. Similarly, the REVOKE statement can be used to both revoke a role from a user, or revoke a privilege from a role.

Once a user has connected, he can obtain all privileges associated with a role by **setting** a role with the SET ROLE statement. The CURRENT_ROLE function returns the currently set role for the session, if any.

Only roles granted directly to a user can be set, roles granted to other roles cannot. Instead the privileges granted to a role, which is, in turn, granted to another role (grantee), will be immediately available to any user who sets this second grantee role.

The SET DEFAULT ROLE statement allows one to set a default role for a user. A default role is automatically enabled when a user connects (an implicit SET ROLE statement is executed immediately after a connection is established).

Roles were implemented as a GSoC 2013 project by Vicentiu Ciorbaru.

System Tables

Information about roles and who they've been granted to can be found in the Information Schema APPLICABLE_ROLES table as well as the mysql.ROLES_MAPPING table.

The Information Schema ENABLED_ROLES table shows the enabled roles for the current session.

Examples

Creating a role and granting a privilege:

```
CREATE ROLE journalist;

GRANT SHOW DATABASES ON *.* TO journalist;

GRANT journalist TO hulda;
```

Note, that hulda has no `SHOW DATABASES` privilege, even though she was granted the journalist role. She needs to **set** the role first:

```
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+

SELECT CURRENT_ROLE;
+--------------+
| CURRENT_ROLE |
+--------------+
| NULL         |
+--------------+

SET ROLE journalist;

SELECT CURRENT_ROLE;
+--------------+
| CURRENT_ROLE |
+--------------+
| journalist   |
+--------------+

SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| ...                |
| information_schema |
| mysql              |
| performance_schema |
| test               |
| ...                |
+--------------------+

SET ROLE NONE;
```

Roles can be granted to roles:

```
CREATE ROLE writer;

GRANT SELECT ON data.* TO writer;

GRANT writer TO journalist;
```

But one does not need to set a role granted to a role. For example, hulda will automatically get all writer privileges when she sets the journalist role:

```
SELECT CURRENT_ROLE;
+--------------+
| CURRENT_ROLE |
+--------------+
| NULL         |
+--------------+

SHOW TABLES FROM data;
Empty set (0.01 sec)

SET ROLE journalist;

SELECT CURRENT_ROLE;
+--------------+
| CURRENT_ROLE |
+--------------+
| journalist   |
+--------------+

SHOW TABLES FROM data;
+------------------------------+
| Tables_in_data               |
+------------------------------+
| set1                         |
| ...                          |
+------------------------------+
```

Roles and Views (and Stored Routines)

When a user sets a role, he, in a sense, has two identities with two associated sets of privileges.
But a view (or a stored routine) can have only one definer. So, when a view (or a stored routine) is created with the `SQL SECURITY DEFINER`, one can specify whether the definer should be `CURRENT_USER` (and the view will have none of the privileges of the user's role) or `CURRENT_ROLE` (in this case, the view will use role's privileges, but none of the user's privileges). As a result, sometimes one can create a view that is impossible to use.

```
CREATE ROLE r1;

GRANT ALL ON db1.* TO r1;

GRANT r1 TO foo@localhost;

GRANT ALL ON db.* TO foo@localhost;

SELECT CURRENT_USER
+---------------+
| current_user  |
+---------------+
| foo@localhost |
+---------------+

SET ROLE r1;

CREATE TABLE db1.t1 (i INT);

CREATE VIEW db.v1 AS SELECT * FROM db1.t1;

SHOW CREATE VIEW db.v1;
+------+-----------------------------------------------------------------------------------------------------------
----------------------------------------+----------------------+----------------------+
| View | Create View
| character_set_client | collation_connection |
+------+-----------------------------------------------------------------------------------------------------------
----------------------------------------+----------------------+----------------------+
| v1   | CREATE ALGORITHM=UNDEFINED DEFINER=`foo`@`localhost` SQL SECURITY DEFINER VIEW `db`.`v1` AS
SELECT `db1`.`t1`.`i` AS `i` from `db1`.`t1` | utf8                 | utf8_general_ci      |
+------+-----------------------------------------------------------------------------------------------------------
----------------------------------------+----------------------+----------------------+

CREATE DEFINER=CURRENT_ROLE VIEW db.v2 AS SELECT * FROM db1.t1;

SHOW CREATE VIEW db.b2;
+------+-----------------------------------------------------------------------------------------------------------
---------------------------+----------------------+----------------------+
| View | Create View
| character_set_client | collation_connection |
+------+-----------------------------------------------------------------------------------------------------------
---------------------------+----------------------+----------------------+
| v2   | CREATE ALGORITHM=UNDEFINED DEFINER=`r1` SQL SECURITY DEFINER VIEW `db`.`v2` AS select
`db1`.`t1`.`a` AS `a` from `db1`.`t1` | utf8                 | utf8_general_ci      |
+------+-----------------------------------------------------------------------------------------------------------
---------------------------+----------------------+----------------------+
```

Other Resources

- Roles Review by Peter Gulutzan

This page is licensed: CC BY-SA / Gnu FDL

System Users, Roles, Privileges

> **Warning:**
>
> The `PUBLIC` role is created implicitly by `GRANT` statements and its creation is not logged, distinguishing it from standard system principals.

MariaDB automatically creates several users and roles for administrative and internal server functions.

System Users

These user accounts are created by the `mariadb-install-db` script during the initial server setup.

`root@localhost`

Creation	Created automatically by `mariadb-install-db`.
Purpose	Serves as the primary administrative account for initial server setup and management.
Management	It is highly recommended to secure this account immediately after installation. *Standard security practices include setting a strong password, renaming the account, or removing it entirely in favor of other named administrative accounts.*

`mariadb.sys@localhost`

Creation	Created automatically by `mariadb-install-db`.
Purpose	A mandatory system user required for internal server operations, such as executing scheduled events.
Management	

System Roles

These roles are built into the server and have special behaviors.

The `PUBLIC` Role

The `PUBLIC` role is a special, built-in concept that represents every user on the server.

Creation

The `PUBLIC` role is created implicitly by the server the first time a `GRANT ... TO PUBLIC` statement is executed. It is not created with `CREATE ROLE`.

Purpose

It provides a convenient way to grant privileges server-wide without having to grant them to each user individually. Privileges granted to `PUBLIC` are inherited by all existing and future users.

Management & Security:

- Because the `PUBLIC` role is created implicitly, its creation is **not written to the audit log or binary log** as a standard DDL event. This is a critical detail for security auditing.
- The `PUBLIC` role cannot be explicitly dropped with `DROP ROLE`.
- To audit the privileges that apply to all users, you must check the grants for `PUBLIC` directly.

Syntax Examples:

```
-- Grant a privilege to all users
GRANT SELECT ON my_app.reports TO PUBLIC;

-- View privileges granted to PUBLIC
SHOW GRANTS FOR PUBLIC;

-- Revoke a privilege from all users
REVOKE SELECT ON my_app.reports FROM PUBLIC;
```

This page is licensed: CC BY-SA / Gnu FDL

Account Locking

Syntax

```
CREATE USER [...]
[lock_option] [password_option]

ALTER USER [...]
[lock_option] [password_option]
```

CURRENT

The *lock_option* and *password_option* clauses can occur in either order.

<10.4.7, < 10.5.8

Prior to MariaDB 10.4.7 and MariaDB 10.5.8, the *lock_option* must be placed before the *password_option*.

Description

Account locking permits privileged administrators to lock/unlock user accounts. No new client connections will be permitted if an account is locked (existing connections are not affected).

Locking Accounts

User accounts can be locked at creation, with the CREATE USER statement, or modified after creation with the ALTER USER statement. For example:

```
CREATE USER 'lorin'@'localhost' ACCOUNT LOCK;
```

or

```
ALTER USER 'marijn'@'localhost' ACCOUNT LOCK;
```

The server will return an `ER_ACCOUNT_HAS_BEEN_LOCKED` error when locked users attempt to connect:

```
mariadb -ulorin
  ERROR 4151 (HY000): Access denied, this account is locked
```

Unlocking Accounts

The ALTER USER statement is used to unlock a user:

```
ALTER USER 'lorin'@'localhost' ACCOUNT UNLOCK;
```

Show Whether a Specific Account is Locked

The SHOW CREATE USER statement will show whether the account is locked:

```
SHOW CREATE USER 'marijn'@'localhost';
+-----------------------------------------------+
| CREATE USER for marijn@localhost              |
+-----------------------------------------------+
| CREATE USER 'marijn'@'localhost' ACCOUNT LOCK |
+-----------------------------------------------+
```

as well as querying the mysql.global_priv table:

```
SELECT CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv))
  FROM mysql.global_priv
  WHERE user='marijn';
+-----------------------------------------------------------------------------------------+
| CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv))                                    |
+-----------------------------------------------------------------------------------------+
| marijn@localhost => {
    "access": 0,
    "plugin": "mysql_native_password",
    "authentication_string": "",
    "account_locked": true,
    "password_last_changed": 1558017158
} |
+-----------------------------------------------------------------------------------------+
```

Find Locked Accounts

This query against the `mysql.global_priv` table will return all accounts which have `"account_locked": true` within the `Priv` json column:

```
SELECT CONCAT(user, '@', host) AS 'Locked Accounts' FROM mysql.global_priv WHERE Priv like
'%account_locked":true%';
```

Example Output:

```
+------------------------+
| Locked Accounts        |
+------------------------+
| mariadb.sys@localhost  |
| bart.simpson@localhost |
+------------------------+
2 rows in set (0.000 sec)
```

See Also

- Account Locking and Password Expiry video tutorial

This page is licensed: CC BY-SA / Gnu FDL

Authentication

For Windows, see Authentication Plugin - GSSAPI.

Overview

There are four **new main features in 10.4** relating to authentication:

- It is possible to use more than one authentication plugin for each user account. For example, this can be useful to slowly migrate users to the more secure ed25519 authentication plugin over time, while allowing the old mysql_native_password authentication plugin as an alternative for the transitional period.
- The `root@localhost` user account created by mariadb-install-db is created with the ability to use two authentication plugins.
  - First, it is configured to try to use the unix_socket authentication plugin. This allows the `root@localhost` user to log in without a password via the local Unix socket file defined by the socket system variable, as long as the login is attempted from a process owned by the operating system `root` user account.
  - Second, if authentication fails with the unix_socket authentication plugin, then it is configured to try to use the mysql_native_password authentication plugin. However, an invalid password is initially set, so in order to authenticate this way, a password must be set with SET PASSWORD.

    - However, just using the unix_socket authentication plugin may be fine for many users, and it is very secure. You may want to try going without password authentication to see how well it works for you. Remember, the best way to keep your password safe is not to have one!

- All user accounts, passwords, and global privileges are now stored in the mysql.global_priv table. The mysql.user table still exists and has exactly the same set of columns as before, but it's now a view that references the mysql.global_priv table. Tools that analyze the mysql.user table should continue to work as before. From MariaDB 10.4.13, the dedicated `mariadb.sys` user is created as the definer of this view. Previously, `root` was the definer, which resulted in privilege problems when this username was changed.
- MariaDB 10.4 adds support for User Password Expiry, which is not active by default.

Description

As a result of the above changes, the open-for-everyone all-powerful root account is finally gone. And installation scripts will no longer demand that you "PLEASE REMEMBER TO SET A PASSWORD FOR THE MariaDB root USER!", because the root account is securely created automatically.

Two all-powerful accounts are created by default — root and the OS user that owns the data directory, typically mysql. They are created as:

```
CREATE USER root@localhost IDENTIFIED VIA unix_socket
    OR mysql_native_password USING 'invalid'
CREATE USER mysql@localhost IDENTIFIED VIA unix_socket
    OR mysql_native_password USING 'invalid'
```

Using unix_socket means that if you are the system root user, you can log in as root@localhost without a password. This technique was pioneered by Otto Kekäläinen in Debian MariaDB packages and has been successfully used in Debian since as early as MariaDB 10.0.

It is based on a simple fact that asking the system root for a password adds no extra security — root has full access to all the data files and all process memory anyway. But not asking for a password means there is no root password to forget (no need for the numerous tutorials on "how to reset MariaDB root password"). And if you want to script some tedious database work, there is no need to store the root password in plain text for the script to use (no need for debian-sys-maint user).

Still, some users may wish to log in as MariaDB root without using sudo. Hence the old authentication method — conventional MariaDB password — is still available. By default, it is disabled ("invalid" is not a valid password hash), but one can set the password with a usual SET PASSWORD statement. And still retain the password-less access via sudo.

If you install MariaDB locally (say from a tarball), you will not want to use sudo to be able to log in. This is why MariaDB creates a second all-powerful user with the same name as a system user that owns the data directory. In local (not system-wide) installations, this will be the user who installed MariaDB — they automatically get convenient password-less root-like access, because they can access all the data files anyway.

Even if MariaDB is installed system-wide, you may not want to run your database maintenance scripts as system root — now you can run them as system mysql user. And you will know that they will never destroy your entire system, even if you make a typo in a shell script.

However, seasoned MariaDB DBAs who are used to the old ways do need to make some changes. See the examples below for common tasks.

Cookbook

After installing MariaDB system-wide, the first thing you've got used to doing is logging in to the unprotected root account and protecting it, that is, setting the root password:

```
$ sudo dnf install MariaDB-server
$ mariadb -uroot
...
MariaDB> set password = password("XH4VmT3_jt");
```

This is not only unnecessary now, but it will also simply not work — there is no unprotected root account. To log in as the root user, use

```
$ sudo dnf install MariaDB-server
$ sudo mariadb
```

Note that it implies you are connecting via the unix socket, not tcp. If you happen to have `protocol=tcp` in a system-wide `/etc/my.cnf` file, use `sudo mariadb --protocol=socket`.

After installing MariaDB locally, you've also used it to connect to the unprotected root account using `mariadb -uroot`. This will not work either; simply use `mariadb` without specifying a username.

If you've forgotten your root password, no problem — you can still connect using sudo and change the password. And if you've also removed unix_socket authentication, to restore access, do as follows:

- restart MariaDB with –skip-grant-tables
- log in to the unprotected server
- run FLUSH PRIVILEGES (note, before 10.4, this would've been the last step, not anymore). This disables `--skip-grant-tables` and allows you to change the stored authentication method
- run SET PASSWORD FOR root@localhost to change the root password.

To view inside privilege tables, the old mysql.user table still exists. You can select from it as before, although you cannot update it anymore. It doesn't show alternative authentication plugins, and this was one of the reasons for switching to the mysql.global_priv table — complex authentication rules did not fit into the rigid structure of a relational table. You can select from the new table, for example:

```
SELECT CONCAT(user, '@', host, ' => ', json_detailed(priv)) FROM mysql.global_priv;
```

Reverting to the Previous Authentication Method for root@localhost

If you don't want the `root@localhost` user account created by mariadb-install-db to use unix_socket authentication by default, then there are a few ways to revert to the previous mysql_native_password authentication method for this user account.

Configuring mariadb-install-db to Revert to the Previous Authentication Method

One way to revert to the previous mysql_native_password authentication method for the `root@localhost` user account is to execute mariadb-install-db with a special option. If mariadb-install-db is executed while `--auth-root-authentication-method=normal` is specified, then it will create the default user accounts using the default behavior of MariaDB 10.3 and before.

This means that the `root@localhost` user account will use mysql_native_password authentication by default. There are some other differences as well. See mariadb-install-db: User Accounts Created by Default for more information.

For example, the option can be set on the command-line while running mariadb-install-db:

```
mariadb-install-db --user=mysql --datadir=/var/lib/mysql --auth-root-authentication-method=normal
```

The option can also be set in an option file in an option group supported by mariadb-install-db. For example:

```
[mysql_install_db]
auth_root_authentication_method=normal
```

If the option is set in an option file and if mariadb-install-db is executed, then mariadb-install-db will read this option from the option file, and it will automatically set this option.

Altering the User Account to Revert to the Previous Authentication Method

If you have already installed MariaDB, and if the `root@localhost` user account is already using unix_socket authentication, then you can revert to the old mysql_native_password authentication method for the user account by executing the following:

```
ALTER USER root@localhost IDENTIFIED VIA mysql_native_password
     USING PASSWORD("verysecret")
```

See Also

- Authentication from MariaDB 10 4 video tutorial
- Authentication in MariaDB 10.4 — understanding the changes (mariadb.org)

This page is licensed: CC BY-SA / Gnu FDL

Incrementing of the access_denied_errors Status Variable

The access_denied_errors status variable is incremented when someone tries to access something they do not have rights to.

This happens in the following cases:

- When accessing a database, table, or column that the user does not have rights to access. The error is sent to the client.
- When a login fails because of the wrong user/password, etc. The error is printed to the general log.
- When the require_secure_transport option is enabled on the server, and the user has not used a secure transport. The error is sent to the client.
- Users try to change to a database/schema they do not have access to. A warning is written to the error log if log_warnings > 1.
- Users try to use a `SHOW` command to access an object they do not have rights to see. The error is sent to the client.
- Users access something that requires global access, like “CREATE SERVER”. The error is sent to the client.

Login failures can be found in the general log. Errors that are sent to the client can be found by using the SQL Error Log Plugin. The plugin captures all errors sent to the client. Starting from MariaDB 10.11.5, it can also optionally capture all warnings sent to the client.

See Also

- Troubleshooting Connection Issues
- GRANT
- Error 1045: Access denied for user (using password)

This page is licensed: CC BY-SA / Gnu FDL

User Password Expiry

Password expiry permits administrators to expire user passwords, either manually or automatically.

System Variables

There are two system variables which affect password expiry: default_password_lifetime, which determines the amount of time between requiring the user to change their password. `0`, the default, means automatic password expiry is not active.

The second variable, disconnect_on_expired_password determines whether a client is permitted to connect if their password has expired, or whether they are permitted to connect in sandbox mode, able to perform a limited subset of queries related to resetting the password, in particular SET PASSWORD and SET.

Setting a Password Expiry Limit for a User

Besides automatic password expiry, as determined by default_password_lifetime, password expiry times can be set on an individual user basis, overriding the global using the CREATE USER or ALTER USER statements, for example:

```
CREATE USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY;
```

```
ALTER USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY;
```

Limits can be disabled by use of the `NEVER` keyword, for example:

```
CREATE USER 'monty'@'localhost' PASSWORD EXPIRE NEVER;
```

```
ALTER USER 'monty'@'localhost' PASSWORD EXPIRE NEVER;
```

A manually set limit can be restored the system default by use of `DEFAULT`, for example:

```
CREATE USER 'monty'@'localhost' PASSWORD EXPIRE DEFAULT;
```

```
ALTER USER 'monty'@'localhost' PASSWORD EXPIRE DEFAULT;
```

Note that the limit is defined as the number of days since the last password change. And the last password change is the value of `CURRENT_TIMESTAMP` when the password was changed last. If the `@@secure_timestamp` variable is set to NO (which is its default value) any user can modify the session timestamp arbitrarily, in particular, they can pretend that the password was changed at some point in time far in the future, effectively disabling any password expiration limit.
To prevent it you need to make sure the `@@secure_timestamp` is set or to audit password expiration limits regularly.

SHOW CREATE USER

The SHOW CREATE USER statement will display information about the password expiry status of the user. Unlike MySQL, it will not display if the user is unlocked, or if the password expiry is set to default.

```
CREATE USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY;
CREATE USER 'konstantin'@'localhost' PASSWORD EXPIRE NEVER;
CREATE USER 'amse'@'localhost' PASSWORD EXPIRE DEFAULT;

SHOW CREATE USER 'monty'@'localhost';
+-----------------------------------------------------------------------+
| CREATE USER for monty@localhost                                       |
+-----------------------------------------------------------------------+
| CREATE USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY |
+-----------------------------------------------------------------------+

SHOW CREATE USER 'konstantin'@'localhost';
+-------------------------------------------------------------+
| CREATE USER for konstantin@localhost                        |
+-------------------------------------------------------------+
| CREATE USER 'konstantin'@'localhost' PASSWORD EXPIRE NEVER |
+-------------------------------------------------------------+

SHOW CREATE USER 'amse'@'localhost';
+--------------------------------+
| CREATE USER for amse@localhost |
+--------------------------------+
| CREATE USER 'amse'@'localhost' |
+--------------------------------+
```

Checking When Passwords Expire

MariaDB starting with 11.5

From MariaDB 11.5, the Information Schema USERS table stores password expiry and password error information. An unprivileged user can access their own user data from the table.

The following query can also be used to check when the current passwords expire for all users:

```
WITH password_expiration_info AS (
  SELECT User, Host,
  IF(
   IFNULL(JSON_EXTRACT(Priv, '$.password_lifetime'), -1) = -1,
   @@global.default_password_lifetime,
   JSON_EXTRACT(Priv, '$.password_lifetime')
  ) AS password_lifetime,
  JSON_EXTRACT(Priv, '$.password_last_changed') AS password_last_changed
  FROM mysql.global_priv
)
SELECT pei.User, pei.Host,
  pei.password_lifetime,
  FROM_UNIXTIME(pei.password_last_changed) AS password_last_changed_datetime,
  FROM_UNIXTIME(
   pei.password_last_changed +
   (pei.password_lifetime * 60 * 60 * 24)
  ) AS password_expiration_datetime
  FROM password_expiration_info pei
  WHERE pei.password_lifetime != 0
   AND pei.password_last_changed IS NOT NULL
UNION
SELECT pei.User, pei.Host,
  pei.password_lifetime,
  FROM_UNIXTIME(pei.password_last_changed) AS password_last_changed_datetime,
  0 AS password_expiration_datetime
  FROM password_expiration_info pei
  WHERE pei.password_lifetime = 0
   OR pei.password_last_changed IS NULL;
```

–connect-expired-password Client Option

The mariadb client `--connect-expired-password` option notifies the server that the client is prepared to handle expired password sandbox mode (even if the `--batch` option was specified).

See Also

- Account Locking and Password Expiry video tutorial

This page is licensed: CC BY-SA / Gnu FDL

Catalogs

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

Catalogs are a user account management feature. This section explains their role in organizing database objects and controlling access permissions.

See Catalogs Overview.

Introduces MariaDB catalogs, a multi-tenancy feature that lets a single server host multiple independent tenants, each with their own users, schemas, and logs.

See Starting with Catalogs.

Guide to initializing a MariaDB server with catalog support using `mariadb-install-db --catalogs` and adding new catalogs to a running instance.

See Catalog-Specific Functions and Variables.

Documents the catalog() function, which returns the current catalog name, and the @@catalogs system variable, which indicates if the server is configured for catalogs.

See Catalog Status Variables.

Covers status variables related to catalog operations and performance, useful for monitoring multi-tenant environments.

See Connecting to a Server Configured for Catalogs.

Explains how to connect to a specific catalog using the –catalog client option or the catalog_name.database_name syntax.

See CREATE CATALOG.

Define external catalogs for data integration. This statement configures connections to remote storage systems, allowing query access to external data sources.

See SHOW CREATE CATALOG.

Reference for the SHOW CREATE CATALOG statement, which displays the SQL statement used to create a specific catalog.

See SHOW CATALOGS.

Reference for the SHOW CATALOGS statement, which lists all available catalogs on the server.

See USE CATALOG.

Reference for the USE CATALOG statement, allowing a user to switch their current session's context to a different catalog.

See DROP CATALOG.

Reference for the DROP CATALOG statement, used to remove a catalog and all its associated databases and users.

This page is licensed: CC BY-SA / Gnu FDL

Catalogs Overview

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

MariaDB catalogs will be a multi-tenancy feature where a single instance MariaDB server handles multiple independent tenants (customers), who have their own users, schemas etc.
See MDEV-31542 "Add multi-tenancy catalogs to MariaDB" for details.

Background

For hosting providers, a common solution, to drive down cost, is to have one MariaDB server support several different customers by creating one named schema for each of them.

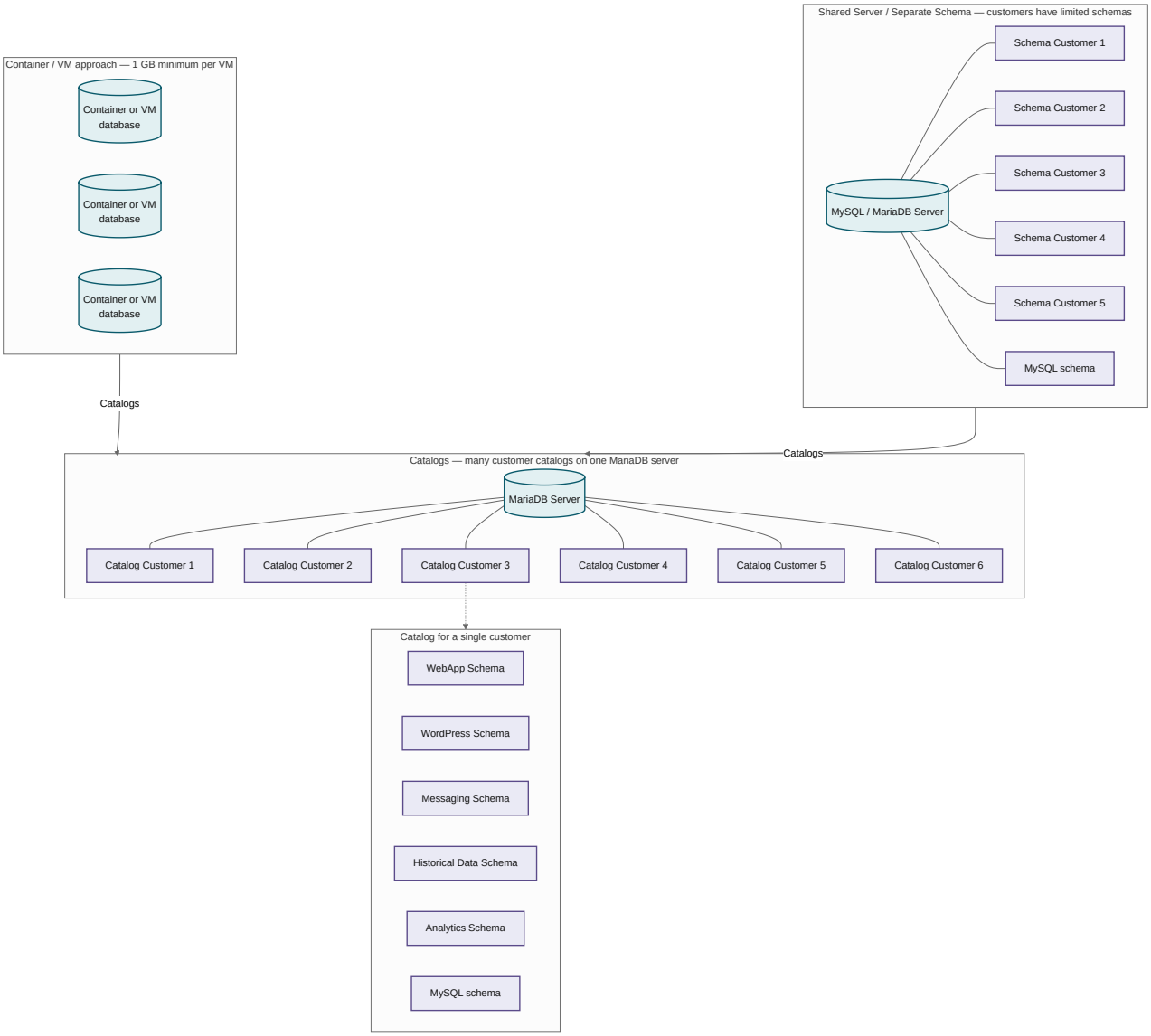
This has however a lot of limitations:

- The user cannot have exactly the same schema(s) on the cloud as they have on premise.
- The user cannot use multiple schemas.
- The user cannot take a backup of all their data (not even with mariadb-dump). This is because the 'mysql' schema, which includes users, stored procedures etc. cannot be copied as its data is shared among all server users.
- The user cannot access the general or error log.

The other option is to have a MariaDB container for each tenant. The problem with this is that, because of the memory needed per container, one can only have a very limited number of tenants per computer.

The suggested solution is to solve all of the above and thus create a better multi-tenant database is to add support for catalogs to MariaDB.

The following picture shows the change:



Moving to catalogs: instead of one VM or one schema per customer, a single MariaDB server hosts many customer catalogs, and each catalog holds all of that customer's schemas.

By each user having their own catalog, they will get very close to the same user experience as if they would have the MariaDB server for themselves.

Catalogs make it possible for hosting providers to have 10-100x more 'not that active' database users on a server compared to having a container or MariaDB server per customer (which limits a 192G server to about 100 customers with a 1G InnoDB buffer each).

User Experience With Catalogs

- Each user is assigned one catalog. The user can specify their catalog in their my.cnf file or as an argument to clients or when connecting to MariaDB server.
- Users can mariadb-dump of all their tables (including the 'mysql' database) and apply it on their own on premise MariaDB or to another 'MariaDB catalog' to duplicate their setup.
- Each catalog has its own privilege system. This allows a MariaDB admin to create users independently in their catalog to users in any other catalog. This also implies that the catalog has to be part of the connect information as otherwise the server does not know which user table to use.

- If the user is using applications that don't yet support catalogs, they can specify the catalog as part of the database when connecting to the server ('catalog.database') or by connecting to a specific port that is associated with a catalog.
- After logging in, a normal user can only see the objects (databases, tables, users etc) from their database. They cannot access other catalogs or change catalogs.
- A normal user cannot change the active catalog with a command. They need to logout from the current catalog and login to another.

For the end user, the MariaDB server will act as a normal a standalone server, with the following differences:

- When connecting to the server, a normal user must specify the catalog. If the connector software does not support catalogs, then the catalog should be specified in the database string. If the catalog is not specified, the 'def' catalog is assumed.
- LOAD DATA INFILE and SELECT … INTO OUTFILE can be configured to only be used with the catalog directory or a directory in it.
- SHUTDOWN command is only for the 'catalog root users'
- Replication (MASTER and SLAVE commands) are only for 'catalog root users'
- Errors from background task (like write error) will be logged into the system error log, not the catalog error log.
- SHOW STATUS will show status data for the whole server, not only for the active catalog.
- The server will handle legacy applications by extending the default database in the connection to contain the catalog in the form "catalog/database". See Appendix for details.
- Tables that are only read from the 'def.mysql' schema:
  - plugin
  - help_* tables
  - time_zone* tables
  - gtid_slave_pos (replication state)
  - innodb_index_stats (innodb internal)
  - servers (federated)
  - transaction_registry (innodb internal)
  - func (udf)
  - performance_schema

New 'catalog root user'

- The 'def' catalog is reserved to store permissions for 'catalog root users', which can access any catalog. * These are meant for admin users that need to do tasks like shutdown, upgrade, create/drop catalogs, managing primaries and replicas etc.
- Only the 'catalog root user' can change to another catalog with 'set catalog catalog_name'.
- A normal user can do 'set catalog current-catalog'. This will be needed to be able to execute a mariadb-dump that includes this command.

New Storage Layout

MariaDB server will be able to run either on 'original mode', where the data layout is exactly as it was before, or on 'catalog' mode, with a new data layout:

When running mariadb-install-db with –use-catalogs, it will create the following new data structure:

- data_directory/
  - engine system data files
  - system files

    - replication files
    - general.log
    - error.log
    - mariadb/
        - mysql/
        - privilege tables
    - catalog1
    - general.log
    - error.log
    - mysql/
        - privilege tables
    - database1/
        - tables for database1
    - database2/
        - tables for database2
    - catalog2/
        - general.log
        - error.log
    - mysql/
        - privilege tables
    - database1/
        - tables for database1
    - database2/
        - tables for database2

The disk structure when not using catalogs is:

- data_directory/
    - engine system data files
    - system files
    - replication files
    - general.log
    - error.log
    - mysql/
        - privilege tables
    - database1/
        - tables for database1
    - database2/
        - tables for database2

The above shows:

- There is a ‘mariadb’ catalog that stores admin users that can access all catalogs, shutdown servers, create new catalogs etc. The ‘system root’ user uses this when connecting.
- Each catalog has their own users, privilege tables, databases, error log and general logs

The MariaDB server will automatically start in catalog mode if it notices the new directory structure.

Catalog SQL Commands/Functions

- USE CATALOG catalog_name;
- CREATE CATALOG
- DROP CATALOG
- ALTER CATALOG
- SHOW CATALOGS (and also information_schema.catalogs)
- SHOW CREATE CATALOG catalog_name;
- SELECT CATALOG();

Changes Needed in MariaDB Codebase

Client changes:

- Add –catalog option to all standard MariaDB clients
- Add support for looping over all existing catalogs to:
  - mariadb-dump
  - mariadb-backup
  - mariadb-upgrade

Changes to mariadb-install-db:

- Allow one to create multiple catalogs at once: -–catalogs=”catalog1,catalog2”
- Init MariaDB with catalog support: —use-catalogs

Changes to mariadb (mysql client):

- Add support for ‘USE CATALOG xxx’’ (and later ‘use database xxx’).

Changes to mysql-test-run:

- Add support of running tests with catalogs (normal tests are run without catalogs)

Changes to MariaDB server (See MDEV-31542):

- Add support for ‘catalog’ in the connection string. For old clients, the user can specify the catalog as part of the database. If catalog is not specified, the ‘def’ catalog (like now) is assumed.
- Add CATALOG() function that returns the current catalog.
- Add ‘USE CATALOG xxx’
- Add ‘USE DATABASE xxx’
- Create a global CATALOG object to hold all information related to the catalog.
- Add the current catalog to the ‘thd’ object.
- Add catalog argument to all functions that take ‘database’ as an argument.
- Add SHOW CATALOGS and information_schema.catalogs
- Move all relevant global variables (users, privileges, mdl-locks(?), open log files) to be stored in the CATALOG structure.
- Add ‘catalog privilege’, for ‘catalog super users’ to allow them to access data in any catalog.
- Add support for accessing tables with ‘catalog.schema.table’ (needed for catalog super users).
- For normal users, only show processes for the current catalog in ‘show processlist’.
- Add loops over all catalogs for information schema for the ‘catalog root user’.
- Update performance schema to take catalogs into account.
- Work with external connectors to get them to support connecting with a catalog.
- Check/update all storage engines to ensure they work also with catalogs.

Notes:

- The storage handler calls will probably not be changed. The storage engine will get the catalog name as part of the database name (catalog/database).
- We don't need a 'catalog' column for tables in the 'mysql' schema (like mysql.proc) as these are stored per catalog.

Some Implementation Ideas

- Instead of sending a catalog string to function, use a pointer to the global catalog object. Do the same later for databases. This allows use to precompute things like 'filename' for catalogs and databases and we don't have to do this for every table open. It also allows us to later support logging information at a catalog and database level.
- Don't take a MDL lock for the catalog for each table. The metadata lock for the catalog will be taken when a user logs in or changes catalog.
- Add system variables 'current_catalog' and 'current_database' and allow users to change these.
- Add support for 'catalog ports' that are connected to catalog. This allows users to connect to a specific catalog from any client software.

Limitations (in addition to limitations listed in "User experience with catalogs")

- Database names cannot contain '.' when connecting from clients without the new catalog connect option.
- One cannot refer to other catalogs in triggers, stored procedures, events etc. This is because a transaction cannot span catalogs.
- Only the catalog root user can use mariadb-backup. This is a normal restriction as one has to be system root to be able to use mariadb-backup.
- Events are global (to save resources). Catalog users can enable/disable events for their catalog.

Stage 2 (not in first release)

- Support usage statistics per catalog and whole server (the last for the 'catalog root user'). This allows the DBA to see the number of queries, type of queries etc. Some 'system' and 'global innodb' statistics will only be shown globally (number of open files, number of sync calls etc).
- Support a my.cnf file in each catalog directory to handle catalog (customer) unique defaults.
- Add quotas per catalog for tables and temporary files.
- Add more support to limit users from overusing resources (cpu, tables, databases, number of connections etc)
- Support 'drop catalog'. (This is in Stage 2 as there may be some issues to drop already active CATALOG objects)
- Add optional catalog support to the S3 engine
- More things will be added later.

Stage 3

- Allow users to manage their own replication stream (maybe?).
- Allow users to have different options for the S3 engine
- More things will be added later.

Appendix

Legacy Connector Support

SQLALchemy test:

```
In [1]: from sqlalchemy.engine import make_url
In [2]: u = make_url('mariadb+mariadbconnector://app_user:Password123!@127.0.0.1:3306/catalog/company')
In [3]: u.database
Out[3]: 'catalog/company'
```

The following tests ensured that inside the server (mysql_change_db), the “catalog/test” was picked up as the database.

PHP PDO test:

```
$ php -r '$db = new
\PDO("mysql:host=localhost;user=dan;dbname=catalog/test;charset=utf8mb4;unix_socket=/tmp/build-mariadb-
server-10.4.sock");'
```

PHP mysqli test:

```
php -r '$dbcon = mysqli_connect("localhost","dan","nopass","catalog/test",3306,"/tmp/build-mariadb-
server-10.4.sock");'
```

Nodejs test:

```
var mysql = require('mysql')
var con = mysql.createConnection({
  socketPath: "/tmp/build-mariadb-server-10.4.sock",
  user: "dan",
  password: "yourpassword",
  database: "catalog/test",
})
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
})
```

(need to map out a few other connectors here to make sure it’s supported well in this form).

Ref: connection.md#handshake-response-packet

Migration of Existing MariaDB Original Mode to the New Catalog Layout

As shared hosting services have a naming scheme from user/schema to database name in MariaDB, to provide a migration to the new catalog layout, the following steps will be required:

- Use mariadb-dump to dump the original data
- On the new server execute:
- mariadb-install-db –catalogs=’catalog_name’
- mariadb –catalog catalog_name < dump_file

This is needed as InnoDB needs to know where the new files are located.

Migration of One Catalog User to Another MariaDB Server

Create a migration tool set / procedure that does the following

- Execute FLUSH TABLES FOR EXPORT for all tables in a catalog.
- Take a copy of the catalog directory
- Copy the data to a new catalog directory to the new server
- Run ALTER TABLE … IMPORT TABLESPACE on each InnoDB table

Note that for partitioned tables the process will be a bit more complex, see above link.

This procedure will be a bit easier after an in-the-works patch for InnoDB related to IMPORT will be pushed. (Should happen before we start on the catalog project)

Other Things

- Drizzle's default catalog was called "local". MariaDB's default will be called 'def', as this is what we already have as the default catalog in information_schema, in current connectors and other places.
- CONNECT engine will need testing against catalogs and maybe a small code change to support them. It could also be a way to join from one catalog to another.

See Also

- MDEV-31542 Add multi-tenancy catalogs to MariaDB

This page is licensed: CC BY-SA / Gnu FDL

Starting with Catalogs

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

Background

mariadb-install-db initializes the MariaDB data directory and creates the system tables in the mysql database.

When used with the `--catalog` options it will initialize MariaDB server to use catalogs.
The mariadbd server will automatically discover if catalogs are used or not.

Note that **one cannot change** a 'normal server' to a server with catalogs or a server with catalogs to a 'normal server'. In the future we will add tools that will allow one to easily move an existing server inside a catalog or move an server inside a catalog to a standalone server.

Initializing a New Server with Catalog Support

To initialize a server with 4 catalogs (the `def` catalog, that holds the catalog root user (CRU) is automatically created):

```
mariadb_install_db --catalogs="cat1 cat2 cat3" --datadir=/my/data/
```

The above will create a directory `/my/data` and the 4 directories under it, one for each catalog.

Adding More Catalogs to a Running Server

Creating Catalogs with CREATE CATALOG

One can create a new catalog with CREATE CATALOG catalog_name

Creating Catalogs with mariadb_install_db

When adding more catalogs to an existing server, `mariadb_install_db` will start the mariadb client to execute the needed commands on the running server. This is why one has to supply user and password to `mariadb_install_db`.

```
mariadb_install_db --catalogs="cat4 cat5 cat6" --datadir=/my/data --catalog-user=monty --catalog-password
```

One benefit of using `mariadb_install_db` is that it's possible to create many catalogs with one command.

See Also

- CREATE CATALOG
- DROP CATALOG
- mariadb-install-db

This page is licensed: CC BY-SA / Gnu FDL

Catalog-Specific Functions and Variables

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

Catalog Functions

catalog()

`catalog()

returns the name of the current catalog.`

```
MariaDB [def.test]> select catalog();
+-----------+
| catalog() |
+-----------+
| def       |
+-----------+
```

Catalog Variables

@@catalogs

One can check if a server supports catalogs with:

```
SELECT @@catalogs;
+------------+
| @@catalogs |
+------------+
|          1 |
+------------+
```

1 means that the server is configured for catalogs.

This page is licensed: CC BY-SA / Gnu FDL

Catalog Status Variables

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

When using a MariaDB Server with catalogs support, all status information is collected for the whole server, per catalog and per session.

```
SHOW SERVER STATUS;
```

shows the status for the whole server. Note that only the super user in the 'def' catalog has privileges for the above statement.

```
SHOW GLOBAL STATUS;
SHOW CATALOG STATUS;
```

Both commands show the status for the current catalog.
The reason that `GLOBAL` shows catalog status is that because catalogs are 'multi-tenant', a catalog user should not be able to see the status from other users (for most things).

```
SHOW [SESSION] STATUS;
```

Shows the status for the current connection.

The main "new thing" is that catalogs enable SAS providers to see the status for a single tenant (catalog user). This makes it much easier to find 'bad neighbors' (tenants that cause problems for other tenants) so that they can be moved to other servers.

When the MariaDB server is not configured for catalogs, the following commands are equivalent:

```
SHOW GLOBAL STATUS
SHOW SERVER STATUS
SHOW CATALOG STATUS
```

See Also

- SHOW STATUS

This page is licensed: CC BY-SA / Gnu FDL

Connecting to a Server Configured for Catalogs

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

When connecting to a MariaDB server configured for catalogs, one has to provide the catalog to connect to. There are several ways to do this:

All new native MariaDB clients will support the `--catalog` option:

```
mariadb --catalog=mine test
```

New and old clients can use the 'catalog_name.database_name' syntax to connect:

```
mariadb mine.test
```

This will connect the user to the 'mine' catalog and the database 'test'.

Note that one consequence of this is that one should not have a database that contains '.' in the name. If such a database exists, one can still connect to it by using the `--catalog=` option or prefixing the database with the catalog, like in `def.data.base.name`.

One will also be able to configure the MariaDB server to automatically choose catalogs depending on the port or IP they are using to connect to the server. This is done by adding the following to the catalog specific my.cnf file, residing in the catalog directory:

```
[[mariadbd]]
--port=#
--connect-ip=
```

If catalogs is not specified either directly (—catalog=#) or indirectly (with port or ip) the catalog `def` will be used.

When connecting to a server not configured for catalogs, one can still use `mariadb --catalog=def` or `mariadb def.datbase_name`.

To check if a server supports catalogs:

```
SELECT @@catalogs;
+------------+
| @@catalogs |
+------------+
|          0 |
+------------+
USE catalog foo;
ERROR 4193 (HY000): MariaDB is not configured to support catalogs
```

0 means that the server is not configured for catalogs.

This page is licensed: CC BY-SA / Gnu FDL

CREATE CATALOG

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

Syntax

```
CREATE CATALOG [IF NOT EXISTS] catalog_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | COMMENT [=] 'comment'
```

Description

Creates a catalog and the `mysql`, `sys` and `performance_schema` schemas inside the catalog.

`CREATE CATALOG` can only be performed by a user in the `def` catalog with the CATALOG privilege.

Note that no users are created.

Example

```
CREATE CATALOG cat1;
USE CATALOG cat1;
CREATE USER root@localhost;
GRANT ALL PRIVILEGES ON *.* TO root@localhost;
CREATE DATABASE test;
```

Limitations

The catalog name is limited to 64 characters. All characters must be in the basic ASCII set:
(A-Z, a-z, -, _)
This limitations is to be able to run catalogs with engines like InnoDB which has limited space in their internal data dictionary.

Pre-Creating Catalog Directories

`CREATE CATALOG` works even if the catalog directory already exists (as long as there is no `mysql` sub directory). This is to allow a database administrator to pre-create the catalog directory and mount it to disk volume and optionally add a configuration file inside the catalog directory.
The directory will not be recognized as a catalog or show up in SHOW CATALOGS until the `mysql` sub directory is created by CREATE CATALOG or mariadb-install-db.

See Also

- marriadb-install-db can be used to create multiple catalogs with a default root user in one go.
- DROP CATALOG

This page is licensed: CC BY-SA / Gnu FDL

SHOW CREATE CATALOG

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

Syntax

```
SHOW CREATE CATALOG catalog_name
```

Description

Shows the CREATE CATALOG statement that creates the given catalog.

Examples

```
SHOW CREATE CATALOG def;
+---------+-----------------------------------------------------------------------------------------------
---------+
| Catalog | Create Catalog
|
+---------+-----------------------------------------------------------------------------------------------
---------+
| def     | CREATE CATALOG `def` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci COMMENT 'default
catalog' |
+---------+-----------------------------------------------------------------------------------------------
---------+
```

See Also

- CREATE CATALOG
- Character Sets and Collations

This page is licensed: CC BY-SA / Gnu FDL

SHOW CATALOGS

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

Syntax

```
SHOW CATALOGS
    [LIKE 'pattern' | WHERE expr]
```

Description

`SHOW CATALOGS` lists the catalogs on the MariaDB server host.
The `LIKE` clause, if present on its own, indicates which catalog names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

You see only use `SHOW CATALOGS` have the CATALOG privilege. Only users of the 'def' schema can have this privilege.

If the server was started with the –skip-show-database option, you cannot use this statement unless you have the SHOW DATABASES privilege.

The list of results returned by `SHOW CATALOGS` is based on directories in the data directory, which is how MariaDB implements catalogs. It only list directories that have a `mysql` directory.

The Information Schema Catalogs table also contains catalog information.

Examples

```
+---------+--------------------+
| Catalog | Comment            |
+---------+--------------------+
| c1      | This is catalog c1 |
| cat2    |                    |
| def     | default catalog    |
+---------+--------------------+
```

```
SHOW CATALOGS LIKE 'c%';
+--------------+--------------------+
| Catalog (c%) | Comment            |
+--------------+--------------------+
| c1           | This is catalog c1 |
| cat2         |                    |
+--------------+--------------------+
```

See Also

- CREATE CATALOG
- Character Sets and Collations
- Information Schema CATALOG Table

This page is licensed: CC BY-SA / Gnu FDL

USE CATALOG

> **Note:**
>
> Catalogs are planned for an upcoming release, and don't yet appear in any current releases.

Syntax

```
USE CATALOG catalog_name
```

Description

Changes to another catalog.
Can only be done by a super user in the ‘def’ catalog.
Changing catalog will update catalog status and reset all session status.

A tenant (a user in any other catalog than ‘def’) cannot change to another catalog.
However tenants can execute `USE CATALOG current_catalog`. This is to allow the user to import SQL scripts that use `USE CATALOG...`.

See Also

- USE database. Changing database.

This page is licensed: CC BY-SA / Gnu FDL

DROP CATALOG

> **Note:**
>
> Catalogs are planned for an upcoming release, and don’t yet appear in any current releases.

Syntax

```
DROP CATALOG catalog_name
```

Description

Deletes a catalog.

Limitations:

- `DROP CATALOG` can only be performed by a super user in the ‘def’ catalog.
- The current catalog cannot be dropped.
- The ‘def’ catalog cannot be dropped.

When dropping a catalog, all databases and files within that catalog will be deleted.

See Also

- CREATE CATALOG

This page is licensed: CC BY-SA / Gnu FDL

Authentication with Enterprise Server

See Authentication for MariaDB Enterprise Server.

Overview of user account authentication using plugins like pam or unix_socket and managing security with password validation plugins.

See Authentication with gssapi.

Instructions for installing and configuring the gssapi plugin to validate user credentials against services like Kerberos or NTLM.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Authentication for MariaDB Enterprise Server

Overview

MariaDB Enterprise Server authentication is performed by database user accounts. Database user accounts are specified by user name, the hostname from which the account is connecting, and the authentication plugins configured for the account, such as mysql_native_password, pam, or unix_socket.

Change Password

The password for a database user account can be changed using the ALTER TABLE, ALTER USER, and SET PASSWORD statements.

```
WITH ALTER USER:

ALTER USER 'USER'@'192.0.2.%'
   IDENTIFIED BY 'PASSWD';
```

```
WITH SET PASSWORD:

SET PASSWORD FOR 'USER'@'192.0.2.%'
   = PASSWORD('PASSWD');
```

Password Validation Plugins

If your MariaDB Enterprise Server node has a password validation plugin installed, then the password should also meet the configured requirements. When you try to set or change a user's password and the password validation plugin rejects the password, the following error message will be shown:

```
ERROR HY000: Your password does not satisfy the current policy requirements.
```

By default, the MariaDB Enterprise Server installs the simple_password_check plugin, but the cracklib_password_check plugin is also available.

For simple_password_check, the password requirements are configured by several system variables:

- simple_password_check_digits
- imple_password_check_letters_same_case
- simple_password_check_minimal_length
- simple_password_check_other_characters

Authentication Plugins

MariaDB Enterprise Server uses authentication plugins to support different authentication methods. The default authentication plugin is mysql_native_password.

- Authentication with ed25519
- Authentication with gssapi
- Authentication with mysql_native_password
- Authentication with mysql_old_password
- Authentication with pam
- Authentication with unix_socket
- Authentication with PARSEC

This page is: Copyright © 2026 MariaDB. All rights reserved.

Authentication with gssapi

Overview

The gssapi authentication plugin validates user credentials against a GSSAPI-based authentication service, like Kerberos or NTLM.

Install Package

The gssapi authentication plugin requires an additional package to be installed on Linux.
On CentOS, RHEL, and Rocky Linux:

```
$ sudo yum install MariaDB-gssapi-server
```

On Debian and Ubuntu:

```
$ sudo apt install mariadb-plugin-gssapi-server
```

On SLES:

```
$ sudo zypper install MariaDB-gssapi-server
```

Configure

The gssapi authentication plugin requires some system variables to be configured, including:

- gssapi_keytab_path
- gssapi_principal_name

For example:

```
[mariadb]
...
gssapi_keytab_path=KEYTAB_PATH
gssapi_principal_name=PRINCIPAL_NAME
```

Install Plugin

The gssapi authentication plugin must be installed before it can be used.

To install with the INSTALL SONAME statement:

```
INSTALL SONAME 'gssapi';
```

To install in a configuration file with the plugin_load_add option:

```
[mariadb]
...
plugin_load_add = auth_gssapi
```

Create User

To create a user account that uses the gssapi authentication plugin, specify the plugin in the CREATE USER statement:

```
CREATE USER 'USER'@'192.0.2.%'
   IDENTIFIED VIA gssapi;
```

An optional realm can be specified:

```
CREATE USER 'USER'@'192.0.2.%'
   IDENTIFIED VIA gssapi USING 'USER@DOMAIN';
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Limiting Size of Created Disk Temporary Files and Tables

See Limiting Size of Created Disk Temporary Files and Tables Overview.

Overview of the feature introduced in MariaDB 11.5 to limit disk space used by temporary files and internal on-disk temporary tables to prevent disk exhaustion.

See max_tmp_session_space_usage System Variable.

Documentation for the system variable that restricts the maximum total size of temporary files and tables allowed for an individual user session.

See max_tmp_total_space_usage System Variable.

Reference for the global system variable that defines the maximum cumulative disk space all user connections can consume for temporary files and tables.

This page is licensed: CC BY-SA / Gnu FDL

Limiting Size of Created Disk Temporary Files and Tables Overview

> **Note:**
>
> This functionality is available from MariaDB 11.5.

You can limit the size of created disk temporary files and tables.
When the internal in-memory temporary table is oversize and converting to MyISAM/Aria table to store on disk, this option will limit the max space of tmp_dir. If a new disk temporary table will cause tmp_dir over the limitation, then this query will return an error.

Temporary Space

The temporary space includes:

- All SQL level temporary files. This includes files for filesort, transaction temporary space, analyze, binlog_stmt_cache etc. It does not include engine internal temporary files used for repair, alter table, index pre sorting etc.
- All internal on disk temporary tables created as part of resolving a SELECT, multi-source update etc.

Special Cases

- When doing a commit, the last flush of the binlog_stmt_cache will not cause an error even if the temporary space limit is exceeded. This is to avoid giving errors on commit. This means that a user can temporary go over the limit with up to binlog_stmt_cache_size.

System Variables

There are two system variables used for controlling this feature:

- max_tmp_session_space_usage: Limits the temporary space allowance per user
- max_tmp_total_space_usage: Limits the temporary space allowance for all users.

Status Variables

- tmp_space_used
- max_tmp_space_used

Information Schema

- New field in information_schema.process_list: TMP_SPACE_USED

Noteworthy issue

- One has to be careful when using small values for max_tmp_space limit together with binary logging and with non transactional tables.
- If a binary log entry for the query is larger than binlog_stmt_cache_size and one hits the limit of max_tmp_space when flushing the entry to disk, the query will abort and the binary log will not contain the last changes to the table. This will also stop the replica!
- This is also true for all Aria tables as Aria cannot do rollback (except in case of crashes)!
- One way to avoid it is to use @@binlog_format=statement for queries that update many lot of rows.

Implementation

- All writes to temporary files or internal temporary tables, that increase the file size, are routed through temp_file_size_cb_func() which updates and checks the temp space usage.
- Most of the temporary file monitoring is done inside IO_CACHE. Temporary file monitoring is done inside the Aria engine.
- MY_TRACK and MY_TRACK_WITH_LIMIT are new flags for ini_io_cache(). MY_TRACK means that we track the file usage. TRACK_WITH_LIMIT means that we track the file usage and we give an error if the limit is breached. This is used to not give an error on commit when binlog_stmt_cache is flushed.
- global_tmp_space_used contains the total tmp space used so far. This is needed quickly check against max_total_tmp_space_usage.
- Temporary space errors are using EE_LOCAL_TMP_SPACE_FULL and handler errors are using HA_ERR_LOCAL_TMP_SPACE_FULL. This is needed until we move general errors to it's own error space so that they cannot conflict with system error numbers.
- Return value of my_chsize() and mysql_file_chsize() has changed so that -1 is returned in the case my_chsize() could not decrease the file size (very unlikely and will not happen on modern systems). All calls to _chsize() are updated to check for > 0 as the error condition.
- At the destruction of THD we check that THD::tmp_file_space == 0
- At server end we check that global_tmp_space_used == 0
- As a precaution against errors in the tmp_space_used code, one can set max_tmp_space_usage and max_total_tmp_space_usage to 0 to disable the tmp space quota errors.
- truncate_io_cache() function added.
- Aria tables using static or dynamic row length are registered in 8K increments to avoid some calls to update_tmp_file_size().

See Also

- MDEV-9101 - Limit size of created disk temporary files and tables

This page is licensed: CC BY-SA / Gnu FDL

max_tmp_session_space_usage System Variable

`max_tmp_session_space_usage`

- Description: The maximum total size of temporary file and temporary table usage. A value of 0 disables this feature. Set in blocks of 65536, rounded down if not a multiple of 65536. See Limiting Size of Created Disk Temporary Files and Tables Overview for details. Named max_tmp_space_usage in MariaDB 11.5.0 only.
- Commandline: `--max-tmp-session-space-usage=num`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric` (bigint unsigned)
- Default Value: `1099511627776`
- Range: `0` to `18446744073709551615` (blocks of `65536`)
- Introduced: MariaDB 11.5

This page is licensed: CC BY-SA / Gnu FDL

max_tmp_total_space_usage System Variable

`max_tmp_total_space_usage`

- Description: The maximum total size in bytes of all temporary file and temporary table usage over all connections. A value of 0 disables this feature. Set in blocks of 65536, rounded down if not a multiple of 65536. See Limiting Size of Created Disk Temporary Files and Tables Overview for details. Called max_total_tmp_space_usage in MariaDB 11.5.0 only.
- Commandline: `--max-tmp-total-space-usage=num`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric` (bigint unsigned)
- Default Value: `1099511627776`
- Range: `0` to `18446744073709551615` (blocks of `65536`)
- Introduced: MariaDB 11.5

This page is licensed: CC BY-SA / Gnu FDL

Security Vulnerabilities (CVE) Fixed in MariaDB

> **Note:**
>
> This page is about security vulnerabilities fixed in MariaDB products. If you are looking for information on securing your MariaDB installation, see Securing MariaDB.

> **Important:**
>
> Sensitive security issues can be reported on https://hackerone.com/mariadb or sent directly to the persons responsible for MariaDB security: security [AT] mariadb (dot) org.

What is a CVE?

Common Vulnerabilities and Exposures (CVE) relate to flaws “in a software, firmware, hardware, or service component resulting from a weakness that can be exploited, causing a negative impact to the confidentiality, integrity, or availability of an impacted component or components.”

It is a publicly available and free-to-use database of known software vulnerabilities maintained at https://cve.mitre.org/.

For additional information, see CVE Glossary at cve.org

What is a CVSS Score?

Common Vulnerability Scoring System (CVSS) is “an open framework for communicating the characteristics and severity of software vulnerabilities.”

For additional information, see Vulnerability Metrics at nvd.nist.gov

Fixed Security Vulnerabilities

See Security Vulnerabilities (CVE) Fixed in MariaDB Enterprise Server.

This page contains a full list of CVEs fixed in all versions and series of MariaDB Enterprise Server.

See Security Vulnerabilities (CVE) Fixed in MariaDB Community Server.

This page contains a full list of CVEs fixed in all versions and series of MariaDB Community Server.

Some CVE apply to MySQL but are not present in MariaDB Enterprise Server or MariaDB Community Server; these are listed on the Security Vulnerabilities fixed in Oracle MySQL that did not exist in MariaDB page.

This page is licensed: CC BY-SA / Gnu FDL

Security Vulnerabilities (CVE) Fixed in MariaDB Enterprise Server

The list is updated continuously. CVEs may be added retroactively to already-released versions, as vulnerabilities can be reported to and assigned by CVE Numbering Authorities (CNAs) after the corresponding fix has been released. As a result, a release may receive additional CVE entries over time even after it has been published.

Table of Fixed Security Vulnerabilities

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Enterprise Server Releases
CVE-2026-3494	2026-03-03	4.3	11.8.8-5 (2026-06-26), 11.4.12-9 (2026-06-26), 10.6.27-23 (2026-06-26)
CVE-2026-44173	2026-06-12	5.0	11.8.8-5 (2026-06-26), 11.4.12-9 (2026-06-26), 10.6.27-23 (2026-06-26)
CVE-2026-44172	2026-06-12	5.0	11.8.8-5 (2026-06-26), 11.4.12-9 (2026-06-26), 10.6.27-23 (2026-06-26)
CVE-2026-44171	2026-06-12	6.3	11.8.8-5 (2026-06-26), 11.4.12-9 (2026-06-26), 10.6.27-23 (2026-06-26)
CVE-2026-44170	2026-06-12	5.0	11.8.8-5 (2026-06-26), 11.4.12-9 (2026-06-26), 10.6.27-23 (2026-06-26)
CVE-2026-44169	2026-06-12	4.3	11.8.8-5 (2026-06-26), 11.4.12-9 (2026-06-26)
CVE-2026-49261	2026-06-11	10.0	11.8.6-4 (2026-06-02), 11.4.10-8 (2026-06-02), 10.6.25-22 (2026-06-02)
CVE-2026-48165	2026-06-12	8.0	11.8.6-4 (2026-06-02), 11.4.10-8 (2026-06-02), 10.6.25-22 (2026-06-02)
CVE-2026-48163	2026-06-12	8.0	11.8.6-4 (2026-06-02), 11.4.10-8 (2026-06-02), 10.6.25-22 (2026-06-02)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Enterprise Server Releases
CVE-2026-44168	2026-06-12	8.0	11.8.6-4 (2026-06-02), 11.4.10-8 (2026-06-02), 10.6.25-22 (2026-06-02)
CVE-2026-32710	2026-03-20	8.6	11.8.6-3 (2026-03-17), 11.4.10-7 (2026-03-17), 10.6.27-23 (2026-06-26)
CVE-2026-35549	2026-04-03	6.5	11.8.6-3 (2026-03-17), 11.4.10-7 (2026-03-17)
CVE-2026-34303	2026-04-21	6.5	11.8.6-3 (2026-03-17), 11.4.10-7 (2026-03-17)
CVE-2026-21968	2026-01-20	6.5	11.8.5-2 (2025-12-11), 11.4.9-6 (2025-12-11), 10.6.24-20 (2025-12-11)
CVE-2025-13699	2025-12-23	7.0	11.8.5-2 (2025-12-11), 11.4.9-6 (2025-12-11), 10.6.24-20 (2025-12-11)
CVE-2025-30693	2025-04-15	5.5	11.4.7-4 (2025-06-11), 10.6.22-18 (2025-06-11), 10.5.29-23 (2025-06-11)
CVE-2023-52969	2025-03-08	4.9	11.4.7-4 (2025-06-11), 10.6.22-18 (2025-06-11), 10.5.29-23 (2025-06-11)
CVE-2023-52970	2025-03-08	4.9	11.4.7-4 (2025-06-11), 10.6.22-18 (2025-06-11), 10.5.29-23 (2025-06-11)
CVE-2023-52971	2025-03-08	4.9	11.4.7-4 (2025-06-11)
CVE-2025-30722	2025-04-15	5.3	11.4.7-4 (2025-06-11), 10.6.22-18 (2025-06-11), 10.5.29-23 (2025-06-11)
CVE-2025-21490	2025-01-21	4.9	11.4.5-3 (2025-03-19), 10.6.21-17 (2025-03-19), 10.5.28-22 (2025-03-19)
CVE-2024-21096	2024-04-16	4.9	10.6.18-14 (2024-06-11), 10.5.25-19 (2024-06-11)
CVE-2023-52968	2025-03-08	4.9	10.6.17-13 (2024-04-24), 10.5.24-18 (2024-03-11), 10.4.33-23 (2024-03-11)
CVE-2023-22084	2023-10-17	4.9	10.6.16-11 (2023-12-12), 10.5.23-17 (2023-12-12), 10.4.32-22 (2023-12-12)
CVE-2022-47015	2023-01-20	N/A (Medium) [#1]	10.5.21-15 (2023-06-13), 10.4.30-20 (2023-06-13), 10.3.39-20 (2023-06-13)
CVE-2022-47015	2023-01-20	N/A (Medium) [#1]	10.6.12-8 (2023-05-24)
CVE-2023-5157	2023-09-26	7.5	10.6.9-5 (2022-09-12), 10.5.17-12 (2022-09-12), 10.4.26-17 (2022-09-12)
CVE-2018-25032	2022-03-25	7.5	10.6.9-5 (2022-09-12), 10.5.17-12 (2022-09-12), 10.4.26-17 (2022-09-12), 10.3.36-17 (2022-09-12)
CVE-2022-32091	2022-07-01	6.5	10.6.9-5 (2022-09-12), 10.5.17-12 (2022-09-12), 10.4.26-17 (2022-09-12), 10.3.36-17 (2022-09-12)
CVE-2022-32089	2022-07-01	6.5	10.6.9-5 (2022-09-12), 10.5.17-12 (2022-09-12), 10.4.26-17 (2022-09-12)
CVE-2022-32084	2022-07-01	6.5	10.6.9-5 (2022-09-12), 10.5.17-12 (2022-09-12), 10.4.26-17 (2022-09-12), 10.3.36-17 (2022-09-12)
CVE-2022-32082	2022-07-01	6.5	10.6.9-5 (2022-09-12), 10.5.17-12 (2022-09-12)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Enterprise Server Releases
CVE-2022-32081	2022-07-01	6.5	10.6.9-5 (2022-09-12), 10.5.17-12 (2022-09-12), 10.4.26-17 (2022-09-12)
CVE-2022-27457	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13)
CVE-2022-27456	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27455	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13)
CVE-2022-27452	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27451	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13)
CVE-2022-27449	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27448	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27447	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27446	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13)
CVE-2022-27445	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27444	2022-04-14	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13)
CVE-2022-27387	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27386	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27384	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27383	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27382	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13)
CVE-2022-27381	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27380	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27379	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27378	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27377	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-27376	2022-04-12	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Enterprise Server Releases
CVE-2022-21451	2022-04-19	7.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13)
CVE-2022-32088	2022-07-01	6.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-32087	2022-07-01	6.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-32086	2022-07-01	6.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13)
CVE-2022-32085	2022-07-01	6.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-32083	2022-07-01	6.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2021-46669	2022-02-01	6.5	10.6.8-4 (2022-06-13), 10.5.16-11 (2022-06-13), 10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2022-21427	2022-04-19	4.9	10.4.25-16 (2022-06-13), 10.3.35-16 (2022-06-13)
CVE-2021-46668	2022-02-01	5.5	10.6.7-3 (2022-03-14), 10.5.15-10 (2022-03-14), 10.4.24-15 (2022-03-14), 10.3.34-15 (2022-03-14), 10.2.43-15 (2022-03-14)
CVE-2021-46665	2022-02-01	5.5	10.6.7-3 (2022-03-14), 10.5.15-10 (2022-03-14), 10.4.24-15 (2022-03-14), 10.3.34-15 (2022-03-14), 10.2.43-15 (2022-03-14)
CVE-2021-46664	2022-02-01	5.5	10.6.7-3 (2022-03-14), 10.5.15-10 (2022-03-14), 10.4.24-15 (2022-03-14), 10.3.34-15 (2022-03-14), 10.2.43-15 (2022-03-14)
CVE-2021-46663	2022-02-01	5.5	10.6.7-3 (2022-03-14), 10.5.15-10 (2022-03-14), 10.4.24-15 (2022-03-14), 10.3.34-15 (2022-03-14), 10.2.43-15 (2022-03-14)
CVE-2021-46661	2022-02-01	5.5	10.6.7-3 (2022-03-14), 10.5.15-10 (2022-03-14), 10.4.24-15 (2022-03-14), 10.3.34-15 (2022-03-14), 10.2.43-15 (2022-03-14)
CVE-2021-46659	2022-01-29	5.5	10.6.7-3 (2022-03-14), 10.5.15-10 (2022-03-14), 10.4.24-15 (2022-03-14), 10.3.34-15 (2022-03-14), 10.2.43-15 (2022-03-14)
CVE-2022-21595	2022-10-18	4.4	10.6.7-3 (2022-03-14), 10.5.15-10 (2022-03-14), 10.4.24-15 (2022-03-14), 10.3.34-15 (2022-03-14), 10.2.43-15 (2022-03-14)
CVE-2022-27385	2022-04-12	7.5	10.6.5-2 (2021-12-13), 10.5.13-9 (2021-12-13), 10.4.22-14 (2021-12-13), 10.3.32-14 (2021-12-13)
CVE-2021-46667	2022-02-01	7.5	10.6.5-2 (2021-12-13), 10.5.13-9 (2021-12-13), 10.4.22-14 (2021-12-13), 10.3.32-14 (2021-12-13), 10.2.41-14 (2021-12-13)
CVE-2022-31624	2022-05-25	6.5	10.6.5-2 (2021-12-13), 10.5.13-9 (2021-12-13), 10.4.22-14 (2021-12-13), 10.3.32-14 (2021-12-13), 10.2.41-14 (2021-12-13)
CVE-2021-46662	2022-02-01	5.5	10.6.5-2 (2021-12-13), 10.5.13-9 (2021-12-13), 10.4.22-14 (2021-12-13), 10.3.32-14 (2021-12-13)
CVE-2021-35604	2021-10-20	5.5	10.5.13-9 (2021-12-13), 10.4.22-14 (2021-12-13), 10.3.32-14 (2021-12-13), 10.2.41-14 (2021-12-13)
CVE-2021-2389	2021-07-20	5.9	10.5.12-8 (2021-09-14), 10.4.21-13 (2021-09-14), 10.3.31-13 (2021-09-14), 10.2.40-13 (2021-09-14)
CVE-2021-46666	2022-02-01	5.5	10.5.12-8 (2021-09-14), 10.4.21-13 (2021-09-14), 10.3.31-13 (2021-09-14), 10.2.40-13 (2021-09-14)
CVE-2021-46658	2022-01-29	5.5	10.5.12-8 (2021-09-14), 10.4.21-13 (2021-09-14), 10.3.31-13 (2021-09-14), 10.2.40-13 (2021-09-14)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Enterprise Server Releases
CVE-2021-46657	2022-01-29	5.5	10.5.12-8 (2021-09-14), 10.4.21-13 (2021-09-14), 10.3.31-13 (2021-09-14), 10.2.40-13 (2021-09-14)
CVE-2021-2372	2021-07-20	4.4	10.5.12-8 (2021-09-14), 10.4.21-13 (2021-09-14), 10.3.31-13 (2021-09-14), 10.2.40-13 (2021-09-14)
CVE-2021-46658	2022-01-29	5.5	10.6.4-1 (2021-08-26)
CVE-2021-2166	2021-04-22	4.9	10.5.10-7 (2021-06-14), 10.4.19-12 (2021-06-14), 10.3.29-12 (2021-06-14), 10.2.38-12 (2021-06-14)
CVE-2021-2154	2021-04-22	4.9	10.5.10-7 (2021-06-14), 10.4.19-12 (2021-06-14), 10.3.29-12 (2021-06-14), 10.2.38-12 (2021-06-14)
CVE-2021-27928	2021-03-19	N/A (Critical) [#1]	10.5.9-6 (2021-03-15), 10.4.18-11 (2021-03-15), 10.3.28-11 (2021-03-15), 10.2.37-11 (2021-03-15)
CVE-2020-14765	2020-10-21	6.5	10.5.8-5 (2020-12-14), 10.4.17-10 (2020-12-14), 10.3.27-10 (2020-12-14), 10.2.36-10 (2020-12-14)
CVE-2022-21427	2022-04-19	4.9	10.5.8-5 (2020-12-14)
CVE-2020-14812	2020-10-21	4.9	10.5.8-5 (2020-12-14), 10.4.17-10 (2020-12-14), 10.3.27-10 (2020-12-14), 10.2.36-10 (2020-12-14)
CVE-2020-14789	2020-10-21	4.9	10.5.8-5 (2020-12-14), 10.4.17-10 (2020-12-14), 10.3.27-10 (2020-12-14), 10.2.36-10 (2020-12-14)
CVE-2020-14776	2020-10-21	4.9	10.5.8-5 (2020-12-14), 10.4.17-10 (2020-12-14), 10.3.27-10 (2020-12-14), 10.2.36-10 (2020-12-14)
CVE-2020-28912	2020-12-24	N/A (Critical) [#1]	10.5.8-5 (2020-12-14), 10.4.17-10 (2020-12-14), 10.3.27-10 (2020-12-14), 10.2.36-10 (2020-12-14)
CVE-2020-15180	2021-05-27	N/A (Critical) [#1]	10.5.6-4 (2020-10-07), 10.4.15-9 (2020-10-07), 10.3.25-9 (2020-10-07), 10.2.34-9 (2020-10-07)
CVE-2021-2022	2021-01-20	4.4	10.5.5-3 (2020-09-08), 10.4.14-8 (2020-09-08), 10.3.24-8 (2020-09-08), 10.2.33-8 (2020-09-08)
CVE-2020-2760	2020-04-15	5.5	10.4.13-7 (2020-06-08), 10.3.23-7 (2020-06-08), 10.2.32-7 (2020-06-08)
CVE-2020-2752	2020-04-15	5.3	10.4.13-7 (2020-06-08), 10.3.23-7 (2020-06-08), 10.2.32-7 (2020-06-08)
CVE-2020-2814	2020-04-15	4.9	10.4.13-7 (2020-06-08), 10.3.23-7 (2020-06-08), 10.2.32-7 (2020-06-08)
CVE-2020-2812	2020-04-15	4.9	10.4.13-7 (2020-06-08), 10.3.23-7 (2020-06-08), 10.2.32-7 (2020-06-08)
CVE-2020-13249	2020-05-20	N/A (Medium) [#1]	10.4.13-7 (2020-06-08), 10.3.23-7 (2020-06-08), 10.2.32-7 (2020-06-08)
CVE-2020-7221	2020-02-04	7.8	10.4.12-6 (2020-03-02)
CVE-2020-2574	2020-01-15	5.9	10.4.12-6 (2020-03-02), 10.3.22-6 (2020-03-02), 10.2.31-6 (2020-03-02)
CVE-2020-2780	2020-04-15	6.5	10.4.10-4 (2019-11-18), 10.3.20-4 (2019-11-18), 10.2.29-4 (2019-11-18)
CVE-2019-2974	2019-10-16	6.5	10.4.10-4 (2019-11-18), 10.3.20-4 (2019-11-18), 10.2.29-4 (2019-11-18)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Enterprise Server Releases
CVE-2019-2938	2019-10-16	4.4	10.4.10-4 (2019-11-18), 10.3.20-4 (2019-11-18), 10.2.29-4 (2019-11-18)
CVE-2019-2805	2019-07-23	6.5	10.4.7-2 (2019-08-19), 10.3.17-2 (2019-08-19), 10.2.26-2 (2019-08-19)
CVE-2019-2740	2019-07-23	6.5	10.4.7-2 (2019-08-19), 10.3.17-2 (2019-08-19), 10.2.26-2 (2019-08-19)
CVE-2019-2758	2019-07-23	5.5	10.4.7-2 (2019-08-19), 10.3.17-2 (2019-08-19), 10.2.26-2 (2019-08-19)
CVE-2019-2739	2019-07-23	5.1	10.4.7-2 (2019-08-19), 10.3.17-2 (2019-08-19), 10.2.26-2 (2019-08-19)
CVE-2019-2737	2019-07-23	4.9	10.4.7-2 (2019-08-19), 10.3.17-2 (2019-08-19), 10.2.26-2 (2019-08-19)
CVE-2021-2007	2021-01-20	3.7	10.4.7-2 (2019-08-19), 10.3.17-2 (2019-08-19), 10.2.26-2 (2019-08-19)
CVE-2020-2922	2020-04-15	3.7	10.4.7-2 (2019-08-19), 10.3.17-2 (2019-08-19), 10.2.26-2 (2019-08-19)

#1:

MariaDB CVEs are assigned a word rating instead of a CVSS base score. See the MariaDB Engineering Policy for details.

What CVEs are listed on this page?

This page provides a cross-reference for all CVEs addressed through a release of MariaDB Enterprise Server.

Enterprise Server Benefits

MariaDB Enterprise Server is an enhanced, hardened, and secured product with:

- Enterprise-grade testing to drive a low defect rate
- Increased code stability through restricted release of new functionality
- Reduced risk from untested or incompatible features through inclusion of only features with level 3 support (Engineering code-level support) from MariaDB Corporation
- Optimized configuration defaults
- Premium features to meet enterprise scaling and operations requirements
- Documentation maintained by MariaDB Corporation

This page is: Copyright © 2026 MariaDB. All rights reserved.

Security Vulnerabilities (CVE) Fixed in MariaDB Community Server

The list is updated continuously. CVEs may be added retroactively to already-released versions, as vulnerabilities can be reported to and assigned by CVE Numbering Authorities (CNAs) after the corresponding fix has been released. As a result, a release may receive additional CVE entries over time even after it has been published.

Table of Fixed Security Vulnerabilities

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2026-49261	—	10.0	11.8.8 (2026-05-27), 11.4.12 (2026-05-27), 10.11.18 (2026-05-27), 10.6.27 (2026-05-27)
CVE-2026-48165	—	8.0	11.8.8 (2026-05-27), 11.4.12 (2026-05-27), 10.11.18 (2026-05-27), 10.6.27 (2026-05-27)
CVE-2026-48163	—	8.0	11.8.8 (2026-05-27), 11.4.12 (2026-05-27), 10.11.18 (2026-05-27), 10.6.27 (2026-05-27)
CVE-2026-44173	—	5.0	12.3.2 (2026-05-28), 11.8.7 (2026-05-18), 11.4.11 (2026-05-18), 10.11.17 (2026-05-18), 10.6.26 (2026-05-18)
CVE-2026-44172	—	5.0	12.3.2 (2026-05-28), 11.8.7 (2026-05-18), 11.4.11 (2026-05-18), 10.11.17 (2026-05-18), 10.6.26 (2026-05-18)
CVE-2026-44171	—	6.3	12.3.2 (2026-05-28), 11.8.7 (2026-05-18), 11.4.11 (2026-05-18), 10.11.17 (2026-05-18), 10.6.26 (2026-05-18)
CVE-2026-44170	—	5.0	12.3.2 (2026-05-28), 11.8.7 (2026-05-18), 11.4.11 (2026-05-18), 10.11.17 (2026-05-18), 10.6.26 (2026-05-18)
CVE-2026-44169	—	4.3	12.3.2 (2026-05-28), 11.8.7 (2026-05-18), 11.4.11 (2026-05-18)
CVE-2026-44168	—	8.0	12.3.2 (2026-05-28), 11.8.7 (2026-05-18), 11.4.11 (2026-05-18), 10.11.17 (2026-05-18), 10.6.26 (2026-05-18)
CVE-2026-3494	2026-03-03	4.3	12.2.2 (2026-02-12), 11.8.6 (2026-02-04), 11.4.10 (2026-02-04), 10.11.16 (2026-02-04), 10.6.25 (2026-02-04)
CVE-2026-32710	2026-03-20	8.6	12.2.2 (2026-02-12), 11.8.6 (2026-02-04), 11.4.10 (2026-02-04)
CVE-2026-35549	2026-04-03	6.5	12.2.2 (2026-02-12), 11.8.6 (2026-02-04), 11.4.10 (2026-02-04)
CVE-2026-34303	2026-04-21	6.5	12.2.2 (2026-02-12), 11.8.6 (2026-02-04), 11.4.10 (2026-02-04), 10.11.16 (2026-02-04)
CVE-2026-21968	2026-01-20	6.5	12.1.2 (2025-11-18), 11.8.4 (2025-11-06), 11.4.9 (2025-11-06), 10.11.15 (2025-11-06), 10.6.24 (2025-11-06)
CVE-2025-13699	2025-12-23	7.0	11.8.4 (2025-11-06), 11.4.9 (2025-11-06), 10.11.15 (2025-11-06), 10.6.24 (2025-11-06)
CVE-2023-52971	2025-03-08	4.9	11.4.6, 10.11.12
CVE-2025-30722	2025-04-15	5.3	11.4.6, 10.11.12, 10.6.22 (2025-05-08), 10.5.29 (2025-05-08)
CVE-2025-30693	2025-04-15	5.5	11.4.6, 10.11.12, 10.6.22 (2025-05-08), 10.5.29 (2025-05-08)
CVE-2025-21490	2025-01-21	4.9	11.7.2 (2025-02-13), 11.4.5 (2025-02-04), 10.6.21 (2025-02-04), 10.5.28 (2025-02-04), 10.11.11 (2025-02-04)
CVE-2024-21096	2024-04-16	4.9	11.4.2 (2024-05-29), 11.2.4 (2024-05-16), 11.1.5 (2024-05-16), 11.0.6 (2024-05-16), 10.6.18 (2024-05-16), 10.5.25 (2024-05-16), 10.11.8 (2024-05-16)
CVE-2023-52970	2025-03-08	4.9	11.4.6, 10.11.12, 10.6.22 (2025-05-08), 10.5.29 (2025-05-08)
CVE-2023-52969	2025-03-08	4.9	11.4.6, 10.11.12, 10.6.22 (2025-05-08), 10.5.29 (2025-05-08)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2023-52968	2025-03-08	4.9	11.4.1 (2024-02-16), 10.11.7 (2024-02-07), 10.6.17 (2024-02-07), 10.5.24 (2024-02-07), 10.4.33 (2024-02-07)
CVE-2023-5157	2023-09-26	7.5	10.8.4 (2022-08-15), 10.7.5 (2022-08-15), 10.6.9 (2022-08-15), 10.5.17 (2022-08-15), 10.4.26 (2022-08-15), 10.3.36 (2022-08-15)
CVE-2023-22084	2023-10-17	4.9	11.2.3 (2024-02-07), 11.2.2 (2023-11-21), 11.1.3 (2023-11-13), 11.0.4 (2023-11-13), 10.6.16 (2023-11-13), 10.5.23 (2023-11-13), 10.4.32 (2023-11-13), 10.11.6 (2023-11-13), 10.10.7 (2023-11-13)
CVE-2022-47015	2023-01-20	N/A (Medium)	10.9.6 (2023-05-10), 10.8.8 (2023-05-10), 10.6.13 (2023-05-10), 10.5.20 (2023-05-10), 10.4.29 (2023-05-10), 10.3.39 (2023-05-10), 10.11.3 (2023-05-10), 10.10.4 (2023-05-10)
CVE-2022-38791	2022-08-27	N/A	10.9.2 (2022-08-22), 10.8.4 (2022-08-15), 10.7.5 (2022-08-15), 10.6.9 (2022-08-15), 10.5.17 (2022-08-15), 10.4.26 (2022-08-15), 10.3.36 (2022-08-15)
CVE-2022-32091	2022-07-01	6.5	10.9.2 (2022-08-22), 10.8.4 (2022-08-15), 10.7.5 (2022-08-15), 10.6.9 (2022-08-15), 10.5.17 (2022-08-15), 10.4.26 (2022-08-15), 10.3.36 (2022-08-15)
CVE-2022-32089	2022-07-01	6.5	10.9.2 (2022-08-22), 10.8.4 (2022-08-15), 10.7.5 (2022-08-15), 10.6.9 (2022-08-15), 10.5.17 (2022-08-15), 10.4.26 (2022-08-15)
CVE-2022-32088	2022-07-01	6.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-32087	2022-07-01	6.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20)
CVE-2022-32086	2022-07-01	6.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20)
CVE-2022-32085	2022-07-01	6.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20)
CVE-2022-32084	2022-07-01	6.5	10.9.2 (2022-08-22), 10.8.4 (2022-08-15), 10.7.5 (2022-08-15), 10.6.9 (2022-08-15), 10.5.17 (2022-08-15), 10.4.26 (2022-08-15), 10.3.36 (2022-08-15)
CVE-2022-32083	2022-07-01	6.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-32082	2022-07-01	6.5	10.9.2 (2022-08-22), 10.8.4 (2022-08-15), 10.7.5 (2022-08-15), 10.6.9 (2022-08-15), 10.5.17 (2022-08-15)
CVE-2022-32081	2022-07-01	6.5	10.9.2 (2022-08-22), 10.8.4 (2022-08-15), 10.7.5 (2022-08-15), 10.6.9 (2022-08-15), 10.5.17 (2022-08-15), 10.4.26 (2022-08-15)
CVE-2022-31624	2022-05-25	6.5	10.6.5 (2021-11-08), 10.5.13 (2021-11-08), 10.4.22 (2021-11-08), 10.3.32 (2021-11-08), 10.2.41 (2021-11-08)
CVE-2022-31623	2022-05-25	N/A	10.6.6 (2022-02-09), 10.5.14 (2022-02-09), 10.4.23 (2022-02-09), 10.3.33 (2022-02-09), 10.2.42 (2022-02-09)
CVE-2022-31622	2022-05-25	N/A	10.6.6 (2022-02-09), 10.5.14 (2022-02-09), 10.4.23 (2022-02-09), 10.3.33 (2022-02-09), 10.2.42 (2022-02-09)
CVE-2022-31621	2022-05-25	N/A	10.6.5 (2021-11-08), 10.5.13 (2021-11-08), 10.4.22 (2021-11-08), 10.3.32 (2021-11-08), 10.2.41 (2021-11-08)
CVE-2022-27457	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20)
CVE-2022-27456	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20)
CVE-2022-27455	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2022-27452	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20)
CVE-2022-27451	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20)
CVE-2022-27449	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20)
CVE-2022-27448	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20)
CVE-2022-27447	2022-04-14	N/A	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20)
CVE-2022-27446	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20)
CVE-2022-27445	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-27444	2022-04-14	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20)
CVE-2022-27387	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-27386	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-27385	2022-04-12	7.5	10.6.5 (2021-11-08), 10.5.13 (2021-11-08), 10.4.22 (2021-11-08), 10.3.32 (2021-11-08)
CVE-2022-27384	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-27383	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-27382	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20)
CVE-2022-27381	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-27380	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-27379	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20)
CVE-2022-27378	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-27377	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-27376	2022-04-12	7.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20)
CVE-2022-24052	2022-02-18	7.0	10.8.1 (2022-02-09), 10.7.2 (2022-02-09), 10.6.6 (2022-02-09), 10.5.14 (2022-02-09), 10.4.23 (2022-02-09), 10.3.33 (2022-02-09), 10.2.42 (2022-02-09)
CVE-2022-24051	2022-02-18	7.0	10.8.1 (2022-02-09), 10.7.2 (2022-02-09), 10.6.6 (2022-02-09), 10.5.14 (2022-02-09), 10.4.23 (2022-02-09), 10.3.33 (2022-02-09), 10.2.42 (2022-02-09)
CVE-2022-24050	2022-02-18	7.0	10.8.1 (2022-02-09), 10.7.2 (2022-02-09), 10.6.6 (2022-02-09), 10.5.14 (2022-02-09), 10.4.23 (2022-02-09), 10.3.33 (2022-02-09), 10.2.42 (2022-02-09)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2022-24048	2022-02-18	7.0	10.8.1 (2022-02-09), 10.7.2 (2022-02-09), 10.6.6 (2022-02-09), 10.5.14 (2022-02-09), 10.4.23 (2022-02-09), 10.3.33 (2022-02-09), 10.2.42 (2022-02-09)
CVE-2022-21595	2022-10-18	4.4	10.7.2 (2022-02-09), 10.6.6 (2022-02-09), 10.5.14 (2022-02-09), 10.4.23 (2022-02-09), 10.3.33 (2022-02-09), 10.2.42 (2022-02-09)
CVE-2022-21451	2022-04-19	7.5	10.5.10 (2021-05-07), 10.4.19 (2021-05-07), 10.3.29 (2021-05-07), 10.2.38 (2021-05-07)
CVE-2022-21427	2022-04-19	4.9	10.5.7 (2020-11-03), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2022-0778	2022-03-15	N/A	10.7.2 (2022-02-09), 10.6.6 (2022-02-09), 10.5.14 (2022-02-09), 10.4.23 (2022-02-09), 10.3.33 (2022-02-09), 10.2.42 (2022-02-09)
CVE-2021-46669	2022-02-01	6.5	10.7.4 (2022-05-20), 10.6.8 (2022-05-20), 10.5.16 (2022-05-20), 10.4.25 (2022-05-20), 10.3.35 (2022-05-20), 10.2.44 (2022-05-20)
CVE-2021-46668	2022-02-01	5.5	10.7.3 (2022-02-12), 10.6.7 (2022-02-12), 10.5.15 (2022-02-12), 10.4.24 (2022-02-12), 10.3.34 (2022-02-12), 10.2.43 (2022-02-12)
CVE-2021-46667	2022-02-01	7.5	10.6.5 (2021-11-08), 10.5.13 (2021-11-08), 10.4.22 (2021-11-08), 10.3.32 (2021-11-08), 10.2.41 (2021-11-08)
CVE-2021-46666	2022-02-01	5.5	10.5.11 (2021-06-23), 10.4.20 (2021-06-23), 10.3.30 (2021-06-23), 10.2.39 (2021-06-23)
CVE-2021-46665	2022-02-01	5.5	10.7.3 (2022-02-12), 10.6.7 (2022-02-12), 10.5.15 (2022-02-12), 10.4.24 (2022-02-12), 10.3.34 (2022-02-12), 10.2.43 (2022-02-12)
CVE-2021-46664	2022-02-01	5.5	10.7.3 (2022-02-12), 10.6.7 (2022-02-12), 10.5.15 (2022-02-12), 10.4.24 (2022-02-12), 10.3.34 (2022-02-12), 10.2.43 (2022-02-12)
CVE-2021-46663	2022-02-01	5.5	10.7.3 (2022-02-12), 10.6.7 (2022-02-12), 10.5.15 (2022-02-12), 10.4.24 (2022-02-12), 10.3.34 (2022-02-12), 10.2.43 (2022-02-12)
CVE-2021-46662	2022-02-01	5.5	10.6.5 (2021-11-08), 10.5.13 (2021-11-08), 10.4.22 (2021-11-08), 10.3.32 (2021-11-08)
CVE-2021-46661	2022-02-01	N/A	10.7.3 (2022-02-12), 10.6.7 (2022-02-12), 10.5.15 (2022-02-12), 10.4.24 (2022-02-12), 10.3.34 (2022-02-12), 10.2.43 (2022-02-12)
CVE-2021-46659	2022-01-29	5.5	10.8.1 (2022-02-09), 10.7.2 (2022-02-09), 10.6.6 (2022-02-09), 10.5.14 (2022-02-09), 10.4.23 (2022-02-09), 10.3.33 (2022-02-09), 10.2.42 (2022-02-09)
CVE-2021-46658	2022-01-29	5.5	10.6.3 (2021-07-06), 10.5.12 (2021-08-06), 10.4.21 (2021-08-06), 10.3.31 (2021-08-06), 10.2.40 (2021-08-06)
CVE-2021-46657	2022-01-29	N/A	10.5.11 (2021-06-23), 10.4.20 (2021-06-23), 10.3.30 (2021-06-23), 10.2.39 (2021-06-23)
CVE-2021-35604	2021-10-20	5.5	10.6.3 (2021-07-06), 10.5.13 (2021-11-08), 10.4.22 (2021-11-08), 10.3.32 (2021-11-08), 10.2.41 (2021-11-08)
CVE-2021-27928	2021-03-19	N/A (Critical)	10.5.9 (2021-02-22), 10.4.18 (2021-02-22), 10.3.28 (2021-02-22), 10.2.37 (2021-02-22)
CVE-2021-2389	2021-07-20	5.9	10.6.4 (2021-08-06) [2 (2021-08-06)], 10.5.12 (2021-08-06), 10.4.21 (2021-08-06), 10.3.31 (2021-08-06), 10.2.40 (2021-08-06)
CVE-2021-2372	2021-07-20	4.4	10.6.4 (2021-08-06), 10.5.12 (2021-08-06), 10.4.21 (2021-08-06), 10.3.31 (2021-08-06), 10.2.40 (2021-08-06)
CVE-2021-2194	2021-04-22	4.9	10.5.7 (2020-11-03), 10.4.16 (2020-11-03), 10.3.26 (2020-11-03), 10.2.35 (2020-11-03)
CVE-2021-2180	2021-04-22	4.9	10.2.38 (2021-05-07)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2021-2174	2021-04-22	4.4	10.2.18 (2018-09-25)
CVE-2021-2166	2021-04-22	4.9	10.5.10 (2021-05-07), 10.4.19 (2021-05-07), 10.3.29 (2021-05-07), 10.2.38 (2021-05-07)
CVE-2021-2154	2021-04-22	4.9	10.5.10 (2021-05-07), 10.4.19 (2021-05-07), 10.3.29 (2021-05-07), 10.2.38 (2021-05-07)
CVE-2021-2144	2021-04-22	7.2	5.5.66 (2019-11-05), 10.4.9 (2019-11-05), 10.3.19 (2019-11-05), 10.2.28 (2019-11-05), 10.1.42 (2019-11-05)
CVE-2021-2032	2021-01-20	4.3	10.0.11 (2014-05-12)
CVE-2021-2022	2021-01-20	4.4	10.5.5 (2020-08-10), 10.4.14 (2020-08-10), 10.3.24 (2020-08-10), 10.2.33 (2020-08-10), 10.1.46 (2020-08-10)
CVE-2021-2011	2021-01-20	5.9	Connector/C 3.0.5, 5.5.61 (2018-07-31), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2021-2007	2021-01-20	3.7	Connector/C 3.1.3, 5.5.65 (2019-07-31), 10.4.7 (2019-07-31), 10.3.17 (2019-07-31), 10.2.26 (2019-07-31), 10.1.41 (2019-07-31)
CVE-2020-7221	2020-02-04	7.8	10.4.12 (2020-01-28)
CVE-2020-2922	2020-04-15	3.7	Connector/C 3.1.3, 5.5.65 (2019-07-31), 10.4.7 (2019-07-31), 10.3.17 (2019-07-31), 10.2.26 (2019-07-31), 10.1.41 (2019-07-31)
CVE-2020-28912	2020-12-24	N/A (Critical)	10.5.7 (2020-11-03), 10.4.16 (2020-11-03), 10.3.26 (2020-11-03), 10.2.35 (2020-11-03), 10.1.48 (2020-11-03)
CVE-2020-2814	2020-04-15	4.9	10.4.13 (2020-05-12), 10.3.23 (2020-05-12), 10.2.32 (2020-05-12), 10.1.45 (2020-05-12)
CVE-2020-2812	2020-04-15	4.9	5.5.68 (2020-05-12), 10.4.13 (2020-05-12), 10.3.23 (2020-05-12), 10.2.32 (2020-05-12), 10.1.45 (2020-05-12)
CVE-2020-2780	2020-04-15	6.5	5.5.66 (2019-11-05), 10.4.9 (2019-11-05), 10.3.19 (2019-11-05), 10.2.28 (2019-11-05), 10.1.42 (2019-11-05)
CVE-2020-2760	2020-04-15	5.5	10.4.13 (2020-05-12), 10.3.23 (2020-05-12), 10.2.32 (2020-05-12)
CVE-2020-2752	2020-04-15	5.3	Connector/C 3.1.8, 5.5.68 (2020-05-12), 10.4.13 (2020-05-12), 10.3.23 (2020-05-12), 10.2.32 (2020-05-12), 10.1.45 (2020-05-12)
CVE-2020-2574	2020-01-15	5.9	Connector/C 3.1.7, 5.5.67 (2020-01-28), 10.4.12 (2020-01-28), 10.3.22 (2020-01-28), 10.2.31 (2020-01-28), 10.1.44 (2020-01-28)
CVE-2020-15180	2021-05-27	N/A (Critical)	10.5.6 (2020-10-07), 10.4.15 (2020-10-07), 10.3.25 (2020-10-07), 10.2.34 (2020-10-07), 10.1.47 (2020-10-07)
CVE-2020-14812	2020-10-21	4.9	10.5.7 (2020-11-03), 10.4.16 (2020-11-03), 10.3.26 (2020-11-03), 10.2.35 (2020-11-03), 10.1.48 (2020-11-03)
CVE-2020-14789	2020-10-21	4.9	10.5.7 (2020-11-03), 10.4.16 (2020-11-03), 10.3.26 (2020-11-03), 10.2.35 (2020-11-03)
CVE-2020-14776	2020-10-21	4.9	10.5.7 (2020-11-03), 10.4.16 (2020-11-03), 10.3.26 (2020-11-03), 10.2.35 (2020-11-03)
CVE-2020-14765	2020-10-21	6.5	10.5.7 (2020-11-03), 10.4.16 (2020-11-03), 10.3.26 (2020-11-03), 10.2.35 (2020-11-03), 10.1.48 (2020-11-03)
CVE-2020-14550	2020-07-15	5.3	Connector/C 3.0.5, 5.5.61 (2018-07-31), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2020-13249	2020-05-20	N/A (Medium)	Connector/C 3.1.8, 10.4.13 (2020-05-12), 10.3.23 (2020-05-12), 10.2.32 (2020-05-12)
CVE-2019-2974	2019-10-16	6.5	5.5.66 (2019-11-05), 10.4.9 (2019-11-05), 10.3.19 (2019-11-05), 10.2.28 (2019-11-05), 10.1.42 (2019-11-05)
CVE-2019-2938	2019-10-16	4.4	10.4.9 (2019-11-05), 10.3.19 (2019-11-05), 10.2.28 (2019-11-05)
CVE-2019-2805	2019-07-23	6.5	5.5.65 (2019-07-31), 10.4.7 (2019-07-31), 10.3.17 (2019-07-31), 10.2.26 (2019-07-31), 10.1.41 (2019-07-31)
CVE-2019-2758	2019-07-23	5.5	10.4.7 (2019-07-31), 10.3.17 (2019-07-31), 10.2.26 (2019-07-31)
CVE-2019-2740	2019-07-23	6.5	5.5.65 (2019-07-31), 10.4.7 (2019-07-31), 10.3.17 (2019-07-31), 10.2.26 (2019-07-31), 10.1.41 (2019-07-31)
CVE-2019-2739	2019-07-23	5.1	5.5.65 (2019-07-31), 10.4.7 (2019-07-31), 10.3.17 (2019-07-31), 10.2.26 (2019-07-31), 10.1.41 (2019-07-31)
CVE-2019-2737	2019-07-23	4.9	5.5.65 (2019-07-31), 10.4.7 (2019-07-31), 10.3.17 (2019-07-31), 10.2.26 (2019-07-31), 10.1.41 (2019-07-31)
CVE-2019-2628	2019-04-23	4.9	10.4.5 (2019-05-21), 10.3.15 (2019-05-14), 10.2.24 (2019-05-09)
CVE-2019-2627	2019-04-23	4.9	5.5.64 (2019-04-29), 10.4.5 (2019-05-21), 10.3.15 (2019-05-14), 10.2.24 (2019-05-09), 10.1.39 (2019-05-02)
CVE-2019-2614	2019-04-23	4.4	5.5.64 (2019-04-29), 10.4.5 (2019-05-21), 10.3.15 (2019-05-14), 10.2.24 (2019-05-09), 10.1.39 (2019-05-02)
CVE-2019-2537	2019-01-16	4.9	10.3.13 (2019-02-21), 10.2.22 (2019-02-11), 10.1.38 (2019-02-06), 10.0.38 (2019-01-31)
CVE-2019-2529	2019-01-16	6.5	5.5.63 (2019-01-30), 10.1.38 (2019-02-06), 10.0.38 (2019-01-31)
CVE-2019-2510	2019-01-16	4.9	10.3.13 (2019-02-21), 10.2.22 (2019-02-11)
CVE-2019-2503	2019-01-16	6.4	5.5.62 (2018-10-26), 10.3.10 (2018-10-04), 10.2.18 (2018-09-25), 10.1.36 (2018-09-08), 10.0.37 (2018-11-01)
CVE-2019-2481	2019-01-16	4.9	5.5.37 (2014-04-17), 10.0.11 (2014-05-12)
CVE-2019-2455	2019-01-16	6.5	5.5.60 (2018-04-23), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-3284	2018-10-17	4.4	10.3.11 (2018-11-20), 10.2.19 (2018-11-13)
CVE-2018-3282	2018-10-17	4.9	5.5.62 (2018-10-26), 10.3.11 (2018-11-20), 10.2.19 (2018-11-13), 10.1.37 (2018-11-02), 10.0.37 (2018-11-01)
CVE-2018-3277	2018-10-17	4.9	10.3.11 (2018-11-20), 10.2.19 (2018-11-13)
CVE-2018-3251	2018-10-17	6.5	10.3.11 (2018-11-20), 10.2.19 (2018-11-13), 10.1.37 (2018-11-02), 10.0.37 (2018-11-01)
CVE-2018-3200	2018-10-17	4.9	10.3.11 (2018-11-20), 10.2.19 (2018-11-13)
CVE-2018-3185	2018-10-17	5.5	10.3.11 (2018-11-20), 10.2.19 (2018-11-13)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2018-3174	2018-10-17	5.3	5.5.62 (2018-10-26), 10.3.11 (2018-11-20), 10.2.19 (2018-11-13), 10.1.37 (2018-11-02), 10.0.37 (2018-11-01)
CVE-2018-3173	2018-10-17	4.9	10.3.11 (2018-11-20), 10.2.19 (2018-11-13)
CVE-2018-3162	2018-10-17	4.9	10.3.11 (2018-11-20), 10.2.19 (2018-11-13)
CVE-2018-3156	2018-10-17	6.5	10.3.11 (2018-11-20), 10.2.19 (2018-11-13), 10.1.37 (2018-11-02), 10.0.37 (2018-11-01)
CVE-2018-3143	2018-10-17	6.5	10.3.11 (2018-11-20), 10.2.19 (2018-11-13), 10.1.37 (2018-11-02), 10.0.37 (2018-11-01)
CVE-2018-3133	2018-10-17	6.5	5.5.59 (2018-01-19), 10.2.12 (2018-01-04), 10.1.30 (2017-12-22), 10.0.34 (2018-01-30)
CVE-2018-3081	2018-07-18	5.0	Connector/C 3.0.5, Connector/C 2.3.7 [2], 5.5.61 (2018-07-31), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-3066	2018-07-18	3.3	5.5.61 (2018-07-31), 10.3.9 (2018-08-15), 10.2.17 (2018-08-14), 10.1.35 (2018-08-07), 10.0.36 (2018-08-01)
CVE-2018-3064	2018-07-18	7.1	10.3.9 (2018-08-15), 10.2.17 (2018-08-14), 10.1.35 (2018-08-07), 10.0.36 (2018-08-01)
CVE-2018-3063	2018-07-18	4.9	5.5.61 (2018-07-31), 10.3.9 (2018-08-15), 10.2.17 (2018-08-14), 10.1.35 (2018-08-07), 10.0.36 (2018-08-01)
CVE-2018-3060	2018-07-18	6.5	10.3.9 (2018-08-15), 10.2.17 (2018-08-14)
CVE-2018-3058	2018-07-18	4.3	5.5.61 (2018-07-31), 10.3.9 (2018-08-15), 10.2.17 (2018-08-14), 10.1.35 (2018-08-07), 10.0.36 (2018-08-01)
CVE-2018-2819	2018-04-19	6.5	5.5.60 (2018-04-23), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2817	2018-04-19	6.5	5.5.60 (2018-04-23), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2813	2018-04-19	4.3	5.5.60 (2018-04-23), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2810	2018-04-19	4.9	10.2.15 (2018-05-17)
CVE-2018-2787	2018-04-19	5.5	10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2786	2018-04-19	5.5	10.2.15 (2018-05-17)
CVE-2018-2784	2018-04-19	6.5	10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2782	2018-04-19	6.5	10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2781	2018-04-19	4.9	5.5.60 (2018-04-23), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2777	2018-04-19	4.9	10.2.15 (2018-05-17)
CVE-2018-2771	2018-04-19	4.4	5.5.60 (2018-04-23), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2018-2767	2018-07-18	3.1	5.5.60 (2018-04-23), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2766	2018-04-19	4.9	10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2761	2018-04-19	5.9	5.5.60 (2018-04-23), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2759	2018-04-19	4.9	10.2.15 (2018-05-17)
CVE-2018-2755	2018-04-19	7.7	5.5.60 (2018-04-23), 10.2.15 (2018-05-17), 10.1.33 (2018-05-09), 10.0.35 (2018-05-03)
CVE-2018-2668	2018-01-18	6.5	5.5.59 (2018-01-19), 10.2.13 (2018-02-13), 10.1.31 (2018-02-06), 10.0.34 (2018-01-30)
CVE-2018-2665	2018-01-18	6.5	5.5.59 (2018-01-19), 10.2.13 (2018-02-13), 10.1.31 (2018-02-06), 10.0.34 (2018-01-30)
CVE-2018-2640	2018-01-18	6.5	5.5.59 (2018-01-19), 10.2.13 (2018-02-13), 10.1.31 (2018-02-06), 10.0.34 (2018-01-30)
CVE-2018-2622	2018-01-18	6.5	5.5.59 (2018-01-19), 10.2.13 (2018-02-13), 10.1.31 (2018-02-06), 10.0.34 (2018-01-30)
CVE-2018-2612	2018-01-18	6.5	10.2.13 (2018-02-13), 10.1.31 (2018-02-06), 10.0.34 (2018-01-30)
CVE-2018-2562	2018-01-18	7.1	5.5.59 (2018-01-19), 10.2.13 (2018-02-13), 10.1.31 (2018-02-06), 10.0.34 (2018-01-30)
CVE-2018-25032	2022-03-25	7.5	Connector/C 3.3.1, Connector/C 3.2.7, Connector/C 3.1.17, 10.9.2 (2022-08-22), 10.8.4 (2022-08-15), 10.7.5 (2022-08-15), 10.6.9 (2022-08-15), 10.5.17 (2022-08-15), 10.4.26 (2022-08-15), 10.3.36 (2022-08-15)
CVE-2017-3653	2017-08-08	3.1	5.5.57 (2017-07-19), 10.2.8 (2017-08-18), 10.1.26 (2017-08-10), 10.0.32 (2017-08-07)
CVE-2017-3651	2017-08-08	4.3	5.5.53 (2016-10-17), 10.1.19 (2016-11-07), 10.0.28 (2016-10-28)
CVE-2017-3641	2017-08-08	4.9	5.5.57 (2017-07-19), 10.2.8 (2017-08-18), 10.1.26 (2017-08-10), 10.0.32 (2017-08-07)
CVE-2017-3636	2017-08-08	5.3	5.5.57 (2017-07-19), 10.2.8 (2017-08-18), 10.1.26 (2017-08-10), 10.0.32 (2017-08-07)
CVE-2017-3600	2017-04-24	6.6	5.5.53 (2016-10-17), 10.1.19 (2016-11-07), 10.0.28 (2016-10-28)
CVE-2017-3464	2017-04-24	4.3	5.5.55 (2017-04-13), 10.2.6 (2017-05-23), 10.1.23 (2017-05-03), 10.0.31 (2017-05-23)
CVE-2017-3456	2017-04-24	4.9	5.5.55 (2017-04-13), 10.2.6 (2017-05-23), 10.1.23 (2017-05-03), 10.0.31 (2017-05-23)
CVE-2017-3453	2017-04-24	6.5	5.5.55 (2017-04-13), 10.2.6 (2017-05-23), 10.1.23 (2017-05-03), 10.0.31 (2017-05-23)
CVE-2017-3318	2017-01-27	4.0	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2017-3317	2017-01-27	4.0	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2017-3313	2017-01-27	4.7	5.5.55 (2017-04-13), 10.2.5 (2017-04-05), 10.1.22 (2017-03-14), 10.0.30 (2017-03-08)
CVE-2017-3312	2017-01-27	6.7	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2017-3309	2017-04-24	7.7	5.5.55 (2017-04-13), 10.2.6 (2017-05-23), 10.1.23 (2017-05-03), 10.0.31 (2017-05-23)
CVE-2017-3308	2017-04-24	7.7	5.5.55 (2017-04-13), 10.2.6 (2017-05-23), 10.1.23 (2017-05-03), 10.0.31 (2017-05-23)
CVE-2017-3302	2017-02-12	N/A	5.5.55 (2017-04-13), 10.2.5 (2017-04-05), 10.1.22 (2017-03-14), 10.0.30 (2017-03-08)
CVE-2017-3291	2017-01-27	6.3	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2017-3265	2017-01-27	5.6	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2017-3258	2017-01-27	6.5	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2017-3257	2017-01-27	6.5	10.2.8 (2017-08-18), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2017-3244	2017-01-27	6.5	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2017-3243	2017-01-27	4.4	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2017-3238	2017-01-27	6.5	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2017-15365	2018-01-25	N/A	10.2.10 (2017-10-31), 10.1.30 (2017-12-22)
CVE-2017-10384	2017-10-19	6.5	5.5.57 (2017-07-19), 10.2.8 (2017-08-18), 10.1.26 (2017-08-10), 10.0.32 (2017-08-07)
CVE-2017-10379	2017-10-19	6.5	5.5.57 (2017-07-19), 10.2.8 (2017-08-18), 10.1.26 (2017-08-10), 10.0.32 (2017-08-07)
CVE-2017-10378	2017-10-19	6.5	5.5.58 (2017-10-18), 10.2.10 (2017-10-31), 10.1.29 (2017-11-14), 10.0.33 (2017-10-30)
CVE-2017-10365	2017-10-19	3.8	10.2.8 (2017-08-18)
CVE-2017-10320	2017-10-19	4.9	10.2.8 (2017-08-18)
CVE-2017-10286	2017-10-19	4.4	10.2.8 (2017-08-18), 10.1.26 (2017-08-10), 10.0.32 (2017-08-07)
CVE-2017-10268	2017-10-19	4.1	5.5.58 (2017-10-18), 10.2.10 (2017-10-31), 10.1.29 (2017-11-14), 10.0.33 (2017-10-30)
CVE-2016-9843	2017-05-23	N/A	5.5.62 (2018-10-26), 10.3.11 (2018-11-20), 10.2.19 (2018-11-13), 10.1.37 (2018-11-02), 10.0.37 (2018-11-01)
CVE-2016-8283	2016-10-25	N/A	5.5.52 (2016-09-13), 10.1.18 (2016-09-30), 10.0.28 (2016-10-28)
CVE-2016-7440	2016-12-13	N/A	5.5.53 (2016-10-17), 10.1.19 (2016-11-07), 10.0.28 (2016-10-28)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2016-6664	2016-12-13	N/A	5.5.54 (2016-12-24), 10.1.21 (2017-01-18), 10.0.29 (2017-01-13)
CVE-2016-6663	2016-12-13	N/A	5.5.52 (2016-09-13), 10.1.18 (2016-09-30), 10.0.28 (2016-10-28)
CVE-2016-6662	2016-09-20	N/A	5.5.51 (2016-08-10), 10.1.17 (2016-08-30), 10.0.27 (2016-08-25)
CVE-2016-5630	2016-10-25	N/A	10.0.27 (2016-08-25)
CVE-2016-5629	2016-10-25	N/A	5.5.52 (2016-09-13), 10.1.18 (2016-09-30), 10.0.28 (2016-10-28)
CVE-2016-5626	2016-10-25	N/A	5.5.52 (2016-09-13), 10.1.18 (2016-09-30), 10.0.28 (2016-10-28)
CVE-2016-5624	2016-10-25	N/A	5.5.52 (2016-09-13), 10.1.18 (2016-09-30), 10.0.28 (2016-10-28)
CVE-2016-5616	2016-10-25	N/A	5.5.52 (2016-09-13), 10.1.18 (2016-09-30), 10.0.28 (2016-10-28)
CVE-2016-5612	2016-10-25	N/A	5.5.51 (2016-08-10), 10.0.27 (2016-08-25)
CVE-2016-5584	2016-10-25	N/A	5.5.53 (2016-10-17), 10.1.19 (2016-11-07), 10.0.28 (2016-10-28)
CVE-2016-5483	2017-04-25	N/A	5.5.53 (2016-10-17), 10.1.19 (2016-11-07), 10.0.28 (2016-10-28)
CVE-2016-5444	2016-07-21	N/A	5.5.49 (2016-04-22), 10.1.14 (2016-05-10), 10.0.25 (2016-04-30)
CVE-2016-5440	2016-07-21	N/A	5.5.50 (2016-06-17), 10.1.15 (2016-07-01), 10.0.26 (2016-06-24)
CVE-2016-3615	2016-07-21	N/A	5.5.50 (2016-06-17), 10.1.15 (2016-07-01), 10.0.26 (2016-06-24)
CVE-2016-3521	2016-07-21	N/A	5.5.50 (2016-06-17), 10.1.15 (2016-07-01), 10.0.26 (2016-06-24)
CVE-2016-3492	2016-10-25	N/A	5.5.52 (2016-09-13), 10.1.18 (2016-09-30), 10.0.28 (2016-10-28)
CVE-2016-3477	2016-07-21	N/A	5.5.50 (2016-06-17), 10.1.15 (2016-07-01), 10.0.26 (2016-06-24)
CVE-2016-3471	2016-07-21	N/A	5.5.46 (2015-10-12), 10.1.9 (2015-11-23), 10.0.22 (2015-10-29)
CVE-2016-3459	2016-07-21	N/A	10.1.14 (2016-05-10), 10.0.25 (2016-04-30)
CVE-2016-3452	2016-07-21	N/A	5.5.49 (2016-04-22), 10.1.14 (2016-05-10), 10.0.25 (2016-04-30)
CVE-2016-2047	2016-01-27	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0668	2016-04-21	N/A	10.1.12 (2016-02-25), 10.0.24 (2016-02-19)
CVE-2016-0666	2016-04-21	N/A	5.5.49 (2016-04-22), 10.1.14 (2016-05-10), 10.0.25 (2016-04-30)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2016-0655	2016-04-21	N/A	10.1.14 (2016-05-10), 10.0.25 (2016-04-30)
CVE-2016-0651	2016-04-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0650	2016-04-21	N/A	5.5.48 (2016-02-11), 10.1.12 (2016-02-25), 10.0.24 (2016-02-19)
CVE-2016-0649	2016-04-21	N/A	5.5.48 (2016-02-11), 10.1.12 (2016-02-25), 10.0.24 (2016-02-19)
CVE-2016-0648	2016-04-21	N/A	5.5.49 (2016-04-22), 10.1.14 (2016-05-10), 10.0.25 (2016-04-30)
CVE-2016-0647	2016-04-21	N/A	5.5.49 (2016-04-22), 10.1.14 (2016-05-10), 10.0.25 (2016-04-30)
CVE-2016-0646	2016-04-21	N/A	5.5.48 (2016-02-11), 10.1.12 (2016-02-25), 10.0.24 (2016-02-19)
CVE-2016-0644	2016-04-21	N/A	5.5.48 (2016-02-11), 10.1.12 (2016-02-25), 10.0.24 (2016-02-19)
CVE-2016-0643	2016-04-21	N/A	5.5.49 (2016-04-22), 10.1.14 (2016-05-10), 10.0.25 (2016-04-30)
CVE-2016-0642	2016-04-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0641	2016-04-21	N/A	5.5.48 (2016-02-11), 10.1.12 (2016-02-25), 10.0.24 (2016-02-19)
CVE-2016-0640	2016-04-21	N/A	5.5.48 (2016-02-11), 10.1.12 (2016-02-25), 10.0.24 (2016-02-19)
CVE-2016-0616	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0610	2016-01-21	N/A	10.1.9 (2015-11-23), 10.0.22 (2015-10-29)
CVE-2016-0609	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0608	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0606	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0600	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0598	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0597	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0596	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0546	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)
CVE-2016-0505	2016-01-21	N/A	5.5.47 (2015-12-10), 10.1.10 (2015-12-24), 10.0.23 (2015-12-18)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2016-0502	2016-01-21	N/A	5.5.32 (2013-07-18), 10.0.4 (2013-08-16)
CVE-2015-7744	2016-01-22	N/A	5.5.46 (2015-10-12), 10.1.9 (2015-11-23), 10.0.22 (2015-10-29)
CVE-2015-4913	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4895	2015-10-21	N/A	10.1.8 (2015-10-17), 10.0.21 (2015-08-06)
CVE-2015-4879	2015-10-21	N/A	5.5.45 (2015-08-06), 10.1.8 (2015-10-17), 10.0.21 (2015-08-06)
CVE-2015-4870	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4866	2015-10-21	N/A	10.1.8 (2015-10-17), 10.0.18 (2015-05-07)
CVE-2015-4864	2015-10-21	N/A	5.5.44 (2015-06-11), 10.1.8 (2015-10-17), 10.0.20 (2015-06-18)
CVE-2015-4861	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4858	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4836	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4830	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4826	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4819	2015-10-21	N/A	5.5.45 (2015-08-06), 10.1.8 (2015-10-17), 10.0.21 (2015-08-06)
CVE-2015-4816	2015-10-21	N/A	5.5.45 (2015-08-06), 10.1.8 (2015-10-17), 10.0.21 (2015-08-06)
CVE-2015-4815	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4807	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4802	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4792	2015-10-21	N/A	5.5.46 (2015-10-12), 10.1.8 (2015-10-17), 10.0.22 (2015-10-29)
CVE-2015-4757	2015-07-16	N/A	5.5.43 (2015-05-01), 10.0.18 (2015-05-07)
CVE-2015-4752	2015-07-16	N/A	5.5.44 (2015-06-11), 10.0.20 (2015-06-18)
CVE-2015-3152	2016-05-16	N/A	5.5.44 (2015-06-11), 10.0.20 (2015-06-18)
CVE-2015-2648	2015-07-16	N/A	5.5.44 (2015-06-11), 10.0.20 (2015-06-18)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2015-2643	2015-07-16	N/A	5.5.44 (2015-06-11), 10.0.20 (2015-06-18)
CVE-2015-2620	2015-07-16	N/A	5.5.44 (2015-06-11), 10.0.20 (2015-06-18)
CVE-2015-2582	2015-07-16	N/A	5.5.44 (2015-06-11), 10.0.20 (2015-06-18)
CVE-2015-2573	2015-04-16	N/A	5.5.42 (2015-02-19), 10.0.17 (2015-02-27)
CVE-2015-2571	2015-04-16	N/A	5.5.43 (2015-05-01), 10.0.18 (2015-05-07)
CVE-2015-2568	2015-04-16	N/A	5.5.42 (2015-02-19), 10.0.17 (2015-02-27)
CVE-2015-2326	2020-01-14	N/A	10.0.18 (2015-05-07)
CVE-2015-2325	2020-01-14	N/A	10.0.18 (2015-05-07)
CVE-2015-0505	2015-04-16	N/A	5.5.43 (2015-05-01), 10.0.18 (2015-05-07)
CVE-2015-0501	2015-04-16	N/A	5.5.43 (2015-05-01), 10.0.18 (2015-05-07)
CVE-2015-0499	2015-04-16	N/A	5.5.43 (2015-05-01), 10.0.18 (2015-05-07)
CVE-2015-0441	2015-04-16	N/A	5.5.42 (2015-02-19), 10.0.17 (2015-02-27)
CVE-2015-0433	2015-04-16	N/A	5.5.42 (2015-02-19), 10.0.17 (2015-02-27)
CVE-2015-0432	2015-01-21	N/A	5.5.41 (2014-12-21), 10.0.16 (2015-01-27)
CVE-2015-0411	2015-01-21	N/A	5.5.41 (2014-12-21), 10.0.16 (2015-01-27)
CVE-2015-0391	2015-01-21	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2015-0382	2015-01-21	N/A	5.5.41 (2014-12-21), 10.0.16 (2015-01-27)
CVE-2015-0381	2015-01-21	N/A	5.5.41 (2014-12-21), 10.0.16 (2015-01-27)
CVE-2015-0374	2015-01-21	N/A	5.5.41 (2014-12-21), 10.0.16 (2015-01-27)
CVE-2014-8964	2014-12-16	N/A	10.0.18 (2015-05-07)
CVE-2014-6568	2015-01-21	N/A	5.5.41 (2014-12-21), 10.0.16 (2015-01-27)
CVE-2014-6564	2014-10-15	N/A	10.0.13 (2014-08-11)
CVE-2014-6559	2014-10-15	N/A	5.5.40 (2014-10-09), 10.0.15 (2014-11-25)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2014-6555	2014-10-15	N/A	5.5.40 (2014-10-09), 10.0.15 (2014-11-25)
CVE-2014-6551	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-6530	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-6520	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-6507	2014-10-15	N/A	5.5.40 (2014-10-09), 10.0.15 (2014-11-25)
CVE-2014-6505	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-6500	2014-10-15	N/A	5.5.40 (2014-10-09), 10.0.15 (2014-11-25)
CVE-2014-6496	2014-10-15	N/A	5.5.40 (2014-10-09), 10.0.15 (2014-11-25)
CVE-2014-6495	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-6494	2014-10-15	N/A	5.5.40 (2014-10-09), 10.0.15 (2014-11-25)
CVE-2014-6491	2014-10-15	N/A	5.5.40 (2014-10-09), 10.0.15 (2014-11-25)
CVE-2014-6489	2014-10-15	N/A	10.0.13 (2014-08-11)
CVE-2014-6484	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-6478	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-6474	2014-10-15	N/A	10.0.13 (2014-08-11)
CVE-2014-6469	2014-10-15	N/A	5.5.40 (2014-10-09), 10.0.15 (2014-11-25)
CVE-2014-6464	2014-10-15	N/A	5.5.40 (2014-10-09), 10.0.15 (2014-11-25)
CVE-2014-6463	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-4287	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-4274	2014-10-15	N/A	5.5.39 (2014-08-05), 10.0.13 (2014-08-11)
CVE-2014-4260	2014-07-17	N/A	5.5.38 (2014-06-09), 10.0.12 (2014-06-16)
CVE-2014-4258	2014-07-17	N/A	5.5.38 (2014-06-09), 10.0.12 (2014-06-16)
CVE-2014-4243	2014-07-17	N/A	5.5.36 (2014-02-25), 10.0.9 (2014-03-10)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2014-4207	2014-07-17	N/A	5.5.38 (2014-06-09), 10.0.12 (2014-06-16)
CVE-2014-3470	2014-06-05	N/A	10.0.13 (2014-08-11)
CVE-2014-2494	2014-07-17	N/A	5.5.38 (2014-06-09), 10.0.12 (2014-06-16)
CVE-2014-2440	2014-04-16	N/A	5.5.37 (2014-04-17), 10.0.11 (2014-05-12)
CVE-2014-2438	2014-04-16	N/A	5.5.36 (2014-02-25), 10.0.9 (2014-03-10)
CVE-2014-2436	2014-04-16	N/A	5.5.37 (2014-04-17), 10.0.11 (2014-05-12)
CVE-2014-2432	2014-04-16	N/A	5.5.36 (2014-02-25), 10.0.9 (2014-03-10)
CVE-2014-2431	2014-04-16	N/A	5.5.37 (2014-04-17), 10.0.11 (2014-05-12)
CVE-2014-2430	2014-04-16	N/A	5.5.37 (2014-04-17), 10.0.11 (2014-05-12)
CVE-2014-2419	2014-04-16	N/A	5.5.36 (2014-02-25), 10.0.9 (2014-03-10)
CVE-2014-0437	2014-01-15	N/A	5.5.35 (2014-01-29), 10.0.8 (2014-02-10)
CVE-2014-0420	2014-01-15	N/A	5.5.35 (2014-01-29), 10.0.8 (2014-02-10)
CVE-2014-0412	2014-01-15	N/A	5.5.35 (2014-01-29), 10.0.8 (2014-02-10)
CVE-2014-0402	2014-01-15	N/A	5.5.34 (2013-11-21), 10.0.7 (2013-12-27)
CVE-2014-0401	2014-01-15	N/A	5.5.35 (2014-01-29), 10.0.8 (2014-02-10)
CVE-2014-0393	2014-01-15	N/A	5.5.34 (2013-11-21), 10.0.7 (2013-12-27)
CVE-2014-0386	2014-01-15	N/A	5.5.34 (2013-11-21), 10.0.7 (2013-12-27)
CVE-2014-0384	2014-04-15	N/A	5.5.36 (2014-02-25), 10.0.9 (2014-03-10)
CVE-2014-0224	2014-06-05	N/A	10.0.13 (2014-08-11)
CVE-2014-0221	2014-06-05	N/A	10.0.13 (2014-08-11)
CVE-2014-0198	2014-05-06	N/A	10.0.13 (2014-08-11)
CVE-2014-0195	2014-06-05	N/A	10.0.13 (2014-08-11)
CVE-2013-5908	2014-01-15	N/A	5.5.35 (2014-01-29), 10.0.8 (2014-02-10)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2013-5891	2014-01-15	N/A	5.5.34 (2013-11-21), 10.0.7 (2013-12-27)
CVE-2013-5807	2013-10-16	N/A	5.5.33 (2013-09-17), 10.0.5 (2013-11-07)
CVE-2013-3839	2013-10-16	N/A	5.5.33 (2013-09-17), 10.0.5 (2013-11-07)
CVE-2013-3812	2013-07-17	N/A	5.5.32 (2013-07-18), 10.0.4 (2013-08-16)
CVE-2013-3809	2013-07-17	N/A	5.5.32 (2013-07-18), 10.0.4 (2013-08-16)
CVE-2013-3808	2013-07-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-3805	2013-07-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-3804	2013-07-17	N/A	5.5.32 (2013-07-18), 10.0.4 (2013-08-16)
CVE-2013-3802	2013-07-17	N/A	5.5.32 (2013-07-18), 10.0.4 (2013-08-16)
CVE-2013-3801	2013-07-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-3794	2013-07-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-3793	2013-07-17	N/A	5.5.32 (2013-07-18), 10.0.4 (2013-08-16)
CVE-2013-3783	2013-07-17	N/A	5.5.32 (2013-07-18), 10.0.4 (2013-08-16)
CVE-2013-2392	2013-04-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-2391	2013-04-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-2389	2013-04-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-2378	2013-04-17	N/A	5.5.30 (2013-03-12), 10.0.2 (2013-04-24)
CVE-2013-2376	2013-04-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-2375	2013-04-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-1861	2013-03-28	N/A	5.5.32 (2013-07-18), 10.0.4 (2013-08-16)
CVE-2013-1555	2013-04-17	N/A	5.5.30 (2013-03-12), 10.0.2 (2013-04-24)
CVE-2013-1552	2013-04-17	N/A	5.5.30 (2013-03-12), 10.0.2 (2013-04-24)
CVE-2013-1548	2013-04-17	N/A	5.5.27 (2012-09-07), 5.1.66 (2012-11-29)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2013-1544	2013-04-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-1532	2013-04-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-1531	2013-04-17	N/A	5.5.29 (2013-01-30), 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2013-1526	2013-04-17	N/A	5.5.30 (2013-03-12), 10.0.2 (2013-04-24)
CVE-2013-1523	2013-04-17	N/A	5.5.30 (2013-03-12), 10.0.2 (2013-04-24)
CVE-2013-1521	2013-04-17	N/A	5.5.30 (2013-03-12), 10.0.2 (2013-04-24)
CVE-2013-1512	2013-04-17	N/A	5.5.30 (2013-03-12), 10.0.2 (2013-04-24)
CVE-2013-1511	2013-04-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-1506	2013-04-17	N/A	5.5.30 (2013-03-12), 10.0.2 (2013-04-24)
CVE-2013-1502	2013-04-17	N/A	5.5.31 (2013-05-23), 10.0.3 (2013-06-11)
CVE-2013-0389	2013-01-17	N/A	5.5.29 (2013-01-30), 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2013-0386	2013-01-17	N/A	5.5.29 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2013-0385	2013-01-17	N/A	5.5.29 (2013-01-30), 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2013-0384	2013-01-17	N/A	5.5.29 (2013-01-30), 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2013-0383	2013-01-17	N/A	5.5.29 (2013-01-30), 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2013-0375	2013-01-17	N/A	5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30)
CVE-2013-0371	2013-01-17	N/A	5.5.29 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2013-0368	2013-01-17	N/A	5.5.29 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2013-0367	2013-01-17	N/A	5.5.29 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2012-5627	2013-10-01	N/A	5.5.29 (2013-01-30) [2 (2013-01-30)], 5.3.12 (2013-01-30) [2 (2013-01-30)], 5.2.14 (2013-01-30) [2 (2013-01-30)], 10.0.1 (2013-02-06)
CVE-2012-5615	2012-12-03	N/A	5.5.29 (2013-01-30) [2 (2013-01-30)], 5.3.12 (2013-01-30) [2 (2013-01-30)], 5.2.14 (2013-01-30) [2 (2013-01-30)], 10.0.13 (2014-08-11), 10.0.1 (2013-02-06)
CVE-2012-5614	2012-12-03	N/A	5.5.30 (2013-03-12), 10.0.2 (2013-04-24)
CVE-2012-5612	2012-12-03	N/A	5.5.29 (2013-01-30) [2 (2013-01-30)], 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2012-5611	2012-12-03	N/A	5.5.29 (2013-01-30), 5.5.28a (2012-11-29), 5.3.12 (2013-01-30), 5.3.11 (2012-11-29), 5.2.14 (2013-01-30), 5.2.13 (2012-11-29), 5.1.67 (2013-01-30), 5.1.66 (2012-11-29), 10.0.1 (2013-02-06)
CVE-2012-5096	2013-01-17	N/A	5.5.29 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2012-5060	2013-01-17	N/A	5.5.28 (2012-10-22), 5.1.66 (2012-11-29)
CVE-2012-4414	2013-01-22	N/A	5.5.27 (2012-09-07) [2 (2012-09-07)], 5.3.8 (2012-08-28) [2 (2012-08-28)], 5.2.13 (2012-11-29) [2 (2012-11-29)], 5.1.66 (2012-11-29) [2 (2012-11-29)], 10.0.0 (2012-11-12) [2 (2012-11-12)]
CVE-2012-3197	2012-10-17	N/A	5.5.27 (2012-09-07), 5.1.66 (2012-11-29)
CVE-2012-3180	2012-10-17	N/A	5.5.28 (2012-10-22), 5.1.66 (2012-11-29)
CVE-2012-3177	2012-10-17	N/A	5.5.28 (2012-10-22), 5.5.27 (2012-09-07), 5.1.66 (2012-11-29)
CVE-2012-3173	2012-10-17	N/A	5.5.27 (2012-09-07), 5.1.66 (2012-11-29)
CVE-2012-3167	2012-10-17	N/A	5.5.27 (2012-09-07), 5.1.66 (2012-11-29)
CVE-2012-3166	2012-10-17	N/A	5.5.27 (2012-09-07), 5.1.66 (2012-11-29)
CVE-2012-3163	2012-10-17	N/A	5.5.27 (2012-09-07), 5.1.66 (2012-11-29)
CVE-2012-3160	2012-10-16	N/A	5.5.28 (2012-10-22), 5.1.66 (2012-11-29)
CVE-2012-3158	2012-10-16	N/A	5.5.27 (2012-09-07), 5.1.66 (2012-11-29)
CVE-2012-3150	2012-10-16	N/A	5.5.27 (2012-09-07), 5.1.66 (2012-11-29)
CVE-2012-2750	2012-08-17	N/A	5.5.23 (2012-04-11)
CVE-2012-1757	2012-07-17	N/A	5.5.24 (2012-05-31)
CVE-2012-1756	2012-07-17	N/A	5.5.24 (2012-05-31)
CVE-2012-1735	2012-07-17	N/A	5.5.24 (2012-05-31)
CVE-2012-1734	2012-07-17	N/A	5.5.24 (2012-05-31), 5.1.66 (2012-11-29)
CVE-2012-1705	2013-01-17	N/A	5.5.29 (2013-01-30), 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2012-1703	2012-05-03	N/A	5.5.22 (2012-03-29), 5.1.62 (2012-04-06)
CVE-2012-1702	2013-01-17	N/A	5.5.29 (2013-01-30), 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)

CVE ID (with cve.org link)	CVE Published	CVSS base score (v3.1)	Community Server Releases
CVE-2012-1697	2012-05-03	N/A	5.5.22 (2012-03-29)
CVE-2012-1690	2012-05-03	N/A	5.5.22 (2012-03-29), 5.1.62 (2012-04-06)
CVE-2012-1689	2012-07-17	N/A	5.5.23 (2012-04-11), 5.1.66 (2012-11-29)
CVE-2012-1688	2012-05-03	N/A	5.5.22 (2012-03-29), 5.1.62 (2012-04-06)
CVE-2012-0578	2013-01-17	N/A	5.5.29 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2012-0574	2013-01-17	N/A	5.5.29 (2013-01-30), 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2012-0572	2013-01-17	N/A	5.5.29 (2013-01-30), 5.3.12 (2013-01-30), 5.2.14 (2013-01-30), 5.1.67 (2013-01-30), 10.0.1 (2013-02-06)
CVE-2012-0540	2012-07-17	N/A	5.5.24 (2012-05-31), 5.1.66 (2012-11-29)
CVE-2010-5298	2014-04-14	N/A	10.0.13 (2014-08-11)
CVE-2009-4484	2009-12-30	N/A	5.1.42 (2010-02-01)
CVE-2005-0004	2005-01-20	N/A	5.5.66 (2019-11-05)

CVEs without specific version numbers:

The following CVEs were fixed in Community Server 5.1 and/or Community Server 5.5 as indicated, but the fix is not tied to a specific MariaDB version.

CVE ID (with cve.org link)	CVE Published	CVSS base score	Community Server Releases
CVE-2012-0113	2012-01-18	N/A	5.1, 5.5
CVE-2011-2262	2012-01-18	N/A	5.1, 5.5
CVE-2012-0116	2012-01-18	N/A	5.1, 5.5
CVE-2012-0118	2012-01-18	N/A	5.1, 5.5
CVE-2012-0496	2012-01-18	N/A	5.5
CVE-2012-0087	2012-01-18	N/A	5.1, 5.1
CVE-2012-0101	2012-01-18	N/A	5.1, 5.1
CVE-2012-0102	2012-01-18	N/A	5.1, 5.1
CVE-2012-0115	2012-01-18	N/A	5.1, 5.5
CVE-2012-0119	2012-01-18	N/A	5.1, 5.5
CVE-2012-0120	2012-01-18	N/A	5.1, 5.5
CVE-2012-0484	2012-01-18	N/A	5.1, 5.5
CVE-2012-0485	2012-01-18	N/A	5.1, 5.5
CVE-2012-0486	2012-01-18	N/A	5.5
CVE-2012-0487	2012-01-18	N/A	5.5

CVE ID (with cve.org link)	CVE Published	CVSS base score	Community Server Releases
CVE-2012-0488	2012-01-18	N/A	5.5
CVE-2012-0489	2012-01-18	N/A	5.5
CVE-2012-0490	2012-01-18	N/A	5.1, 5.5
CVE-2012-0491	2012-01-18	N/A	5.5
CVE-2012-0495	2012-01-18	N/A	5.5
CVE-2012-0112	2012-01-18	N/A	5.1, 5.5
CVE-2012-0117	2012-01-18	N/A	5.5
CVE-2012-0114	2012-01-18	N/A	5.1, 5.5
CVE-2012-0492	2012-01-18	N/A	5.1, 5.5
CVE-2012-0493	2012-01-18	N/A	5.5
CVE-2012-0075	2012-01-18	N/A	5.1, 5.5
CVE-2012-0494	2012-01-18	N/A	5.5

Lists for individual MariaDB Community Server Series

Separate lists of CVEs fixed in specific MariaDB Community Server series are maintained on their respective pages:

- Community Server 12.3
- Community Server 12.2
- Community Server 12.1
- Community Server 12.0
- Community Server 11.8
- Community Server 11.7
- Community Server 11.6
- Community Server 11.5
- Community Server 11.4
- Community Server 11.3
- Community Server 11.2
- Community Server 11.1
- Community Server 11.0
- Community Server 10.11
- Community Server 10.10
- Community Server 10.9
- Community Server 10.8
- Community Server 10.7
- Community Server 10.6
- Community Server 10.5
- Community Server 10.4
- Community Server 10.3
- Community Server 10.2
- Community Server 10.1
- Community Server 10.0
- Community Server 5.5
- Community Server 5.3

- Community Server 5.2
- Community Server 5.1

This page is licensed: CC BY-SA / Gnu FDL

Security Vulnerabilities (CVE) Fixed in Oracle MySQL That Did Not Exist in MariaDB

About

CVE® stands for *Common Vulnerabilities and Exposures*. It is a publicly available and free-to-use database of known software vulnerabilities maintained at https://cve.mitre.org/.

CPU stands for *Critical Patch Update*. Oracle publishes Critical Patch Update Advisories four times a year, on the Tuesday closest to the 17th day of January, April, July and October. MySQL vulnerabilities are included in these CPU Advisories.

Some vulnerabilities found in MySQL apply to MariaDB as well, they are listed on the Security Vulnerabilities Fixed in MariaDB Enterprise Server and Security Vulnerabilities Fixed in MariaDB Community Server pages.

Other vulnerabilities found in MySQL do not apply to MariaDB.

This page lists all CVEs that were fixed in MySQL and mentioned in Oracle CPU Advisories, but that — *to the best of our knowledge* — were never present in MariaDB.

Full List of CVEs Fixed in Oracle MySQL That Never Existed in MariaDB

- ▶ June 2026
- ▶ April 2026
- ▶ January 2026
- ▶ October 2025
- ▶ July 2025
- ▶ April 2025
- ▶ January 2025
- ▶ October 2024
- ▶ July 2024
- ▶ April 2024
- ▶ January 2024
- ▶ October 2023
- ▶ July 2023
- ▶ April 2023
- ▶ January 2023
- ▶ October 2022
- ▶ July 2022
- ▶ April 2022
- ▶ January 2022
- ▶ October 2021
- ▶ July 2021
- ▶ April 2021
- ▶ January 2021
- ▶ October 2020
- ▶ July 2020
- ▶ April 2020
- ▶ January 2020
- ▶ October 2019

- July 2019
- April 2019
- January 2019
- October 2018
- July 2018
- April 2018
- January 2018
- October 2017
- July 2017
- April 2017
- January 2017
- October 2016
- July 2016
- April 2016
- January 2016
- October 2015
- July 2015
- April 2015
- October 2015
- July 2014
- April 2014
- January 2014
- October 2013
- July 2013
- April 2013
- October 2012
- April 2012

This page is licensed: CC BY-SA / Gnu FDL

Architecture

MariaDB Server's architecture.

See Server Constraints.

Details the various constraints within MariaDB Server, including primary keys, unique keys, and foreign keys.

See Topologies.

Provides information on different MariaDB Server deployment topologies, including standalone, replication, and clustered configurations.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Topologies

See Topologies Overview.

MariaDB offers varied deployment topologies by workload and technology, each named and diagrammed with benefits listed. Custom configurations are also supported.

See Compatibility.

Operating-system support for MariaDB products and versions, with links to the engineering policies that list supported platforms.

See Columnstore Object Storage.

Deploy the MariaDB ColumnStore object-storage topology, a columnar OLAP store that keeps data on S3-compatible object storage.

See ColumnStore Shared Local Storage.

Deploy the MariaDB ColumnStore shared-local-storage topology, a columnar OLAP store for analytical workloads.

See Galera Cluster.

Deploy the Galera Cluster topology for MariaDB — synchronous, multi-primary replication for high availability.

See HTAP.

Deploy the HTAP topology, pairing MariaDB row storage with ColumnStore columnar storage for mixed transactional and analytical workloads.

See Installing Primary/Replica.

This page explains how to set up a standard Primary/Replica replication topology for MariaDB Enterprise Server.

See Single Node Topologies.

Install and configure single-node MariaDB deployments, including a single- node columnar store with optional S3-compatible object storage.

See Spider Storage Engine.

Scale MariaDB horizontally with the Spider storage engine, which shards tables across remote nodes through virtual federated tables.

See MariaDB Enterprise Spider Topologies.

Explore MariaDB Enterprise Spider topologies with MaxScale. This section details how it integrates with Spider to manage & route traffic efficiently across sharded & distributed database environments.

See Spider Federated.

Deploy Spider Federated Topology

See Spider Sharded.

Deploy Spider Sharded Topology

This page is: Copyright © 2026 MariaDB. All rights reserved.

Topologies Overview

MariaDB products can be deployed in many different topologies. The topologies described in this section are representative. MariaDB products can be deployed to form other topologies, leverage advanced product capabilities, or combine the capabilities of multiple topologies.

Topologies are the arrangement of nodes and links to achieve a purpose. This documentation describes a few of the many topologies that can be deployed using MariaDB database products.

We group topologies by workload (transactional, analytical, hybrid) and technologies (Enterprise Spider). Single-node topologies are listed separately.

To help you select the correct topology:

- Each topology is named and this name is used consistently throughout the documentation.
- A thumbnail diagram provides a small-scale summary of the topology's architecture.
- Finally, we provide a list of the benefits of the topology.

Although multiple topologies are listed on this page, the listed topologies are not the only options. MariaDB products are flexible, configurable, and extensible, so it possible to deploy different topologies that combine the capabilities of multiple topologies listed on this page. The topologies listed on this page are primarily intended to be representative of the most commonly requested use cases.

Transactional (OLTP)

Primary/Replica Topology

Diagram	Features

Galera Cluster Topology

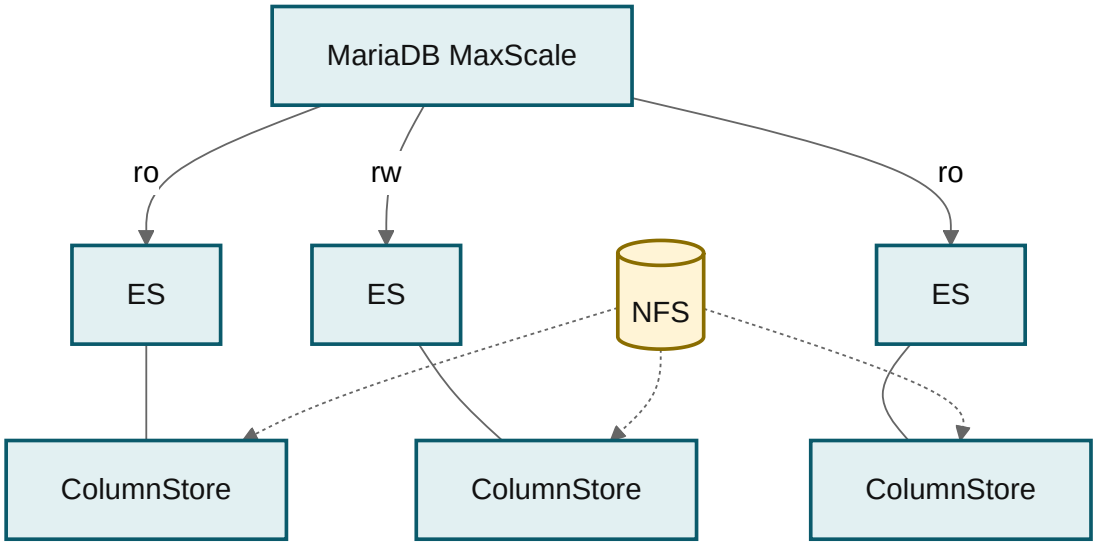
Diagram	Features

Analytical (OLAP, Data Warehousing, DSS)

ColumnStore Object Storage Topology

Diagram	Features

ColumnStore Shared Local Storage Topology



MaxScale routes to a three-node ColumnStore cluster sharing NFS storage.

Columnar storage engine with shared local storage

- Highly available
- Automatic failover via MaxScale and CMAPI
- Scales reads via MaxScale
- Bulk data import
- Enterprise Server 10.5, Enterprise ColumnStore 5, MaxScale 2.5
- Enterprise Server 10.6, Enterprise ColumnStore 23.02, MaxScale 22.08

Hybrid Workloads

HTAP Topology

Diagram	Features

Spider Topologies

Spider Federated Topology

Diagram	Features

Spider Sharded Topology

Diagram	Features

Single Node Topologies

- Deploy Single Node Topologies

This page is: Copyright © 2026 MariaDB. All rights reserved.

Compatibility

Operating System support varies by product and product version. Operating System support is detailed in MariaDB plc Engineering Policies.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Columnstore Object Storage

Overview

This procedure describes the deployment of the ColumnStore Object Storage topology with MariaDB Enterprise Server 10.5, MariaDB Enterprise ColumnStore 5, and MariaDB MaxScale 2.5.

MariaDB Enterprise ColumnStore 5 is a columnar storage engine for MariaDB Enterprise Server 10.5. Enterprise ColumnStore is suitable for Online Analytical Processing (OLAP) workloads.

This procedure has 9 steps, which are executed in sequence.

This procedure represents basic product capability and deploys 3 Enterprise ColumnStore nodes and 1 MaxScale node.

This page provides an overview of the topology, requirements, and deployment procedures.

Please read and understand this procedure before executing.

Procedure Steps

Step	Description
Step 1	Prepare ColumnStore Nodes
Step 2	Configure Shared Local Storage
Step 3	Install MariaDB Enterprise Server
Step 4	Start and Configure MariaDB Enterprise Server
Step 5	Test MariaDB Enterprise Server
Step 6	Install MariaDB MaxScale
Step 7	Start and Configure MariaDB MaxScale
Step 8	Test MariaDB MaxScale
Step 9	Import Data

Support

Customers can obtain support by submitting a support case.

Components

The following components are deployed during this procedure:

Component	Function
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.
MariaDB MaxScale	Database proxy that extends the availability, scalability, and security of MariaDB Enterprise Servers

MariaDB Enterprise Server Components

Component	Description
MariaDB Enterprise ColumnStore	• Columnar storage engine • Highly available • Optimized for Online Analytical Processing (OLAP) workloads • Scalable query execution • Cluster Management API (CMAPI) provides a REST API for multi-node administration

MariaDB MaxScale Components

Component	Description
Listener	Listens for client connections to MaxScale then passes them to the router service
MariaDB Monitor	Tracks changes in the state of MariaDB Enterprise Servers.
Read Connection Router	Routes connections from the listener to any available Enterprise ColumnStore node
Read/Write Split Router	Routes read operations from the listener to any available Enterprise ColumnStore node, and routes write operations from the listener to a specific server that MaxScale uses as the primary server
Server Module	Connection configuration in MaxScale to an Enterprise ColumnStore node

Topology

The MariaDB Enterprise ColumnStore topology with Object Storage delivers production analytics with high availability, fault tolerance, and limitless data storage by leveraging S3-compatible storage.

The topology consists of:

- One or more MaxScale nodes
- An odd number of ColumnStore nodes (minimum of 3) running ES, Enterprise ColumnStore, and CMAPI

The MaxScale nodes:

- Monitor the health and availability of each ColumnStore node using the MariaDB Monitor (mariadbmon)
- Accept client and application connections
- Route queries to ColumnStore nodes using the Read/Write Split Router (readwritesplit)

The ColumnStore nodes:

- Receive queries from MaxScale
- Execute queries
- Use S3-compatible object storage for data
- Use shared local storage for the Storage Manager directory

Requirements

These requirements are for the ColumnStore Object Storage topology when deployed with MariaDB Enterprise Server 10.5, MariaDB Enterprise ColumnStore 5, and MariaDB MaxScale 2.5.

- Node Count
- Operating System
- Minimum Hardware Requirements
- Recommended Hardware Requirements
- Storage Requirements

- S3-Compatible Object Storage Requirements
- Preferred Object Storage Providers: Cloud
- Preferred Object Storage Providers: Hardware
- Shared Local Storage Directories
- Shared Local Storage Options
- Recommended Storage Options

Node Count

- MaxScale nodes, 1 or more are required.
- Enterprise ColumnStore nodes, 3 or more are required for high availability. You should always have an odd number of nodes in a multi-node ColumnStore deployment to avoid split brain scenarios.

Operating System

In alignment to the enterprise lifecycle, the ColumnStore Object Storage topology with MariaDB Enterprise Server 10.5, MariaDB Enterprise ColumnStore 5, and MariaDB MaxScale 2.5 is provided for:

- CentOS Linux 7 (x86_64)
- Debian 10 (x86_64)
- Red Hat Enterprise Linux 7 (x86_64)
- Red Hat Enterprise Linux 8 (x86_64)
- Ubuntu 18.04 LTS (x86_64)
- Ubuntu 20.04 LTS (x86_64)

Minimum Hardware Requirements

MariaDB Enterprise ColumnStore's minimum hardware requirements are not intended for production environments, but the minimum hardware requirements can be appropriate for development and test environments. For production environments, see the recommended hardware requirements instead.

The minimum hardware requirements are:

Component	CPU	Memory
MaxScale node	4+ cores	4+ GB
Enterprise ColumnStore node	4+ cores	4+ GB

MariaDB Enterprise ColumnStore will refuse to start if the system has less than 3 GB of memory.

If Enterprise ColumnStore is started on a system with less memory, the following error message will be written to the ColumnStore system log called crit.log:

```
Apr 30 21:54:35 a1ebc96a2519 PrimProc[1004]: 35.668435 |0|0|0| C 28 CAL0000: Error total memory available
is less than 3GB.
```

And the following error message will be raised to the client:

```
ERROR 1815 (HY000): Internal error: System is not ready yet. Please try again.
```

Recommended Hardware Requirements

MariaDB Enterprise ColumnStore's recommended hardware requirements are intended for production analytics.

The recommended hardware requirements are:

Component	CPU	Memory
MaxScale node	8+ cores	16+ GB
Enterprise ColumnStore node	64+ cores	128+ GB

Storage Requirements

The ColumnStore Object Storage topology requires the following storage types:

Storage Type	Description
S3-Compatible Object Storage	The ColumnStore Object Storage topology uses S3-compatible object storage to store data.
Shared Local Storage	The ColumnStore Object Storage topology uses shared local storage for the Storage Manager directory to store metadata.

S3-Compatible Object Storage Requirements

The ColumnStore Object Storage topology uses S3-compatible object storage to store data.

Many S3-compatible object storage services exist. MariaDB Corporation cannot make guarantees about all S3-compatible object storage services, because different services provide different functionality.

For the preferred S3-compatible object storage providers that provide cloud and hardware solutions, see the following sections:

- Cloud
- Hardware

The use of non-cloud and non-hardware providers is at your own risk.

If you have any questions about using specific S3-compatible object storage with MariaDB Enterprise ColumnStore, contact us.

Preferred Object Storage Providers: Cloud

- Amazon Web Services (AWS) S3
- Google Cloud Storage
- Azure Storage
- Alibaba Cloud Object Storage Service

Preferred Object Storage Providers: Hardware

- Cloudian HyperStore
- Cohesity S3
- Dell EMC
- IBM Cloud Object Storage
- Seagate Lyve Rack
- Quantum ActiveScale

Shared Local Storage Directories

The ColumnStore Object Storage topology uses shared local storage for the Storage Manager directory to store metadata.

The Storage Manager directory is located at the following path by default:

- `/var/lib/columnstore/storagemanager`

Shared Local Storage Options

The most common shared local storage options for the ColumnStore Object Storage topology are:

Shared Local Storage	Common Usage	Description
EBS (Elastic Block Store) Multi-Attach	AWS	• EBS is a high-performance block-storage service for AWS (Amazon Web Services). • EBS Multi-Attach allows an EBS volume to be attached to multiple instances in AWS. Only clustered file systems, such as GFS2, are supported. • For deployments in AWS, EBS Multi-Attach is a recommended option for the Storage Manager directory, and Amazon S3 storage is the recommended option for data.
EFS (Elastic File System)	AWS	• EFS is a scalable, elastic, cloud-native NFS file system for AWS (Amazon Web Services). • For deployments in AWS, EFS is a recommended option for the Storage Manager directory, and Amazon S3 storage is the recommended option for data. EFS is a scalable, elastic, cloud-native NFS file system for AWS (Amazon Web Services).
Filestore	GCP	• Filestore is high-performance, fully managed storage for GCP (Google Cloud Platform). • For deployments in GCP, Filestore is the recommended option for the Storage Manager directory, and Google Object Storage (S3-compatible) is the recommended option for data.
GlusterFS	On-premises	• GlusterFS is a distributed file system. • GlusterFS supports replication and failover.
NFS (Network File System)	On-premises	• NFS is a distributed file system. • If NFS is used, the storage should be mounted with the sync option to ensure that each node flushes its changes immediately. • For on-premises deployments, NFS is the recommended option for the Storage Manager directory, and any S3-compatible storage is the recommended option for data.

Recommended Storage Options

For best results, MariaDB Corporation would recommend the following storage options:

Environment	Object Storage For Data	Shared Local Storage For Storage Manager
AWS	Amazon S3 storage	EBS Multi-Attach or EFS
GCP	Google Object Storage (S3-compatible)	Filestore
On-premises	Any S3-compatible object storage	NFS

Enterprise ColumnStore Management with CMAPI

Enterprise ColumnStore's CMAPI (Cluster Management API) is a REST API that can be used to manage a multi-node Enterprise ColumnStore cluster.

Many tools are capable of interacting with REST APIs. For example, the curl utility could be used to make REST API calls from the command-line.

Many programming languages also have libraries for interacting with REST APIs.

The examples below show how to use the CMAPI with curl.

URL Endpoint Format for REST API

```
https://{server}:{port}/cmapi/{version}/{route}/{command}
```

For example:

- `https://mcs1:8640/cmapi/0.4.0/cluster/shutdown`
- `https://mcs1:8640/cmapi/0.4.0/cluster/start`
- `https://mcs1:8640/cmapi/0.4.0/cluster/status`

With CMAPI 1.4 and later:

- `https://mcs1:8640/cmapi/0.4.0/cluster/node`

With CMAPI 1.3 and earlier:

- `https://mcs1:8640/cmapi/0.4.0/cluster/add-node`
- `https://mcs1:8640/cmapi/0.4.0/cluster/remove-node`

Required Request Headers

- `'x-api-key': '93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd'`
- `'Content-Type': 'application/json'`

`x-api-key` can be set to any value of your choice during the first call to the server. Subsequent connections will require this same key.

Get Status

> **NOTE**
>
> `curl` examples remain valid but are now considered **legacy**.

`$ mcs cluster status`

```
$ curl -k -s https://mcs1:8640/cmapi/0.4.0/cluster/status \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      | jq .
```

Start Cluster

`$ mcs cluster start --timeout 20`

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/start \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20}' \
      | jq .
```

Stop Cluster

`$ mcs cluster shutdown --timeout 20`

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/shutdown \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20}' \
      | jq .
```

Add Node

- With CMAPI 1.4 and later:

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/node \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20, "node": "192.0.2.2"}' \
      | jq .
```

- With CMAPI 1.3 and earlier:

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/add-node \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20, "node": "192.0.2.2"}' \
      | jq .
```

Remove Node

- With CMAPI 1.4 and later:

```
$ curl -k -s -X DELETE https://mcs1:8640/cmapi/0.4.0/cluster/node \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20, "node": "192.0.2.2"}' \
      | jq .
```

- With CMAPI 1.3 and earlier:

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/remove-node \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20, "node": "192.0.2.2"}' \
      | jq .
```

Quick Reference

MariaDB Enterprise Server Configuration Management

Method	Description
Configuration File	Configuration files (such as /etc/my.cnf) can be used to set system-variables and options. The server must be restarted to apply changes made to configuration files.
Command-line	The server can be started with command-line options that set system-variables and options.
SQL	Users can set system-variables that support dynamic changes on-the-fly using the SET statement.

MariaDB Enterprise Server packages are configured to read configuration files from different paths, depending on the operating system. Making custom changes to Enterprise Server default configuration files is not recommended because custom changes may be overwritten by other default configuration files that are loaded later.

To ensure that your custom changes will be read last, create a custom configuration file with the z- prefix in one of the include directories.

Distribution	Example Configuration File Path
Distribution	Example Configuration File Path
• CentOS • Red Hat Enterprise Linux (RHEL)	/etc/my.cnf.d/z-custom-mariadb.cnf

Distribution	Example Configuration File Path
• Debian • Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

MariaDB Enterprise Server Service Management

The systemctl command is used to start and stop the MariaDB Enterprise Server service.

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

For additional information, see "Starting and Stopping MariaDB".

MariaDB Enterprise Server Logs

MariaDB Enterprise Server produces log data that can be helpful in problem diagnosis.

Log filenames and locations may be overridden in the server configuration. The default location of logs is the data directory. The data directory is specified by the datadir system variable.

Log	System Variable/Option	Default Filename
MariaDB Error Log	log_error	`<hostname>.err`
MariaDB Enterprise Audit Log	server_audit_file_path	`server_audit.log`
Slow Query Log	slow_query_log_file	`<hostname>-slow.log`
General Query Log	general_log_file	`<hostname>.log`
Binary Log	log_bin	`<hostname>-bin`

Enterprise ColumnStore Service Management

The systemctl command is used to start and stop the ColumnStore service.

Operation	Command
Start	`sudo systemctl start mariadb-columnstore`
Stop	`sudo systemctl stop mariadb-columnstore`
Restart	`sudo systemctl restart mariadb-columnstore`
Enable during startup	`sudo systemctl enable mariadb-columnstore`
Disable during startup	`sudo systemctl disable mariadb-columnstore`
Status	`sudo systemctl status mariadb-columnstore`

In the ColumnStore Object Storage topology, the mariadb-columnstore service should not be enabled. The CMAPI service restarts Enterprise ColumnStore as needed, so it does not need to start automatically upon reboot.

Enterprise ColumnStore CMAPI Service Management

The systemctl command is used to start and stop the CMAPI service.

Operation	Command
Start	`sudo systemctl start mariadb-columnstore-cmapi`
Stop	`sudo systemctl stop mariadb-columnstore-cmapi`
Restart	`sudo systemctl restart mariadb-columnstore-cmapi`
Enable during startup	`sudo systemctl enable mariadb-columnstore-cmapi`
Disable during startup	`sudo systemctl disable mariadb-columnstore-cmapi`
Status	`sudo systemctl status mariadb-columnstore-cmapi`

For additional information on endpoints, see “CMAPI”.

MaxScale Configuration Management

MaxScale can be configured using several methods. These methods make use of MaxScale’s REST API.

Method	Benefits
MaxCtrl	Command-line utility to perform administrative tasks through the REST API. See MaxCtrl Commands.
MaxGUI	MaxGUI is a graphical utility that can perform administrative tasks through the REST API.
REST API	The REST API can be used directly. For example, the curl utility could be used to make REST API calls from the command-line. Many programming languages also have libraries to interact with REST APIs.

The procedure on these pages configures MaxScale using MaxCtrl.

MaxScale Service Management

The `systemctl` command is used to start and stop the MaxScale service.

Operation	Command
Start	`sudo systemctl start maxscale`
Stop	`sudo systemctl stop maxscale`
Restart	`sudo systemctl restart maxscale`
Enable during startup	`sudo systemctl enable maxscale`
Disable during startup	`sudo systemctl disable maxscale`
Status	`sudo systemctl status maxscale`

For additional information, see “Starting and Stopping MariaDB”.

Next Step

Navigation in the procedure “Deploy ColumnStore Object Storage Topology”:

Next: Step 1: Prepare ColumnStore Nodes.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 1: Prepare ColumnStore Nodes

Overview

This page details step 1 of the 9-step procedure “Deploy ColumnStore Object Storage Topology”.

This step prepares systems to host MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Optimize Linux Kernel Parameters

MariaDB Enterprise ColumnStore performs best with Linux kernel optimizations.

On each server to host an Enterprise ColumnStore node, optimize the kernel:

1. Set the relevant kernel parameters in a sysctl configuration file. To ensure proper change management, use an Enterprise ColumnStore-specific configuration file.

Create `a /etc/sysctl.d/90-mariadb-enterprise-columnstore.conf file`:

```
# minimize swapping
vm.swappiness = 1

# Increase the TCP max buffer size
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216

# Increase the TCP buffer limits
# min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

# don't cache ssthresh from previous connection
net.ipv4.tcp_no_metrics_save = 1

# for 1 GigE, increase this to 2500
# for 10 GigE, increase this to 30000
net.core.netdev_max_backlog = 2500
```

2. Use the sysctl command to set the kernel parameters at runtime

```
$ sudo sysctl --load=/etc/sysctl.d/90-mariadb-enterprise-columnstore.conf
```

Temporarily Configure Linux Security Modules (LSM)

The Linux Security Modules (LSM) should be temporarily disabled on each Enterprise ColumnStore node during installation.

The LSM will be configured and re-enabled later in this deployment procedure.

The steps to disable the LSM depend on the specific LSM used by the operating system.

CentOS / RHEL Stop SELinux

SELinux must be set to permissive mode before installing MariaDB Enterprise ColumnStore.

To set SELinux to permissive mode:

1. Set SELinux to permissive mode:

```
$ sudo setenforce permissive
```

2. Set SELinux to permissive mode by setting SELINUX=permissive in /etc/selinux/config.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. Confirm that SELinux is in permissive mode:

```
$ sudo getenforce
Permissive
```

SELinux will be configured and re-enabled later in this deployment procedure. This configuration is not persistent. If you restart the server before configuring and re-enabling SELinux later in the deployment procedure, you must reset the enforcement to permissive mode.

Debian / Ubuntu AppArmor

AppArmor must be disabled before installing MariaDB Enterprise ColumnStore.

1. Disable AppArmor:

```
$ sudo systemctl disable apparmor
```

2. Reboot the system.
3. Confirm that no AppArmor profiles are loaded using aa-status:

```
$ sudo aa-status
```

```
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

AppArmor will be configured and re-enabled later in this deployment procedure.

Temporarily Configure Firewall for Installation

MariaDB Enterprise ColumnStore requires the following TCP ports:

TCP Ports	Description
3306	Port used for MariaDB Client traffic
8600-8630	Port range used for inter-node communication
8640	Port used by CMAPI
8700	Port used for inter-node communication
8800	Port used for inter-node communication

The firewall should be temporarily disabled on each Enterprise ColumnStore node during installation.

The firewall will be configured and re-enabled later in this deployment procedure.

The steps to disable the firewall depend on the specific firewall used by the operating system.

CentOS / RHEL Stop firewalld

1. Check if the firewalld service is running:

```
$ sudo systemctl status firewalld
```

2. If the firewalld service is running, stop it:

```
$ sudo systemctl stop firewalld
```

Firewalld will be configured and re-enabled later in this deployment procedure.

Ubuntu Stop UFW

1. Check if the UFW service is running:

```
$ sudo ufw status verbose
```

2. If the UFW service is running, stop it:

```
$ sudo ufw disable
```

UFW will be configured and re-enabled later in this deployment procedure.

Configure the AWS Security Group

To install Enterprise ColumnStore on Amazon Web Services (AWS), the security group must be modified prior to installation.

Enterprise ColumnStore requires all internal communications to be open between Enterprise ColumnStore nodes. Therefore, the security group should allow all protocols and all ports to be open between the Enterprise ColumnStore nodes and the MaxScale proxy.

Configure Character Encoding

When using MariaDB Enterprise ColumnStore, it is recommended to set the system's locale to UTF-8.

1. On RHEL 8, install additional dependencies:

```
$ sudo yum install glibc-locale-source glibc-langpack-en
```

2. Set the system's locale to en_US.UTF-8 by executing localedef:

```
$ sudo localedef -i en_US -f UTF-8 en_US.UTF-8
```

Configure DNS

MariaDB Enterprise ColumnStore requires all nodes to have host names that are resolvable on all other nodes. If your infrastructure does not configure DNS centrally, you may need to configure static DNS entries in the /etc/hosts file of each server.

On each Enterprise ColumnStore node, edit the /etc/hosts file to map host names to the IP address of each Enterprise ColumnStore node:

```
192.0.2.1     mcs1
192.0.2.2     mcs2
192.0.2.3     mcs3
192.0.2.100   mxs1
```

Replace the IP addresses with the addresses in your own environment.

Create an S3 Bucket

With the ColumnStore Object Storage topology, it is important to create the S3 bucket before you start ColumnStore. All Enterprise ColumnStore nodes access data from the same bucket.

If you already have an S3 bucket, confirm that the bucket is empty.

S3 bucket configuration will be performed later in this procedure.

Next Step

Navigation in the procedure “Deploy ColumnStore Object Storage Topology”:

This page was step 1 of 9.

Next: Step 2: Configure Shared Local Storage.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 2: Configure Shared Local Storage

Overview

This page details step 2 of the 9-step procedure “Deploy ColumnStore Object Storage Topology”.

This step configures shared local storage on systems hosting Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Directories for Shared Local Storage

In a ColumnStore Object Storage topology, MariaDB Enterprise ColumnStore requires the Storage Manager directory to be located on shared local storage.

The Storage Manager directory is at the following path:

- `/var/lib/columnstore/storagemanager`

The Storage Manager directory must be mounted on every ColumnStore node.

Choose a Shared Local Storage Solution

Select a Shared Local Storage solution for the Storage Manager directory:

- EBS (Elastic Block Store) Multi-Attach
- EFS (Elastic File System)
- Filestore
- GlusterFS
- NFS (Network File System)

For additional information, see “Shared Local Storage Options”.

Configure EBS Multi-Attach

EBS is a high-performance block-storage service for AWS (Amazon Web Services). EBS Multi-Attach allows an EBS volume to be attached to multiple instances in AWS. Only clustered file systems, such as GFS2, are supported.

For Enterprise ColumnStore deployments in AWS:

- EBS Multi-Attach is a recommended option for the Storage Manager directory.
- Amazon S3 storage is the recommended option for data.
- Consult the vendor documentation for details on how to configure EBS Multi-Attach.

Configure Elastic File System (EFS)

EFS is a scalable, elastic, cloud-native NFS file system for AWS (Amazon Web Services)

For deployments in AWS:

- EFS is a recommended option for the Storage Manager directory.
- Amazon S3 storage is the recommended option for data.
- Consult the vendor documentation for details on how to configure EFS.

Configure Filestore

Filestore is high-performance, fully managed storage for GCP (Google Cloud Platform).

For Enterprise ColumnStore deployments in GCP:

- Filestore is the recommended option for the Storage Manager directory.
- Google Object Storage (S3-compatible) is the recommended option for data.
- Consult the vendor documentation for details on how to configure Filestore.

Configure GlusterFS

GlusterFS is a distributed file system.

GlusterFS is a shared local storage option, but it is not one of the recommended options.

For more information, see “Recommended Storage Options”.

Install GlusterFS

On each Enterprise ColumnStore node, install GlusterFS.

Install on CentOS / RHEL 8 (YUM):

```
$ sudo yum install --enablerepo=PowerTools glusterfs-server
```

Install on CentOS / RHEL 7 (YUM):

```
$ sudo yum install centos-release-gluster
$ sudo yum install glusterfs-server
```

Install on Debian (APT):

```
$ wget -O - https://download.gluster.org/pub/gluster/glusterfs/LATEST/rsa.pub | apt-key add -

$ DEBID=$(grep 'VERSION_ID=' /etc/os-release | cut -d '=' -f 2 | tr -d '"')
$ DEBVER=$(grep 'VERSION=' /etc/os-release | grep -Eo '[a-z]+')
$ DEBARCH=$(dpkg --print-architecture)
$ echo deb https://download.gluster.org/pub/gluster/glusterfs/LATEST/Debian/${DEBID}/${DEBARCH}/apt
${DEBVER} main > /etc/apt/sources.list.d/gluster.list
```

```
$ sudo apt update
```

```
$ sudo apt install glusterfs-server
```

Install on Ubuntu (APT):

```
$ sudo apt update
```

```
$ sudo apt install glusterfs-server
```

Start the GlusterFS Daemon

Start the GlusterFS daemon:

```
$ sudo systemctl start glusterd
```

```
$ sudo systemctl enable glusterd
```

Probe the GlusterFS Peers

Before you can create a volume with GlusterFS, you must probe each node from a peer node.

1. On the primary node, probe all of the other cluster nodes:

```
$ sudo gluster peer probe mcs2
```

```
$ sudo gluster peer probe mcs3
```

2. On one of the replica nodes, probe the primary node to confirm that it is connected:

```
$ sudo gluster peer probe mcs1
```

```
peer probe: Host mcs1 port 24007 already in peer list
```

3. On the primary node, check the peer status:

```
$ sudo gluster peer status
```

```
Number of Peers: 2

Hostname: mcs2
Uuid: 3c8a5c79-22de-45df-9034-8ae624b7b23e
State: Peer in Cluster (Connected)

Hostname: mcs3
Uuid: 862af7b2-bb5e-4b1c-8311-630fa32ed451
State: Peer in Cluster (Connected)
```

Configure and Mount GlusterFS Volumes

Create the GlusterFS volumes for MariaDB Enterprise ColumnStore. Each volume must have the same number of replicas as the number of Enterprise ColumnStore nodes.

1. On each Enterprise ColumnStore node, create the directory for each brick in the /brick directory:

```
$ sudo mkdir -p /brick/storagemanager
```

2. On the primary node, create the GlusterFS volumes:

```
$ sudo gluster volume create storagemanager \
      replica 3 \
      mcs1:/brick/storagemanager \
      mcs2:/brick/storagemanager \
      mcs3:/brick/storagemanager \
      force
```

3. On the primary node, start the volume:

```
$ sudo gluster volume start storagemanager
```

4. On each Enterprise ColumnStore node, create mount points for the volumes:

```
$ sudo mkdir -p /var/lib/columnstore/storagemanager
```

5. On each Enterprise ColumnStore node, add the mount points to /etc/fstab:

```
127.0.0.1:storagemanager /var/lib/columnstore/storagemanager glusterfs defaults,_netdev 0 0
```

6. On each Enterprise ColumnStore node, mount the volumes:

```
$ sudo mount -a
```

Configure Network File System (NFS)

NFS is a distributed file system. NFS is available in most Linux distributions. If NFS is used for an Enterprise ColumnStore deployment, the storage must be mounted with the sync option to ensure that each node flushes its changes immediately.

For on-premises deployments:

- NFS is the recommended option for the Storage Manager directory.
- Any S3-compatible storage is the recommended option for data.

Consult the documentation for your NFS implementation for details on how to configure NFS.

Next Step

Navigation in the procedure “Deploy ColumnStore Object Storage Topology”:

This page was **step 2 of 9**.

Next: Step 3: Install MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 3: Install MariaDB Enterprise Server

Overview

This page details step 3 of the 9-step procedure “Deploy ColumnStore Object Storage Topology”.

This step installs MariaDB Enterprise Server, MariaDB Enterprise ColumnStore 23.10, CMAPI, and dependencies.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Download Token

MariaDB Corporation provides package repositories for CentOS / RHEL (YUM) and Debian / Ubuntu (APT). A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to https://customers.mariadb.com/downloads/token/
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On each Enterprise ColumnStore node, install the prerequisites for downloading the software from the Web. Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

2. On each Enterprise ColumnStore node, configure package repositories and specify Enterprise Server:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-maxscale \
      --skip-tools \
      --mariadb-server-version="11.4"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install Enterprise Server and Enterprise ColumnStore

1. On each Enterprise ColumnStore node, install additional dependencies:

Install on CentOS and RHEL (YUM):

```
$ sudo yum install jemalloc jq curl
```

Install on Debian 9 and Ubuntu 18.04 (APT)

```
$ sudo apt install libjemalloc1 jq curl
```

Install on Debian 10 and Ubuntu 20.04 (APT):

```
$ sudo apt install libjemalloc2 jq curl
```

2. On each Enterprise ColumnStore node, install MariaDB Enterprise Server and MariaDB Enterprise ColumnStore:

Install on CentOS / RHEL (YUM):

```
$ sudo yum install MariaDB-server \
   MariaDB-backup \
   MariaDB-shared \
   MariaDB-client \
   MariaDB-columnstore-engine \
   MariaDB-columnstore-cmapi
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install mariadb-server \
   mariadb-backup \
   libmariadb3 \
   mariadb-client \
   mariadb-plugin-columnstore \
   mariadb-columnstore-cmapi
```

Next Step

Navigation in the procedure “Deploy ColumnStore Object Storage Topology”.

This page was **step 3 of 9**.

Next: Step 4: Start and Configure MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 4: Start and Configure MariaDB Enterprise Server

Overview

This page details step 4 of the 9-step procedure “Deploy ColumnStore Object Storage Topology”.

This step starts and configures MariaDB Enterprise Server, and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Stop the Enterprise ColumnStore Services

The installation process might have started some of the ColumnStore services. The services should be stopped prior to making configuration changes.

1. On each Enterprise ColumnStore node, stop the MariaDB Enterprise Server service:

```
$ sudo systemctl stop mariadb
```

2. On each Enterprise ColumnStore node, stop the MariaDB Enterprise ColumnStore service:

```
$ sudo systemctl stop mariadb-columnstore
```

3. On each Enterprise ColumnStore node, stop the CMAPI service:

```
$ sudo systemctl stop mariadb-columnstore-cmapi
```

Configure Enterprise ColumnStore

On each Enterprise ColumnStore node, configure Enterprise Server.

Connector	MariaDB Connector/R2DBC
character_set_server	Set this system variable to `utf8`
collation_server	Set this system variable to `utf8_general_ci`
loose-columnstore_use_import_for_batchinsert	Set this system variable to ALWAYS to always use cpimport for LOAD DATA INFILE and INSERT…SELECT statements.
gtid_strict_mode	Set this system variable to ON.
log_bin	Set this option to the file you want to use for the Binary Log. Setting this option enables binary logging.
log_bin_index	Set this option to the file you want to use to track binlog filenames.
log_slave_updates	Set this system variable to ON.
relay_log	Set this option to the file you want to use for the Relay Logs. Setting this option enables relay logging.
relay_log_index	Set this option to the file you want to use to index Relay Log filenames.
server_id	Sets the numeric Server ID for this MariaDB Enterprise Server. The value set on this option must be unique to each node.

> **Warning:**
>
> The `loose-` prefix is required for ColumnStore system variables in the configuration file. Without it, MariaDB Server will fail to start if the ColumnStore plugin is not installed or has been removed.

Mandatory system variables and options for ColumnStore Object Storage include:

Example Configuration

```
[mariadb]
bind_address                            = 0.0.0.0
log_error                               = mariadbd.err
character_set_server                    = utf8
collation_server                        = utf8_general_ci
log_bin                                 = mariadb-bin
log_bin_index                           = mariadb-bin.index
relay_log                               = mariadb-relay
relay_log_index                         = mariadb-relay.index
log_slave_updates                       = ON
gtid_strict_mode                        = ON

# This must be unique on each Enterprise ColumnStore node
server_id                               = 1
```

Configure the S3 Storage Manager

On each Enterprise ColumnStore node, configure S3 Storage Manager to use S3-compatible storage by editing the /etc/columnstore/storagemanager.cnf configuration file:

```
[ObjectStorage]
…
service = S3
…
[S3]
bucket                = your_columnstore_bucket_name
endpoint              = your_s3_endpoint
aws_access_key_id     = your_s3_access_key_id
aws_secret_access_key = your_s3_secret_key
# iam_role_name       = your_iam_role
# sts_region          = your_sts_region
# sts_endpoint        = your_sts_endpoint
# ec2_iam_mode        = enabled

[Cache]
cache_size = your_local_cache_size
path       = your_local_cache_path
```

The S3-compatible object storage options are configured under [S3]:

- The bucket option must be set to the name of the bucket that you created in “Create an S3 Bucket”.
- The endpoint option must be set to the endpoint for the S3-compatible object storage.
- The `aws_access_key_id and aws_secret_access_key` options must be set to the access key ID and secret access key for the S3-compatible object storage.
- To use a specific IAM role, you must uncomment and set `iam_role_name, sts_region, and sts_endpoint`.
- To use the IAM role assigned to an EC2 instance, you must uncomment `ec2_iam_mode=enabled`.

The local cache options are configured under [Cache]:

- The cache_size option is set to 2 GB by default.
- The path option is set to `/var/lib/columnstore/storagemanager/cache` by default.

Ensure that the specified path has sufficient storage space for the specified cache size.

Start the Enterprise ColumnStore Services

1. On each Enterprise ColumnStore node, start and enable the MariaDB Enterprise Server service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb
```

```
$ sudo systemctl enable mariadb
```

2. On each Enterprise ColumnStore node, stop the MariaDB Enterprise ColumnStore service:

```
$ sudo systemctl stop mariadb-columnstore
```

After the CMAPI service is installed in the next step, CMAPI will start the Enterprise ColumnStore service as-needed on each node. CMAPI disables the Enterprise ColumnStore service to prevent systemd from automatically starting Enterprise ColumnStore upon reboot.

3. On each Enterprise ColumnStore node, start and enable the CMAPI service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb-columnstore-cmapi
```

```
$ sudo systemctl enable mariadb-columnstore-cmapi
```

For additional information, see “Starting and Stopping MariaDB”.

Create User Accounts

The ColumnStore Object Storage topology requires several user accounts. Each user account should be created on the primary server, so that it is replicated to the replica servers.

Create the Utility User

Enterprise ColumnStore requires a mandatory utility user account to perform cross-engine joins and similar operations.

1. On the primary server, create the user account with the CREATE USER statement:

```
CREATE USER 'util_user'@'127.0.0.1'
IDENTIFIED BY 'util_user_passwd';
```

2. On the primary server, grant the user account SELECT privileges on all databases with the GRANT statement:

```
GRANT SELECT, PROCESS ON *.*
TO 'util_user'@'127.0.0.1';
```

3. On each Enterprise ColumnStore node, configure the ColumnStore utility user:

```
$ sudo mcsSetConfig CrossEngineSupport Host 127.0.0.1
```

```
$ sudo mcsSetConfig CrossEngineSupport Port 3306
```

```
$ sudo mcsSetConfig CrossEngineSupport User util_user
```

4. On each Enterprise ColumnStore node, set the password:

```
$ sudo mcsSetConfig CrossEngineSupport Password util_user_passwd
```

For details about how to encrypt the password, see “Credentials Management for MariaDB Enterprise ColumnStore”.

Passwords should meet your organization’s password policies. If your MariaDB Enterprise Server instance has a password validation plugin installed, then the password should also meet the configured requirements.

Create the Replication User

ColumnStore Object Storage uses MariaDB Replication to replicate writes between the primary and replica servers. As MaxScale can promote a replica server to become a new primary in the event of node failure, all nodes must have a replication user.

The action is performed **on the primary server**.

Create the replication user and grant it the required privileges:

1. Use the CREATE USER statement to create replication user.

```
CREATE USER 'repl'@'192.0.2.%' IDENTIFIED BY 'repl_passwd';
```

Replace the referenced IP address with the relevant address for your environment.

Ensure that the user account can connect to the primary server from each replica.

2. Grant the user account the required privileges with the GRANT statement.

```
GRANT REPLICA MONITOR,
   REPLICATION REPLICA,
   REPLICATION REPLICA ADMIN,
   REPLICATION MASTER ADMIN
ON *.* TO 'repl'@'192.0.2.%';
```

Create MaxScale User

ColumnStore Object Storage 23.10 uses MariaDB MaxScale 22.08 to load balance between the nodes.

This action is performed on the primary server.

1. Use the CREATE USER statement to create the MaxScale user:

```
CREATE USER 'mxs'@'192.0.2.%'
IDENTIFIED BY 'mxs_passwd';
```

Replace the referenced IP address with the relevant address for your environment.

Ensure that the user account can connect from the IP address of the MaxScale instance.

2. Use the GRANT statement to grant the privileges required by the router:

```
GRANT SHOW DATABASES ON *.* TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.columns_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.db TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.procs_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.proxies_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.roles_mapping TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.tables_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.user TO 'mxs'@'192.0.2.%';
```

3. Use the GRANT statement to grant privileges required by the MariaDB Monitor.

```
GRANT BINLOG ADMIN,
   READ_ONLY ADMIN,
   RELOAD,
   REPLICA MONITOR,
   REPLICATION MASTER ADMIN,
   REPLICATION REPLICA ADMIN,
   REPLICATION REPLICA,
   SHOW DATABASES,
   SELECT
ON *.* TO 'mxs'@'192.0.2.%';
```

Configure MariaDB Replication

On each replica server, configure MariaDB Replication:

1. Use the CHANGE MASTER TO statement to configure the connection to the primary server:

```
CHANGE MASTER TO
   MASTER_HOST='192.0.2.1',
   MASTER_USER='repl',
   MASTER_PASSWORD='repl_passwd',
   MASTER_USE_GTID=slave_pos;
```

2. Start replication using the START REPLICA statement:

```
START REPLICA;
```

3. Confirm that replication is working using the SHOW REPLICA STATUS statement:

```
SHOW REPLICA STATUS;
```

Ensure that the replica server cannot accept local writes by setting the read_only system variable to ON using the SET GLOBAL statement:

```
SET GLOBAL read_only=ON;
```

Initiate the Primary Server with CMAPI

Initiate the primary server using CMAPI.

1. Create an API key for the cluster.
   This API key should be stored securely and kept confidential, because it can be used to add cluster nodes to the multi-node Enterprise ColumnStore deployment.

For example, to create a random 256-bit API key using openssl rand:

```
$ openssl rand -hex 32

93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd
```

This document will use the following API key in further examples, but users should create their own:

2. Use CMAPI to add the primary server to the cluster and set the API key.
   The new API key needs to be provided as part of the X-API-key HTML header.

For example, if the primary server's host name is mcs1 and its IP address is 192.0.2.1, use the following node command:

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/node \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   --data '{"timeout":120, "node": "192.0.2.1"}' \
   | jq .
```

```
{
  "timestamp": "2020-10-28 00:39:14.672142",
  "node_id": "192.0.2.1"
}
```

3. Use CMAPI to check the status of the cluster node:

```
$ curl -k -s https://mcs1:8640/cmapi/0.4.0/cluster/status \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   | jq .
```

```
{
  "timestamp": "2020-12-15 00:40:34.353574",
  "192.0.2.1": {
    "timestamp": "2020-12-15 00:40:34.362374",
    "uptime": 11467,
    "dbrm_mode": "master",
    "cluster_mode": "readwrite",
    "dbroots": [
      "1"
    ],
    "module_id": 1,
    "services": [
      {
        "name": "workernode",
        "pid": 19202
      },
      {
        "name": "controllernode",
        "pid": 19232
      },
      {
        "name": "PrimProc",
        "pid": 19254
      },
      {
        "name": "ExeMgr",
        "pid": 19292
      },
      {
        "name": "WriteEngine",
        "pid": 19316
      },
      {
        "name": "DMLProc",
        "pid": 19332
      },
      {
        "name": "DDLProc",
        "pid": 19366
      }
    ]
  }
```


Add Replica Servers with CMAPI

Add the replica servers with CMAPI:

1. For each replica server, use CMAPI to add the replica server to the cluster.
   The previously set API key needs to be provided as part of the X-API-key HTML header.

For example, if the primary server's host name is mcs1 and the replica server's IP address is 192.0.2.2, use the following node command:

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/node \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   --data '{"timeout":120, "node": "192.0.2.2"}' \
   | jq .
```


```
{
  "timestamp": "2020-10-28 00:42:42.796050",
  "node_id": "192.0.2.2"
}
```

2. After all replica servers have been added, use CMAPI to confirm that all cluster nodes have been successfully added:

```
$ curl -k -s https://mcs1:8640/cmapi/0.4.0/cluster/status \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   | jq .
```

```
{
  "timestamp": "2020-12-15 00:40:34.353574",
  "192.0.2.1": {
    "timestamp": "2020-12-15 00:40:34.362374",
    "uptime": 11467,
    "dbrm_mode": "master",
    "cluster_mode": "readwrite",
    "dbroots": [
      "1"
    ],
    "module_id": 1,
    "services": [
      {
        "name": "workernode",
        "pid": 19202
      },
      {
        "name": "controllernode",
        "pid": 19232
      },
      {
        "name": "PrimProc",
        "pid": 19254
      },
      {
        "name": "ExeMgr",
        "pid": 19292
      },
      {
        "name": "WriteEngine",
        "pid": 19316
      },
      {
        "name": "DMLProc",
        "pid": 19332
      },
      {
        "name": "DDLProc",
        "pid": 19366
      }
    ]
  },
  "192.0.2.2": {
    "timestamp": "2020-12-15 00:40:34.428554",
    "uptime": 11437,
    "dbrm_mode": "slave",
    "cluster_mode": "readonly",
    "dbroots": [
      "2"
    ],
    "module_id": 2,
    "services": [
      {
        "name": "workernode",
        "pid": 17789
      },
      {
        "name": "PrimProc",
        "pid": 17813
      },
      {
        "name": "ExeMgr",
        "pid": 17854
      },
      {
        "name": "WriteEngine",
        "pid": 17877
      }
```

```
    ]
  },
  "192.0.2.3": {
    "timestamp": "2020-12-15 00:40:34.428554",
    "uptime": 11437,
    "dbrm_mode": "slave",
    "cluster_mode": "readonly",
    "dbroots": [
      "2"
    ],
    "module_id": 2,
    "services": [
      {
        "name": "workernode",
        "pid": 17789
      },
      {
        "name": "PrimProc",
        "pid": 17813
      },
      {
        "name": "ExeMgr",
        "pid": 17854
      },
      {
        "name": "WriteEngine",
        "pid": 17877
      }
    ]
  },
  "num_nodes": 3
}
```

Configure Linux Security Modules (LSM)

The specific steps to configure the security module depend on the operating system.

Configure SELinux (CentOS, RHEL)

Configure SELinux for Enterprise ColumnStore:

1. To configure SELinux, you have to install the packages required for audit2allow.
   On CentOS 7 and RHEL 7, install the following:

```
$ sudo yum install policycoreutils policycoreutils-python
```

On RHEL 8, install the following:

```
$ sudo yum install policycoreutils python3-policycoreutils policycoreutils-python-utils
```

2. Allow the system to run under load for a while to generate SELinux audit events.
3. After the system has taken some load, generate an SELinux policy from the audit events using audit2allow:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local
```

If no audit events were found, this will print the following:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local

Nothing to do
```

4. If audit events were found, the new SELinux policy can be loaded using semodule:

```
$ sudo semodule -i mariadb_local.pp
```

5. Set SELinux to enforcing mode:

```
$ sudo setenforce enforcing
```

6. Set SELinux to enforcing mode by setting SELINUX=enforcing in /etc/selinux/config.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

7. Confirm that SELinux is in enforcing mode:

```
$ sudo getenforce
```

```
Enforcing
```

Configure AppArmor (Ubuntu)

For information on how to create a profile, see How to create an AppArmor Profile on Ubuntu.com.

Configure Firewalls

The specific steps to configure the firewall service depend on the platform.

Configure firewalld (CentOS, RHEL)

Configure firewalld for Enterprise Cluster on CentOS and RHEL:

1. Check if the firewalld service is running:

```
$ sudo systemctl status firewalld
```

2. If the firewalld service was stopped to perform the installation, start it now:

For example, if your cluster nodes are in the 192.0.2.0/24 subnet:

```
$ sudo systemctl start firewalld
```

3. Open up the relevant ports using firewall-cmd:

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="3306" protocol="tcp"
   accept'
```

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="8600-8630" protocol="tcp"
   accept'
```

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="8640" protocol="tcp"
   accept'
```

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="8700" protocol="tcp"
   accept'
```

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="8800" protocol="tcp"
   accept'
```

4. Reload the runtime configuration:

```
$ sudo firewall-cmd --reload
```

Configure UFW (Ubuntu)

Configure UFW for Enterprise ColumnStore on Ubuntu:

1. Check if the UFW service is running:

```
$ sudo ufw status verbose
```

2. If the UFW service was stopped to perform the installation, start it now:

```
$ sudo ufw enable
```

3. Open up the relevant ports using ufw.

For example, if your cluster nodes are in the 192.0.2.0/24 subnet in the range 192.0.2.1 - 192.0.2.3:

```
$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 3306 proto tcp

$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 8600:8630 proto tcp

$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 8640 proto tcp

$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 8700 proto tcp

$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 8800 proto tcp
```

4. Reload the runtime configuration:

```
$ sudo ufw reload
```

Next Step

Navigation in the procedure “Deploy ColumnStore Object Storage Topology”:

This page was **step 4 of 9**.

Next: Step 5: Test MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 5: Test MariaDB Enterprise Server

Overview

This page details step 5 of the 9-step procedure “Deploy ColumnStore Object Storage Topology”.

This step tests MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Test S3 Connection

MariaDB Enterprise ColumnStore 23.10 includes a `testS3Connection` command to test the S3 configuration, permissions, and connectivity.

This action is performed **on each Enterprise ColumnStore node**.

Test the S3 configuration by executing the following:

```
$ sudo testS3Connection
```

```
StorageManager[26887]: Using the config file found at /etc/columnstore/storagemanager.cnf
StorageManager[26887]: S3Storage: S3 connectivity & permissions are OK
S3 Storage Manager Configuration OK
```

If the `testS3Connection` command does not return `OK`, investigate the S3 configuration.

Test Enterprise Server Service

Use Systemd to test whether the MariaDB Enterprise Server service is running.

This action is performed **on each Enterprise ColumnStore node**.

Check if the MariaDB Enterprise Server service is running by executing the following:

```
$ systemctl status mariadb
```

If the service is not running on any node, start the service by executing the following on that node:

```
$ sudo systemctl start mariadb
```

Test Local Client Connections

Use MariaDB Client to test the local connection to the Enterprise Server node.

This action is performed on each Enterprise ColumnStore node:

```
$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The `sudo` command is used here to connect to the Enterprise Server node using the root@localhost user account, which authenticates using the unix_socket authentication plugin. Other user accounts can be used by specifying the –user and –password command-line options.

Test ColumnStore Storage Engine Plugin

Query the information_schema.PLUGINS table to confirm that the ColumnStore storage engine is loaded.

This action is performed on each Enterprise ColumnStore node.

Execute the following query:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS
FROM information_schema.PLUGINS
WHERE PLUGIN_LIBRARY LIKE 'ha_columnstore%';

+---------------------+---------------+
| PLUGIN_NAME         | PLUGIN_STATUS |
+---------------------+---------------+
| Columnstore         | ACTIVE        |
| COLUMNSTORE_COLUMNS | ACTIVE        |
| COLUMNSTORE_TABLES  | ACTIVE        |
| COLUMNSTORE_FILES   | ACTIVE        |
| COLUMNSTORE_EXTENTS | ACTIVE        |
+---------------------+---------------+
```

The `PLUGIN_STATUS` column for each ColumnStore-related plugin should contain ACTIVE.

Test CMAPI Service

Use Systemd to test whether the CMAPI service is running.

This action is performed on each Enterprise ColumnStore node.

Check if the CMAPI service is running by executing the following:

```
$ systemctl status mariadb-columnstore-cmapi
```

If the service is not running on any node, start the service by executing the following on that node:

```
$ sudo systemctl start mariadb-columnstore-cmapi
```

Test ColumnStore Status

Use CMAPI to request the ColumnStore status. The API key needs to be provided as part of the `X-API-key` HTML header.

This action is performed with the CMAPI service on the primary server.

Check the ColumnStore status using curl by executing the following:

```
$ curl -k -s https://mcs1:8640/cmapi/0.4.0/cluster/status \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   | jq .

{
  "timestamp": "2020-12-15 00:40:34.353574",
  "192.0.2.1": {
    "timestamp": "2020-12-15 00:40:34.362374",
    "uptime": 11467,
    "dbrm_mode": "master",
    "cluster_mode": "readwrite",
    "dbroots": [
      "1"
    ],
    "module_id": 1,
    "services": [
      {
        "name": "workernode",
        "pid": 19202
      },
      {
        "name": "controllernode",
        "pid": 19232
      },
      {
        "name": "PrimProc",
        "pid": 19254
      },
      {
        "name": "ExeMgr",
        "pid": 19292
      },
      {
        "name": "WriteEngine",
        "pid": 19316
      },
      {
        "name": "DMLProc",
        "pid": 19332
      },
      {
        "name": "DDLProc",
        "pid": 19366
      }
    ]
  },
  "192.0.2.2": {
    "timestamp": "2020-12-15 00:40:34.428554",
    "uptime": 11437,
    "dbrm_mode": "slave",
    "cluster_mode": "readonly",
    "dbroots": [
      "2"
    ],
    "module_id": 2,
    "services": [
      {
        "name": "workernode",
        "pid": 17789
      },
      {
        "name": "PrimProc",
        "pid": 17813
      },
      {
        "name": "ExeMgr",
        "pid": 17854
```

```
      },
      {
        "name": "WriteEngine",
        "pid": 17877
      }
    ]
  },
  "192.0.2.3": {
    "timestamp": "2020-12-15 00:40:34.428554",
    "uptime": 11437,
    "dbrm_mode": "slave",
    "cluster_mode": "readonly",
    "dbroots": [
      "2"
    ],
    "module_id": 2,
    "services": [
      {
        "name": "workernode",
        "pid": 17789
      },
      {
        "name": "PrimProc",
        "pid": 17813
      },
      {
        "name": "ExeMgr",
        "pid": 17854
      },
      {
        "name": "WriteEngine",
        "pid": 17877
      }
    ]
  },
  "num_nodes": 3
}
```


Test DDL

Use MariaDB Client to test DDL.

1. On the primary server, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

2. Create a test database and ColumnStore table:

```
CREATE DATABASE IF NOT EXISTS test;

CREATE TABLE IF NOT EXISTS test.contacts (
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   email VARCHAR(100)
) ENGINE = ColumnStore;
```

3. On each replica server, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

4. Confirm that the database and table exist:

```
SHOW CREATE TABLE test.contacts\G;
```

If the database or table do not exist on any node, then check the replication configuration.

Test DML

Use MariaDB Client to test DML.

1. On the primary server, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

2. Insert sample data into the table created in the DDL test:

```
INSERT INTO test.contacts (first_name, last_name, email)
   VALUES
   ("Kai", "Devi", "kai.devi@example.com"),
   ("Lee", "Wang", "lee.wang@example.com");
```

3. On each replica server, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

4. Execute a SELECT query to retrieve the data:

```
SELECT * FROM test.contacts;

+------------+-----------+----------------------+
| first_name | last_name | email                |
+------------+-----------+----------------------+
| Kai        | Devi      | kai.devi@example.com |
| Lee        | Wang      | lee.wang@example.com |
+------------+-----------+----------------------+
```

If the data is not returned on any node, check the ColumnStore status and the storage configuration.

Next Step

Navigation in the procedure 'Deploy ColumnStore Object Storage Topology".

This page was **step 5 of 9**.

Next: Step 6: Install MariaDB MaxScale.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 6: Install MariaDB MaxScale

Overview

This page details step 6 of the 9-step procedure "Deploy ColumnStore Object Storage Topology".

This step installs MariaDB MaxScale 22.08.

ColumnStore Object Storage requires 1 or more MaxScale nodes.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Customer Download Token

MariaDB Corporation provides package repositories for `CentOS / RHEL (YUM) and Debian / Ubuntu (APT)`. A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to https://customers.mariadb.com/downloads/token/
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On the MaxScale node, install the prerequisites for downloading the software from the Web.
   Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

2. On the MaxScale node, configure package repositories and specify MariaDB MaxScale 22.08:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup

$ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -

$ chmod +x mariadb_es_repo_setup

$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-server \
      --skip-tools \
      --mariadb-maxscale-version="22.08"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install MaxScale

On the MaxScale node, install MariaDB MaxScale.

Install on CentOS / RHEL (YUM):

```
$ sudo yum install maxscale
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install maxscale
```

Next Step

Navigation in the procedure “Deploy ColumnStore Object Storage Topology”:

This page was **step 6 of 9**.

Next: Step 7: Start and Configure MariaDB MaxScale.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 7: Start and Configure MariaDB MaxScale

Overview

This page details step 7 of the 9-step procedure "Deploy ColumnStore Object Storage Topology".

This step starts and configures MariaDB MaxScale 22.08.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Replace the Default Configuration File

MariaDB MaxScale installations include a configuration file with some example objects. This configuration file should be replaced.

On the MaxScale node, replace the default `/etc/maxscale.cnf` with the following configuration:

```
[maxscale]
threads          = auto
admin_host       = 0.0.0.0
admin_secure_gui = false
```

For additional information, see "Global Parameters".

Restart MaxScale

On the MaxScale node, restart the MaxScale service to ensure that MaxScale picks up the new configuration:

```
$ sudo systemctl restart maxscale
```

For additional information, see "Start and Stop Services".

Configure Server Objects

On the MaxScale node, use maxctrl create to create a server object for each Enterprise ColumnStore node:

```
$ maxctrl create server mcs1 192.0.2.101
```

```
$ maxctrl create server mcs2 192.0.2.102
```

```
$ maxctrl create server mcs3 192.0.2.103
```

Configure MariaDB Monitor

MaxScale uses monitors to retrieve additional information from the servers. This information is used by other services in filtering and routing connections based on the current state of the node. For MariaDB Enterprise ColumnStore, use the MariaDB Monitor (mariadbmon).

On the MaxScale node, use maxctrl create monitor to create a MariaDB Monitor:

```
$ maxctrl create monitor columnstore_monitor mariadbmon \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     replication_user=repl \
     replication_password='REPLICATION_USER_PASSWORD' \
     --servers mcs1 mcs2 mcs3
```

In this example:

- `columnstore_monitor` is an arbitrary name that is used to identify the new monitor.

- `mariadbmon` is the name of the module that implements the MariaDB Monitor.
- `user=MAXSCALE_USER` sets the user parameter to the database user account that MaxScale uses to monitor the ColumnStore nodes.
- `password='MAXSCALE_USER_PASSWORD'` sets the password parameter to the password used by the database user account that MaxScale uses to monitor the ColumnStore nodes.
- `replication_user=REPLICATION_USER` sets the replication_user parameter to the database user account that MaxScale uses to setup replication.
- `replication_password='REPLICATION_USER_PASSWORD'` sets the replication_password parameter to the password used by the database user account that MaxScale uses to setup replication.
- `--servers` sets the servers parameter to the set of nodes that MaxScale should monitor. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by mariadbmon in MaxScale 22.08 can also be specified.

Choose a MaxScale Router

Routers control how MaxScale balances the load between Enterprise ColumnStore nodes. Each router uses a different approach to routing queries. Consider the specific use case of your application and database load and select the router that best suits your needs.

Router	Configuration Procedure	Description
Read Connection (readconnroute)	Configure Read Connection Router	**Connection-based load balancing** • Routes connections to Enterprise ColumnStore nodes designated as replica servers for a read-only pool • Routes connections to an Enterprise ColumnStore node designated as the primary server for a read-write pool.|
Read/Write Split (readwritesplit)	Configure Read/Write Split	**Query-based load balancing** • Routes write queries to an Enterprise ColumnStore node designated as the primary server • Routes read queries to Enterprise ColumnStore node designated as replica servers • Automatically reconnects after node failures • Automatically replays transactions after node failures • Optionally enforces causal reads|

Configure Read Connection Router

Use MaxScale Read Connection Router (readconnroute) to route connections to replica servers for a read-only pool.

On the MaxScale node, use maxctrl create service to create a router:

```
$ maxctrl create service connection_router_service readconnroute \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     router_options=slave \
     --servers mcs1 mcs2 mcs3
```

In this example:

- `connection_router_service` is an arbitrary name that is used to identify the new service.
- readconnroute is the name of the module that implements the Read Connection Router.

- `user=MAXSCALE_USER` sets the user parameter to the database user account that MaxScale uses to connect to the ColumnStore nodes.
- `password=MAXSCALE_USER_PASSWORD` sets the password parameter to the password used by the database user account that MaxScale uses to connect to the ColumnStore nodes.
- `router_options=slave` sets the `router_options` parameter to slave, so that MaxScale only routes connections to the replica nodes.
- `--servers` sets the servers parameter to the set of nodes to which MaxScale should route connections. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by `readconnroute` in MaxScale 22.08 can also be specified.

Configure Listener for the Read Connection Router

These instructions reference TCP port 3308. You can use a different TCP port. The TCP port used must not be bound by any other listener.

On the MaxScale node, use the maxctrl create listener command to configure MaxScale to use a listener for the Read Connection Router (readconnroute):

```
$ maxctrl create listener connection_router_service connection_router_listener 3308 \
     protocol=MariaDBClient
```

In this example:

- `connection_router_service` is the name of the `readconnroute` service that was previously created.
- `connection_router_listener` is an arbitrary name that is used to identify the new listener.
- 3308 is the TCP port.
- `protocol=MariaDBClient` sets the protocol parameter.
- Other Module Parameters supported by listeners in MaxScale 22.08 can also be specified.

Configure Read/Write Split Router for Queries

MaxScale Read/Write Split Router (readwritesplit) performs query-based load balancing. The router routes write queries to the primary and read queries to the replicas.

On the MaxScale node, use the maxctrl create service command to configure MaxScale to use the Read/Write Split Router (readwritesplit):

```
$ maxctrl create service query_router_service readwritesplit  \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     --servers mcs1 mcs2 mcs3
```

In this example:

- `query_router_service` is an arbitrary name that is used to identify the new service.
- `readwritesplit` is the name of the module that implements the Read/Write Split Router.
- `user=MAXSCALE_USER` sets the user parameter to the database user account that MaxScale uses to connect to the ColumnStore nodes.
- `password=MAXSCALE_USER_PASSWORD` sets the password parameter to the password used by the database user account that MaxScale uses to connect to the ColumnStore nodes.
- `--servers` sets the servers parameter to the set of nodes to which MaxScale should route queries. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by `readwritesplit` in MaxScale 22.08 can also be specified.

Configure a Listener for the Read/Write Split Router

These instructions reference TCP port 3307. You can use a different TCP port. The TCP port used must not be bound by any other listener.

On the MaxScale node, use the maxctrl create listener command to configure MaxScale to use a listener for the Read/Write Split Router (readwritesplit):

```
$ maxctrl create listener query_router_service query_router_listener 3307 \
     protocol=MariaDBClient
```

In this example:

- `query_router_service` is the name of the readwritesplit service that was previously created.
- `query_router_listener` is an arbitrary name that is used to identify the new listener.
- 3307 is the TCP port.
- `protocol=MariaDBClient` sets the protocol parameter.
- Other Module Parameters supported by listeners in MaxScale 22.08 can also be specified.

Start Services

To start the services and monitors, on the MaxScale node use maxctrl start services:

```
$ maxctrl start services
```

Next Step

Navigation in the procedure “Deploy ColumnStore Object Storage Topology”:

This page was **step 7 of 9**.

Next: Step 8: Test MariaDB MaxScale

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 8: Test MariaDB MaxScale

Overview

This page details step 8 of the 9-step procedure “Deploy ColumnStore Object Storage Topology”.

This step tests MariaDB MaxScale 22.08.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Check Global Configuration

Use maxctrl show maxscale command to view the global MaxScale configuration.

This action is performed **on the MaxScale node**:

```
$ maxctrl show maxscale
```

```
┌──────────────┬──────────────────────────────────────────────────────────────┐
│ Version      │ 22.08.15                                                     │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Commit       │ 3761fa7a52046bc58faad8b5a139116f9e33364c                     │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Started At   │ Thu, 05 Aug 2021 20:21:20 GMT                                │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Activated At │ Thu, 05 Aug 2021 20:21:20 GMT                                │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Uptime       │ 868                                                          │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Config Sync  │ null                                                         │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Parameters   │ {                                                            │
│              │     "admin_auth": true,                                      │
│              │     "admin_enabled": true,                                   │
│              │     "admin_gui": true,                                       │
│              │     "admin_host": "0.0.0.0",                                 │
│              │     "admin_log_auth_failures": true,                         │
│              │     "admin_pam_readonly_service": null,                      │
│              │     "admin_pam_readwrite_service": null,                     │
│              │     "admin_port": 8989,                                      │
│              │     "admin_secure_gui": false,                               │
│              │     "admin_ssl_ca_cert": null,                               │
│              │     "admin_ssl_cert": null,                                  │
│              │     "admin_ssl_key": null,                                   │
│              │     "admin_ssl_version": "MAX",                              │
│              │     "auth_connect_timeout": "10000ms",                       │
│              │     "auth_read_timeout": "10000ms",                          │
│              │     "auth_write_timeout": "10000ms",                         │
│              │     "cachedir": "/var/cache/maxscale",                       │
│              │     "config_sync_cluster": null,                             │
│              │     "config_sync_interval": "5000ms",                        │
│              │     "config_sync_password": "******",                        │
│              │     "config_sync_timeout": "10000ms",                        │
│              │     "config_sync_user": null,                                │
│              │     "connector_plugindir": "/usr/lib64/mysql/plugin",        │
│              │     "datadir": "/var/lib/maxscale",                          │
│              │     "debug": null,                                           │
│              │     "dump_last_statements": "never",                         │
│              │     "execdir": "/usr/bin",                                   │
│              │     "language": "/var/lib/maxscale",                         │
│              │     "libdir": "/usr/lib64/maxscale",                         │
│              │     "load_persisted_configs": true,                          │
│              │     "local_address": null,                                   │
│              │     "log_debug": false,                                      │
│              │     "log_info": false,                                       │
│              │     "log_notice": true,                                      │
│              │     "log_throttling": {                                      │
│              │         "count": 10,                                         │
│              │         "suppress": 10000,                                   │
│              │         "window": 1000                                       │
│              │     },                                                       │
│              │     "log_warn_super_user": false,                            │
│              │     "log_warning": true,                                     │
│              │     "logdir": "/var/log/maxscale",                           │
│              │     "max_auth_errors_until_block": 10,                       │
│              │     "maxlog": true,                                          │
│              │     "module_configdir": "/etc/maxscale.modules.d",           │
│              │     "ms_timestamp": false,                                   │
│              │     "passive": false,                                        │
│              │     "persistdir": "/var/lib/maxscale/maxscale.cnf.d",        │
│              │     "piddir": "/var/run/maxscale",                           │
│              │     "query_classifier": "qc_sqlite",                         │
│              │     "query_classifier_args": null,                           │
│              │     "query_classifier_cache_size": 289073971,                │
│              │     "query_retries": 1,                                      │
│              │     "query_retry_timeout": "5000ms",                         │
```

```
|               |     "rebalance_period": "0ms",                          |
|               |     "rebalance_threshold": 20,                          |
|               |     "rebalance_window": 10,                             |
|               |     "retain_last_statements": 0,                        |
|               |     "session_trace": 0,                                 |
|               |     "skip_permission_checks": false,                    |
|               |     "sql_mode": "default",                              |
|               |     "syslog": true,                                     |
|               |     "threads": 1,                                       |
|               |     "users_refresh_interval": "0ms",                    |
|               |     "users_refresh_time": "30000ms",                    |
|               |     "writeq_high_water": 16777216,                      |
|               |     "writeq_low_water": 8192                            |
|               | }                                                       |
└───────────────┴─────────────────────────────────────────────────────────┘
```

Output should align to the global MaxScale configuration in the new configuration file you created.

Check Server Configuration

Use the maxctrl list servers and maxctrl show server commands to view the configured server objects.

This action is performed **on the MaxScale node**:

1. Obtain the full list of servers objects:

```
$ maxctrl list servers
```

```
┌────────┬─────────────┬──────┬─────────────┬─────────────────┬──────────┐
| Server | Address     | Port | Connections | State           | GTID     |
├────────┼─────────────┼──────┼─────────────┼─────────────────┼──────────┤
| mcs1   | 192.0.2.1   | 3306 | 1           | Master, Running | 0-1-25 |
├────────┼─────────────┼──────┼─────────────┼─────────────────┼──────────┤
| mcs2   | 192.0.2.2   | 3306 | 1           | Slave, Running  | 0-1-25 |
├────────┼─────────────┼──────┼─────────────┼─────────────────┼──────────┤
| mcs3   | 192.0.2.3   | 3306 | 1           | Slave, Running  | 0-1-25 |
└────────┴─────────────┴──────┴─────────────┴─────────────────┴──────────┘
```

For each server object, view the configuration:

```
$ maxctrl show server mcs1
```

```
┌──────────────────┬──────────────────────────────────────────────┐
│ Server           │ mcs1                                         │
├──────────────────┼──────────────────────────────────────────────┤
│ Address          │ 192.0.2.1                                    │
├──────────────────┼──────────────────────────────────────────────┤
│ Port             │ 3306                                         │
├──────────────────┼──────────────────────────────────────────────┤
│ State            │ Master, Running                              │
├──────────────────┼──────────────────────────────────────────────┤
│ Version          │ 11.4.5-3-MariaDB-enterprise-log              │
├──────────────────┼──────────────────────────────────────────────┤
│ Last Event       │ master_up                                    │
├──────────────────┼──────────────────────────────────────────────┤
│ Triggered At     │ Thu, 05 Aug 2021 20:22:26 GMT                │
├──────────────────┼──────────────────────────────────────────────┤
│ Services         │ connection_router_service                    │
│                  │ query_router_service                         │
├──────────────────┼──────────────────────────────────────────────┤
│ Monitors         │ columnstore_monitor                          │
├──────────────────┼──────────────────────────────────────────────┤
│ Master ID        │ -1                                           │
├──────────────────┼──────────────────────────────────────────────┤
│ Node ID          │ 1                                            │
├──────────────────┼──────────────────────────────────────────────┤
│ Slave Server IDs │                                              │
├──────────────────┼──────────────────────────────────────────────┤
│ Current Connections │ 1                                         │
├──────────────────┼──────────────────────────────────────────────┤
│ Total Connections │ 1                                           │
├──────────────────┼──────────────────────────────────────────────┤
│ Max Connections  │ 1                                            │
├──────────────────┼──────────────────────────────────────────────┤
│ Statistics       │ {                                            │
│                  │     "active_operations": 0,                  │
│                  │     "adaptive_avg_select_time": "0ns",       │
│                  │     "connection_pool_empty": 0,              │
│                  │     "connections": 1,                        │
│                  │     "max_connections": 1,                    │
│                  │     "max_pool_size": 0,                      │
│                  │     "persistent_connections": 0,             │
│                  │     "reused_connections": 0,                 │
│                  │     "routed_packets": 0,                     │
│                  │     "total_connections": 1                   │
│                  │ }                                            │
├──────────────────┼──────────────────────────────────────────────┤
│ Parameters       │ {                                            │
│                  │     "address": "192.0.2.1",                  │
│                  │     "disk_space_threshold": null,            │
│                  │     "extra_port": 0,                         │
│                  │     "monitorpw": null,                       │
│                  │     "monitoruser": null,                     │
│                  │     "persistmaxtime": "0ms",                 │
│                  │     "persistpoolmax": 0,                     │
│                  │     "port": 3306,                            │
│                  │     "priority": 0,                           │
│                  │     "proxy_protocol": false,                 │
│                  │     "rank": "primary",                       │
│                  │     "socket": null,                          │
│                  │     "ssl": false,                            │
│                  │     "ssl_ca_cert": null,                     │
│                  │     "ssl_cert": null,                        │
│                  │     "ssl_cert_verify_depth": 9,              │
│                  │     "ssl_cipher": null,                      │
│                  │     "ssl_key": null,                         │
│                  │     "ssl_verify_peer_certificate": false,    │
│                  │     "ssl_verify_peer_host": false,           │
│                  │     "ssl_version": "MAX"                     │
```

```
│                     │ }                                                  │
└─────────────────────┴────────────────────────────────────────────────────┘
```

Output should align to the Server Object configuration you performed.

Check Monitor Configuration

Use the maxctrl list monitors and maxctrl show monitor commands to view the configured monitors.

This action is performed on the MaxScale node:

1. Obtain the full list of monitors:

```
$ maxctrl list monitors
```

```
┌─────────────────────┬─────────┬──────────────────┐
│ Monitor             │ State   │ Servers          │
├─────────────────────┼─────────┼──────────────────┤
│ columnstore_monitor │ Running │ mcs1, mcs2, mcs3 │
└─────────────────────┴─────────┴──────────────────┘
```

2. For each monitor, view the monitor configuration:

```
$ maxctrl show monitor columnstore_monitor
```

```
┌─────────────────────┬──────────────────────────────────────────┐
│ Monitor             │ columnstore_monitor                      │
├─────────────────────┼──────────────────────────────────────────┤
│ Module              │ mariadbmon                               │
├─────────────────────┼──────────────────────────────────────────┤
│ State               │ Running                                  │
├─────────────────────┼──────────────────────────────────────────┤
│ Servers             │ mcs1                                     │
│                     │ mcs2                                     │
│                     │ mcs3                                     │
├─────────────────────┼──────────────────────────────────────────┤
│ Parameters          │ {                                        │
│                     │     "backend_connect_attempts": 1,       │
│                     │     "backend_connect_timeout": 3,        │
│                     │     "backend_read_timeout": 3,           │
│                     │     "backend_write_timeout": 3,          │
│                     │     "disk_space_check_interval": 0,      │
│                     │     "disk_space_threshold": null,        │
│                     │     "events": "all",                     │
│                     │     "journal_max_age": 28800,            │
│                     │     "module": "mariadbmon",              │
│                     │     "monitor_interval": 2000,            │
│                     │     "password": "******",                │
│                     │     "script": null,                      │
│                     │     "script_timeout": 90,                │
│                     │     "user": "mxs"                        │
│                     │ }                                        │
├─────────────────────┼──────────────────────────────────────────┤
│ Monitor Diagnostics │ {}                                       │
└─────────────────────┴──────────────────────────────────────────┘
```

Output should align to the MariaDB Monitor (mariadbmon) configuration you performed.

Check Service Configuration

Use the maxctrl list services and maxctrl show service commands to view the configured routing services.

This action is performed on the MaxScale node:

1. Obtain the full list of routing services:

```
$ maxctrl list services
```

```
┌──────────────────────────┬─────────────┬─────────────┬─────────────────┬──────────────────┐
│ Service                  │ Router      │ Connections │ Total Connections │ Servers          │
├──────────────────────────┼─────────────┼─────────────┼─────────────────┼──────────────────┤
│ connection_router_Service │ readconnroute │ 0           │ 0               │ mcs1, mcs2, mcs3 │
├──────────────────────────┼─────────────┼─────────────┼─────────────────┼──────────────────┤
│ query_router_service     │ readwritesplit │ 0           │ 0               │ mcs1, mcs2, mcs3 │
└──────────────────────────┴─────────────┴─────────────┴─────────────────┴──────────────────┘
```

2. For each service, view the service configuration:

```
$ maxctrl show service query_router_service
```

```
┌────────────────────┬──────────────────────────────────────────────────────────────────────────────┐
│ Service            │ query_router_service                                                         │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Router             │ readwritesplit                                                               │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ State              │ Started                                                                      │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Started At         │ Sat Aug 28 21:41:16 2021                                                     │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Current Connections │ 0                                                                           │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Total Connections  │ 0                                                                            │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Max Connections    │ 0                                                                            │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Cluster            │                                                                              │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Servers            │ mcs1                                                                         │
│                    │ mcs2                                                                         │
│                    │ mcs3                                                                         │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Services           │                                                                              │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Filters            │                                                                              │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Parameters         │ {                                                                            │
│                    │     "auth_all_servers": false,                                               │
│                    │     "causal_reads": "false",                                                 │
│                    │     "causal_reads_timeout": "10000ms",                                       │
│                    │     "connection_keepalive": "300000ms",                                      │
│                    │     "connection_timeout": "0ms",                                             │
│                    │     "delayed_retry": false,                                                  │
│                    │     "delayed_retry_timeout": "10000ms",                                      │
│                    │     "disable_sescmd_history": false,                                         │
│                    │     "enable_root_user": false,                                               │
│                    │     "idle_session_pool_time": "-1000ms",                                     │
│                    │     "lazy_connect": false,                                                   │
│                    │     "localhost_match_wildcard_host": true,                                   │
│                    │     "log_auth_warnings": true,                                               │
│                    │     "master_accept_reads": false,                                            │
│                    │     "master_failure_mode": "fail_instantly",                                 │
│                    │     "master_reconnection": false,                                            │
│                    │     "max_connections": 0,                                                    │
│                    │     "max_sescmd_history": 50,                                                │
│                    │     "max_slave_connections": 255,                                            │
│                    │     "max_slave_replication_lag": "0ms",                                      │
│                    │     "net_write_timeout": "0ms",                                              │
│                    │     "optimistic_trx": false,                                                 │
│                    │     "password": "******",                                                    │
│                    │     "prune_sescmd_history": true,                                            │
│                    │     "rank": "primary",                                                       │
│                    │     "retain_last_statements": -1,                                            │
│                    │     "retry_failed_reads": true,                                              │
│                    │     "reuse_prepared_statements": false,                                      │
│                    │     "router": "readwritesplit",                                              │
│                    │     "session_trace": false,                                                  │
│                    │     "session_track_trx_state": false,                                        │
│                    │     "slave_connections": 255,                                                │
│                    │     "slave_selection_criteria": "LEAST_CURRENT_OPERATIONS",                  │
│                    │     "strict_multi_stmt": false,                                              │
│                    │     "strict_sp_calls": false,                                                │
│                    │     "strip_db_esc": true,                                                    │
│                    │     "transaction_replay": false,                                             │
│                    │     "transaction_replay_attempts": 5,                                        │
│                    │     "transaction_replay_max_size": 1073741824,                               │
│                    │     "transaction_replay_retry_on_deadlock": false,                           │
│                    │     "type": "service",                                                       │
│                    │     "use_sql_variables_in": "all",                                           │
```

```
|                      |     "user": "mxs",                                                  |
|                      |     "version_string": null                                          |
|                      | }                                                                   |
├──────────────────────┼─────────────────────────────────────────────────────────────────────┤
| Router Diagnostics   | {                                                                   |
|                      |     "avg_sescmd_history_length": 0,                                 |
|                      |     "max_sescmd_history_length": 0,                                 |
|                      |     "queries": 0,                                                   |
|                      |     "replayed_transactions": 0,                                     |
|                      |     "ro_transactions": 0,                                           |
|                      |     "route_all": 0,                                                 |
|                      |     "route_master": 0,                                              |
|                      |     "route_slave": 0,                                               |
|                      |     "rw_transactions": 0,                                           |
|                      |     "server_query_statistics": []                                   |
|                      | }                                                                   |
└──────────────────────┴─────────────────────────────────────────────────────────────────────┘
```

Output should align to the Read Connection Router (readconnroute) or Read/Write Split Router (readwritesplit) configuration you performed.

Test Application User

Applications should use a dedicated user account. The user account must be created on the primary server.

When users connect to MaxScale, MaxScale authenticates the user connection before routing it to an Enterprise Server node. Enterprise Server authenticates the connection as originating from the IP address of the MaxScale node.

The application users must have one user account with the host IP address of the application server and a second user account with the host IP address of the MaxScale node.

The requirement of a duplicate user account can be avoided by enabling the `proxy_protocol` parameter for MaxScale and the proxy_protocol_networks for Enterprise Server.

Create a User to Connect from MaxScale

This action is performed on the primary Enterprise ColumnStore node:

1. Connect to the primary Enterprise ColumnStore node:

```
$ sudo mariadb
```

2. Create the database user account for your MaxScale node:

```
CREATE USER 'app_user'@'192.0.2.10' IDENTIFIED BY 'app_user_passwd';
```

Replace 192.0.2.10 with the relevant IP address specification for your MaxScale node.

Passwords should meet your organization's password policies.

3. Grant the privileges required by your application to the database user account for your MaxScale node:

```
GRANT ALL ON test.* TO 'app_user'@'192.0.2.10';
```

The privileges shown are designed to allow the tests in the subsequent sections to work. The user account for your production application may require different privileges.

Create a User to Connect from the Application Server

This action is performed on the primary Enterprise ColumnStore node:

1. Create the database user account for your application server:

```
CREATE USER 'app_user'@'192.0.2.11' IDENTIFIED BY 'app_user_passwd';
```

Replace 192.0.2.11 with the relevant IP address specification for your application server.

Passwords should meet your organization's password policies.

2. Grant the privileges required by your application to the d database user account for your application server:

```
GRANT ALL ON test.* TO 'app_user'@'192.0.2.11';
```

The privileges shown are designed to allow the tests in the subsequent sections to work. The user account for your production application may require different privileges.

Test Connection with Application User

To test the connection, use the MariaDB Client from your application server to connect to an Enterprise ColumnStore node through MaxScale.

This action is performed **on a client connected to the MaxScale node**:

```
$ mariadb --host 192.0.2.10 --port 3307
      --user app_user --password
```

Test Connection with Read Connection Router

If you configured the Read Connection Router, confirm that MaxScale routes connections to the replica servers.

On the MaxScale node, use the maxctrl list listeners command to view the available listeners and ports:

```
$ maxctrl list listeners
```

```
┌──────────────────────────┬──────┬──────┬─────────┬──────────────────────────┐
│ Name                     │ Port │ Host │ State   │ Service                  │
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ connection_router_listener │ 3308 │ ::   │ Running │ connection_router_service │
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ query_router_listener    │ 3307 │ ::   │ Running │ query_router_service     │
└──────────────────────────┴──────┴──────┴─────────┴──────────────────────────┘
```

2. Open multiple terminals connected to your application server, in each, use MariaDB Client to connect to the listener port for the Read Connection Router (in the example, 3308):

```
$ mariadb --host 192.0.2.10 --port 3308 \
      --user app_user --password
```

Use the application user credentials you created for the `--user and --password` options.

3. In each terminal, query the hostname and server_id system variable and option to identify to which you're connected:

```
SELECT @@global.hostname, @@global.server_id;

+-------------------+--------------------+
| @@global.hostname | @@global.server_id |
+-------------------+--------------------+
|              mcs2 |                  2 |
+-------------------+--------------------+
```

Different terminals should return different values since MaxScale routes the connections to different nodes.

Since the router was configured with the slave router option, the Read Connection Router only routes connections to replica servers.

Test Write Queries with Read/Write Split Router

If you configured the Read/Write Split Router, confirm that MaxScale routes write queries on this router to the primary Enterprise ColumnStore node.

on the MaxScale node, use the maxctrl list listeners command to view the available listeners and ports:

```
$ maxctrl list listeners
```

```
┌──────────────────────────┬──────┬──────┬─────────┬──────────────────────────┐
| Name                     | Port | Host | State   | Service                  |
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
| connection_router_listener | 3308 | ::   | Running | connection_router_service |
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
| query_router_listener    | 3307 | ::   | Running | query_router_service     |
└──────────────────────────┴──────┴──────┴─────────┴──────────────────────────┘
```

2. Open multiple terminals connected to your application server, in each, use MariaDB Client to connect to the listener port for the Read/Write Split Router (in the example, 3307):

```
$ mariadb --host 192.0.2.10 --port 3307 \
      --user app_user --password
```

Use the application user credentials you created for the `--user and --password` options.

3. In one terminal, create the test table:

```
CREATE TABLE test.load_balancing_test (
   id INT PRIMARY KEY AUTO_INCREMENT,
   hostname VARCHAR(256),
   server_id INT
);
```

4. In each terminal, issue an insert.md statement to add a row to the example table with the values of the hostname and server_id system variable and option:

```
INSERT INTO test.load_balancing_test (hostname, server_id)
VALUES (@@global.hostname, @@global.server_id);
```

5. In one terminal, issue a SELECT statement to query the results:

```
SELECT * FROM test.load_balancing_test;
```

```
+----+----------+-----------+
| id | hostname | server_id |
+----+----------+-----------+
|  1 | mcs1     |         1 |
|  2 | mcs1     |         1 |
|  3 | mcs1     |         1 |
+----+----------+-----------+
```

While MaxScale is handling multiple connections from different terminals, it routed all connections to the current primary Enterprise ColumnStore node, which in the example is `mcs1#.`

Test Read Queries with Read/Write Split Router

If you configured the Read/Write Split Router (readwritesplit), confirm that MaxScale routes read queries on this router to replica servers.

1. On the MaxScale node, use the maxctrl list listeners command to view the available listeners and ports:

```
$ maxctrl list listeners
```

```
┌──────────────────────────┬──────┬──────┬─────────┬──────────────────────────┐
│ Name                     │ Port │ Host │ State   │ Service                  │
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ connection_router_listener │ 3308 │ ::   │ Running │ connection_router_service │
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ query_router_listener    │ 3307 │ ::   │ Running │ query_router_service     │
└──────────────────────────┴──────┴──────┴─────────┴──────────────────────────┘
```

2. In a terminal connected to your application server, use MariaDB Client to connect to the listener port for the Read/Write Split Router (readwritesplit) (in the example, 3307):

```
$ mariadb --host 192.0.2.10 --port 3307 \
      --user app_user --password
```

Use the application user credentials you created for the –user and –password options.

3. Query the hostname and server_id to identify which server MaxScale routed you to.

```
SELECT @@global.hostname, @@global.server_id;
```

```
+-------------------+--------------------+
| @@global.hostname | @@global.server_id |
+-------------------+--------------------+
|              mcs2 |                  2 |
+-------------------+--------------------+
```

4. Resend the query:

```
SELECT @@global.hostname, @@global.server_id;
```

```
+-------------------+--------------------+
| @@global.hostname | @@global.server_id |
+-------------------+--------------------+
|              mcs3 |                  3 |
+-------------------+--------------------+
```

Confirm that MaxScale routes the SELECT statements to different replica servers.

For more information on different routing criteria, see slave_selection_criteria

Next Step

Navigation in the procedure “Deploy ColumnStore Object Storage Topology”:

This page was **step 8 of 9**.

Next: Step 9: Import Data

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 9: Import Data

Overview

This page details step 9 of the 9-step procedure "Deploy ColumnStore Object Storage Topology".

This step bulk imports data to Enterprise ColumnStore.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Import the Schema

Before data can be imported into the tables, create a matching schema.

On the primary server, create the schema:

1. For each database that you are importing, create the database with the CREATE DATABASE statement:

```
CREATE DATABASE inventory;
```

2. For each table that you are importing, create the table with the CREATE TABLE statement:

```
CREATE TABLE inventory.products (
   product_name VARCHAR(11) NOT NULL DEFAULT '',
   supplier VARCHAR(128) NOT NULL DEFAULT '',
   quantity VARCHAR(128) NOT NULL DEFAULT '',
   unit_cost VARCHAR(128) NOT NULL DEFAULT ''
) ENGINE=Columnstore DEFAULT CHARSET=utf8;
```

Import the Data

Enterprise ColumnStore supports multiple methods to import data into ColumnStore tables.

Interface	Method	Benefits
Shell	cpimport	• SQL access is not required
SQL	LOAD DATA INFILE	• Shell access is not required
Remote Database	Remote Database Import	• Use normal database client • Avoid dumping data to intermediate filed

cpimport

MariaDB Enterprise ColumnStore includes cpimport, which is a command-line utility designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a TSV (tab-separated values) file, on the primary server run cpimport:

```
$ sudo cpimport -s '\t' inventory products /tmp/inventory-products.tsv
```

LOAD DATA INFILE

When data is loaded with the LOAD DATA INFILE statement, MariaDB Enterprise ColumnStore loads the data using cpimport, which is a command-line utility designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a TSV (tab-separated values) file, on the primary server use LOAD DATA INFILE statement:

```
LOAD DATA INFILE '/tmp/inventory-products.tsv'
INTO TABLE inventory.products;
```

Import from Remote Database

MariaDB Enterprise ColumnStore can also import data directly from a remote database. A simple method is to query the table using the SELECT statement, and then pipe the results into cpimport, which is a command-line utility that is designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a remote MariaDB database:

```
$ mariadb --quick \
   --skip-column-names \
   --execute="SELECT * FROM inventory.products" \
   | cpimport -s '\t' inventory products
```

Next Step

Navigation in the procedure “Deploy ColumnStore Object Storage Topology”:

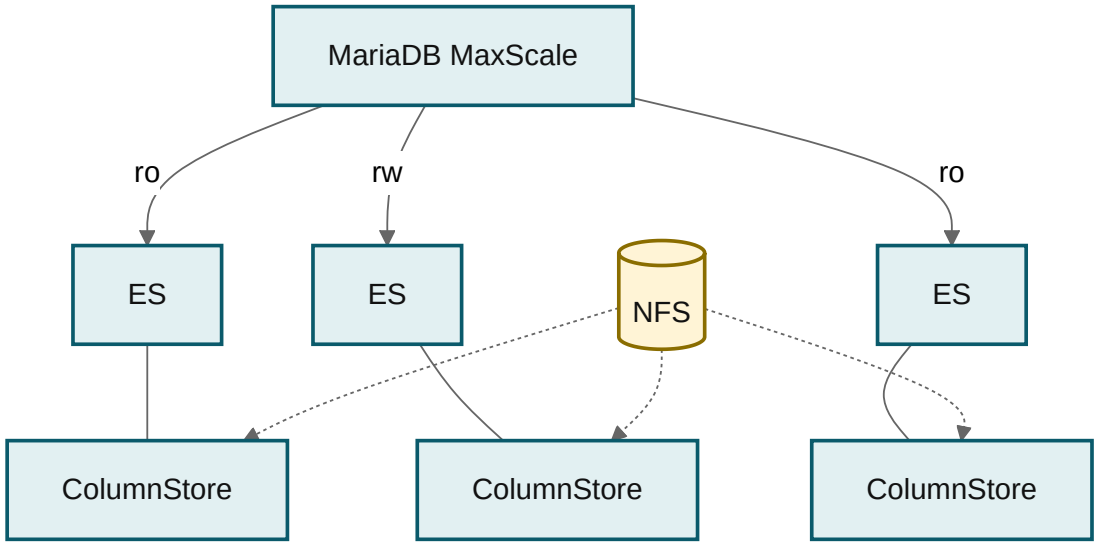
This page was **step 9 of 9**.

This procedure is complete.

This page is: Copyright © 2026 MariaDB. All rights reserved.

ColumnStore Shared Local Storage

Overview



MaxScale routes to a three-node ColumnStore cluster sharing NFS storage.

- Enterprise Server 10.5
- Enterprise Server 10.6
- Enterprise Server 11.4

Columnar storage engine with S3-compatible object storage

- Highly available
- Automatic failover via MaxScale and CMAPI
- Scales reads via MaxScale
- Bulk data import
- Enterprise Server 10.5, Enterprise ColumnStore 5, MaxScale 2.5

- Enterprise Server 10.6, Enterprise ColumnStore 23.02, MaxScale 22.08

This procedure describes the deployment of the ColumnStore Shared Local Storage topology with MariaDB Enterprise Server 10.5, MariaDB Enterprise ColumnStore 5, and MariaDB MaxScale 2.5.

MariaDB Enterprise ColumnStore 5 is a columnar storage engine for MariaDB Enterprise Server 10.5. Enterprise ColumnStore is suitable for Online Analytical Processing (OLAP) workloads.

This procedure has 9 steps, which are executed in sequence.

This procedure represents basic product capability and deploys 3 Enterprise ColumnStore nodes and 1 MaxScale node.

This page provides an overview of the topology, requirements, and deployment procedures.

Please read and understand this procedure before executing.

Procedure Steps

Step	Description
Step 1	Prepare ColumnStore Nodes
Step 2	Configure Shared Local Storage
Step 3	Install MariaDB Enterprise Server
Step 4	Start and Configure MariaDB Enterprise Server
Step 5	Test MariaDB Enterprise Server
Step 6	Install MariaDB MaxScale
Step 7	Start and Configure MariaDB MaxScale
Step 8	Test MariaDB MaxScale
Step 9	Import Data

Support

Customers can obtain support by submitting a support case.

Components

The following components are deployed during this procedure:

Component	Function
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.
MariaDB MaxScale	Database proxy that extends the availability, scalability, and security of MariaDB Enterprise Servers

MariaDB Enterprise Server Components

Component	Description
MariaDB Enterprise ColumnStore	• Columnar storage engine • Highly available • Optimized for Online Analytical Processing (OLAP) workloads • Scalable query execution • Cluster Management API (CMAPI) provides a REST API for multi-node administration.

MariaDB MaxScale Components

Component	Description
Listener	Listens for client connections to MaxScale then passes them to the router service

Component	Description
MariaDB Monitor	Tracks changes in the state of MariaDB Enterprise Servers.
Read Connection Router	Routes connections from the listener to any available Enterprise ColumnStore node
Read/Write Split Router	Routes read operations from the listener to any available Enterprise ColumnStore node, and routes write operations from the listener to a specific server that MaxScale uses as the primary server
Server Module	Connection configuration in MaxScale to an Enterprise ColumnStore node

Topology

The MariaDB Enterprise ColumnStore topology with Object Storage delivers production analytics with high availability, fault tolerance, and limitless data storage by leveraging S3-compatible storage.

The topology consists of:

- One or more MaxScale nodes
- An odd number of ColumnStore nodes (minimum of 3) running ES, Enterprise ColumnStore, and CMAPI

The MaxScale nodes:

- Monitor the health and availability of each ColumnStore node using the MariaDB Monitor (mariadbmon)
- Accept client and application connections
- Route queries to ColumnStore nodes using the Read/Write Split Router (readwritesplit)

The ColumnStore nodes:

- Receive queries from MaxScale
- Execute queries
  - Use shared local storage for the Storage Manager directory

Requirements

These requirements are for the ColumnStore Object Storage topology when deployed with MariaDB Enterprise Server 10.5, MariaDB Enterprise ColumnStore 5, and MariaDB MaxScale 2.5.

- Node Count
- Operating System
- Minimum Hardware Requirements
- Recommended Hardware Requirements
- Storage Requirements
- S3-Compatible Object Storage Requirements
- Preferred Object Storage Providers: Cloud
- Preferred Object Storage Providers: Hardware
- Shared Local Storage Directories
- Shared Local Storage Options
- Recommended Storage Options

Node Count

- MaxScale nodes, 1 or more are required.
- Enterprise ColumnStore nodes, 3 or more are required for high availability. You should always have an odd number of nodes in a multi-node ColumnStore deployment to avoid split brain scenarios.

Operating System

In alignment to the enterprise lifecycle, the ColumnStore Object Storage topology with MariaDB Enterprise Server 10.5, MariaDB Enterprise ColumnStore 5, and MariaDB MaxScale 2.5 is provided for:

- CentOS Linux 7 (x86_64)
- Debian 10 (x86_64)
- Red Hat Enterprise Linux 7 (x86_64)
- Red Hat Enterprise Linux 8 (x86_64)
- Ubuntu 18.04 LTS (x86_64)
- Ubuntu 20.04 LTS (x86_64)

Minimum Hardware Requirements

MariaDB Enterprise ColumnStore's minimum hardware requirements are not intended for production environments, but the minimum hardware requirements can be appropriate for development and test environments. For production environments, see the recommended hardware requirements instead.

The minimum hardware requirements are:

Component	CPU	Memory
MaxScale node	4+ cores	4+ GB
Enterprise ColumnStore node	4+ cores	4+ GB

MariaDB Enterprise ColumnStore will refuse to start if the system has less than 3 GB of memory.

If Enterprise ColumnStore is started on a system with less memory, the following error message will be written to the ColumnStore system log called crit.log:

```
Apr 30 21:54:35 a1ebc96a2519 PrimProc[1004]: 35.668435 |0|0|0| C 28 CAL0000: Error total memory available
is less than 3GB.
```

And the following error message will be raised to the client:

```
ERROR 1815 (HY000): Internal error: System is not ready yet. Please try again.
```

Recommended Hardware Requirements

MariaDB Enterprise ColumnStore's recommended hardware requirements are intended for production analytics.

The recommended hardware requirements are:

Component	CPU	Memory
MaxScale node	8+ cores	16+ GB
Enterprise ColumnStore node	64+ cores	128+ GB

Storage Requirements

The ColumnStore Object Storage topology requires the following storage types:

Storage Type	Description
Shared Local Storage	The ColumnStore Object Storage topology uses shared local storage for the Storage Manager directory to store metadata.

Shared Local Storage Directories

The ColumnStore Object Storage topology uses shared local storage for the Storage Manager directory to store metadata.

The Storage Manager directory is located at the following path by default:

- `/var/lib/columnstore/storagemanager`

Shared Local Storage Options

The most common shared local storage options for the ColumnStore Object Storage topology are:

Shared Local Storage	Common Usage	Description
EBS (Elastic Block Store) Multi-Attach	AWS	• EBS is a high-performance block-storage service for AWS (Amazon Web Services). • EBS Multi-Attach allows an EBS volume to be attached to multiple instances in AWS. Only clustered file systems, such as GFS2, are supported. • For deployments in AWS, EBS Multi-Attach is a recommended option for the Storage Manager directory, and Amazon S3 storage is the recommended option for data.
EFS (Elastic File System)	AWS	• EFS is a scalable, elastic, cloud-native NFS file system for AWS (Amazon Web Services). • For deployments in AWS, EFS is a recommended option for the Storage Manager directory, and Amazon S3 storage is the recommended option for data. EFS is a scalable, elastic, cloud-native NFS file system for AWS (Amazon Web Services).
Filestore	GCP	• Filestore is high-performance, fully managed storage for GCP (Google Cloud Platform). • For deployments in GCP, Filestore is the recommended option for the Storage Manager directory, and Google Object Storage (S3-compatible) is the recommended option for data.
GlusterFS	On-premises	• GlusterFS is a distributed file system. • GlusterFS supports replication and failover.
NFS (Network File System)	On-premises	• NFS is a distributed file system. • If NFS is used, the storage should be mounted with the sync option to ensure that each node flushes its changes immediately. • For on-premises deployments, NFS is the recommended option for the Storage Manager directory, and any S3-compatible storage is the recommended option for data.

Enterprise ColumnStore Management with CMAPI

Enterprise ColumnStore's CMAPI (Cluster Management API) is a REST API that can be used to manage a multi-node Enterprise ColumnStore cluster.

Many tools are capable of interacting with REST APIs. For example, the curl utility could be used to make REST API calls from the command-line.

Many programming languages also have libraries for interacting with REST APIs.

The examples below show how to use the CMAPI with curl.

URL Endpoint Format for REST API

```
https://{server}:{port}/cmapi/{version}/{route}/{command}
```

For example:

- shutdown
- start
- status

Required Request Headers

- `'x-api-key'`: '93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd'
- `'Content-Type'`: 'application/json'

x-api-key can be set to any value of your choice during the first call to the server. Subsequent connections will require this same key.

Get Status

```
$ curl -k -s https://mcs1:8640/cmapi/0.4.0/cluster/status \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      | jq .
```

Start Cluster

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/start \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20}' \
      | jq .
```

Stop Cluster

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/shutdown \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20}' \
      | jq .
```

Add Node

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/node \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20, "node": "192.0.2.2"}' \
      | jq .
```

Remove Node

```
$ curl -k -s -X DELETE https://mcs1:8640/cmapi/0.4.0/cluster/node \
      --header 'Content-Type:application/json' \
      --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
      --data '{"timeout":20, "node": "192.0.2.2"}' \
      | jq .
```

Quick Reference

MariaDB Enterprise Server Configuration Management

Method	Description
Configuration File	Configuration files (such as /etc/my.cnf) can be used to set system-variables and options. The server must be restarted to apply changes made to configuration files.
Command-line	The server can be started with command-line options that set system-variables and options.
SQL	Users can set system-variables that support dynamic changes on-the-fly using the SET statement.

MariaDB Enterprise Server packages are configured to read configuration files from different paths, depending on the operating system. Making custom changes to Enterprise Server default configuration files is not recommended because custom changes may be overwritten by other default configuration files that are loaded later.

To ensure that your custom changes will be read last, create a custom configuration file with the z- prefix in one of the include directories.

Distribution	Example Configuration File Path
• CentOS • Red Hat Enterprise Linux (RHEL)	/etc/my.cnf.d/z-custom-mariadb.cnf
• Debian • Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

MariaDB Enterprise Server Service Management

The systemctl command is used to start and stop the MariaDB Enterprise Server service.

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

For additional information, see “Starting and Stopping MariaDB”.

MariaDB Enterprise Server Logs

MariaDB Enterprise Server produces log data that can be helpful in problem diagnosis.

Log filenames and locations may be overridden in the server configuration. The default location of logs is the data directory. The data directory is specified by the datadir system variable.

Log	System Variable/Option	Default Filename
MariaDB Error Log	log_error	`<hostname>.err`
MariaDB Enterprise Audit Log	server_audit_file_path	`server_audit.log`
Slow Query Log	slow_query_log_file	`<hostname>-slow.log`
General Query Log	general_log_file	`<hostname>.log`
Binary Log	log_bin	`<hostname>-bin`

Enterprise ColumnStore Service Management

The systemctl command is used to start and stop the ColumnStore service.

Operation	Command
Start	`sudo systemctl start mariadb-columnstore`
Stop	`sudo systemctl stop mariadb-columnstore`
Restart	`sudo systemctl restart mariadb-columnstore`
Enable during startup	`sudo systemctl enable mariadb-columnstore`

Operation	Command
Disable during startup	`sudo systemctl disable mariadb-columnstore`
Status	`sudo systemctl status mariadb-columnstore`

In the ColumnStore Object Storage topology, the mariadb-columnstore service should not be enabled. The CMAPI service restarts Enterprise ColumnStore as needed, so it does not need to start automatically upon reboot.

Enterprise ColumnStore CMAPI Service Management

The systemctl command is used to start and stop the CMAPI service.

Operation	Command
Start	`sudo systemctl start mariadb-columnstore-cmapi`
Stop	`sudo systemctl stop mariadb-columnstore-cmapi`
Restart	`sudo systemctl restart mariadb-columnstore-cmapi`
Enable during startup	`sudo systemctl enable mariadb-columnstore-cmapi`
Disable during startup	`sudo systemctl disable mariadb-columnstore-cmapi`
Status	`sudo systemctl status mariadb-columnstore-cmapi`

For additional information on endpoints, see “CMAPI”.

MaxScale Configuration Management

MaxScale can be configured using several methods. These methods make use of MaxScale’s REST API.

Method	Benefits
MaxCtrl	Command-line utility to perform administrative tasks through the REST API. See MaxCtrl Commands.
MaxGUI	MaxGUI is a graphical utility that can perform administrative tasks through the REST API.
REST API	The REST API can be used directly. For example, the curl utility could be used to make REST API calls from the command-line. Many programming languages also have libraries to interact with REST APIs.

The procedure on these pages configures MaxScale using MaxCtrl.

MaxScale Service Management

The systemctl command is used to start and stop the MaxScale service.>

Operation	Command
Start	`sudo systemctl start maxscale`
Stop	`sudo systemctl stop maxscale`
Restart	`sudo systemctl restart maxscale`
Enable during startup	`sudo systemctl enable maxscale`
Disable during startup	`sudo systemctl disable maxscale`
Status	`sudo systemctl status maxscale`

For additional information, see “Start and Stop Services”.

Next Step

Navigation in the procedure Shared Local Storage topology

Next: Step 1: Prepare ColumnStore Nodes.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 1: Prepare ColumnStore Nodes

Overview

This page details step 1 of the 9-step procedure "Deploy ColumnStore Shared Local Storage Topology".

This step prepares systems to host MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Optimize Linux Kernel Parameters

MariaDB Enterprise ColumnStore performs best with Linux kernel optimizations.

On each server to host an Enterprise ColumnStore node, optimize the kernel:

1. Set the relevant kernel parameters in a sysctl configuration file. To ensure proper change management, use an Enterprise ColumnStore-specific configuration file.

   Create `a /etc/sysctl.d/90-mariadb-enterprise-columnstore.conf file`:

```
# minimize swapping
vm.swappiness = 1

# Increase the TCP max buffer size
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216

# Increase the TCP buffer limits
# min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

# don't cache ssthresh from previous connection
net.ipv4.tcp_no_metrics_save = 1

# for 1 GigE, increase this to 2500
# for 10 GigE, increase this to 30000
net.core.netdev_max_backlog = 2500
```

2. Use the sysctl command to set the kernel parameters at runtime

```
$ sudo sysctl --load=/etc/sysctl.d/90-mariadb-enterprise-columnstore.conf
```

Temporarily Configure Linux Security Modules (LSM)

The Linux Security Modules (LSM) should be temporarily disabled on each Enterprise ColumnStore node during installation.

The LSM will be configured and re-enabled later in this deployment procedure.

The steps to disable the LSM depend on the specific LSM used by the operating system.

CentOS / RHEL Stop SELinux

SELinux must be set to permissive mode before installing MariaDB Enterprise ColumnStore.

To set SELinux to permissive mode:

1. Set SELinux to permissive mode:

```
$ sudo setenforce permissive
```

2. Set SELinux to permissive mode by setting SELINUX=permissive in /etc/selinux/config.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. Confirm that SELinux is in permissive mode:

```
$ sudo getenforce
Permissive
```

SELinux will be configured and re-enabled later in this deployment procedure. This configuration is not persistent. If you restart the server before configuring and re-enabling SELinux later in the deployment procedure, you must reset the enforcement to permissive mode.

Debian / Ubuntu AppArmor

AppArmor must be disabled before installing MariaDB Enterprise ColumnStore.

1. Disable AppArmor:

```
$ sudo systemctl disable apparmor
```

2. Reboot the system.
3. Confirm that no AppArmor profiles are loaded using aa-status:

```
$ sudo aa-status
```

```
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

AppArmor will be configured and re-enabled later in this deployment procedure.

Temporarily Configure Firewall for Installation

MariaDB Enterprise ColumnStore requires the following TCP ports:

TCP Ports	Description
3306	Port used for MariaDB Client traffic
8600-8630	Port range used for inter-node communication
8640	Port used by CMAPI
8700	Port used for inter-node communication
8800	Port used for inter-node communication

The firewall should be temporarily disabled on each Enterprise ColumnStore node during installation.

The firewall will be configured and re-enabled later in this deployment procedure.

The steps to disable the firewall depend on the specific firewall used by the operating system.

CentOS / RHEL Stop firewalld

1. Check if the firewalld service is running:

```
$ sudo systemctl status firewalld
```

2. If the firewalld service is running, stop it:

```
$ sudo systemctl stop firewalld
```

Firewalld will be configured and re-enabled later in this deployment procedure.

Ubuntu Stop UFW

1. Check if the UFW service is running:

```
$ sudo ufw status verbose
```

2. If the UFW service is running, stop it:

```
$ sudo ufw disable
```

UFW will be configured and re-enabled later in this deployment procedure.

Configure the AWS Security Group

To install Enterprise ColumnStore on Amazon Web Services (AWS), the security group must be modified prior to installation.

Enterprise ColumnStore requires all internal communications to be open between Enterprise ColumnStore nodes. Therefore, the security group should allow all protocols and all ports to be open between the Enterprise ColumnStore nodes and the MaxScale proxy.

Configure Character Encoding

When using MariaDB Enterprise ColumnStore, it is recommended to set the system's locale to UTF-8.

1. On RHEL 8, install additional dependencies:

```
$ sudo yum install glibc-locale-source glibc-langpack-en
```

2. Set the system's locale to en_US.UTF-8 by executing localedef:

```
$ sudo localedef -i en_US -f UTF-8 en_US.UTF-8
```

Configure DNS

MariaDB Enterprise ColumnStore requires all nodes to have host names that are resolvable on all other nodes. If your infrastructure does not configure DNS centrally, you may need to configure static DNS entries in the /etc/hosts file of each server.

On each Enterprise ColumnStore node, edit the /etc/hosts file to map host names to the IP address of each Enterprise ColumnStore node:

```
192.0.2.1     mcs1
192.0.2.2     mcs2
192.0.2.3     mcs3
192.0.2.100   mxs1
```

Replace the IP addresses with the addresses in your own environment.

Next Step

Navigation in the procedure “Deploy ColumnStore Shared Local Storage Topology”.

This page was step 1 of 9.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 2: Configure Shared Local Storage

Overview

This page details step 2 of the 9-step procedure “Deploy ColumnStore Shared Local Storage Topology”.

This step configures shared local storage on systems hosting Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Directories for Shared Local Storage

In a ColumnStore Object Storage topology, MariaDB Enterprise ColumnStore requires the Storage Manager directory to be located on shared local storage.

The Storage Manager directory is at the following path:

- `/var/lib/columnstore/storagemanager`

The N in dataN represents a range of integers that starts at 1 and stops at the number of nodes in the deployment. For example, with a 3-node Enterprise ColumnStore deployment, this would refer to the following directories:

- `/var/lib/columnstore/data1`
- `/var/lib/columnstore/data2`
- `/var/lib/columnstore/data3`

The DB Root directories must be mounted on every ColumnStore node.

Choose a Shared Local Storage Solution

Select a Shared Local Storage solution for the Storage Manager directory:

- EBS (Elastic Block Store) Multi-Attach
- EFS (Elastic File System)
- Filestore
- GlusterFS
- NFS (Network File System)

For additional information, see “Shared Local Storage Options”.

Configure EBS Multi-Attach

EBS is a high-performance block-storage service for AWS (Amazon Web Services). EBS Multi-Attach allows an EBS volume to be attached to multiple instances in AWS. Only clustered file systems, such as GFS2, are supported.

For Enterprise ColumnStore deployments in AWS:

- EBS Multi-Attach is a recommended option for the Storage Manager directory.
- Amazon S3 storage is the recommended option for data.
- Consult the vendor documentation for details on how to configure EBS Multi-Attach.

Configure Elastic File System (EFS)

EFS is a scalable, elastic, cloud-native NFS file system for AWS (Amazon Web Services)

For deployments in AWS:

- EFS is a recommended option for the Storage Manager directory.
- Amazon S3 storage is the recommended option for data.
- Consult the vendor documentation for details on how to configure EFS.

Configure Filestore

Filestore is high-performance, fully managed storage for GCP (Google Cloud Platform).

For Enterprise ColumnStore deployments in GCP:

- Filestore is the recommended option for the Storage Manager directory.
- Google Object Storage (S3-compatible) is the recommended option for data.
- Consult the vendor documentation for details on how to configure Filestore.

Configure GlusterFS

GlusterFS is a distributed file system.

GlusterFS is a shared local storage option, but it is not one of the recommended options.

For more information, see “Recommended Storage Options”.

Install GlusterFS

On each Enterprise ColumnStore node, install GlusterFS.

Install on CentOS / RHEL 8 (YUM):

```
$ sudo yum install --enablerepo=PowerTools glusterfs-server
```

Install on CentOS / RHEL 7 (YUM):

```
$ sudo yum install centos-release-gluster
$ sudo yum install glusterfs-server
```

Install on Debian (APT):

```
$ wget -O - https://download.gluster.org/pub/gluster/glusterfs/LATEST/rsa.pub | apt-key add -

$ DEBID=$(grep 'VERSION_ID=' /etc/os-release | cut -d '=' -f 2 | tr -d '"')
$ DEBVER=$(grep 'VERSION=' /etc/os-release | grep -Eo '[a-z]+')
$ DEBARCH=$(dpkg --print-architecture)
$ echo deb https://download.gluster.org/pub/gluster/glusterfs/LATEST/Debian/${DEBID}/${DEBARCH}/apt
${DEBVER} main > /etc/apt/sources.list.d/gluster.list
```

```
$ sudo apt update
```

```
$ sudo apt install glusterfs-server
```

Install on Ubuntu (APT):

```
$ sudo apt update
```

```
$ sudo apt install glusterfs-server
```

Start the GlusterFS Daemon

Start the GlusterFS daemon:

```
$ sudo systemctl start glusterd
```

```
$ sudo systemctl enable glusterd
```

Probe the GlusterFS Peers

Before you can create a volume with GlusterFS, you must probe each node from a peer node.

1. On the primary node, probe all of the other cluster nodes:

```
$ sudo gluster peer probe mcs2
```

```
$ sudo gluster peer probe mcs3
```

2. On one of the replica nodes, probe the primary node to confirm that it is connected:

```
$ sudo gluster peer probe mcs1

peer probe: Host mcs1 port 24007 already in peer list
```

3. On the primary node, check the peer status:

```
$ sudo gluster peer status
```

Number of Peers: 2

```
Hostname: mcs2
Uuid: 3c8a5c79-22de-45df-9034-8ae624b7b23e
State: Peer in Cluster (Connected)

Hostname: mcs3
Uuid: 862af7b2-bb5e-4b1c-8311-630fa32ed451
State: Peer in Cluster (Connected)
```

Configure and Mount GlusterFS Volumes

Create the GlusterFS volumes for MariaDB Enterprise ColumnStore. Each volume must have the same number of replicas as the number of Enterprise ColumnStore nodes.

1. On each Enterprise ColumnStore node, create the directory for each brick in the /brick directory:

```
$ sudo mkdir -p /brick/storagemanager
```

2. On the primary node, create the GlusterFS volumes:

```
$ sudo gluster volume create storagemanager \
      replica 3 \
      mcs1:/brick/storagemanager \
      mcs2:/brick/storagemanager \
      mcs3:/brick/storagemanager \
      force
```

3. On the primary node, start the volume:

```
$ sudo gluster volume start storagemanager
```

4. On each Enterprise ColumnStore node, create mount points for the volumes:

```
$ sudo mkdir -p /var/lib/columnstore/storagemanager
```

5. On each Enterprise ColumnStore node, add the mount points to /etc/fstab:

```
127.0.0.1:storagemanager /var/lib/columnstore/storagemanager glusterfs defaults,_netdev 0 0
```

6. On each Enterprise ColumnStore node, mount the volumes:

```
$ sudo mount -a
```

Configure Network File System (NFS)

NFS is a distributed file system. NFS is available in most Linux distributions. If NFS is used for an Enterprise ColumnStore deployment, the storage must be mounted with the sync option to ensure that each node flushes its changes immediately.

For on-premises deployments:

- NFS is the recommended option for the Storage Manager directory.
- Any S3-compatible storage is the recommended option for data.

Consult the documentation for your NFS implementation for details on how to configure NFS.

Next Step

Navigation in the procedure “Deploy ColumnStore Shared Local Storage Topology”.

This page was step 2 of 9.

Next: Step 3: Install MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 3: Install MariaDB Enterprise Server

Overview

This page details step 3 of the 9-step procedure "Deploy ColumnStore Shared Local Storage Topology".

This step installs MariaDB Enterprise Server, MariaDB Enterprise ColumnStore 23.10, CMAPI, and dependencies.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Download Token

MariaDB Corporation provides package repositories for CentOS / RHEL (YUM) and Debian / Ubuntu (APT). A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to https://customers.mariadb.com/downloads/token/
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On each Enterprise ColumnStore node, install the prerequisites for downloading the software from the Web. Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

2. On each Enterprise ColumnStore node, configure package repositories and specify Enterprise Server:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-maxscale \
      --skip-tools \
      --mariadb-server-version="11.4"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install Enterprise Server and Enterprise ColumnStore

1. On each Enterprise ColumnStore node, install additional dependencies:

Install on CentOS and RHEL (YUM):

```
$ sudo yum install jemalloc jq curl
```

Install on Debian 9 and Ubuntu 18.04 (APT)

```
$ sudo apt install libjemalloc1 jq curl
```

Install on Debian 10 and Ubuntu 20.04 (APT):

```
$ sudo apt install libjemalloc2 jq curl
```

2. On each Enterprise ColumnStore node, install MariaDB Enterprise Server and MariaDB Enterprise ColumnStore:

Install on CentOS / RHEL (YUM):

```
$ sudo yum install MariaDB-server \
   MariaDB-backup \
   MariaDB-shared \
   MariaDB-client \
   MariaDB-columnstore-engine \
   MariaDB-columnstore-cmapi
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install mariadb-server \
   mariadb-backup \
   libmariadb3 \
   mariadb-client \
   mariadb-plugin-columnstore \
   mariadb-columnstore-cmapi
```

Next Step

Navigation in the procedure “Deploy ColumnStore Shared Local Storage Topology”.

This page was step 3 of 9.

Next: Step 4: Start and Configure MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 4: Start and Configure MariaDB Enterprise Server

Overview

This page details step 4 of the 9-step procedure “Deploy ColumnStore Shared Local Storage Topology”.

This step starts and configures MariaDB Enterprise Server, and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Stop the Enterprise ColumnStore Services

The installation process might have started some of the ColumnStore services. The services should be stopped prior to making configuration changes.

1. On each Enterprise ColumnStore node, stop the MariaDB Enterprise Server service:

```
$ sudo systemctl stop mariadb
```

2. On each Enterprise ColumnStore node, stop the MariaDB Enterprise ColumnStore service:

```
$ sudo systemctl stop mariadb-columnstore
```

3. On each Enterprise ColumnStore node, stop the CMAPI service:

```
$ sudo systemctl stop mariadb-columnstore-cmapi
```

Configure Enterprise ColumnStore

On each Enterprise ColumnStore node, configure Enterprise Server.

Connector	MariaDB Connector/R2DBC
character_set_server	Set this system variable to utf8
collation_server	Set this system variable to utf8_general_ci
loose-columnstore_use_import_for_batchinsert	Set this system variable to ALWAYS to always use cpimport for LOAD DATA INFILE and INSERT…SELECT statements.
gtid_strict_mode	Set this system variable to ON.
log_bin	Set this option to the file you want to use for the Binary Log. Setting this option enables binary logging.
log_bin_index	Set this option to the file you want to use to track binlog filenames.
log_slave_updates	Set this system variable to ON.
relay_log	Set this option to the file you want to use for the Relay Logs. Setting this option enables relay logging.
relay_log_index	Set this option to the file you want to use to index Relay Log filenames.
server_id	Sets the numeric Server ID for this MariaDB Enterprise Server. The value set on this option must be unique to each node.

Mandatory system variables and options for ColumnStore Object Storage include:

Example Configuration

```
[mariadb]
bind_address                            = 0.0.0.0
log_error                               = mariadbd.err
character_set_server                    = utf8
collation_server                        = utf8_general_ci
log_bin                                 = mariadb-bin
log_bin_index                           = mariadb-bin.index
relay_log                               = mariadb-relay
relay_log_index                         = mariadb-relay.index
log_slave_updates                       = ON
gtid_strict_mode                        = ON

# This must be unique on each Enterprise ColumnStore node
server_id                               = 1
```

Start the Enterprise ColumnStore Services

1. On each Enterprise ColumnStore node, start and enable the MariaDB Enterprise Server service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb
```

```
$ sudo systemctl enable mariadb
```

2. On each Enterprise ColumnStore node, stop the MariaDB Enterprise ColumnStore service:

```
$ sudo systemctl stop mariadb-columnstore
```

After the CMAPI service is installed in the next step, CMAPI will start the Enterprise ColumnStore service as-needed on each node. CMAPI disables the Enterprise ColumnStore service to prevent systemd from automatically starting Enterprise ColumnStore upon reboot.

3. On each Enterprise ColumnStore node, start and enable the CMAPI service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb-columnstore-cmapi
```

```
$ sudo systemctl enable mariadb-columnstore-cmapi
```

For additional information, see “Start and Stop Services”.

Create User Accounts

The ColumnStore Object Storage topology requires several user accounts. Each user account should be created on the primary server, so that it is replicated to the replica servers.

Create the Utility User

Enterprise ColumnStore requires a mandatory utility user account to perform cross-engine joins and similar operations.

1. On the primary server, create the user account with the CREATE USER statement:

```
CREATE USER 'util_user'@'127.0.0.1'
IDENTIFIED BY 'util_user_passwd';
```

2. On the primary server, grant the user account SELECT privileges on all databases with the GRANT statement:

```
GRANT SELECT, PROCESS ON *.*
TO 'util_user'@'127.0.0.1';
```

3. On each Enterprise ColumnStore node, configure the ColumnStore utility user:

```
$ sudo mcsSetConfig CrossEngineSupport Host 127.0.0.1
```

```
$ sudo mcsSetConfig CrossEngineSupport Port 3306
```

```
$ sudo mcsSetConfig CrossEngineSupport User util_user
```

4. On each Enterprise ColumnStore node, set the password:

```
$ sudo mcsSetConfig CrossEngineSupport Password util_user_passwd
```

For details about how to encrypt the password, see “Credentials Management for MariaDB Enterprise ColumnStore”.

Passwords should meet your organization’s password policies. If your MariaDB Enterprise Server instance has a password validation plugin installed, then the password should also meet the configured requirements.

Create the Replication User

ColumnStore Object Storage uses MariaDB Replication to replicate writes between the primary and replica servers. As MaxScale can promote a replica server to become a new primary in the event of node failure, all nodes must have a replication user.

The action is performed **on the primary server**.

Create the replication user and grant it the required privileges:

1. Use the CREATE USER statement to create replication user.

```
CREATE USER 'repl'@'192.0.2.%' IDENTIFIED BY 'repl_passwd';
```

Replace the referenced IP address with the relevant address for your environment.

Ensure that the user account can connect to the primary server from each replica.

2. Grant the user account the required privileges with the GRANT statement.

```
GRANT REPLICA MONITOR,
   REPLICATION REPLICA,
   REPLICATION REPLICA ADMIN,
   REPLICATION MASTER ADMIN
ON *.* TO 'repl'@'192.0.2.%';
```

Create MaxScale User

ColumnStore Object Storage 23.10 uses MariaDB MaxScale 22.08 to load balance between the nodes.

This action is performed on the primary server.

1. Use the CREATE USER statement to create the MaxScale user:

```
CREATE USER 'mxs'@'192.0.2.%'
IDENTIFIED BY 'mxs_passwd';
```

Replace the referenced IP address with the relevant address for your environment.

Ensure that the user account can connect from the IP address of the MaxScale instance.

2. Use the GRANT statement to grant the privileges required by the router:

```
GRANT SHOW DATABASES ON *.* TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.columns_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.db TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.procs_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.proxies_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.roles_mapping TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.tables_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.user TO 'mxs'@'192.0.2.%';
```

3. Use the GRANT statement to grant privileges required by the MariaDB Monitor.

```
GRANT BINLOG ADMIN,
   READ_ONLY ADMIN,
   RELOAD,
   REPLICA MONITOR,
   REPLICATION MASTER ADMIN,
   REPLICATION REPLICA ADMIN,
   REPLICATION REPLICA,
   SHOW DATABASES,
   SELECT
ON *.* TO 'mxs'@'192.0.2.%';
```

Configure MariaDB Replication

On each replica server, configure MariaDB Replication:

1. Use the CHANGE MASTER TO statement to configure the connection to the primary server:

```
CHANGE MASTER TO
   MASTER_HOST='192.0.2.1',
   MASTER_USER='repl',
   MASTER_PASSWORD='repl_passwd',
   MASTER_USE_GTID=slave_pos;
```

2. Start replication using the START REPLICA statement:

```
START REPLICA;
```

3. Confirm that replication is working using the SHOW REPLICA STATUS statement:

```
SHOW REPLICA STATUS;
```

Ensure that the replica server cannot accept local writes by setting the read_only system variable to ON using the SET GLOBAL statement:

```
SET GLOBAL read_only=ON;
```

Initiate the Primary Server with CMAPI

Initiate the primary server using CMAPI.

1. Create an API key for the cluster.
   This API key should be stored securely and kept confidential, because it can be used to add cluster nodes to the multi-node Enterprise ColumnStore deployment.

For example, to create a random 256-bit API key using openssl rand:

```
$ openssl rand -hex 32

93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd
```

This document will use the following API key in further examples, but users should create their own:

2. Use CMAPI to add the primary server to the cluster and set the API key.
   The new API key needs to be provided as part of the X-API-key HTML header.

For example, if the primary server's host name is mcs1 and its IP address is 192.0.2.1, use the following node command:

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/node \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   --data '{"timeout":120, "node": "192.0.2.1"}' \
   | jq .
```

```
{
  "timestamp": "2020-10-28 00:39:14.672142",
  "node_id": "192.0.2.1"
}
```

3. Use CMAPI to check the status of the cluster node:

```
$ curl -k -s https://mcs1:8640/cmapi/0.4.0/cluster/status \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   | jq .
```

```
{
  "timestamp": "2020-12-15 00:40:34.353574",
  "192.0.2.1": {
    "timestamp": "2020-12-15 00:40:34.362374",
    "uptime": 11467,
    "dbrm_mode": "master",
    "cluster_mode": "readwrite",
    "dbroots": [
      "1"
    ],
    "module_id": 1,
    "services": [
      {
        "name": "workernode",
        "pid": 19202
      },
      {
        "name": "controllernode",
        "pid": 19232
      },
      {
        "name": "PrimProc",
        "pid": 19254
      },
      {
        "name": "ExeMgr",
        "pid": 19292
      },
      {
        "name": "WriteEngine",
        "pid": 19316
      },
      {
        "name": "DMLProc",
        "pid": 19332
      },
      {
        "name": "DDLProc",
        "pid": 19366
      }
    ]
  }
```

Add Replica Servers with CMAPI

Add the replica servers with CMAPI:

1. For each replica server, use CMAPI to add the replica server to the cluster.
   The previously set API key needs to be provided as part of the X-API-key HTML header.

For example, if the primary server's host name is mcs1 and the replica server's IP address is 192.0.2.2, use the following node command:

```
$ curl -k -s -X PUT https://mcs1:8640/cmapi/0.4.0/cluster/node \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   --data '{"timeout":120, "node": "192.0.2.2"}' \
   | jq .
```

```
{
  "timestamp": "2020-10-28 00:42:42.796050",
  "node_id": "192.0.2.2"
}
```

2. After all replica servers have been added, use CMAPI to confirm that all cluster nodes have been successfully added:

```
$ curl -k -s https://mcs1:8640/cmapi/0.4.0/cluster/status \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   | jq .
```

```
{
  "timestamp": "2020-12-15 00:40:34.353574",
  "192.0.2.1": {
    "timestamp": "2020-12-15 00:40:34.362374",
    "uptime": 11467,
    "dbrm_mode": "master",
    "cluster_mode": "readwrite",
    "dbroots": [
      "1"
    ],
    "module_id": 1,
    "services": [
      {
        "name": "workernode",
        "pid": 19202
      },
      {
        "name": "controllernode",
        "pid": 19232
      },
      {
        "name": "PrimProc",
        "pid": 19254
      },
      {
        "name": "ExeMgr",
        "pid": 19292
      },
      {
        "name": "WriteEngine",
        "pid": 19316
      },
      {
        "name": "DMLProc",
        "pid": 19332
      },
      {
        "name": "DDLProc",
        "pid": 19366
      }
    ]
  },
  "192.0.2.2": {
    "timestamp": "2020-12-15 00:40:34.428554",
    "uptime": 11437,
    "dbrm_mode": "slave",
    "cluster_mode": "readonly",
    "dbroots": [
      "2"
    ],
    "module_id": 2,
    "services": [
      {
        "name": "workernode",
        "pid": 17789
      },
      {
        "name": "PrimProc",
        "pid": 17813
      },
      {
        "name": "ExeMgr",
        "pid": 17854
      },
      {
        "name": "WriteEngine",
        "pid": 17877
      }
```

```
    ]
  },
  "192.0.2.3": {
    "timestamp": "2020-12-15 00:40:34.428554",
    "uptime": 11437,
    "dbrm_mode": "slave",
    "cluster_mode": "readonly",
    "dbroots": [
      "2"
    ],
    "module_id": 2,
    "services": [
      {
        "name": "workernode",
        "pid": 17789
      },
      {
        "name": "PrimProc",
        "pid": 17813
      },
      {
        "name": "ExeMgr",
        "pid": 17854
      },
      {
        "name": "WriteEngine",
        "pid": 17877
      }
    ]
  },
  "num_nodes": 3
}
```

Configure Linux Security Modules (LSM)

The specific steps to configure the security module depend on the operating system.

Configure SELinux (CentOS, RHEL)

Configure SELinux for Enterprise ColumnStore:

1. To configure SELinux, you have to install the packages required for audit2allow.
   On CentOS 7 and RHEL 7, install the following:

```
$ sudo yum install policycoreutils policycoreutils-python
```

On RHEL 8, install the following:

```
$ sudo yum install policycoreutils python3-policycoreutils policycoreutils-python-utils
```

2. Allow the system to run under load for a while to generate SELinux audit events.
3. After the system has taken some load, generate an SELinux policy from the audit events using audit2allow:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local
```

If no audit events were found, this will print the following:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local

Nothing to do
```

4. If audit events were found, the new SELinux policy can be loaded using semodule:

```
$ sudo semodule -i mariadb_local.pp
```

5. Set SELinux to enforcing mode:

```
$ sudo setenforce enforcing
```

6. Set SELinux to enforcing mode by setting SELINUX=enforcing in /etc/selinux/config.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

7. Confirm that SELinux is in enforcing mode:

```
$ sudo getenforce
```

```
Enforcing
```

Configure AppArmor (Ubuntu)

For information on how to create a profile, see How to create an AppArmor Profile on Ubuntu.com.

Configure Firewalls

The specific steps to configure the firewall service depend on the platform.

Configure firewalld (CentOS, RHEL)

Configure firewalld for Enterprise Cluster on CentOS and RHEL:

1. Check if the firewalld service is running:

```
$ sudo systemctl status firewalld
```

2. If the firewalld service was stopped to perform the installation, start it now:

For example, if your cluster nodes are in the 192.0.2.0/24 subnet:

```
$ sudo systemctl start firewalld
```

3. Open up the relevant ports using firewall-cmd:

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="3306" protocol="tcp"
   accept'
```

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="8600-8630" protocol="tcp"
   accept'
```

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="8640" protocol="tcp"
   accept'
```

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="8700" protocol="tcp"
   accept'
```

```
$ sudo firewall-cmd --permanent --add-rich-rule='
   rule family="ipv4"
   source address="192.0.2.0/24"
   destination address="192.0.2.0/24"
   port port="8800" protocol="tcp"
   accept'
```

4. Reload the runtime configuration:

```
$ sudo firewall-cmd --reload
```

Configure UFW (Ubuntu)

Configure UFW for Enterprise ColumnStore on Ubuntu:

1. Check if the UFW service is running:

```
$ sudo ufw status verbose
```

2. If the UFW service was stopped to perform the installation, start it now:

```
$ sudo ufw enable
```

3. Open up the relevant ports using ufw.

For example, if your cluster nodes are in the 192.0.2.0/24 subnet in the range 192.0.2.1 - 192.0.2.3:

```
$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 3306 proto tcp

$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 8600:8630 proto tcp

$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 8640 proto tcp

$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 8700 proto tcp

$ sudo ufw allow from 192.0.2.0/24 to 192.0.2.3 port 8800 proto tcp
```

4. Reload the runtime configuration:

```
$ sudo ufw reload
```

Next Step

Navigation in the procedure “Deploy ColumnStore Shared Local Storage Topology”.

This page was step 4 of 9.

Next: Step 5: Test MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 5: Test MariaDB Enterprise Server

Overview

This page details step 5 of the 9-step procedure “Deploy ColumnStore Shared Local Storage Topology”.

This step tests MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Test Enterprise Server Service

Use Systemd to test whether the MariaDB Enterprise Server service is running.

This action is performed **on each Enterprise ColumnStore node**.

Check if the MariaDB Enterprise Server service is running by executing the following:

```
$ systemctl status mariadb
```

If the service is not running on any node, start the service by executing the following on that node:

```
$ sudo systemctl start mariadb
```

Test Local Client Connections

Use MariaDB Client to test the local connection to the Enterprise Server node.

This action is performed **on each Enterprise ColumnStore node**:

```
$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The `sudo` command is used here to connect to the Enterprise Server node using the root@localhost user account, which authenticates using the unix_socket authentication plugin. Other user accounts can be used by specifying the –user and –password command-line options.

Test ColumnStore Storage Engine Plugin

Query the information_schema.PLUGINS table to confirm that the ColumnStore storage engine is loaded.

This action is performed on each Enterprise ColumnStore node.

Execute the following query:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS
FROM information_schema.PLUGINS
WHERE PLUGIN_LIBRARY LIKE 'ha_columnstore%';

+--------------------+---------------+
| PLUGIN_NAME        | PLUGIN_STATUS |
+--------------------+---------------+
| Columnstore        | ACTIVE        |
| COLUMNSTORE_COLUMNS | ACTIVE        |
| COLUMNSTORE_TABLES  | ACTIVE        |
| COLUMNSTORE_FILES   | ACTIVE        |
| COLUMNSTORE_EXTENTS | ACTIVE        |
+--------------------+---------------+
```

The `PLUGIN_STATUS` column for each ColumnStore-related plugin should contain ACTIVE.

Test CMAPI Service

Use Systemd to test whether the CMAPI service is running.

This action is performed on each Enterprise ColumnStore node.

Check if the CMAPI service is running by executing the following:

```
$ systemctl status mariadb-columnstore-cmapi
```

If the service is not running on any node, start the service by executing the following on that node:

```
$ sudo systemctl start mariadb-columnstore-cmapi
```

Test ColumnStore Status

Use CMAPI to request the ColumnStore status. The API key needs to be provided as part of the `X-API-key` HTML header.

This action is performed with the CMAPI service on the primary server.

Check the ColumnStore status using curl by executing the following:

```
$ curl -k -s https://mcs1:8640/cmapi/0.4.0/cluster/status \
   --header 'Content-Type:application/json' \
   --header 'x-api-key:93816fa66cc2d8c224e62275bd4f248234dd4947b68d4af2b29671dd7d5532dd' \
   | jq .
```

```
{
  "timestamp": "2020-12-15 00:40:34.353574",
  "192.0.2.1": {
    "timestamp": "2020-12-15 00:40:34.362374",
    "uptime": 11467,
    "dbrm_mode": "master",
    "cluster_mode": "readwrite",
    "dbroots": [
      "1"
    ],
    "module_id": 1,
    "services": [
      {
        "name": "workernode",
        "pid": 19202
      },
      {
        "name": "controllernode",
        "pid": 19232
      },
      {
        "name": "PrimProc",
        "pid": 19254
      },
      {
        "name": "ExeMgr",
        "pid": 19292
      },
      {
        "name": "WriteEngine",
        "pid": 19316
      },
      {
        "name": "DMLProc",
        "pid": 19332
      },
      {
        "name": "DDLProc",
        "pid": 19366
      }
    ]
  },
  "192.0.2.2": {
    "timestamp": "2020-12-15 00:40:34.428554",
    "uptime": 11437,
    "dbrm_mode": "slave",
    "cluster_mode": "readonly",
    "dbroots": [
      "2"
    ],
    "module_id": 2,
    "services": [
      {
        "name": "workernode",
        "pid": 17789
      },
      {
        "name": "PrimProc",
        "pid": 17813
      },
      {
        "name": "ExeMgr",
        "pid": 17854
      },
      {
        "name": "WriteEngine",
        "pid": 17877
      }
```

```
    ]
  },
  "192.0.2.3": {
    "timestamp": "2020-12-15 00:40:34.428554",
    "uptime": 11437,
    "dbrm_mode": "slave",
    "cluster_mode": "readonly",
    "dbroots": [
      "2"
    ],
    "module_id": 2,
    "services": [
      {
        "name": "workernode",
        "pid": 17789
      },
      {
        "name": "PrimProc",
        "pid": 17813
      },
      {
        "name": "ExeMgr",
        "pid": 17854
      },
      {
        "name": "WriteEngine",
        "pid": 17877
      }
    ]
  },
  "num_nodes": 3
}
```

Test DDL

Use MariaDB Client to test DDL.

1. On the primary server, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

2. Create a test database and ColumnStore table:

```
CREATE DATABASE IF NOT EXISTS test;

CREATE TABLE IF NOT EXISTS test.contacts (
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   email VARCHAR(100)
) ENGINE = ColumnStore;
```

3. On each replica server, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

4. Confirm that the database and table exist:

```
SHOW CREATE TABLE test.contacts\G;
```

If the database or table do not exist on any node, then check the replication configuration.

Test DML

Use MariaDB Client to test DML.

1. On the primary server, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

2. Insert sample data into the table created in the DDL test:

```
INSERT INTO test.contacts (first_name, last_name, email)
   VALUES
   ("Kai", "Devi", "kai.devi@example.com"),
   ("Lee", "Wang", "lee.wang@example.com");
```

3. On each replica server, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

4. Execute a SELECT query to retrieve the data:

```
SELECT * FROM test.contacts;

+------------+-----------+----------------------+
| first_name | last_name | email                |
+------------+-----------+----------------------+
| Kai        | Devi      | kai.devi@example.com |
| Lee        | Wang      | lee.wang@example.com |
+------------+-----------+----------------------+
```

If the data is not returned on any node, check the ColumnStore status and the storage configuration.

Next Step

Navigation in the procedure “Deploy ColumnStore Shared Local Storage Topology”.

This page was step 5 of 9.

Next: Step 6: Install MariaDB MaxScale.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 6: Install MariaDB MaxScale

Overview

This page details step 6 of the 9-step procedure “Deploy ColumnStore Shared Local Storage Topology”.

This step installs MariaDB MaxScale 22.08.

ColumnStore Object Storage requires 1 or more MaxScale nodes.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Customer Download Token

MariaDB Corporation provides package repositories for `CentOS / RHEL (YUM) and Debian / Ubuntu (APT)`. A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On the MaxScale node, install the prerequisites for downloading the software from the Web.
   Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

2. On the MaxScale node, configure package repositories and specify MariaDB MaxScale 22.08:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup

$ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -

$ chmod +x mariadb_es_repo_setup

$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-server \
      --skip-tools \
      --mariadb-maxscale-version="22.08"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install MaxScale

On the MaxScale node, install MariaDB MaxScale.

Install on CentOS / RHEL (YUM):

```
$ sudo yum install maxscale
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install maxscale
```

Next Step

Navigation in the procedure “Deploy ColumnStore Shared Local Storage Topology”.

This page was step 6 of 9.

Next: Step 7: Start and Configure MariaDB MaxScale.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 7: Start and Configure MariaDB MaxScale

Overview

This page details step 7 of the 9-step procedure “Deploy ColumnStore Shared Local Storage Topology”.

This step starts and configures MariaDB MaxScale 22.08.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Replace the Default Configuration File

MariaDB MaxScale installations include a configuration file with some example objects. This configuration file should be replaced.

On the MaxScale node, replace the default `/etc/maxscale.cnf` with the following configuration:

```
[maxscale]
threads          = auto
admin_host       = 0.0.0.0
admin_secure_gui = false
```

For additional information, see “Global Parameters”.

Restart MaxScale

On the MaxScale node, restart the MaxScale service to ensure that MaxScale picks up the new configuration:

```
$ sudo systemctl restart maxscale
```

For additional information, see “Start and Stop Services”.

Configure Server Objects

On the MaxScale node, use maxctrl create to create a server object for each Enterprise ColumnStore node:

```
$ maxctrl create server mcs1 192.0.2.101
```

```
$ maxctrl create server mcs2 192.0.2.102
```

```
$ maxctrl create server mcs3 192.0.2.103
```

Configure MariaDB Monitor

MaxScale uses monitors to retrieve additional information from the servers. This information is used by other services in filtering and routing connections based on the current state of the node. For MariaDB Enterprise ColumnStore, use the MariaDB Monitor (mariadbmon).

On the MaxScale node, use maxctrl create monitor to create a MariaDB Monitor:

```
$ maxctrl create monitor columnstore_monitor mariadbmon \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     replication_user=repl \
     replication_password='REPLICATION_USER_PASSWORD' \
     --servers mcs1 mcs2 mcs3
```

In this example:

- `columnstore_monitor` is an arbitrary name that is used to identify the new monitor.
- `mariadbmon` is the name of the module that implements the MariaDB Monitor.
- `user=MAXSCALE_USER` sets the user parameter to the database user account that MaxScale uses to monitor the ColumnStore nodes.
- `password='MAXSCALE_USER_PASSWORD'` sets the password parameter to the password used by the database user account that MaxScale uses to monitor the ColumnStore nodes.

- `replication_user=REPLICATION_USER` sets the replication_user parameter to the database user account that MaxScale uses to setup replication.
- `replication_password='REPLICATION_USER_PASSWORD'` sets the replication_password parameter to the password used by the database user account that MaxScale uses to setup replication.
- `--servers` sets the servers parameter to the set of nodes that MaxScale should monitor. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by mariadbmon in MaxScale 22.08 can also be specified.

Choose a MaxScale Router

Routers control how MaxScale balances the load between Enterprise ColumnStore nodes. Each router uses a different approach to routing queries. Consider the specific use case of your application and database load and select the router that best suits your needs.

Router	Configuration Procedure	Description
Read Connection (readconnroute)	Configure Read Connection Router	
Read/Write Split (readwritesplit)	Configure Read/Write Split	

Configure Read Connection Router

Use MaxScale Read Connection Router (readconnroute) to route connections to replica servers for a read-only pool.

On the MaxScale node, use maxctrl create service to create a router:

```
$ maxctrl create service connection_router_service readconnroute \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     router_options=slave \
     --servers mcs1 mcs2 mcs3
```

In this example:

- `connection_router_service` is an arbitrary name that is used to identify the new service.
- readconnroute is the name of the module that implements the Read Connection Router.
- `user=MAXSCALE_USER` sets the user parameter to the database user account that MaxScale uses to connect to the ColumnStore nodes.
- `password=MAXSCALE_USER_PASSWORD` sets the password parameter to the password used by the database user account that MaxScale uses to connect to the ColumnStore nodes.
- `router_options=slave` sets the `router_options` parameter to slave, so that MaxScale only routes connections to the replica nodes.
- `--servers` sets the servers parameter to the set of nodes to which MaxScale should route connections. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by `readconnroute` in MaxScale 22.08 can also be specified.

Configure Listener for the Read Connection Router

These instructions reference TCP port 3308. You can use a different TCP port. The TCP port used must not be bound by any other listener.

On the MaxScale node, use the maxctrl create listener command to configure MaxScale to use a listener for the Read Connection Router (readconnroute):

```
$ maxctrl create listener connection_router_service connection_router_listener 3308 \
     protocol=MariaDBClient
```

In this example:

- `connection_router_service` is the name of the `readconnroute` service that was previously created.
- `connection_router_listener` is an arbitrary name that is used to identify the new listener.
- 3308 is the TCP port.
- `protocol=MariaDBClient` sets the protocol parameter.
- Other Module Parameters supported by listeners in MaxScale 22.08 can also be specified.

Configure Read/Write Split Router for Queries

MaxScale Read/Write Split Router (readwritesplit) performs query-based load balancing. The router routes write queries to the primary and read queries to the replicas.

On the MaxScale node, use the maxctrl create service command to configure MaxScale to use the Read/Write Split Router (readwritesplit):

```
$ maxctrl create service query_router_service readwritesplit  \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     --servers mcs1 mcs2 mcs3
```

In this example:

- `query_router_service` is an arbitrary name that is used to identify the new service.
- `readwritesplit` is the name of the module that implements the Read/Write Split Router.
- `user=MAXSCALE_USER` sets the user parameter to the database user account that MaxScale uses to connect to the ColumnStore nodes.
- `password=MAXSCALE_USER_PASSWORD` sets the password parameter to the password used by the database user account that MaxScale uses to connect to the ColumnStore nodes.
- `--servers` sets the servers parameter to the set of nodes to which MaxScale should route queries. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by `readwritesplit` in MaxScale 22.08 can also be specified.

Configure a Listener for the Read/Write Split Router

These instructions reference TCP port 3307. You can use a different TCP port. The TCP port used must not be bound by any other listener.

On the MaxScale node, use the maxctrl create listener command to configure MaxScale to use a listener for the Read/Write Split Router (readwritesplit):

```
$ maxctrl create listener query_router_service query_router_listener 3307 \
     protocol=MariaDBClient
```

In this example:

- `query_router_service` is the name of the readwritesplit service that was previously created.
- `query_router_listener` is an arbitrary name that is used to identify the new listener.
- 3307 is the TCP port.
- `protocol=MariaDBClient` sets the protocol parameter.
- Other Module Parameters supported by listeners in MaxScale 22.08 can also be specified.

Start Services

To start the services and monitors, on the MaxScale node use maxctrl start services:

```
$ maxctrl start services
```

Next Step

Navigation in the procedure “Deploy ColumnStore Shared Local Storage Topology”.

This page was step 7 of 9.

Next: Next: Step 8: Test MariaDB MaxScale.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 8: Test MariaDB MaxScale

Overview

This page details step 8 of the 9-step procedure “Deploy ColumnStore Shared Local Storage Topology”.

This step tests MariaDB MaxScale 22.08.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Check Global Configuration

Use maxctrl show maxscale command to view the global MaxScale configuration.

This action is performed **on the MaxScale node**:

```
$ maxctrl show maxscale
```

```
┌──────────────┬──────────────────────────────────────────────────────────────┐
│ Version      │ 22.08.15                                                     │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Commit       │ 3761fa7a52046bc58faad8b5a139116f9e33364c                     │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Started At   │ Thu, 05 Aug 2021 20:21:20 GMT                                │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Activated At │ Thu, 05 Aug 2021 20:21:20 GMT                                │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Uptime       │ 868                                                          │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Config Sync  │ null                                                         │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Parameters   │ {                                                            │
│              │     "admin_auth": true,                                      │
│              │     "admin_enabled": true,                                   │
│              │     "admin_gui": true,                                       │
│              │     "admin_host": "0.0.0.0",                                 │
│              │     "admin_log_auth_failures": true,                         │
│              │     "admin_pam_readonly_service": null,                      │
│              │     "admin_pam_readwrite_service": null,                     │
│              │     "admin_port": 8989,                                      │
│              │     "admin_secure_gui": false,                               │
│              │     "admin_ssl_ca_cert": null,                               │
│              │     "admin_ssl_cert": null,                                  │
│              │     "admin_ssl_key": null,                                   │
│              │     "admin_ssl_version": "MAX",                              │
│              │     "auth_connect_timeout": "10000ms",                       │
│              │     "auth_read_timeout": "10000ms",                          │
│              │     "auth_write_timeout": "10000ms",                         │
│              │     "cachedir": "/var/cache/maxscale",                       │
│              │     "config_sync_cluster": null,                             │
│              │     "config_sync_interval": "5000ms",                        │
│              │     "config_sync_password": "******",                        │
│              │     "config_sync_timeout": "10000ms",                        │
│              │     "config_sync_user": null,                                │
│              │     "connector_plugindir": "/usr/lib64/mysql/plugin",        │
│              │     "datadir": "/var/lib/maxscale",                          │
│              │     "debug": null,                                           │
│              │     "dump_last_statements": "never",                         │
│              │     "execdir": "/usr/bin",                                   │
│              │     "language": "/var/lib/maxscale",                         │
│              │     "libdir": "/usr/lib64/maxscale",                         │
│              │     "load_persisted_configs": true,                          │
│              │     "local_address": null,                                   │
│              │     "log_debug": false,                                      │
│              │     "log_info": false,                                       │
│              │     "log_notice": true,                                      │
│              │     "log_throttling": {                                      │
│              │         "count": 10,                                         │
│              │         "suppress": 10000,                                   │
│              │         "window": 1000                                       │
│              │     },                                                       │
│              │     "log_warn_super_user": false,                            │
│              │     "log_warning": true,                                     │
│              │     "logdir": "/var/log/maxscale",                           │
│              │     "max_auth_errors_until_block": 10,                       │
│              │     "maxlog": true,                                          │
│              │     "module_configdir": "/etc/maxscale.modules.d",           │
│              │     "ms_timestamp": false,                                   │
│              │     "passive": false,                                        │
│              │     "persistdir": "/var/lib/maxscale/maxscale.cnf.d",        │
│              │     "piddir": "/var/run/maxscale",                           │
│              │     "query_classifier": "qc_sqlite",                         │
│              │     "query_classifier_args": null,                           │
│              │     "query_classifier_cache_size": 289073971,                │
│              │     "query_retries": 1,                                      │
│              │     "query_retry_timeout": "5000ms",                         │
```

```
|               |     "rebalance_period": "0ms",                          |
|               |     "rebalance_threshold": 20,                          |
|               |     "rebalance_window": 10,                             |
|               |     "retain_last_statements": 0,                        |
|               |     "session_trace": 0,                                 |
|               |     "skip_permission_checks": false,                    |
|               |     "sql_mode": "default",                              |
|               |     "syslog": true,                                     |
|               |     "threads": 1,                                       |
|               |     "users_refresh_interval": "0ms",                    |
|               |     "users_refresh_time": "30000ms",                    |
|               |     "writeq_high_water": 16777216,                      |
|               |     "writeq_low_water": 8192                            |
|               | }                                                       |
└───────────────┴─────────────────────────────────────────────────────────┘
```

Output should align to the global MaxScale configuration in the new configuration file you created.

Check Server Configuration

Use the maxctrl list servers and maxctrl show server commands to view the configured server objects.

This action is performed **on the MaxScale node**:

1. Obtain the full list of servers objects:

```
$ maxctrl list servers
```

```
┌────────┬───────────────┬──────┬─────────────┬─────────────────┬──────────┐
| Server | Address       | Port | Connections | State           | GTID     |
├────────┼───────────────┼──────┼─────────────┼─────────────────┼──────────┤
| mcs1   | 192.0.2.1     | 3306 | 1           | Master, Running | 0-1-25 |
├────────┼───────────────┼──────┼─────────────┼─────────────────┼──────────┤
| mcs2   | 192.0.2.2     | 3306 | 1           | Slave, Running  | 0-1-25 |
├────────┼───────────────┼──────┼─────────────┼─────────────────┼──────────┤
| mcs3   | 192.0.2.3     | 3306 | 1           | Slave, Running  | 0-1-25 |
└────────┴───────────────┴──────┴─────────────┴─────────────────┴──────────┘
```

For each server object, view the configuration:

```
$ maxctrl show server mcs1
```

```
┌──────────────────┬──────────────────────────────────────────────────┐
│ Server           │ mcs1                                             │
├──────────────────┼──────────────────────────────────────────────────┤
│ Address          │ 192.0.2.1                                        │
├──────────────────┼──────────────────────────────────────────────────┤
│ Port             │ 3306                                             │
├──────────────────┼──────────────────────────────────────────────────┤
│ State            │ Master, Running                                  │
├──────────────────┼──────────────────────────────────────────────────┤
│ Version          │ 11.4.5-3-MariaDB-enterprise-log                  │
├──────────────────┼──────────────────────────────────────────────────┤
│ Last Event       │ master_up                                        │
├──────────────────┼──────────────────────────────────────────────────┤
│ Triggered At     │ Thu, 05 Aug 2021 20:22:26 GMT                    │
├──────────────────┼──────────────────────────────────────────────────┤
│ Services         │ connection_router_service                        │
│                  │ query_router_service                             │
├──────────────────┼──────────────────────────────────────────────────┤
│ Monitors         │ columnstore_monitor                              │
├──────────────────┼──────────────────────────────────────────────────┤
│ Master ID        │ -1                                               │
├──────────────────┼──────────────────────────────────────────────────┤
│ Node ID          │ 1                                                │
├──────────────────┼──────────────────────────────────────────────────┤
│ Slave Server IDs │                                                  │
├──────────────────┼──────────────────────────────────────────────────┤
│ Current Connections │ 1                                             │
├──────────────────┼──────────────────────────────────────────────────┤
│ Total Connections │ 1                                               │
├──────────────────┼──────────────────────────────────────────────────┤
│ Max Connections  │ 1                                                │
├──────────────────┼──────────────────────────────────────────────────┤
│ Statistics       │ {                                                │
│                  │     "active_operations": 0,                      │
│                  │     "adaptive_avg_select_time": "0ns",           │
│                  │     "connection_pool_empty": 0,                  │
│                  │     "connections": 1,                            │
│                  │     "max_connections": 1,                        │
│                  │     "max_pool_size": 0,                          │
│                  │     "persistent_connections": 0,                 │
│                  │     "reused_connections": 0,                     │
│                  │     "routed_packets": 0,                         │
│                  │     "total_connections": 1                       │
│                  │ }                                                │
├──────────────────┼──────────────────────────────────────────────────┤
│ Parameters       │ {                                                │
│                  │     "address": "192.0.2.1",                      │
│                  │     "disk_space_threshold": null,                │
│                  │     "extra_port": 0,                             │
│                  │     "monitorpw": null,                           │
│                  │     "monitoruser": null,                         │
│                  │     "persistmaxtime": "0ms",                     │
│                  │     "persistpoolmax": 0,                         │
│                  │     "port": 3306,                                │
│                  │     "priority": 0,                               │
│                  │     "proxy_protocol": false,                     │
│                  │     "rank": "primary",                           │
│                  │     "socket": null,                              │
│                  │     "ssl": false,                                │
│                  │     "ssl_ca_cert": null,                         │
│                  │     "ssl_cert": null,                            │
│                  │     "ssl_cert_verify_depth": 9,                  │
│                  │     "ssl_cipher": null,                          │
│                  │     "ssl_key": null,                             │
│                  │     "ssl_verify_peer_certificate": false,        │
│                  │     "ssl_verify_peer_host": false,               │
│                  │     "ssl_version": "MAX"                         │
```

```
│                     │ }                                                  │
└─────────────────────┴────────────────────────────────────────────────────┘
```

Output should align to the Server Object configuration you performed.

Check Monitor Configuration

Use the maxctrl list monitors and maxctrl show monitor commands to view the configured monitors.

This action is performed on the MaxScale node:

1. Obtain the full list of monitors:

```
$ maxctrl list monitors
```

```
┌──────────────────────┬─────────┬──────────────────┐
│ Monitor              │ State   │ Servers          │
├──────────────────────┼─────────┼──────────────────┤
│ columnstore_monitor │ Running │ mcs1, mcs2, mcs3 │
└──────────────────────┴─────────┴──────────────────┘
```

2. For each monitor, view the monitor configuration:

```
$ maxctrl show monitor columnstore_monitor
```

```
┌──────────────────────┬──────────────────────────────────────────┐
│ Monitor              │ columnstore_monitor                      │
├──────────────────────┼──────────────────────────────────────────┤
│ Module               │ mariadbmon                               │
├──────────────────────┼──────────────────────────────────────────┤
│ State                │ Running                                  │
├──────────────────────┼──────────────────────────────────────────┤
│ Servers              │ mcs1                                     │
│                      │ mcs2                                     │
│                      │ mcs3                                     │
├──────────────────────┼──────────────────────────────────────────┤
│ Parameters           │ {                                        │
│                      │     "backend_connect_attempts": 1,       │
│                      │     "backend_connect_timeout": 3,        │
│                      │     "backend_read_timeout": 3,           │
│                      │     "backend_write_timeout": 3,          │
│                      │     "disk_space_check_interval": 0,      │
│                      │     "disk_space_threshold": null,        │
│                      │     "events": "all",                     │
│                      │     "journal_max_age": 28800,            │
│                      │     "module": "mariadbmon",              │
│                      │     "monitor_interval": 2000,            │
│                      │     "password": "******",                │
│                      │     "script": null,                      │
│                      │     "script_timeout": 90,                │
│                      │     "user": "mxs"                        │
│                      │ }                                        │
├──────────────────────┼──────────────────────────────────────────┤
│ Monitor Diagnostics  │ {}                                       │
└──────────────────────┴──────────────────────────────────────────┘
```

Output should align to the MariaDB Monitor (mariadbmon) configuration you performed.

Check Service Configuration

Use the maxctrl list services and maxctrl show service commands to view the configured routing services.

This action is performed on the MaxScale node:

1. Obtain the full list of routing services:

```
$ maxctrl list services
```

```
┌──────────────────────────┬─────────────┬─────────────┬─────────────────┬──────────────────┐
│ Service                  │ Router      │ Connections │ Total Connections │ Servers          │
├──────────────────────────┼─────────────┼─────────────┼─────────────────┼──────────────────┤
│ connection_router_Service │ readconnroute │ 0           │ 0               │ mcs1, mcs2, mcs3 │
├──────────────────────────┼─────────────┼─────────────┼─────────────────┼──────────────────┤
│ query_router_service     │ readwritesplit │ 0           │ 0               │ mcs1, mcs2, mcs3 │
└──────────────────────────┴─────────────┴─────────────┴─────────────────┴──────────────────┘
```

2. For each service, view the service configuration:

```
$ maxctrl show service query_router_service
```

```
┌──────────────────┬──────────────────────────────────────────────────────────────────────────────┐
│ Service          │ query_router_service                                                         │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Router           │ readwritesplit                                                               │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ State            │ Started                                                                      │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Started At       │ Sat Aug 28 21:41:16 2021                                                     │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Current Connections │ 0                                                                         │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Total Connections │ 0                                                                           │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Max Connections  │ 0                                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Cluster          │                                                                              │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Servers          │ mcs1                                                                         │
│                  │ mcs2                                                                         │
│                  │ mcs3                                                                         │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Services         │                                                                              │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Filters          │                                                                              │
├──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Parameters       │ {                                                                            │
│                  │     "auth_all_servers": false,                                               │
│                  │     "causal_reads": "false",                                                 │
│                  │     "causal_reads_timeout": "10000ms",                                       │
│                  │     "connection_keepalive": "300000ms",                                      │
│                  │     "connection_timeout": "0ms",                                             │
│                  │     "delayed_retry": false,                                                  │
│                  │     "delayed_retry_timeout": "10000ms",                                      │
│                  │     "disable_sescmd_history": false,                                         │
│                  │     "enable_root_user": false,                                               │
│                  │     "idle_session_pool_time": "-1000ms",                                     │
│                  │     "lazy_connect": false,                                                   │
│                  │     "localhost_match_wildcard_host": true,                                   │
│                  │     "log_auth_warnings": true,                                               │
│                  │     "master_accept_reads": false,                                            │
│                  │     "master_failure_mode": "fail_instantly",                                 │
│                  │     "master_reconnection": false,                                            │
│                  │     "max_connections": 0,                                                    │
│                  │     "max_sescmd_history": 50,                                                │
│                  │     "max_slave_connections": 255,                                            │
│                  │     "max_slave_replication_lag": "0ms",                                      │
│                  │     "net_write_timeout": "0ms",                                              │
│                  │     "optimistic_trx": false,                                                 │
│                  │     "password": "******",                                                    │
│                  │     "prune_sescmd_history": true,                                            │
│                  │     "rank": "primary",                                                       │
│                  │     "retain_last_statements": -1,                                            │
│                  │     "retry_failed_reads": true,                                              │
│                  │     "reuse_prepared_statements": false,                                      │
│                  │     "router": "readwritesplit",                                              │
│                  │     "session_trace": false,                                                  │
│                  │     "session_track_trx_state": false,                                        │
│                  │     "slave_connections": 255,                                                │
│                  │     "slave_selection_criteria": "LEAST_CURRENT_OPERATIONS",                  │
│                  │     "strict_multi_stmt": false,                                              │
│                  │     "strict_sp_calls": false,                                                │
│                  │     "strip_db_esc": true,                                                    │
│                  │     "transaction_replay": false,                                             │
│                  │     "transaction_replay_attempts": 5,                                        │
│                  │     "transaction_replay_max_size": 1073741824,                               │
│                  │     "transaction_replay_retry_on_deadlock": false,                           │
│                  │     "type": "service",                                                       │
│                  │     "use_sql_variables_in": "all",                                           │
```

```
|                      |     "user": "mxs",                                                  |
|                      |     "version_string": null                                          |
|                      | }                                                                   |
├──────────────────────┼─────────────────────────────────────────────────────────────────────┤
| Router Diagnostics   | {                                                                   |
|                      |     "avg_sescmd_history_length": 0,                                 |
|                      |     "max_sescmd_history_length": 0,                                 |
|                      |     "queries": 0,                                                   |
|                      |     "replayed_transactions": 0,                                     |
|                      |     "ro_transactions": 0,                                           |
|                      |     "route_all": 0,                                                 |
|                      |     "route_master": 0,                                              |
|                      |     "route_slave": 0,                                               |
|                      |     "rw_transactions": 0,                                           |
|                      |     "server_query_statistics": []                                   |
|                      | }                                                                   |
└──────────────────────┴─────────────────────────────────────────────────────────────────────┘
```

Output should align to the Read Connection Router (readconnroute) or Read/Write Split Router (readwritesplit) configuration you performed.

Test Application User

Applications should use a dedicated user account. The user account must be created on the primary server.

When users connect to MaxScale, MaxScale authenticates the user connection before routing it to an Enterprise Server node. Enterprise Server authenticates the connection as originating from the IP address of the MaxScale node.

The application users must have one user account with the host IP address of the application server and a second user account with the host IP address of the MaxScale node.

The requirement of a duplicate user account can be avoided by enabling the `proxy_protocol` parameter for MaxScale and the proxy_protocol_networks for Enterprise Server.

Create a User to Connect from MaxScale

This action is performed on the primary Enterprise ColumnStore node:

1. Connect to the primary Enterprise ColumnStore node:

```
$ sudo mariadb
```

2. Create the database user account for your MaxScale node:

```
CREATE USER 'app_user'@'192.0.2.10' IDENTIFIED BY 'app_user_passwd';
```

Replace 192.0.2.10 with the relevant IP address specification for your MaxScale node.

Passwords should meet your organization's password policies.

3. Grant the privileges required by your application to the database user account for your MaxScale node:

```
GRANT ALL ON test.* TO 'app_user'@'192.0.2.10';
```

The privileges shown are designed to allow the tests in the subsequent sections to work. The user account for your production application may require different privileges.

Create a User to Connect from the Application Server

This action is performed on the primary Enterprise ColumnStore node:

1. Create the database user account for your application server:

```
CREATE USER 'app_user'@'192.0.2.11' IDENTIFIED BY 'app_user_passwd';
```

Replace 192.0.2.11 with the relevant IP address specification for your application server.

Passwords should meet your organization's password policies.

2. Grant the privileges required by your application to the d database user account for your application server:

```
GRANT ALL ON test.* TO 'app_user'@'192.0.2.11';
```

The privileges shown are designed to allow the tests in the subsequent sections to work. The user account for your production application may require different privileges.

Test Connection with Application User

To test the connection, use the MariaDB Client from your application server to connect to an Enterprise ColumnStore node through MaxScale.

This action is performed **on a client connected to the MaxScale node**:

```
$ mariadb --host 192.0.2.10 --port 3307
      --user app_user --password
```

Test Connection with Read Connection Router

If you configured the Read Connection Router, confirm that MaxScale routes connections to the replica servers.

On the MaxScale node, use the maxctrl list listeners command to view the available listeners and ports:

```
$ maxctrl list listeners
```

```
┌─────────────────────────────┬──────┬──────┬─────────┬──────────────────────────┐
│ Name                        │ Port │ Host │ State   │ Service                  │
├─────────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ connection_router_listener  │ 3308 │ ::   │ Running │ connection_router_service │
├─────────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ query_router_listener       │ 3307 │ ::   │ Running │ query_router_service      │
└─────────────────────────────┴──────┴──────┴─────────┴──────────────────────────┘
```

2. Open multiple terminals connected to your application server, in each, use MariaDB Client to connect to the listener port for the Read Connection Router (in the example, 3308):

```
$ mariadb --host 192.0.2.10 --port 3308 \
      --user app_user --password
```

Use the application user credentials you created for the `--user and --password` options.

3. In each terminal, query the hostname and server_id system variable and option to identify to which you're connected:

```
SELECT @@global.hostname, @@global.server_id;

+-------------------+--------------------+
| @@global.hostname | @@global.server_id |
+-------------------+--------------------+
|              mcs2 |                  2 |
+-------------------+--------------------+
```

Different terminals should return different values since MaxScale routes the connections to different nodes.

Since the router was configured with the slave router option, the Read Connection Router only routes connections to replica servers.

Test Write Queries with Read/Write Split Router

If you configured the Read/Write Split Router, confirm that MaxScale routes write queries on this router to the primary Enterprise ColumnStore node.

on the MaxScale node, use the maxctrl list listeners command to view the available listeners and ports:

```
$ maxctrl list listeners
```

```
┌──────────────────────────────┬──────┬──────┬─────────┬──────────────────────────────┐
| Name                         | Port | Host | State   | Service                      |
├──────────────────────────────┼──────┼──────┼─────────┼──────────────────────────────┤
| connection_router_listener | 3308 | ::   | Running | connection_router_service |
├──────────────────────────────┼──────┼──────┼─────────┼──────────────────────────────┤
| query_router_listener      | 3307 | ::   | Running | query_router_service      |
└──────────────────────────────┴──────┴──────┴─────────┴──────────────────────────────┘
```

2. Open multiple terminals connected to your application server, in each, use MariaDB Client to connect to the listener port for the Read/Write Split Router (in the example, 3307):

```
$ mariadb --host 192.0.2.10 --port 3307 \
      --user app_user --password
```

Use the application user credentials you created for the `--user and --password` options.

3. In one terminal, create the test table:

```
CREATE TABLE test.load_balancing_test (
   id INT PRIMARY KEY AUTO_INCREMENT,
   hostname VARCHAR(256),
   server_id INT
);
```

4. In each terminal, issue an insert.md statement to add a row to the example table with the values of the hostname and server_id system variable and option:

```
INSERT INTO test.load_balancing_test (hostname, server_id)
VALUES (@@global.hostname, @@global.server_id);
```

5. In one terminal, issue a SELECT statement to query the results:

```
SELECT * FROM test.load_balancing_test;
```

```
+----+----------+-----------+
| id | hostname | server_id |
+----+----------+-----------+
|  1 | mcs1     |         1 |
|  2 | mcs1     |         1 |
|  3 | mcs1     |         1 |
+----+----------+-----------+
```

While MaxScale is handling multiple connections from different terminals, it routed all connections to the current primary Enterprise ColumnStore node, which in the example is `mcs1#.`

Test Read Queries with Read/Write Split Router

If you configured the Read/Write Split Router (readwritesplit), confirm that MaxScale routes read queries on this router to replica servers.

1. On the MaxScale node, use the maxctrl list listeners command to view the available listeners and ports:

```
$ maxctrl list listeners
```

```
┌──────────────────────────┬──────┬──────┬─────────┬──────────────────────────┐
│ Name                     │ Port │ Host │ State   │ Service                  │
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ connection_router_listener │ 3308 │ ::   │ Running │ connection_router_service │
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ query_router_listener    │ 3307 │ ::   │ Running │ query_router_service     │
└──────────────────────────┴──────┴──────┴─────────┴──────────────────────────┘
```

2. In a terminal connected to your application server, use MariaDB Client to connect to the listener port for the Read/Write Split Router (readwritesplit) (in the example, 3307):

```
$ mariadb --host 192.0.2.10 --port 3307 \
      --user app_user --password
```

Use the application user credentials you created for the –user and –password options.

3. Query the hostname and server_id to identify which server MaxScale routed you to.

```
SELECT @@global.hostname, @@global.server_id;
```

```
+-------------------+--------------------+
| @@global.hostname | @@global.server_id |
+-------------------+--------------------+
|              mcs2 |                  2 |
+-------------------+--------------------+
```

4. Resend the query:

```
SELECT @@global.hostname, @@global.server_id;
```

```
+-------------------+--------------------+
| @@global.hostname | @@global.server_id |
+-------------------+--------------------+
|              mcs3 |                  3 |
+-------------------+--------------------+
```

Confirm that MaxScale routes the SELECT statements to different replica servers.

For more information on different routing criteria, see `slave_selection_criteria`

Next Step

“Deploy ColumnStore Shared Local Storage Topology”.

This page was step 8 of 9.

Next: Step 9: Import Data.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 9: Import Data

Overview

This page details step 9 of the 9-step procedure "Deploy ColumnStore Shared Local Storage Topology".

This step bulk imports data to Enterprise ColumnStore.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Import the Schema

Before data can be imported into the tables, create a matching schema.

On the primary server, create the schema:

1. For each database that you are importing, create the database with the CREATE DATABASE statement:

```
CREATE DATABASE inventory;
```

2. For each table that you are importing, create the table with the CREATE TABLE statement:

```
CREATE TABLE inventory.products (
   product_name VARCHAR(11) NOT NULL DEFAULT '',
   supplier VARCHAR(128) NOT NULL DEFAULT '',
   quantity VARCHAR(128) NOT NULL DEFAULT '',
   unit_cost VARCHAR(128) NOT NULL DEFAULT ''
) ENGINE=Columnstore DEFAULT CHARSET=utf8;
```

Import the Data

Enterprise ColumnStore supports multiple methods to import data into ColumnStore tables.

Interface	Method	Benefits
Shell	cpimport	
SQL	LOAD DATA INFILE	
Remote Database	Remote Database Import	

cpimport

MariaDB Enterprise ColumnStore includes cpimport, which is a command-line utility designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a TSV (tab-separated values) file, on the primary server run cpimport:

```
$ sudo cpimport -s '\t' inventory products /tmp/inventory-products.tsv
```

LOAD DATA INFILE

When data is loaded with the LOAD DATA INFILE statement, MariaDB Enterprise ColumnStore loads the data using cpimport, which is a command-line utility designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a TSV (tab-separated values) file, on the primary server use LOAD DATA INFILE statement:

```
LOAD DATA INFILE '/tmp/inventory-products.tsv'
INTO TABLE inventory.products;
```

Import from Remote Database

MariaDB Enterprise ColumnStore can also import data directly from a remote database. A simple method is to query the table using the SELECT statement, and then pipe the results into cpimport, which is a command-line utility that is designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a remote MariaDB database:

```
$ mariadb --quick \
   --skip-column-names \
   --execute="SELECT * FROM inventory.products" \
   | cpimport -s '\t' inventory products
```

Next Step

Navigation in the procedure “Deploy ColumnStore Shared Local Storage Topology”.

This page was step 9 of 9.

This procedure is complete.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Galera Cluster

Overview

Software Version	Diagram	Features

This procedure describes the deployment of the **Galera Cluster topology** with MariaDB Enterprise Server 11.4 and MariaDB MaxScale 25.01.

MariaDB Enterprise Cluster is powered by Galera.

MariaDB Enterprise Cluster provides read scalability and fault tolerance through virtually synchronous multi-primary certification-based write-set replication (wsrep).

This procedure has 6 steps, which are executed in sequence.

MariaDB products can be deployed in many different topologies to suit specific use cases. Enterprise Cluster can be deployed on its own, or integrated with MariaDB Replication to integrate with other clusters or topologies.

This procedure represents basic product capability with 3 Enterprise Cluster nodes and 1 MaxScale node.

This page provides an overview of the topology, requirements, and deployment procedures.

Please read and understand this procedure before executing.

Procedure Steps

Step	Description
Step 1	Install MariaDB Enterprise Server
Step 2	Start and Configure MariaDB Enterprise Server
Step 3	Test MariaDB Enterprise Server
Step 4	Install MariaDB MaxScale
Step 5	Start and Configure MariaDB MaxScale
Step 6	Test MariaDB MaxScale

Support

Customers can obtain support by submitting a support case.

Components

The following components are deployed during this procedure:

Component	Function
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.
MariaDB MaxScale 25.01	Database proxy that extends the availability, scalability, and security of MariaDB Enterprise Servers

MariaDB Enterprise Server Components

Component	Description
Enterprise Cluster	MariaDB Enterprise Server leverages the Galera Enterprise 4 wsrep provider plugin • Provides virtually synchronous multi-primary replication for MariaDB Enterprise Server • All nodes can handle both reads and writes • Replicates write-sets to all other nodes in the cluster • Supports data-at-rest encryption of the write-set cache
InnoDB	• General purpose storage engine • ACID-compliant • Performance • Required for Enterprise Cluster

MariaDB MaxScale Components

Component	Description
Galera Monitor	Tracks changes in the state of MariaDB Enterprise Servers operating as Enterprise Cluster nodes.
Listener	Listens for client connections to MaxScale, then passes them to the router service associated with the listener
Read Connection Router	Routes connections from the listener to any available Enterprise Cluster node
Read/Write Split Router	Routes read operations from the listener to any available Enterprise Cluster node, and routes write operations from the listener to a specific server that MaxScale uses as the primary server
Server Module	Connection configuration in MaxScale to an Enterprise Cluster node

Topology

MariaDB Enterprise Cluster topology provides read scalability through certification-based write-set replication (wsrep) that is multi-primary and virtually synchronous.

The Enterprise Cluster topology consists of:

- 1 or more MaxScale nodes
- 3 or more MariaDB Enterprise Servers (ES) configured as Enterprise Cluster nodes

The MaxScale nodes:

- Monitor the health and availability of each Enterprise Cluster node using the Galera Monitor (galeramon)
- Accept client and application connections

- Route queries to the Enterprise Cluster nodes using the Read Connection (readconnroute) or the Read/Write Split (readwritesplit) routers.

The Enterprise Cluster nodes:

- Receive queries from MaxScale
- Store data locally using the InnoDB storage engine
- Perform certification-based virtually synchronous replication to other Enterprise Cluster nodes
- Provide State Snapshot Transfers (SST) to bring MariaDB Enterprise Server nodes into sync with the cluster

Requirements

These requirements are for the Galera Cluster topology when deployed with MariaDB Enterprise Server and MariaDB MaxScale 25.01.

- Node Count
- Operating System

Node Count

- MaxScale nodes, 1 or more are required.
- Enterprise Cluster nodes, 3 or more are required.

To avoid problems in establishing a quorum in the event of a network partition or outage, MariaDB recommends deploying an odd number of Enterprise Cluster nodes. When using multiple network switches, deploy across an odd number of switches, each with an odd number of nodes. When using multiple data centers, deploy across an odd number of data centers, each with an odd number of switches.

Operating System

In alignment to the enterprise lifecycle, the Galera Cluster topology with MariaDB Enterprise Server and MariaDB MaxScale 25.01 is provided for:

- AlmaLinux 8 (x86_64, ARM64)
- AlmaLinux 9 (x86_64, ARM64)
- Debian 11 (x86_64, ARM64)
- Debian 12 (x86_64, ARM64)
- Red Hat Enterprise Linux 8 (x86_64, ARM64)
- Red Hat Enterprise Linux 9 (x86_64, PPC64LE, ARM64)
- Red Hat UBI 8 (x86_64, ARM64)
- Rocky Linux 8 (x86_64, ARM64)
- Rocky Linux 9 (x86_64, ARM64)
- SUSE Linux Enterprise Server 15 (x86_64, ARM64)
- Ubuntu 20.04 LTS (x86_64, ARM64)
- Ubuntu 22.04 LTS (x86_64, ARM64)
- Ubuntu 24.04 LTS (x86_64, ARM64)

Quick Reference

MariaDB Enterprise Server Configuration Management

Method	Description
Configuration File	Configuration files (such as `/etc/my.cnf`) can be used to set system-variables and options. The server must be restarted to apply changes made to configuration files.

Method	Description
Command-line	The server can be started with command-line options that set system-variables and options.
SQL	Users can set system-variables that support dynamic changes on-the-fly using the SET statement.

MariaDB Enterprise Server packages are configured to read configuration files from different paths, depending on the operating system. Making custom changes to Enterprise Server default configuration files is not recommended because custom changes may be overwritten by other default configuration files that are loaded later.

To ensure that your custom changes will be read last, create a custom configuration file with the `z-` prefix in one of the include directories.

Distribution	Example Configuration File Path
• CentOS • Red Hat Enterprise Linux (RHEL) • SUSE Linux Enterprise Server (SLES)	`/etc/my.cnf.d/z-custom-mariadb.cnf`
• Debian • Ubuntu	`/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf`

MariaDB Enterprise Server Service Management

The `systemctl` command is used to start and stop the MariaDB Enterprise Server service. The `galera_new_cluster and galera_recovery` scripts are used for Enterprise Cluster-specific operations.

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`
Bootstrap a cluster node	`sudo galera_new_cluster`
Recover a cluster node's position	`sudo galera_recovery`

For additional information, see "Starting and Stopping MariaDB".

MariaDB Enterprise Server Logs

MariaDB Enterprise Server produces log data that can be helpful in problem diagnosis.

Log filenames and locations may be overridden in the server configuration. The default location of logs is the data directory. The data directory is specified by the datadir system variable.

Log	System Variable/Option	Default Filename
MariaDB Error Log	log_error	`<hostname>.err`
MariaDB Enterprise Audit Log	server_audit_file_path	`server_audit.log`
Slow Query Log	slow_query_log_file	`<hostname>-slow.log`
General Query Log	general_log_file	`<hostname>.log`
Binary Log	log_bin	`<hostname>-bin`

MaxScale Configuration Management

MaxScale can be configured using several methods. These methods make use of MaxScale's REST API.

Method	Benefits
MaxCtrl	Command-line utility to perform administrative tasks through the REST API. See MaxCtrl Commands.
MaxGUI	MaxGUI is a graphical utility that can perform administrative tasks through the REST API.
REST API	The REST API can be used directly. For example, the curl utility could be used to make REST API calls from the command-line. Many programming languages also have libraries to interact with REST APIs.

The procedure on these pages configures MaxScale using MaxCtrl.

MaxScale Service Management

The systemctl command is used to start and stop the MaxScale service.>

Operation	Command
Start	`sudo systemctl start maxscale`
Stop	`sudo systemctl stop maxscale`
Restart	`sudo systemctl restart maxscale`
Enable during startup	`sudo systemctl enable maxscale`
Disable during startup	`sudo systemctl disable maxscale`
Status	`sudo systemctl status maxscale`

For additional information, see "Starting and Stopping MariaDB".

Next Step

Navigation in the procedure "Deploy Galera Cluster Topology":

Next: Step 1: Install MariaDB Enterprise Server

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 1: Install MariaDB Enterprise Server

Overview

This page details step 1 of the 6-step procedure "Deploy Galera Cluster Topology".

This step installs MariaDB Enterprise Server.

MariaDB Enterprise Server installations support MariaDB Enterprise Cluster, powered by Galera. MariaDB Enterprise Cluster uses the Galera Enterprise 4 wsrep provider plugin.

MariaDB Enterprise Cluster requires an odd number of 3 or more nodes. Nodes must meet requirements.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Download Token

MariaDB Corporation provides package repositories for CentOS / RHEL (YUM) and Debian / Ubuntu (APT). A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to https://customers.mariadb.com/downloads/token/.
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On each Enterprise ColumnStore node, install the prerequisites for downloading the software from the Web. Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

2. On each Enterprise ColumnStore node, configure package repositories and specify Enterprise Server:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-maxscale \
      --skip-tools \
      --mariadb-server-version="11.4"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install Enterprise Server

1. On each Enterprise Cluster node, install MariaDB Enterprise Server and MariaDB Enterprise Backup.

Install via CentOS / RHEL (YUM):

```
$ sudo yum install MariaDB-server MariaDB-backup
```

Install via Debian / Ubuntu (APT):

```
$ sudo apt update

$ sudo apt install mariadb-server mariadb-backup
```

Install via SLES (ZYpp):

```
$ sudo zypper install MariaDB-server MariaDB-backup
```

Next Step

Navigation in the procedure “Deploy Galera Cluster Topology”:

This page was step 1 of 6.

Next: Step 2: Start and Configure MariaDB Enterprise Server

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 2: Start and Configure MariaDB Enterprise Server

Overview

This page details step 2 of the 6-step procedure "Deploy Galera Cluster Topology".

This step configures MariaDB Enterprise Servers to operate as Enterprise Cluster nodes and starts MariaDB Enterprise Cluster.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Stop the Enterprise Server Service

The installation process might have started the Enterprise Server service. The service should be stopped prior to making configuration changes.

On each Enterprise Cluster node, stop the MariaDB Enterprise Server service:

```
$ sudo systemctl stop mariadb
```

Acquire TLS Certificate

By default, MariaDB Enterprise Cluster requires data-in-transit encryption to secure Galera replication traffic.

MariaDB Enterprise Cluster encrypts the data using the Transport Layer Security (TLS) protocol, which is a newer version of the Secure Socket Layer (SSL) protocol.

In MariaDB Enterprise Cluster 10.5 and earlier, TLS was supported, but not required. For backward compatibility, MariaDB Enterprise Cluster supports the Provider WSREP TLS Mode, which is equivalent to Enterprise Cluster's TLS implementation in ES 10.5 and earlier. For additional information, see "WSREP TLS Modes".

TLS configuration requires 3 files.

Example Filename	Description
ca-cert.pem	Certificate Authority (CA) file.
server-cert.pem	X.509 certificate file.
server-key.pem	X.509 key file.

Self-signed certificates are supported. However, in environments where security is critical, it is recommended to use certificates signed by a trusted Certificate Authority (CA).

For additional information, see "Data-in-Transit Encryption".

Configure Enterprise Cluster

MariaDB Enterprise Server installations support MariaDB Enterprise Cluster, powered by Galera. MariaDB Enterprise Cluster uses the Galera Enterprise 4 wsrep provider plugin. The path to the wsrep provider plugin must be configured using the wsrep_provider system variable.

Required System Variables and Options

Enterprise Cluster nodes require that you set the following system variables and options:

System Variable/Option	Description
bind_address	The network socket Enterprise Cluster listens on for incoming TCP/IP client connections. On Debian or Ubuntu, this system variable must be set to override the 127.0.0.1 default configuration.
binlog_format	Enterprise Cluster requires use of the ROW Binary Log format.
innodb_autoinc_lock_mode	Enterprise Cluster requires an auto-increment lock mode of 2.
ssl_ca	Certificate Authority (CA) file in PEM format.
ssl_cert	X.509 certificate file in PEM format.
ssl_key	X.509 key file in PEM format.
wsrep_cluster_address	Sets the Group Communications back-end (usually gcomm:), followed by a comma-separated list of IP addresses or domain names for each Cluster Node. It is best practice to include all Enterprise Cluster nodes in this list.
wsrep_cluster_name	Sets the logical name for the cluster. Must be the same on all Cluster Nodes.
wsrep_on	Enables Enterprise Cluster.
wsrep_provider	Path to the Galera Enterprise 4 wsrep provider plugin. In MariaDB Enterprise Cluster 11.4, the path is /usr/lib/galera/libgalera_enterprise_smm.so on Debian and Ubuntu and /usr/lib64/galera/libgalera_enterprise_smm.so on CentOS, RHEL, and SLES.
wsrep_provider_options	Accepts options for the Galera Enterprise 4 wsrep provider plugin. Multiple options can be specified, separated by a semi-colon (;). For example, can be used to configure Galera cache size or to enable TLS. For a complete list of available options, see wsrep_provider_options

Example Configuration

Edit a configuration file and set these system variables and options:

```
[mariadb]
bind_address             = 0.0.0.0
binlog_format            = ROW
innodb_autoinc_lock_mode = 2
wsrep_cluster_address    = gcomm://192.0.2.101,192.0.2.102,192.0.2.103
wsrep_cluster_name       = example-cluster
wsrep_on                 = ON

# wsrep provider path for Debian and Ubuntu:
wsrep_provider = /usr/lib/galera/libgalera_enterprise_smm.so

# wsrep provider path for CentOS, RHEL, and SLES:
# wsrep_provider = /usr/lib64/galera/libgalera_enterprise_smm.so

wsrep_provider_options = "gcache.size=2G;gcs.fc_limit=128"

# TLS Configuration
ssl_ca   = /path/to/ca-cert.pem
ssl_cert = /path/to/sever-cert.pem
ssl_key  = /path/to/server-key.pem
```

For additional information, see “MariaDB Enterprise Server Configuration Management”.

Configure MariaDB Replication

MariaDB Enterprise Cluster can be deployed alongside MariaDB Replication. Deploying MariaDB Enterprise Cluster with MariaDB Replication enables integrating Enterprise Cluster with other products and clusters, for example as

separate clusters in different data centers, or as a small dedicated write cluster with two larger dedicated read clusters.

For additional information, see “Replication Configuration”.

Bootstrap the Primary Component

When an Enterprise Cluster node starts, it checks the addresses in the wsrep_cluster_address system variable to establish connections with other nodes. The node does not become active until it finds a node that belongs to the Primary Component.

To start the cluster when all nodes are down, you must bootstrap the Primary Component on one node. This allows the other nodes to connect to a working cluster.

On one Enterprise Cluster node, when all nodes are down, bootstrap the Primary Component.

1. Bootstrap the Primary Component:

```
$ sudo galera_new_cluster
```

For additional information, see “Bootstrap a Galera Cluster”.

2. Connect with MariaDB Client:

```
$ sudo mariadb
```

The sudo command is used here to connect to the Enterprise Server node using the `root@localhost` user account, which authenticates using the `unix_socket` authentication plugin. Other user accounts can be used by specifying the –user and –password command-line options.

Use the SHOW STATUS statement to check the wsrep_cluster_size status variable:

```
SHOW GLOBAL STATUS LIKE 'wsrep_cluster_size';

+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size |     1 |
+--------------------+-------+
```

The Enterprise Cluster node launches as the Primary Component of a single-node cluster.

Add Nodes to the Cluster

To add nodes to a cluster that has a Primary Component running, complete the following procedure for each Enterprise Cluster node to be added. Nodes should be added one at a time.

1. On the Enterprise Cluster node being added, start MariaDB Enterprise Server:

```
$ sudo systemctl start mariadb
```

For additional information, see “Start and Stop Services”.

2. On the Enterprise Cluster node being added, connect with MariaDB Client:

```
$ sudo mariadb
```

3. On the bootstrapped Enterprise Cluster node, use the SHOW STATUS statement to check the wsrep_cluster_size status variable:

```
SHOW GLOBAL STATUS LIKE 'wsrep_cluster_size';

+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size |     2 |
+--------------------+-------+
```

4. On the Enterprise Cluster node being added, use the SHOW STATUS statement to check the wsrep_local_state_comment status variable. If wsrep_local_state_comment is `SYNCED`, the node has been successfully added, and the Add Node procedure can be repeated to add more nodes.

```
SHOW GLOBAL STATUS LIKE 'wsrep_local_state_comment';

+---------------------------+--------+
| Variable_name             | Value  |
+---------------------------+--------+
| wsrep_local_state_comment | Synced |
+---------------------------+--------+
```

When each new Enterprise Cluster node joins the cluster, it requests the current cluster position. If the new node is missing transactions, it initiates either a State Snapshot Transfer (SST) or an Incremental State Transfer (IST) from a donor node to synchronize its data with the Primary Component. Depending on the value of wsrep_sst_method , the donor node may or may not be blocked during an SST.

When the new Enterprise Cluster node finishes its state transfer, the node updates the wsrep_local_state_comment status variable to `SYNCED`. MaxScale registers the change and begins routing connections or queries to the new node.

Next Step

Navigation in the procedure “Deploy Galera Cluster Topology”:

This page was step 2 of 6.

Next: Step 3: Test MariaDB Enterprise Server

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 3: Test MariaDB Enterprise Server

Overview

This page details step 3 of the 6-step procedure “Deploy Galera Cluster Topology”.

This step tests MariaDB Enterprise Server.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Test Enterprise Server Service

Use Systemd to test whether the MariaDB Enterprise Server service is running.

This action is performed on each Enterprise Cluster node.

Check if the MariaDB Enterprise Server service is running by executing the following:

```
$ systemctl status mariadb
```

If the service is not running on any node, start the service by executing the following on that node:

```
$ sudo systemctl start mariadb
```

Test Local Client Connections

Use MariaDB Client to test the local connection to the Enterprise Server node.

This action is performed on each Enterprise Cluster node:

```
$ sudo mariadb

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The sudo command is used here to connect to the Enterprise Server node using the `root@localhost` user account, which authenticates using the `unix_socket` authentication plugin. Other user accounts can be used by specifying the –user and –password command-line options.

Test Cluster Status

MariaDB Enterprise Cluster is operational when the cluster has a Primary Component. Query the wsrep_cluster_status status variable with SHOW GLOBAL STATUS to confirm that each node belongs to the Primary Component.

This action is performed on each Enterprise Cluster node.

Check the cluster status by executing the following:

```
SHOW GLOBAL STATUS LIKE 'wsrep_cluster_status';

+----------------------+---------+
| Variable_name        | Value   |
+----------------------+---------+
| wsrep_cluster_status | Primary |
+----------------------+---------+
```

If the Value column does not contain Primary on any node, then the node is not part of the Primary Component. Investigate network connectivity between the node and the nodes in the Primary Component.

Test Cluster Size

MariaDB Enterprise Cluster maintains a count of the cluster size. Query the wsrep_cluster_size status variable with SHOW GLOBAL STATUS to confirm the number of nodes currently in the cluster.

This action is performed on each Enterprise Cluster node.

Check the cluster size by executing the following:

```
SHOW GLOBAL STATUS LIKE 'wsrep_cluster_size';

+--------------------+---------+
| Variable_name      | Value   |
+--------------------+---------+
| wsrep_cluster_size |       3 |
+--------------------+---------+
```

If the Value column does not contain the expected number of nodes, then some nodes might not be in the cluster. Check the value of the wsrep_cluster_address on each node to confirm that all nodes are in the same cluster.

Test DDL

Use MariaDB Client to test DDL.

1. On a single Enterprise Cluster node, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

Create a test database and InnoDB table:

```
CREATE DATABASE IF NOT EXISTS test;

CREATE TABLE test.contacts (
   id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   email VARCHAR(100)
);
```

3. On each other Enterprise Cluster node, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

4. Confirm that the database and table exist:

```
SHOW CREATE TABLE test.contacts\G;
```

If the database or table do not exist on any node, then check that:

The nodes are in the Primary Component of the same cluster.

The wsrep_osu_method system variable is not set to RSU.

Test DML

Use MariaDB Client to test DML.

1. On a single Enterprise Cluster node, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

2. Insert sample data into the table created in the DDL test:

```
INSERT INTO test.contacts (first_name, last_name, email)
   VALUES
   ("Kai", "Devi", "kai.devi@example.com"),
   ("Lee", "Wang", "lee.wang@example.com");
```

3. On each other Enterprise Cluster node, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

4. Execute a SELECT query to retrieve the data:

```
SELECT * FROM test.contacts;

+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
| 1  | Kai        | Devi      | kai.devi@example.com |
| 2  | Lee        | Wang      | lee.wang@example.com |
+----+------------+-----------+----------------------+
```

If the data is not returned on any node, then check that:

- The nodes are in the Primary Component of the same cluster.
- The table uses the InnoDB storage engine.
- The wsrep_on system variable is set to ON.

Next Step

Navigation in the procedure “Deploy Galera Cluster Topology”:

This page was step 3 of 6.

Next: Step 4: Install MariaDB MaxScale

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 4: Install MariaDB MaxScale

Overview

This page details step 4 of the 6-step procedure “Deploy Galera Cluster Topology”.

This step installs MariaDB MaxScale.

MariaDB Enterprise Cluster requires 1 or more MaxScale nodes.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Customer Download Token

MariaDB Corporation provides package repositories for YUM (RHEL / CentOS), APT (Debian / Ubuntu), and ZYpp (SLES). A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On the MaxScale node, install the prerequisites for downloading the software from the Web.

Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

Install on SLES (ZYpp):

```
$ sudo zypper install curl
```

2. On the MaxScale node, configure package repositories and specify MariaDB MaxScale 25.01:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup

$ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -

$ chmod +x mariadb_es_repo_setup

$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-server \
      --skip-tools \
      --mariadb-maxscale-version="25.01"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install MaxScale

1. On the MaxScale node, install MariaDB MaxScale.

Install on CentOS / RHEL (YUM):

```
$ sudo yum install maxscale
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install maxscale
```

Install on SLES (ZYpp):

```
$ sudo zypper install maxscale
```

Next Step

Navigation in the procedure “Deploy Galera Cluster Topology”:

This page was step 4 of 6.

Next: Step 5: Start and Configure MariaDB MaxScale

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 5: Start and Configure MariaDB MaxScale

Overview

This page details step 5 of the 6-step procedure “Deploy Galera Cluster Topology”.

This step configures MariaDB MaxScale to route connections to MariaDB Enterprise Cluster.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Replace the Default Configuration File

MariaDB MaxScale installations include a configuration file with some example objects. This configuration file should be replaced.

On the MaxScale node, replace the default /etc/maxscale.cnf with the following configuration:

```
[maxscale]
threads          = auto
admin_host       = 0.0.0.0
admin_secure_gui = false
```

For additional information, see “Global Parameters”.

Restart MaxScale

On the MaxScale node, restart the MaxScale service to ensure that MaxScale picks up the new configuration:

```
$ sudo systemctl restart maxscale
```

For additional information, see “Start and Stop Services”.

Create MaxScale User Account

MariaDB MaxScale connects to Enterprise Cluster through the client port. MaxScale requires its own user account to monitor and orchestrate Enterprise Cluster nodes.

1. On any Enterprise Cluster node, use the CREATE USER statement to create a new user for MaxScale:

```
CREATE USER mxs@192.0.2.104 IDENTIFIED BY "passwd";
```

2. Use the GRANT statement to grant required privileges to the MaxScale user:

```
GRANT SHOW DATABASES ON *.* TO mxs@192.0.2.104;
GRANT SELECT ON mysql.columns_priv TO mxs@192.0.2.104;
GRANT SELECT ON mysql.db TO mxs@192.0.2.104;
GRANT SELECT ON mysql.procs_priv TO mxs@192.0.2.104;
GRANT SELECT ON mysql.proxies_priv TO mxs@192.0.2.104;
GRANT SELECT ON mysql.roles_mapping TO mxs@192.0.2.104;
GRANT SELECT ON mysql.tables_priv TO mxs@192.0.2.104;
GRANT SELECT ON mysql.user TO mxs@192.0.2.104;
```

Enterprise Cluster replicates the new user and privileges to the other Enterprise Cluster nodes.

Configure Server Objects

MariaDB MaxScale uses server objects to define the connections it makes to MariaDB Enterprise Servers.

On the MaxScale node, use maxctrl create server to create a server object for each MariaDB Enterprise Server:

```
$ maxctrl create server node1 192.0.2.101

$ maxctrl create server node2 192.0.2.102

$ maxctrl create server node3 192.0.2.103
```

Configure the Galera Monitor

MaxScale uses monitors to retrieve additional information from the servers. This information is used by other services in filtering and routing connections based on the current state of the node. For Enterprise Cluster, use the Galera Monitor (galeramon).

On the MaxScale node, use maxctrl create monitor to create a Galera Monitor for the cluster:

```
$ maxctrl create monitor cluster_monitor galeramon \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     --servers node1 node2 node3
```

In this example:

- `cluster_monitor` is an arbitrary name that is used to identify the new monitor.
- `galeramon` is the name of the module that implements the Galera Monitor.
- `user=MAXSCALE_USER` sets the user parameter to the database user account that MaxScale uses to monitor the ES nodes.
- `password='MAXSCALE_USER_PASSWORD'` sets the password parameter to the password used by the database user account that MaxScale uses to monitor the ES nodes.
- `--servers sets the servers` parameter to the set of nodes that MaxScale should monitor. All non-option arguments after –servers are interpreted as server names.
- Other Module Parameters supported by galeramon in MaxScale 25.01 can also be specified.

Choose a MaxScale Router

Routers control how MaxScale balances the load between Enterprise Cluster nodes. Each router uses a different approach to routing queries. Consider the specific use case of your application and database load and select the router that best suits your needs.

Router	Configuration Procedure	Description
Read Connection (readconnroute)	Configure Read Connection Router	• Connection-based load balancing • Routes connections to Enterprise ColumnStore nodes designated as replica servers for a read-only pool • Routes connections to an Enterprise ColumnStore node designated as the primary server for a read-write pool.
Read/Write Split (readwritesplit)	Configure Read/Write Split	• Query-based load balancing • Routes write queries to an Enterprise ColumnStore node designated as the primary server • Routes read queries to Enterprise ColumnStore node designated as replica servers • Automatically reconnects after node failures • Automatically replays transactions after node failures • Optionally enforces causal reads

Configure Read Connection Router

Use MaxScale Read Connection Router (readconnroute) to route connections to replica servers for a read-only pool.

On the MaxScale node, use maxctrl create service to create a router:

```
$ maxctrl create service connection_router_service readconnroute \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     router_options=slave \
     --servers node1 node2 node3
```

In this example:

- `connection_router_service` is an arbitrary name that is used to identify the new service.
- `readconnroute` is the name of the module that implements the Read Connection Router.
- `user=MAXSCALE_USER` sets the user parameter to the database user account that MaxScale uses to connect to the ES nodes.

- `password=MAXSCALE_USER_PASSWORD` sets the password parameter to the password used by the database user account that MaxScale uses to connect to the ES nodes.
- `router_options=slave` sets the router_options parameter to slave, so that MaxScale only routes connections to the replica nodes.
- `--servers sets the servers` parameter to the set of nodes to which MaxScale should route connections. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by readconnroute in MaxScale 25.01 can also be specified.

Configure Listener for the Read Connection Router

These instructions reference TCP port 3308. You can use a different TCP port. The TCP port used must not be bound by any other listener.

On the MaxScale node, use the maxctrl create listener command to configure MaxScale to use a listener for the Read Connection Router (readconnroute):

```
$ maxctrl create listener connection_router_service connection_router_listener 3308 \
     protocol=MariaDBClient
```

In this example:

- `connection_router_service` is the name of the readconnroute service that was previously created.
- `connection_router_listener` is an arbitrary name that is used to identify the new listener.
- 3308 is the TCP port.
- `protocol=MariaDBClient` sets the protocol parameter.
- Other Module Parameters supported by listeners in MaxScale 25.01 can also be specified.

Configure Read/Write Split Router for Queries

MaxScale Read/Write Split Router (readwritesplit) performs query-based load balancing. The router routes write queries to the primary and read queries to the replicas.

On the MaxScale node, use the maxctrl create service command to configure MaxScale to use the Read/Write Split Router (readwritesplit):

```
$ maxctrl create service query_router_service readwritesplit  \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     --servers node1 node2 node3
```

In this example:

- `query_router_service` is an arbitrary name that is used to identify the new service.
- `readwritesplit` is the name of the module that implements the Read/Write Split Router.
- `user=MAXSCALE_USER` sets the user parameter to the database user account that MaxScale uses to connect to the ES nodes.
- `password=MAXSCALE_USER_PASSWORD` sets the password parameter to the password used by the database user account that MaxScale uses to connect to the ES nodes.
- `--servers sets the servers` parameter to the set of nodes to which MaxScale should route queries. All non-option arguments after –servers are interpreted as server names.
- Other Module Parameters supported by readwritesplit in MaxScale 25.01 can also be specified.

Configure a Listener for the Read/Write Split Router

These instructions reference TCP port 3307. You can use a different TCP port. The TCP port used must not be bound by any other listener.

On the MaxScale node, use the maxctrl create listener command to configure MaxScale to use a listener for the Read/Write Split Router (readwritesplit):

```
$ maxctrl create listener query_router_service query_router_listener 3307 \
     protocol=MariaDBClient
```

In this example:

- `query_router_service` is the name of the readwritesplit service that was previously created.
- `query_router_listener` is an arbitrary name that is used to identify the new listener.
- 3307 is the TCP port.
- `protocol=MariaDBClient` sets the protocol parameter.
- Other Module Parameters supported by listeners in MaxScale 25.01 can also be specified.

Start Services

To start the services and monitors, on the MaxScale node use maxctrl start services:

```
$ maxctrl start services
```

Next Step

Navigation in the procedure “Deploy Galera Cluster Topology”:

This page was step 5 of 6.

Next: Step 6: Test MariaDB MaxScale

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 6: Test MariaDB MaxScale

Overview

This page details step 6 of the 6-step procedure “Deploy Galera Cluster Topology”.

This step tests MariaDB MaxScale.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Check Global Configuration

Use maxctrl show maxscale command to view the global MaxScale configuration.

This action is performed on the MaxScale node:

```
$ maxctrl show maxscale
```

```
┌──────────────┬──────────────────────────────────────────────────────────────┐
│ Version      │ 25.01.2                                                      │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Commit       │ 3761fa7a52046bc58faad8b5a139116f9e33364c                     │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Started At   │ Thu, 05 Aug 2021 20:21:20 GMT                                │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Activated At │ Thu, 05 Aug 2021 20:21:20 GMT                                │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Uptime       │ 868                                                          │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Config Sync  │ null                                                         │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Parameters   │ {                                                            │
│              │     "admin_auth": true,                                      │
│              │     "admin_enabled": true,                                   │
│              │     "admin_gui": true,                                       │
│              │     "admin_host": "0.0.0.0",                                 │
│              │     "admin_log_auth_failures": true,                         │
│              │     "admin_pam_readonly_service": null,                      │
│              │     "admin_pam_readwrite_service": null,                     │
│              │     "admin_port": 8989,                                      │
│              │     "admin_secure_gui": false,                               │
│              │     "admin_ssl_ca_cert": null,                               │
│              │     "admin_ssl_cert": null,                                  │
│              │     "admin_ssl_key": null,                                   │
│              │     "admin_ssl_version": "MAX",                              │
│              │     "auth_connect_timeout": "10000ms",                       │
│              │     "auth_read_timeout": "10000ms",                          │
│              │     "auth_write_timeout": "10000ms",                         │
│              │     "cachedir": "/var/cache/maxscale",                       │
│              │     "config_sync_cluster": null,                             │
│              │     "config_sync_interval": "5000ms",                        │
│              │     "config_sync_password": "******",                        │
│              │     "config_sync_timeout": "10000ms",                        │
│              │     "config_sync_user": null,                                │
│              │     "connector_plugindir": "/usr/lib64/mysql/plugin",        │
│              │     "datadir": "/var/lib/maxscale",                          │
│              │     "debug": null,                                           │
│              │     "dump_last_statements": "never",                         │
│              │     "execdir": "/usr/bin",                                   │
│              │     "language": "/var/lib/maxscale",                         │
│              │     "libdir": "/usr/lib64/maxscale",                         │
│              │     "load_persisted_configs": true,                          │
│              │     "local_address": null,                                   │
│              │     "log_debug": false,                                      │
│              │     "log_info": false,                                       │
│              │     "log_notice": true,                                      │
│              │     "log_throttling": {                                      │
│              │         "count": 10,                                         │
│              │         "suppress": 10000,                                   │
│              │         "window": 1000                                       │
│              │     },                                                       │
│              │     "log_warn_super_user": false,                            │
│              │     "log_warning": true,                                     │
│              │     "logdir": "/var/log/maxscale",                           │
│              │     "max_auth_errors_until_block": 10,                       │
│              │     "maxlog": true,                                          │
│              │     "module_configdir": "/etc/maxscale.modules.d",           │
│              │     "ms_timestamp": false,                                   │
│              │     "passive": false,                                        │
│              │     "persistdir": "/var/lib/maxscale/maxscale.cnf.d",        │
│              │     "piddir": "/var/run/maxscale",                           │
│              │     "query_classifier": "qc_sqlite",                         │
│              │     "query_classifier_args": null,                           │
│              │     "query_classifier_cache_size": 289073971,                │
│              │     "query_retries": 1,                                      │
│              │     "query_retry_timeout": "5000ms",                         │
```

```
|              |     "rebalance_period": "0ms",                          |
|              |     "rebalance_threshold": 20,                          |
|              |     "rebalance_window": 10,                             |
|              |     "retain_last_statements": 0,                        |
|              |     "session_trace": 0,                                 |
|              |     "skip_permission_checks": false,                    |
|              |     "sql_mode": "default",                              |
|              |     "syslog": true,                                     |
|              |     "threads": 1,                                       |
|              |     "users_refresh_interval": "0ms",                    |
|              |     "users_refresh_time": "30000ms",                    |
|              |     "writeq_high_water": 16777216,                      |
|              |     "writeq_low_water": 8192                            |
|              | }                                                       |
└──────────────┴─────────────────────────────────────────────────────────┘
```

Output should align to the global MaxScale configuration in the new configuration file you created.

Check Server Configuration

Use the maxctrl list servers and maxctrl show server commands to view the configured server objects.

This action is performed on the MaxScale node:

1. Obtain the full list of servers objects:

```
$ maxctrl list servers
```

```
┌────────┬─────────────┬──────┬─────────────┬─────────────────────────┬──────┐
| Server | Address     | Port | Connections | State                   | GTID |
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
| node1  | 192.0.2.101 | 3306 | 0           | Slave, Synced, Running  |      |
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
| node2  | 192.0.2.102 | 3306 | 0           | Slave, Synced, Running  |      |
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
| node3  | 192.0.2.103 | 3306 | 0           | Master, Synced, Running |      |
└────────┴─────────────┴──────┴─────────────┴─────────────────────────┴──────┘
```

2. For each server object, view the configuration:

```
$ maxctrl show server node3
```

```
┌──────────────────┬──────────────────────────────────────────────────────────────┐
│ Server           │ node3                                                        │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Address          │ 192.0.2.103                                                  │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Port             │ 3306                                                         │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ State            │ Master, Synced, Running                                      │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Version          │ 11.4.5-3-MariaDB-enterprise-log                              │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Last Event       │ master_up                                                    │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Triggered At     │ Thu, 05 Aug 2021 20:22:26 GMT                                │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Services         │ connection_router_service                                    │
│                  │ query_router_service                                         │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Monitors         │ cluster_monitor                                              │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Master ID        │ -1                                                           │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Node ID          │ 1                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Slave Server IDs │                                                              │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Current Connections │ 1                                                         │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Total Connections │ 1                                                           │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Max Connections  │ 1                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Statistics       │ {                                                            │
│                  │     "active_operations": 0,                                  │
│                  │     "adaptive_avg_select_time": "0ns",                       │
│                  │     "connection_pool_empty": 0,                              │
│                  │     "connections": 1,                                        │
│                  │     "max_connections": 1,                                    │
│                  │     "max_pool_size": 0,                                      │
│                  │     "persistent_connections": 0,                             │
│                  │     "reused_connections": 0,                                 │
│                  │     "routed_packets": 0,                                     │
│                  │     "total_connections": 1                                   │
│                  │ }                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Parameters       │ {                                                            │
│                  │     "address": "192.0.2.103",                                │
│                  │     "disk_space_threshold": null,                            │
│                  │     "extra_port": 0,                                         │
│                  │     "monitorpw": null,                                       │
│                  │     "monitoruser": null,                                     │
│                  │     "persistmaxtime": "0ms",                                 │
│                  │     "persistpoolmax": 0,                                     │
│                  │     "port": 3306,                                            │
│                  │     "priority": 0,                                           │
│                  │     "proxy_protocol": false,                                 │
│                  │     "rank": "primary",                                       │
│                  │     "socket": null,                                          │
│                  │     "ssl": false,                                            │
│                  │     "ssl_ca_cert": null,                                     │
│                  │     "ssl_cert": null,                                        │
│                  │     "ssl_cert_verify_depth": 9,                              │
│                  │     "ssl_cipher": null,                                      │
│                  │     "ssl_key": null,                                         │
│                  │     "ssl_verify_peer_certificate": false,                    │
│                  │     "ssl_verify_peer_host": false,                           │
│                  │     "ssl_version": "MAX"                                     │
```

```
│                     │ }                                                   │
└─────────────────────┴─────────────────────────────────────────────────────┘
```

Output should align to the Server Object configuration you performed.

Check Monitor Configuration

Use the maxctrl list monitors and maxctrl show monitor commands to view the configured monitors.

This action is performed on the MaxScale node:

1. Obtain the full list of monitors:

```
$ maxctrl list monitors
```

```
┌─────────────────┬─────────┬──────────────────────┐
│ Monitor         │ State   │ Servers              │
├─────────────────┼─────────┼──────────────────────┤
│ cluster_monitor │ Running │ node1, node2, node3 │
└─────────────────┴─────────┴──────────────────────┘
```

2. For each monitor, view the monitor configuration:

```
$ maxctrl show monitor cluster_monitor
```

```
┌──────────────────┬──────────────────────────────────────────────────────────┐
│ Monitor          │ cluster_monitor                                          │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Module           │ galeramon                                                │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ State            │ Running                                                  │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Servers          │ node1                                                    │
│                  │ node2                                                    │
│                  │ node3                                                    │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Parameters       │ {                                                        │
│                  │     ..                                                   │
│                  │ }                                                        │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Monitor Diagnostics │ {                                                     │
│                  │     ..                                                   │
│                  │ }                                                        │
└──────────────────┴──────────────────────────────────────────────────────────┘
```

Output should align to the Galera Monitor (galeramon) configuration you performed.

Check Service Configuration

Use the maxctrl list services and maxctrl show service commands to view the configured routing services.

This action is performed on the MaxScale node:

1. Obtain the full list of routing services:

```
$ maxctrl list services
```

```
┌──────────────────────────┬──────────────┬─────────────┬─────────────────┬──────────────────┐
│ Service                  │ Router       │ Connections │ Total Connections │ Servers          │
├──────────────────────────┼──────────────┼─────────────┼─────────────────┼──────────────────┤
│ connection_router_service │ readconnroute │ 0           │ 0               │ node1, node2, node3 │
├──────────────────────────┼──────────────┼─────────────┼─────────────────┼──────────────────┤
│ query_router_service     │ readwritesplit │ 1           │ 1               │ node1, node2, node3 │
└──────────────────────────┴──────────────┴─────────────┴─────────────────┴──────────────────┘
```

2. For each service, view the service configuration:

```
$ maxctrl show service query_router_service
```

```
┌────────────────────┬──────────────────────────────────────────────────────────────────────────────┐
│ Service            │ query_router_service                                                         │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Router             │ readwritesplit                                                               │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ State              │ Started                                                                      │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Started At         │ Thu Aug  5 20:23:38 2021                                                     │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Current Connections │ 1                                                                           │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Total Connections  │ 1                                                                            │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Max Connections    │ 1                                                                            │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Cluster            │                                                                              │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Servers            │ node1                                                                        │
│                    │ node2                                                                        │
│                    │ node3                                                                        │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Services           │                                                                              │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Filters            │                                                                              │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Parameters         │ {                                                                            │
│                    │     "auth_all_servers": false,                                               │
│                    │     "causal_reads": "false",                                                 │
│                    │     "causal_reads_timeout": "10000ms",                                       │
│                    │     "connection_keepalive": "300000ms",                                      │
│                    │     "connection_timeout": "0ms",                                             │
│                    │     "delayed_retry": false,                                                  │
│                    │     "delayed_retry_timeout": "10000ms",                                      │
│                    │     "disable_sescmd_history": false,                                         │
│                    │     "enable_root_user": false,                                               │
│                    │     "idle_session_pool_time": "-1000ms",                                     │
│                    │     "lazy_connect": false,                                                   │
│                    │     "localhost_match_wildcard_host": true,                                   │
│                    │     "log_auth_warnings": true,                                               │
│                    │     "master_accept_reads": false,                                            │
│                    │     "master_failure_mode": "fail_instantly",                                 │
│                    │     "master_reconnection": false,                                            │
│                    │     "max_connections": 0,                                                    │
│                    │     "max_sescmd_history": 50,                                                │
│                    │     "max_slave_connections": 255,                                            │
│                    │     "max_slave_replication_lag": "0ms",                                      │
│                    │     "net_write_timeout": "0ms",                                              │
│                    │     "optimistic_trx": false,                                                 │
│                    │     "password": "******",                                                    │
│                    │     "prune_sescmd_history": true,                                            │
│                    │     "rank": "primary",                                                       │
│                    │     "retain_last_statements": -1,                                            │
│                    │     "retry_failed_reads": true,                                              │
│                    │     "reuse_prepared_statements": false,                                      │
│                    │     "router": "readwritesplit",                                              │
│                    │     "session_trace": false,                                                  │
│                    │     "session_track_trx_state": false,                                        │
│                    │     "slave_connections": 255,                                                │
│                    │     "slave_selection_criteria": "LEAST_CURRENT_OPERATIONS",                  │
│                    │     "strict_multi_stmt": false,                                              │
│                    │     "strict_sp_calls": false,                                                │
│                    │     "strip_db_esc": true,                                                    │
│                    │     "transaction_replay": false,                                             │
│                    │     "transaction_replay_attempts": 5,                                        │
│                    │     "transaction_replay_max_size": 1073741824,                               │
│                    │     "transaction_replay_retry_on_deadlock": false,                           │
│                    │     "type": "service",                                                       │
│                    │     "use_sql_variables_in": "all",                                           │
```

```
|                      |     "user": "mxs",                                                 |
|                      |     "version_string": null                                         |
|                      | }                                                                  |
├──────────────────────┼────────────────────────────────────────────────────────────────────┤
| Router Diagnostics   | {                                                                  |
|                      |     "avg_sescmd_history_length": 0,                                |
|                      |     "max_sescmd_history_length": 0,                                |
|                      |     "queries": 1,                                                  |
|                      |     "replayed_transactions": 0,                                    |
|                      |     "ro_transactions": 0,                                          |
|                      |     "route_all": 0,                                                |
|                      |     "route_master": 0,                                             |
|                      |     "route_slave": 1,                                              |
|                      |     "rw_transactions": 0,                                          |
|                      |     "server_query_statistics": [                                   |
|                      |         {                                                          |
|                      |             "avg_selects_per_session": 0,                          |
|                      |             "avg_sess_duration": "0ns",                            |
|                      |             "id": "node2",                                         |
|                      |             "read": 1,                                             |
|                      |             "total": 1,                                            |
|                      |             "write": 0                                             |
|                      |         }                                                          |
|                      |     ]                                                              |
|                      | }                                                                  |
└──────────────────────┴────────────────────────────────────────────────────────────────────┘
```


Output should align to the Read Connection Router (readconnroute) or Read/Write Split Router (readwritesplit) configuration you performed.

Test Application User

Applications should use a dedicated user account. The user account must be created on the primary server.

When users connect to MaxScale, MaxScale authenticates the user connection before routing it to an Enterprise Server node. Enterprise Server authenticates the connection as originating from the IP address of the MaxScale node.

The application users must have one user account with the host IP address of the application server and a second user account with the host IP address of the MaxScale node.

The requirement of a duplicate user account can be avoided by enabling the `proxy_protocol` parameter for MaxScale and the proxy_protocol_networks for Enterprise Server.

Create a User to Connect from MaxScale

This action is performed on any Enterprise Cluster node:

1. Connect to the node:

```
$ sudo mariadb
```

2. Create the database user account for your MaxScale node:

```
CREATE USER 'app_user'@'192.0.2.104' IDENTIFIED BY 'app_user_passwd';
```

Replace 192.0.2.104 with the relevant IP address specification for your MaxScale node.

Passwords should meet your organization's password policies.

3. Grant the privileges required by your application to the database user account for your MaxScale node:

```
GRANT ALL ON test.* TO 'app_user'@'192.0.2.104';
```

The privileges shown are designed to allow the tests in the subsequent sections to work. The user account for your production application may require different privileges.

Create a User to Connect from the Application Server

This action is performed on any Enterprise Cluster node:

1. Create the database user account for your application server:

```
CREATE USER 'app_user'@'192.0.2.11' IDENTIFIED BY 'app_user_passwd';
```

Replace 192.0.2.11 with the relevant IP address specification for your application server.

Passwords should meet your organization's password policies.

2. Grant the privileges required by your application to the database user account for your application server:

```
GRANT ALL ON test.* TO 'app_user'@'192.0.2.11';
```

The privileges shown are designed to allow the tests in the subsequent sections to work. The user account for your production application may require different privileges.

Test Connection with Application User

To test the connection, use the MariaDB Client from your application server to connect to an Enterprise Cluster node through MaxScale.

This action is performed on the application server:

```
$ mariadb --host 192.0.2.104 --port 3307
      --user app_user --password
```

Test Connection with Read Connection Router

If you configured the Read Connection Router, confirm that MaxScale routes connections to the replica servers.

1. On the MaxScale node, use the maxctrl list listeners command to view the available listeners and ports:

```
$ maxctrl list listeners
```

```
┌──────────────────────────┬──────┬──────┬─────────┬──────────────────────────┐
│ Name                     │ Port │ Host │ State   │ Service                  │
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ connection_router_listener │ 3308 │ ::   │ Running │ connection_router_service │
├──────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ query_router_listener    │ 3307 │ ::   │ Running │ query_router_service     │
└──────────────────────────┴──────┴──────┴─────────┴──────────────────────────┘
```

2. Open multiple terminals connected to your application server, in each use MariaDB Client to connect to the listener port for the Read Connection Router (in the example 3308):

```
$ mariadb --host 192.0.2.104 --port 3308 \
      --user app_user --password
```

Use the application user credentials you created for the `--user` and `--password` options.

3. In each terminal, query the hostname system variable to identify to which you're connected:

```
SELECT @@global.hostname;

+-------------------+
| @@global.hostname |
+-------------------+
|             node2 |
+-------------------+
```

Different terminals should return different values since MaxScale routes the connections to different nodes.

Since the router was configured the slave router option, the Read Connection Router only routes connections to replica servers.

Test Write Queries with the Read/Write Split Router

If you configured the Read/Write Split Router, confirm that readwritesplit correctly routes write queries.

This action is performed with multiple client connections to the MaxScale node.

1. On the MaxScale node, use the maxctrl list servers command to identify the Enterprise Cluster node currently operating as the primary server:

```
$ maxctrl list servers
```

```
┌────────┬───────────────┬──────┬─────────────┬─────────────────────────┬──────┐
│ Server │ Address       │ Port │ Connections │ State                   │ GTID │
├────────┼───────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node1  │ 192.0.2.101 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼───────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node2  │ 192.0.2.102 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼───────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node3  │ 192.0.2.103 │ 3306 │ 0           │ Master, Synced, Running │      │
└────────┴───────────────┴──────┴─────────────┴─────────────────────────┴──────┘
```

The server listed as Master is currently operating as the primary server.

2. On the MaxScale node, use the maxctrl list listeners command to identify the correct listener port:

```
$ maxctrl list listeners galerarouter
```

```
┌─────────────────────────────┬──────┬──────┬─────────┬──────────────────────────────┐
│ Name                        │ Port │ Host │ State   │ Service                      │
├─────────────────────────────┼──────┼──────┼─────────┼──────────────────────────────┤
│ connection_router_listener │ 3308 │      │ Running │ connection_router_service │
│ query_router_listener      │ 3307 │      │ Running │ query_router_service      │
└─────────────────────────────┴──────┴──────┴─────────┴──────────────────────────────┘
```

In the example, the listener port for the Read/Write Split router is 3307.

3. Use the MariaDB Client to establish multiple connections to the listener configured for the Read/Write Split routing service, `query_router_listener`, on the MaxScale node:

```
$ mariadb --host=192.0.2.104 --port=3307 \
      --user=app_user --password=app_user_passwd
```

The database user account for your application server should be specified by the `--user` option.

4. Using any client connection, create a test table:

```
CREATE TABLE test.load_balancing_test (
   id INT PRIMARY KEY AUTO_INCREMENT,
   hostname VARCHAR(256)
);
```

5. Using each client connection, insert the values of the hostname system variable into the table using the INSERT statement to identify the node that executes the statement:

```
INSERT INTO test.load_balancing_test (hostname)
VALUES (@@global.hostname);
```

6. Using any client connection, query the table using the SELECT statement:

```
SELECT * FROM test.load_balancing_test;
```

```
+----+----------+
| id | hostname |
+----+----------+
|  1 | node3    |
|  4 | node3    |
|  7 | node3    |
+----+----------+
```

The output shows the hostname from the Enterprise Cluster node operating as the primary server. (Enterprise Cluster offsets auto-increment values by node to avoid write conflicts.)

Confirm that MaxScale is routing write queries to the Enterprise Cluster node operating as the primary server by checking that the test table only contains the hostname of the correct Enterprise Cluster node.

Test Read Queries with the Read/Write Split Router

If you configured the Read/Write Split Router, confirm that readwritesplit properly routes read queries to multiple replica servers.

This action is performed with multiple clients connected to the MaxScale node.

1. On the MaxScale node, use maxctrl list servers to identify the Enterprise Cluster nodes that are currently operating as replica servers:

```
$ maxctrl list servers
```

```
┌────────┬─────────────┬──────┬─────────────┬─────────────────────────┬──────┐
│ Server │ Address     │ Port │ Connections │ State                   │ GTID │
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node1  │ 192.0.2.101 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node2  │ 192.0.2.102 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node3  │ 192.0.2.103 │ 3306 │ 0           │ Master, Synced, Running │      │
└────────┴─────────────┴──────┴─────────────┴─────────────────────────┴──────┘
```

The servers listed as Slave are currently operating as replica servers.

2. On the MaxScale node, use the maxctrl list listeners command to identify the correct listener port:

```
$ maxctrl list listeners
```

```
┌──────────────────────────────┬──────┬──────┬─────────┬──────────────────────────┐
| Name                         | Port | Host | State   | Service                  |
├──────────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
| connection_router_listener | 3308 |      | Running | connection_router_service |
| query_router_listener      | 3307 |      | Running | query_router_service      |
└──────────────────────────────┴──────┴──────┴─────────┴──────────────────────────┘
```

In the example, the listener port for the Read/Write Split router is 3307.

3. Use the MariaDB Client to establish multiple connections to `query_router_listener` which is the listener configured for the Read/Write Split routing service on the MaxScale node:

```
$ mariadb --host=192.0.2.104 --port=3307 \
   --user=app_user --password=app_user_passwd
```

The database user account for your application server should be specified by the –user option.

4. Using each client connection, query the hostname system variable to identify the node that executes the statement:

```
SELECT @@global.hostname;

+-------------------+
| @@global.hostname |
+-------------------+
|             node2 |
+-------------------+
```

The output shows the hostname value from one of the Enterprise Cluster nodes operating as replica servers.

Confirm that MaxScale is routing read queries to multiple Enterprise Cluster nodes operating as replica servers by checking that different client connections return different hostname values.

For more information on different routing criteria, `slave_selection_criteria`

Next Step

Navigation in the procedure “Deploy Galera Cluster Topology”:

This page was step 6 of 6.

This procedure is complete.

This page is: Copyright © 2026 MariaDB. All rights reserved.

HTAP

Overview

Software Version	Diagram	Features
• Enterprise Server 10.5 • Enterprise Server 10.6 • Enterprise Server 11.4		**Single-stack hybrid transactional/analytical workloads** • ColumnStore for analytics with scalable S3-compatible object storage • InnoDB for transactions • Cross-engine JOINs • Enterprise Server 10.5, Enterprise ColumnStore 5, MaxScale 2.5 • Enterprise Server 10.6, Enterprise ColumnStore 23.02, MaxScale 22.08

This procedure describes the deployment of the **HTAP topology** with MariaDB Enterprise Server and MariaDB Enterprise ColumnStore.

MariaDB Enterprise ColumnStore is a columnar storage engine for MariaDB Enterprise Server. This topology is best suited for Hybrid Transactional-Analytical Processing (HTAP) workloads.

This procedure has 4 steps, which are executed in sequence.

This procedure represents the basic product capability and uses 1 Enterprise ColumnStore node.

This page provides an overview of the topology, requirements, and deployment procedures.

Please read and understand this procedure before executing.

Procedure Steps

Step	Description
Step 1	Prepare ColumnStore Node
Step 2	Install MariaDB Enterprise Server
Step 3	Start and Configure MariaDB Enterprise Server
Step 4	Test MariaDB Enterprise Server

Support

Customers can obtain support by submitting a support case.

Components

The following components are deployed during this procedure:

Component	Function
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.

MariaDB Enterprise Server Components

Component	Description
InnoDB	• General purpose storage engine • Support for Online Transactional Processing (OLTP) workloads • ACID-compliant • Performance
MariaDB Enterprise ColumnStore	• Columnar storage engine • Optimized for Online Analytical • Processing (OLAP) workloads • Scalable query execution

Topology

The MariaDB Enterprise ColumnStore HTAP topology is designed for hybrid transactional-analytical processing (HTAP) workloads.

The topology consists of:

- One MaxScale node
- One ColumnStore node running ES and Enterprise ColumnStore

The MaxScale node:

- Monitors the health and availability of the ColumnStore node using the MariaDB Monitor (mariadbmon)
- Accepts client and application connections

- Routes queries to the ColumnStore node using the Read/Write Split Router (readwritesplit)

The ColumnStore node:

- Receives queries from MaxScale
- Executes queries
- Uses a row-based storage engine, such as InnoDB to handle transactional queries
- Uses Enterprise ColumnStore as the columnar storage engine to handle analytical queries
- Uses cross-engine JOINs to join transactional and analytical tables
- Replicates data between engines using MariaDB Replication
- Optionally uses S3-compatible object storage for Enterprise ColumnStore data

Requirements

These requirements are for the HTAP topology when deployed with MariaDB Enterprise Server 11.4 and MariaDB Enterprise ColumnStore.

- Operating System
- S3-Compatible Object Storage Option
- Preferred Object Storage Providers: Cloud
- Preferred Object Storage Providers: Hardware

Operating System

In alignment to the enterprise lifecycle, the HTAP topology with MariaDB Enterprise Server 11.4 and MariaDB Enterprise ColumnStore is provided for:

- Debian 11 (x86_64, ARM64)
- Debian 12 (x86_64, ARM64)
- Red Hat Enterprise Linux 8 (x86_64, ARM64)
- Red Hat Enterprise Linux 9 (x86_64, ARM64)
- Rocky Linux 8 (x86_64, ARM64)
- Rocky Linux 9 (x86_64, ARM64)
- Ubuntu 20.04 LTS (x86_64, ARM64)
- Ubuntu 22.04 LTS (x86_64, ARM64)
- Ubuntu 24.04 LTS (x86_64, ARM64)

S3-Compatible Object Storage Option

The HTAP topology can use S3-compatible object storage to store ColumnStore data, but it is not required.

Many S3-compatible object storage services exist. MariaDB Corporation cannot make guarantees about all S3-compatible object storage services, because different services provide different functionality.

For the preferred S3-compatible object storage providers that provide cloud and hardware solutions, see the following sections:

- Cloud
- Hardware

The use of non-cloud and non-hardware providers is at your own risk.

If you have any questions about using specific S3-compatible object storage with MariaDB Enterprise ColumnStore, contact us.

Preferred Object Storage Providers: Cloud

- Amazon Web Services (AWS) S3
- Google Cloud Storage
- Azure Storage
- Alibaba Cloud Object Storage Service

Preferred Object Storage Providers: Hardware

- Cloudian HyperStore
- Dell EMC
- Seagate Lyve Rack
- Quantum ActiveScale
- IBM Cloud Object Storage

Limitations

Cross-Database Queries

This implementation relies on replicate_rewrite_db, so it does not support cross-database queries with statement-based replication.

For example, if the replicated database is selected by the USE, then the query will replicate properly:

```
USE innodb_db;

INSERT INTO htap_test1
   VALUES (100);
```

```
SELECT * FROM columnstore_db.htap_test1;

+------+
| id   |
+------+
|  100 |
+------+
```

However, if the replicated database is not selected, and it is instead prefixed the table name in the query, then the query will *not* replicate properly:

```
USE columnstore_db;

INSERT INTO innodb_db.htap_test1
   VALUES (200);
```

```
SELECT * FROM columnstore_db.htap_test1;

+------+
| id   |
+------+
|  100 |
+------+
```

Semi-Synchronous Replication

This implementation has not been tested with semi-synchronous replication.

Parallel Replication

This implementation has not been tested with parallel replication.

Row-Based Replication

This implementation requires the binlog_format system variable to be set to `STATEMENT`. Row-based replication is not currently supported.

Quick Reference

MariaDB Enterprise Server Configuration Management

Method	Description
Configuration File	Configuration files (such as /etc/my.cnf) can be used to set system-variables and options. The server must be restarted to apply changes made to configuration files.
Command-line	The server can be started with command-line options that set system-variables and options.
SQL	Users can set system-variables that support dynamic changes on-the-fly using the SET statement.

MariaDB Enterprise Server packages are configured to read configuration files from different paths, depending on the operating system. Making custom changes to Enterprise Server default configuration files is not recommended because custom changes may be overwritten by other default configuration files that are loaded later.

To ensure that your custom changes will be read last, create a custom configuration file with the z- prefix in one of the include directories.

Distribution	Example Configuration File Path
• CentOS • Red Hat Enterprise Linux (RHEL)	/etc/my.cnf.d/z-custom-mariadb.cnf
• Debian • Ubuntu	/etc/my.cnf.d/z-custom-mariadb.cnf

MariaDB Enterprise Server Service Management

The systemctl command is used to start and stop the MariaDB Enterprise Server service.

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

For additional information, see “Starting and Stopping MariaDB”.

MariaDB Enterprise Server Logs

MariaDB Enterprise Server produces log data that can be helpful in problem diagnosis.

Log filenames and locations may be overridden in the server configuration. The default location of logs is the data directory. The data directory is specified by the datadir system variable.

Log	System Variable/Option	Default Filename
MariaDB Error Log	log_error	`<hostname>.err`
MariaDB Enterprise Audit Log	server_audit_file_path	`server_audit.log`
Slow Query Log	slow_query_log_file	`<hostname>-slow.log`
General Query Log	general_log_file	`<hostname>.log`
Binary Log	log_bin	`<hostname>-bin`

Enterprise ColumnStore Service Management

The systemctl command is used to start and stop the ColumnStore service.

Operation	Command
Start	`sudo systemctl start mariadb-columnstore`
Stop	`sudo systemctl stop mariadb-columnstore`
Restart	`sudo systemctl restart mariadb-columnstore`
Enable during startup	`sudo systemctl enable mariadb-columnstore`
Disable during startup	`sudo systemctl disable mariadb-columnstore`
Status	`sudo systemctl status mariadb-columnstore`

Next Step

Navigation in the procedure “Deploy HTAP Topology”.

Next: Step 1: Prepare ColumnStore Node.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 1: Prepare ColumnStore Node

Overview

This page details step 1 of the 4-step procedure “Deploy HTAP Topology”.

This step prepares the system to host MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Optimize Linux Kernel Parameters

MariaDB Enterprise ColumnStore performs best with Linux kernel optimizations.

To optimize the kernel:

1. Set the relevant kernel parameters in a sysctl configuration file. To ensure proper change management, use an Enterprise ColumnStore-specific configuration file.

Create `a /etc/sysctl.d/90-mariadb-enterprise-columnstore.conf` file:

```
# minimize swapping
vm.swappiness = 1

# Increase the TCP max buffer size
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216

# Increase the TCP buffer limits
# min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

# don't cache ssthresh from previous connection
net.ipv4.tcp_no_metrics_save = 1

# for 1 GigE, increase this to 2500
# for 10 GigE, increase this to 30000
net.core.netdev_max_backlog = 2500
```

2. Use the sysctl command to set the kernel parameters at runtime

```
$ sudo sysctl --load=/etc/sysctl.d/90-mariadb-enterprise-columnstore.conf
```

Temporarily Configure Linux Security Modules (LSM)

The Linux Security Modules (LSM) should be temporarily disabled on each Enterprise ColumnStore node during installation.

The LSM will be configured and re-enabled later in this deployment procedure.

The steps to disable the LSM depend on the specific LSM used by the operating system.

CentOS / RHEL Stop SELinux

SELinux must be set to permissive mode before installing MariaDB Enterprise ColumnStore.

To set SELinux to permissive mode:

1. Set SELinux to permissive mode:

```
$ sudo setenforce permissive
```

2. Set SELinux to permissive mode by setting SELINUX=permissive in /etc/selinux/config.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. Confirm that SELinux is in permissive mode:

```
$ sudo getenforce
Permissive
```

SELinux will be configured and re-enabled later in this deployment procedure. This configuration is not persistent. If you restart the server before configuring and re-enabling SELinux later in the deployment procedure, you must reset the enforcement to permissive mode.

Debian / Ubuntu AppArmor

AppArmor must be disabled before installing MariaDB Enterprise ColumnStore.

1. Disable AppArmor:

```
$ sudo systemctl disable apparmor
```

2. Reboot the system.
3. Confirm that no AppArmor profiles are loaded using aa-status:

```
$ sudo aa-status

apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

AppArmor will be configured and re-enabled later in this deployment procedure.

Configure Character Encoding

When using MariaDB Enterprise ColumnStore, it is recommended to set the system's locale to UTF-8.

1. On RHEL 8, install additional dependencies:

```
$ sudo yum install glibc-locale-source glibc-langpack-en
```

2. Set the system's locale to en_US.UTF-8 by executing localedef:

```
$ sudo localedef -i en_US -f UTF-8 en_US.UTF-8
```

Create an S3 Bucket

With the HTAP topology, it is important to create the S3 bucket before you start ColumnStore.

If you already have an S3 bucket, confirm that the bucket is empty.

S3 bucket configuration will be performed later in this procedure.

Next Step

Navigation in the procedure "Deploy HTAP Topology".

This page was step 1 of 4.

Next: Step 2: Install MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 2: Install MariaDB Enterprise Server

Overview

This page details step 2 of the 4-step procedure "Deploy HTAP Topology".

This step installs MariaDB Enterprise Server MariaDB Enterprise ColumnStore 23.10, and dependencies.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Download Token

MariaDB Corporation provides package repositories for CentOS / RHEL (YUM) and Debian / Ubuntu (APT). A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to https://customers.mariadb.com/downloads/token/
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On each Enterprise ColumnStore node, install the prerequisites for downloading the software from the Web.
   Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

2. On each Enterprise ColumnStore node, configure package repositories and specify Enterprise Server:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-maxscale \
      --skip-tools \
      --mariadb-server-version="11.4"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install Enterprise Server and Enterprise ColumnStore

1. **On each Enterprise ColumnStore node**, install additional dependencies:
   Install on CentOS / RHEL (YUM)

```
$ sudo yum install epel-release

$ sudo yum install jemalloc
```

Install of Debian 10 and Ubuntu 20.04 (APT):

```
$ sudo apt install libjemalloc2
```

Install on Debian 9 and Ubuntu 18.04 (APT):

```
$ sudo apt install libjemalloc1
```

2. **On the Enterprise ColumnStore node**, install MariaDB Enterprise Server and MariaDB Enterprise ColumnStore:

Install on CentOS / RHEL (YUM):

```
$ sudo yum install MariaDB-server \
      MariaDB-backup \
      MariaDB-shared \
      MariaDB-client \
      MariaDB-columnstore-engine
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install mariadb-server \
      mariadb-backup \
      libmariadb3 \
      mariadb-client \
      mariadb-plugin-columnstore
```

Next Step

Navigation in the procedure “Deploy HTAP Topology”.

This page was step 2 of 4.

Next: Step 3: Start and Configure MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 3: Start and Configure MariaDB Enterprise Server

Overview

This page details step 3 of the 4-step procedure “Deploy HTAP Topology”.

This step starts and configures MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Stop the Enterprise ColumnStore Services

The installation process might have started some of the ColumnStore services. The services should be stopped prior to making configuration changes.

1. On each Enterprise ColumnStore node, stop the MariaDB Enterprise Server service:

```
$ sudo systemctl stop mariadb
```

2. On each Enterprise ColumnStore node, stop the MariaDB Enterprise ColumnStore service:

```
$ sudo systemctl stop mariadb-columnstore
```

3. On each Enterprise ColumnStore node, stop the CMAPI service:

```
$ sudo systemctl stop mariadb-columnstore-cmapi
```

Configure Enterprise ColumnStore

On each Enterprise ColumnStore node, configure Enterprise Server.

Connector	MariaDB Connector/R2DBC
binlog_do_db	Set this to the name of the database to replicate from InnoDB to ColumnStore.
binlog_format	Set this to STATEMENT for HTAP.
character_set_server	Set this system variable to utf8
collation_server	Set this system variable to utf8_general_ci
loose-columnstore_use_import_for_batchinsert	Set this system variable to ALWAYS to always use cpimport for LOAD DATA INFILE and INSERT…SELECT statements.
gtid_strict_mode	Set this system variable to ON.
log_bin	Set this option to the file you want to use for the Binary Log. Setting this option enables binary logging.
log_slave_updates	Set this system variable to ON.
replicate_rewrite_db	Set this option to the file you want to use for the Relay Logs. Setting this option enables relay logging.
replicate_wild_do_table	Set this option to the file you want to use to index Relay Log filenames.
server_id	Sets the numeric Server ID for this MariaDB Enterprise Server. The value set on this option must be unique to each node.

Example Configuration

```
[mariadb]
log_error                              = mariadbd.err
character_set_server                   = utf8
collation_server                       = utf8_general_ci

# Replication Configuration (HTAP Server)
server_id                      = 1
log_bin                        = mariadb-bin
binlog_format                  = STATEMENT
log_slave_updates              = OFF
loose-columnstore_replication_slave = ON

# HTAP filtering rules

# Transactions replicate from same server
replicate_same_server_id = ON

# Only write queries that touch 'innodb_db' to the binary log
binlog_do_db = innodb_db

# Rewrite innodb_db to columnstore_db prior to applying transaction
replicate_rewrite_db = innodb_db->columnstore_db

# Only replicate tables that begin with "htap"
replicate_wild_do_table = columnstore_db.htap%
```

> **Warning:**
>
> The `loose-` prefix is required for ColumnStore system variables in the configuration file. Without it, MariaDB Server will fail to start if the ColumnStore plugin is not installed or has been removed.

Configure the S3 Storage Manager

Configure Enterprise ColumnStore S3 Storage Manager to use S3-compatible storage by editing the `/etc/columnstore/storagemanager.cnf` configuration file:

```
[ObjectStorage]
…
service = S3
…
[S3]
bucket                = your_columnstore_bucket_name
endpoint              = your_s3_endpoint
aws_access_key_id     = your_s3_access_key_id
aws_secret_access_key = your_s3_secret_key
# iam_role_name       = your_iam_role
# sts_region          = your_sts_region
# sts_endpoint        = your_sts_endpoint
# ec2_iam_mode        = enabled

[Cache]
cache_size = your_local_cache_size
path       = your_local_cache_path
```

The S3-compatible object storage options are configured under [S3]:

- The bucket option must be set to the name of the bucket that you created in “Create an S3 Bucket”.
- The endpoint option must be set to the endpoint for the S3-compatible object storage.
- The `aws_access_key_id and aws_secret_access_key` options must be set to the access key ID and secret access key for the S3-compatible object storage.
- To use a specific IAM role, you must uncomment and set `iam_role_name, sts_region, and sts_endpoint`.
- To use the IAM role assigned to an EC2 instance, you must uncomment `ec2_iam_mode=enabled`.

The local cache options are configured under [Cache]:

- The `cache_size` option is set to 2 GB by default.
- The path option is set to `/var/lib/columnstore/storagemanager/cache` by default.
- Ensure that the specified path has sufficient storage space for the specified cache size.

Start the Enterprise ColumnStore Services

Start and enable the MariaDB Enterprise Server service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb

$ sudo systemctl enable mariadb
```

Start and enable the MariaDB Enterprise ColumnStore service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb-columnstore

$ sudo systemctl enable mariadb-columnstore
```

For additional information, see “Starting and Stopping MariaDB”.

Create User Accounts

The HTAP topology requires several user accounts.

Create the Utility User

Enterprise ColumnStore requires a mandatory utility user account. By default, it connects to the server using the root user with no password. MariaDB Enterprise Server 10.6 will reject this login attempt by default, so you will need to configure Enterprise ColumnStore to use a different user account and password and create this user account on Enterprise Server.

1. On the Enterprise ColumnStore node, create the user account with the CREATE USER statement:

```
CREATE USER 'util_user'@'127.0.0.1'
IDENTIFIED BY 'util_user_passwd';
```

2. On the Enterprise ColumnStore node, grant the user account SELECT privileges on all databases with the GRANT statement:

```
GRANT SELECT, PROCESS ON *.*
TO 'util_user'@'127.0.0.1';
```

3. Configure Enterprise ColumnStore to use the utility user:

```
$ sudo mcsSetConfig CrossEngineSupport Host 127.0.0.1

$ sudo mcsSetConfig CrossEngineSupport Port 3306

$ sudo mcsSetConfig CrossEngineSupport User util_user
```

4. Set the password:

```
$ sudo mcsSetConfig CrossEngineSupport Password util_user_passwd
```

For details about how to encrypt the password, see "Credentials Management for MariaDB Enterprise ColumnStore".

Passwords should meet your organization's password policies. If your MariaDB Enterprise Server instance has a password validation plugin installed, then the password should also meet the configured requirements.

Create the Replication User

Enterprise HTAP uses MariaDB Replication to replicate writes between InnoDB tables and ColumnStore tables.

Create a replication user and grant it the required privileges:

1. Use the CREATE USER statement to create replication users for each replica server:

```
CREATE USER 'repl'@'localhost' IDENTIFIED BY 'passwd';
```

2. Grant the user account several global privileges with the GRANT statement.

```
GRANT REPLICA MONITOR,
   REPLICATION REPLICA
ON *.* TO 'repl'@'localhost';
```

Configure MariaDB Replication

1. Set the GTID position by setting the gtid_slave_pos system variable.
   If this is a new deployment, then it would be set to the empty string:

```
SET GLOBAL gtid_slave_pos='';
```

2. Use the CHANGE MASTER TO statement to configure the server to replicate from itself starting from this position:

```
CHANGE MASTER TO
   MASTER_HOST='localhost',
   MASTER_USER='htap_replication',
   MASTER_PASSWORD='passwd',
   MASTER_USE_GTID=slave_pos;
```

3. Start replication using the START REPLICA statement:

```
START REPLICA;
```

4. Confirm that replication is working using the SHOW REPLICA STATUS statement:

```
SHOW REPLICA STATUS;
```

Configure Linux Security Modules (LSM)

The specific steps to configure the security module depend on the operating system.

Configure SELinux (CentOS, RHEL)

Configure SELinux for Enterprise ColumnStore:

1. To configure SELinux, you have to install the packages required for audit2allow.
   On CentOS 7 and RHEL 7, install the following:

```
$ sudo yum install policycoreutils policycoreutils-python
```

On RHEL 8, install the following:

```
$ sudo yum install policycoreutils python3-policycoreutils policycoreutils-python-utils
```

2. Allow the system to run under load for a while to generate SELinux audit events.
3. After the system has taken some load, generate an SELinux policy from the audit events using audit2allow:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local
```

If no audit events were found, this will print the following:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local

Nothing to do
```

4. If audit events were found, the new SELinux policy can be loaded using semodule:

```
$ sudo semodule -i mariadb_local.pp
```

5. Set SELinux to enforcing mode by setting SELINUX=enforcing in /etc/selinux/config.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

6. Set SELinux to enforcing mode:

```
$ sudo setenforce enforcing
```

Configure AppArmor (Ubuntu)

For information on how to create a profile, see How to create an AppArmor Profile on ubuntu.com.

Next Step

Navigation in the procedure “Deploy HTAP Topology”.

This page was step 3 of 4.

Next: Step 4: Test MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 4: Test MariaDB Enterprise Server

Overview

This page details step 4 of the 4-step procedure “Deploy HTAP Topology”.

This step tests Enterprise ColumnStore.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Test S3 Connection

MariaDB Enterprise ColumnStore includes a `testS3Connection` command to test the S3 configuration, permissions, and connectivity.

Test the S3 configuration by executing the following:

```
$ sudo testS3Connection
```

```
StorageManager[26887]: Using the config file found at /etc/columnstore/storagemanager.cnf
StorageManager[26887]: S3Storage: S3 connectivity & permissions are OK
S3 Storage Manager Configuration OK
```

If the testS3Connection command does not return OK, investigate the S3 configuration.

Test Enterprise Server Service

Use Systemd to test whether the MariaDB Enterprise Server service is running.

Check if the MariaDB Enterprise Server service is running by executing the following:

```
$ systemctl status mariadb
```

If the service is not running, start the service by executing the following:

```
$ sudo systemctl start mariadb
```

Test Local Client Connections

Use MariaDB Client to test the local connection to the Enterprise Server node:

```
$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The sudo command is used here to connect to the Enterprise Server node using the root@localhost user account, which authenticates using the unix_socket authentication plugin. Other user accounts can be used by specifying the –user and –password command-line options.

Test ColumnStore Storage Engine Plugin

Query the information_schema.PLUGINS table to confirm that the ColumnStore storage engine is loaded.

Execute the following query:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS
FROM information_schema.PLUGINS
WHERE PLUGIN_LIBRARY LIKE 'ha_columnstore%';

+---------------------+---------------+
| PLUGIN_NAME         | PLUGIN_STATUS |
+---------------------+---------------+
| Columnstore         | ACTIVE        |
| COLUMNSTORE_COLUMNS | ACTIVE        |
| COLUMNSTORE_TABLES  | ACTIVE        |
| COLUMNSTORE_FILES   | ACTIVE        |
| COLUMNSTORE_EXTENTS | ACTIVE        |
+---------------------+---------------+
```

The `PLUGIN_STATUS` column for each ColumnStore-related plugin should contain ACTIVE.

Test Replication Status

Use the SHOW REPLICA STATUS statement to check the status of MariaDB Replication:

```
SHOW REPLICA STATUS\G

*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: localhost
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mariadb-bin.000002
          Read_Master_Log_Pos: 695
               Relay_Log_File: mysqld-relay-bin.000002
                Relay_Log_Pos: 996
        Relay_Master_Log_File: mariadb-bin.000002
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table: columnstore_db.htap%
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 695
              Relay_Log_Space: 1306
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
               Master_SSL_Crl:
           Master_SSL_Crlpath:
                   Using_Gtid: Slave_Pos
                  Gtid_IO_Pos: 0-1-7
      Replicate_Do_Domain_Ids:
  Replicate_Ignore_Domain_Ids:
               Parallel_Mode: optimistic
                   SQL_Delay: 0
         SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
             Slave_DDL_Groups: 1
Slave_Non_Transactional_Groups: 0
    Slave_Transactional_Groups: 1
```

Test ColumnStore Table Creation

1. Create a test database, if it does not exist:

```
CREATE DATABASE IF NOT EXISTS test;
```

2. Create a ColumnStore table:

```
CREATE TABLE IF NOT EXISTS test.contacts (
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   email VARCHAR(100)
) ENGINE=ColumnStore;
```

3. Add sample data into the table:

```
INSERT INTO test.contacts (first_name, last_name, email)
   VALUES
   ("Kai", "Devi", "kai.devi@example.com"),
   ("Lee", "Wang", "lee.wang@example.com");
```

4. Read data from table:

```
SELECT * FROM test.contacts;

+------------+-----------+----------------------+
| first_name | last_name | email                |
+------------+-----------+----------------------+
| Kai        | Devi      | kai.devi@example.com |
| Lee        | Wang      | lee.wang@example.com |
+------------+-----------+----------------------+
```

Test Cross Engine Join

1. Create an InnoDB table:

```
CREATE TABLE test.addresses (
   email VARCHAR(100),
   street_address VARCHAR(255),
   city VARCHAR(100),
   state_code VARCHAR(2)
) ENGINE = InnoDB;
```

2. Add data to the table:

```
INSERT INTO test.addresses (email, street_address, city, state_code)
   VALUES
   ("kai.devi@example.com", "1660 Amphibious Blvd.", "Redwood City", "CA"),
   ("lee.wang@example.com", "32620 Little Blvd", "Redwood City", "CA");
```

3. Perform a cross-engine join:

```
SELECT name AS "Name", addr AS "Address"
FROM (SELECT CONCAT(first_name, " ", last_name) AS name,
   email FROM test.contacts) AS contacts
INNER JOIN (SELECT CONCAT(street_address, ", ", city, ", ", state_code) AS addr,
   email FROM test.addresses) AS addr
WHERE  contacts.email = addr.email;

+----------+----------------------------------------+
| Name     | Address                                |
+----------+----------------------------------------+
| Kai Devi | 1660 Amphibious Blvd., Redwood City, CA |
| Lee Wang | 32620 Little Blvd, Redwood City, CA     |
+----------+----------------------------------------+
```

Test HTAP Replication

1. Connect to the server using MariaDB Client using the root@localhost user account:

```
$ sudo mariadb
```

2. Create the databases for the InnoDB and ColumnStore tables using the CREATE DATABASE statement:

```
CREATE DATABASE columnstore_db;

CREATE DATABASE innodb_db;
```

3. Create the InnoDB versions of the HTAP tables using the CREATE TABLE statement:

```
USE innodb_db;

CREATE TABLE htap_test1 (
   id INT
) ENGINE = InnoDB;

CREATE TABLE htap_test2 (
   id INT
) ENGINE = InnoDB;
```

4. Confirm that the tables were replicated using the SHOW TABLES statement:

```
SHOW TABLES FROM columnstore_db;

+--------------------------+
| Tables_in_columnstore_db |
+--------------------------+
| htap_test1               |
| htap_test2               |
+--------------------------+
```

5. The replication initially creates empty InnoDB tables, which need to be transformed into ColumnStore tables and which need to be populated with the initial copy of the data:

```
DROP TABLE IF EXISTS columnstore_db.htap_test1;

CREATE TABLE columnstore_db.htap_test1
ENGINE=COLUMNSTORE
SELECT * FROM innodb_db.htap_test1;

DROP TABLE IF EXISTS columnstore_db.htap_test2;

CREATE TABLE columnstore_db.htap_test2
ENGINE=COLUMNSTORE
SELECT * FROM innodb_db.htap_test2;
```

6. Insert data into the InnoDB versions of the HTAP tables using the INSERT statement:

```
USE innodb_db;

INSERT INTO htap_test1
VALUES (100);

INSERT INTO htap_test2
VALUES (200);
```

7. Confirm that the data was replicated using the SELECT statement:

```
SELECT * FROM columnstore_db.htap_test1;

+------+
| id   |
+------+
|  100 |
+------+

SELECT * FROM columnstore_db.htap_test2;

+------+
| id   |
+------+
|  200 |
+------+
```

8. Create an InnoDB table that will not be replicated:

```
USE innodb_db;
CREATE TABLE transactional_test1 (
   id INT
) ENGINE = InnoDB;
```

9. Confirm that the table was not replicated:

```
SHOW TABLES FROM columnstore_db LIKE 'transactional_%';

Empty set (0.02 sec)
```

10. Create a ColumnStore table that will not be replicated:

```
USE columnstore_db;
CREATE TABLE analytical_test1 (
   id INT
) ENGINE = ColumnStore;
```

11. Confirm that the table was not replicated:

```
SHOW TABLES FROM innodb_db LIKE 'analytical_%';

Empty set (0.02 sec)
```

Next Step

Navigation in the procedure “Deploy HTAP Topology”.
This page was step 4 of 4.

This procedure is complete.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 1: Install MariaDB Enterprise Server

Overview

This page details step 1 of the 7-step procedure “Deploy Primary/Replica Topology”.

This step installs MariaDB Enterprise Server

The Primary/Replica topology requires 3 or more MariaDB Enterprise Server nodes for High Availability (HA). Nodes must meet requirements.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Download Token

MariaDB Corporation provides package repositories for CentOS / RHEL (YUM) and Debian / Ubuntu (APT). A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to https://customers.mariadb.com/downloads/token/
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On each Enterprise ColumnStore node, install the prerequisites for downloading the software from the Web. Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

2. On each Enterprise ColumnStore node, configure package repositories and specify Enterprise Server:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-maxscale \
      --skip-tools \
      --mariadb-server-version="11.4"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install Enterprise Server

1. On each Enterprise Cluster node, install MariaDB Enterprise Server and MariaDB Enterprise Backup.

Install via CentOS / RHEL (YUM):

```
$ sudo yum install MariaDB-server MariaDB-backup
```

Install via Debian / Ubuntu (APT):

```
$ sudo apt update

$ sudo apt install mariadb-server mariadb-backup
```

Install via SLES (ZYpp):

```
$ sudo zypper install MariaDB-server MariaDB-backup
```

Next Step

Navigation in the procedure “Deploy Primary/Replica Topology”:

This page was step 1 of 7.

Next: Step 2: Start and Configure MariaDB Enterprise Server on Primary Server

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 2: Start and Configure MariaDB Enterprise Server on Primary Server

Overview

This page details step 2 of the 7-step procedure “Deploy Primary/Replica Topology”.

This step starts and configures a MariaDB Enterprise Server to operate as a primary server in MariaDB Replication.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Stop the Enterprise Server Service

The installation process might have started the Enterprise Server service. The service should be stopped prior to making configuration changes.

Stop the MariaDB Enterprise Server service:

```
$ sudo systemctl stop mariadb
```

Configure Enterprise Server

Enterprise Server nodes require that you set the following system variables and options:

System Variable/Option	Description
bind_address	The network socket Enterprise Server listens on for incoming TCP/IP client connections. On Debian or Ubuntu, this system variable must be set to override the 127.0.0.1 default configuration.
log_bin	Set this option to the file you want to use for the Binary Log. Setting this option enables binary logging.
server_id	Sets the numeric Server ID for this MariaDB Enterprise Server. The value set on this option must be unique to each node.

MariaDB Enterprise Server also supports group commit.

Group Commit

Group commit can help performance by reducing I/O.

If you would like to configure parallel replication on replica servers, then you must also configure group commit on the primary server.

System Variable	Description
binlog_commit_wait_count	Sets the number of transactions that the server commits as a group to the binary log.
binlog_commit_wait_usec	Sets the number of microseconds that the server waits for transactions to group commit before it commits the current group.

Example Configuration

On each Enterprise Server node, edit a configuration file and set these system variables and options:

```
[mariadb]
bind_address = 0.0.0.0
log_bin      = mariadb-bin.log
server_id    = 1
```

Set the server_id option to a value that is unique for each Enterprise Server node.

Start Primary Server

Start MariaDB Enterprise Server. If the Enterprise Server process is already running, restart it to apply the changes from the configuration file.

```
$ systemctl start mariadb
```

For additional information, see "Starting and Stopping MariaDB".

Create User Accounts

The Primary/Replica topology requires several user accounts. Each user account should be created on the primary server, so that it is replicated to the replica servers.

Create the Replication User

Primary/Replica uses MariaDB Replication to replicate writes between the primary and replica servers. As MaxScale can promote a replica server to become a new primary in the event of node failure, all nodes must have a replication user.

The action is performed on the primary server.

Create the replication user and grant it the required privileges:

1. Use the CREATE USER statement to create replication user.

```
CREATE USER 'repl'@'192.0.2.%' IDENTIFIED BY 'repl_passwd';
```

Replace the referenced IP address with the relevant address for your environment.

Ensure that the user account can connect to the primary server from each replica.

2. Grant the user account the required privileges with the GRANT statement.

The following privileges are required:

```
GRANT REPLICATION SLAVE,
   REPLICATION CLIENT
ON *.* TO repl@'%';
```

Use this username and password for the `MASTER_USER and MASTER_PASSWORD` in the CHANGE MASTER TO statement when configuring replica servers in Step 3.

Create MaxScale User

Primary/Replica uses MariaDB MaxScale 25.01 to load balance between the nodes. MaxScale requires a database user to connect to the primary server when routing queries and to promote replicas in the event that the primary server fails.

This action is performed on the primary server.

1. Use the CREATE USER statement to create the MaxScale user:

```
CREATE USER 'mxs'@'192.0.2.%'
IDENTIFIED BY 'mxs_passwd';
```

Replace the referenced IP address with the relevant address for your environment.

Ensure that the user account can connect from the IP address of the MaxScale instance.

2. Use the GRANT statement to grant the privileges required by the router:

```
GRANT SHOW DATABASES ON *.* TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.columns_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.db TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.procs_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.proxies_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.roles_mapping TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.tables_priv TO 'mxs'@'192.0.2.%';

GRANT SELECT ON mysql.user TO 'mxs'@'192.0.2.%';
```

3. Use the GRANT statement to grant privileges required by the MariaDB Monitor.

The following privileges are required:

```
GRANT SUPER,
   REPLICATION CLIENT,
   RELOAD,
   PROCESS,
   SHOW DATABASES,
   EVENT
ON *.* TO 'mxs'@'192.0.2.%';
```

Next Step

Navigation in the procedure “Deploy Primary/Replica Topology”:

This page was step 2 of 7.

Next: Step 3: Start and Configure MariaDB Enterprise Server on Replica Servers

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 3: Start and Configure MariaDB Enterprise Server on Replica Servers

Overview

This page details step 3 of the 7-step procedure “Deploy Primary/Replica Topology”.

This page starts and configures a MariaDB Enterprise Server 11.4 to operate as a replica server in MariaDB Replication.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Stop the Enterprise Server Service

The installation process might have started the Enterprise Server service. The service should be stopped prior to making configuration changes.

On each Enterprise Server node, stop the MariaDB Enterprise Server service:

```
$ sudo systemctl stop mariadb
```

Configure Enterprise Server

Enterprise Server nodes require that you set the following system variables and options:

System Variable/Option	Description
bind_address	The network socket Enterprise Server listens on for incoming TCP/IP client connections. On Debian or Ubuntu, this system variable must be set to override the 127.0.0.1 default configuration.
log_bin	Enables binary logging and sets the name of the binlog file.
server_id	Unique numeric identifier for each Enterprise Server node.

MariaDB Enterprise Server also supports group commit.

Parallel Replication

Writes to the primary server that are group committed or logged with a Global Transaction ID in different replication domains can be applied on the replica server using parallel threads to improve performance.

System Variable/Option	Description
slave_parallel_threads	Sets the number of threads the replica server uses to apply replication events in parallel. Use a non-zero value to enable Parallel Replication.
slave_parallel_mode	Sets how the replica server applies replicated transactions.

Example Configuration

On each Enterprise Server node, edit a configuration file and set these system variables and options:

```
[mariadb]
bind_address = 0.0.0.0
log_bin      = mariadb-bin.log
server_id    = 1
```

Set the server_id option to a value that is unique for each Enterprise Server node.

Initialize Replica Database

When deploying a new replica server to an existing system, back up the primary server and restore it on the replica server to initialize the database.

Back up the Primary Server

Use MariaDB Backup to back up the primary server.

1. On the primary server, take a full backup:

```
$ sudo mariadb-backup --backup \
      --user=mariadb-backup_user \
      --password=mariadb-backup_passwd \
      --target-dir=/data/backup/replica_backup
```

Confirm successful completion of the backup operation.

2. On the primary server, prepare the backup:

```
$ sudo mariadb-backup --prepare \
      --target-dir=/data/backup/replica_backup
```

Confirm successful completion of the prepare operation.

Restore the Backup to the Replica Server

1. **On the primary server**, copy the backup directory to each replica server:

```
$ sudo rsync -av /data/backup/replica_backup 192.0.2.11:/data/backup/
```

2. **On the replica server**, move the default datadir to another location:

```
$ sudo mv /var/lib/mysql /var/lib/mysql_backup
```

3. **On the replica server**, use MariaDB Backup to restore the backup to the datadir:

```
$ sudo mariadb-backup --copy-back \
   --target-dir=/data/backup/replica_backup
```

4. **On the replica server**, set the file permissions for the datadir:

```
$ sudo chown -R mysql:mysql /var/lib/mysql
```

Start Replica Server

Start MariaDB Enterprise Server. If the Enterprise Server process is already running, restart it to apply the changes from the configuration file.

```
$ systemctl start mariadb
```

For additional information, see “Starting and Stopping MariaDB”.

Set the Global Transaction ID Position

If the replica server was restored from a backup of the primary, set the GTID position.

1. Get the GTID position that corresponds to the restored backup. This can be found in the xtrabackup_binlong_info file.

```
$ cat xtrabackup_binlog_info
mariadb-bin.000096 568 0-1-2001,1-2-5139
```

The GTID position from the above output is `0-1-2001,1-2-5139`.

2. Connect to the replica server:

```
$ sudo mariadb
```

Set the gtid_slave_pos system variable to the GTID position:

```
SET GLOBAL gtid_slave_pos='0-1-2001,1-2-5139';
```

Start Replication

1. Execute the CHANGE MASTER TO statement to configure the replica server to connect to the primary server at this position:

```
CHANGE MASTER TO
   MASTER_USER = "repl",
   MASTER_HOST = "192.0.2.10",
   MASTER_PASSWORD = "repl_passwd",
   MASTER_USE_GTID = slave_pos;
```

The above statement configures the replica server to connect to a primary server located at 192.0.2.10 using the repl user account. This account must first be configured on the primary server.

2. Use the START REPLICA statement to start replication:

```
START REPLICA;
```

3. Use SHOW REPLICA STATUS statement to confirm replication is running:

```
SHOW REPLICA STATUS\G

*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.0.2.10
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mariadb-bin.000001
          Read_Master_Log_Pos: 645
               Relay_Log_File: li282-189-relay-bin.000002
                Relay_Log_Pos: 946
        Relay_Master_Log_File: mariadb-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 645
              Relay_Log_Space: 1259
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
               Master_SSL_Crl:
           Master_SSL_Crlpath:
                   Using_Gtid: Slave_Pos
                  Gtid_IO_Pos: 0-1-2
      Replicate_Do_Domain_Ids:
  Replicate_Ignore_Domain_Ids:
               Parallel_Mode: optimistic
                   SQL_Delay: 0
         SQL_Remaining_Delay: NULL
     Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
            Slave_DDL_Groups: 2
Slave_Non_Transactional_Groups: 0
    Slave_Transactional_Groups: 0
```

Next Step

Navigation in the procedure “Deploy Primary/Replica Topology”:

This page was step 3 of 7.

Next: Step 4: Test MariaDB Enterprise Server

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 4: Test MariaDB Enterprise Server

Overview

This page details step 4 of the 7-step procedure "Deploy Primary/Replica Topology".

This step tests MariaDB Enterprise Server

Several actions require connection to MariaDB Enterprise Server. A command-line client (mariadb) was included with your ES installation. These instructions describe connection via Unix domain socket. Alternatively, a different client and connection method could be used.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Test Enterprise Server Service

Use Systemd to test whether the MariaDB Enterprise Server service is running.

This action is performed **on each Enterprise Server node**.

Check if the MariaDB Enterprise Server service is running by executing the following:

```
$ systemctl status mariadb
```

If the service is not running on any node, start the service by executing the following on that node:

```
$ sudo systemctl start mariadb
```

Test Local Client Connections

Use mariadb Client to test the local connection to the Enterprise Server node.

This action is performed **on each Enterprise Server node**:

```
$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The sudo command is used here to connect to the Enterprise Server node using the root@localhost user account, which authenticates using the unix_socket authentication plugin. Other user accounts can be used by specifying the `--user` and `--password` command-line options.

Test Replica Status

Use SHOW REPLICA STATUS to check that replication is running properly on the replica servers.

This action is performed **on each replica server**.

Execute the following:

```
SHOW REPLICA STATUS\G

*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.0.2.1
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mariadb-bin.000001
          Read_Master_Log_Pos: 645
               Relay_Log_File: li282-189-relay-bin.000002
                Relay_Log_Pos: 946
        Relay_Master_Log_File: mariadb-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 645
              Relay_Log_Space: 1259
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
               Master_SSL_Crl:
           Master_SSL_Crlpath:
                   Using_Gtid: Slave_Pos
                  Gtid_IO_Pos: 0-1-2
      Replicate_Do_Domain_Ids:
  Replicate_Ignore_Domain_Ids:
               Parallel_Mode: optimistic
                   SQL_Delay: 0
         SQL_Remaining_Delay: NULL
     Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
            Slave_DDL_Groups: 2
Slave_Non_Transactional_Groups: 0
    Slave_Transactional_Groups: 0
```

If `Slave_IO_Running` column is not `Yes` on any replica server, then check:

- The network connectivity between the replica server and the primary server
- The `Last_IO_Error` column for details on any errors

If `Slave_SQL_Running` column is not Yes on any replica server, then check:

- The GTID position in gtid_slave_pos
- The `Last_SQL_Error` column for details on any errors

If both columns are not `Yes` on any replica server, then check:

- The replication configuration on the replica server.

If you need to make any corrections, the slave threads can be restarted with START REPLICA.

Test DDL

Use mariadb Client to test DDL.

1. **On the primary server**, use the mariadb Client to connect to the node:

```
$ sudo mariadb
```

2. Create a test database and table:

```
CREATE DATABASE IF NOT EXISTS test;

CREATE TABLE test.contacts (
   id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   email VARCHAR(100)
);
```

3. **On each replica server**, use the mariadb Client to connect to the node:

```
$ sudo mariadb
```

4. Confirm that the database and table exist:

```
SHOW CREATE TABLE test.contacts\G;
```

If the database or table do not exist on any node, then check the replication status on the node.

Test DML

Use mariadb Client to test DML.

1. **On the primary server**, use the MariaDB Client to connect to the node:

```
$ sudo mariadb
```

2. Insert sample data into the table created in the DDL test:

```
INSERT INTO test.contacts (first_name, last_name, email)
   VALUES
   ("Kai", "Devi", "kai.devi@example.com"),
   ("Lee", "Wang", "lee.wang@example.com");
```

3. **On each replica server**, use the mariadb Client to connect to the node:

```
$ sudo mariadb
```

4. Execute a SELECT query to retrieve the data:

```
SELECT * FROM test.contacts;

+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
| 1  | Kai        | Devi      | kai.devi@example.com |
| 2  | Lee        | Wang      | lee.wang@example.com |
+----+------------+-----------+----------------------+
```

If the data is not returned on any node, then check the replication status on the node.

Next Step

Navigation in the procedure “Deploy Primary/Replica Topology”:

This page was step 4 of 7.

Next: Step 5: Install MariaDB MaxScale

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 5: Install MariaDB MaxScale

Overview

This page details step 5 of the 7-step procedure “Deploy Primary/Replica Topology”.

This step install MariaDB MaxScale 25.01.

The MaxScale node must meet requirements.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Customer Download Token

MariaDB Corporation provides package repositories for YUM (RHEL, CentOS), APT (Debian, Ubuntu), and ZYpp (SLES). A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to https://customers.mariadb.com/downloads/token/
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. **On the MaxScale node**, install the prerequisites for downloading the software from the Web.

Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

Install on SLES (ZYpp):

```
$ sudo zypper install curl
```

2. **On the MaxScale node**, configure package repositories and specify MariaDB MaxScale 25.01:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup

$ echo "${checksum}  mariadb_es_repo_setup" \
       | sha256sum -c -

$ chmod +x mariadb_es_repo_setup

$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-server \
      --skip-tools \
      --mariadb-maxscale-version="25.01"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install MaxScale

On the MaxScale node, install MariaDB MaxScale.

Install on CentOS / RHEL (YUM):

```
$ sudo yum install maxscale
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install maxscale
```

Install on SLES (ZYpp):

```
$ sudo zypper install maxscale
```

Next Step

Navigation in the procedure “Deploy Primary/Replica Topology”:

This page was step 5 of 7.

Next: Step 6: Start and Configure MariaDB MaxScale

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 6: Start and Configure MariaDB MaxScale

Overview

This page details step 6 of the 7-step procedure “Deploy Primary/Replica Topology”.

This step starts and configures MariaDB MaxScale.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Replace the Default Configuration File

MariaDB MaxScale installations include a configuration file with some example objects. This configuration file should be replaced.

On the MaxScale node, replace the default `/etc/maxscale.cnf` with the following configuration:

```
[maxscale]
threads          = auto
admin_host       = 0.0.0.0
admin_secure_gui = false
```

For additional information, see "Global Settings".

Restart MaxScale

On the MaxScale node, restart the MaxScale service to ensure that MaxScale picks up the new configuration:

```
$ sudo systemctl restart maxscale
```

For additional information, see "Starting and Stopping MariaDB".

Configure Server Objects

On the MaxScale node, use maxctrl create server to create a server object for each MariaDB Enterprise Server:

```
$ maxctrl create server node1 192.0.2.101

$ maxctrl create server node2 192.0.2.102

$ maxctrl create server node3 192.0.2.103
```

Configure MariaDB Monitor

MaxScale uses monitors to retrieve additional information from the servers. This information is used by other services in filtering and routing connections based on the current state of the node. For MariaDB Replication, use the MariaDB Monitor (mariadbmon).

On the MaxScale node, use maxctrl create monitor to create a MariaDB Monitor:

```
$ maxctrl create monitor mdb_monitor mariadbmon \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     replication_user=repl \
     replication_password='REPLICATION_USER_PASSWORD' \
     --servers node1 node2 node3
```

In this example:

- `mdb_monitor` is an arbitrary name that is used to identify the new monitor.
- mariadbmon is the name of the module that implements the MariaDB Monitor.
- `user=MAXSCALE_USER` sets the `user` parameter to the database user account that MaxScale uses to monitor the ES nodes.
- `password='MAXSCALE_USER_PASSWORD'` sets the `password` parameter to the password used by the database user account that MaxScale uses to monitor the ES nodes.
- `replication_user=REPLICATION_USER` sets the `replication_user` parameter to the database user account that MaxScale uses to setup replication.
- `replication_password='REPLICATION_USER_PASSWORD'` sets the `replication_password` parameter to the password used by the database user account that MaxScale uses to setup replication.
- `--servers` sets the `servers` parameter to the set of nodes that MaxScale should monitor. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by `mariadbmon` in MaxScale can also be specified.

Choose a MaxScale Router

Routers control how MaxScale balances the load between Enterprise Server nodes. Each router uses a different approach to routing queries. Consider the specific use case of your application and database load and select the router that best suits

Router	Configuration Procedure	Description
Read Connection (readconnroute)	Configure Read Connection Router	**Connection-based load balancing** • Routes connections to Enterprise ColumnStore nodes designated as replica servers for a read-only pool • Routes connections to an Enterprise ColumnStore node designated as the primary server for a read-write pool.
Read/Write Split (readwritesplit)	Configure Read/Write Split	**Query-based load balancing** • Routes write queries to an Enterprise ColumnStore node designated as the primary server • Routes read queries to Enterprise ColumnStore node designated as replica servers • Automatically reconnects after node failures • Automatically replays transactions after node failures • Optionally enforces causal reads

Configure Read Connection Router

Use MaxScale Read Connection Router (readconnroute) to route connections to replica servers for a read-only pool.

On the MaxScale node, use `maxctrl create service` to create a router:

```
$ maxctrl create service connection_router_service readconnroute \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     router_options=slave \
     --servers node1 node2 node3
```

In this example:

- `connection_router_service` is an arbitrary name that is used to identify the new service.
- `readconnroute` is the name of the module that implements the Read Connection Router.
- `user=MAXSCALE_USER` sets the `user` parameter to the database user account that MaxScale uses to connect to the ES nodes.
- `password=MAXSCALE_USER_PASSWORD` sets the password parameter to the password used by the database user account that MaxScale uses to connect to the ES nodes.
- `router_options=slave` sets the `router_options` parameter to `slave`, so that MaxScale only routes connections to the replica nodes.
- `--servers sets the servers` parameter to the set of nodes to which MaxScale should route connections. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by `readconnroute` in MaxScale 25.01 can also be specified.

Configure Listener for the Read Connection Router

These instructions reference TCP port 3308. You can use a different TCP port. The TCP port used must not be bound by any other listener.

On the MaxScale node, use the maxctrl create listener command to configure MaxScale to use a listener for the Read Connection Router (readconnroute):

```
$ maxctrl create listener connection_router_service connection_router_listener 3308 \
     protocol=MariaDBClient
```

In this example:

- `connection_router_service` is the name of the readconnroute service that was previously created.
- `connection_router_listener` is an arbitrary name that is used to identify the new listener.
- `3308` is the TCP port.
- `protocol=MariaDBClient` sets the protocol parameter.
- Other Module Parameters supported by listeners in MaxScale can also be specified.

Configure Read/Write Split Router for Queries

MaxScale Read/Write Split Router (readwritesplit) performs query-based load balancing. The router routes write queries to the primary and read queries to the replicas.

On the MaxScale node, use the maxctrl create service command to configure MaxScale to use the Read/Write Split Router (readwritesplit):

```
$ maxctrl create service query_router_service readwritesplit  \
     user=mxs \
     password='MAXSCALE_USER_PASSWORD' \
     --servers node1 node2 node3
```

In this example:

- `query_router_service` is an arbitrary name that is used to identify the new service.
- `readwritesplit` is the name of the module that implements the Read/Write Split Router.
- `user=MAXSCALE_USER` sets the `user` parameter to the database user account that MaxScale uses to connect to the ES nodes.
- `password=MAXSCALE_USER_PASSWORD` sets the `password` parameter to the password used by the database user account that MaxScale uses to connect to the ES nodes.
- `--servers sets the servers` parameter to the set of nodes to which MaxScale should route queries. All non-option arguments after `--servers` are interpreted as server names.
- Other Module Parameters supported by `readwritesplit` in MaxScale can also be specified.

Configure a Listener for the Read/Write Split Router

These instructions reference TCP port `3307`. You can use a different TCP port. The TCP port used must not be bound by any other listener.

On the MaxScale node, use the `maxctrl create listener` command to configure MaxScale to use a listener for the Read/Write Split Router (readwritesplit):

```
$ maxctrl create listener query_router_service query_router_listener 3307 \
     protocol=MariaDBClient
```

In this example:

- `query_router_service` is the name of the `readwritesplit` service that was previously created.
- `query_router_listener` is an arbitrary name that is used to identify the new listener.
- 3307 is the TCP port.

- `protocol=MariaDBClient` sets the `protocol` parameter.
- Other Module Parameters supported by listeners in MaxScale can also be specified.

Start Services

To start the services and monitors, on the MaxScale node use maxctrl start services:

```
$ maxctrl start services
```

Next Step

Navigation in the procedure “Deploy Primary/Replica Topology”:

This page was step 6 of 7.

Next: Step 7: Test MariaDB MaxScale

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 7: Test MariaDB MaxScale

Overview

This page details step 7 of the 7-step procedure “Deploy Primary/Replica Topology”.

This step tests MariaDB MaxScale.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Check Global Configuration

Use maxctrl show maxscale command to view the global MaxScale configuration.

This action is performed on the MaxScale node:

```
$ maxctrl show maxscale
```

```
┌──────────────┬──────────────────────────────────────────────────────────────┐
│ Version      │ 25.01.2                                                      │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Commit       │ 3761fa7a52046bc58faad8b5a139116f9e33364c                     │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Started At   │ Thu, 05 Aug 2021 20:21:20 GMT                                │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Activated At │ Thu, 05 Aug 2021 20:21:20 GMT                                │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Uptime       │ 868                                                          │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Config Sync  │ null                                                         │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Parameters   │ {                                                            │
│              │     "admin_auth": true,                                      │
│              │     "admin_enabled": true,                                   │
│              │     "admin_gui": true,                                       │
│              │     "admin_host": "0.0.0.0",                                 │
│              │     "admin_log_auth_failures": true,                         │
│              │     "admin_pam_readonly_service": null,                      │
│              │     "admin_pam_readwrite_service": null,                     │
│              │     "admin_port": 8989,                                      │
│              │     "admin_secure_gui": false,                               │
│              │     "admin_ssl_ca_cert": null,                               │
│              │     "admin_ssl_cert": null,                                  │
│              │     "admin_ssl_key": null,                                   │
│              │     "admin_ssl_version": "MAX",                              │
│              │     "auth_connect_timeout": "10000ms",                       │
│              │     "auth_read_timeout": "10000ms",                          │
│              │     "auth_write_timeout": "10000ms",                         │
│              │     "cachedir": "/var/cache/maxscale",                       │
│              │     "config_sync_cluster": null,                             │
│              │     "config_sync_interval": "5000ms",                        │
│              │     "config_sync_password": "******",                        │
│              │     "config_sync_timeout": "10000ms",                        │
│              │     "config_sync_user": null,                                │
│              │     "connector_plugindir": "/usr/lib64/mysql/plugin",        │
│              │     "datadir": "/var/lib/maxscale",                          │
│              │     "debug": null,                                           │
│              │     "dump_last_statements": "never",                         │
│              │     "execdir": "/usr/bin",                                   │
│              │     "language": "/var/lib/maxscale",                         │
│              │     "libdir": "/usr/lib64/maxscale",                         │
│              │     "load_persisted_configs": true,                          │
│              │     "local_address": null,                                   │
│              │     "log_debug": false,                                      │
│              │     "log_info": false,                                       │
│              │     "log_notice": true,                                      │
│              │     "log_throttling": {                                      │
│              │         "count": 10,                                         │
│              │         "suppress": 10000,                                   │
│              │         "window": 1000                                       │
│              │     },                                                       │
│              │     "log_warn_super_user": false,                            │
│              │     "log_warning": true,                                     │
│              │     "logdir": "/var/log/maxscale",                           │
│              │     "max_auth_errors_until_block": 10,                       │
│              │     "maxlog": true,                                          │
│              │     "module_configdir": "/etc/maxscale.modules.d",           │
│              │     "ms_timestamp": false,                                   │
│              │     "passive": false,                                        │
│              │     "persistdir": "/var/lib/maxscale/maxscale.cnf.d",        │
│              │     "piddir": "/var/run/maxscale",                           │
│              │     "query_classifier": "qc_sqlite",                         │
│              │     "query_classifier_args": null,                           │
│              │     "query_classifier_cache_size": 289073971,                │
│              │     "query_retries": 1,                                      │
│              │     "query_retry_timeout": "5000ms",                         │
```

```
│              │     "rebalance_period": "0ms",                          │
│              │     "rebalance_threshold": 20,                          │
│              │     "rebalance_window": 10,                             │
│              │     "retain_last_statements": 0,                        │
│              │     "session_trace": 0,                                 │
│              │     "skip_permission_checks": false,                    │
│              │     "sql_mode": "default",                              │
│              │     "syslog": true,                                     │
│              │     "threads": 1,                                       │
│              │     "users_refresh_interval": "0ms",                    │
│              │     "users_refresh_time": "30000ms",                    │
│              │     "writeq_high_water": 16777216,                      │
│              │     "writeq_low_water": 8192                            │
│              │ }                                                       │
└──────────────┴─────────────────────────────────────────────────────────┘
```

Output should align to the global MaxScale configuration in the new configuration file you created.

Check Server Configuration

Use the maxctrl list servers and maxctrl show server commands to view the configured server objects.

This action is performed on the MaxScale node:

1. Obtain the full list of servers objects:

```
$ maxctrl list servers
```

```
┌────────┬─────────────┬──────┬─────────────┬─────────────────────────┬──────┐
│ Server │ Address     │ Port │ Connections │ State                   │ GTID │
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node1  │ 192.0.2.101 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node2  │ 192.0.2.102 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node3  │ 192.0.2.103 │ 3306 │ 0           │ Master, Synced, Running │      │
└────────┴─────────────┴──────┴─────────────┴─────────────────────────┴──────┘
```

2. For each server object, view the configuration:

```
$ maxctrl show server node3
```

```
┌──────────────────┬──────────────────────────────────────────────────────────┐
│ Server           │ node3                                                    │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Address          │ 192.0.2.103                                              │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Port             │ 3306                                                     │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ State            │ Master, Synced, Running                                  │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Version          │ 11.4.5-3-MariaDB-enterprise-log                          │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Last Event       │ master_up                                                │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Triggered At     │ Thu, 05 Aug 2021 20:22:26 GMT                            │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Services         │ connection_router_service                                │
│                  │ query_router_service                                     │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Monitors         │ cluster_monitor                                          │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Master ID        │ -1                                                       │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Node ID          │ 1                                                        │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Slave Server IDs │                                                          │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Current Connections │ 1                                                     │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Total Connections │ 1                                                       │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Max Connections  │ 1                                                        │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Statistics       │ {                                                        │
│                  │     "active_operations": 0,                              │
│                  │     "adaptive_avg_select_time": "0ns",                   │
│                  │     "connection_pool_empty": 0,                          │
│                  │     "connections": 1,                                    │
│                  │     "max_connections": 1,                                │
│                  │     "max_pool_size": 0,                                  │
│                  │     "persistent_connections": 0,                         │
│                  │     "reused_connections": 0,                             │
│                  │     "routed_packets": 0,                                 │
│                  │     "total_connections": 1                               │
│                  │ }                                                        │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Parameters       │ {                                                        │
│                  │     "address": "192.0.2.103",                            │
│                  │     "disk_space_threshold": null,                        │
│                  │     "extra_port": 0,                                     │
│                  │     "monitorpw": null,                                   │
│                  │     "monitoruser": null,                                 │
│                  │     "persistmaxtime": "0ms",                             │
│                  │     "persistpoolmax": 0,                                 │
│                  │     "port": 3306,                                        │
│                  │     "priority": 0,                                       │
│                  │     "proxy_protocol": false,                             │
│                  │     "rank": "primary",                                   │
│                  │     "socket": null,                                      │
│                  │     "ssl": false,                                        │
│                  │     "ssl_ca_cert": null,                                 │
│                  │     "ssl_cert": null,                                    │
│                  │     "ssl_cert_verify_depth": 9,                          │
│                  │     "ssl_cipher": null,                                  │
│                  │     "ssl_key": null,                                     │
│                  │     "ssl_verify_peer_certificate": false,                │
│                  │     "ssl_verify_peer_host": false,                       │
│                  │     "ssl_version": "MAX"                                 │
```

```
│                     │ }                                                   │
└─────────────────────┴─────────────────────────────────────────────────────┘
```

Output should align to the Server Object configuration you performed.

Check Monitor Configuration

Use the maxctrl list monitors and maxctrl show monitor commands to view the configured monitors.

This action is performed on the MaxScale node:

1. Obtain the full list of monitors:

```
$ maxctrl list monitors
```

```
┌─────────────────┬─────────┬──────────────────────┐
│ Monitor         │ State   │ Servers              │
├─────────────────┼─────────┼──────────────────────┤
│ cluster_monitor │ Running │ node1, node2, node3 │
└─────────────────┴─────────┴──────────────────────┘
```

2. For each monitor, view the monitor configuration:

```
$ maxctrl show monitor cluster_monitor
```

```
┌──────────────────┬──────────────────────────────────────────────────────────┐
│ Monitor          │ cluster_monitor                                          │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Module           │ galeramon                                                │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ State            │ Running                                                  │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Servers          │ node1                                                    │
│                  │ node2                                                    │
│                  │ node3                                                    │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Parameters       │ {                                                        │
│                  │     ..                                                   │
│                  │ }                                                        │
├──────────────────┼──────────────────────────────────────────────────────────┤
│ Monitor Diagnostics │ {                                                     │
│                  │     ..                                                   │
│                  │ }                                                        │
└──────────────────┴──────────────────────────────────────────────────────────┘
```

Output should align to the Galera Monitor (galeramon) configuration you performed.

Check Service Configuration

Use the maxctrl list services and maxctrl show service commands to view the configured routing services.

This action is performed on the MaxScale node:

1. Obtain the full list of routing services:

```
$ maxctrl list services
```

```
┌──────────────────────────┬──────────────┬─────────────┬──────────────────┬──────────────────────┐
│ Service                  │ Router       │ Connections │ Total Connections │ Servers              │
├──────────────────────────┼──────────────┼─────────────┼──────────────────┼──────────────────────┤
│ connection_router_service │ readconnroute │ 0           │ 0                │ node1, node2, node3 │
├──────────────────────────┼──────────────┼─────────────┼──────────────────┼──────────────────────┤
│ query_router_service     │ readwritesplit │ 1           │ 1                │ node1, node2, node3 │
└──────────────────────────┴──────────────┴─────────────┴──────────────────┴──────────────────────┘
```

2. For each service, view the service configuration:

```
$ maxctrl show service query_router_service
```

```
┌────────────────────┬──────────────────────────────────────────────────────────────────────────────┐
│ Service            │ query_router_service                                                         │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Router             │ readwritesplit                                                               │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ State              │ Started                                                                      │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Started At         │ Thu Aug  5 20:23:38 2021                                                     │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Current Connections │ 1                                                                           │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Total Connections  │ 1                                                                            │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Max Connections    │ 1                                                                            │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Cluster            │                                                                              │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Servers            │ node1                                                                        │
│                    │ node2                                                                        │
│                    │ node3                                                                        │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Services           │                                                                              │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Filters            │                                                                              │
├────────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ Parameters         │ {                                                                            │
│                    │     "auth_all_servers": false,                                               │
│                    │     "causal_reads": "false",                                                 │
│                    │     "causal_reads_timeout": "10000ms",                                       │
│                    │     "connection_keepalive": "300000ms",                                      │
│                    │     "connection_timeout": "0ms",                                             │
│                    │     "delayed_retry": false,                                                  │
│                    │     "delayed_retry_timeout": "10000ms",                                      │
│                    │     "disable_sescmd_history": false,                                         │
│                    │     "enable_root_user": false,                                               │
│                    │     "idle_session_pool_time": "-1000ms",                                     │
│                    │     "lazy_connect": false,                                                   │
│                    │     "localhost_match_wildcard_host": true,                                   │
│                    │     "log_auth_warnings": true,                                               │
│                    │     "master_accept_reads": false,                                            │
│                    │     "master_failure_mode": "fail_instantly",                                 │
│                    │     "master_reconnection": false,                                            │
│                    │     "max_connections": 0,                                                    │
│                    │     "max_sescmd_history": 50,                                                │
│                    │     "max_slave_connections": 255,                                            │
│                    │     "max_slave_replication_lag": "0ms",                                      │
│                    │     "net_write_timeout": "0ms",                                              │
│                    │     "optimistic_trx": false,                                                 │
│                    │     "password": "******",                                                    │
│                    │     "prune_sescmd_history": true,                                            │
│                    │     "rank": "primary",                                                       │
│                    │     "retain_last_statements": -1,                                            │
│                    │     "retry_failed_reads": true,                                              │
│                    │     "reuse_prepared_statements": false,                                      │
│                    │     "router": "readwritesplit",                                              │
│                    │     "session_trace": false,                                                  │
│                    │     "session_track_trx_state": false,                                        │
│                    │     "slave_connections": 255,                                                │
│                    │     "slave_selection_criteria": "LEAST_CURRENT_OPERATIONS",                  │
│                    │     "strict_multi_stmt": false,                                              │
│                    │     "strict_sp_calls": false,                                                │
│                    │     "strip_db_esc": true,                                                    │
│                    │     "transaction_replay": false,                                             │
│                    │     "transaction_replay_attempts": 5,                                        │
│                    │     "transaction_replay_max_size": 1073741824,                               │
│                    │     "transaction_replay_retry_on_deadlock": false,                           │
│                    │     "type": "service",                                                       │
│                    │     "use_sql_variables_in": "all",                                           │
```

```
|                      |     "user": "mxs",                                                  |
|                      |     "version_string": null                                          |
|                      | }                                                                   |
├──────────────────────┼─────────────────────────────────────────────────────────────────────┤
| Router Diagnostics   | {                                                                   |
|                      |     "avg_sescmd_history_length": 0,                                 |
|                      |     "max_sescmd_history_length": 0,                                 |
|                      |     "queries": 1,                                                   |
|                      |     "replayed_transactions": 0,                                     |
|                      |     "ro_transactions": 0,                                           |
|                      |     "route_all": 0,                                                 |
|                      |     "route_master": 0,                                              |
|                      |     "route_slave": 1,                                               |
|                      |     "rw_transactions": 0,                                           |
|                      |     "server_query_statistics": [                                    |
|                      |         {                                                           |
|                      |             "avg_selects_per_session": 0,                           |
|                      |             "avg_sess_duration": "0ns",                             |
|                      |             "id": "node2",                                          |
|                      |             "read": 1,                                              |
|                      |             "total": 1,                                             |
|                      |             "write": 0                                              |
|                      |         }                                                           |
|                      |     ]                                                               |
|                      | }                                                                   |
└──────────────────────┴─────────────────────────────────────────────────────────────────────┘
```

Output should align to the Read Connection Router (readconnroute) or Read/Write Split Router (readwritesplit) configuration you performed.

Test Application User

Applications should use a dedicated user account. The user account must be created on the primary server.

When users connect to MaxScale, MaxScale authenticates the user connection before routing it to an Enterprise Server node. Enterprise Server authenticates the connection as originating from the IP address of the MaxScale node.

The application users must have one user account with the host IP address of the application server and a second user account with the host IP address of the MaxScale node.

The requirement of a duplicate user account can be avoided by enabling the `proxy_protocol` parameter for MaxScale and the proxy_protocol_networks for Enterprise Server.

Create a User to Connect from MaxScale

This action is performed on any Enterprise Cluster node:

1. Connect to the node:

```
$ sudo mariadb
```

2. Create the database user account for your MaxScale node:

```
CREATE USER 'app_user'@'192.0.2.104' IDENTIFIED BY 'app_user_passwd';
```

Replace 192.0.2.104 with the relevant IP address specification for your MaxScale node.

Passwords should meet your organization's password policies.

3. Grant the privileges required by your application to the database user account for your MaxScale node:

```
GRANT ALL ON test.* TO 'app_user'@'192.0.2.104';
```

The privileges shown are designed to allow the tests in the subsequent sections to work. The user account for your production application may require different privileges.

Create a User to Connect from the Application Server

This action is performed on any Enterprise Cluster node:

1. Create the database user account for your application server:

```
CREATE USER 'app_user'@'192.0.2.11' IDENTIFIED BY 'app_user_passwd';
```

Replace 192.0.2.11 with the relevant IP address specification for your application server.

Passwords should meet your organization's password policies.

2. Grant the privileges required by your application to the database user account for your application server:

```
GRANT ALL ON test.* TO 'app_user'@'192.0.2.11';
```

The privileges shown are designed to allow the tests in the subsequent sections to work. The user account for your production application may require different privileges.

Test Connection with Application User

To test the connection, use the MariaDB Client from your application server to connect to an Enterprise Cluster node through MaxScale.

This action is performed on the application server:

```
$ mariadb --host 192.0.2.104 --port 3307
      --user app_user --password
```

Test Connection with Read Connection Router

If you configured the Read Connection Router, confirm that MaxScale routes connections to the replica servers.

1. On the MaxScale node, use the maxctrl list listeners command to view the available listeners and ports:

```
$ maxctrl list listeners
```

```
┌─────────────────────────────┬──────┬──────┬─────────┬──────────────────────────┐
│ Name                        │ Port │ Host │ State   │ Service                  │
├─────────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ connection_router_listener  │ 3308 │ ::   │ Running │ connection_router_service │
├─────────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
│ query_router_listener       │ 3307 │ ::   │ Running │ query_router_service      │
└─────────────────────────────┴──────┴──────┴─────────┴──────────────────────────┘
```

2. Open multiple terminals connected to your application server, in each use MariaDB Client to connect to the listener port for the Read Connection Router (in the example 3308):

```
$ mariadb --host 192.0.2.104 --port 3308 \
      --user app_user --password
```

Use the application user credentials you created for the `--user` and `--password` options.

3. In each terminal, query the hostname system variable to identify to which you're connected:

```
SELECT @@global.hostname;

+-------------------+
| @@global.hostname |
+-------------------+
|             node2 |
+-------------------+
```

Different terminals should return different values since MaxScale routes the connections to different nodes.

Since the router was configured the slave router option, the Read Connection Router only routes connections to replica servers.

Test Write Queries with the Read/Write Split Router

If you configured the Read/Write Split Router, confirm that readwritesplit correctly routes write queries.

This action is performed with multiple client connections to the MaxScale node.

1. On the MaxScale node, use the maxctrl list servers command to identify the Enterprise Cluster node currently operating as the primary server:

```
$ maxctrl list servers
```

```
┌────────┬───────────────┬──────┬─────────────┬─────────────────────────┬──────┐
│ Server │ Address       │ Port │ Connections │ State                   │ GTID │
├────────┼───────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node1  │ 192.0.2.101 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼───────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node2  │ 192.0.2.102 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼───────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node3  │ 192.0.2.103 │ 3306 │ 0           │ Master, Synced, Running │      │
└────────┴───────────────┴──────┴─────────────┴─────────────────────────┴──────┘
```

The server listed as Master is currently operating as the primary server.

2. On the MaxScale node, use the maxctrl list listeners command to identify the correct listener port:

```
$ maxctrl list listeners galerarouter
```

```
┌──────────────────────────────┬──────┬──────┬─────────┬──────────────────────────────┐
│ Name                         │ Port │ Host │ State   │ Service                      │
├──────────────────────────────┼──────┼──────┼─────────┼──────────────────────────────┤
│ connection_router_listener │ 3308 │      │ Running │ connection_router_service │
│ query_router_listener      │ 3307 │      │ Running │ query_router_service      │
└──────────────────────────────┴──────┴──────┴─────────┴──────────────────────────────┘
```

In the example, the listener port for the Read/Write Split router is 3307.

3. Use the MariaDB Client to establish multiple connections to the listener configured for the Read/Write Split routing service, `query_router_listener`, on the MaxScale node:

```
$ mariadb --host=192.0.2.104 --port=3307 \
      --user=app_user --password=app_user_passwd
```

The database user account for your application server should be specified by the `--user` option.

4. Using any client connection, create a test table:

```
CREATE TABLE test.load_balancing_test (
   id INT PRIMARY KEY AUTO_INCREMENT,
   hostname VARCHAR(256)
);
```

5. Using each client connection, insert the values of the hostname system variable into the table using the INSERT statement to identify the node that executes the statement:

```
INSERT INTO test.load_balancing_test (hostname)
VALUES (@@global.hostname);
```

6. Using any client connection, query the table using the SELECT statement:

```
SELECT * FROM test.load_balancing_test;
```

```
+----+----------+
| id | hostname |
+----+----------+
|  1 | node3    |
|  4 | node3    |
|  7 | node3    |
+----+----------+
```

The output shows the hostname from the Enterprise Cluster node operating as the primary server. (Enterprise Cluster offsets auto-increment values by node to avoid write conflicts.)

Confirm that MaxScale is routing write queries to the Enterprise Cluster node operating as the primary server by checking that the test table only contains the hostname of the correct Enterprise Cluster node.

Test Read Queries with the Read/Write Split Router

If you configured the Read/Write Split Router, confirm that readwritesplit properly routes read queries to multiple replica servers.

This action is performed with multiple clients connected to the MaxScale node.

1. On the MaxScale node, use maxctrl list servers to identify the Enterprise Cluster nodes that are currently operating as replica servers:

```
$ maxctrl list servers
```

```
┌────────┬─────────────┬──────┬─────────────┬─────────────────────────┬──────┐
│ Server │ Address     │ Port │ Connections │ State                   │ GTID │
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node1  │ 192.0.2.101 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node2  │ 192.0.2.102 │ 3306 │ 0           │ Slave, Synced, Running  │      │
├────────┼─────────────┼──────┼─────────────┼─────────────────────────┼──────┤
│ node3  │ 192.0.2.103 │ 3306 │ 0           │ Master, Synced, Running │      │
└────────┴─────────────┴──────┴─────────────┴─────────────────────────┴──────┘
```

The servers listed as Slave are currently operating as replica servers.

2. On the MaxScale node, use the maxctrl list listeners command to identify the correct listener port:

```
$ maxctrl list listeners
```

```
┌──────────────────────────────┬──────┬──────┬─────────┬──────────────────────────┐
| Name                         | Port | Host | State   | Service                  |
├──────────────────────────────┼──────┼──────┼─────────┼──────────────────────────┤
| connection_router_listener | 3308 |      | Running | connection_router_service |
| query_router_listener      | 3307 |      | Running | query_router_service      |
└──────────────────────────────┴──────┴──────┴─────────┴──────────────────────────┘
```

In the example, the listener port for the Read/Write Split router is 3307.

3. Use the MariaDB Client to establish multiple connections to `query_router_listener` which is the listener configured for the Read/Write Split routing service on the MaxScale node:

```
$ mariadb --host=192.0.2.104 --port=3307 \
   --user=app_user --password=app_user_passwd
```

The database user account for your application server should be specified by the –user option.

4. Using each client connection, query the hostname system variable to identify the node that executes the statement:

```
SELECT @@global.hostname;

+-------------------+
| @@global.hostname |
+-------------------+
|             node2 |
+-------------------+
```

Confirm that MaxScale routes the SELECT statements to different replica servers.

For more information on different routing criteria, see `slave_selection_criteria`

Next Step

Navigation in the procedure “Deploy Primary/Replica Topology”:

This page was step 7 of 7.

This procedure is complete.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Single Node Topologies

Overview

Installation and configuration instructions are available for the deployment of single-node topologies of MariaDB Enterprise Server and MariaDB Community Server.

MariaDB ColumnStore

See Community Server with ColumnStore.

These instructions detail a single-node deployment of a columnar data store, for MariaDB Community Server, with data optionally stored on S3-compatible object storage.

For high availability and scalability, see “Enterprise ColumnStore with Object Storage” or “Enterprise ColumnStore with Shared Local Storage”.

MariaDB Community Server

See Community Server.

These instructions detail a single-node deployment of MariaDB Community Server.

MariaDB Enterprise ColumnStore with Local Storage

See Enterprise Server with ColumnStore (Local Storage).

These instructions detail a single-node deployment of a columnar data store.

For high availability and scalability, see “Enterprise ColumnStore with Shared Local Storage”.

MariaDB Enterprise ColumnStore with Object Storage

See Enterprise Server with ColumnStore (Object Storage).

These instructions detail a single-node deployment of a columnar data store for MariaDB Enterprise Server, with data stored on S3-compatible object storage.

For high availability and scalability, see “Enterprise ColumnStore with Object Storage”.

MariaDB Enterprise Server

See Installing MariaDB Enterprise Server.

These instructions detail a single-node deployment of MariaDB Enterprise Server.

\

This page is: Copyright © 2026 MariaDB. All rights reserved.

Community Server

Overview

These instructions detail the deployment of **MariaDB Community Server 10.5** in a **Single Standalone Server** configuration on a range of supported Operating Systems.

These instructions detail how to deploy a single-node row database, which is suited for a transactional or OLTP workload that does not require high availability (HA). This deployment type is generally for non-production use cases, such as for development and testing.

MariaDB Community Server Components

These instructions detail the deployment of the following MariaDB Community Server components:

Component	Description
InnoDB	• It is a general purpose storage engine. • It is ACID-compliant. • It is performant. • It is the transactional component of MariaDB’s single stack Hybrid Transactional/Analytical Processing (HTAP) solution.

Term Definitions

Term	Definition
row database	• A database where all columns of each row are stored together. • Best suited for transactional and OLTP workloads. • Also known as a “row-oriented database”.

Installation

MariaDB Corporation provides package repositories for YUM (RHEL, CentOS), APT (Debian, Ubuntu), and ZYpp (SLES).

Install via YUM (CentOS, RHEL)

1. Configure the YUM package repository.

   Prefix the version with `mariadb-` and pass the version string to the `--mariadb-server-version` flag to mariadb_repo_setup. The following directions reference `11.4`.

   To configure YUM package repositories:

   ```
   $ sudo yum install curl
   ```

   ```
   $ curl -LsSO https://r.mariadb.com/downloads/mariadb_repo_setup
   ```

   ```
   $ echo "${checksum} mariadb_repo_setup" \
       | sha256sum -c -
   ```

   ```
   $ chmod +x mariadb_repo_setup
   ```

   ```
   $ sudo ./mariadb_repo_setup \
      --mariadb-server-version="mariadb-11.4"
   ```

   1. *Checksums of the various releases of the `mariadb_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

2. Install MariaDB Community Server and package dependencies:

   ```
   $ sudo yum install MariaDB-server MariaDB-backup
   ```

3. Configure MariaDB.

   Installation only loads MariaDB Community Server to the system. MariaDB Community Server requires configuration before the database server is ready for use.

   See Configuration.

Install via APT (Debian, Ubuntu)

1. Configure the APT package repository.

   Prefix the version with `mariadb-` and pass the version string to the `--mariadb-server-version` flag to mariadb_repo_setup. The following directions reference `11.4`.

   To configure APT package repositories:

   ```
   $ sudo apt install curl
   ```

   ```
   $ curl -LsSO https://r.mariadb.com/downloads/mariadb_repo_setup
   ```

   ```
   $ echo "${checksum} mariadb_repo_setup" \
       | sha256sum -c -
   ```

```
$ chmod +x mariadb_repo_setup
```

```
$ sudo ./mariadb_repo_setup \
   --mariadb-server-version="mariadb-11.4"
```

```
$ sudo apt update
```

1. *Checksums of the various releases of the `mariadb_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

2. Install MariaDB Community Server and package dependencies:

```
$ sudo apt install mariadb-server mariadb-backup
```

3. Configure MariaDB.

   Installation only loads MariaDB Community Server to the system. MariaDB Community Server requires configuration before the database server is ready for use.

   See Configuration.

Install via ZYpp (SLES)

1. Configure the ZYpp package repository.

   Prefix the version with `mariadb-` and pass the version string to the `--mariadb-server-version` flag to mariadb_repo_setup. The following directions reference `11.4`.

   To configure ZYpp package repositories:

```
$ sudo zypper install curl
```

```
$ curl -LsSO https://r.mariadb.com/downloads/mariadb_repo_setup
```

```
$ echo "${checksum} mariadb_repo_setup" \
    | sha256sum -c -
```

```
$ chmod +x mariadb_repo_setup
```

```
$ sudo ./mariadb_repo_setup \
   --mariadb-server-version="mariadb-11.4"
```

1. *Checksums of the various releases of the `mariadb_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

2. Install MariaDB Community Server and package dependencies:

```
$ sudo zypper install MariaDB-server MariaDB-backup
```

3. Configure MariaDB.

   Installation only loads MariaDB Community Server to the system. MariaDB Community Server requires configuration before the database server is ready for use.

   See Configuration.

Configuration

MariaDB Community Server can be configured in the following ways:

- System variables and options can be set in a configuration file (such as `/etc/my.cnf`). MariaDB Community Server must be restarted to apply changes made to the configuration file.
- System variables and options can be set on the command-line.
- If a system variable supports dynamic changes, then it can be set on-the-fly using the SET statement.

Configuration Files

MariaDB's packages include several bundled configuration files. It is also possible to create custom configuration files.

On RHEL, CentOS, and SLES, MariaDB's packages bundle the following configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

And on RHEL, CentOS, and SLES, custom configuration files from the following directories are read by default:

- `/etc/my.cnf.d/`

On Debian and Ubuntu, MariaDB's packages bundle the following configuration files:

- `/etc/mysql/my.cnf`
- `/etc/mysql/mariadb.cnf`
- `/etc/mysql/mariadb.conf.d/50-client.cnf`
- `/etc/mysql/mariadb.conf.d/50-mysql-clients.cnf`
- `/etc/mysql/mariadb.conf.d/50-mysqld_safe.cnf`
- `/etc/mysql/mariadb.conf.d/50-server.cnf`
- `/etc/mysql/mariadb.conf.d/60-galera.cnf`

And on Debian and Ubuntu, custom configuration files from the following directories are read by default:

- `/etc/mysql/conf.d/`
- `/etc/mysql/mariadb.conf.d/`

Configuring MariaDB

1. Determine which system variables and options you need to configure.

   Useful system variables and options for MariaDB Community Server include:

System Variable/Option	Description
datadir	Sets the path to the data directory. MariaDB Community Server writes data files to this directory, including tablespaces, logs, and schemas. Change it to use a non-standard location or to start the Server on a different data directory for testing.
bind_address	Sets the local TCP/IP address on which MariaDB Community Server listens for incoming connections. When testing on a local system, bind the address to the local host at `127.0.0.1` to prevent network access.
port	Sets the port MariaDB Community Server listens on. Use this system variable to use a non-standard port or when running multiple Servers on the same host for testing.

System Variable/Option	Description
max_connections	Sets the maximum number of simultaneous connections MariaDB Community Server allows.
thread_handling	Sets how MariaDB Community Server handles threads for client connections.
log_error	Sets the file name for the error log.
innodb_buffer_pool_size	Sets the amount of memory InnoDB reserves for the Buffer Pool.
innodb_log_file_size	Sets the size for each Redo Log file and innodb_log_files_in_group sets the number of Redo Log files used by InnoDB.
innodb_io_capacity	Sets the maximum number of I/O operations per second that InnoDB can use.

2. Choose a configuration file in which to configure your system variables and options.

   It is not recommended to make custom changes to one of the bundled configuration files. Instead, it is recommended to create a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. If you want your custom configuration file to override the bundled configuration files, then it is a good idea to prefix the custom configuration file's name with a string that will be sorted last, such as `z-`.

   - On RHEL, CentOS, and SLES, a good custom configuration file would be: `/etc/my.cnf.d/z-custom-my.cnf`
   - On Debian and Ubuntu, a good custom configuration file would be: `/etc/mysql/mariadb.conf.d/z-custom-my.cnf`

3. Set your system variables and options in the configuration file.

   They need to be set in a group that will be read by mariadbd, such as `[mariadb]` or `[server]`.

   For example:

```
[mariadb]
log_error                = mariadb-test.err
innodb_buffer_pool_size   = 1G
```

Starting the Server

MariaDB Community Server includes configuration to start, stop, restart, enable/disable on boot, and check the status of the Server using the operating system default process management system.

For distributions that use `systemd` (most supported OSes), you can manage the Server process using the `systemctl` command:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

Testing

When MariaDB Community Server is up and running on your system, you should test that it is working and there weren't any issues during startup.

1. Connect to the Server using mariadb Client using the `root@localhost` user account:

```
$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 10.5.28-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Community Server with ColumnStore

These instructions detail the deployment of **MariaDB ColumnStore 6** with **MariaDB Community Server 10.6** in a **Single-node ColumnStore Deployment** configuration on a range of supported Operating Systems.

These instructions detail how to deploy a single-node columnar database, which is suited for an analytical or OLAP workload that does not require high availability (HA). This deployment type is generally for non-production use cases, such as for development and testing.

Community Server Components

These instructions detail the deployment of the following MariaDB Community Server components:

Component	Description
MariaDB ColumnStore 6	• It is a columnar storage engine that provides distributed, columnar storage for scalable analytical processing and smart transactions. • It is the analytical component of MariaDB's single stack Hybrid Transactional/Analytical Processing (HTAP) solution.

Term Definitions

Term	Definition
columnar database	• A database where the columns of each row are stored separately. • Best suited for analytical and OLAP workloads. • Also known as a "column-oriented database".
row database	• A database where all columns of each row are stored together. • Best suited for transactional and OLTP workloads. • Also known as a "row-oriented database".

High Availability

Single-node ColumnStore 6 does not support high availability.

For high availability and scalability, instead see "ColumnStore Object Storage Topology" or "ColumnStore Shared Local Storage Topology".

System Preparation

Systems hosting a ColumnStore deployment requires some additional configuration prior to installation:

1. Optimize Linux kernel parameters
2. Configure the Linux security module to support installation
3. Configure the character encoding
4. Optionally configure S3-compatible storage

Optimize Linux Kernel Parameters

MariaDB ColumnStore performs best when certain Linux kernel parameters are optimized.

1. Set the relevant kernel parameters in a sysctl configuration file. For proper change management, we recommend setting them in a ColumnStore-specific configuration file.

   For example, create a `/etc/sysctl.d/90-mariadb-columnstore.conf` file with the following contents:

   ```
   # minimize swapping
   vm.swappiness = 1

   # Increase the TCP max buffer size
   net.core.rmem_max = 16777216
   net.core.wmem_max = 16777216

   # Increase the TCP buffer limits
   # min, default, and max number of bytes to use
   net.ipv4.tcp_rmem = 4096 87380 16777216
   net.ipv4.tcp_wmem = 4096 65536 16777216

   # don't cache ssthresh from previous connection
   net.ipv4.tcp_no_metrics_save = 1

   # for 1 GigE, increase this to 2500
   # for 10 GigE, increase this to 30000
   net.core.netdev_max_backlog = 2500
   ```

2. Set the same kernel parameters at runtime using the `sysctl` command:

   ```
   $ sudo sysctl --load=/etc/sysctl.d/90-mariadb-columnstore.conf
   ```

LSM Configuration for Install

To avoid confusion and potential problems, we recommend configuring the system's Linux Security Module (LSM) during installation. The specific steps to configure the security module will depend on the platform.

In the Configure the Linux Security Module section, we will configure the security module and restart it.

SELinux (RHEL, CentOS)

SELinux must be set to `permissive` mode before installing MariaDB ColumnStore.

1. Set SELinux to permissive mode by setting `SELINUX=permissive` in `/etc/selinux/config`.

   For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2. Reboot the system.
3. Confirm that SELinux is in `permissive` mode using `getenforce`:

```
$ sudo getenforce
```

```
Permissive
```

Note:

Remember to configure and re-enable SELinux after the installation is complete.

AppArmor (Debian, Ubuntu)

AppArmor must be disabled before installing MariaDB ColumnStore.

1. Disable AppArmor:

```
$ sudo systemctl disable apparmor
```

2. Reboot the system.
3. Confirm that no AppArmor profiles are loaded using `aa-status`:

```
$ sudo aa-status
```

Example output:

```
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

Note:

Remember to configure and re-enable AppArmor after the installation is complete.

Configure the Character Encoding

When using MariaDB ColumnStore, it is recommended to set the system's locale to UTF-8.

1. On RHEL 8, install additional dependencies.

```
$ sudo yum install glibc-locale-source glibc-langpack-en
```

2. Set the system's locale to `en_US.UTF-8` by executing `localedef`:

```
$ sudo localedef -i en_US -f UTF-8 en_US.UTF-8
```

S3-Compatible Storage

MariaDB ColumnStore supports S3-compatible object storage.

S3-compatible object storage is optional, but **highly recommended**.

S3-compatible object storage is:

- **Compatible**: Many object storage services are compatible with the Amazon S3 API.
- **Economical**: S3-compatible object storage is often very low cost.
- **Flexible**: S3-compatible object storage is available for both cloud and on-premises deployments.
- **Limitless**: S3-compatible object storage is often virtually limitless.
- **Resilient**: S3-compatible object storage is often low maintenance and highly available, since many services use resilient cloud infrastructure.
- **Scalable**: S3-compatible object storage is often highly optimized for read and write scaling.
- **Secure**: S3-compatible object storage is often encrypted-at-rest.

Many S3-compatible object storage services exist. MariaDB Corporation cannot make guarantees about all S3-compatible object storage services, because different services provide different functionality.

If you have any questions about using specific S3-compatible object storage with MariaDB ColumnStore, contact us.

Create an S3 Bucket

If you want to use S3-compatible storage, it is important to create the S3 bucket before you start ColumnStore.

If you already have an S3 bucket, confirm that the bucket is empty.

We will configure ColumnStore to use the S3 bucket later in the Configure the S3 Storage Manager section.

> **Note:**
>
> The specific steps to create the S3 bucket will depend on what S3-compatible storage you are using.

ColumnStore Installation

MariaDB Corporation provides package repositories for YUM (RHEL, CentOS) and APT (Debian, Ubuntu).

MariaDB ColumnStore ships as a storage engine plugin for MariaDB Community Server and a platform engine to handle back-end storage processes. MariaDB Community Server 10.6 does not require any additional software to operate as a single-node analytics database.

Install ColumnStore via YUM (RHEL, CentOS)

1. Configure the YUM package repository.

   MariaDB ColumnStore `6` is available on MariaDB Community Server `10.6`.

   To configure YUM package repositories:

   ```
   $ sudo yum install curl
   ```

   ```
   $ curl -LsSO https://r.mariadb.com/downloads/mariadb_repo_setup
   ```

```
$ echo "${checksum} mariadb_repo_setup" \
    | sha256sum -c -
```

```
$ chmod +x mariadb_repo_setup
```

```
$ sudo ./mariadb_repo_setup \
   --mariadb-server-version="mariadb-10.6"
```

1. *Checksums of the various releases of the `mariadb_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

2. Install the EPEL repository:

```
$ sudo yum install epel-release
```

3. Install some additional dependencies for ColumnStore:

```
$ sudo yum install jemalloc
```

4. Install MariaDB ColumnStore and package dependencies:

```
$ sudo yum install MariaDB-server MariaDB-backup \
   MariaDB-shared MariaDB-client \
   MariaDB-columnstore-engine
```

5. Configure MariaDB ColumnStore.

   Installation only loads MariaDB ColumnStore to the system. MariaDB ColumnStore requires configuration and additional post-installation steps before the database server is ready for use.

   See ColumnStore Configuration.

Install ColumnStore via APT (Debian, Ubuntu)

1. Configure the APT package repository.

   MariaDB ColumnStore `6` is available on MariaDB Community Server `10.6`.

   To configure APT package repositories:

```
$ sudo apt install curl
```

```
$ curl -LsSO https://r.mariadb.com/downloads/mariadb_repo_setup
```

```
$ echo "${checksum} mariadb_repo_setup" \
    | sha256sum -c -
```

```
$ chmod +x mariadb_repo_setup
```

```
$ sudo ./mariadb_repo_setup \
   --mariadb-server-version="mariadb-10.6"
```

```
$ sudo apt update
```

1. *Checksums of the various releases of the `mariadb_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.*

2. Install some additional dependencies for ColumnStore.

   On Debian 10 and Ubuntu 20.04, install the following:

   ```
   $ sudo apt install libjemalloc2
   ```

   On Debian 9 and Ubuntu 18.04, install the following:

   ```
   $ sudo apt install libjemalloc1
   ```

3. Install MariaDB ColumnStore and package dependencies:

   ```
   $ sudo apt install mariadb-server mariadb-backup \
      libmariadb3 mariadb-client \
      mariadb-plugin-columnstore
   ```

4. Configure MariaDB ColumnStore.

   Installation only loads MariaDB ColumnStore to the system. MariaDB ColumnStore requires configuration and additional post-installation steps before the database server is ready for use.

   See ColumnStore Configuration.

ColumnStore Configuration

MariaDB ColumnStore requires configuration after it is installed. The configuration file location depends on your operating system.

Community Server Configuration

MariaDB Community Server can be configured in the following ways:

- System variables and options can be set in a configuration file (such as `/etc/my.cnf`). MariaDB Community Server must be restarted to apply changes made to the configuration file.
- System variables and options can be set on the command-line.
- If a system variable supports dynamic changes, then it can be set on-the-fly using the SET statement.

Configuration Files

MariaDB's packages include several bundled configuration files. It is also possible to create custom configuration files.

On RHEL and CentOS, MariaDB's packages bundle the following configuration files:

- `/etc/my.cnf`
- `/etc/my.cnf.d/client.cnf`
- `/etc/my.cnf.d/mysql-clients.cnf`
- `/etc/my.cnf.d/server.cnf`

And on RHEL and CentOS, custom configuration files from the following directories are read by default:

- `/etc/my.cnf.d/`

On Debian and Ubuntu, MariaDB's packages bundle the following configuration files:

- `/etc/mysql/my.cnf`

- `/etc/mysql/mariadb.cnf`
- `/etc/mysql/mariadb.conf.d/50-client.cnf`
- `/etc/mysql/mariadb.conf.d/50-mysql-clients.cnf`
- `/etc/mysql/mariadb.conf.d/50-mysqld_safe.cnf`
- `/etc/mysql/mariadb.conf.d/50-server.cnf`
- `/etc/mysql/mariadb.conf.d/60-galera.cnf`

And on Debian and Ubuntu, custom configuration files from the following directories are read by default:

- `/etc/mysql/conf.d/`
- `/etc/mysql/mariadb.conf.d/`

Configuring MariaDB for ColumnStore

1. Determine which system variables and options you need to configure.

   Mandatory system variables and options for single-node MariaDB ColumnStore include:

Connector	MariaDB Connector/R2DBC
character_set_server	Set this system variable to `utf8`
collation_server	Set this system variable to `utf8_general_ci`
loose-columnstore_use_import_for_batchinsert	Set this system variable to `ALWAYS` to always use `cpimport` for LOAD DATA INFILE and INSERT…SELECT statements.

> **Warning:**
>
> The `loose-` prefix is required for ColumnStore system variables in the configuration file. Without it, MariaDB Server will fail to start if the ColumnStore plugin is not installed or has been removed.

1. Choose a configuration file in which to configure your system variables and options.

We recommend **not** making custom changes to one of the bundled configuration files. Instead, create a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. If you want your custom configuration file to override the bundled configuration files, it is a good idea to prefix the custom configuration file's name with a string that will be sorted last, such as `z-`.

- On RHEL and CentOS, a good custom configuration file would be: `/etc/my.cnf.d/z-custom-my.cnf`
- On Debian and Ubuntu, a good custom configuration file would be: `/etc/mysql/mariadb.conf.d/z-custom-my.cnf`

3. Set your system variables and options in the configuration file.

   They need to be set in a group that will be read by mariadbd, such as `[mariadb]` or `[server]`.

   For example:

```
[mariadb]
log_error                       = mariadbd.err
character_set_server            = utf8
collation_server                = utf8_general_ci
```

Configure Cross Engine Joins

When a cross engine join is executed, the ExeMgr process connects to the server using the `root` user with no password by default. MariaDB Community Server 10.6 will reject this login attempt by default. If you plan to use

Cross Engine Joins, you need to configure ColumnStore to use a different user account and password. These directions are for configuring the cross engine join user. Directions for creating the cross engine join user are in the Create the Cross Engine Join User section.

To configure cross engine joins, perform the following steps, use the mcsSetConfig command.

For example, to configure ColumnStore to use the `cross_engine` user account to connect to the server at `127.0.0.1`:

```
$ sudo mcsSetConfig CrossEngineSupport Host 127.0.0.1
$ sudo mcsSetConfig CrossEngineSupport Port 3306
$ sudo mcsSetConfig CrossEngineSupport User cross_engine
$ sudo mcsSetConfig CrossEngineSupport Password cross_engine_passwd
```

> **Note:**
>
> Please choose a password that meets your organization's password policies. If your MariaDB Community Server instance has a password validation plugin installed, then the password should also meet the configured requirements.

Configure the S3 Storage Manager

MariaDB ColumnStore can use S3-compatible storage, but it is not required. S3-compatible storage must be configured before it can be used.

To configure ColumnStore to use S3-compatible storage, edit `/etc/columnstore/storagemanager.cnf`:

```
[ObjectStorage]
…
service = S3
…
[S3]
bucket = your_columnstore_bucket_name
endpoint = your_s3_endpoint
aws_access_key_id = your_s3_access_key_id
aws_secret_access_key = your_s3_secret_key
# iam_role_name = your_iam_role
# sts_region = your_sts_region
# sts_endpoint = your_sts_endpoint

[Cache]
cache_size = your_local_cache_size
path = your_local_cache_path
```

- The default local cache size is 2 GB.
- The default local cache path is `/var/lib/columnstore/storagemanager/cache`.
- Ensure that the local cache path has sufficient store space to store the local cache.
- The `bucket` option must be set to the name of the bucket that you created in the Create an S3 Bucket step.
- To use an IAM role, you must also uncomment and set `iam_role_name`, `sts_region`, and `sts_endpoint`.

Start the ColumnStore Processes

The Community Server and ColumnStore processes can be started using the `systemctl` command. In case the processes were started during the installation process, use the restart command to ensure that the processes pick up the new configuration. Perform the following procedure.

1. Start the MariaDB Community Server process and configure it to start automatically:

   ```
   $ sudo systemctl restart mariadb
   ```

```
$ sudo systemctl enable mariadb
```

2. Start the MariaDB ColumnStore processes and configure them to start automatically:

```
$ sudo systemctl restart mariadb-columnstore
```

```
$ sudo systemctl enable mariadb-columnstore
```

Create User Accounts

For single-node ColumnStore deployments, only a single user account needs to be created.

Create the Cross Engine Join User

The credentials for cross engine joins were previously configured in the Cross Engine Joins section. The user account must also be created and granted the necessary privileges to access data.

1. Connect to the server using MariaDB Client using the `root@localhost` user account:

```
$ sudo mariadb
```

2. Create the user account with the CREATE USER statement:

```
CREATE USER 'cross_engine'@'127.0.0.1'
   IDENTIFIED BY "cross_engine_passwd";
```

```
CREATE USER 'cross_engine'@'localhost'
   IDENTIFIED BY "cross_engine_passwd";
```

Note:

Please choose the same user name and password that was configured in the Cross Engine Joins section.

1. Grant the user account `SELECT` privileges on all databases with the GRANT statement:

```
GRANT SELECT, PROCESS ON *.*
   TO 'cross_engine'@'127.0.0.1';
```

```
GRANT SELECT, PROCESS ON *.*
   TO 'cross_engine'@'localhost';
```

Bulk Import Data

Now that the ColumnStore system is running, you can bulk import your data.

Import the Schema

Before data can be imported into the tables, the schema needs to be created.

1. Connect to the server using mariadb Client using the `root@localhost` user account:

```
$ sudo mariadb
```

2. For each database that you are importing, create the database with the CREATE DATABASE statement:

```
CREATE DATABASE inventory;
```

3. For each table that you are importing, create the table with the CREATE TABLE statement:

```
CREATE TABLE inventory.products (
   product_name VARCHAR(11) NOT NULL DEFAULT '',
   supplier VARCHAR(128) NOT NULL DEFAULT '',
   quantity VARCHAR(128) NOT NULL DEFAULT '',
   unit_cost VARCHAR(128) NOT NULL DEFAULT ''
) ENGINE=Columnstore DEFAULT CHARSET=utf8;
```

cpimport

MariaDB ColumnStore includes cpimport, which is a command-line utility that is designed to efficiently load data in bulk.

To import your data from a TSV (tab-separated values) file with cpimport:

```
$ sudo cpimport -s '\t' inventory products /tmp/inventory-products.tsv
```

LOAD DATA INFILE

When data is loaded with the LOAD DATA INFILE statement, MariaDB ColumnStore loads the data using cpimport, which is a command-line utility that is designed to efficiently load data in bulk.

To import your data from a TSV (tab-separated values) file with LOAD DATA INFILE statement:

```
LOAD DATA INFILE '/tmp/inventory-products.tsv'
INTO TABLE inventory.products;
```

Import from Remote Database

MariaDB ColumnStore can also import data directly from a remote database. A simple method is to query the table using the SELECT statement, and then pipe the results into cpimport, which is a command-line utility that is designed to efficiently load data in bulk.

To import your data from a remote MariaDB database:

```
$ mariadb --quick \
   --skip-column-names \
   --execute="SELECT * FROM inventory.products" \
   | cpimport -s '\t' inventory products
```

Configure the Linux Security Module

If you stopped the Linux Security Module (LSM) during installation, you can restart the module and configure.

The specific steps to configure the security module depend on the operating system.

Configure SELinux (RHEL, CentOS)

We set SELinux to permissive mode in the SELinux section, but we have to create an SELinux policy for ColumnStore before re-enabling it. This will ensure that SELinux does not interfere with ColumnStore's functionality. A policy can be generated while SELinux is still in permissive mode using the `audit2allow` command.

1. To configure SELinux, you have to install the packages required for `audit2allow`.

   On RHEL 7 and CentOS 7, install the following:

   ```
   $ sudo yum install policycoreutils policycoreutils-python
   ```

   On RHEL 8, install the following:

```
$ sudo yum install policycoreutils python3-policycoreutils policycoreutils-python-utils
```

2. Allow the system to run under load for a while to generate SELinux audit events.
3. After the system has taken some load, generate an SELinux policy from the audit events using `audit2allow`:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local
```

If no audit events were found, this will print the following:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local
```

```
Nothing to do
```

4. If audit events were found, the new SELinux policy can be loaded using `semodule`:

```
$ sudo semodule -i mariadb_local.pp
```

5. Set SELinux to enforcing mode by setting `SELINUX=enforcing` in `/etc/selinux/config`:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

6. Reboot the system.
7. Confirm that SELinux is in enforcing mode using `getenforce`:

```
$ sudo getenforce
```

Configure AppArmor (Debian, Ubuntu)

We disabled AppArmor in the AppArmor section, but we have to create an AppArmor profile for ColumnStore before re-enabling it. This will ensure that AppArmor does not interfere with ColumnStore's functionality.

For information on how to create a profile, see How to create an AppArmor Profile on ubuntu.com.

Administration

ColumnStore has several components. Each of those components needs to be administered.

Community Server Administration

MariaDB Community Server uses `systemctl` to start and stop the server processes:

Operation	Command
Start	`sudo systemctl start mariadb`
Stop	`sudo systemctl stop mariadb`
Restart	`sudo systemctl restart mariadb`
Enable during startup	`sudo systemctl enable mariadb`

Operation	Command
Disable during startup	`sudo systemctl disable mariadb`
Status	`sudo systemctl status mariadb`

ColumnStore Administration

MariaDB ColumnStore uses `systemctl` to start and stop the ColumnStore processes:

Operation	Command
Start	`sudo systemctl start mariadb-columnstore`
Stop	`sudo systemctl stop mariadb-columnstore`
Restart	`sudo systemctl restart mariadb-columnstore`
Enable during startup	`sudo systemctl enable mariadb-columnstore`
Disable during startup	`sudo systemctl disable mariadb-columnstore`
Status	`sudo systemctl status mariadb-columnstore`

Testing

When you have MariaDB ColumnStore up and running, you should test it to ensure that it is in working order and that there were not any issues during startup.

Checking Server Status

1. Connect to the server using mariadb Client using the `root@localhost` user account:

```
$ sudo mariadb
```

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 10.6.21-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 1: Prepare Systems for Enterprise ColumnStore Nodes

Step 1: Prepare Systems for Enterprise ColumnStore Nodes

Overview

This page details step 1 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Local storage.

This step prepares the system to host MariaDB Enterprise Server and MariaDB Enterprise ColumnStore.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Optimize Linux Kernel Parameters

MariaDB Enterprise ColumnStore performs best with Linux kernel optimizations.

On each server to host an Enterprise ColumnStore node, optimize the kernel:

1. Set the relevant kernel parameters in a sysctl configuration file. To ensure proper change management, use an Enterprise ColumnStore-specific configuration file.

Create a `/etc/sysctl.d/90-mariadb-enterprise-columnstore.conf` file:

```
# minimize swapping
vm.swappiness = 1

# Increase the TCP max buffer size
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216

# Increase the TCP buffer limits
# min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

# don't cache ssthresh from previous connection
net.ipv4.tcp_no_metrics_save = 1

# for 1 GigE, increase this to 2500
# for 10 GigE, increase this to 30000
net.core.netdev_max_backlog = 2500
```

2. Use the `sysctl` command to set the kernel parameters at runtime

```
$ sudo sysctl --load=/etc/sysctl.d/90-mariadb-enterprise-columnstore.conf
```

Temporarily Configure Linux Security Modules (LSM)

The Linux Security Modules (LSM) should be temporarily disabled on each Enterprise ColumnStore node during installation.

The LSM will be configured and re-enabled later in this deployment procedure.

The steps to disable the LSM depend on the specific LSM used by the operating system.

CentOS / RHEL Stop SELinux

SELinux must be set to permissive mode before installing MariaDB Enterprise ColumnStore.

To set SELinux to permissive mode:

1. Set SELinux to permissive mode:

```
$ sudo setenforce permissive
```

2. Set SELinux to permissive mode by setting SELINUX=permissive in /etc/selinux/config.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. Confirm that SELinux is in permissive mode:

```
$ sudo getenforce
Permissive
```

SELinux will be configured and re-enabled later in this deployment procedure. This configuration is not persistent. If you restart the server before configuring and re-enabling SELinux later in the deployment procedure, you must reset the enforcement to permissive mode.

Debian / Ubuntu AppArmor

AppArmor must be disabled before installing MariaDB Enterprise ColumnStore.

1. Disable AppArmor:

```
$ sudo systemctl disable apparmor
```

2. Reboot the system.
3. Confirm that no AppArmor profiles are loaded using aa-status:

```
$ sudo aa-status
```

```
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

AppArmor will be configured and re-enabled later in this deployment procedure.

Configure Character Encoding

When using MariaDB Enterprise ColumnStore, it is recommended to set the system's locale to UTF-8.

1. On RHEL 8, install additional dependencies:

```
$ sudo yum install glibc-locale-source glibc-langpack-en
```

2. Set the system's locale to en_US.UTF-8 by executing localedef:

```
$ sudo localedef -i en_US -f UTF-8 en_US.UTF-8
```

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Local storage deployment procedure:

This page was step 1 of 5.

Next: Step 2: Install MariaDB Enterprise ColumnStore.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 2: Install Enterprise ColumnStore

Overview

This page details step 2 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Local storage.

This step installs MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Download Token

MariaDB Corporation provides package repositories for CentOS / RHEL (YUM) and Debian / Ubuntu (APT). A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to https://customers.mariadb.com/downloads/token/
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On each Enterprise ColumnStore node, install the prerequisites for downloading the software from the Web. Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

2. On each Enterprise ColumnStore node, configure package repositories and specify Enterprise Server:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \

 | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-maxscale \
      --skip-tools \
      --mariadb-server-version="11.4"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install Enterprise ColumnStore

Install additional dependencies:

Install on CentOS / RHEL (YUM)

```
$ sudo yum install epel-release

$ sudo yum install jemalloc
```

Install of Debian 10 and Ubuntu 20.04 (APT):

```
$ sudo apt install libjemalloc2
```

Install on Debian 9 and Ubuntu 18.04 (APT):

```
$ sudo apt install libjemalloc1
```

Install MariaDB Enterprise Server and MariaDB Enterprise ColumnStore:

Install on CentOS / RHEL (YUM):

```
$ sudo yum install MariaDB-server \
   MariaDB-backup \
   MariaDB-shared \
   MariaDB-client \
   MariaDB-columnstore-engine
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install mariadb-server \
   mariadb-backup \
   libmariadb3 \
   mariadb-client \
   mariadb-plugin-columnstore
```

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Local storage deployment procedure:

This page was step 2 of 5.

Next: Step 3: Start and Configure MariaDB Enterprise ColumnStore.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 3: Start and Configure Enterprise ColumnStore

Overview

This page details step 3 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Local storage.

This step starts and configures MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Configure Enterprise ColumnStore

Mandatory system variables and options for Single-Node Enterprise ColumnStore include:

Connector	MariaDB Connector/R2DBC
character_set_server	Set this system variable to `utf8`
collation_server	Set this system variable to `utf8_general_ci`

Connector	MariaDB Connector/R2DBC
loose-columnstore_use_import_for_batchinsert	Set this system variable to `ALWAYS` to always use `cpimport` for LOAD DATA INFILE and INSERT…SELECT statements.

Warning:

The `loose-` prefix is required for ColumnStore system variables in the configuration file. Without it, MariaDB Server will fail to start if the ColumnStore plugin is not installed or has been removed.

Example Configuration

```
[mariadb]
log_error                               = mariadbd.err
character_set_server                    = utf8
collation_server                        = utf8_general_ci
```

Start the Enterprise ColumnStore Services

Start and enable the MariaDB Enterprise Server service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb

$ sudo systemctl enable mariadb
```

Start and enable the MariaDB Enterprise ColumnStore service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb-columnstore

$ sudo systemctl enable mariadb-columnstore
```

Create the Utility User

Enterprise ColumnStore requires a mandatory utility user account. By default, it connects to the server using the root user with no password. MariaDB Enterprise Server 10.6 will reject this login attempt by default, so you will need to configure Enterprise ColumnStore to use a different user account and password and create this user account on Enterprise Server.

1. On the Enterprise ColumnStore node, create the user account with the CREATE USER statement:

```
CREATE USER 'util_user'@'127.0.0.1'
IDENTIFIED BY 'util_user_passwd';
```

2. On the Enterprise ColumnStore node, grant the user account SELECT privileges on all databases with the GRANT statement:

```
GRANT SELECT, PROCESS ON *.*
TO 'util_user'@'127.0.0.1';
```

3. Configure Enterprise ColumnStore to use the utility user:

```
$ sudo mcsSetConfig CrossEngineSupport Host 127.0.0.1

$ sudo mcsSetConfig CrossEngineSupport Port 3306

$ sudo mcsSetConfig CrossEngineSupport User util_user
```

4. Set the password:

```
$ sudo mcsSetConfig CrossEngineSupport Password util_user_passwd
```

For details about how to encrypt the password, see "Credentials Management for MariaDB Enterprise ColumnStore".

Passwords should meet your organization's password policies. If your MariaDB Enterprise Server instance has a password validation plugin installed, then the password should also meet the configured requirements.

Configure Linux Security Modules (LSM)

The specific steps to configure the security module depend on the operating system.

Configure SELinux (CentOS, RHEL)

Configure SELinux for Enterprise ColumnStore:

1. To configure SELinux, you have to install the packages required for audit2allow.
   On CentOS 7 and RHEL 7, install the following:

```
$ sudo yum install policycoreutils policycoreutils-python
```

On RHEL 8, install the following:

```
$ sudo yum install policycoreutils python3-policycoreutils policycoreutils-python-utils
```

2. Allow the system to run under load for a while to generate SELinux audit events.
3. After the system has taken some load, generate an SELinux policy from the audit events using audit2allow:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local
```

If no audit events were found, this will print the following:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local

Nothing to do
```

4. If audit events were found, the new SELinux policy can be loaded using semodule:

```
$ sudo semodule -i mariadb_local.pp
```

5. Set SELinux to enforcing mode by setting SELINUX=enforcing in /etc/selinux/config.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

6. Set SELinux to enforcing mode:

```
$ sudo setenforce enforcing
```

Configure AppArmor (Ubuntu)

For information on how to create a profile, see How to create an AppArmor Profile on ubuntu.com.

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Local storage deployment procedure:

This page was step 3 of 5.

Next: Step 4: Test MariaDB Enterprise ColumnStore.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 4: Test Enterprise ColumnStore

Overview

This page details step 4 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Local storage.

This step tests MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Test Local Connection

Connect to the server using MariaDB Client using the root@localhost user account:

```
$ sudo mariadb
```

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Test ColumnStore Plugin Status

Query information_schema.PLUGINS and confirm that the ColumnStore storage engine plugin is `ACTIVE`:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS
FROM information_schema.PLUGINS
WHERE PLUGIN_LIBRARY LIKE 'ha_columnstore%';
```

```
+--------------------+---------------+
| PLUGIN_NAME        | PLUGIN_STATUS |
+--------------------+---------------+
| Columnstore        | ACTIVE        |
| COLUMNSTORE_COLUMNS | ACTIVE        |
| COLUMNSTORE_TABLES  | ACTIVE        |
| COLUMNSTORE_FILES   | ACTIVE        |
| COLUMNSTORE_EXTENTS | ACTIVE        |
+--------------------+---------------+
```

Test ColumnStore Table Creation

1. Create a test database, if it does not exist:

```
CREATE DATABASE IF NOT EXISTS test;
```

2. Create a ColumnStore table:

```
CREATE TABLE IF NOT EXISTS test.contacts (
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   email VARCHAR(100)
) ENGINE=ColumnStore;
```

3. Add sample data into the table:

```
INSERT INTO test.contacts (first_name, last_name, email)
   VALUES
   ("Kai", "Devi", "kai.devi@example.com"),
   ("Lee", "Wang", "lee.wang@example.com");
```

4. Read data from table:

```
SELECT * FROM test.contacts;
```

```
+------------+-----------+----------------------+
| first_name | last_name | email                |
+------------+-----------+----------------------+
| Kai        | Devi      | kai.devi@example.com |
| Lee        | Wang      | lee.wang@example.com |
+------------+-----------+----------------------+
```

Test Cross Engine Join

1. Create an InnoDB table:

```
CREATE TABLE test.addresses (
   email VARCHAR(100),
   street_address VARCHAR(255),
   city VARCHAR(100),
   state_code VARCHAR(2)
) ENGINE = InnoDB;
```

2. Add data to the table:

```
INSERT INTO test.addresses (email, street_address, city, state_code)
   VALUES
   ("kai.devi@example.com", "1660 Amphibious Blvd.", "Redwood City", "CA"),
   ("lee.wang@example.com", "32620 Little Blvd", "Redwood City", "CA");
```

3. Perform a cross-engine join:

```
SELECT name AS "Name", addr AS "Address"
FROM (SELECT CONCAT(first_name, " ", last_name) AS name,
   email FROM test.contacts) AS contacts
INNER JOIN (SELECT CONCAT(street_address, ", ", city, ", ", state_code) AS addr,
   email FROM test.addresses) AS addr
WHERE  contacts.email = addr.email;
```

```
+----------+----------------------------------------+
| Name     | Address                                |
+----------+----------------------------------------+
| Kai Devi | 1660 Amphibious Blvd., Redwood City, CA |
| Lee Wang | 32620 Little Blvd, Redwood City, CA     |
+----------+----------------------------------------+

+------------------+-------------------------------------+
| Name             | Address                             |
+------------------+-------------------------------------+
| Walker Percy     | 500 Thomas More Dr., Covington, LA  |
| Flannery O'Connor | 300 Tarwater Rd., Milledgeville, GA |
+------------------+-------------------------------------+
```

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Local storage deployment procedure:

This page was step 4 of 5.

Next: Step 5: Bulk Import of Data.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 5: Bulk Import of Data

Overview

This page details step 5 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Local storage.

This step bulk imports data to Enterprise ColumnStore.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Import the Schema

Before data can be imported into the tables, create a matching schema.

On the primary server, create the schema:

1. For each database that you are importing, create the database with the CREATE DATABASE statement:

```
CREATE DATABASE inventory;
```

2. For each table that you are importing, create the table with the CREATE TABLE statement:

```
CREATE TABLE inventory.products (
   product_name VARCHAR(11) NOT NULL DEFAULT '',
   supplier VARCHAR(128) NOT NULL DEFAULT '',
   quantity VARCHAR(128) NOT NULL DEFAULT '',
   unit_cost VARCHAR(128) NOT NULL DEFAULT ''
) ENGINE=Columnstore DEFAULT CHARSET=utf8;
```

Import the Data

Enterprise ColumnStore supports multiple methods to import data into ColumnStore tables.

cpimport

MariaDB Enterprise ColumnStore includes cpimport, which is a command-line utility designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a TSV (tab-separated values) file, on the primary server run cpimport:

```
$ sudo cpimport -s '\t' inventory products /tmp/inventory-products.tsv
```

LOAD DATA INFILE

When data is loaded with the LOAD DATA INFILE statement, MariaDB Enterprise ColumnStore loads the data using cpimport, which is a command-line utility designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a TSV (tab-separated values) file, on the primary server use LOAD DATA INFILE statement:

```
LOAD DATA INFILE '/tmp/inventory-products.tsv'
INTO TABLE inventory.products;
```

Import from Remote Database

MariaDB Enterprise ColumnStore can also import data directly from a remote database. A simple method is to query the table using the SELECT statement, and then pipe the results into cpimport, which is a command-line utility that is designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a remote MariaDB database:

```
$ mariadb --quick \
   --skip-column-names \
   --execute="SELECT * FROM inventory.products" \
   | cpimport -s '\t' inventory products
```

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Local storage deployment procedure:

This page was step 5 of 5.

This procedure is complete.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 1: Prepare Systems for Enterprise ColumnStore Nodes

Overview

This page details step 1 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Object storage.

This step prepares the system to host MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Optimize Linux Kernel Parameters

MariaDB Enterprise ColumnStore performs best with Linux kernel optimizations.

On each server to host an Enterprise ColumnStore node, optimize the kernel:

1. Set the relevant kernel parameters in a sysctl configuration file. To ensure proper change management, use an Enterprise ColumnStore-specific configuration file.

Create a `/etc/sysctl.d/90-mariadb-enterprise-columnstore.conf` file:

```
# minimize swapping
vm.swappiness = 1

# Increase the TCP max buffer size
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216

# Increase the TCP buffer limits
# min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

# don't cache ssthresh from previous connection
net.ipv4.tcp_no_metrics_save = 1

# for 1 GigE, increase this to 2500
# for 10 GigE, increase this to 30000
net.core.netdev_max_backlog = 2500
```

2. Use the sysctl command to set the kernel parameters at runtime

```
$ sudo sysctl --load=/etc/sysctl.d/90-mariadb-enterprise-columnstore.conf
```

Temporarily Configure Linux Security Modules (LSM)

The Linux Security Modules (LSM) should be temporarily disabled on each Enterprise ColumnStore node during installation.

The LSM will be configured and re-enabled later in this deployment procedure.

The steps to disable the LSM depend on the specific LSM used by the operating system.

CentOS / RHEL Stop SELinux

SELinux must be set to permissive mode before installing MariaDB Enterprise ColumnStore.

To set SELinux to permissive mode:

1. Set SELinux to permissive mode:

```
$ sudo setenforce permissive
```

2. Set SELinux to permissive mode by setting SELINUX=permissive in /etc/selinux/config.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. Confirm that SELinux is in permissive mode:

```
$ sudo getenforce
Permissive
```

SELinux will be configured and re-enabled later in this deployment procedure. This configuration is not persistent. If you restart the server before configuring and re-enabling SELinux later in the deployment procedure, you must reset the enforcement to permissive mode.

Debian / Ubuntu AppArmor

AppArmor must be disabled before installing MariaDB Enterprise ColumnStore.

1. Disable AppArmor:

```
$ sudo systemctl disable apparmor
```

2. Reboot the system.
3. Confirm that no AppArmor profiles are loaded using aa-status:

```
$ sudo aa-status
```

```
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

AppArmor will be configured and re-enabled later in this deployment procedure.

Configure Character Encoding

When using MariaDB Enterprise ColumnStore, it is recommended to set the system's locale to UTF-8.

1. On RHEL 8, install additional dependencies:

```
$ sudo yum install glibc-locale-source glibc-langpack-en
```

2. Set the system's locale to en_US.UTF-8 by executing localedef:

```
$ sudo localedef -i en_US -f UTF-8 en_US.UTF-8
```

Create an S3 Bucket

If you want to use S3-compatible storage, it is important to create the S3 bucket before you start ColumnStore.

If you already have an S3 bucket, confirm that the bucket is empty.

S3 bucket configuration will be performed later in this procedure.

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Object storage deployment procedure:

This page was step 1 of 5.

Next: Step 2: Install MariaDB Enterprise ColumnStore.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 2: Install Enterprise ColumnStore

Overview

This page details step 2 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Object storage.

This step installs MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Retrieve Download Token

MariaDB Corporation provides package repositories for CentOS / RHEL (YUM) and Debian / Ubuntu (APT). A download token is required to access the MariaDB Enterprise Repository.

Customer Download Tokens are customer-specific and are available through the MariaDB Customer Portal.

To retrieve the token for your account:

1. Navigate to https://customers.mariadb.com/downloads/token/
2. Log in.
3. Copy the Customer Download Token.

Substitute your token for `CUSTOMER_DOWNLOAD_TOKEN` when configuring the package repositories.

Set Up Repository

1. On each Enterprise ColumnStore node, install the prerequisites for downloading the software from the Web. Install on CentOS / RHEL (YUM):

```
$ sudo yum install curl
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install curl apt-transport-https
```

2. On each Enterprise ColumnStore node, configure package repositories and specify Enterprise Server:

```
$ curl -LsSO https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
```

```
$ echo "${checksum}  mariadb_es_repo_setup" \

 | sha256sum -c -
```

```
$ chmod +x mariadb_es_repo_setup
```

```
$ sudo ./mariadb_es_repo_setup --token="CUSTOMER_DOWNLOAD_TOKEN" --apply \
      --skip-maxscale \
      --skip-tools \
      --mariadb-server-version="11.4"
```

Checksums of the various releases of the `mariadb_es_repo_setup` script can be found in the Versions section at the bottom of the MariaDB Package Repository Setup and Usage page. Substitute `${checksum}` in the example above with the latest checksum.

Install Enterprise ColumnStore

Install additional dependencies:

Install on CentOS / RHEL (YUM)

```
$ sudo yum install epel-release

$ sudo yum install jemalloc
```

Install of Debian 10 and Ubuntu 20.04 (APT):

```
$ sudo apt install libjemalloc2
```

Install on Debian 9 and Ubuntu 18.04 (APT):

```
$ sudo apt install libjemalloc1
```

Install MariaDB Enterprise Server and MariaDB Enterprise ColumnStore:

Install on CentOS / RHEL (YUM):

```
$ sudo yum install MariaDB-server \
   MariaDB-backup \
   MariaDB-shared \
   MariaDB-client \
   MariaDB-columnstore-engine
```

Install on Debian / Ubuntu (APT):

```
$ sudo apt install mariadb-server \
   mariadb-backup \
   libmariadb3 \
   mariadb-client \
   mariadb-plugin-columnstore
```

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Object storage deployment procedure:

This page was step 2 of 5.

Next: Step 3: Start and Configure MariaDB Enterprise ColumnStore.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 3: Start and Configure Enterprise ColumnStore

Overview

This page details step 3 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Object storage.

This step starts and configures MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Configure Enterprise ColumnStore

Mandatory system variables and options for Single-Node Enterprise ColumnStore include:

Connector	MariaDB Connector/R2DBC
character_set_server	Set this system variable to `utf8`
collation_server	Set this system variable to `utf8_general_ci`

Connector	MariaDB Connector/R2DBC
loose-columnstore_use_import_for_batchinsert	Set this system variable to `ALWAYS` to always use `cpimport` for LOAD DATA INFILE and INSERT…SELECT statements.

Warning:

The `loose-` prefix is required for ColumnStore system variables in the configuration file. Without it, MariaDB Server will fail to start if the ColumnStore plugin is not installed or has been removed.

Example Configuration

```
[mariadb]
log_error                              = mariadbd.err
character_set_server                   = utf8
collation_server                       = utf8_general_ci
```

Configure the S3 Storage Manager

Configure Enterprise ColumnStore S3 Storage Manager to use S3-compatible storage by editing the `/etc/columnstore/storagemanager.cnf` configuration file:

```
[ObjectStorage]
…
service = S3
…
[S3]
bucket                = your_columnstore_bucket_name
endpoint              = your_s3_endpoint
aws_access_key_id     = your_s3_access_key_id
aws_secret_access_key = your_s3_secret_key
# iam_role_name       = your_iam_role
# sts_region          = your_sts_region
# sts_endpoint        = your_sts_endpoint
# ec2_iam_mode        = enabled

[Cache]
cache_size = your_local_cache_size
path       = your_local_cache_path
```

The S3-compatible object storage options are configured under `[S3]`:

- The `bucket` option must be set to the name of the bucket that you created in "Create an S3 Bucket".
- The `endpoint` option must be set to the endpoint for the S3-compatible object storage.
- The `aws_access_key_id` and `aws_secret_access_key` options must be set to the access key ID and secret access key for the S3-compatible object storage.
- To use a specific IAM role, you must uncomment and set `iam_role_name`, `sts_region`, and `sts_endpoint`.
- To use the IAM role assigned to an EC2 instance, you must uncomment `ec2_iam_mode=enabled`.

The local cache options are configured under [Cache]:

- The `cache_size` option is set to 2 GB by default.
- The `path` option is set to `/var/lib/columnstore/storagemanager/cache` by default.

Ensure that the specified path has sufficient storage space for the specified cache size.

Start the Enterprise ColumnStore Services

Start and enable the MariaDB Enterprise Server service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb

$ sudo systemctl enable mariadb
```

Start and enable the MariaDB Enterprise ColumnStore service, so that it starts automatically upon reboot:

```
$ sudo systemctl start mariadb-columnstore

$ sudo systemctl enable mariadb-columnstore
```

Create the Utility User

Enterprise ColumnStore requires a mandatory utility user account to perform cross-engine joins and similar operations.

1. Create the user account with the CREATE USER statement:

```
CREATE USER 'util_user'@'127.0.0.1'
IDENTIFIED BY 'util_user_passwd';
```

2. Grant the user account `SELECT` privileges on all databases with the GRANT statement:

```
GRANT SELECT, PROCESS ON *.*
TO 'util_user'@'127.0.0.1';
```

3. Configure Enterprise ColumnStore to use the utility user:

```
$ sudo mcsSetConfig CrossEngineSupport Host 127.0.0.1

$ sudo mcsSetConfig CrossEngineSupport Port 3306

$ sudo mcsSetConfig CrossEngineSupport User util_user
```

4. Set the password:

```
$ sudo mcsSetConfig CrossEngineSupport Password util_user_passwd
```

For details about how to encrypt the password, see "Credentials Management for MariaDB Enterprise ColumnStore".

Passwords should meet your organization's password policies. If your MariaDB Enterprise Server instance has a password validation plugin installed, then the password should also meet the configured requirements.

Configure Linux Security Modules (LSM)

The specific steps to configure the security module depend on the operating system.

Configure SELinux (CentOS, RHEL)

Configure SELinux for Enterprise ColumnStore:

1. To configure SELinux, you have to install the packages required for audit2allow.
   On CentOS 7 and RHEL 7, install the following:

```
$ sudo yum install policycoreutils policycoreutils-python
```

On RHEL 8, install the following:

```
$ sudo yum install policycoreutils python3-policycoreutils policycoreutils-python-utils
```

2. Allow the system to run under load for a while to generate SELinux audit events.
3. After the system has taken some load, generate an SELinux policy from the audit events using audit2allow:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local
```

If no audit events were found, this will print the following:

```
$ sudo grep mysqld /var/log/audit/audit.log | audit2allow -M mariadb_local

Nothing to do
```

4. If audit events were found, the new SELinux policy can be loaded using semodule:

```
$ sudo semodule -i mariadb_local.pp
```

5. Set SELinux to enforcing mode by setting `SELINUX=enforcing` in `/etc/selinux/config`.

For example, the file will usually look like this after the change:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

6. Set SELinux to enforcing mode:

```
$ sudo setenforce enforcing
```

Configure AppArmor (Ubuntu)

For information on how to create a profile, see How to create an AppArmor Profile on ubuntu.com.

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Object storage deployment procedure:

This page was step 3 of 5.

Next: Step 4: Test MariaDB Enterprise ColumnStore.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 4: Test Enterprise ColumnStore

Overview

This page details step 4 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Object storage.

This step tests MariaDB Enterprise Server and MariaDB Enterprise ColumnStore 23.10.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Test S3 Connection

MariaDB Enterprise ColumnStore 23.10 includes a testS3Connection command to test the S3 configuration, permissions, and connectivity.

On each Enterprise ColumnStore node, test the S3 configuration:

```
$ sudo testS3Connection
```

```
StorageManager[26887]: Using the config file found at /etc/columnstore/storagemanager.cnf
StorageManager[26887]: S3Storage: S3 connectivity & permissions are OK
S3 Storage Manager Configuration OK
```

If the `testS3Connection` command does not return `OK`, investigate the S3 configuration.

Test Local Connection

Connect to the server using MariaDB Client using the `root@localhost` user account:

```
$ sudo mariadb
```

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Test ColumnStore Plugin Status

Query information_schema.PLUGINS and confirm that the ColumnStore storage engine plugin is `ACTIVE`:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS
FROM information_schema.PLUGINS
WHERE PLUGIN_LIBRARY LIKE 'ha_columnstore%';
```

```
+---------------------+---------------+
| PLUGIN_NAME         | PLUGIN_STATUS |
+---------------------+---------------+
| Columnstore         | ACTIVE        |
| COLUMNSTORE_COLUMNS | ACTIVE        |
| COLUMNSTORE_TABLES  | ACTIVE        |
| COLUMNSTORE_FILES   | ACTIVE        |
| COLUMNSTORE_EXTENTS | ACTIVE        |
+---------------------+---------------+
```

Test ColumnStore Table Creation

1. Create a test database, if it does not exist:

```
CREATE DATABASE IF NOT EXISTS test;
```

2. Create a ColumnStore table:

```
CREATE TABLE IF NOT EXISTS test.contacts (
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   email VARCHAR(100)
) ENGINE=ColumnStore;
```

3. Add sample data into the table:

```
INSERT INTO test.contacts (first_name, last_name, email)
   VALUES
   ("Kai", "Devi", "kai.devi@example.com"),
   ("Lee", "Wang", "lee.wang@example.com");
```

4. Read data from table:

```
SELECT * FROM test.contacts;

+------------+-----------+----------------------+
| first_name | last_name | email                |
+------------+-----------+----------------------+
| Kai        | Devi      | kai.devi@example.com |
| Lee        | Wang      | lee.wang@example.com |
+------------+-----------+----------------------+
```

Test Cross Engine Join

1. Create an InnoDB table:

```
CREATE TABLE test.addresses (
   email VARCHAR(100),
   street_address VARCHAR(255),
   city VARCHAR(100),
   state_code VARCHAR(2)
) ENGINE = InnoDB;
```

2. Add data to the table:

```
INSERT INTO test.addresses (email, street_address, city, state_code)
   VALUES
   ("kai.devi@example.com", "1660 Amphibious Blvd.", "Redwood City", "CA"),
   ("lee.wang@example.com", "32620 Little Blvd", "Redwood City", "CA");
```

3. Perform a cross-engine join:

```
SELECT name AS "Name", addr AS "Address"
FROM (SELECT CONCAT(first_name, " ", last_name) AS name,
   email FROM test.contacts) AS contacts
INNER JOIN (SELECT CONCAT(street_address, ", ", city, ", ", state_code) AS addr,
   email FROM test.addresses) AS addr
WHERE  contacts.email = addr.email;
```

```
+----------+----------------------------------------+
| Name     | Address                                |
+----------+----------------------------------------+
| Kai Devi | 1660 Amphibious Blvd., Redwood City, CA |
| Lee Wang | 32620 Little Blvd, Redwood City, CA     |
+----------+----------------------------------------+

+------------------+-------------------------------------+
| Name             | Address                             |
+------------------+-------------------------------------+
| Walker Percy     | 500 Thomas More Dr., Covington, LA  |
| Flannery O'Connor | 300 Tarwater Rd., Milledgeville, GA |
+------------------+-------------------------------------+
```

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Object storage deployment procedure:

This page was step 4 of 5.

Next: Step 5: Bulk Import of Data.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 5: Bulk Import of Data

Overview

This page details step 5 of a 5-step procedure for deploying Single-Node Enterprise ColumnStore with Object storage.

This step bulk imports data to Enterprise ColumnStore.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Import the Schema

Before data can be imported into the tables, create a matching schema.

On the primary server, create the schema:

1. For each database that you are importing, create the database with the CREATE DATABASE statement:

```
CREATE DATABASE inventory;
```

2. For each table that you are importing, create the table with the CREATE TABLE statement:

```
CREATE TABLE inventory.products (
   product_name VARCHAR(11) NOT NULL DEFAULT '',
   supplier VARCHAR(128) NOT NULL DEFAULT '',
   quantity VARCHAR(128) NOT NULL DEFAULT '',
   unit_cost VARCHAR(128) NOT NULL DEFAULT ''
) ENGINE=Columnstore DEFAULT CHARSET=utf8;
```

Import the Data

Enterprise ColumnStore supports multiple methods to import data into ColumnStore tables.

cpimport

MariaDB Enterprise ColumnStore includes cpimport, which is a command-line utility designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a TSV (tab-separated values) file, on the primary server run cpimport:

```
$ sudo cpimport -s '\t' inventory products /tmp/inventory-products.tsv
```

LOAD DATA INFILE

When data is loaded with the LOAD DATA INFILE statement, MariaDB Enterprise ColumnStore loads the data using cpimport, which is a command-line utility designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a TSV (tab-separated values) file, on the primary server use LOAD DATA INFILE statement:

```
LOAD DATA INFILE '/tmp/inventory-products.tsv'
INTO TABLE inventory.products;
```

Import from Remote Database

MariaDB Enterprise ColumnStore can also import data directly from a remote database. A simple method is to query the table using the SELECT statement, and then pipe the results into cpimport, which is a command-line utility that is designed to efficiently load data in bulk. Alternative methods are available.

To import your data from a remote MariaDB database:

```
$ mariadb --quick \
   --skip-column-names \
   --execute="SELECT * FROM inventory.products" \
   | cpimport -s '\t' inventory products
```

Next Step

Navigation in the Single-Node Enterprise ColumnStore topology with Object storage deployment procedure:

This page was step 5 of 5.

This procedure is complete.

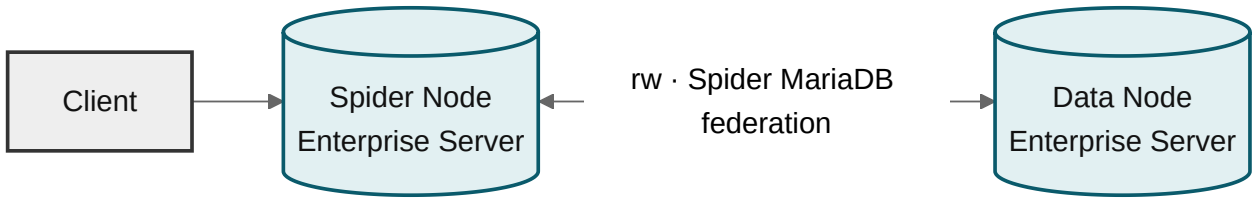
This page is: Copyright © 2026 MariaDB. All rights reserved.

Spider Storage Engine

Spider Topologies

Spider Federated

MariaDB Enterprise Spider enables reading from and writing to tables on remote Enterprise Server nodes. It uses the Spider storage engine for “virtual” Federated Spider Tables, querying Data Tables on Data Nodes (which use non-Spider engines) via a MariaDB foreign data wrapper. This solution supports transactions and is available with Enterprise Server 10.3+.

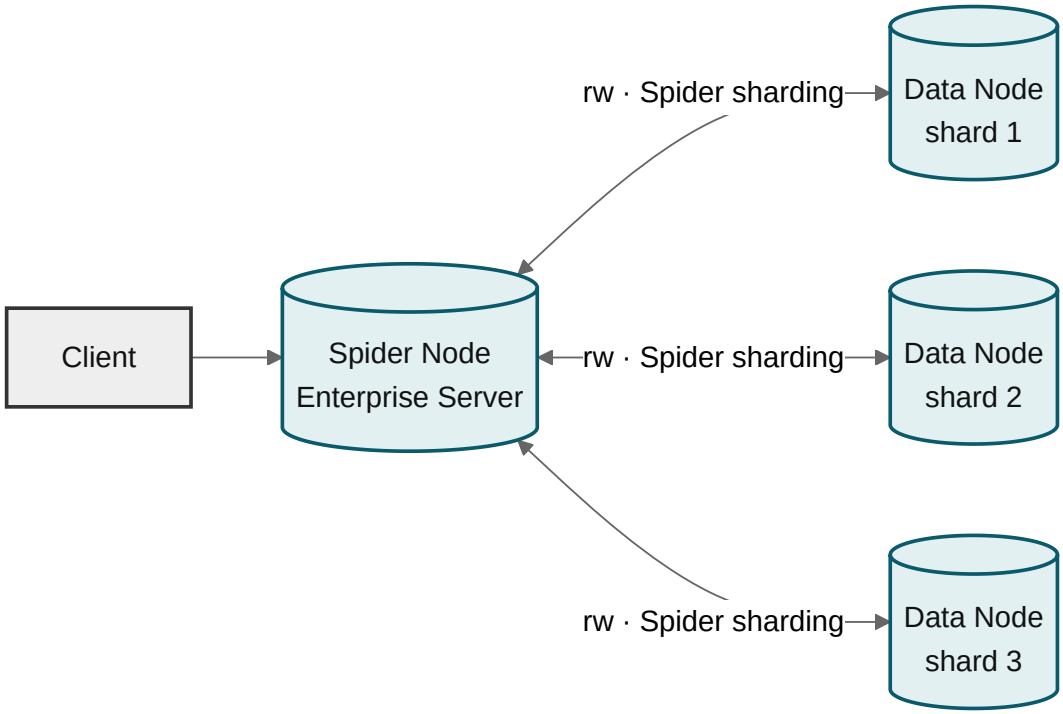


Spider Federated: a Spider Node reads from and writes to a Data Table on a remote Data Node via the Spider MariaDB federation.

- **Read from and write to tables on remote ES nodes**
- Spider Node uses Spider storage engine for Federated Spider Tables
- Federated Spider Table is a “virtual” table
- Spider uses MariaDB foreign data wrapper to query Data Table on Data Node
- Data Node uses non-Spider storage engine for Data Tables
- Supports transactions
- Enterprise Server 10.3+, Enterprise Spider

Spider Sharded

MariaDB Enterprise Spider facilitates horizontal scalability by sharding tables. It uses the Spider storage engine for partitioned “virtual” Sharded Spider Tables, querying Data Tables on Data Nodes (using non-Spider engines) for each partition via a MariaDB foreign data wrapper. This solution supports transactions and is available with Enterprise Server 10.3+.



Spider Sharded: a Spider Node distributes the partitions of a virtual sharded table across multiple Data Nodes (shards) via the Spider foreign data wrapper.

- **Shard tables for horizontal scalability**
- Spider Node uses Spider storage engine for Sharded Spider Tables
- Sharded Spider Table is a partitioned “virtual” table
- Spider uses MariaDB foreign data wrapper to query Data Tables on Data Nodes for each partition
- Data Node uses non-Spider storage engine for Data Tables
- Supports transactions
- Enterprise Server 10.3+, Enterprise Spider

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Enterprise Spider Topologies

See Federated MariaDB Enterprise Spider Topology.

Query, join, or migrate tables on a remote MariaDB Enterprise Server node from a Spider Node using virtual Spider Tables and the MariaDB foreign data wrapper.

See ODBC MariaDB Enterprise Spider Topology.

Read from and write to external ODBC data sources from a Spider Node using virtual Spider Tables and the ODBC foreign data wrapper (Enterprise Server 10.5 and later).

See Sharded MariaDB Enterprise Spider Topology.

Partition a large table across multiple MariaDB Enterprise Server Data Nodes using virtual Spider Tables and standard partitioning syntax for horizontal scalability.

See Also

See MariaDB Enterprise Spider Operations.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Federated MariaDB Enterprise Spider Topology

Federated MariaDB Enterprise Spider Topology

Overview

In the Federated MariaDB Enterprise Spider topology, a Spider Node contains one or more “virtual” Spider Tables. A Spider Table does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses a MariaDB foreign data wrapper to read from and write to a Data Table on a Data Node.

Benefits

MariaDB Enterprise Spider:

- Supports a MariaDB foreign data wrapper. The MariaDB foreign data wrapper can be used to replace the older Federated and FederatedX storage engines.
- Supports an ODBC foreign data wrapper in MariaDB Enterprise Server 10.5 and later. The ODBC foreign data wrapper is beta maturity. The maturity can be confirmed by querying the information_schema.SPIDER_WRAPPER_PROTOCOLS table.

The Spider Federated topology:

- Can be used to query tables located on a different MariaDB Enterprise Server node from the Spider Node using the MariaDB foreign data wrapper.
- Can be used to join tables located on a different MariaDB Enterprise Server node with tables on the Spider Node using the MariaDB foreign data wrapper.
- Can be used to migrate tables located on a different MariaDB Enterprise Server node to the Spider Node using the MariaDB foreign data wrapper.

Federated MariaDB Enterprise Spider Topology

In the Spider Federated topology, a Spider Node contains one or more “virtual” Spider Tables. A Spider Table does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses a MariaDB foreign data wrapper to read from and write to a Data Table on a Data Node.

The Spider Federated topology consists of:

- One MariaDB Enterprise Server node is a Spider Node
- One MariaDB Enterprise Server node is a Data Node

The Spider Node:

- Contains one or more Spider Tables
- Uses the Enterprise Spider storage engine plugin for Spider Tables
- Uses a MariaDB foreign data wrapper to query the Data Table on the Data Node

The Data Node:

- Contains a Data Table for each Spider Table
- Uses a non-Spider storage engine for each Data Table, such as InnoDB or ColumnStore

Term Definitions

Term	Definition
Data Node	A Data Node is a MariaDB Enterprise Server node that contains one or more Data Tables.
Data Table	A Data Table stores data for a Spider Table. When a Spider Table is queried, the Enterprise Spider storage engine uses the MariaDB foreign data wrapper to read from and write to the Data Table on a Data Node. The Data Table must be created on the Data Node with the same structure as the Spider Table. The Data Table must use a non-Spider storage engine, such as InnoDB or ColumnStore.
ODBC Data Source	An ODBC Data Source relies on an ODBC Driver and an ODBC Driver Manager to query an external data source.
ODBC Driver	An ODBC Driver is a library that integrates with a ODBC Driver Manager to query an external data source.
ODBC Driver Manager	An ODBC Driver Manager allows applications to use ODBC Drivers.
Spider Node	A Spider Node is a MariaDB Enterprise Server node that contains one or more Spider Tables.
Spider Table	A Spider Table is a virtual table that does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses foreign data wrappers to read from and write to Data Tables on Data Nodes or ODBC Data Sources.

Example Use Cases

Query MariaDB Enterprise Server Nodes

The Spider Federated topology can be used to query tables located on another MariaDB Enterprise Server node:

- The MariaDB Enterprise Server node with the desired table is configured as a Data Node.
- The MariaDB Enterprise Server node that needs to query the table is configured as a Spider Node.
- The Data Table is the desired table on the Data Node.
- A Spider Table is created on the Spider Node that references the Data Table on the Data Node.
- On the Spider Node, the Data Table is queried by querying the Spider Table like the following:

```
SELECT * FROM spider_tab;
```

- Non-Spider tables can also be referenced in queries with the Spider Table:

```
SELECT *
FROM spider_tab s
JOIN innodb_tab i
ON s.id=i.id;
```

Migrate Tables from MariaDB Enterprise Server Nodes

The Spider Federated topology can be used to migrate tables from one MariaDB Enterprise Server node to another MariaDB Enterprise Server node:

- The MariaDB Enterprise Server node with the source table is configured as a Data Node.
- The MariaDB Enterprise Server node with the destination table is configured as a Spider Node.
- The Data Table is the source table on the Data Node.
- A Spider Table is created on the Spider Node that references the Data Table on the Data Node.
- On the Spider node, the Data Table's data is migrated to the destination table by querying the Spider Table like the following:

```
INSERT INTO innodb_tab
   SELECT * FROM spider_tab;
```

Examples

Load Spider with Configuration File (ES 10.4+)

```
[mariadb]
...
plugin_load_add = "ha_spider"
```

Load Spider with INSTALL SONAME (ES 10.4+)

```
INSTALL SONAME "ha_spider";
```

View Foreign Data Wrappers (ES 10.5+)

```
SELECT * FROM information_schema.SPIDER_WRAPPER_PROTOCOLS;
```

Create Federated Spider Table

```
CREATE SERVER hq_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.2",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "hq_sales"
);

CREATE DATABASE spider_hq_sales;

CREATE TABLE spider_hq_sales.invoices (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
COMMENT='server "hq_server", table "invoices"';
```

Resources

Deployment

- Deploy MariaDB Enterprise Spider

Operations

- Operations
- Backup and Restore
- Migrate Tables

Schema Design

- Schema Design

Storage Engines

- Enterprise Spider Storage Engine

This page is: Copyright © 2026 MariaDB. All rights reserved.

ODBC MariaDB Enterprise Spider Topology

Overview

In the ODBC MariaDB Enterprise Spider topology, a Spider Node contains one or more “virtual” Spider Tables. A Spider Table does not store data. When the Spider Table is queried in this topology, the Enterprise Spider storage engine uses an ODBC foreign data wrapper to read from and write to an ODBC Data Source.

Benefits

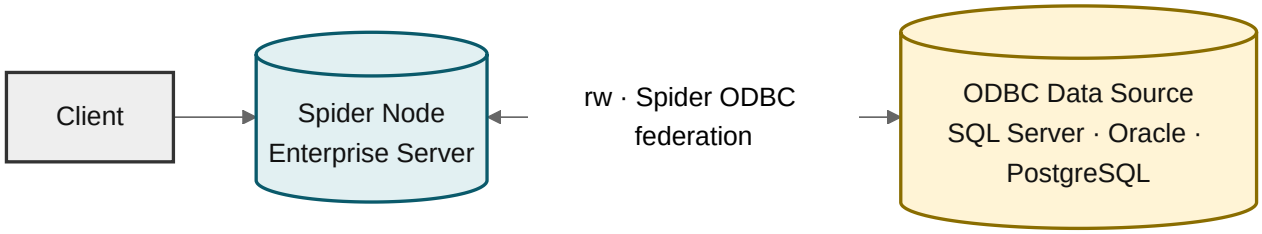
MariaDB Enterprise Spider:

- Supports a MariaDB foreign data wrapper. The MariaDB foreign data wrapper can be used to replace the older Federated and FederatedX storage engines.

- Supports an ODBC foreign data wrapper in MariaDB Enterprise Server 10.5 and later. The maturity can be confirmed by querying the information_schema.SPIDER_WRAPPER_PROTOCOLS table.

The Spider ODBC topology:

- Can be used to query ODBC Data Sources from the Spider Node using the ODBC foreign data wrapper.
- Can be used to join ODBC Data Sources with tables on the Spider Node using the ODBC foreign data wrapper.
- Can be used to migrate table data from ODBC Data Sources to the Spider Node using the ODBC foreign data wrapper.



In the Spider ODBC topology, a Spider Node uses an ODBC foreign data wrapper to read from and write to an external ODBC Data Source.

In the Spider ODBC topology, a Spider Node contains one or more “virtual” Spider Tables. A Spider Table does not store data. When the Spider Table is queried in this topology, the Enterprise Spider storage engine uses an ODBC foreign data wrapper to read from and write to an ODBC Data Source.

MariaDB Enterprise Spider implemented support for the ODBC foreign data wrapper in MariaDB Enterprise Server 10.5.

The maturity can be confirmed by querying the information_schema.SPIDER_WRAPPER_PROTOCOLS table.

The Spider ODBC topology consists of:

- One MariaDB Enterprise Server node is a Spider Node
- One ODBC Data Source stores data for Spider Tables

The Spider Node:

- Contains one or more Spider Tables
- Uses the Enterprise Spider storage engine plugin for Spider Tables
- Uses an ODBC foreign data wrapper to query the ODBC Data Source

The ODBC Data Source:

- Contains the Data for each Spider Table
- Can be a non-MariaDB database server, such as Microsoft SQL Server, Oracle, or PostgreSQL

Term Definitions

Term	Definition
Data Node	A Data Node is a MariaDB Enterprise Server node that contains one or more Data Tables.
Data Table	A Data Table stores data for a Spider Table. When a Spider Table is queried, the Enterprise Spider storage engine uses the MariaDB foreign data wrapper to read from and write to the Data Table on a Data Node. The Data Table must be created on the Data Node with the same structure as the Spider Table. The Data Table must use a non-Spider storage engine, such as InnoDB or ColumnStore.

Term	Definition
ODBC Data Source	An ODBC Data Source relies on an ODBC Driver and an ODBC Driver Manager to query an external data source.
ODBC Driver	An ODBC Driver is a library that integrates with a ODBC Driver Manager to query an external data source.
ODBC Driver Manager	An ODBC Driver Manager allows applications to use ODBC Drivers.
Spider Node	A Spider Node is a MariaDB Enterprise Server node that contains one or more Spider Tables.
Spider Table	A Spider Table is a virtual table that does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses foreign data wrappers to read from and write to Data Tables on Data Nodes or ODBC Data Sources.

Example Use Cases

Query Non-MariaDB Databases

The Spider ODBC topology can be used to query tables located on non-MariaDB databases:

- The non-MariaDB database is configured as an ODBC Data Source in the ODBC Driver Manager.
- The MariaDB Enterprise Server node that needs to query the ODBC Data Source is configured as a Spider Node.
- A Spider Table is created on the Spider Node that references the ODBC Data Source.
- On the Spider Node, the ODBC Data Source is queried by querying the Spider Table like the following:

```
SELECT * FROM spider_tab;
```

- Local tables can also be referenced in queries:

```
SELECT *
FROM spider_tab s
JOIN local_tab l
ON s.id=l.id;
```

Migrate Tables from Non-MariaDB Databases

- The Spider ODBC topology can be used to migrate tables from a non-MariaDB database to a MariaDB Enterprise Server node:
- The non-MariaDB database is configured as an ODBC Data Source in the ODBC Driver Manager.
- The MariaDB Enterprise Server node with the destination table is configured as a Spider Node.
- A Spider Table is created on the Spider Node that references the ODBC Data Source.
- On the Spider Node, the ODBC Data Source's data is migrated to the destination table by querying the Spider Table like the following:

```
INSERT INTO destination_tab
SELECT * FROM spider_tab;
```

Examples

Load Spider with Configuration File (ES 10.4+)

```
[mariadb]
...
plugin_load_add = "ha_spider"
```

Load Spider with INSTALL SONAME (ES 10.4+)

```
INSTALL SONAME "ha_spider";
```

View Foreign Data Wrappers (ES 10.5+)

```
SELECT * FROM information_schema.SPIDER_WRAPPER_PROTOCOLS;
```

Create ODBC Spider Table (with an Oracle remote server)

Follow the link under Operations for further information on the setup.

```
INSTALL SONAME 'ha_spider';
CREATE DATABASE spider_test;
USE spider_test;
CREATE OR REPLACE TABLE contacts (
    contact_id  BIGINT NOT NULL PRIMARY KEY,
    first_name  VARCHAR(255) NOT NULL,
    last_name   VARCHAR(255) NOT NULL,
    email       VARCHAR(255) NOT NULL,
    phone       VARCHAR(20),
    customer_id BIGINT
) ENGINE=SPIDER COMMENT='wrapper "odbc", dsn "ORARDS", table "CONTACTS"';
```

Resources

Schema Design

- Schema Design

Operations

- Use Spider ODBC to connect to Oracle

Storage Engines

- Enterprise Spider Storage Engine

This page is: Copyright © 2026 MariaDB. All rights reserved.

Sharded MariaDB Enterprise Spider Topology

Sharded MariaDB Enterprise Spider Topology

Overview

In the Sharded MariaDB Enterprise Spider topology, a Spider Node contains one or more “virtual” Spider Tables. A Spider Table does not store data. When a Spider Table is queried in this topology, the Enterprise Spider storage engine uses a MariaDB foreign data wrapper to read from and write to Data Tables on Data Nodes. The data for the Spider Table is partitioned among the Data Nodes using the regular partitioning syntax.

Benefits

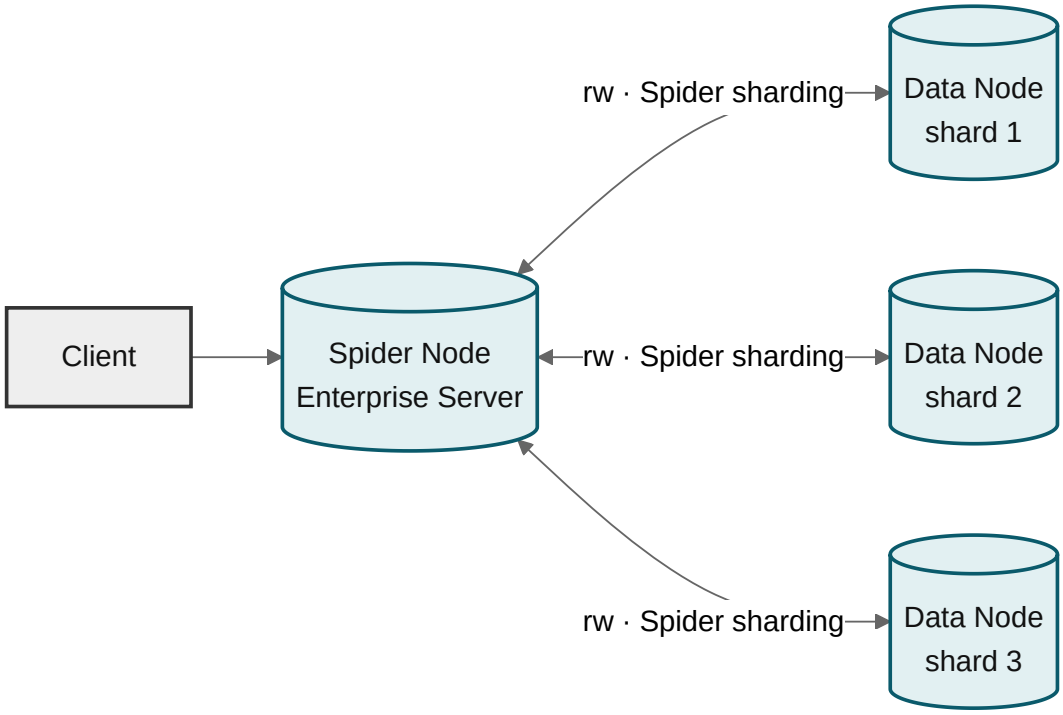
MariaDB Enterprise Spider:

- Supports a MariaDB foreign data wrapper. The MariaDB foreign data wrapper can be used to replace the older Federated and FederatedX storage engines.
- Supports an ODBC foreign data wrapper in MariaDB Enterprise Server 10.5 and later. The ODBC foreign data wrapper was backported to MariaDB Enterprise Server in a previous version. The ODBC foreign data wrapper is beta maturity. The maturity can be confirmed by querying the information_schema.SPIDER_WRAPPER_PROTOCOLS table.

The Spider Sharded topology:

- Can be used to consolidate multiple tables on multiple MariaDB Enterprise Server nodes into a single "virtual" table on the Spider Node using the MariaDB foreign data wrapper.
- Can be used to partition a large table across multiple MariaDB Enterprise Server nodes for horizontal scalability using the MariaDB foreign data wrapper.
- Defines Sharded Spider Tables with MariaDB Enterprise Server's regular partitioning syntax.

Sharded MariaDB Enterprise Spider Topology



Sharded Spider: a Spider Node distributes the partitions of a virtual sharded table across multiple Data Nodes (shards) via the Spider foreign data wrapper.

In the Spider Sharded topology, a Spider Node contains one or more "virtual" Spider Tables. A Spider Table does not store data. When a Spider Table is queried in this topology, the Enterprise Spider storage engine uses a MariaDB foreign data wrapper to read from and write to Data Tables on Data Nodes. The data for the Spider Table is partitioned among the Data Nodes using the regular partitioning syntax.

The Spider Sharded topology consists of:

- One MariaDB Enterprise Server node is a Spider Node
- One or more MariaDB Enterprise Server nodes are Data Nodes

The Spider Node:

- Contains one or more partitioned Spider Tables
- Uses the Enterprise Spider storage engine plugin for Spider Tables
- Uses a MariaDB foreign data wrapper to query the Data Tables on the Data Nodes

The Data Nodes:

- Contain Data Tables for one or more partitions of the Spider Table
- Use a non-Spider storage engine for each Data Table, such as InnoDB or ColumnStore

Term Definitions

Term	Definition
Data Node	A Data Node is a MariaDB Enterprise Server node that contains one or more Data Tables.
Data Table	A Data Table stores data for a Spider Table. When a Spider Table is queried, the Enterprise Spider storage engine uses the MariaDB foreign data wrapper to read from and write to the Data Table on a Data Node. The Data Table must be created on the Data Node with the same structure as the Spider Table. The Data Table must use a non-Spider storage engine, such as InnoDB or ColumnStore.
ODBC Data Source	An ODBC Data Source relies on an ODBC Driver and an ODBC Driver Manager to query an external data source.
ODBC Driver	An ODBC Driver is a library that integrates with a ODBC Driver Manager to query an external data source.
ODBC Driver Manager	An ODBC Driver Manager allows applications to use ODBC Drivers.
Spider Node	A Spider Node is a MariaDB Enterprise Server node that contains one or more Spider Tables.
Spider Table	A Spider Table is a virtual table that does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses foreign data wrappers to read from and write to Data Tables on Data Nodes or ODBC Data Sources.

Example Use Cases

Shard Big Tables

The Spider Sharded topology can be used to split table data into multiple shards stored on remote MariaDB Enterprise Server nodes for horizontal scalability:

- One MariaDB Enterprise Server node is configured as a Spider Node and accepts application queries.
- One or more MariaDB Enterprise Server nodes are configured as Data Nodes and store shards.

Consolidate Data for Multi-Location Businesses

The Spider Sharded topology can be used to implement a consolidated view of multiple remote databases:

- One MariaDB Enterprise Server is configured as a Spider Node and provides a consolidated view using Spider Tables.
- One or more MariaDB Enterprise Server nodes are configured as Data Nodes and contain the local data.

Examples

Load Spider with Configuration File

```
[mariadb]
...
plugin_load_add = "ha_spider"
```

Load Spider with INSTALL SONAME

```
INSTALL SONAME "ha_spider";
```

View Foreign Data Wrappers (ES 10.5+)

```
SELECT * FROM information_schema.SPIDER_WRAPPER_PROTOCOLS;
```

Create Sharded Spider Table

```
CREATE SERVER hq_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.2",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "hq_sales"
);

CREATE SERVER eastern_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.3",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "eastern_sales"
);

CREATE SERVER western_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.4",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "western_sales"
);

CREATE DATABASE spider_sharded_sales;

CREATE TABLE spider_sharded_sales.invoices (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2)
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
PARTITION BY LIST(branch_id) (
   PARTITION hq_partition VALUES IN (1) COMMENT = 'server "hq_server", table "invoices"',
   PARTITION eastern_partition VALUES IN (2) COMMENT = 'server "eastern_server", table "invoices"',
   PARTITION western_partition VALUES IN (3) COMMENT = 'server "western_server", table "invoices"'
);
```

Resources

Deployment

- Deploy MariaDB Enterprise Spider

Schema Design

- Schema Design

Operations

- Operations
- Add a Shard
- Backup and Restore

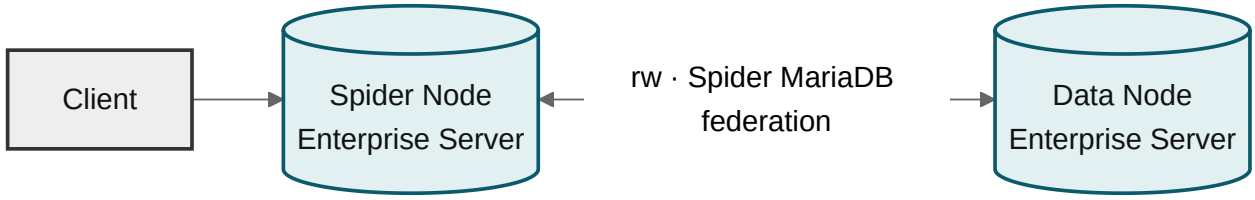
Storage Engines

- Enterprise Spider Storage Engine

This page is: Copyright © 2026 MariaDB. All rights reserved.

Spider Federated

Overview



In the Spider Federated topology, a Spider Node uses the MariaDB Spider federation to read from and write to a Data Table on a separate Data Node.

- Enterprise Server 10.4
- Enterprise Server 10.5
- Enterprise Server 10.6
- Enterprise Server 11.4

Read from and write to tables on remote ES nodes

- Spider Node uses Spider storage engine for Federated Spider Tables
- Federated Spider Table is a “virtual” table
- Spider uses MariaDB foreign data wrapper to query Data Table on Data Node
- Data Node uses non-Spider storage engine for Data Tables
- Supports transactions
- Enterprise Server 10.3+, Enterprise Spider

This procedure describes the deployment of the **Spider Federated topology** with MariaDB Enterprise Server

This procedure incrementally deploys MariaDB Enterprise Spider on an existing MariaDB Enterprise Server deployment.

In the Spider Federated topology, a Spider Node contains one or more “virtual” Spider Tables. A Spider Table does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses a MariaDB foreign data wrapper to read from and write to a Data Table on a Data Node.

This procedure has 3 steps, which are executed in sequence.

This page provides an overview of the topology, requirements, and deployment procedure.

The topology described is representative of basic product capabilities. MariaDB products can be deployed to form other topologies, leverage advanced product capabilities, or combine the capabilities of multiple topologies.

Prerequisites

If you have not yet deployed MariaDB Enterprise Server on the Spider Node and Data Node, first deploy a topology containing MariaDB Enterprise Server. Several topologies are documented.

Procedure Steps

Step	Description
Step 1	Install Enterprise Spider
Step 2	Configure Spider Node and Data Node
Step 3	Test Spider Federated Topology

Support

Customers can obtain support by submitting a support case.

Components

The following components are deployed during this procedure:

Component	Function
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.
MariaDB Enterprise Spider	Storage engine used by Spider Tables to read from and write to Data Tables using the MariaDB foreign data wrapper.

Term Definitions

Term	Definition
Data Node	A Data Node is a MariaDB Enterprise Server node that contains one or more Data Tables.
Data Table	A Data Table stores data for a Spider Table. When a Spider Table is queried, the Enterprise Spider storage engine uses the MariaDB foreign data wrapper to read from and write to the Data Table on a Data Node. The Data Table must be created on the Data Node with the same structure as the Spider Table. The Data Table must use a non-Spider storage engine, such as InnoDB or ColumnStore.
ODBC Data Source	An ODBC Data Source relies on an ODBC Driver and an ODBC Driver Manager to query an external data source.
ODBC Driver	An ODBC Driver is a library that integrates with a ODBC Driver Manager to query an external data source.
ODBC Driver Manager	An ODBC Driver Manager allows applications to use ODBC Drivers.
Spider Node	A Spider Node is a MariaDB Enterprise Server node that contains one or more Spider Tables.

Term	Definition
Spider Node	A Spider Table is a virtual table that does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses foreign data wrappers to read from and write to Data Tables on Data Nodes or ODBC Data Sources.

Topology

In the Spider Federated topology, a Spider Node contains one or more “virtual” Spider Tables. A Spider Table does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses a MariaDB foreign data wrapper to read from and write to a Data Table on a Data Node.

The Spider Federated topology consists of:

- One MariaDB Enterprise Server node is a Spider Node
- One MariaDB Enterprise Server node is a Data Node

The Spider Node:

- Contains one or more Spider Tables
- Uses the Enterprise Spider storage engine plugin for Spider Tables
- Uses a MariaDB foreign data wrapper to query the Data Table on the Data Node

The Data Node:

- Contains a Data Table for each Spider Table
- Uses a non-Spider storage engine for each Data Table, such as InnoDB or ColumnStore

For additional information, see “Spider Federated Topology Operations”.

Requirements

These requirements are for the Spider Federated topology when deployed with MariaDB Enterprise Server

- Node Count
- Operating Systems

Node Count

- One or more MariaDB Enterprise Server nodes must be deployed as Spider Nodes. The Spider Nodes contain Spider Tables.
- One or more MariaDB Enterprise Server nodes must be deployed as Data Nodes. The Data Nodes contain Data Tables.

Operating Systems

In alignment to the enterprise lifecycle, the Spider Federated topology with MariaDB Enterprise Server is provided for:

- AlmaLinux 8 (x86_64, ARM64)
- AlmaLinux 9 (x86_64, ARM64)
- Debian 11 (x86_64, ARM64)
- Debian 12 (x86_64, ARM64)
- Microsoft Windows (x86_64)
- Red Hat Enterprise Linux 8 (x86_64, ARM64)
- Red Hat Enterprise Linux 9 (x86_64, PPC64LE, ARM64)
- Red Hat UBI 8 (x86_64, ARM64)

- Rocky Linux 8 (x86_64, ARM64)
- Rocky Linux 9 (x86_64, ARM64)
- SUSE Linux Enterprise Server 15 (x86_64, ARM64)
- Ubuntu 20.04 LTS (x86_64, ARM64)
- Ubuntu 22.04 LTS (x86_64, ARM64)
- Ubuntu 24.04 LTS (x86_64, ARM64)

Next Step

Navigation in the procedure “Deploy Spider Federated Topology”:

Next: Step 1: Install Enterprise Spider

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 1: Install Enterprise Spider

Overview

This page details step 1 of the 3-step procedure “Deploy Spider Federated Topology”.

This step installs the Enterprise Spider storage engine plugin on the Spider Node.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Allow Interconnect

MariaDB Enterprise Spider depends on interconnect between the Spider Node and the Data Node. This may require adjustment to firewall and security settings.

Install Spider Package

The Enterprise Spider storage engine plugin is not installed with MariaDB Enterprise Server by default. An additional package must be installed.

Install via YUM (CentOS, RHEL)

On the Spider Node, install MariaDB Enterprise Spider:

```
$ sudo yum install MariaDB-spider-engine
```

Install via APT (Debian, Ubuntu)

On the Spider Node, install MariaDB Enterprise Spider:

```
$ sudo apt install mariadb-plugin-spider
```

Install via ZYpp (SLES)

On the Spider Node, install MariaDB Enterprise Spider:

```
$ sudo zypper install MariaDB-spider-engine
```

Load the Spider Plugin

The Enterprise Spider storage engine plugin must be loaded by MariaDB Enterprise Server.

On the Spider Node, use one of the following methods to configure MariaDB Enterprise Server to load the Enterprise Spider storage engine plugin:

Interface	Method	Benefits
Shell	Configuration File	• SQL access is not required • SUPER privilege is not required • Configuration file can be version controlled
SQL	INSTALL SONAME Statement	• Shell access is not required • File system privileges on configuration file are not required • Plugin is included in backup of mysql.plugin system table • Spider Node restart is not required

Load Spider with Configuration File

On the Spider Node, set the `plugin_load_add` option to `ha_spider` in a configuration file. This option configures MariaDB Enterprise Server to load the Enterprise Spider storage engine plugin. The Spider Node must be restarted to detect the configuration change.

1. Choose a configuration file for custom changes to system variables and options.

   It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

   Ensure that your custom changes will be read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the z- prefix in the file name.

   Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
• CentOS • RHEL • Rocky Linux • SLES	`/etc/my.cnf.d/z-custom-mariadb.cnf`
• Debian • Ubuntu	`/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf`

2. Set the `plugin_load_add` option in the configuration file.

   It must be set in a group that will be read by MariaDB Server, such as `[mariadb]` or `[server]`.

   For example:

```
[mariadb]
...
plugin_load_add = "ha_spider"
```

3. Restart MariaDB Enterprise Server:

```
$ sudo systemctl restart mariadb
```

Load Spider with INSTALL SONAME

On the Spider Node, execute the INSTALL SONAME statement with the library name `ha_spider`. The `INSTALL SONAME` statement configures MariaDB Enterprise Server to load the Enterprise Spider storage engine plugin. The `INSTALL SONAME` statement requires the SUPER privilege.

The `INSTALL SONAME` statement adds the Enterprise Spider storage engine to the mysql.plugin system table. When the Spider Node is restarted, MariaDB Enterprise Server reads the system table and reloads the plugin, so the statement only needs to be executed once.

1. Connect to the Spider Node using mariadb Client:

```
$ sudo mariadb
```

2. Use the INSTALL SONAME statement to install the Enterprise Spider storage engine plugin:

```
INSTALL SONAME "ha_spider";
```

Test Plugin Installation

On the Spider Node, confirm that the Enterprise Spider storage engine plugin is loaded by querying the information_schema.PLUGINS table:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS
FROM information_schema.PLUGINS
WHERE PLUGIN_LIBRARY LIKE "ha_spider%";

+--------------------------+---------------+
| PLUGIN_NAME              | PLUGIN_STATUS |
+--------------------------+---------------+
| SPIDER                   | ACTIVE        |
| SPIDER_ALLOC_MEM         | ACTIVE        |
| SPIDER_WRAPPER_PROTOCOLS | ACTIVE        |
+--------------------------+---------------+
```

When the Enterprise Spider storage engine is loaded, the `PLUGIN_NAME` column contains the value `SPIDER` and the `PLUGIN_STATUS` column contains the value `ACTIVE`.

Next Step

Navigation in the procedure “Deploy Spider Federated Topology”.

This page was step 1 of 3.

Next: Step 2: Configure Spider Node and Data Node

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 2: Configure Spider Node and Data Node

Overview

This page details step 2 of the 3-step procedure “Deploy Spider Federated Topology”.

This step configures the Spider Node and Data Node and creates the Spider Table and Data Table.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Create Spider User

The data node requires a user account that the Spider Node uses to connect.

On the Data Node, create the Spider user account for the Spider Node using the CREATE USER statement:

```
CREATE USER spider_user@192.0.2.1 IDENTIFIED BY "password";
```

Privileges will be granted to the user account in Grant Privileges on the Data Table.

Test Spider User

On the Spider Node, confirm that the Spider user account can connect to the Data Node using MariaDB Client:

```
$ mariadb --user spider_user --host 192.0.2.2 --password
```

Configure Connection Details

The Spider Node requires connection details for the Data Node.

On the Spider Node, create a server object to configure the connection details for the Data Node using the CREATE SERVER statement:

```
CREATE SERVER hq_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.2",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "hq_sales"
);
```

The Data Node runs MariaDB Enterprise Server, so the `FOREIGN DATA WRAPPER` is set to `mariadb`.

Using a server object for connection details is optional. Alternatively, the connection details for the Data Node can be specified in the `COMMENT` table option of the CREATE TABLE statement when creating the Spider Table.

Create the Data Table

When queries read and write to a Spider Table, Spider reads and writes to the Data Table on the Data Node. The Data Table must be created on the Data Node with the same structure as the Spider Table.

If your Data Table already exists, grant privileges on the table to the Spider user.

On the Data Node, create the Data Table:

```
CREATE DATABASE hq_sales;

CREATE SEQUENCE hq_sales.invoice_seq;

CREATE TABLE hq_sales.invoices (
   branch_id INT NOT NULL DEFAULT (1) CHECK (branch_id=1),
   invoice_id INT NOT NULL DEFAULT (NEXT VALUE FOR hq_sales.invoice_seq),
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=InnoDB;

INSERT INTO hq_sales.invoices
   (customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, '2020-05-10 12:35:10', 1087.23, 'CREDIT_CARD'),
   (2, '2020-05-10 14:17:32', 1508.57, 'WIRE_TRANSFER'),
   (3, '2020-05-10 14:25:16', 227.15, 'CASH');
```

The Spider Node reads and writes to the Data Table using the server and user account configured in "Configure Connection Details". The user account must have privileges on the tables.

Grant Privileges

The Spider Node connects to the Data Node with the user account configured in "Create Spider User".

On the Data Node, grant the Spider user sufficient privileges to operate on the Data Table:

```
GRANT ALL PRIVILEGES ON hq_sales.invoices TO 'spider_user'@'192.0.2.1';
```

Privileges for Spider BKA Mode

By default, the Spider user also requires the CREATE TEMPORARY TABLES privilege on the database containing the Data Table. The `CREATE TEMPORARY TABLES` privilege is required, because Spider uses temporary tables to optimize read queries when Spider BKA Mode is `1`.

Spider BKA Mode is configured using the following methods:

- The session value is configured by setting the spider_bka_mode system variable on the Spider Node. The default value is `-1`. When the session value is `-1`, the value for each Spider Table is used.
- The value for each Spider Table is configured by setting the `bka_mode` option in the `COMMENT` table option. When the `bka_mode` option is not set, the implicit value is `1`.

The default spider_bka_mode value is `-1`, and the implicit Spider Table value is `1`, so the default Spider BKA Mode is `1`.

On the Data Node, grant the Spider user the `CREATE TEMPORARY TABLES` privilege on the database:

```
GRANT CREATE TEMPORARY TABLES ON hq_sales.* TO 'spider_user'@'192.0.2.1';
```

Create the Spider Table

The Spider Table must be created on the Spider Node with the same structure as the Data Table.

On the Spider Node, create the Spider Table and reference the Data Node in the `COMMENT` table option:

```
CREATE DATABASE spider_hq_sales;

CREATE TABLE spider_hq_sales.invoices (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
COMMENT='server "hq_server", table "invoices"';
```

The `COMMENT` table option is used to configure the Data Node and the Data Table. Set the server option to the server object configured in "Configure Connection Details". Set the `table` option to the Data Table.

An alternative syntax is available. When you don't want to create a server object, the connection details for the Data Node can be specified in the `COMMENT` table option:

```
CREATE TABLE spider_hq_sales.invoices_alternate (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
COMMENT='table "invoices", host "192.0.2.2", port "5801", user "spider_user", password "user_password",
database "hq_sales"';
```

Next Step

Navigation in the procedure “Deploy Spider Federated Topology”:

This page was step 2 of 3.

Next: Step 3: Test Spider Federated Topology.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 3: Test Spider Federated Topology

Overview

This page details step 3 of the 3-step procedure “Deploy Spider Federated Topology”.

This step tests MariaDB Enterprise Spider.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Test Enterprise Server Service

Use Systemd to test whether the MariaDB Enterprise Server service is running.

This action is performed **on the Spider Node and Data Node**.

Check if the MariaDB Enterprise Server service is running by executing the following:

```
$ systemctl status mariadb
```

If the service is not running on any node, start the service by executing the following on that node:

```
$ sudo systemctl start mariadb
```

Test Local Client Connections

Use MariaDB Client to test the local connection to the Enterprise Server node.

This action is performed on the Spider Node and Data Node:

```
$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The sudo command is used here to connect to the Enterprise Server node using the `root@localhost` user account, which authenticates using the `unix_socket` authentication plugin. Other user accounts can be used by specifying the `--user` and `--password` command-line options.

Test Data Node Client Connection

Use mariadb Client to test a client connection to the Data Node from the Spider Node using the Spider user.

This action is performed **on the Spider Node**:

```
$ mariadb \
   --host 192.0.2.2 \
   --user spider_user \
   --password
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The host and port of the Data Node can be provided using the `--host` and `--port` command-line options. The credentials for the Spider user can be provided using the `--user` and `--password` command-line options.

If the Spider user is unable to connect to the Data Node from the Spider Node, check the password for the Spider user account on the Data Node.

For additional information, see "Create Spider User".

Test Spider Storage Engine Plugin

Query the information_schema.PLUGINS table to confirm that the Enterprise Spider storage engine is loaded.

This action is performed **on the Spider Node**.

Execute the following query:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS
FROM information_schema.PLUGINS
WHERE PLUGIN_LIBRARY LIKE 'ha_spider%';

+--------------------------+---------------+
| PLUGIN_NAME              | PLUGIN_STATUS |
+--------------------------+---------------+
| SPIDER                   | ACTIVE        |
| SPIDER_ALLOC_MEM         | ACTIVE        |
| SPIDER_WRAPPER_PROTOCOLS | ACTIVE        |
+--------------------------+---------------+
```

The `PLUGIN_STATUS` column for each Spider-related plugin should contain `ACTIVE`.

For additional information, see "Load the Spider Plugin".

Test Write Operations

Write to the Spider Table using an INSERT statement to test write operations.

This action is performed **on the Spider Node**.

Execute the following query:

```
INSERT INTO spider_hq_sales.invoices
   (branch_id, invoice_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES (1, 4, 1, '2021-03-10 12:45:10', 3045.73, 'CREDIT_CARD');
```

Test Read Operations

Read from the Spider Table using a SELECT statement to test read operations.

This action is performed **on the Spider Node**.

Execute the following query:

```
SELECT * FROM spider_hq_sales.invoices;

+-----------+------------+-------------+----------------------------+---------------+----------------+
| branch_id | invoice_id | customer_id | invoice_date               | invoice_total | payment_method |
+-----------+------------+-------------+----------------------------+---------------+----------------+
|         1 |          1 |           1 | 2020-05-10 12:35:10.000000 |       1087.23 | CREDIT_CARD    |
|         1 |          2 |           2 | 2020-05-10 14:17:32.000000 |       1508.57 | WIRE_TRANSFER  |
|         1 |          3 |           3 | 2020-05-10 14:25:16.000000 |        227.15 | CASH           |
|         1 |          4 |           1 | 2021-03-10 12:45:10.000000 |       3045.73 | CREDIT_CARD    |
+-----------+------------+-------------+----------------------------+---------------+----------------+
```

Next Step

Navigation in the procedure “Deploy Spider Federated Topology”.

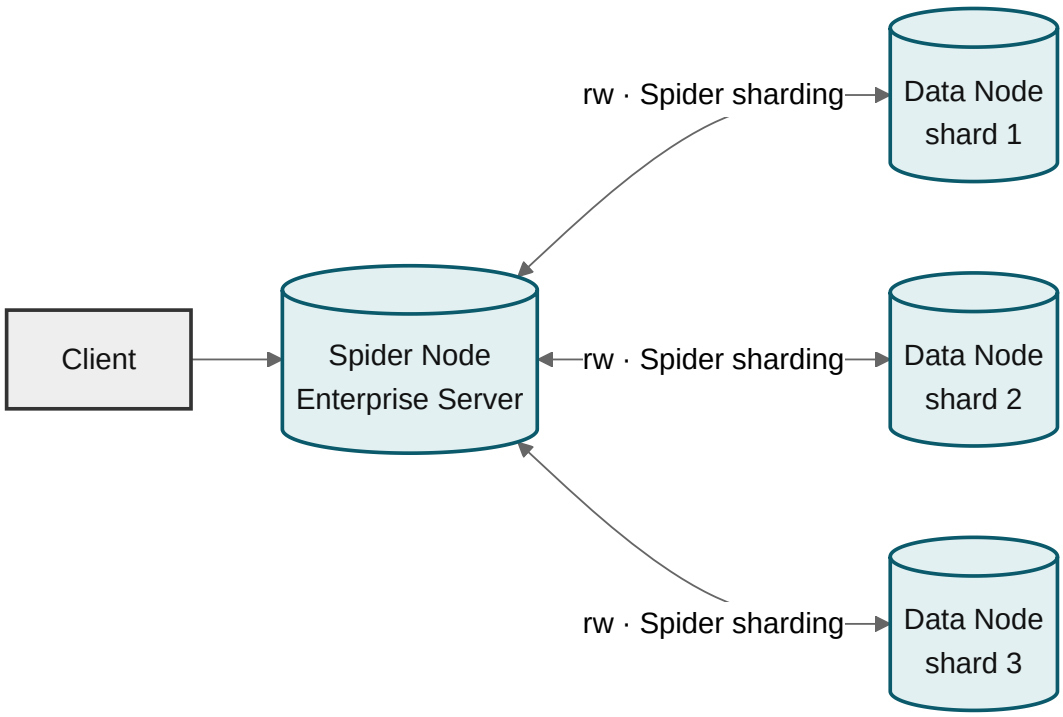
This page was step 3 of 3.

This procedure is complete.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Spider Sharded

Overview



Spider Sharded: a Spider Node distributes the partitions of a virtual sharded table across multiple Data Nodes (shards) via the Spider foreign data wrapper.

- Enterprise Server 10.4
- Enterprise Server 10.5
- Enterprise Server 10.6
- Enterprise Server 11.4

Shard tables for horizontal scalability

- Spider Node uses Spider storage engine for Sharded Spider Tables
- Sharded Spider Table is a partitioned “virtual” table
- Spider uses MariaDB foreign data wrapper to query Data Tables on Data Nodes for each partition
- Data Node uses non-Spider storage engine for Data Tables
- Supports transactions
- Enterprise Server 10.3+, Enterprise Spider

This procedure describes the deployment of the **Spider Sharded topology** with MariaDB Enterprise Server.

This procedure incrementally deploys MariaDB Enterprise Spider on an existing MariaDB Enterprise Server deployment.

In the Spider Sharded topology, a Spider Node contains one or more “virtual” Spider Tables. A Spider Table does not store data. When a Spider Table is queried in this topology, the Enterprise Spider storage engine uses a MariaDB foreign data wrapper to read from and write to Data Tables on Data Nodes. The data for the Spider Table is partitioned among the Data Nodes using the regular partitioning syntax.

This procedure has 3 steps, which are executed in sequence.

This page provides an overview of the topology, requirements, and deployment procedure.

The topology described is representative of basic product capabilities. MariaDB products can be deployed to form other topologies, leverage advanced product capabilities, or combine the capabilities of multiple topologies.

Prerequisites

If you have not yet deployed MariaDB Enterprise Server on the Spider Node and Data Nodes, first deploy a topology containing MariaDB Enterprise Server. Several topologies are documented.

Procedure Steps

Step	Description
Step 1	Install Enterprise Spider
Step 2	Configure Spider Node and Data Nodes
Step 3	Test Spider Sharded Topology

Support

Customers can obtain support by submitting a support case.

Components

The following components are deployed during this procedure:

Component	Function
MariaDB Enterprise Server	Modern SQL RDBMS with high availability, pluggable storage engines, hot online backups, and audit logging.
MariaDB Enterprise Spider	Storage engine used by Spider Tables to read from and write to Data Tables using the MariaDB foreign data wrapper.

Term Definitions

Term	Definition
Data Node	A Data Node is a MariaDB Enterprise Server node that contains one or more Data Tables.
Data Table	A Data Table stores data for a Spider Table. When a Spider Table is queried, the Enterprise Spider storage engine uses the MariaDB foreign data wrapper to read from and write to the Data Table on a Data Node. The Data Table must be created on the Data Node with the same structure as the Spider Table. The Data Table must use a non-Spider storage engine, such as InnoDB or ColumnStore.
ODBC Data Source	An ODBC Data Source relies on an ODBC Driver and an ODBC Driver Manager to query an external data source.
ODBC Driver	An ODBC Driver is a library that integrates with a ODBC Driver Manager to query an external data source.
ODBC Driver Manager	An ODBC Driver Manager allows applications to use ODBC Drivers.
Spider Node	A Spider Node is a MariaDB Enterprise Server node that contains one or more Spider Tables.
Spider Node	A Spider Table is a virtual table that does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses foreign data wrappers to read from and write to Data Tables on Data Nodes or ODBC Data Sources.

Topology

In the Spider Sharded topology, a Spider Node contains one or more “virtual” Spider Tables. A Spider Table does not store data. When a Spider Table is queried, the Enterprise Spider storage engine uses a MariaDB foreign data wrapper to read from and write to Data Tables on Data Nodes. The data for the Spider Table is partitioned among the Data Nodes using the regular partitioning syntax.

The Spider Sharded topology consists of:

- One MariaDB Enterprise Server node is a Spider Node
- One or more MariaDB Enterprise Server nodes are Data Nodes

The Spider Node:

- Contains one or more Spider Tables
- Uses the Enterprise Spider storage engine plugin for Spider Tables
- Uses a MariaDB foreign data wrapper to query the Data Tables on the Data Nodes

The Data Nodes:

- Contain Data Tables for one or more partitions of the Spider Table
- Uses a non-Spider storage engine for each Data Table, such as InnoDB or ColumnStore

For additional information, see “Spider Sharded Topology”.

Requirements

These requirements are for the Spider Sharded topology when deployed with MariaDB Enterprise Server

- Node Count

- Operating Systems

Node Count

- One or more MariaDB Enterprise Server nodes must be deployed as Spider Nodes. The Spider Nodes contain Spider Tables.
- One or more MariaDB Enterprise Server nodes must be deployed as Data Nodes. The Data Nodes contain Data Tables.

Operating Systems

In alignment to the enterprise lifecycle, the Spider Sharded topology with MariaDB Enterprise Server is provided for:

- AlmaLinux 8 (x86_64, ARM64)
- AlmaLinux 9 (x86_64, ARM64)
- Debian 11 (x86_64, ARM64)
- Debian 12 (x86_64, ARM64)
- Microsoft Windows (x86_64)
- Red Hat Enterprise Linux 8 (x86_64, ARM64)
- Red Hat Enterprise Linux 9 (x86_64, PPC64LE, ARM64)
- Red Hat UBI 8 (x86_64, ARM64)
- Rocky Linux 8 (x86_64, ARM64)
- Rocky Linux 9 (x86_64, ARM64)
- SUSE Linux Enterprise Server 15 (x86_64, ARM64)
- Ubuntu 20.04 LTS (x86_64, ARM64)
- Ubuntu 22.04 LTS (x86_64, ARM64)
- Ubuntu 24.04 LTS (x86_64, ARM64)

Next Step

Navigation in the procedure “Deploy Spider Sharded Topology”:

Next: Step 1: Install Enterprise Spider

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 1: Install Enterprise Spider

Overview

This page details step 1 of the 3-step procedure “Deploy Spider Sharded Topology”.

This step installs the Enterprise Spider storage engine plugin on the Spider Node.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Allow Interconnect

MariaDB Enterprise Spider depends on interconnect between the Spider Node and all Data Nodes. This may require adjustment to firewall and security settings.

Install Spider Package

The Enterprise Spider storage engine plugin is not installed with MariaDB Enterprise Server by default. An additional package must be installed.

Install via YUM (CentOS, RHEL)

On the Spider Node, install MariaDB Enterprise Spider:

```
$ sudo yum install MariaDB-spider-engine
```

Install via APT (Debian, Ubuntu)

On the Spider Node, install MariaDB Enterprise Spider:

```
$ sudo apt install mariadb-plugin-spider
```

Install via ZYpp (SLES)

On the Spider Node, install MariaDB Enterprise Spider:

```
$ sudo zypper install MariaDB-spider-engine
```

Load the Spider Plugin

The Enterprise Spider storage engine plugin must be loaded by MariaDB Enterprise Server

On the Spider Node, use one of the following methods to configure MariaDB Enterprise Server to load the Enterprise Spider storage engine plugin:

Interface	Method	Benefits
Shell	Configuration File	• SQL access is not required • SUPER privilege is not required • Configuration file can be version controlled
SQL	INSTALL SONAME Statement	• Shell access is not required • File system privileges on configuration file are not required • Plugin is included in backup of mysql.plugin system table • Spider Node restart is not required

Load Spider with Configuration File

On the Spider Node, set the `plugin_load_add` option to `ha_spider` in a configuration file. The plugin_load_add option configures MariaDB Enterprise Server to load the Enterprise Spider storage engine plugin. The Spider Node must be restarted to detect the configuration change.

1. Choose a configuration file for custom changes to system variables and options.
   It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

   Ensure that your custom changes will be read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Ensure that your custom configuration file is read last by using the z- prefix in the file name.

   Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
• CentOS • RHEL • Rocky Linux • SLES	`/etc/my.cnf.d/z-custom-mariadb.cnf`
• Debian • Ubuntu	`/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf`

2. Set the plugin_load_add option in the configuration file.

   It must be set in a group that will be read by MariaDB Server, such as `[mariadb]` or `[server]`.

   For example:

```
[mariadb]
...
plugin_load_add = "ha_spider"
```

3. Restart MariaDB Enterprise Server:

```
$ sudo systemctl restart mariadb
```

Load Spider with INSTALL SONAME

On the Spider Node, execute the INSTALL SONAME statement with the library name `ha_spider`. The `INSTALL SONAME` statement configures MariaDB Enterprise Server to load the Enterprise Spider storage engine plugin. The `INSTALL SONAME` statement requires the SUPER privilege.

The `INSTALL SONAME` statement adds the Enterprise Spider storage engine to the mysql.plugin system table. When the Spider Node is restarted, MariaDB Enterprise Server reads the system table and reloads the plugin, so the statement only needs to be executed once.

1. Connect to the Spider Node using MariaDB Client:

```
$ sudo mariadb
```

2. Use the INSTALL SONAME statement to install the Enterprise Spider storage engine plugin:

```
INSTALL SONAME "ha_spider";
```

Test Plugin Installation

On the Spider Node, confirm that the Enterprise Spider storage engine plugin is loaded by querying the information_schema.PLUGINS table:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS
FROM information_schema.PLUGINS
WHERE PLUGIN_LIBRARY LIKE "ha_spider%";

+--------------------------+---------------+
| PLUGIN_NAME              | PLUGIN_STATUS |
+--------------------------+---------------+
| SPIDER                   | ACTIVE        |
| SPIDER_ALLOC_MEM         | ACTIVE        |
| SPIDER_WRAPPER_PROTOCOLS | ACTIVE        |
+--------------------------+---------------+
```

When the Enterprise Spider storage engine is loaded, the `PLUGIN_NAME` column contains the value `SPIDER` and the `PLUGIN_STATUS` column contains the value `ACTIVE`.

Next Step

Navigation in the procedure “Deploy Spider Sharded Topology”:

This page was step 1 of 3.

Next: Step 2: Configure Spider Node and Data Nodes.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 2: Configure Spider Node and Data Nodes

Overview

This page details step 2 of the 3-step procedure “Deploy Spider Sharded Topology”.

This step configures the Spider Node and Data Nodes and creates the Spider Table and Data Tables.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Create Spider User

The data node requires a user account that the Spider Node uses to connect.

On each Data Node, create the Spider user account for the Spider Node using the CREATE USER statement:

```
CREATE USER spider_user@192.0.2.1 IDENTIFIED BY "password";
```

Privileges will be granted to the user account in Grant Privileges on the Data Table.

Test Spider User

On the Spider Node, confirm that the Spider user account can connect to the Data Node using MariaDB Client:

```
$ mariadb --user spider_user --host 192.0.2.2 --password
```

Configure Connection Details

The Spider Node requires connection details for each Data Node.

On the Spider Node, create a server object to configure the connection details for each Data Node using the CREATE SERVER statement:

1. Create a Server object to configure the connection details for the Data Node at the headquarters branch:\

```
CREATE SERVER hq_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.2",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "hq_sales"
);
```

2. Create a server object to configure the connection details for the Data Node at the eastern branch:\

```
CREATE SERVER eastern_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.3",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "eastern_sales"
);
```

3. Create a server object to configure the connection details for the Data Node at the western branch:

```
CREATE SERVER western_server
FOREIGN DATA WRAPPER mariadb
OPTIONS (
   HOST "192.0.2.4",
   PORT 5801,
   USER "spider_user",
   PASSWORD "password",
   DATABASE "western_sales"
);
```

The Data Node runs MariaDB Enterprise Server, so the `FOREIGN DATA WRAPPER` is set to `mariadb`.

Using a server object for connection details is optional. Alternatively, the connection details for the Data Node can be specified in the `COMMENT` table option of the CREATE TABLE statement when creating the Spider Table.

Create the Data Tables

When queries read and write to a Spider Table, Spider reads and writes to the Data Tables for each partition on the on the Data Nodes. The Data Tables must be created on the Data Nodes with the same structure as the Spider Table.

If your Data Tables already exist, grant privileges on the tables to the Spider user.

On each Data Node, create the Data Tables:

1. On the Data Node for the headquarters server, create a database and table and add sample data:

```
CREATE DATABASE hq_sales;

CREATE SEQUENCE hq_sales.invoice_seq;

CREATE TABLE hq_sales.invoices (
   branch_id INT NOT NULL DEFAULT (1) CHECK (branch_id=1),
   invoice_id INT NOT NULL DEFAULT (NEXT VALUE FOR hq_sales.invoice_seq),
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=InnoDB;

INSERT INTO hq_sales.invoices
   (customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, '2020-05-10 12:35:10', 1087.23, 'CREDIT_CARD'),
   (2, '2020-05-10 14:17:32', 1508.57, 'WIRE_TRANSFER'),
   (3, '2020-05-10 14:25:16', 227.15, 'CASH');
```

The Spider Node reads and writes to the Data Table using the server and user account configured in “Configure Connection Details”. The user account must have privileges on the tables.

2. On the Data Node for the eastern branch of the business, create a database and table and add sample data:\\

```
CREATE DATABASE eastern_sales;

CREATE SEQUENCE eastern_sales.invoice_seq;

CREATE TABLE eastern_sales.invoices (
   branch_id INT NOT NULL DEFAULT (2) CHECK (branch_id=2),
   invoice_id INT NOT NULL DEFAULT (NEXT VALUE FOR eastern_sales.invoice_seq),
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=InnoDB;

INSERT INTO eastern_sales.invoices
   (customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (2, '2020-05-10 12:31:00', 1351.04, 'CREDIT_CARD'),
   (2, '2020-05-10 12:45:27', 162.11, 'WIRE_TRANSFER'),
   (4, '2020-05-10 13:11:23', 350.00, 'CASH');
```

   The Spider Node reads and writes to the Data Table using the server and user account configured in "Configure Connection Details". The user account must have privileges on the tables.\

3. On the Data Node for the western branch of the business, create a database and table and add sample data:\\

```
CREATE DATABASE western_sales;

CREATE SEQUENCE western_sales.invoice_seq;

CREATE TABLE western_sales.invoices (
   branch_id INT NOT NULL DEFAULT (3) CHECK (branch_id=3),
   invoice_id INT NOT NULL DEFAULT (NEXT VALUE FOR western_sales.invoice_seq),
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=InnoDB;

INSERT INTO western_sales.invoices
   (customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (5, '2020-05-10 12:31:00', 111.50, 'CREDIT_CARD'),
   (8, '2020-05-10 12:45:27', 1509.23, 'WIRE_TRANSFER'),
   (3, '2020-05-10 13:11:23', 3301.66, 'CASH');
```

   The Spider Node reads and writes to the Data Table using the server and user account configured in "Configure Connection Details". The user account must have privileges on the tables.\

Grant Privileges

The Spider Node connects to the Data Nodes with the user account configured in "Create Spider User".

On each Data Node, grant the Spider user sufficient privileges to operate on the Data Table:

```
GRANT ALL PRIVILEGES ON hq_sales.invoices TO 'spider_user'@'192.0.2.1';
```

Privileges for Spider BKA Mode

By default, the Spider user also requires the `CREATE TEMPORARY TABLES` privilege on the database containing the Data Table. The `CREATE TEMPORARY TABLES` privilege is required, because Spider uses temporary tables to optimize read queries when Spider BKA Mode is `1`.

Spider BKA Mode is configured using the following methods:

- The session value is configured by setting the spider_bka_mode system variable on the Spider Node. The default value is `-1`. When the session value is `-1`, the value for each Spider Table is used.
- The value for each Spider Table is configured by setting the `bka_mode` option in the `COMMENT` table option. When the `bka_mode` option is not set, the implicit value is 1.

The default spider_bka_mode value is `-1`, and the implicit Spider Table value is `1`, so the default Spider BKA Mode is `1`.

On the Data Node, grant the Spider user the `CREATE TEMPORARY TABLES` privilege on the database:

```
GRANT CREATE TEMPORARY TABLES ON hq_sales.* TO 'spider_user'@'192.0.2.1';
```

Create the Spider Table

The Spider Table must be created on the Spider Node with the same structure as the Data Tables. The Spider Table must have a partition for each Data Table.

On the Spider Node, create the Spider Table and reference the Data Node in the `COMMENT` table option:

```
CREATE DATABASE spider_sharded_sales;

CREATE TABLE spider_sharded_sales.invoices (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
PARTITION BY LIST(branch_id) (
   PARTITION hq_partition VALUES IN (1) COMMENT = 'server "hq_server", table "invoices"',
   PARTITION eastern_partition VALUES IN (2) COMMENT = 'server "eastern_server", table "invoices"',
   PARTITION western_partition VALUES IN (3) COMMENT = 'server "western_server", table "invoices"'
);
```

The `COMMENT` partition option is used to configure the Data Node and the Data Table for each partition. Set the server option to the server object for the partition configured in "Configure Connection Details". Set the `table` option to the Data Table for the partition.

An alternative syntax is available. When you don't want to create a server object, the connection details for the Data Nodes can be specified in the `COMMENT` partition option:

```
CREATE TABLE spider_hq_sales.invoices_alternate (
   branch_id INT NOT NULL,
   invoice_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(branch_id, invoice_id)
) ENGINE=Spider
COMMENT='table "invoices", host "192.0.2.2", port "5801", user "spider_user", password "user_password",
database "hq_sales"';
```

Next Step

Navigation in the procedure “Deploy Spider Sharded Topology”:

This page was **step 2 of 3**.

Next: Step 3: Test Spider Sharded Topology.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Step 3: Test Spider Sharded Topology

Overview

This page details step 3 of the 3-step procedure “Deploy Spider Sharded Topology”.

This step tests MariaDB Enterprise Spider.

Interactive commands are detailed. Alternatively, the described operations can be performed using automation.

Test Enterprise Server Service

Use Systemd to test whether the MariaDB Enterprise Server service is running.

This action is performed **on the Spider Node and each Data Node**.

Check if the MariaDB Enterprise Server service is running by executing the following:

```
$ systemctl status mariadb
```

If the service is not running on any node, start the service by executing the following on that node:

```
$ sudo systemctl start mariadb
```

Test Local Client Connections

Use mariadb Client to test the local connection to the Enterprise Server node.

This action is performed **on the Spider Node and each Data Node**:

```
$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The `sudo` command is used here to connect to the Enterprise Server node using the `root@localhost` user account, which authenticates using the `unix_socket` authentication plugin. Other user accounts can be used by specifying the `--user` and `--password` command-line options.

Test Data Node Client Connection

Use mariadb Client to test a client connection to the Data Node from the Spider Node using the Spider user.

This action is performed **on the Spider Node**:

```
$ mariadb \
   --host 192.0.2.2 \
   --user spider_user \
   --password
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 11.4.5-3-MariaDB-Enterprise MariaDB Enterprise Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The host and port of the Data Node can be provided using the `--host` and `--port` command-line options. The credentials for the Spider user can be provided using the `--user` and `--password` command-line options.

If the Spider user is unable to connect to the Data Node from the Spider Node, check the password for the Spider user account on the Data Node.

For additional information, see "Create Spider User".

Test Spider Storage Engine Plugin

Query the `information_schema.PLUGINS` table to confirm that the Enterprise Spider storage engine is loaded.

This action is performed **on the Spider Node**.

Execute the following query:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS
FROM information_schema.PLUGINS
WHERE PLUGIN_LIBRARY LIKE 'ha_spider%';

+--------------------------+---------------+
| PLUGIN_NAME              | PLUGIN_STATUS |
+--------------------------+---------------+
| SPIDER                   | ACTIVE        |
| SPIDER_ALLOC_MEM         | ACTIVE        |
| SPIDER_WRAPPER_PROTOCOLS | ACTIVE        |
+--------------------------+---------------+
```

The `PLUGIN_STATUS` column for each Spider-related plugin should contain `ACTIVE`.

For additional information, see "Load the Spider Plugin".

Test Write Operations

Write to the Spider Table using an INSERT statement to test write operations.

This action is performed **on the Spider Node**.

Execute the following query:

```
INSERT INTO spider_hq_sales.invoices
   (branch_id, invoice_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES (1, 4, 1, '2021-03-10 12:45:10', 3045.73, 'CREDIT_CARD');
```

Test Read Operations

Read from the Spider Table using a SELECT statement to test read operations.

This action is performed **on the Spider Node**.

Execute the following query:

```
SELECT * FROM spider_sharded_sales.invoices;
```

```
+-----------+------------+-------------+----------------------------+---------------+----------------+
| branch_id | invoice_id | customer_id | invoice_date               | invoice_total | payment_method |
+-----------+------------+-------------+----------------------------+---------------+----------------+
|         1 |          1 |           1 | 2020-05-10 12:35:10.000000 |       1087.23 | CREDIT_CARD    |
|         1 |          2 |           2 | 2020-05-10 14:17:32.000000 |       1508.57 | WIRE_TRANSFER  |
|         1 |          3 |           3 | 2020-05-10 14:25:16.000000 |        227.15 | CASH           |
|         1 |          4 |           1 | 2021-03-10 12:45:10.000000 |       3045.73 | CREDIT_CARD    |
|         2 |          1 |           2 | 2020-05-10 12:31:00.000000 |       1351.04 | CREDIT_CARD    |
|         2 |          2 |           2 | 2020-05-10 12:45:27.000000 |        162.11 | WIRE_TRANSFER  |
|         2 |          3 |           4 | 2020-05-10 13:11:23.000000 |        350.00 | CASH           |
|         3 |          1 |           5 | 2020-05-10 12:31:00.000000 |        111.50 | CREDIT_CARD    |
|         3 |          2 |           8 | 2020-05-10 12:45:27.000000 |       1509.23 | WIRE_TRANSFER  |
|         3 |          3 |           3 | 2020-05-10 13:11:23.000000 |       3301.66 | CASH           |
+-----------+------------+-------------+----------------------------+---------------+----------------+
```

Test Sharding

Use the EXPLAIN PARTITIONS statement with a SELECT statement to determine which shards Spider will read for the query.

This action is performed **on the Spider Node**.

Execute the following query:

```
EXPLAIN PARTITIONS
SELECT * FROM spider_sharded_sales.invoices
WHERE customer_id = 4;
```

```
+------+-------------+----------+-------------------------------------------------+------+-------------
-+------+---------+------+------+---------------------------------+
| id   | select_type | table    | partitions                                      | type | possible_keys
| key  | key_len | ref  | rows | Extra                           |
+------+-------------+----------+-------------------------------------------------+------+-------------
-+------+---------+------+------+---------------------------------+
|    1 | SIMPLE      | invoices | hq_partition,eastern_partition,western_partition | ALL  | NULL
| NULL | NULL    | NULL | 9    | Using where with pushed condition |
+------+-------------+----------+-------------------------------------------------+------+-------------
-+------+---------+------+------+---------------------------------+
```

The specific shards read by the query are listed in the `partitions` column. If partition pruning does not eliminate unnecessary shards for a query with a restrictive filter, then check the partition definitions.

Next Step

Navigation in the procedure “Deploy Spider Sharded Topology”:

This page was step 3 of 3.

This procedure is complete.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Server Constraints

See Server Constraints Overview.

Provides a high-level overview of how MariaDB Server enforces data integrity through various architectural constraint types.

See AUTO_INCREMENT Constraints.

Explains the limitations and behavior of `AUTO_INCREMENT` columns, including how they handle maximum values and gaps in sequences.

See FOREIGN KEY Constraints.

Details the requirements and limitations for defining foreign keys to ensure referential integrity between tables.

See NOT NULL Constraints.

Describes how `NOT NULL` constraints prevent `NULL` values from being stored in specific columns.

See PRIMARY KEY Constraints.

Outlines the rules for primary keys, which uniquely identify each record in a table and cannot contain `NULL` values.

See UNIQUE Constraints.

Explains the specific implementation and considerations for unique constraints when using MariaDB Enterprise Server.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Server Constraints Overview

MariaDB Enterprise Server provides DDL (Data Definition Language) syntax to `create, alter, or drop` constraints.

In the context of `ISO 9075-2:2016`, this page uses “DDL (Data Definition Language)” to refer to SQL statements in the standard’s “SQL-schema statements” group.

This page is: Copyright © 2026 MariaDB. All rights reserved.

AUTO_INCREMENT Constraints

To define an `AUTO_INCREMENT` column, select an integer data type to accommodate the range of anticipated values. Common choices include `INT`, `BIGINT`, `TINYINT`, `SMALLINT`, and `MEDIUMINT`, depending on size requirements. Ensure the chosen type provides sufficient capacity to avoid overflow while optimizing storage efficiency.

Overview

MariaDB Enterprise Server supports `AUTO_INCREMENT` constraints:

- A column with an `AUTO_INCREMENT` constraint can act as a table’s primary key when a natural primary key is not available
- Generated values for auto-increment columns are guaranteed to be unique and monotonically increasing
- Auto-increment columns provide compatibility with schemas designed for MariaDB Enterprise Server and MySQL

Alternatively, MariaDB Enterprise Server can use sequences as the primary key instead of columns with `AUTO_INCREMENT` constraints. Sequences are compliant with the SQL standard, while `AUTO_INCREMENT` constraints are not, so sequences are the better option for applications that require standard-compliant features.

Choosing a Data Type for an `AUTO_INCREMENT` Column

When designing a schema, `AUTO_INCREMENT` columns should use integer data types. The following types can be used:

Data Type	Signed Range	Unsigned Range
TINYINT	-128 - 127	0 - 255
SMALLINT	-32768 - 32767	0 - 65535
MEDIUMINT	-8388608 - 8388607	0 - 16777215
INT	-2147483648 - 2147483647	0 - 4294967295
BIGINT	-9223372036854775808 - 9223372036854775807	0 - 18446744073709551615

To determine which type to use, consider the following points:

1. Do you want to be able to manually insert negative values? If not, then specify the UNSIGNED attribute for the column.

InnoDB can't generate negative `AUTO_INCREMENT` values, so it is only beneficial to use a signed integer column if you want the option to manually insert negative values, which would bypass the `AUTO_INCREMENT` handling.

2. How large will your table grow?

If your `AUTO_INCREMENT` column is being used as the table's primary key, then the maximum value for the chosen data type should be considered the maximum number of rows that can fit in the table:

Data Type	Signed Range	Unsigned Range
TINYINT	127	255
SMALLINT	32767	65535
MEDIUMINT	8388607	16777215
INT	2147483647	4294967295
BIGINT	9223372036854775807	18446744073709551615

If you want to give your table the most room to grow, then it would be best to choose BIGINT UNSIGNED.

Creating an InnoDB Table with an AUTO_INCREMENT Column

Let's create an InnoDB table with an AUTO_INCREMENT column after confirming that the default storage engine is InnoDB:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the table using the CREATE TABLE statement:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
);
```

Inserting a Row with an AUTO_INCREMENT Column

If a column is specified as `AUTO_INCREMENT`, then its value will be automatically generated. There are multiple ways to insert rows with these automatically generated values.

Omitting the Column

If the column is not specified, then InnoDB will automatically generate the value.

Let's insert a row into the table created in the Creating an InnoDB Table with an AUTO_INCREMENT Column section:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Insert a row with the INSERT statement, but do not specify the `AUTO_INCREMENT` column:

```
INSERT INTO hq_sales.invoices
   (branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 1, '2020-05-10 12:35:10', 1087.23, 'CREDIT_CARD');
```

3. Select the same row with the SELECT statement to confirm that a value was automatically generated:

```
SELECT invoice_id
FROM hq_sales.invoices
WHERE branch_id = 1
AND customer_id = 1
AND invoice_date = '2020-05-10 12:35:10';
```

```
+------------+
| invoice_id |
+------------+
|          1 |
+------------+
```

Specifying the Column's Value as 0

If the column's value is specified as 0, then InnoDB will automatically generate the value if the sql_mode system variable does not contain `NO_AUTO_VALUE_ON_ZERO`.

Let's insert a row into the table created in the Creating an InnoDB Table with an AUTO_INCREMENT Column section:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the session's value of the sql_mode system variable does not contain NO_AUTO_VALUE_ON_ZERO with the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'sql_mode';
```

```
+---------------+-----------------------------------------------------------------------------------------------
---+
| Variable_name | Value
|
+---------------+-----------------------------------------------------------------------------------------------
---+
| sql_mode      |
STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+---------------+-----------------------------------------------------------------------------------------------
---+
```

3. Insert a row with the INSERT statement, and specify the `AUTO_INCREMENT` column's value as 0:

```
INSERT INTO hq_sales.invoices
   (invoice_id, branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (0, 1, 2, '2020-05-10 14:17:32', 1508.57, 'WIRE_TRANSFER');
```

4. Select the same row with the SELECT statement to confirm that a value was automatically generated:

```
SELECT invoice_id
FROM hq_sales.invoices
WHERE branch_id = 1
AND customer_id = 2
AND invoice_date = '2020-05-10 14:17:32';
```

```
+------------+
| invoice_id |
+------------+
|          2 |
+------------+
```

Specifying the Column's Value as NULL

If the column's value is specified as NULL, then InnoDB will automatically generate the value if the column is defined as NOT NULL.

Let's insert a row into the table created in the Creating an InnoDB Table with an AUTO_INCREMENT Column section:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Insert a row with the INSERT statement, and specify the `AUTO_INCREMENT` column's value as NULL:

```
INSERT INTO hq_sales.invoices
   (invoice_id, branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (NULL, 1, 3, '2020-05-10 14:25:16', 227.15, 'CASH');
```

3. Select the same row with the SELECT statement to confirm that a value was automatically generated:

```
SELECT invoice_id
FROM hq_sales.invoices
WHERE branch_id = 1
AND customer_id = 3
AND invoice_date = '2020-05-10 14:25:16';
```

```
+------------+
| invoice_id |
+------------+
|          3 |
+------------+
```

Getting the Last Inserted AUTO_INCREMENT Value

After InnoDB inserts an automatically generated value into an `AUTO_INCREMENT` column, the application sometimes needs to know what value it inserted. For example, the application may need to use the value to insert a foreign key column in a dependent table. The LAST_INSERT_ID() function can be used to get the lasted inserted value for an `AUTO_INCREMENT` column without re-reading the row from the table.

Let's insert a row into the table created in the Creating an InnoDB Table with an AUTO_INCREMENT Column section and then use the LAST_INSERT_ID() function:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Insert a row with the INSERT statement, but do not specify the `AUTO_INCREMENT` column:

```
INSERT INTO hq_sales.invoices
   (branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 4, '2020-05-10 12:37:22', 104.19, 'CREDIT_CARD');
```

3. Execute the LAST_INSERT_ID() function to get the `AUTO_INCREMENT` value for the new row:

```
SELECT LAST_INSERT_ID();

+------------------+
| LAST_INSERT_ID() |
+------------------+
|                4 |
+------------------+
```

4. Select the same row with the SELECT statement to confirm that the `AUTO_INCREMENT` column has the same value:

```
SELECT invoice_id
FROM hq_sales.invoices
WHERE branch_id = 1
AND customer_id = 4
AND invoice_date = '2020-05-10 12:37:22';
```

```
+------------+
| invoice_id |
+------------+
|          4 |
+------------+
```

Choosing an AUTO_INCREMENT Lock Mode for InnoDB

When multiple rows are inserted into a table concurrently, InnoDB needs to be able to generate multiple values concurrently in a safe manner. It has several different modes that can be used to do this, and each mode has its own advantages and disadvantages.

InnoDB's `AUTO_INCREMENT` lock mode is configured with the innodb_autoinc_lock_mode system variable. Users can choose between 3 different values:

Value	Description
0	• This value configures Traditional Lock Mode. • Don't use traditional lock mode. • Traditional lock mode performs very poorly. • In traditional lock mode, InnoDB holds a table-level lock while generating AUTO_INCREMENT values.
1	• This value configures Consecutive Lock Mode. • Consecutive lock mode is the default lock mode. • In consecutive lock mode, InnoDB holds a table-level lock while generating AUTO_INCREMENT values for statements that insert multiple new rows. However, InnoDB uses a lightweight internal lock to improve performance when generating an AUTO_INCREMENT value for statements that insert a single new row.
2	• This value configures Interleaved Lock Mode. • Interleaved lock mode is the recommended lock mode for best performance. • If Galera Cluster is being used, then interleaved lock mode must be configured.

• In interleaved lock mode, InnoDB never holds a table-level lock while generating `AUTO_INCREMENT` values. • Interleaved lock mode is not safe to use if binlog_format is set to `STATEMENT`.|

Configuring the AUTO_INCREMENT Lock Mode for InnoDB

The innodb_autoinc_lock_mode system variable configures the `AUTO_INCREMENT` Lock Mode for InnoDB.

1. Choose a configuration file for custom changes to system variables and options.

It is not recommended to make custom changes to Enterprise Server's default configuration files, because your custom changes can be overwritten by other default configuration files that are loaded after.

Ensure that your custom changes are read last by creating a custom configuration file in one of the included directories. Configuration files in included directories are read in alphabetical order. Use the z- prefix in the file name to ensure that your custom configuration file is read last.

Some example configuration file paths for different distributions are shown in the following table:

Distributions	Example configuration file path
CentOS RHEL Rocky Linux SLES	/etc/my.cnf.d/z-custom-mariadb.cnf
Debian Ubuntu	/etc/mysql/mariadb.conf.d/z-custom-mariadb.cnf

2. Set the innodb_autoinc_lock_mode system variable in the configuration file.
   It needs to be set in a group that will be read by MariaDB Server, such as [mariadb] or [server].

For example:

```
[mariadb]
...
innodb_autoinc_lock_mode = 2
```

3. Restart the server:

```
$ sudo systemctl restart mariadb
```

Setting a Table's Next AUTO_INCREMENT Value

A table's next `AUTO_INCREMENT` value can be set with the ALTER TABLE statement. The value is set using the `AUTO_INCREMENT` table option.

Let's alter the `AUTO_INCREMENT` value for the table created in the Creating an InnoDB Table with an AUTO_INCREMENT Column section and then insert a row into the table, so we can confirm that it uses the new value:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Alter the table's next `AUTO_INCREMENT` value with the ALTER TABLE statement:

```
ALTER TABLE hq_sales.invoices
   AUTO_INCREMENT = 100;
```

3. Insert a row with the INSERT statement, but do not specify the `AUTO_INCREMENT` column:

```
INSERT INTO hq_sales.invoices
   (branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 5, '2020-05-10 12:43:19', 1105.98, 'CREDIT_CARD');
```

4. Execute the LAST_INSERT_ID() function to get the `AUTO_INCREMENT` value for the new row:

```
SELECT LAST_INSERT_ID();

+------------------+
| LAST_INSERT_ID() |
+------------------+
|              100 |
+------------------+
```

5. Select the same row with the SELECT statement to confirm that the `AUTO_INCREMENT` column has the same value:

```
SELECT invoice_id
FROM hq_sales.invoices
WHERE branch_id = 1
AND customer_id = 5
AND invoice_date = '2020-05-10 12:43:19';

+------------+
| invoice_id |
+------------+
|        100 |
+------------+
```

Configure the Offset and Increment Values

The offset and increment values can be configured by setting the auto_increment_offset and auto_increment_increment system variables.

When Galera Cluster is used, the offset and increment values are managed automatically by default. They can be managed manually by disabling the wsrep_auto_increment_control system variable.

This page is: Copyright © 2026 MariaDB. All rights reserved.

FOREIGN KEY Constraints

In MariaDB Server, `FOREIGN KEY` constraints are key elements in defining relationships between InnoDB tables: they ensure referential integrity by maintaining valid associations between rows in parent and child tables. Changes in the parent table can either be restricted or propagated to maintain consistency with child tables.

Overview

MariaDB Server supports `FOREIGN KEY` constraints to define referential constraints between InnoDB tables:

- Referential constraints allow the database to automatically ensure that each row in the child table is associated with a valid row in the parent table
- If the row in the parent table changes, MariaDB Server can block the change to protect child rows, or propagate the change to the child rows

Term Definitions

Term	Definition
parent table	A table that has a foreign key.
child table	A table that is referenced by a foreign key.

Uses Cases for Foreign Keys

Foreign keys allow the database to handle certain types of referential constraint checks, so that the application does not have to handle them. As a consequence, application logic can be simplified.

Some example use cases are described below.

Managing Accounts Use Case

Many different applications have account management features. Foreign keys can be used to simplify certain aspects of managing accounts, such as:

- If an account is deleted, foreign keys can be used to automatically delete data associated with the account.
- If an account is deleted, foreign keys can be used to automatically disassociate its data, so that it is no longer associated with any account.
- If an account's user name or user ID is updated, foreign keys can be used to automatically associate its data with the new user name or user ID.

Managing Store Inventory Use Case

Foreign keys can also be used to simplify certain aspects of managing a store inventory, such as:

- If a product is deleted, foreign keys can be used to automatically delete data associated with the product, such as reviews and images.
- If a product's name is changed, foreign keys can be used to automatically associate the new name with the product's other data.
- If a customer purchases a product, foreign keys can be used to associate the order with the product.
- If a customer reviews a product, foreign keys can be used to associate the review with the product.

Operating on InnoDB Tables with Foreign Keys

When an InnoDB table has a foreign key or when it is referenced by a foreign key, InnoDB performs a referential constraint check when certain types of operations try to write to the table. The specific details depend on whether the table is a parent table or a child table.

Operating on a Parent Table

When an InnoDB table has a foreign key, it is known as a parent table. InnoDB performs referential constraint checks for the following operations on parent tables:

- UPDATE
- DELETE

When InnoDB performs a referential constraint check, the outcome depends on several factors. The following table describes the details:

Operation	Result of Constraint Check	Action	Consequence
UPDATE	InnoDB finds corresponding rows in the child table	ON UPDATE RESTRICT	• Fails with ER_ROW_IS_REFERENCED_2 error code
UPDATE	InnoDB finds corresponding rows in the child table	ON UPDATE NO ACTION	•. Fails with ER_ROW_IS_REFERENCED_2 error code
UPDATE	InnoDB finds corresponding rows in the child table	ON UPDATE CASCADE	• Success. • Row in the parent table is updated. • Corresponding rows in the child table are also updated with the new foreign key value. If the child table has an update trigger, the trigger will not be executed for the update.
UPDATE	InnoDB finds corresponding rows in the child table	ON UPDATE SET NULL	• Success. • Row in the parent table is updated. • Corresponding rows in the child table are also updated with NULL. If the child table has an update trigger, the trigger will not be executed for the update.
UPDATE	InnoDB does not find any corresponding rows in the child table	NA	• Success. • Row in the parent table is updated
DELETE	InnoDB finds corresponding rows in the child table	ON DELETE RESTRICT	• Fails with ER_ROW_IS_REFERENCED_2 error code
DELETE	InnoDB finds corresponding rows in the child table	ON DELETE NO ACTION	• Fails with EER_ROW_IS_REFERENCED_2 error code
DELETE	InnoDB finds corresponding rows in the child table	ON DELETE CASCADE	• Success. • Row in the parent table is deleted. • Corresponding rows in the child table are also deleted. If the child table has a delete trigger, the trigger will not be executed for the delete.
DELETE	InnoDB finds rows in the child table for the row	ON DELETE SET NULL	• Success. • Row in the parent table is deleted. • Corresponding rows in the child table are updated with NULL. If the child table has an update trigger, the trigger will not be executed for the update.
DELETE	InnoDB does not find any rows in the child table for the row	NA	• Success. • Row in the parent table is deleted

Operating on a Child Table

When an InnoDB table is referenced by a foreign key, it is known as a child table. InnoDB performs referential constraint checks for the following operations on child tables:

- INSERT
- UPDATE
- DROP TABLE

When InnoDB performs a referential constraint check, the outcome depends on several factors. The following table describes the details:

Operation	Result of Constraint Check	Consequence
INSERT	New foreign key value is present in parent table	Success
INSERT	New foreign key value is not present in parent table	Fails with ER_NO_REFERENCED_ROW_2 error code
INSERT	New foreign key value is NULL	Success
UPDATE	New foreign key value is present in parent table	Success
UPDATE	New foreign key value is not present in parent table	Fails with ER_NO_REFERENCED_ROW_2 error code
UPDATE	New foreign key value is NULL	Success
DROP TABLE	Table is referenced by a foreign key	Fails with ER_NO_REFERENCED_ROW_2 error code

Creating InnoDB Tables with a Foreign Key Constraint

Let's create InnoDB tables with a foreign key constraint after confirming that the default storage engine is InnoDB:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the sequence and table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the parent table using the CREATE TABLE statement:

```
CREATE TABLE hq_sales.customers (
   customer_id BIGINT AUTO_INCREMENT NOT NULL,
   customer_name VARCHAR(500) NOT NULL,
   customer_email VARCHAR(200),
   PRIMARY KEY(customer_id)
);
```

5. Create the child table using the CREATE TABLE statement:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id BIGINT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id),
   CONSTRAINT fk_invoices_customers
      FOREIGN KEY (customer_id) REFERENCES hq_sales.customers (customer_id)
      ON DELETE RESTRICT
      ON UPDATE RESTRICT
);
```

6. Insert some rows into the parent table using the INSERT statement:

```
INSERT INTO hq_sales.customers (customer_id, name)
   VALUES
   (1, 'John Doe'),
   (2, 'Jane Doe');
```

7. Insert a row into the child table for each row in the parent table using the INSERT statement:

```
INSERT INTO hq_sales.invoices
   (branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 1, '2020-05-10 12:35:10', 1087.23, 'CREDIT_CARD'),
   (1, 2, '2020-05-10 14:17:32', 1508.57, 'WIRE_TRANSFER');
```

8. Attempt to delete a row from the parent table that has a corresponding row in the child table using the `DELETE` statement:

```
DELETE FROM hq_sales.customers
   WHERE customer_id = 1;
```

This will fail with the ER_ROW_IS_REFERENCED_2 error code as explained in the Operating on a Parent Table section:

```
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
   (`hq_sales`.`invoices`, CONSTRAINT `fk_invoices_customers`
   FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`))
```

Attempt to insert a row into the child table for a non-existent row in the parent table using the INSERT statement:

```
INSERT INTO hq_sales.invoices
   (branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 3, '2020-05-10 14:25:16', 227.15, 'CASH');
```

This will fail with the ER_NO_REFERENCED_ROW_2 error code as explained in the Operating on a Child Table section:

```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
   (`hq_sales`.`invoices`, CONSTRAINT `fk_invoices_customers`
   FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`))
```

Adding a Foreign Key Constraint to an InnoDB Table

Let's create InnoDB tables after confirming that the default storage engine is InnoDB, and then let's add a foreign key constraint between them:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the sequence and table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the parent table using the CREATE TABLE statement:

```
CREATE TABLE hq_sales.customers (
   customer_id BIGINT AUTO_INCREMENT NOT NULL,
   customer_name VARCHAR(500) NOT NULL,
   customer_email VARCHAR(200),
   PRIMARY KEY(customer_id)
);
```

5. Create the child table using the CREATE TABLE statement:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT AUTO_INCREMENT NOT NULL,
   branch_id INT NOT NULL,
   customer_id BIGINT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
);
```

6. Alter the child table to add the foreign key constraint using the ALTER TABLE statement:

```
ALTER TABLE hq_sales.invoices ADD CONSTRAINT fk_invoices_customers
      FOREIGN KEY (customer_id) REFERENCES hq_sales.customers (customer_id)
      ON DELETE RESTRICT
      ON UPDATE RESTRICT;
```

7. Insert some rows into the parent table using the INSERT statement:

```
INSERT INTO hq_sales.customers (customer_id, name)
   VALUES
   (1, 'John Doe'),
   (2, 'Jane Doe');
```

8. Insert a row into the child table for each row in the parent table using the INSERT statement:

```
INSERT INTO hq_sales.invoices
   (branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 1, '2020-05-10 12:35:10', 1087.23, 'CREDIT_CARD'),
   (1, 2, '2020-05-10 14:17:32', 1508.57, 'WIRE_TRANSFER');
```

9. Attempt to delete a row from the parent table that has a corresponding row in the child table using the DELETE statement:

```
DELETE FROM hq_sales.customers
   WHERE customer_id = 1;
```

This will fail with the ER_ROW_IS_REFERENCED_2 error code as explained in the Operating on a Parent Table section:

```
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
   (`hq_sales`.`invoices`, CONSTRAINT `fk_invoices_customers`
   FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`))
```

10. Attempt to insert a row into the child table for a non-existent row in the parent table using the INSERT statement:

```
INSERT INTO hq_sales.invoices
   (branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 3, '2020-05-10 14:25:16', 227.15, 'CASH');
```

This will fail with the ER_NO_REFERENCED_ROW_2 error code as explained in the Operating on a Child Table section:

```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
   (`hq_sales`.`invoices`, CONSTRAINT `fk_invoices_customers`
   FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`))
```

Dropping a Foreign Key Constraint from an InnoDB Table

Let's drop the foreign key constraint from the child table created in the Creating InnoDB Tables with a Foreign Key Constraint section:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Obtain the name of the foreign key constraint by querying the information_schema.TABLE_CONSTRAINTS table:

```
SELECT CONSTRAINT_NAME
FROM information_schema.TABLE_CONSTRAINTS
WHERE TABLE_SCHEMA = 'hq_sales'
AND TABLE_NAME = 'invoices'
AND CONSTRAINT_TYPE = 'FOREIGN KEY';
```

```
+-----------------------+
| CONSTRAINT_NAME       |
+-----------------------+
| fk_invoices_customers |
+-----------------------+
```

3. Drop the foreign key constraint from the child table using the ALTER TABLE statement:

```
ALTER TABLE hq_sales.invoices DROP FOREIGN KEY fk_invoices_customers;
```

Temporarily Disabling Foreign Key Constraint Checks

When performing bulk data loads, dropping a table, or rebuilding a table, it can improve performance to temporarily disable foreign key constraint checks. Foreign key constraint checks can be temporarily disabled by setting the session value for the `foreign_key_checks` system variable. If foreign key constraint checks are temporarily disabled, then rows can be inserted that violate the constraint, so users must proceed with caution. This feature is most useful when you are loading a data set that is known to be valid.

Let's temporarily disable foreign key constraint checks, and then perform some tasks with the tables created in the Creating InnoDB Tables with a Foreign Key Constraint section:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Temporarily disable foreign key constraint checks by setting the foreign_key_checks system variable with the SET SESSION statement:

```
SET SESSION foreign_key_checks=OFF;
```

3. If you want to test how to introduce inconsistencies, then attempt to delete a row from the parent table that has a corresponding row in the child table using the DELETE statement:

```
DELETE FROM hq_sales.customers
   WHERE customer_id = 1;
```

This operation would usually fail with the ER_NO_REFERENCED_ROW error code as explained in the Operating on a Parent Table section, but if foreign key constraint checks are disable, then it will succeed.

4. If you want to test how to introduce inconsistencies, then also attempt to insert a row into the child table for a non-existent row in the parent table using the INSERT statement:

```
INSERT INTO hq_sales.invoices
   (branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 3, '2020-05-10 14:25:16', 227.15, 'CASH');
```

This operation would usually fail with the ER_NO_REFERENCED_ROW_2 error code as explained in the Operating on a Child Table section, but if foreign key constraint checks are disable, then it will succeed.

5. Re-enable foreign key constraint checks by setting the foreign_key_checks system variable with the SET SESSION statement:

```
SET SESSION foreign_key_checks=ON;
```

Special Cases

Foreign Key Constraint Namespace

When a foreign key constraint is created without a name, InnoDB implicitly gives it a name in the format:

```
<table_name>_ibfk_<constraint_count>
```

If foreign key constraints are explicitly created with names in the same format, it is possible for collisions to occur during table renames. In that case, the MariaDB Error Log would contain messages like the following:

```
2021-06-16 11:50:39 139702710404864 [ERROR] InnoDB: Possible reasons:
2021-06-16 11:50:39 139702710404864 [ERROR] InnoDB: (1) Table rename would cause two FOREIGN KEY
constraints to have the same internal name in case-insensitive comparison.
2021-06-16 11:50:39 139702710404864 [ERROR] InnoDB: (2) Table `test`.`t3` exists in the InnoDB internal
data dictionary though MySQL is trying to rename table `test`.`t2b` to it. Have you deleted the .frm file
and not used DROP TABLE?
2021-06-16 11:50:39 139702710404864 [ERROR] InnoDB: If table `test`.`t3` is a temporary table #sql...,
then it can be that there are still queries running on the table, and it will be dropped automatically
when the queries end. You can drop the orphaned table inside InnoDB by creating an InnoDB table with the
same name in another database and copying the .frm file to the current database. Then MySQL thinks the
table exists, and DROP TABLE will succeed.
```

Dependence on an Index

A foreign key constraint requires an index on the column. If a foreign key constraint is added to a column without an index, InnoDB will automatically create an index to enforce the foreign key constraint.

When the foreign_key_checks system variable is disabled, it is possible to drop the index used by a foreign key constraint. When the foreign_key_checks system variable is re-enabled, InnoDB will have no way to enforce the foreign key constraint, so all operations that could potentially violate the foreign key constraint will fail.

This page is: Copyright © 2026 MariaDB. All rights reserved.

NOT NULL Constraints

When creating a table, you can define a column with a `NOT NULL` constraint to prevent null values. This constraint ensures data integrity by guaranteeing that a column must always contain a value. If an attempt is made to insert or update a row with a null value in a `NOT NULL` column, the operation will be rejected, thus maintaining the integrity of the database.

Overview

MariaDB Server supports `NOT NULL` constraints to ensure that a column's value is not set to `NULL`:

When a column is declared with a `NOT NULL` constraint, Enterprise Server rejects operations that would write a NULL value to the column

CREATE TABLE

CREATE TABLE and NOT NULL Constraints

With MariaDB Server, the CREATE TABLE statement can be used to create a new table with a `NOT NULL` constraint on one or more columns:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY (invoice_id)
);
```

ALTER TABLE

ALTER TABLE .. MODIFY COLUMN .. NOT NULL

With MariaDB Server, the ALTER TABLE statement can be used to add the `NOT NULL` constraint to a column using the `MODIFY COLUMN` clause:

```
ALTER TABLE hq_sales.invoices
   MODIFY COLUMN customer_id INT NOT NULL;
ALTER TABLE .. MODIFY COLUMN .. NULL
```

With MariaDB Server, the ALTER TABLE statement can be used to remove the `NOT NULL` constraint from a column using the `MODIFY COLUMN` clause:

```
ALTER TABLE hq_sales.invoices
   MODIFY COLUMN customer_id INT NULL;
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

PRIMARY KEY Constraints

Using an `AUTO_INCREMENT` column as the primary key in MariaDB is beneficial for ensuring unique row identification and efficient data retrieval. This approach automatically generates a unique integer for each new row, simplifying primary key management. It is especially useful in tables where data insertion occurs frequently, as it facilitates quick indexing and minimizes storage overhead due to its numeric nature. This practice helps maintain optimal performance in database operations, particularly when combined with clustered indexing in InnoDB.

Overview

MariaDB Server supports `PRIMARY KEY` constraints to uniquely identify rows:

There can only be a single primary key for a given table.

- InnoDB uses the primary key as a clustered index, which means that InnoDB stores table data in the order determined by the primary key.
- Primary key indexes are B+ trees, which are very efficient for searching for exact values, performing range scans, and checking uniqueness.
- If no primary key is defined for a table, then InnoDB will use the table's first `NOT NULL` unique index as the table's primary key.
- If no primary key or `NOT NULL` unique index is defined for a table, then InnoDB will automatically create a primary key called `GEN_CLUST_INDEX`, using an internal `48-bit DB_ROW_ID` column as the key. Replication with such tables can be very slow, especially when binlog_format is `MIXED or ROW`!
- For best performance, users should always create primary keys for their tables, and primary keys should be short, because the primary key columns are duplicated in every secondary index record.

Using an AUTO_INCREMENT Column as the Primary Key

If your table does not have a column or a set of columns that could act as a natural primary key, then you can define a single `AUTO_INCREMENT` column, which can serve as the table's primary key. See InnoDB AUTO_INCREMENT Columns for more details.

Using a Sequence-backed Column as the Primary Key

If your table does not have a column or a set of columns that could act as a natural primary key, then you can use a sequence to generate an integer value to use as the table's primary key. Sequences were first added in MariaDB Server 10.3 and MariaDB Community Server 10.3. See InnoDB Sequences for more details.

Creating an InnoDB Table with a Single Column Primary Key

Let's create an InnoDB table with a single column primary key after confirming that the default storage engine is InnoDB:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the table using the CREATE TABLE statement and specify the primary key with the PRIMARY KEY() clause:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id)
);
```

For a single column primary key, the primary key can also be specified with the `PRIMARY KEY` column option:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED NOT NULL PRIMARY KEY,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD')
);
```

Creating an InnoDB Table with a Composite Primary Key

Let's create an InnoDB table with a composite (multi-column) primary key after confirming that the default storage engine is InnoDB:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the table using the CREATE TABLE statement and specify the primary key with the PRIMARY KEY() clause:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id, branch_id)
);
```

Adding a Primary Key to an InnoDB Table

Let's create an InnoDB table without a primary key, and then add a primary key to it:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the table without a primary key using the CREATE TABLE statement:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD')
);
```

5. Alter the table using the ALTER TABLE statement and specify the new primary key with the ADD PRIMARY KEY() clause:

```
ALTER TABLE hq_sales.invoices ADD PRIMARY KEY (invoice_id);
```

Dropping a Primary Key from an InnoDB Table

Let's drop the primary key from the table created in the create an InnoDB table without a primary key section:

1. Alter the table using the ALTER TABLE statement and specify the `DROP PRIMARY KEY` clause:

```
ALTER TABLE hq_sales.invoices DROP PRIMARY KEY;
```

Changing a Primary Key for an InnoDB Table

Let's change the primary key from the table created in the create an InnoDB table without a primary key section:

1. Alter the table using the ALTER TABLE statement and specify the `DROP PRIMARY KEY` clause to drop the old primary key, and specify the new primary key with the `ADD PRIMARY KEY()` clause:

```
ALTER TABLE hq_sales.invoices
   DROP PRIMARY KEY,
   ADD PRIMARY KEY (invoice_id, branch_id);
```

Finding Tables without Primary Keys

For best performance, every InnoDB table should have a primary key. It is possible to find tables without a primary key using a basic SELECT statement.

Let's create an InnoDB table without a primary key, and then use a SELECT statement to confirm that it does not have one:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the table without a primary key using the CREATE TABLE statement:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD')
);
```

5. Query the information_schema.TABLES and information_schema.KEY_COLUMN_USAGE tables to find InnoDB tables that do not have a primary key:

```
SELECT t.TABLE_SCHEMA, t.TABLE_NAME
FROM information_schema.TABLES AS t
LEFT JOIN information_schema.KEY_COLUMN_USAGE AS c
ON t.TABLE_SCHEMA = c.CONSTRAINT_SCHEMA
   AND t.TABLE_NAME = c.TABLE_NAME
   AND c.CONSTRAINT_NAME = 'PRIMARY'
WHERE t.TABLE_SCHEMA != 'mysql'
   AND t.ENGINE = 'InnoDB'
   AND c.CONSTRAINT_NAME IS NULL;
```

```
+--------------+------------+
| TABLE_SCHEMA | TABLE_NAME |
+--------------+------------+
| hq_sales     | invoices   |
+--------------+------------+
```

6. To add a primary key, alter the table using the ALTER TABLE statement, and specify the primary key with the `ADD PRIMARY KEY()` clause:

```
ALTER TABLE hq_sales.invoices ADD PRIMARY KEY (invoice_id);
```

Handling Duplicate Key Errors

A primary key uniquely identifies every row. Therefore, if a second row is inserted with an identical value, it will fail.

Let's try to insert two identical primary key values into the table created in the Creating an InnoDB Table with a Single Column Primary Key section:

1. Insert a row with the INSERT statement:

```
INSERT INTO hq_sales.invoices
   (invoice_id, branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 1, 1, '2020-05-10 12:35:10', 1087.23, 'CREDIT_CARD');
```

2. Insert a second row that has the same primary key value with the INSERT statement:

```
INSERT INTO hq_sales.invoices
   (invoice_id, branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (1, 1, 2, '2020-05-10 14:17:32', 1508.57, 'WIRE_TRANSFER');
```

This will fail with the ER_DUP_ENTRY error code:

```
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

3. Fix the problem by inserting the row with a unique primary key value:

```
INSERT INTO hq_sales.invoices
   (invoice_id, branch_id, customer_id, invoice_date, invoice_total, payment_method)
VALUES
   (2, 1, 2, '2020-05-10 14:17:32', 1508.57, 'WIRE_TRANSFER');
```

To easily generate unique values for a primary key, consider using one of the following options:

- InnoDB AUTO_INCREMENT Columns
- InnoDB Sequences

This page is: Copyright © 2026 MariaDB. All rights reserved.

UNIQUE Constraints

The `unique_checks` system variable in MariaDB Server can be utilized to disable unique checks, which might be useful in specific scenarios where enforcing unique constraints temporarily is not required. This can enhance performance during bulk inserts, for instance.

Overview

MariaDB Server supports `UNIQUE` constraints to ensure that a column's value is unique within a table:

- MariaDB Server uses unique indexes to speed up query execution and enforce unique constraints
- MariaDB Server supports single column and composite (multi-column) unique indexes
- MariaDB Community Server can have up to 64 total indexes for a given table.
- MariaDB Server can have up to 128 total indexes for a given table.
- InnoDB stores unique indexes in the same tablespace file as the clustered index and data.
- Unique indexes are B+ trees, which are very efficient for searching for exact values, performing range scans, and checking uniqueness.
- If no primary key is defined for a table, then InnoDB will use the table's first `NOT NULL` unique index as the table's primary key.
- If no primary key or `NOT NULL` unique index is defined for a table, then InnoDB will automatically create a primary key called `GEN_CLUST_INDEX`, using an internal `48-bit DB_ROW_ID` column as the key. Replication with such tables can be very slow, especially when binlog_format is `MIXED or ROW`.

unique_checks System Variable

MariaDB Server provides the unique_checks system variable, which can be used to disable unique checks.

When unique checks are disabled, the InnoDB change buffer is used for inserts into unique indexes, and duplicate values will not be detected.

Disabling unique checks can speed up bulk data loads, but it is dangerous to do so.

Creating an InnoDB Table with a Single Column Unique Index

Let's create an InnoDB table with a single column unique index after confirming that the default storage engine is InnoDB:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the table using the CREATE TABLE statement and specify the unique index with the `UNIQUE INDEX()` clause:

```
CREATE TABLE hq_sales.customers (
   customer_id BIGINT AUTO_INCREMENT NOT NULL,
   customer_name VARCHAR(500),
   customer_email VARCHAR(200),
   PRIMARY KEY(customer_id),
   UNIQUE INDEX(customer_email)
);
```

5. For a single column unique index, the unique index can also be specified with the `UNIQUE` column option:

```
CREATE TABLE hq_sales.customers (
   customer_id BIGINT AUTO_INCREMENT NOT NULL,
   customer_name VARCHAR(500),
   customer_email VARCHAR(200) UNIQUE,
   PRIMARY KEY(customer_id)
);
```

Creating an InnoDB Table with a Composite Unique Index

Let's create an InnoDB table with a composite (multi-column) unique index after confirming that the default storage engine is InnoDB:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the table using the CREATE TABLE statement and specify the unique index with the UNIQUE INDEX() clause:

```
CREATE TABLE hq_sales.invoices (
   invoice_id BIGINT UNSIGNED NOT NULL,
   branch_id INT NOT NULL,
   customer_id INT,
   invoice_date DATETIME(6),
   invoice_total DECIMAL(13, 2),
   payment_method ENUM('NONE', 'CASH', 'WIRE_TRANSFER', 'CREDIT_CARD', 'GIFT_CARD'),
   PRIMARY KEY(invoice_id),
   UNIQUE INDEX(invoice_date, customer_id, branch_id)
);
```

Creating an InnoDB Table with a Unique Index on a Column Prefix

Let's create an InnoDB table with a unique index on a single column prefix after confirming that the default storage engine is InnoDB:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the table using the CREATE TABLE statement and specify the unique index with the `UNIQUE INDEX()` clause:

```
CREATE TABLE hq_sales.products (
   product_id BIGINT AUTO_INCREMENT NOT NULL,
   product_name VARCHAR(500),
   product_brand VARCHAR(500),
   product_description TEXT,
   PRIMARY KEY(product_id),
   UNIQUE INDEX(product_description(1000))
);
```

The unique index is specified with the `product_description(1000)` prefix, so only the first 1000 characters of the `product_description` column for each row will be indexed and checked for uniqueness.

Adding a Unique Index to an InnoDB Table

Let's create an InnoDB table without a unique index, and then add a unique index to it:

1. Connect to the server using MariaDB Client:

```
$ mariadb --user=root
```

2. Confirm that the default storage engine is InnoDB by checking the default_storage_engine system variable using the SHOW SESSION VARIABLES statement:

```
SHOW SESSION VARIABLES
   LIKE 'default_storage_engine';
```

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

3. If the database does not exist, then create the database for the table using the CREATE DATABASE statement:

```
CREATE DATABASE hq_sales;
```

4. Create the table without a primary key using the CREATE TABLE statement:

```
CREATE TABLE hq_sales.customers (
   customer_id BIGINT AUTO_INCREMENT NOT NULL,
   customer_name VARCHAR(500),
   customer_email VARCHAR(200),
   PRIMARY KEY(customer_id)
);
```

5. Alter the table using the ALTER TABLE statement and specify the new unique index with the `ADD UNIQUE INDEX()` clause:

```
ALTER TABLE hq_sales.customers ADD UNIQUE INDEX (customer_email);
```

Dropping a Unique Index from an InnoDB Table

Let's drop the unique index from the table created in the Creating an InnoDB Table with a Single Column Unique Index section:

1. Obtain the name of the index by joining the information_schema.INNODB_SYS_INDEXES, information_schema.INNODB_SYS_TABLES, and information_schema.INNODB_SYS_FIELDS tables:

```
SELECT isi.NAME AS index_name, isf.NAME AS index_column
FROM information_schema.INNODB_SYS_INDEXES isi
JOIN information_schema.INNODB_SYS_TABLES ist
ON isi.TABLE_ID = ist.TABLE_ID
JOIN information_schema.INNODB_SYS_FIELDS isf
ON isi.INDEX_ID = isf.INDEX_ID
WHERE ist.NAME = 'hq_sales/customers'
ORDER BY isf.INDEX_ID, isf.POS;
```

```
+----------------+----------------+
| index_name     | index_column   |
+----------------+----------------+
| PRIMARY        | customer_id    |
| customer_email | customer_email |
+----------------+----------------+
```

2. Alter the table using the ALTER TABLE statement and specify the `DROP INDEX` clause:

```
ALTER TABLE hq_sales.customers DROP INDEX customer_email;
```

Rebuilding a Unique Index in an InnoDB Table

Let's rebuild the unique index in the table created in the Creating an InnoDB Table with a Single Column Unique Index section:

1. Obtain the name of the index by joining the information_schema.INNODB_SYS_INDEXES, information_schema.INNODB_SYS_TABLES, and information_schema.INNODB_SYS_FIELDS tables:

```
SELECT isi.NAME AS index_name, isf.NAME AS index_column
FROM information_schema.INNODB_SYS_INDEXES isi
JOIN information_schema.INNODB_SYS_TABLES ist
ON isi.TABLE_ID = ist.TABLE_ID
JOIN information_schema.INNODB_SYS_FIELDS isf
ON isi.INDEX_ID = isf.INDEX_ID
WHERE ist.NAME = 'hq_sales/customers'
ORDER BY isf.INDEX_ID, isf.POS;
```

```
+----------------+----------------+
| index_name     | index_column   |
+----------------+----------------+
| PRIMARY        | customer_id    |
| customer_email | customer_email |
+----------------+----------------+
```

2. Alter the table using the ALTER TABLE statement and specify the `DROP INDEX` clause:

```
ALTER TABLE hq_sales.customers DROP INDEX customer_email;
```

3. Alter the table using the ALTER TABLE statement and specify the unique index with the `ADD UNIQUE INDEX()` clause:

```
ALTER TABLE hq_sales.customers ADD UNIQUE INDEX (customer_email);
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Clients & Utilities

See Administrative Tools.

Explore administrative tools. This section introduces various command line utilities and graphical interfaces designed to help you manage, monitor, and configure your database efficiently.

See Aria Clients and Utilities.

Explore Aria clients and utilities for MariaDB Server. This section details specialized tools for managing and interacting with tables that utilize the Aria storage engine.

See Backup, Restore and Import Clients.

Explore backup, restore, and import clients for MariaDB Server. This section details various utilities that facilitate data protection, recovery, and seamless data loading into your databases.

See Deployment Tools.

Explore deployment tools for MariaDB Server. This section introduces utilities and scripts that simplify the installation, configuration, and management of MariaDB in various environments.

See Graphical and Enhanced Clients.

Explore graphical and enhanced clients for MariaDB Server. This section surveys GUI tools and third-party applications for querying, designing, and administering your databases.

See Logging Tools.

Explore logging tools for MariaDB Server. This section introduces utilities and methods for managing and analyzing server logs, crucial for monitoring activity and troubleshooting issues.

See mariadb Client.

Official mariadb client reference: command-line interface, mariadb/mysql naming, Linux symlink access, Windows binary, and connection syntax.

See Networking Tools.

Explore networking tools for MariaDB Server. This section introduces utilities that help you manage and troubleshoot network connections to your database, including monitoring and security.

See Table Tools.

Explore table-related tools for MariaDB Server. This section details various utilities for managing table structures, performing maintenance, and optimizing data storage.

See Testing Tools.

Explore testing tools for MariaDB Server. This section introduces utilities that help you validate server functionality, test configurations, and ensure the reliability of your database deployments.

See MyISAM Clients and Utilities.

Explore MyISAM clients and utilities for MariaDB Server. This section details specialized tools for managing and maintaining tables that utilize the MyISAM storage engine.

See Server & Client Software.

Explore server and client software for MariaDB. This section introduces various tools and applications for connecting to, managing, and interacting with your MariaDB Server instances effectively.

This page is licensed: CC BY-SA / Gnu FDL

Administrative Tools

See dbdeployer.

See innochecksum.

See mariadb-access.

See mariadb-admin.

See mariadb-conv.

See mariadb-setpermission.

See mariadb-show.

See mariadb-plugin.

See mariadb-report.

See mariadb-find-rows.

See mariadb-tzinfo-to-sql.

See mariadb-waitpid.

See my_print_defaults.

See mariadb-embedded.

See perror.

See replace.

See resolve_stack_dump.

This page is licensed: CC BY-SA / Gnu FDL

dbdeployer

> **Note:**
>
> dbdeployer is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

dbdeployer is a tool for installing multiple versions of MariaDB or MySQL in isolation from each other. It is primarily used for easily testing different server versions. It is written in Go, and is a replacement for MySQL Sandbox.

Visit www.dbdeployer.com for details on how to install and use it.

This page is licensed: CC BY-SA / Gnu FDL

innochecksum

innochecksum is a tool for printing checksums for InnoDB files.

Usage

```
innochecksum [options] file_name
```

Description

The tool reads an InnoDB tablespace file, calculates the checksum for each page, compares the calculated checksum to the stored checksum, and reports mismatches, which indicate damaged pages. It was originally developed to speed up verifying the integrity of tablespace files after power outages, but can also be used after file copies. Because checksum mismatches causes InnoDB to deliberately shut down a running server, it can be preferable to use innochecksum rather than waiting for a server in production usage to encounter damaged pages.

> **Note:**
>
> Multiple filenames can be specified by a wildcard (on non-Windows systems only).

innochecksum works with compressed pages, and includes options to analyze leaf pages to estimate how fragmented an index is and how much benefit can be gained from defragmentation.

> **Note:**
>
> innochecksum cannot be used on tablespace files that the server already has open. For such files, you should use CHECK TABLE to check tables within the tablespace. If checksum mismatches are found, you would normally restore the tablespace from backup or start the server and attempt to use mariadb-dump to make a backup of the tables within the tablespace.

Options

innochecksum supports the following options. For options that refer to page numbers, the numbers are zero-based.

Option	Description
-a, –allow-mismatches=#	Maximum checksum mismatch allowed before innochecksum terminates. Defaults to `0`, which terminates on the first mismatch.
-c, –count	Print a count of the number of pages in the file.
-e num, –end-page=#	End at this page number (0-based).
-?, –help	Display help and exits.
-I, –info	Synonym for –help.
-f, –leaf	Examine leaf index pages.
-l fn, –log=fn	Log output to the specified filename *`fn`*.
-m num, –merge=#	Leaf page count, in case of a merge, given number of consecutive pages.
-n, –no-check	Ignore the checksum verification. Before MariaDB 10.6, must be used with the `--write` option.
-p num, –page=#	Check only this page number (0-based).
-D, –page-type-dump=name	Dump the page type info for each page in a tablespace.
-S, –page-type-summary	Display a count of each page type in a tablespace.
-i, –per-page-details	Print out detailed information per page.
-u, –skip-corrupt	Skip corrupt pages.
-r, –skip-freed-pages	
-s num, –start-page=#	Start at this page number (0-based).
-C, –strict-check=name	
-v, –verbose	Verbose mode; print a progress indicator every five seconds.
-V, –version	Display version information and exit.
-w, –write=name	

Examples

Rewriting a CRC32 checksum to replace an invalid checksum:

```
innochecksum --no-check --write crc32 tablename.ibd
```

A count of each page type:

```
innochecksum --page-type-summary data/mysql/gtid_slave_pos.ibd

File::data/mysql/gtid_slave_pos.ibd
================PAGE TYPE SUMMARY==============
#PAGE_COUNT PAGE_TYPE
===============================================
       1    Index page
       0    Undo log page
       1    Inode page
       0    Insert buffer free list page
       2    Freshly allocated page
       1    Insert buffer bitmap
       0    System page
       0    Transaction system page
       1    File Space Header
       0    Extent descriptor page
       0    BLOB page
       0    Compressed BLOB page
       0    Page compressed page
       0    Page compressed encrypted page
       0    Other type of page

===============================================
Additional information:
Undo page type: 0 insert, 0 update, 0 other
Undo page state: 0 active, 0 cached, 0 to_free, 0 to_purge, 0 prepared, 0 other
index_id    #pages      #leaf_pages #recs_per_page  #bytes_per_page
24      1       1       0       0

index_id    page_data_bytes_histgram(empty,...,oversized)
24      1   0   0   0   0   0   0   0   0   0   0   0
```

This page is licensed: CC BY-SA / Gnu FDL

mariadb-access

`mariadb-access` is a tool for checking access privileges, developed by Yves Carlier.

> **CURRENT**
>
> The client tool can alternatively be called by its former name, `mysqlaccess`, via a symlink in Linux, or an alternate binary in Windows.
>
> **< 10.5**
>
> The client tool is called `mysqlaccess`.

It checks the access privileges for a host name, user name, and database combination.

> **Note:**
>
> mariadb-access checks access using only the user, db, and host tables. It does not check table, column, or routine privileges specified in the tables_priv, columns_priv, or procs_priv tables.

Usage

```
mariadb-access [host [user [db]]] OPTIONS
```

If your MariaDB distribution is installed in some non-standard location, you must change the location where *mariadb-access* expects to find the mariadb client. Edit the *mariadb-access* script at approximately line 18. Search for a line that looks like this:

```
$MYSQL = ´/usr/local/bin/mariadb;
path to mariadb executable
```

Change the path to reflect the location where the *mariadb* client tool is stored on your system. Otherwise, a *Broken pipe error* occurs when running *mariadb-access*.

Options

Option	Description
-?, –help	Display help and exit.
-v, –version	Display version.
-u username, –user=username	Username for logging in to the server.
-p[password], –password[=password]	Password to use for user. If omitted, mariadb-access prompts for one.
-h hostname, –host=hostname	Name or IP of the host.
-d dbname, –db=dbname	Name of the database.
-U username, –superuser=username	Connect as superuser.
-P password, –spassword=password	Password for superuser.
-H server, –rhost=server	Remote server to connect to.
–old_server	Connect to a very old MySQL servers (before version 3.21) that does not know how to handle full WHERE clauses.
-b, –brief	Single-line tabular report.
-t, –table	Report in table-format.
–relnotes	Print release-notes.
–plan	Print suggestions/ideas for future releases.
–howto	Some examples of how to run `mariadb-access'.
–debug=N	Enter debug level N (0..3).
–copy	Reload temporary grant tables from original ones.
–preview	Show differences in privileges after making changes in (temporary) grant tables.
–commit	Copy grant rules from temporary tables to grant tables (the grant tables must be flushed afterwards, for example with mariadb-admin reload).
–rollback	Undo the last changes to the grant tables.

Note

At least the user (`-u`) and the database (`-d`) options must be given, even when using wildcards. If no host is provided, `` `localhost' `` is assumed. Wildcards (`?` , `%` , and `_`) are allowed for host, user, and database, but be sure

to escape them from your shell.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-admin

Overview

`mariadb-admin` is an administration program for the mariadbd daemon. It can be used to:

- Monitor what the MariaDB clients are doing (processlist);
- Get usage statistics and variables from the MariaDB server;
- Create and drop databases;
- Flush (reset) logs, statistics, and tables;
- Kill running queries;
- Stop the server (shutdown);
- Start and stop replicas;
- Check if the server is alive (ping).

> **CURRENT**
>
> The client tool name is `mariadb-admin`. However, it can still be accessed under the old name, `mysqladmin`, via a symlink on Linux or an alternate binary on Windows.
>
> **< 10.5**
>
> The name of the client tool is `mysqladmin`.

Usage

The command to use `mariadb-admin` and the general syntax is:

```
mariadb-admin [options] command [command-arg] [command [command-arg]] ...
```

Options

`mariadb-admin` supports the following options:

Option	Description
–character-sets-dir=*dir*	Directory where the character set files are located.
-C, –compress	Compress all information sent between the client and the server if both support compression.
–connect_timeout=*val*	Maximum time in seconds before connection timeout. The default value is `43200` (12 hours).
-c *val*, –count=*val*	Number of iterations to make. This works with `-i` (`--sleep`) only.

Option	Description
-# [*debug_options*], –debug[=*debug_options*]	Write a debugging log. A typical `debug_options` string is `d:t:o,file_name`. The default is `d:t:o,/tmp/mysqladmin.trace`.
–debug-check	Check memory and open file usage on exit.
–debug-info	Print debugging information, memory, and CPU usage statistics when the program exits.
–default-auth=*plugin*	Default authentication client-side plugin to use.
–default-character-set=*name*	Set the default character set.
-f, –force	Don't ask for confirmation on drop database. With multiple statements, continue even if an error occurs.
-?, –help	Display help and exit.
-h *name*, –host=*name*	Hostname to connect to.
-l, –local	Suppress SQL statements from being written to the binary log, by enabling sql_log_bin=0 for flush commands, using `FLUSH LOCAL` rather than `SET sql_log_bin=0`. The privilege required is `RELOAD`.
-b, –no-beep	Turn off beep on error.
-p[*password*], –password[=*password*]	Password to use when connecting to server. If password is not given, it's asked from the terminal.
–pipe, -W	On Windows, connect to the server via a named pipe. This option applies only if the server supports named-pipe connections.
-P *portnum*, –port=*portnum*	
–protocol={tcp|socket|pipe|memory}	The protocol to use for the connection (`tcp`, `socket`, `pipe`, `memory`).
-r, –relative	Show difference between current and previous values when used with `-i`. Only works in combination with `extended-status`.
-O *val*, –set-variable=*val*	Change the value of a variable to *val*. This option is deprecated; you can set variables directly with `--variable-name=val`.

Option	Description
–shutdown_timeout=*val*	Maximum number of seconds to wait for server shutdown. The default value is 3600 (1 hour).
-s, –silent	Silently exit if you can't connect to server.
-i *delay*, –sleep=*delay*	Execute commands repeatedly, sleeping for *delay* seconds in between. The `--count` option determines the number of iterations. If `--count` is not given, `mariadb-admin` executes commands indefinitely until interrupted.
-S *name*, –socket=*name*	For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.
–ssl	Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The `--ssl` option doesn't enable verifying the server certificate by default. In order to verify the server certificate, you must specify the `--ssl-verify-server-cert` option.
–ssl-ca=*name*	Define a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires the use of an absolute path, not a relative path. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option implies the `--ssl` option.
–ssl-capath=*name*	Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires the use of an absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the `--ssl` option.
–ssl-cert=*name*	Define a path to the X509 certificate file to use for TLS. This option requires the use of an absolute path, not a relative path. This option implies the `--ssl` option.
–ssl-cipher=*name*	List of permitted ciphers or cipher suites to use for TLS. This option implies the `--ssl` option.
–ssl-crl=*name*	Define a path to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL or Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
–ssl-crlpath=*name*	Define a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
–ssl-key=*name*	Define a path to a private key file to use for TLS. This option requires the use of an absolute path, not a relative path. This option implies the `--ssl` option.
–ssl-verify-server-cert	Enable server certificate verification. This option is disabled by default.
–tls-version=*name*	Accept a comma-separated list of TLS protocol versions. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted. See Secure Connections Overview: TLS Protocol Versions for more information.

Option	Description
-u, –user=*name*	User for login, if not current user.
-v, –verbose	Write more information.
-V, –version	Output version information and exit.
-E, –vertical	Print output vertically. Is similar to '–relative', but prints output vertically.
-w[*count*], –wait[=*count*]	If the connection cannot be established, wait and retry instead of aborting. If a count value is given, it indicates the number of times to retry. The default is one time.
–wait-for-all-slaves	Wait for the last binlog event to be sent to all connected replicas before shutting down. This option is off by default.

Option Files

In addition to reading options from the command line, `mariadb-admin` also reads options from option files. If an unknown option is provided to `mariadb-admin` in an option file, the option is ignored.

The following options relate to how MariaDB command line tools handles option files. They must be given as the first argument on the command line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

`mariadb-admin` is linked with MariaDB Connector/C. However, MariaDB Connector/C does not yet handle the parsing of option files for this client. That is still performed by the server option file parsing code. See MDEV-19035 for more information.

Option Groups

`mariadb-admin` reads options from the following option groups.

Group	Description
[mysqladmin]	Options read by `mysqladmin`, which includes both MariaDB Server and MySQL Server.
[mariadb-admin]	Options read by mariadb-admin.
[client]	Options read by all MariaDB and MySQL client programs, which includes both MariaDB and MySQL clients. For example, mysqldump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

Variables

Variables can be set with `--variable-name=value`.

Variables and boolean options	Default value
count	`0`
debug-check	`FALSE`
debug-info	`FALSE`
force	`FALSE`
compress	`FALSE`
character-sets-dir	(No default value)
default-character-set	(No default value)
host	(No default value)
no-beep	`FALSE`
port	`3306`
relative	`FALSE`
socket	`/var/run/mariadbd/mariadbd.sock`
sleep	`0`
ssl	`FALSE`
ssl-ca	(No default value)
ssl-capath	(No default value)
ssl-cert	(No default value)
ssl-cipher	(No default value)
ssl-key	(No default value)
ssl-verify-server-cert	`FALSE`
user	(No default value)
verbose	`FALSE`
vertical	`FALSE`
connect_timeout	43200
shutdown_timeout	3600

Commands

```
mariadb-admin [options] command [command-arg] [command [command-arg]] ...
```

Command is one or more of the following. Commands can be shortened to a unique prefix.

Command	Description
create *db*	Create a database named *db*.
debug	Instruct server to write debug information to log.
drop *db*	Delete a database named *db* and all its tables.

Command	Description
extended-status	Return all status variables and their values.
flush-all-statistics	Flush all statistics tables
flush-all-status	Flush status and statistics.
flush-binary-log	Flush binary log.
flush-client-statistics	Flush client statistics.
flush-engine-log	Flush engine log.
flush-error-log	Flush error log.
flush-general-log	Flush general query log.
flush-hosts	Flush all cached hosts.
flush-index-statistics	Flush index statistics.
flush-logs	Flush all logs.
flush-privileges	Reload grant tables (same as reload).
flush-relay-log	Flush relay log.
flush-slow-log	Flush slow query log.
flush-ssl	Flush SSL certificates. Available from MariaDB 10.6.0.
flush-status	Clear status variables.
flush-table-statistics	Clear table statistics.

Command	Description
flush-tables	Flush all tables.
flush-threads	Flush the thread cache.
flush-user-resources	Flush user resources.
flush-user-statistics	Flush user statistics.
kill id,id,…	Kill mysql threads.
password new-password	Change old password to new-password. The new password can be passed on the command line as the next argument (for example, mariadb-admin password “new_password”, or can be omitted (as long as no other command follows), in which case the user is prompted for a password. If the password contains special characters, it needs to be enclosed in quotation marks. In Windows, the quotes can only be double quotes, as single quotes are assumed to be part of the password. If the server was started with the –skip-grant-tables option, changing the password in this way will have no effect.
old-password new-password	Change old password to new-password using the old pre-MySQL 4.1 format.
ping	Check if `mariadbd` is alive. Return status is `0` if the server is running (even in the case of an error such as access denied), `1` if it is not.
processlist	Show list of active threads in server, equivalent to SHOW PROCESSLIST. With `--verbose`, it is an equivalent to SHOW FULL PROCESSLIST.
reload	Reload grant tables.
refresh	Flush all tables and close and open log files.
shutdown	Take the server down by executing the SHUTDOWN command on the server. If connected to a local server using a Unix socket file, `mariadb-admin` waits until the server’s process ID file is removed, to ensure that the server stopped properly. See also the `--wait-for-all-slaves` option.
status	Give a short status message from the server.
start-all-slaves	Start all replicas.
start-slave	Start replication on a replica server.
stop-all-slaves	Stop all replicas.
stop-slave	Stop replication on a replica server.
variables	Print variables available.
version	Return version as well as status info from the server.

The shutdown Command and the –wait-for-all-slaves Option

The `--wait-for-all-slaves` option can be used in the following scenario.

When a primary server is shut down and goes through the normal shutdown process, it kills client threads in random order. By default, the primary also considers its binary log dump threads to be regular client threads. As a consequence, the binary log dump threads can be killed while client threads still exist. This means that data can be written on the primary during a normal shutdown that won't be replicated. This is true even if semi-synchronous replication is being used.

This problem can be solved by shutting down the server with the `mariadb-admin` utility and by providing the `--wait-for-all-slaves` option:

```
mariadb-admin --wait-for-all-slaves shutdown
```

When the `--wait-for-all-slaves` option is provided, the server only kills its binary log dump threads after all client threads have been killed, and it only completes the shutdown after the last binary log has been sent to all connected replicas.

See Replication Threads: Binary Log Dump Threads and the Shutdown Process for more information.

Examples

Quick check of what the server is doing:

```
shell> mariadb-admin status
Uptime: 8023 Threads: 1 Questions: 14 Slow queries: 0 Opens: 15 Flush tables: 1 Open tables: 8 Queries
per second avg: 0.1
shell> mariadb-admin processlist
+----+-------+-----------+----+---------+------+-------+------------------+
| Id | User | Host | db | Command | Time | State | Info |
+----+-------+-----------+----+---------+------+-------+------------------+
....
+----+-------+-----------+----+---------+------+-------+------------------+
```

More extensive information of what is currently happening, great for troubleshooting a slow server:

```
shell> mariadb-admin --relative --sleep=1 extended-status | grep -v " 0 "
```

Check the variables for a running server:

```
shell> mariadb-admin variables | grep datadir
| datadir | /my/data/ |
```

Use a shortened prefix for the `version` command:

```
shell> mariadb-admin ver
mariadb-admin from 11.1.0-preview-MariaDB, client 9.1 for linux-systemd (x86_64)
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Server version      11.1.0-preview-MariaDB
Protocol version    10
Connection      localhost via TCP/IP
TCP port        11100
Uptime:         3 min 21 sec

Threads: 1  Questions: 1  Slow queries: 0  Opens: 17  Open tables: 10  Queries per second avg: 0.004
```

Other Ways To Stop mariadbd (Unix)

If connecting as a user that does not have the `SUPER` or `SHUTDOWN` privilege, an error occurs:

```
mariadb-admin: shutdown failed; error: 'Access denied; you need (at least one of) the SHUTDOWN
privilege(s) for this operation'
```

If you don't know the user password, you can still take the `mariadbd` process down like this:

UNIX-LIKE SYSTEMS

```
kill -SIGTERM pid-of-mariadbd-process
```

This command is identical to `mariadb-admin shutdown`.

WINDOWS

```
NET STOP MySQL
```

You can use the SHUTDOWN command from any client.

See Also

- SHUTDOWN
- mytop, a tool similar to `top` on Unix-like systems, that allows to see what the server is doing.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-conv

> **Note:**
>
> This tool is available from MariaDB 10.5.

`mariadb-conv` is a character set conversion utility for MariaDB.

Usage

```
mariadb-conv [OPTION...] [FILE...]
```

Options

`mariadb-conv` supports the following options:

Option	Description
-f, –from=name	Specifies the encoding of the input.
-t, –to=name	Specifies the encoding of the output.
-c, –continue	Silently ignore conversion errors.
–delimiter=name	Treat the specified characters as delimiters.

By default, `mariadb-conv` exits whenever it encounters any conversion problems, like

- the input byte sequence is not valid in the source character set;
- the character cannot be converted to the target character set.

The `-c` option makes `mariadb-conv` ignore such errors and use the question mark (`?`) to replace bytes in bad input sequences, or unconvertible characters.

The `--delimiter=...` option makes `mariadb-conv` treat the specified characters as delimiters rather than data to convert, so the input is treated as a combination of

- data chunks, which are converted according to the `-f` and `-t` options;
- delimiters, which are not converted and are copied from the input to the output as is.

Examples

Convert the file `file.latin1.txt` from `latin1` to `utf8` :

```
mariadb-conv -f latin1 -t utf8 file.latin1.txt
```

Convert the file `file.latin1.txt` from `latin1` to `utf8`, reading the input data from `stdin`:

```
mariadb-conv -f latin1 -t utf8 < file.latin1.txt
```

Using `mariadb-conv` in a pipe:

```
echo test | ./mariadb-conv -f utf8 -t ucs2 >file.ucs2.txt
```

As a side effect, `mariadb-conv` can be used to list MariaDB data directories in a human readable form. Suppose you create the following tables:

```
SET NAMES utf8;
CREATE OR REPLACE TABLE t1 (a INT);
CREATE OR REPLACE TABLE ß (a INT);
CREATE OR REPLACE TABLE абв (a INT);
CREATE OR REPLACE TABLE ▫▫ (a INT);
```

This statement creates the following files in the MariaDB data directory:

```
@1j.frm
@1j.ibd
@684c@5b50.frm
@684c@5b50.ibd
@g0@h0@i0.frm
@g0@h0@i0.ibd
t1.frm
t1.ibd
```

It's not precisely clear which file stores which table, because MariaDB uses a special table-name-to-file-name encoding.

This command on Linux (assuming an utf-8 console) can print the table list in a readable way:

```
ls | mariadb-conv -f filename -t utf8 --delimiter=".\n"

ß.frm
ß.ibd
▫▫.frm
▫▫.ibd
абв.frm
абв.ibd
t1.frm
t1.ibd
```

> **Note:**
>
> The `--delimiter="\n"` option is needed to make `mariadb-conv` treat the dot character (delimiting the encoded table name from the file extension) and the newline character (delimiting separate lines) as delimiters, rather than as the data to convert (otherwise the conversion would fail).

Windows users can use the following command to list the data directory in the ANSI text console:

```
dir /b | mariadb-conv -c -f filename -t cp850 --delimiter=".\r\n"
```

Note:

- The `-t` options assume a Western machine.
- The `-c` option is needed to ignore conversion errors for Cyrillic and CJK characters.
- `--delimiter=` additionally needs the carriage return character.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-setpermission

Syntax

```
mariadb-setpermission [options]
```

Description

`mariadb-setpermission` is a Perl script that was originally written and contributed by Luuk de Boer. It requires the DBI and DBD::mysql Perl modules to be installed. `mariadb-setpermission` can help you add users or databases or change passwords in MariaDB.

It interactively sets permissions in the MariaDB grant tables, but does not check permissions which have already been set in MariaDB. So if you can't connect to MariaDB using the permission you just added, take a look at the permissions which have already been set in MariaDB.

The account used when you connect determines which permissions you have when attempting to modify existing permissions in the grant tables.

`mariadb-setpermission` also reads options from the `[client]` and `[perl]` groups in the `.my.cnf` file in your home directory, if the file exists.

The following options are available:

Options

–help	Display a help message and exit.
–host=host_name	Connect to the MariaDB server on the given host.
–password=password	The password to use when connecting to the server. Note that the password value is not optional for this option, unlike for other MariaDB programs Specifying a password on the command line should be considered insecure. You can use an option file to avoid giving the password on the command line.
–port=port_num	The TCP/IP port number to use for the connection.
–socket=path	For connections to localhost, the Unix socket file to use.

–user=user_name	The MariaDB user name to use when connecting to the server.

Example

```
./mariadb-setpermission --user=msandbox --password=msandbox --host=127.0.0.1 --port=11200
########################################################################
## Welcome to the permission setter 1.4 for MariaDB.
## made by Luuk de Boer
########################################################################
What would you like to do:
  1. Set password for an existing user.
  2. Create a database + user privilege for that database
     and host combination (user can only do SELECT)
  3. Create/append user privilege for an existing database
     and host combination (user can only do SELECT)
  4. Create/append broader user privileges for an existing
     database and host combination
     (user can do SELECT,INSERT,UPDATE,DELETE)
  5. Create/append quite extended user privileges for an
     existing database and host combination (user can do
     SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,INDEX,
     LOCK TABLES,CREATE TEMPORARY TABLES)
  6. Create/append full privileges for an existing database
     and host combination (user has FULL privilege)
  7. Remove all privileges for an existing database and
     host combination.
     (user will have all permission fields set to N)
  0. exit this program
```

This page is licensed: CC BY-SA / Gnu FDL

mariadb-show

Shows the structure of a MariaDB database (databases, tables, columns and indexes).

To get similar functionality, you can use SHOW DATABASES, SHOW TABLES, SHOW COLUMNS, SHOW INDEX and SHOW TABLE STATUS, as well as the Information Schema tables (TABLES, COLUMNS, STATISTICS).

Usage

```
mariadb-show [OPTIONS] [database [table [column]]]
```

The output displays only the names of those databases, tables, or columns for which you have some privileges.

If no database is given then all matching databases are shown. If no table is given, then all matching tables in database are shown. If no column is given, then all matching columns and column types in table are shown.

If the last argument contains a shell or SQL wildcard (`*` , `?` , `%` , or `_`), only what's matched by the wildcard is shown. If a database name contains any underscores, those should be escaped with a backslash (some Unix shells require two) to get a list of the proper tables or columns. The `*` and `?` characters are converted into SQL `%` and `_` wildcard characters. This might cause some confusion when trying to display the columns for a table whose name contains an underscore character (`_`), because `mariadb-show` shows only the table names that match the pattern. To fix this, add an extra `%` last on the command line, as a separate argument.

Options

`mariadb-show` supports the following options:

Option	Description
-c name, –character-sets-dir=name	Directory for character set files.
-C, –compress	Use compression in server/client protocol if both support it.
–count	Show number of rows per table (may be slow for non-MyISAM tables).
[name], –debug[=name]	Output debug log. Typical is `d:t:o,filename`, the default is `d:t:o`.
–debug-check	Check memory and open file usage at exit.
–debug-info	Print some debug info at exit.
–default-auth=name	Default authentication client-side plugin to use.
–default-character-set=name	Set the default character set.
–defaults-extra-file=name	Read the file name after the global files are read. Must be given as the first option.
–defaults-file=name	Only read default options from the given file name. Must be given as the first option.
–defaults-group-suffix=suffix	In addition to the given groups, also read groups with this suffix.
-?, –help	Display help and exit.
-h name, –host=name	Connect to the MariaDB server on the given host.
-k, –keys	Show indexes for table.
–no-defaults	Don’t read default options from any option file. Must be given as the first option.

Option	Description
-p[password], –password[=password]	Password to use when connecting to server. If password is not given, it's solicited on the command line. Specifying a password on the command line should be considered insecure. You can use an option file to avoid giving the password on the command line.
-W, –pipe	On Windows, connect to the server via a named pipe. This option applies only if the server supports named-pipe connections.
–plugin-dir=name	Directory for client-side plugins.
-P num, –port=num	Port number to use for connection or 0 for default to, in order of preference, `my.cnf`, `$MYSQL_TCP_PORT`, `/etc/services`, built-in default (`3306`).
–print-defaults	Print the program argument list and exit. Must be given as the first option.
–protocol=name	The protocol to use for connection (tcp, socket, pipe, memory).
–shared-memory-base-name=name	On Windows, the shared-memory name to use, for connections made using shared memory to a local server. The default value is MYSQL. The shared-memory name is case sensitive. The server must be started with the –shared-memory option to enable shared-memory connections.
-t, –show-table-type	Show table type column, as in SHOW FULL TABLES. The type is `BASE TABLE` or `VIEW`.
-S name, –socket=name	For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.
–ssl	Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The `--ssl` option does not enable verifying the server certificate by default. In order to verify the server certificate, the user must specify the `--ssl-verify-server-cert` option.
–ssl-ca=name	Defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option implies the –ssl option.
–ssl-capath=name	Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the –ssl option.
–ssl-cert=name	Defines a path to the X509 certificate file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the –ssl option.
–ssl-cipher=name	List of permitted ciphers or cipher suites to use for TLS. This option implies the –ssl option.

Option	Description
–ssl-key=name	Defines a path to a private key file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the –ssl option.
–ssl-crl=name	Defines a path to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL or Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the –ssl option.
–ssl-crlpath=name	Defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the –ssl option.
–ssl-verify-server-cert	Enables (or disables) server certificate verification. This option is disabled by default.
-i, –status	Shows a lot of extra information about each table. See the INFORMATION_SCHEMA.TABLES table for more details on the returned information.
–tls-version=name	This option accepts a comma-separated list of TLS protocol versions. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted. See Secure Connections Overview: TLS Protocol Versions for more information.
-u, –user=name	User for login, if not current user.
-v, –verbose	More verbose output; you can use this multiple times to get even more verbose output.
-V, –version	Output version information and exit.

Option Files

In addition to reading options from the command line, `mariadb-show` can also read options from option files. If an unknown option is provided to `mariadb-show` in an option file, then it is ignored.

The following options relate to how MariaDB command line tools handles option files. They must be given as the first argument on the command line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

`mariadb-show` is linked with MariaDB Connector/C. However, MariaDB Connector/C does not handle the parsing of option files for this client. This is performed by the server option file parsing code. See MDEV-19035 for more information.

Option Groups

`mariadb-show` reads options from the following option groups from option files:

Group	Description
[mysqlshow]	Options read by `mysqlshow`, which includes both MariaDB Server and MySQL Server.
[mariadb-show]	Options read by mariadb-show.
[client]	Options read by all MariaDB and MySQL client programs, which includes both MariaDB and MySQL clients.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

Examples

Getting a list of databases:

```
bin/mariadb-show
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| test               |
+--------------------+
```

Getting a list of tables in the `test` database:

```
bin/mariadb-show test
Database: test
+---------+
| Tables  |
+---------+
| author  |
| book    |
| city    |
| country |
+---------+
```

Getting a list of columns in the `test`.`book` table:

```
bin/mariadb-show test book
Database: test  Table: book
+-----------+-----------------------+-------------------+------+-----+---------+----------------+--------
-----------------------+---------+
| Field     | Type                  | Collation         | Null | Key | Default | Extra          |
Privileges                     | Comment |
+-----------+-----------------------+-------------------+------+-----+---------+----------------+--------
-----------------------+---------+
| id        | mediumint(8) unsigned |                   | NO   | PRI |         | auto_increment |
select,insert,update,references |         |
| title     | varchar(200)          | latin1_swedish_ci | NO   |     |         |                |
select,insert,update,references |         |
| author_id | smallint(5) unsigned  |                   | NO   | MUL |         |                |
select,insert,update,references |         |
+-----------+-----------------------+-------------------+------+-----+---------+----------------+--------
-----------------------+---------+
```

This page is licensed: CC BY-SA / Gnu FDL

mariadb-plugin

`mariadb-plugin` is a tool for enabling or disabling plugins.

The client was previously called `mysql_plugin`, and can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

It is a command line alternative to the INSTALL PLUGIN and UNINSTALL PLUGIN statements, and the `--plugin-load option` to mariadbd.

`mariadb-plugin` must be run while the server is offline, and works by adding or removing rows from the mysql.plugin table.

`mariadb-plugin` basically has two use cases:

1. To add a plugin even before the first real server startup.
2. To remove a plugin that crashes the server on startup.

For the installation use case, adding a plugin-load-add entry to `my.cnf` or in a separate include option file is a better alternative. In case of a plugin loaded via a `mysql.plugin` crashing the server, uninstalling the plugin with the help of `mariadb-plugin` can be the only viable action, though.

Usage

```
mariadb-plugin [options] <plugin> ENABLE|DISABLE
```

`mariadb-plugin` expects to find a configuration file that indicates how to configure the plugins. The configuration file has, by default, the same name as the plugin, with an `.ini` extension:

```
mariadb-plugin crazyplugins ENABLE
```

Here, `mariadb-plugin` looks for a file called `crazyplugins.ini` :

```
crazyplugins
crazyplugin1
crazyplugin2
crazyplugin3
```

The first line should contain the name of the library object file, with no extension. The other lines list the names of the components. Each value should be on a separate line. The `#` character at the start of a line indicates a comment.

Options

The following options can be specified on the command line, while some can be specified in the `[mysqld]` group of any option file. For options specified in a `[mysqld]` group, only the `--basedir`, `--datadir`, and `--plugin-dir` options can be used — the rest are ignored.

Option	Description
-b, –basedir=name	The base directory for the server.
-d, –datadir=name	The data directory for the server.
-?, –help	Display help and exit.
-f, –my-print-defaults=name	Path to my_print_defaults executable. Example: /source/temp11/extra
-m, –mysqld=name	Path to mysqld executable. Example: /sbin/temp1/mysql/bin
-n, –no-defaults	Do not read values from configuration file.
-p, –plugin-dir=name	The plugin directory for the server.

Option	Description
-i, –plugin-ini=name	Read plugin information from configuration file specified instead of from /<plugin_name>.ini.
-P, –print-defaults	Show default values from configuration file.
-v, –verbose	More verbose output; you can use this multiple times to get even more verbose output.
-V, –version	Output version information and exit.

See Also

- List of Plugins
- Plugin Overview
- INFORMATION_SCHEMA.PLUGINS Table
- INSTALL PLUGIN
- INSTALL SONAME
- UNINSTALL PLUGIN
- UNINSTALL SONAME

This page is licensed: CC BY-SA / Gnu FDL

mariadb-report

`mariadb-report` makes a friendly report of important MariaDB status values.

Previously, the client was called `mysqlreport`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

`mariadb-report` makes a report of nearly every status value from SHOW STATUS. Unlike `SHOW STATUS`, which simply dumps over 100 values to the screen in one long list, `mariadb-report` interprets and formats the values, and presents the basic values and many more inferred values in a human-readable format. Numerous example reports are available in this report: hackmysql.com/mysqlreport, archived here.

The benefit of mariadb-report is that it allows you to very quickly see a wide array of performance indicators for your MariaDB server which would otherwise need to be calculated by hand from all the various `SHOW STATUS` values. For example, the Index Read Ratio is an important value but it's not present in `SHOW STATUS`; it's an inferred value (the ratio of `Key_reads` to `Key_read_requests`).

This documentation outlines all the command line options in `mariadb-report`, most of which control which reports are printed. This document does not address how to interpret these reports; that topic is covered in the document Guide To Understanding mysqlreport, archived here.

Usage

```
mariadb-report [options]
```

mariadb-report options

Technically, command line options are in the form `--option`, but `-option` works, too. All options can be abbreviated if the abbreviation is unique. For example, option `--host` can be abbreviated to `--ho` but not `--h` because `--h` is ambiguous: it could mean `--host` or `--help`.

Option	Description
–all	Equivalent to `--dtq --dms --com 3 --sas --qcache`. (Note that `--tab` is not invoked by `--all`.)

Option	Description
–com N	Print top N number of non-DMS Com_ status values in descending order (after DMS in Questions report). If N is not given, default is 3. Such non-DMS Com_ values include Com_change_db, Com_show_tables, Com_rollback, etc.
–dms	Print Data Manipulation Statements (DMS) report (under DMS in Questions report). DMS are those from the Data Manipulation section. mariadb-report considers only SELECT, INSERT, REPLACE, UPDATE, and DELETE. Each DMS is listed in descending order by count.
–dtq	Print Distribution of Total Queries (DTQ) report (under Total in Questions report). Queries (or Questions) can be divided into four main areas: `DMS` (see `--dms`), `Com_` (see `--com`), `COM_QUIT` (see `COM_QUIT` and Questions, archived here), and `Unknown`. `--dtq` lists the number of queries in each of these areas in descending order.
–email *address*	After printing the report to screen, email the report to *address*. This option requires sendmail in `/usr/sbin/`, therefore it does not work on Windows. `/usr/sbin/sendmail` can be a symlink to qmail, or any MTA (mail transfer agent) that emulates sendmail's `-t` command line option and operation. The FROM: field is "mariadb-report", SUBJECT: is "MySQL status report".
–flush-status	Execute a FLUSH STATUS after generating the reports. If you do not have permissions in MariaDB to do this an error from `DBD::mysql::st` is printed after the reports.
–help	Output help information and exit.
–host ADDRESS	Host address.
–infile *file*	Instead of getting SHOW STATUS values from MariaDB, read values from *file*. *file* is often a copy of the output of `SHOW STATUS` including formatting characters (+, -). `mariadb-report` expects *file* to have the format "*value* number", where *value* is only alpha and underscore characters (A-Z and _) and number is a positive integer. Anything before, between, or after value and number is ignored. mariadb-report also needs the following MariaDB server variables: version, table_cache, max_connections, key_buffer_size, query_cache_size. These values can be specified in `INFILE` in the format "name = value", where name is one of the aforementioned server variables and value is a positive integer with or without a trailing M and possible periods (for version). For example, to specify an 18M key_buffer_size, specify `key_buffer_size = 18M`. Or, for a 256 byte table_cache, specify `table_cache = 256`. The M implies Megabytes, so 18M means 18,874,368. If these server variables are not specified, the following defaults are used (respectively) which may cause strange values to be reported: 0.0.0, 64, 100, 8M, 0.
–no-mycnf	Makes `mariadb-report` not read `/.my.cnf` which it does by default otherwise. `--user` and `--password` always override values from `/.my.cnf`.
–outfile *file*	After printing the report to screen, print the report to *file* too. Internally, `mariadb-report` always writes the report to a temporary file first: `/tmp/mysqlreport`.PID on *nix, c:sqlreport. PID on Windows (PID is the script's process ID). Then it prints the temp file to screen. Then if –outfile is specified, the temp file is copied to `OUTFILE`. After `--email` (above), the temp file is deleted.
–password	`--password` can take the password on the command line, like `--password FOO`. Using `--password` without an argument causes `mariadb-report` to prompt for a password.
–port *port*	Port number.

Option	Description
–qcache	Print Query Cache report.
–sas	Print report for Select_ and Sort_ status values (after Questions report). See MySQL Select and Sort Status Variables, archived here.
–socket *socket*	For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.
–tab	Print Threads, Aborted, and Bytes status reports (after Created temp report). The Threads report reports on all Threads_ status values.
–user *username*	Username.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-find-rows

`mariadb-find-rows` reads files containing SQL statements and extracts statements that match a given regular expression or that contain USE db_name or SET statements.

Previously, the client was called `mysql_find_rows`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

The utility was written for use with update log files (as used prior to MySQL 5.0), and as such expects statements to be terminated with semicolon (`;`) characters. It may be useful with other files that contain SQL statements, as long as statements are terminated with semicolons.

Usage

```
mariadb-find-rows [options] [file_name ...]
```

Each file_name argument should be the name of file containing SQL statements. If no file names are given, the tool reads from `stdin` (standard input).

Options

`mariadb-find-rows` supports the following options:

Option	Description
–help, –Information	Display help and exit.
–regexp=*pattern*	Display queries that match *pattern*.
–rows=N	Quit after displaying *N* queries.
–skip-use-db	Do not include USE db_name statements in the output.
–start_row=*N*	Start output from this row (first row is `1`).

Examples

```
mariadb-find-rows --regexp=problem_table --rows=20 < update.log
mariadb-find-rows --regexp=problem_table  update-log.1 update-log.2
```

This page is licensed: CC BY-SA / Gnu FDL

mariadb-tzinfo-to-sql

`mariadb-tzinfo-to-sql` is a tool used to load time zones on systems that have a zoneinfo database to load the time zone tables (time_zone, time_zone_leap_second, time_zone_name, time_zone_transition and time_zone_transition_type) into the mysql database.

Previously, the client was called `mysql_tzinfo_to_sql`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Most Linux, Mac OS X, FreeBSD and Solaris systems will have a zoneinfo database - Windows does not. The database is commonly found in the `/usr/share/zoneinfo` directory, or, on Solaris, the `/usr/share/lib/zoneinfo` directory.

Usage

`mariadb-tzinfo-to-sql` can be called in several ways. The output is usually passed straight to the mariadb client for direct loading in the mysql database.

```
shell> mariadb-tzinfo-to-sql timezone_dir
shell> mariadb-tzinfo-to-sql timezone_file timezone_name
shell> mariadb-tzinfo-to-sql --leap timezone_file
```

Resetting timezone tables

If there is a need to reset the timezone to the default, to before using `mariadb-tzinfo-to-sql`, one can do that by executing:

```
TRUNCATE TABLE mysql.time_zone;
TRUNCATE TABLE mysql.time_zone_name;
TRUNCATE TABLE mysql.time_zone_transition;
TRUNCATE TABLE mysql.time_zone_transition_type;
TRUNCATE TABLE mysql.time_zone_leap_second;
```

The old timezone values is in effect until the server is restarted.

Examples

Most commonly, the whole directory is passed:

```
shell> mariadb-tzinfo-to-sql /usr/share/zoneinfo | mariadb -u root mysql
```

Load a single time zone file, `timezone_file`, corresponding to the time zone called `timezone_name` :

```
shell> mariadb-tzinfo-to-sql timezone_file timezone_name | mariadb -u root mysql
```

A separate command for each time zone and time zone file the server needs is required.

To account for leap seconds, use:

```
shell> mariadb-tzinfo-to-sql --leap timezone_file | mariadb -u root mysql
```

After populating the time zone tables, you should usually restart the server so that the new time zone data are correctly loaded.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-waitpid

`mariadb_waitpid` is a utility for terminating processes.

> **Note:**
>
> It runs on Unix-like systems, making use of the `kill()` system call.

Previously, the client was called `mysql_waitpid`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Usage

```
mariadb-waitpid [options] pid time
```

Description

`mariadb-waitpid` sends signal `0` to the process *pid* and waits up to *time* seconds for the process to terminate. *pid*. *time* must be positive integers.

Returns `0` if the process terminates in time or does not exist, and `1` otherwise.

Signal `1` is used if the `kill()` system call cannot handle signal `0`.

Options

Option	Description
-?, –help	Display help and exit.
-I, –help	Synonym for `-?`.
-v, –verbose	Be more verbose. Give a warning, if kill can't handle signal `0`.
-V, –version	Print version information and exit.

This page is licensed: CC BY-SA / Gnu FDL

my_print_defaults

`my_print_defaults` displays the options from option groups of option files. It is useful to see which options a particular tool will use.

Output is one option per line, displayed in the form in which they would be specified on the command line.

Usage

```
my_print_defaults [OPTIONS] [groups]
```

Options

Option	Description
-c, –config-file=*name*	Deprecated, use `--defaults-file` instead. Name of config file to read; if no extension is given, default extension (`.ini` or `.cnf`) are added.

Option	Description
-d, –debug[=#]	In debug versions, write a debugging log. A typical debug_options string is `d:t:o,file_name`. The default is `d:t:o,/tmp/my_print_defaults.trace`.
-c, –defaults-file=*name*	Like `--config-file`, except if it is the first option. In that case, read this file only, do not read global or per-user config files; should be the first option. Removed in MariaDB 10.8.
-e, –defaults-extra-file=*name*	Read this file after the global config file and before the config file in the users home directory; should be the first option. Removed in MariaDB 10.8.
-g, –defaults-group-suffix=*name*	In addition to the given groups, read also groups with this suffix. Removed in MariaDB 10.8.
-e, –extra-file=*name*	Deprecated. Synonym for `--defaults-extra-file`.
–mariadbd	Read the same set of groups that the mariadbd server does. Available from MariaDB 10.11.3.
–mysqld	Read the same set of groups that the mysqld server does.
-n, –no-defaults	Return an empty string (useful for scripts).
?, –help	Display this help message and exit.
-v, –verbose	Increase the output level.
-V, –version	Output version information and exit.

Examples

```
my_print_defaults --defaults-file=example.cnf client client-server mysql
```

mariadb-check reads from the `[mariadb-check]` and `[client]` sections in option files, so the following would display the mariadb-check options.

```
my_print_defaults mariadb-check client
```

This page is licensed: CC BY-SA / Gnu FDL

mariadb-embedded

`mariadb-embedded` is a mariadb client, statically linked to `libmariadbd`, the embedded server.

Previously, the client was called `mysql_embedded`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Upon execution, an embedded MariaDB server is instantiated, and you can execute statements just as you would using the normal mariadb client, using the same options.

> **Important:**
>
> Do not run *mariadb-embedded* using the same database as a running MariaDB server!

Examples

```
sudo mariadb-embedded -e 'select user, host, password from mysql.user where user="root"'
+------+-----------+-------------------------------------------+
| user | host      | password                                  |
+------+-----------+-------------------------------------------+
| root | localhost | *196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7 |
| root | db1       | *196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7 |
| root | 127.0.0.1 | *196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7 |
| root | ::1       | *196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7 |
+------+-----------+-------------------------------------------+
```

Sending options with `--server-arg`:

```
sudo mariadb-embedded --server-arg='--skip-innodb'
  --server-arg='--default-storage-engine=myisam'
  --server-arg='--log-error=/tmp/mysql.err'
  -e 'select user, host, password from mysql.user where user="root"'
+------+-----------+-------------------------------------------+
| user | host      | password                                  |
+------+-----------+-------------------------------------------+
| root | localhost | *196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7 |
| root | db1       | *196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7 |
| root | 127.0.0.1 | *196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7 |
| root | ::1       | *196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7 |
+------+-----------+-------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

perror

perror is a utility that displays descriptions for system or storage engine error codes.

See MariaDB Error Codes for a full list of MariaDB error codes, and Operating System Error Codes for a list of Linux and Windows error codes.

Usage

```
perror [OPTIONS] [ERRORCODE [ERRORCODE...]]
```

If you need to describe a negative error code, use `--` before the first error code to end the options.

Options

Option	Description
-?, –help	Display help and exit.
-I, –info	Synonym for –help.
-s, –silent	Only print the error message.
-v, –verbose	Print error code and message (default). (Defaults to on; use –skip-verbose to disable.)
-V, –version	Displays version information and exits.

Examples

System error code:

```
shell> perror 96
OS error code  96:  Protocol family not supported
```

MariaDB/MySQL error code:

```
shell> perror 1005 1006
MySQL error code 1005 (ER_CANT_CREATE_TABLE): Can't create table %`s.%`s (errno: %M)
MySQL error code 1006 (ER_CANT_CREATE_DB): Can't create database '%-.192s' (errno: %M)
```

```
shell> perror --silent 1979
You are not owner of query %lu
```

This page is licensed: CC BY-SA / Gnu FDL

replace

Description

The `replace` utility program changes strings in place in files or on the standard input. Invoke replace in one of the following ways:

```
shell> replace from to [from to] ... -- file_name [file_name] ...
shell> replace from to [from to] ... < file_name
```

`from` represents a string to look for, and `to` represents its replacement. There can be one or more pairs of strings.

A from-string can contain these special characters:

Character	Description
^	Match start of line.
$	Match end of line.
\b	Match space-character, start of line or end of line. For an end `\b`, the next replace starts looking at the end space character ( ). A `\b` alone in a string matches only a space character.

Use the `--` option to indicate where the string-replacement list ends and the file names begin. Any file named on the command line is modified in place, so you may want to make a copy of the original before converting it. `replace` prints a message indicating which of the input files it actually modifies.

If the `--` option is not given, `replace` reads standard input and writes to `stdout` (standard output).

`replace` uses a finite state machine to match longer strings first. It can be used to swap strings. For example, the following command swaps “a” and “b” in the given files, *file1 and file2*:

```
shell> replace a b b a -- file1 file2 ...
```

Options

`replace` supports the following options:

Option	Description
-?, -I	Display a help message and exit.
-#debug_options	Enable debugging.
-s	Silent mode. Print less information about what the program does.
-v	Verbose mode. Print more information about what the program does.
-V	Display version information and exit.

This page is licensed: CC BY-SA / Gnu FDL

resolve_stack_dump

`resolve_stack_dump` is a tool that resolves numeric stack strace dumps into symbols.

Usage

```
resolve_stack_dump [OPTIONS] symbols-file [numeric-dump-file]
```

The symbols-file should include the output from: `nm --numeric-sort mariadbd`. The numeric-dump-file should contain a numeric stack trace from mariadbd. If the numeric-dump-file is not given, the stack trace is read from stdin.

Options

Option	Description
-h, –help	Display this help and exit.
-V, –version	Output version information and exit.
-s, –symbols-file=*name*	Use specified symbols file.
-n, –numeric-dump-file=*name*	Read the dump from specified file.

This page is licensed: CC BY-SA / Gnu FDL

Aria Clients and Utilities

See aria_chk.

Check, repair, optimize, and report information about Aria tables with the aria_chk command-line tool.

See aria_pack.

Compress Aria tables into read-only form that is typically 40 to 70 percent smaller with aria_pack.

See aria_read_log.

Display and apply records from the Aria transaction log with the aria_read_log tool.

This page is licensed: CC BY-SA / Gnu FDL

aria_chk

`aria_chk` is used to check, repair, optimize, sort and get information about Aria tables.

With the MariaDB server, you can use CHECK TABLE,REPAIR TABLE and OPTIMIZE TABLE to do similar things.

> **Important:**
>
> Note: `aria_chk` should not be used when MariaDB is running. MariaDB Server assumes that no one is changing the tables it's using.

Usage

```
aria_chk [OPTIONS] aria_tables[.MAI]
```

Aria table information is stored in 2 files: the `.MAI` file contains base table information and the index and the `.MAD` file contains the data.`aria_chk` takes one or more `.MAI` files as arguments.

The following groups are read from the my.cnf files:

- `[maria_chk]`
- `[aria_chk]`

Options and Variables

Global Options

The following options to handle option files may be given as the first argument:

Option	Description
–print-defaults	Print the program argument list and exit.

Option	Description
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.

Main Arguments

Option	Description
-#, –debug=…	Output debug log. Often this is `d:t:o,filename`.
-H, –HELP	Display this help and exit.
-?, –help	Display this help and exit.
–datadir=path	Path for control file (and logs if the `--logdir` option is not used).
–ignore-control-file	
–logdir=path	Path for log files.
–require-control-file	Abort if we can't find/read the `maria_log_control` file.
-s, –silent	Only print errors. You can use two `-s` options to make `aria_chk` very silent.
-t, –tmpdir=path	Path for temporary files. Multiple paths can be specified, separated by colon (`:`) on Unix or semicolon (`;`) on Windows. They are used in a round-robin fashion.
-v, –verbose	Print more information. This can be used with `--description` and `--check`. Use many `-v` options for more verbosity.
-V, –version	Print version and exit.
-w, –wait	Wait if table is locked.

Check Options

`-` `-check` is the default action for `aria_chk`:

Option	Description
-c, –check	Check table for errors.
-e, –extend-check	Check the table *very* thoroughly. Only use this in extreme cases, as `aria_chk` should normally be able to find out if the table is okay, even without this switch.
-F, –fast	Check only tables that haven't been closed properly.
-C, –check-only-changed	Check only tables that have changed since last check.
-f, –force	Restart with `-r` if there are any errors in the table. States is updated as with `--update-state`.
-i, –information	Print statistics information about table that is checked.
-m, –medium-check	Faster than extend-check, and finds 99.99% of all errors. Should be good enough for most cases.
-U, –update-state	Mark tables as crashed if any errors were found and clean if check didn't find any errors but table was marked as 'not clean' before. This allows one to get rid of warnings like 'table not properly closed'. If table was updated, update also the timestamp for when the check was made. This option is on by default! Use –skip-update-state to disable.

Option	Description
-T, –read-only	Don't mark table as checked.

Recover (Repair) Options

When using `--recover` or `--safe-recover'` , these options are available:

Option	Description
-B, –backup	Make a backup of the `.MAD` file as *`filename`* `-time.BAK'` .
–correct-checksum	Correct checksum information for table.
-D, –data-file-length=#	Max length of data file (when recreating data file when it's full).
-e, –extend-check	Try to recover every possible row from the data file Normally this will also find a lot of garbage rows; Don't use this option if you are not totally desperate.
-f, –force	Overwrite old temporary files.
-k, –keys-used=#	Tell MariaDB to update only some specific keys. # is a bit mask of which keys to use. This can be used to get faster inserts.
–max-record-length=#	Skip rows bigger than this if `aria_chk` can't allocate memory to hold it.
-r, –recover	Can fix almost anything except unique keys that aren't unique.
-n, –sort-recover	Forces recovering with sorting even if the temporary file would be very big.
-p, –parallel-recover	Uses the same technique as `-r` and `-n` , but creates all the keys in parallel, in different threads.
-o, –safe-recover	Uses an old recovery method, which is slower than using `-r` but can handle a couple of cases where `-r` reports that it can't fix the data file.
–transaction-log	Log repair command to transaction log. This is needed if one wants to use the `maria_read_log` option to repeat the repair.
–character-sets-dir=…	Directory where character sets are.
–set-collation=name	Change the collation used by the index.
-q, –quick	
-u, –unpack	Unpack file packed with aria_pack.

Other Options

Option	Description
-a, –analyze	Analyze distribution of keys. Will make some joins in MariaDB faster. You can check the calculated distribution by using `--description --verbose`` *`table_name`* '.
–stats_method=name	Specifies how index statistics collection code should treat NULLs. Possible values of name are "nulls_unequal" (default for 4.1/5.0), "nulls_equal" (emulate 4.0), and "nulls_ignored".
-d, –description	Prints some information about table.
-A, –set-auto-increment[=value]	Force auto_increment to start at this or higher value If no value is given, then sets the next auto_increment value to the highest used value for the auto key + 1.

Option	Description
-S, –sort-index	Sort index blocks. This speeds up ‘read-next’ in applications.
-R, –sort-records=#	Sort records according to an index. This makes your data much more localized and may speed up things (It may be VERY slow to do a sort the first time!).
-b, –block-search=#	Find a record, a block at given offset belongs to.
-z, –zerofill	Remove transaction id’s from the data and index files and fills empty space in the data and index files with zeroes. Zerofilling makes it possible to move the table from one system to another without the server having to do an automatic zerofill. It also allows one to compress the tables better if one want to archive them.
–zerofill-keep-lsn	Like –zerofill but does not zero out LSN of data/index pages.

Variables

Option	Description
page_buffer_size	Size of page buffer. Used by `--safe-repair`.
read_buffer_size	Read buffer size for sequential reads during scanning.
write_buffer_size	Write buffer size for sequential writes during repair of fixed size or dynamic size rows.
sort_buffer_size	Size of sort buffer. Used by `--recover`.
sort_key_blocks	Internal buffer for sorting keys. Do not use!

Examples

The main usage of `aria_chk` is a fast check of all Aria tables in your system. This is faster than doing it in MariaDB, as you can allocate all free memory to the buffers.

Assuming you have a bit more than 2G free memory, the following commands, run in the MariaDB data directory, check all your tables and repairs only those that have an error:

```
aria_chk --check --sort_order --force --sort_buffer_size=1G */*.MAI
```

To optimize all your tables, run this command (the `--zerofill` option is used here to fill up empty space with `\0` which can speed up compressed backups):

```
aria_chk --analyze --sort-index --page_buffer_size=1G --zerofill */*.MAI
```

In case you have a serious problem and have to use `--safe-recover`, run this command:

```
aria_chk --safe-recover --zerofill --page_buffer_size=2G */*.MAI
```

This page is licensed: CC BY-SA / Gnu FDL

aria_pack

`aria_pack` is a tool for compressing Aria tables. The resulting tables are read-only, and usually about 40% to 70% smaller.

`aria_pack` is run as follows:

```
aria_pack [options] file_name [file_name2...]
```

The file name is the `.MAI` index file. The extension can be omitted, although keeping it permits wildcards, for instance to compress all the files, such as:

```
aria_pack *.MAI
```

`aria_pack` compresses each column separately. This means, when the resulting data is read, only the individual rows and columns required need to be decompressed, allowing for quicker reading.

Once a table has been packed, use aria_chk -rq (the quick and recover options) to rebuild its indexes.

Options

The following variables can be set as command line options to `aria_pack`, or set in the `[ariapack]` section in your my.cnf file:

Option	Description
-b, –backup	Make a backup of the table as *`table_name`* `.OLD`.
–character-sets-dir=*name*	Directory where character sets are located.
-h, –datadir	
-#, –debug[=name]	Output debug log. Often this is `d:t:o,filename`.
-?, –help	Display help and exit.
-f, –force	Force packing of table even if it gets bigger or if a temporary file exists.
–ignore-control-file	
-j, –join=*name*	Join all given tables into *name*. All tables must have identical layouts.
–require-control-file	
-s, –silent	Only write output when an error occurs.
-t, –test	Don’t pack table, only test packing it.
-T, –tmpdir=name	Use temporary directory to store temporary table.
-v, –verbose	Write info about progress and packing result. Use multiple `-v` options for more verbosity.
-V, –version	Output version information and exit.
-w, –wait	Wait and retry if table is in use.

Unpacking

To unpack a table compressed with aria_pack, use the aria_chk -u option.

Example

```
> aria_pack /my/data/test/posts
Compressing /my/data/test/posts.MAD: (1690 records)
- Calculating statistics
- Compressing file
37.71%
> aria_chk -rq --ignore-control-file /my/data/test/posts
- check record delete-chain
- recovering (with keycache) Aria-table '/my/data/test/posts'
Data records: 1690
State updated
```

See Also

- FLUSH TABLES FOR EXPORT

- myisamchk

This page is licensed: CC BY-SA / Gnu FDL

aria_read_log

`aria_read_log` is a tool for displaying and applying log records from an Aria transaction log.

> **Warning:**
>
> Aria is compiled without `-DIDENTICAL_PAGES_AFTER_RECOVERY` which means that the table files are not byte-to-byte identical to files created during normal execution. This should be okay, except for test scripts that try to compare files before and after recovery.

Usage

```
aria_read_log OPTIONS
```

As an option, you need to use at least one of `-d` or `-a`.

Options

The following variables can be set while passed as command line options to `aria_read_log`, or set in the `[aria_read_log]` section in your my.cnf file.

Option	Description
-a, –apply	
–character-sets-dir=*name*	Directory where character sets are.
-c, –check	Used with the `--display-only` option, check if record is fully readable (for debugging).
-?, –help	Display help and exit.
-d, –display-only	Display brief info read from records' header.
-e, –end-lsn=#	Stop applying at this lsn. If end-lsn is used, UNDO operations are not be applied.
-h, –aria-log-dir-path=*name*	Path to the directory where to store transactional log.
-P, –page-buffer-size=#	The size of the buffer used for index blocks of Aria tables.
-l, –print-log-control-file	Print the content of the `aria_log_control_file`.
-o, –start-from-lsn=#	Start reading log from this lsn.
-C, –start-from-checkpoint	Start applying from last checkpoint.
-s, –silent	Print less information during apply/undo phase.
-T, –tables-to-redo=*name*	List of comma-separated tables that we should apply REDO on. Use this if you only want to recover some tables.
-t, –tmpdir=name	Path for temporary files. Multiple paths can be specified, separated by colon (`:`).
–translog-buffer-size=#	The size of the buffer used for transaction log of Aria tables.
-u, –undo	Apply UNDO records to tables. Disable with `--disable-undo`. Defaults to on; use `--skip-undo` to disable.
-v, –verbose	Print more information during apply/undo phase.
-V, –version	Print version and exit.

This page is licensed: CC BY-SA / Gnu FDL

Backup, Restore and Import Clients

See mariadb-dump.

Complete MariaDB backup and recovery guide. Complete resource for backup methods, mariabackup usage, scheduling, and restoration for production use.

See mariadb-hotcopy.

Quickly back up MyISAM and Aria tables by copying table files with the mariadb-hotcopy Perl script (formerly mysqlhotcopy).

See mariadb-import.

Complete mariadb-import reference: load tables from text files, –fields-terminated-by delimiter, –local file option, and –parallel/-j processing.

See dbForge Studio.

dbForge Studio is a proprietary third-party tool, not included with MariaDB, for backup, restore, and data export and import.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-dump

The `mariadb-dump` client is a backup program originally written by Igor Romanenko.

> **Note:**
>
> Previously, the client used to be called `mysqldump`, and can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows. From MariaDB 11.0, the symlink `mysqldump` is deprecated and removed from the `mariadb` Docker Official Image.

`mariadb-dump` generates a command at the beginning of the dump to enable sandbox mode. This command cannot be interpreted by earlier versions of the `mariadb` command-line client or by the MySQL command-line client `mysql` – an error is thrown if used against client versions that do not support it. This does not affect other methods of importing the data.

`mariadb-dump` is used to dump a database or a collection of databases for backup, or transferring data to another database server (not necessarily MariaDB or MySQL). The dump contains SQL statements to create databases, tables, table data, and more. It can also be used to generate files in CSV, XML, or other formats that use delimiters.

> **Note:**
>
> If you are doing a backup on the server and all your tables are MyISAM tables, consider using mariadb-hotcopy instead, because it can accomplish faster backups and faster restores.

`mariadb-dump` dumps trigger along with tables, as these are part of the table definition. However, stored procedures, views, and events are not dumped, and need extra parameters to be recreated explicitly (for example, `--routines` and `--events`). Procedures and functions are also part of the system tables (for example, mysql.proc).

`mariadb-dump` supports the enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT.

Performance

`mariadb-dump` doesn't consume much CPU – by default, it uses a single thread. This method is good for a heavily loaded server.

Disk input/output per second (IOPS) can, however, increase for multiple reasons. When backing up on the same device as the database, this produces unnecessary random IOPS. The dump is done sequentially, on a per-table basis, causing a full table scan and many buffer page misses on tables that are not fully cached in memory.

> **Warning:**
>
> It's recommended that you back up from a network location to remove disk IOPS on the database server, but it is vital to use a separate network card to keep network bandwidth available for regular traffic.

Although `mariadb-dump` by default preserves your resources for regular spindle disks and low-core hardware, this doesn't mean that concurrent dumps cannot benefit from hardware architecture like SAN, flash storage, or low write workload. The backup time would benefit from a tool such as MyDumper.

Usage

There are four general ways to invoke `mariadb-dump`:

```
shell> mariadb-dump [options] db_name [tbl_name ...]
shell> mariadb-dump [options] --databases db_name ...
shell> mariadb-dump [options] --all-databases
shell> mariadb-dump [options] --system=[option_list]
```

If you do not name any tables after specifying `db_name`, or if you use the `--databases` or `--all-databases` option, entire databases are dumped.

`mariadb-dump` does not dump the `INFORMATION_SCHEMA` (or `PERFORMANCE_SCHEMA`, if enabled) database by default. MariaDB dumps the `INFORMATION_SCHEMA` if you name it explicitly on the command line, although you must also use the `--skip-lock-tables` option.

To see a list of the options your version of `mariadb-dump` supports, execute `mariadb-dump --help`.

Row by Row vs. Buffering

`mariadb-dump` can retrieve and dump table contents row by row, or it can retrieve the entire content from a table and buffer it in memory before dumping it. Buffering in memory can be a problem if you are dumping large tables. To dump tables row by row, use the `--quick` option (or `--opt`, which enables `--quick`). The `--opt` option (and hence `--quick`) is enabled by default, so to enable memory buffering, use `--skip-quick`.

mysql.transaction_registry_table

`mariadb-dump` includes logic to cater for the mysql.transaction_registry table.

Generated Columns

From MariaDB 13.1, `mariadb-dump` no longer writes the computed values of generated columns into the `INSERT` statements it produces. The values are redundant, because the server recalculates them when the dump is reloaded.

Given this table:

```
CREATE TABLE t1 (pk INTEGER, a INTEGER, b INTEGER, c VARCHAR(16),
                 sum INTEGER GENERATED ALWAYS AS (a+b) VIRTUAL,
                 sub VARCHAR(4) GENERATED ALWAYS AS (SUBSTRING(c, 1, 4)) VIRTUAL);
```

by default a generated column keeps its position in the row, and its value is written as `DEFAULT`:

```
INSERT INTO `t1` VALUES
(1,11,12,'oneone',DEFAULT,DEFAULT),
(2,21,22,'twotwo',DEFAULT,DEFAULT);
```

With `--complete-insert`, generated columns are left out of both the column list and the values:

```
INSERT INTO `t1` (`pk`, `a`, `b`, `c`) VALUES (1,11,12,'oneone'),
(2,21,22,'twotwo');
```

An INVISIBLE column enables `--complete-insert` for that table, so the dump gets a column list even when the option is not given. Generated columns are then only left out from the first `INVISIBLE` column onward: a generated column declared *before* it stays in the column list, with its value written as `DEFAULT`. Given this table:

```
CREATE TABLE t4 (pk INTEGER, a INTEGER,
                 b INTEGER GENERATED ALWAYS AS (a+1),
                 c INTEGER INVISIBLE,
                 d INTEGER GENERATED ALWAYS AS (a+pk));
```

`b` precedes the `INVISIBLE` column and is kept, while `d` follows it and is omitted:

```
INSERT INTO `t4` (`pk`, `a`, `b`, `c`) VALUES (1,2,DEFAULT,10),
(4,5,DEFAULT,20);
```

Passing `--complete-insert` explicitly leaves out every generated column, whatever its position.

If every column in a table is generated, `--complete-insert` writes an empty column list, which preserves the row count:

```
INSERT INTO `t5` () VALUES (),
(),
();
```

> **Note:**
>
> This applies to the SQL output only. The `--xml` output is unchanged, and still contains generated columns with their computed values. `--tab` and `--dir` are also unaffected, because the server writes those data files with `SELECT ... INTO OUTFILE`. System-versioned `row_start` and `row_end` columns are not treated as generated columns.

Old Versions of MySQL

If you are using a recent version of `mariadb-dump` to generate a dump to be reloaded into a very old MySQL server, you should *not* use the `--opt` or `--extended-insert` option. Use `--skip-opt` instead.

Options

`mariadb-dump` supports the following options:

Scope

–all

Deprecated. Use –create-options instead.

-A, –all-databases

Dump all the databases. This is the same as `--databases` with all databases selected.

-Y, –all-tablespaces

Dump all the tablespaces.

-y, –no-tablespaces

Do not dump any tablespace information.

Other Options

–add-drop-database

Add a DROP DATABASE before each create. Typically used in conjunction with the `--all-databases` or `--databases` option, because no CREATE DATABASE statements are written, unless one of those options is specified.

–add-drop-table

Add a DROP TABLE before each create.

–add-drop-trigger

Add a DROP TRIGGER statement before each CREATE TRIGGER.

–add-locks

Add locks around INSERT statements, which results in faster inserts when the dump file is reloaded. Use –skip-add-locks to disable.

–allow-keywords

Allow creation of column names that are keywords. This works by prefixing each column name with the table name.

–apply-slave-statements

Adds STOP SLAVE prior to CHANGE MASTER and START SLAVE to the bottom of the dump.

–as-of=name

As of the given timestamp, dump the system versioned table(s). Note that table structures are always dumped regardless of the date/time specified; only the data is filtered by the timestamp. The `--tz-utc` setting is used to understand the timestamp argument; for important interaction details, see –tz-utc. This option is available from MariaDB 10.7.

If no data is available for the specified timestamp (for example, a date before any records were entered), only the table structure (CREATE, TABLE statement) will be dumped.

For example:

```
CREATE TABLE t1 (a INT) WITH SYSTEM VERSIONING;
INSERT t1 (a) VALUES (1),(2),(3);
DELETE FROM t1 WHERE a=1;
```

Dump result:

```
./bin/mysqldump -uroot -S./socket.sock --compact --as-of="1990-01-01 00:01" test t1
```

This dumps the table structure but not the data (since no records existed at that time). See MDEV-26943 for more information regarding the interaction with `--tz-utc`.

Note: When `--tz-utc` is enabled (which is the default), the timestamp value supplied to `--as-of` is interpreted in UTC. Remember that the effective filter time can differ if you are passing a local time. See –tz-utc and –skip-tz-utc.

–character-sets-dir=name

Directory for character set files.

-i, –comments

Write additional information in the dump file, such as program version, server version, and host. Disable with `--skip-comments`.

–compact

Give less verbose output (useful for debugging). Disables structure comments and header/footer constructs. Enables the `--skip-add-drop-table`, `--skip-add-locks`, `--skip-comments`, `--skip-disable-keys`, and `--skip-set-charset` options.

–compatible=name

Change the dump to be compatible with a given mode. By default, tables are dumped in a format optimized for MariaDB and MySQL. Legal modes are: ansi, `mysql323`, `mysql40`, `postgresql`, `oracle`, `mssql`, `db2`, `maxdb`, `no_key_options`, `no_table_options`, and `no_field_options`. You can use several modes, separated by commas. This option does not guarantee compatibility with other servers. It only enables those SQL mode values that are available for making dump output more compatible. For example, `--compatible=oracle` does not map data types to Oracle types or use Oracle comment syntax.

-c, –complete-insert

Use complete INSERT statements that include column names. From MariaDB 13.1, generated columns are omitted from the column list.

-C, –compress

Use compression in the server/client protocol. Both client and server must support compression for this to work.

–copy-s3-tables

By default, S3 tables are ignored. With this option set, the result file will contain a `CREATE` statement for a similar Aria table, followed by the table data and ending with an `ALTER TABLE`` `` *`table`* `` ``ENGINE=S3`.

-a, –create-options

Include all MariaDB and/or MySQL-specific create options in `CREATE TABLE` statements. Use `--skip-create-options` to disable.

-B, –databases

Dump several databases. Normally, `mariadb-dump` treats the first name argument on the command line as a database name and the following names as table names. With this option, it treats all name arguments as database names. CREATE DATABASE and USE statements are included in the output before each new database.

-#, –debug[=#]

If using a debug version of MariaDB, write a debugging log. A typical `debug_options` string is `d:t:o,file_name`. The default value is `d:t:o,/tmp/mysqldump.trace`. If using a non-debug version, `mariadb-dump` will catch this and exit.

–debug-check

Check memory and open file usage at exit.

–debug-info

Print some debug info at exit.

–default-auth=*name*

Default authentication client-side plugin to use.

–default-character-set=*name*

Set the default character set to *name*. If no character set is specified, `mariadb-dump` uses utf8mb4.

–defaults-extra-file=*name*

Read the file *name* after the global files are read. Must be given as the first argument.

–defaults-file=*name*

Only read default options from the given file *name*. Must be given as the first argument.

–defaults-group-suffix=*str*

Also read groups with a suffix of *str*. For example, since `mariadb-dump` normally reads the `[client]` and `[mariadb-dump]` (or `[mysqldump]`) groups, `--defaults-group-suffix=` *`x`* would cause it to also read the groups `[mariadb-dump_` *`x`* `]` (or `[mysqldump_` *`x`* `]`) and `[client_` *`x`* `]`.

–delayed-insert

Insert rows with INSERT DELAYED instead of INSERT.

–delete-master-logs

On a primary replication server, delete the binary logs by sending a PURGE BINARY LOGS statement to the server after performing the dump operation. This option automatically enables `--master-data=2`.

–dir

Parallel dump of multiple databases. Works just like `--tab`, with regard to output (sql file for table definition and tab-separated for data, same options, for example, `--parallel`). It also allows the `--databases` and `--all-databases` options. When `--dir` is used, it creates the directory structure in the output directory pointed to by `--dir`. For every database to be dumped, there is a directory with the database name. All options that `--tab` supports are also supported by `--dir`, in particular `--parallel`. This option is available from MariaDB 11.5.

-K, –disable-keys

'`/*!40000 ALTER TABLE`` ``` *`tb_name`* ``DISABLE KEYS`` */;` *and* `'/ !40000 ALTER TABLE`` ``` *`tb_name`* ``` ``ENABLE KEYS */;` are written to the output. This makes loading the dump file faster because the indexes are created after all rows are inserted. This option is effective only for non-unique indexes of MyISAM tables. Disable with `--skip-disable-keys`.

–dump-date

If the `--comments` option and this option are given, `mariadb-dump` produces a comment at the end of the dump of the following form: `-- Dump completed on`` ``` *`date`*. However, the date causes dump files taken at different times to appear to be different, even if the data are otherwise identical. `--dump-date` and `--skip-dump-date`

control whether the date is added to the comment. The default is `--dump-date` (include the date in the comment). `--skip-dump-date` suppresses date printing.

-H, –dump-history

Dump tables with history. This option is available from MariaDB 10.11. Until this option was introduced, `mariadb-dump` could not read historical rows from versioned tables, and so historical data would not be backed up. Use also `--update-history` if upgrading to MariaDB 11.5 or newer from a version before 11.5.

–dump-slave[=value]

Used for producing a dump file from a replica server that can be used to set up another replica server with the same primary. Causes the binary log position and filename of the primary to be appended to the dumped data output. Setting the value to `1` (the default) prints it as a CHANGE MASTER command in the dumped data output; if set to `2`, that command is prefixed with a comment symbol. This option will turn on `--lock-all-tables`, unless `--single-transaction` is specified, too (in which case a global read lock is only taken a short time at the beginning of the dump. Make sure to read about `--single-transaction` below). In all cases, any action on logs happens at the exact moment of the dump. This option automatically turns off `--lock-tables`.

> **CURRENT**
>
> This option pauses any running SQL threads during the dump.
>
> **< 10.11**
>
> This option stops any running SQL threads before the dump, and restarts **all stopped IO and SQL** threads after completion.

-E, –events

Include Event Scheduler events for the dumped databases in the output.

-e, –extended-insert

Use multiple-row INSERT syntax that includes several value lists. This results in a smaller dump file and speeds up inserts when the file is reloaded. Defaults to `ON`; use `--skip-extended-insert` to disable.

–fields-terminated-by=*string*

Fields in the output file are terminated by the given *string*. Used with the `--tab` option and has the same meaning as the corresponding `FIELDS` clause for LOAD DATA INFILE.

–fields-enclosed-by=*character*

Fields in the output file are enclosed by the given *character*. Used with the `--tab` option and has the same meaning as the corresponding `FIELDS` clause for LOAD DATA INFILE.

–fields-optionally-enclosed-by=*character*

Fields in the output file are optionally enclosed by the given *character*. Used with the `--tab` option and has the same meaning as the corresponding `FIELDS` clause for LOAD DATA INFILE.

–fields-escaped-by=*character*

Fields in the output file are escaped by the given *character*. Used with the –tab option and has the same meaning as the corresponding `FIELDS` clause for LOAD DATA INFILE.

–first-slave

Removed in MariaDB 5.5. Use `--lock-all-tables` instead.

-F, –flush-logs

Flush the MariaDB server log files before starting the dump. This option requires the RELOAD privilege. If you use this option in combination with the `--databases=` or `--all-databases` option, the logs are flushed for each database dumped. The exception is when using `--lock-all-tables` or `--master-data`: In this case, the logs are flushed only once, corresponding to the moment all tables are locked. If you want your dump and the log flush to happen at the same exact moment, you should use `--flush-logs` together with either `--lock-all-tables` or `--master-data`.

–flush-privileges

Send a FLUSH PRIVILEGES statement to the server after dumping the mysql database. This option should be used any time the dump contains the mysql database and any other database that depends on the data in the mysql database for proper restoration.

-f, –force

Continue even if an SQL error occurs during a table dump. One use for this option is to cause mariadb-dump to continue executing even when it encounters a view that has become invalid because the definition refers to a table that has been dropped. Without `--force` in this example, mariadb-dump exits with an error message. With `--force`, mariadb-dump prints the error message, but it also writes an SQL comment containing the view definition to the dump output and continues executing.

–gtid

Used together with `--master-data` and `--dump-slave` to more conveniently set up a new GTID replica. It causes those options to output SQL statements that configure the replica to use the global transaction ID to connect to the primary instead of old-style filename/offset positions. The old-style positions are still included in comments when –gtid is used; likewise, the GTID position is included in comments even if `--gtid` is not used.

–header

When used together with `--tab`, enabling this option adds a header row containing column names at the top of the output `.txt` files.

-?, –help

Display a help message and exit.

–hex-blob

Dump binary strings in hexadecimal format (for example, ´abc´ becomes 0x616263). The affected data types are BINARY, VARBINARY, the BLOB types, and BIT.

-h name, –host=*host*

Connect to and dump data from the MariaDB or MySQL server on the given *host*. The default host is `localhost`.

–ignore-database=*database*

Do not dump the specified *database*. To specify more than one database to ignore, use the directive multiple times, once for each database. Only takes effect when used together with `--all-databases` or `-A`.

–ignore-table=*table*

Do not dump the specified *table*. To specify more than one table to ignore, use the directive multiple times, once for each table. Each table must be specified with both database and table names, for example, `--ignore-table=` *`database.table`*. This option can also be used to ignore views.

–ignore-table-data=*table*

Do not dump the specified *table* data (only the structure). To specify more than one table to ignore, use the directive multiple times, once for each table. Each table must be specified with both the database and table names. See also `--no-data`.

–include-master-host-port

Add the `MASTER_HOST` and `MASTER_PORT` options for the CHANGE MASTER TO statement when using the `--dump-slave` option for a replica dump.

–insert-ignore

Insert rows with `INSERT IGNORE` instead of `INSERT`.

–lines-terminated-by=*string*

Lines in the output file are terminated by the given *string*. This option is used with the `--tab` option and has the same meaning as the corresponding `LINES` clause for LOAD DATA INFILE.

-x, –lock-all-tables

Lock all tables across all databases. This is achieved by acquiring a global read lock for the duration of the whole dump by executing FLUSH TABLES WITH READ LOCK. This option automatically turns off `--single-transaction` and `--lock-tables`.

-l, –lock-tables

For each dumped database, lock all tables to be dumped before dumping them. The tables are locked with `READ LOCAL` to allow concurrent inserts in the case of MyISAM tables. For transactional tables such as InnoDB, `--single-transaction` is a much better option than `--lock-tables` because it does not need to lock the tables at all. Because `--lock-tables` locks tables for each database separately, this option does not guarantee that the tables in the dump file are logically consistent between databases. Tables in different databases may be dumped in completely different states. Use `--skip-lock-tables` to disable.

-L, –wildcards

> **Note:**
>
> This option is available from MariaDB 12.1.

Usage of wildcards in the table or database name. Without the –databases option, wildcards are only recognized in table names. With the “databases” option - in databases names.

–log-error=*file*

Log warnings and errors by appending them to the named *file*. The default is no logging.

–log-queries

When restoring the dump, if logging is turned on, the server logs queries to the general and slow query log. Defaults to `ON`; use `--skip-log-queries` to disable.

–master-data[=#]

Causes the binary log position and filename to be appended to the output, useful for dumping a primary replication server to produce a dump file that can be used to set up another server as a replica of the primary. These are the primary server coordinates from which the replica should start replicating after you load the dump file into the replica.

If the option is set to `1` (the default), print it as a CHANGE MASTER command; if set to `2`, that command is prefixed with a comment symbol. This `--master-data` option turns `--lock-all-tables` on, unless `--single-transaction` is specified, too. In all cases, any action on logs will happen at the exact moment of the dump. This option automatically turns `--lock-tables` off. In all cases, any action on logs happens at the exact moment of the dump. It is also possible to set up a replica by dumping an existing replica of the primary. To do this, use the following procedure on the existing replica:

Step 1

Stop the SQL thread of the replica and get its current status.

```
STOP SLAVE SQL_THREAD;
SHOW REPLICA STATUS;
```

Step 2

From the output of the `SHOW REPLICA STATUS` statement, the binary log coordinates of the primary server from which the new replica should start replicating are the values of the `Relay_Master_Log_File` and `Exec_Master_Log_Pos` fields. Denote those values as `file_name` and `file_pos`.

Step 3

Dump the replica server.

```
mariadb-dump --master-data=2 --all-databases > dumpfile
```

Step 4

Restart the replica.

```
START REPLICA;
```

Step 5

On the new replica, load the dump file.

```
mariadb < dumpfile
```

Step 6

On the new replica, set the replication coordinates to those of the primary server obtained earlier.

```
CHANGE MASTER TO
       MASTER_LOG_FILE = ´file_name´,
       MASTER_LOG_POS = file_pos;
```

The `CHANGE MASTER TO` statement might also need other parameters, such as `MASTER_HOST` to point the replica to the correct primary server host. Add any such parameters as necessary.

–max-allowed-packet=#

The maximum packet length to send to or receive from the server. The maximum is 1GB.

–max-statement-time=#

Sets the maximum time any statement can run before being timed out by the server. Default value is `0` (no limit).

–net-buffer-length=#

The initial buffer size for client/server TCP/IP and socket communication. This can be used to limit the size of rows in the dump. When creating multiple-row `INSERT` statements (as with the `--extended-insert` or `--opt` option), `mariadb-dump` creates rows up to `net_buffer_length` length.

–no-autocommit

Enclose the INSERT statements for each dumped table within SET autocommit = 0 and COMMIT statements. `ON` by default from MariaDB 11.8 to allow faster data loading by InnoDB, writing only one undo log for the whole operation.

-n, –no-create-db

This option suppresses the CREATE DATABASE … IF EXISTS statement that normally is output for each dumped database if `--all-databases` or `--databases` is given.

-t, –no-create-info

Do not write CREATE TABLE statements, which re-create each dumped table.

-d, –no-data

Do not write any table row information (that is, do not dump table contents). This is useful if you want to dump only the CREATE TABLE statement for the table (for example, to create an empty copy of the table by loading the dump file). See also `--ignore-table-data`.

–no-data-med

Do not dump rows for engines that manage external data (for instance, MRG_MyISAM, MRG_ISAM, CONNECT, OQGRAPH, Spider, VP, Federated). This option is enabled by default. If you want to dump data for these engines, you need to set `--no-data-med=0`.

–no-defaults

Don't read default options from any option file. Must be given as the first argument.

-N, –no-set-names

Suppress the `SET NAMES` statement. This has the same effect as `--skip-set-charset`.

–opt

This option is shorthand. It is the same as specifying `--add-drop-table`, `--add-locks`, `--create-options`, `--quick`, `--extended-insert`, `--lock-tables`, `--set-charset`, and `--disable-keys`. Enabled by default, disable with `--skip-opt`. It should give you a fast dump operation and produce a dump file that can be reloaded into a MariaDB server quickly. The `--opt` option is enabled by default. Use `--skip-opt` to disable it. See the discussion at the beginning of this section for information about selectively enabling or disabling a subset of the options affected by `--opt`.

–order-by-primary

Sorts each table's rows by primary key, or first unique key, if such a key exists. This is useful when dumping a MyISAM table to be loaded into an InnoDB table, but it will make the dump itself take considerably longer.

–order-by-size

Dump each table according to its size, smallest first. Useful when using `--single-transaction` on tables which get truncated/altered often. The assumption here is that smaller tables get truncated more often, and by dumping those first, this reduces the chance that a `--single-transaction` dump will fail with 'Table definition has changed, please retry transaction'. This option is available from MariaDB 10.9.1.

-j, –parallel=#

Number of dump table jobs executed in parallel (only for use with the `--tab` option). Testing indicates that performance can be increased (dump time decreased) up to 4 times on smaller-sized dumps, when the database fits into memory. There is a point at which the disk becomes the bottleneck, after which adding more parallel jobs does not bring better performance. From MariaDB 11.4.1.

-p[*password*], –password=*password*

The *password* to use when connecting to the server. If you use the short option form (`-p`), you cannot have a space between the option and the password. If you omit the password value following the `--password` or `-p` option on the command line, `mariadb-dump` prompts for one. Specifying a password on the command line should be considered insecure. You can use an option file to avoid giving the password on the command line.

-W, –pipe

On Windows, connect to the server via a named pipe. This option applies only if the server supports named-pipe connections.

–plugin-dir

Directory for client-side plugins.

-P *port*, –port=*port*

The TCP/IP *port number* to use for the connection.

–print-defaults

Print the program argument list and exit. Must be given as the first argument.

–protocol=name

The connection protocol to use for connecting to the server (`TCP`, `SOCKET`, `PIPE`, `MEMORY`). It is useful when the other connection parameters normally would cause a protocol to be used other than the one you want.

-q, –quick

This option is useful for dumping large tables. It forces `mariadb-dump` to retrieve rows for a table from the server, a row at a time, and to then dump the results directly to stdout rather than retrieving the entire row set and buffering it in memory before writing it out. Defaults to `ON`, use `--skip-quick` to disable.

-Q, –quote-names

Quote identifiers (such as database, table, and column names) within backtick (`` ` ``) characters. If the `ANSI_QUOTES` SQL mode is enabled, identifiers are quoted with (`"`) characters. This option is enabled by default. It can be disabled with `--skip-quote-names`, but this option should be given after any option such as `--compatible` that may enable `--quote-names`.

–replace

Use REPLACE INTO statements instead of INSERT INTO statements.

-r, –result-file=*file*

Direct output to a given *file*. This option should be used on Windows to prevent newline (`\n`) characters from being converted to `\r\n` carriage return/newline sequences. The result file is created and its previous contents overwritten, even if an error occurs while generating the dump.

-R, –routines

Include stored routines (procedures and functions) for the dumped databases in the output. Use of this option requires the SELECT privilege for the mysql.proc table. The output generated using –routines contains CREATE PROCEDURE and CREATE FUNCTION statements to re-create the routines. However, these statements do not include attributes such as the routine creation and modification timestamps. This means that when the routines are reloaded, they are created with the timestamps equal to the reload time. If you require routines to be re-created with their original timestamp attributes, do not use `--routines`. Instead, dump and reload the contents of the mysql.proc table directly, using a MariaDB account, which has appropriate privileges for the mysql database.

–set-charset

Add `'SET NAMES default_character_set'` to the output in order to set the character set. Enabled by default; suppress with `--skip-set-charset`.

-O, –set-variable=*variable*

Change the value of a *variable*. This option is deprecated; you can set variables directly with `-- variable-name = value`.

–shared-memory-base-name

Shared-memory name to use for Windows connections using shared memory to a local server (started with the `--shared-memory` option). Case-sensitive. Defaults to `MYSQL`.

–single-transaction

This option sends a START TRANSACTION SQL statement to the server before dumping data. It is useful only with transactional tables such as InnoDB, because then it dumps the consistent state of the database at the time when `BEGIN` was issued, without blocking any applications. When using this option, you should keep in mind that only InnoDB tables are dumped in a consistent state. The single-transaction feature depends not only on the engine being transactional and capable of `REPEATABLE-READ`, but also on `START TRANSACTION WITH CONSISTENT SNAPSHOT`. The dump is not guaranteed to be consistent for other storage engines. For example, any MyISAM, or MEMORY tables dumped while using this option may still change state. While a `--single-transaction` dump is in process, to ensure a valid dump file (correct table contents and binary log coordinates), no other connection should use the following statements: ALTER TABLE, CREATE TABLE, DROP TABLE, RENAME TABLE, or TRUNCATE TABLE. A consistent read is not isolated from those statements, so use of them on a table to be dumped can cause the `SELECT` (performed by mariadb-dump to retrieve the table contents) to obtain incorrect contents or fail. The `--single-transaction` option and the `--lock-tables` option are mutually exclusive, because LOCK TABLES causes any pending transactions to be committed implicitly. So, this option automatically turns off `--lock-tables`. To dump large tables, you should combine the `--single-transaction` option with `--quick`.

–skip-add-locks

Disable the `--add-locks` option.

–skip-comments

Disable the `--comments` option.

–skip-disable-keys

Disable the `--disable-keys` option.

–skip-extended-insert

Disable the `--extended-insert` option.

–skip-opt

Disable the `--opt` option (disables `--add-drop-table`, `--add-locks`, `--create-options`, `--quick`, `--extended-insert`, `--lock-tables`, `--set-charset`, and `--disable-keys`).

–skip-quick

Disable the `--quick` option.

–skip-quote-name

Disable the `--quote-names` option.

–skip-set-charset

Disable the `--set-charset` option.

–skip-triggers

Disable the `--triggers` option.

–skip-tz-utc

Disables the `--tz-utc` option. When `--skip-tz-utc` is specified, the `--tz-utc` behavior is disabled. `mariadb-dump` will not set its connection time zone to UTC before the dump, and will not add `SET TIME_ZONE='+00:00'` to the top of the dump file. TIMESTAMP values will therefore be dumped in the local time zone of the server. When the dump is reloaded on a server in a different time zone or during daylight saving time transitions, remember that this may result in different timestamp values. See –tz-utc.

-S name, –socket={*socket|named-pipe*}

For connections to localhost, the Unix *socket file* to use, or, on Windows, the name of the *named pipe* to use.

–ssl

Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The `--ssl` option does not enable verifying the server certificate by default. In order to verify the server certificate, the user must specify the `--ssl-verify-server-cert` option.

–ssl-ca=*pem-file*

Defines a path to a *PEM file* that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option implies the `--ssl` option.

–ssl-capath=*pem-directory*

Defines a *path to a directory that contains one or more PEM files* that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the `--ssl` option.

–ssl-cert=*certificate-file*

Defines a path to the *X509 certificate file* to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

–ssl-cipher=*cipher-list*

List of permitted ciphers or cipher suites to use for TLS. This option implies the `--ssl` option.

–ssl-crl=*pem-file*

Defines a *path to a PEM file* that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL or Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

–ssl-crlpath=*pem-directory*

Defines a *path to a directory that contains one or more PEM files* that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

–ssl-fp=name

Server certificate fingerprint (implies `--ssl`).

–ssl-fplist=name

File with accepted server certificate fingerprints, one per line (implies `--ssl`).

–ssl-key=*private-key*

Defines a *path to a private key file* to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

–ssl-verify-server-cert

Enables server certificate verification. This option is disabled by default.

–system=*option*[,*option*]]

Dump the database's system tables in a logical form. With this option, the mysql database tables are dumped as CREATE USER, CREATE SERVER, and other forms of logical portable SQL statements. The option values here are from the set of `all`, `users`, `plugins`, `udfs`, `servers`, `stats`, and `timezones`.

-T, –tab=*name*

Produce tab-separated text-format data files. With this option, for each dumped table, `mariadb-dump` creates a `tbl_name.sql` file containing the `CREATE TABLE` statement that creates the table, and a `tbl_name.txt` file containing the table's data. The option value is the directory in which to write the files. Note: This option can only be used when `mariadb-dump` is run on the same machine as the `mariadbd` server. You must have the `FILE` privilege, and the server must have permission to write files in the directory that you specify. By default, the `.txt` data files are formatted using tab characters between column values and a newline at the end of each line. The format can be specified explicitly using the `--fields-` *`xxx`* and `--lines-terminated-by` options. Column values are converted to the character set specified by the `--default-character-set` option.

–tables

This option overrides the `--databases` (`-B`) option. `mariadb-dump` regards all name arguments following the option as table names.

–tls-version=*protocol*

This option accepts a *comma-separated list of TLS protocol versions*. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted. See Secure Connections Overview: TLS Protocol Versions for more information.

–triggers

Include triggers for each dumped table in the output. This option is enabled by default; disable it with `--skip-triggers`.

–tz-utc

This option enables TIMESTAMP columns to be dumped and reloaded between servers in different time zones. `mariadb-dump` sets its connection time zone to UTC before starting the dump and adds `SET TIME_ZONE='+00:00'` to the top of the dump file. Without this option, `TIMESTAMP` columns are dumped and reloaded in the time zones local to the source and destination servers, which can cause the values to change if the servers are in different time zones. This option also protects against changes due to daylight saving time. The time zone setting affects how timestamp arguments (like those in `--as-of`) are interpreted.

`--tz-utc` is enabled by default. To disable it, use `--skip-tz-utc`.

Note the interaction of this option with `--as-of` when dumping versioned data.

–update-history

This option updates the `row_end` history timestamp to support dates up to the year 2106. When enabled, it also activates `tz-utc`. The option should be used when upgrading to **MariaDB 11.5 or later**, ensuring compatibility with extended date ranges and proper time zone handling.

-u *username*, –user=*username*

The MariaDB *user name* to use when connecting to the server.

-v, –verbose

Verbose mode. Print more information about what the program is doing during various stages.

-V, –version

Output version information and exit.

-w *condition*, –where=*condition*

Dump only rows selected by the given `WHERE` *condition*. Quotes around the condition are mandatory if it contains spaces or other characters that are special to your command interpreter. Example:\ `--where="user = ´jimf´" -w"userid > 1" -w"userid < 1"` .

-X, –xml

Dump a database as well-formed XML.

Group Options

Some `mariadb-dump` options are shorthand for groups of other options:

- Use of `--opt` is the same as specifying `--add-drop-table` , `--add-locks` , `--create-options` , `--disable-keys` , `--extended-insert` , `--lock-tables` , `--quick` , and `--set-charset` . All of the options that `--opt` stands for are also on by default because `--opt` is on by default.
- Use of `--compact` is the same as specifying `--skip-add-drop-table` , `--skip-add-locks` , `--skip-comments` , `--skip-disable-keys` , and `--skip-set-charset` options.

To reverse the effect of a group option, use its `--skip-xxx` form (`--skip-opt` or `--skip-compact`). It is also possible to select only part of the effect of a group option by following it with options that enable or disable specific features. Here are some examples:

- To select the effect of `--opt` except for some features, use the `--skip` option for each feature. To disable extended inserts and memory buffering, use `--opt--skip-extended-insert` `--skip-quick` . (Actually, `--skip-extended-insert--skip-quick` is sufficient because `--opt` is on by default.)
- To reverse `--opt` for all features except index disabling and table locking, use `--skip-opt--disable-keys` `--lock-tables` .

When you selectively enable or disable the effect of a group option, the order is important because options are processed first to last. For example, `--disable-keys` `--lock-tables--skip-opt` would not have the intended effect; it is the same as `--skip-opt` by itself.

Special Characters in Option Values

Some options, like `--lines-terminated-by` , accept a string. The string can be quoted, if necessary. For example, on Unix systems, this is the option to enclose fields within double quotes:

```
--fields-enclosed-by='"'
```

An alternative is to specify the hexadecimal value of a character. For example, the following syntax works on any platform:

```
--fields-enclosed-by=0x22
```

Option Files

In addition to reading options from the command line, `mariadb-dump` can also read options from option files. If an unknown option is provided to `mariadb-dump` in an option file, then it is ignored.

The following options relate to how MariaDB command-line tools handle option files. They must be given as the first argument on the command line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don’t read default options from any option file.

Option	Description
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

`mariadb-dump` is linked with MariaDB Connector/C. However, MariaDB Connector/C does not handle the parsing of option files for this client. That is performed by the server's option file parsing code. See MDEV-19035 for more information.

Option Groups

`mariadb-dump` reads options from the following option groups from the option files:

Group	Description
[mysqldump]	Options read by mariadb-dump, which includes both MariaDB Server and MySQL Server.
[mariadb-dump]	Options read by `mariadb-dump`.
[client]	Options read by all MariaDB and MySQL client programs, which include both MariaDB and MySQL clients. For example, mysqldump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which are common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

NULL, ´NULL´, and Empty Values in XML

For a column named `column_name`, the `NULL` value, an empty string, and the string value `´NULL´` are distinguished from one another in the output generated by this option as follows.

Value	XML Representation
`NULL` (unknown value)	`<field name="column_name" xsi:nil="true" />`
`''` (empty string)	`<field name="column_name"></field>`
`'NULL'` (string value)	`<field name="column_name">NULL</field>`

The output from the mariadb client when run using the `--xml` option also follows the preceding rules.

XML output from mariadb-dump includes the XML namespace, as shown here:

```
shell> mariadb-dump --xml -u root world City
<?xml version="1.0"?>
<mariadb-dump xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<database name="world">
<table_structure name="City">
<field Field="ID" Type="int(11)" Null="NO" Key="PRI" Extra="auto_increment" />
<field Field="Name" Type="char(35)" Null="NO" Key="" Default="" Extra="" />
<field Field="CountryCode" Type="char(3)" Null="NO" Key="" Default="" Extra="" />
<field Field="District" Type="char(20)" Null="NO" Key="" Default="" Extra="" />
<field Field="Population" Type="int(11)" Null="NO" Key="" Default="0" Extra="" />
<key Table="City" Non_unique="0" Key_name="PRIMARY" Seq_in_index="1" Column_name="ID"
Collation="A" Cardinality="4079" Null="" Index_type="BTREE" Comment="" />
<options Name="City" Engine="MyISAM" Version="10" Row_format="Fixed" Rows="4079"
Avg_row_length="67" Data_length="273293" Max_data_length="18858823439613951"
Index_length="43008" Data_free="0" Auto_increment="4080"
Create_time="2007-03-31 01:47:01" Update_time="2007-03-31 01:47:02"
Collation="latin1_swedish_ci" Create_options="" Comment="" />
</table_structure>
<table_data name="City">
<row>
<field name="ID">1</field>
<field name="Name">Kabul</field>
<field name="CountryCode">AFG</field>
<field name="District">Kabol</field>
<field name="Population">1780000</field>
</row>
...
<row>
<field name="ID">4079</field>
<field name="Name">Rafah</field>
<field name="CountryCode">PSE</field>
<field name="District">Rafah</field>
<field name="Population">92020</field>
</row>
</table_data>
</database>
</mariadb-dump>
```

Restoring Dumps

To restore a backup created with `mariadb-dump`, use the [mariadb client](../mariadb-client/mariadb-command line-client.md) to import the dump:

```
mariadb db_name < backup-file.sql
```

Variables

You can also set the following variables (`--variable-name=value`) and boolean options `{FALSE|TRUE}` by using:

Name	Default Values
all	TRUE
all-databases	FALSE
all-tablespaces	FALSE
no-tablespaces	FALSE
add-drop-database	FALSE
add-drop-table	TRUE
add-drop-trigger	FALSE
add-locks	TRUE
allow-keywords	FALSE

Name	Default Values
apply-slave-statements	FALSE
as-of	(No default value)
character-sets-dir	(No default value)
comments	TRUE
compatible	(No default value)
compact	FALSE
complete-insert	FALSE
compress	FALSE
copy-s3-tables	FALSE
create-options	TRUE
databases	FALSE
debug-check	FALSE
debug-info	FALSE
default-character-set	utf8mb4
delayed-insert	FALSE
delete-master-logs	FALSE
dir	(No default value)
disable-keys	TRUE
dump-history	FALSE
dump-slave	0
events	FALSE
extended-insert	TRUE
fields-terminated-by	(No default value)
fields-enclosed-by	(No default value)
fields-optionally-enclosed-by	(No default value)
fields-escaped-by	(No default value)
flush-logs	FALSE
flush-privileges	FALSE
force	FALSE
gtid	FALSE
header	FALSE
hex-blob	FALSE
host	(No default value)
include-master-host-port	FALSE
insert-ignore	FALSE
lines-terminated-by	(No default value)
lock-all-tables	FALSE
lock-tables	TRUE
log-error	(No default value)

Name	Default Values
log-queries	TRUE
master-data	0
max_allowed_packet	1073741824
max-statement-time	0
net-buffer-length	1046528
no-autocommit	TRUE
no-create-db	FALSE
no-create-info	FALSE
no-data	FALSE
no-data-med	TRUE
order-by-primary	FALSE
order-by-size	FALSE
parallel	0
port	3306
quick	TRUE
quote-names	TRUE
replace	FALSE
routines	FALSE
set-charset	TRUE
single-transaction	FALSE
dump-date	TRUE
socket	No default value)
ssl	TRUE
ssl-ca	(No default value)
ssl-capath	(No default value)
ssl-cert	(No default value)
ssl-cipher	(No default value)
ssl-crl	(No default value)
ssl-crlpath	(No default value)
ssl-fp	(No default value)
ssl-fplist	(No default value)
ssl-key	(No default value)
ssl-verify-server-cert	TRUE
system	(No default value)
tab	(No default value)
triggers	TRUE
tz-utc	TRUE
tls-version	(No default value)
update-history	FALSE

Name	Default Values
user	(No default value)
verbose	FALSE
wildcards	FALSE
where	(No default value)
plugin-dir	(determined by installation path, e.g. `.../lib/plugin`)
default-auth	(No default value)

Examples

A common use of `mariadb-dump` is making a backup of an entire database:

```
shell> mariadb-dump db_name > backup-file.sql
```

You can load the dump file back into the server like this:

```
shell> mariadb db_name < backup-file.sql
```

Or like this:

```
shell> mariadb -e "source /path-to-backup/backup-file.sql" db_name
```

`mariadb-dump` is also very useful for populating databases by copying data from one MariaDB server to another:

```
shell> mariadb-dump --opt db_name | mariadb --host=remote_host -C db_name
```

It is possible to dump several databases with one command:

```
shell> mariadb-dump --databases db_name1 [db_name2 ...] > my_databases.sql
```

To dump all databases, use the `--all-databases` option:

```
shell> mariadb-dump --all-databases > all_databases.sql
```

For InnoDB tables, `mariadb-dump` provides a way of making an online backup:

```
shell> mariadb-dump --all-databases --single-transaction all_databases.sql
```

This backup acquires a global read lock on all tables (using `FLUSH TABLES WITH READ LOCK`) at the beginning of the dump. As soon as this lock has been acquired, the binary log coordinates are read, and the lock is released. If long updating statements are running when the `FLUSH` statement is issued, the MariaDB server may get stalled until those statements finish. After that, the dump becomes lock-free and does not disturb reads and writes on the tables. If the update statements that the MariaDB server receives are short (in terms of execution time), the initial lock period should not be noticeable, even with many updates.

For point-in-time recovery (also known as “roll-forward,” when you need to restore an old backup and replay the changes that happened since that backup), it is often useful to rotate the binary log or at least know the binary log coordinates to which the dump corresponds:

```
shell> mariadb-dump --all-databases --master-data=2 > all_databases.sql
```

Or:

```
shell> mariadb-dump --all-databases --flush-logs --master-data=2 > all_databases.sql
```

The `--master-data` and `--single-transaction` options can be used simultaneously, which provides a convenient way to make an online backup suitable for use prior to point-in-time recovery if tables are stored that use the InnoDB storage engine.

See Also

- mariadb-backup
- MariaDB point-in-time recovery (video)
- MariaDB Enterprise Backup
- Upgrading to a newer major version of MariaDB (video)
- MariaDB dump file compatibility change (blog, 2024)

This page is licensed: CC BY-SA / Gnu FDL

mariadb-hotcopy

`mariadb-hotcopy` is a Perl script that was originally written and contributed by Tim Bunce. It uses FLUSH TABLES, LOCK TABLES, and cp or scp to make a database backup.

> **Note:**
>
> Previously, the client was called `mysqlhotcopy`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

It is a fast way to make a backup of the database or single tables, but it can be run only on the same machine where the database directories are located. `mariadb-hotcopy` works only for backing up MyISAM and ARCHIVE tables. It runs on Unix and NetWare.

To use `mariadb-hotcopy`, you must have read access to the files for the tables that you are backing up, the `SELECT` privilege for those tables, the `RELOAD` privilege (to be able to execute `FLUSH TABLES`), and the `LOCK TABLES` privilege (to be able to lock the tables).

```
shell> mariadb-hotcopy db_name [/path/to/new_directory]
shell> mariadb-hotcopy db_name_1 ... db_name_n /path/to/new_directory
```

Back up tables in the given database that match a regular expression:

```
shell> mariadb-hotcopy db_name./regex/
```

The regular expression for the table name can be negated by prefixing it with a tilde (`~`):

```
shell> mariadb-hotcopy db_name./~regex/
```

`mariadb-hotcopy` supports the following options, which can be specified on the command line or in the [`mariadb-hotcopy`] and [`client`] option file groups.

`--help`, `-?`

Display a help message and exit.

`--addtodest`

Do not rename target directory (if it exists); merely add files to it.

`--allowold`

Do not abort if a target exists; rename it by adding an _old suffix.

`--checkpoint` ***=db_name.tbl_name***

Insert checkpoint entries into the specified database *db_name* and table *tbl_name*.

`--chroot` ***=directory***

Base *directory* of the chroot jail in which mariadbd operates. The path value should match that of the `--chroot` option given to `mariadbd`.

`--debug`

Enable debug output.

`--dryrun`, **`-n`**

Report actions without performing them.

`--flushlog`

Flush logs after all tables are locked.

`--host` ***=host_name***, **`-h`** ***host_name***

The *host name* of the local host to use for making a TCP/IP connection to the local server. By default, the connection is made to localhost using a Unix socket file.

`--keepold`

Do not delete previous (renamed) target when done.

`--method` ***=method***

The *method* for copying files (`cp` or `scp`). The default is `cp`.

`--noindices`

Do not include full index files for MyISAM tables in the backup. This makes the backup smaller and faster. The indexes for reloaded tables can be reconstructed later with myisamchk -rq.

`--old-server`

Connect to old MySQL server (before MySQL 5.5) which doesn't have FLUSH TABLES WITH READ LOCK fully implemented.

`--password` ***=password***, **`-p`** ***password***

The *password* to use when connecting to the server. The password value is mandatory for this option, unlike for other MariaDB programs. Specifying a password on the command line should be considered insecure. You can use an option file to avoid giving the password on the command line.

`--port` ***=port-number***, **`-P`** ***port-number***

The TCP/IP *port number* to use when connecting to the local server.

`--quiet`, `-q`

Be silent except for errors.

`--record\_log\_pos`=*db_name*._tbl_name_

Record master and slave status in the specified database *db_name* and table *tbl_name*.

`--regexp`=*expression*

Copy all databases with names that match the given regular *expression*.

`--resetmaster`

Reset the binary log after locking all the tables.

`--resetslave`

Reset the `master.info` file after locking all the tables.

`--socket`=*socket-file*, `-S` *socket-file*

The Unix *socket file* to use for connections to localhost.

`--suffix`=*string*

The *suffix string* to use for names of copied databases.

`--tmpdir`=*directory*

The temporary *directory*. The default is `/tmp`.

`--user`=*username*, `-u` *username*

The MariaDB *username* to use when connecting to the server.

Use perldoc for additional `mariadb-hotcopy` documentation, including information about the structure of the tables needed for the `--checkpoint` and ``–record`_log_pos` options:

```
shell> perldoc mariadb-hotcopy
```

See Also

- mariadb-dump
- mariadb-backup

This page is licensed: CC BY-SA / Gnu FDL

mariadb-import

`mariadb-import` loads tables from text files in various formats.

> **Note:**
>
> Previously, the client was called `mysqlimport`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

`mariadb-import` loads tables from text files in various formats. The base name of the text file must be the name of the table that should be used. If one uses sockets to connect to the MariaDB server, the server will open and read

the text file directly. In other cases the client will open the text file. The SQL statement LOAD DATA INFILE is used to import the rows.

Usage

The command to use `mariadb-import` and the general syntax is:

```
mariadb-import [OPTIONS] database textfile1 [textfile2 ...]
```

Options

`mariadb-import` supports the following options:

`--character-sets-dir`=*directory*

Directory for character set files.

`-c` *cols*, `--columns`=*cols*

Use only these columns to import the data to. Give the column names in a comma separated list. This is same as giving columns to LOAD DATA INFILE.

`-C`, `--compress`

Use compression in server/client protocol.

`--database`=*database*

Restore the specified database, ignoring others.To specify more than one database to include, use the directive multiple times, once for each database. Only takes effect when used together with the `--dir` option. This option is available from MariaDB 11.6.

`--debug` [=options]

Output debug log. Often this is `d:t:o,filename`. The default is `d:t:o`.

`--debug-check`

Check memory and open file usage at exit.

`--debug-info`

Print some debug info at exit.

`--default-auth`=*plugin*

Default authentication client-side *plugin* to use.

`--default-character-set`=*character-set*

Set the default *character set*.

`--defaults-extra-file`=*file*

Read this *file* after the global files are read. Must be given as the first option.

`--defaults-file`=*file*

Only read default options from the given *file* name Must be given as the first option.

`--defaults-group-suffix` ***=group-suffix***

In addition to the given groups, also read groups with this *suffix*.

`-d`, **`--delete`**

First delete all rows from table.

`--dir` ***=directory***

Restore all tables from backup directory created using mariadb-dump –dir. This option is available from MariaDB 11.6.

`--fields-terminated-by` ***=string***

Fields in the input file are terminated by the given *string*.

`--fields-enclosed-by` ***=character***

Fields in the import file are enclosed by the given *character*.

`--fields-optionally-enclosed-by` ***=character***

Fields in the input file are optionally enclosed by the given *character*.

`--fields-escaped-by` ***=character***

Fields in the input file are escaped by the given *character*.

`-f`, **`--force`**

Continue even if we get an SQL error.

`-?`, **`--help`**

Display this help and exits.

`-h` ***host***, **`--host`** ***=host***

Connect to host.

`-i`, **`--ignore`**

If duplicate unique key was found, keep old row.

`-k`, **`--ignore-foreign-keys`**

Disable foreign key checks while importing the data.

`--ignore-database` ***=database***

Do not restore the specified database. To specify more than one database to ignore, use the directive multiple times, once for each database. Only takes effect when used together with the `--dir` option. This option is available from MariaDB 11.6.

`--ignore-lines` ***=n***

Ignore first n lines of data infile.

`--ignore-table` ***=table***

Do not restore the specified table. To specify more than one table to ignore, use the directive multiple times, once for each table. Each table must be specified with both database and table names, for instance, `--ignore-table`=*database*._table_. Only takes effect when used together with the `--dir` option. This option is available from MariaDB 11.6.

`--innodb-optimize-keys`

Create secondary indexes after data load, which speeds up loading (InnoDB only). Defaults to on; use `--skip-innodb-optimize-keys` to disable. This option is available from MariaDB 11.8.

`--lines-terminated-by` ***=string***

Lines in the input file are terminated by the given *string*.

`-L`, **`--local`**

Read all files through the client.

`-l`, **`--lock-tables`**

Lock all tables for write (this disables threads).

`--low-priority`

Use `LOW_PRIORITY` when updating the table.

`--no-defaults`

Don't read default options from any option file. Must be given as the first option.

`-j`, **`--parallel`** ***=number***

Number of LOAD DATA jobs executed in parallel. This option is available from MariaDB 11.4.1. `--use-threads` is a synonym.

`-p` ***password***, **`--password`** ***password***

Password to use when connecting to server. If password is not given, it's asked from the terminal. Specifying a password on the command line should be considered insecure. You can use an option file to avoid giving the password on the command line.

`--pipe`, **`-W`**

n Windows, connect to the server via a named pipe. This option applies only if the server supports named-pipe connections.

`--plugin-dir`

Directory for client-side plugins.

`-P` ***port-number***, **`--port`** ***=port-number***

Port number to use for connection or `0` for default to, in order of preference, `my.cnf`, the `MYSQL_TCP_PORT` environment variable, `/etc/services`. Default is `3306`.

`--print-defaults`

Print the program argument list and exit. Must be given as the first option.

`--protocol`=*protocol*

The *protocol* to use for connection (`tcp`, `socket`, `pipe`, `memory`).

`-r`, **`--replace`**

If duplicate unique key was found, replace old row.

`--shared-memory-base-name`

Shared-memory name to use for Windows connections using shared memory to a local server (started with the `--shared-memory` option). Case sensitive.

`-s`, **`--silent`**

Silent mode. Produce output only when errors occur.

`-S`, **`--socket`**={*socket*|*named-pipe*}

For connections to localhost, the Unix *socket* file to use, or, on Windows, the name of the *named pipe* to use.

`--ssl`

Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The `--ssl` option does not enable verifying the server certificate by default. In order to verify the server certificate, the user must specify the `--ssl-verify-server-cert` option.

`--ssl-ca`=*pem-file*

Define a path to a *PEM file* that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option implies the `--ssl` option.

`--ssl-capath`=*pem-directory*

Define a *path to a directory that contains one or more PEM files* that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the `--ssl` option. |

`--ssl-cert`=*file*

Define a path to the X509 *certificate file* to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

`--ssl-cipher`=*cipher-list*

List of permitted ciphers or cipher suites to use for TLS. This option implies the `--ssl` option.

`--ssl-crl`=*pem-file*

Defines a path to a *PEM file* that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL or

Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

`--ssl-crlpath` *=pem-directory*

Define a *path to a directory that contains one or more PEM files* that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

`--ssl-key` *=key-file*

Define a path to a private *key file* to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

`--ssl-verify-server-cert`

Enable server certificate verification. This option is disabled by default.

`--table` *=table*

Restore the specified *table* ignoring others. Use `--table`=dbname.tablename with this option. To specify more than one table to include, use the directive multiple times, once for each table. Only takes effect when used together with the `--dir` option. This option is available from MariaDB 11.6.

`--tls-version` *=tls-list*

This option accepts a *comma-separated list of TLS protocol versions*. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted. See Secure Connections Overview: TLS Protocol Versions for more information.

`--use-threads` *=number*

Load files in parallel. The argument is the *number of threads* to use for loading data. From MariaDB 11.4.1, a synonym for -j, `--parallel`=num.

`-u` *username*, `--user` *=username*

User for login if not current user.

`-v`, `--verbose`

Print info about the various stages.

`-V`, `--version`

Output version information and exit.

Option Files

In addition to reading options from the command line, `mariadb-import` can also read options from option files. If an unknown option is provided to `mariadb-import` in an option file, then it is ignored.

The following options relate to how MariaDB command line tools handles option files. They must be given as the first argument on the command line:

Option	Description
`--print-defaults`	Print the program argument list and exit.
`--no-defaults`	Don't read default options from any option file.
`--defaults-file` =#	Only read default options from the given file #.
`--defaults-extra-file` =#	Read this file after the global files are read.

`mariadb-import` is linked with MariaDB Connector/C. Therefore, it may be helpful to see Configuring MariaDB Connector/C with Option Files for more information on how MariaDB Connector/C handles option files.

Option Groups

`mariadb-import` reads options from the following option groups from option files:

Group	Description
[mysqlimport]	Options read by mysqlimport, which includes both MariaDB Server and MySQL Server.
[mariadb-import]	Options read by mysqlimport.
[client]	Options read by all MariaDB and MySQL client programs, which includes both MariaDB and MySQL clients. For example, mysqldump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

Default Values

Variables	Value (after reading options)
character-sets-dir	(No default value)
default-character-set	latin1
columns	(No default value)
compress	FALSE
debug-check	FALSE
debug-info	FALSE
delete	FALSE
fields-terminated-by	(No default value)
fields-enclosed-by	(No default value)
fields-optionally-enclosed-by	(No default value)
fields-escaped-by	(No default value)
force	FALSE
host	(No default value)
ignore	FALSE
ignore-lines	0
lines-terminated-by	(No default value)
local	FALSE
lock-tables	FALSE

Variables	Value (after reading options)
low-priority	FALSE
port	3306
replace	FALSE
silent	FALSE
socket	/var/run/mysqld/mysqld.sock
ssl	FALSE
ssl-ca	(No default value)
ssl-capath	(No default value)
ssl-cert	(No default value)
ssl-cipher	(No default value)
ssl-key	(No default value)
ssl-verify-server-cert	FALSE
use-threads	0
user	(No default value)
verbose	FALSE

This page is licensed: CC BY-SA / Gnu FDL

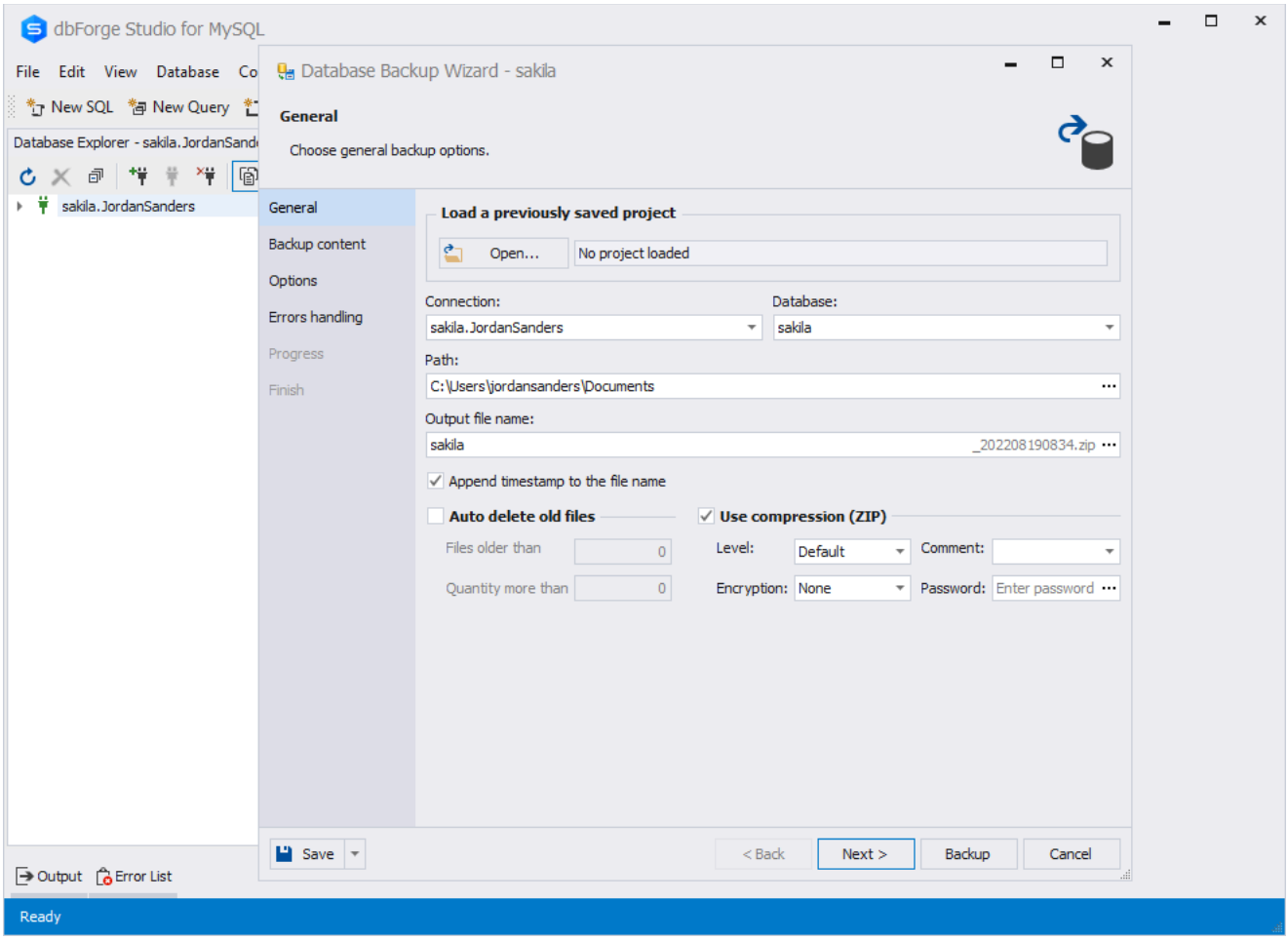
dbForge Studio

> **Note:**
>
> dbForge Studio is proprietary third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Content contributed by Devart; refer to its own documentation and license terms.

Without a doubt, you want your backup/restore and export/import operations to be fast, easy, and automated wherever possible. You can have it all that way with dbForge Studio for MySQL. As the name implies, it is an IDE for MySQL development, management, and administration, yet it works just as perfectly as a MariaDB GUI client. Now, let's see how it tackles routine database backups.

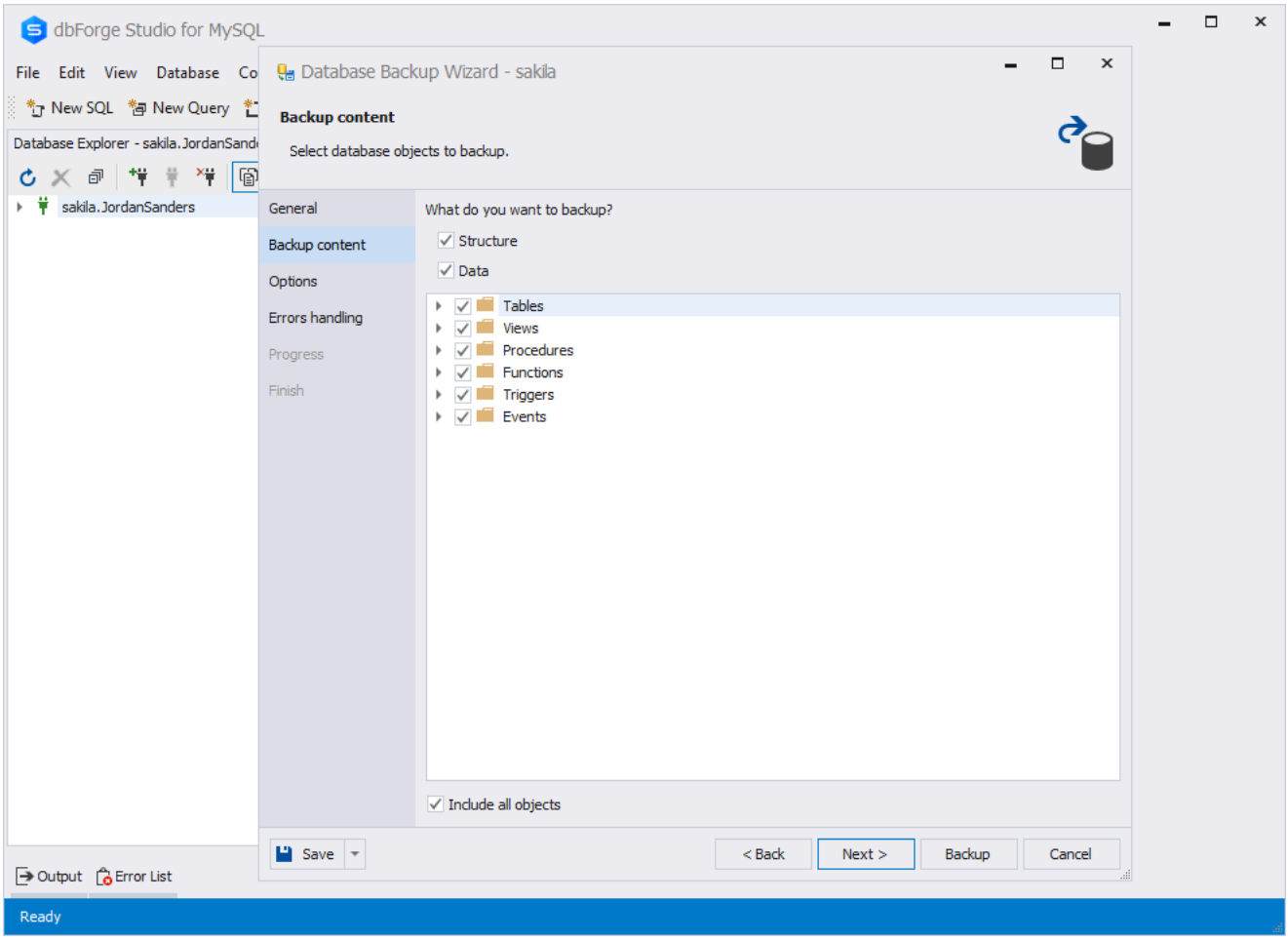
Create a MariaDB backup

1. On the **Database** menu, go to **Backup and Restore**, and click **Backup Database** to open **Database Backup Wizard**.
2. On the **General** page, specify the required connection and database, the path for the backup file to be saved to, and the output file name in the respective fields. Optionally, you can append a timestamp to the file name, enable the auto-deletion of old files, and compress your backup into an archive. After you set it all up, click **Next**.



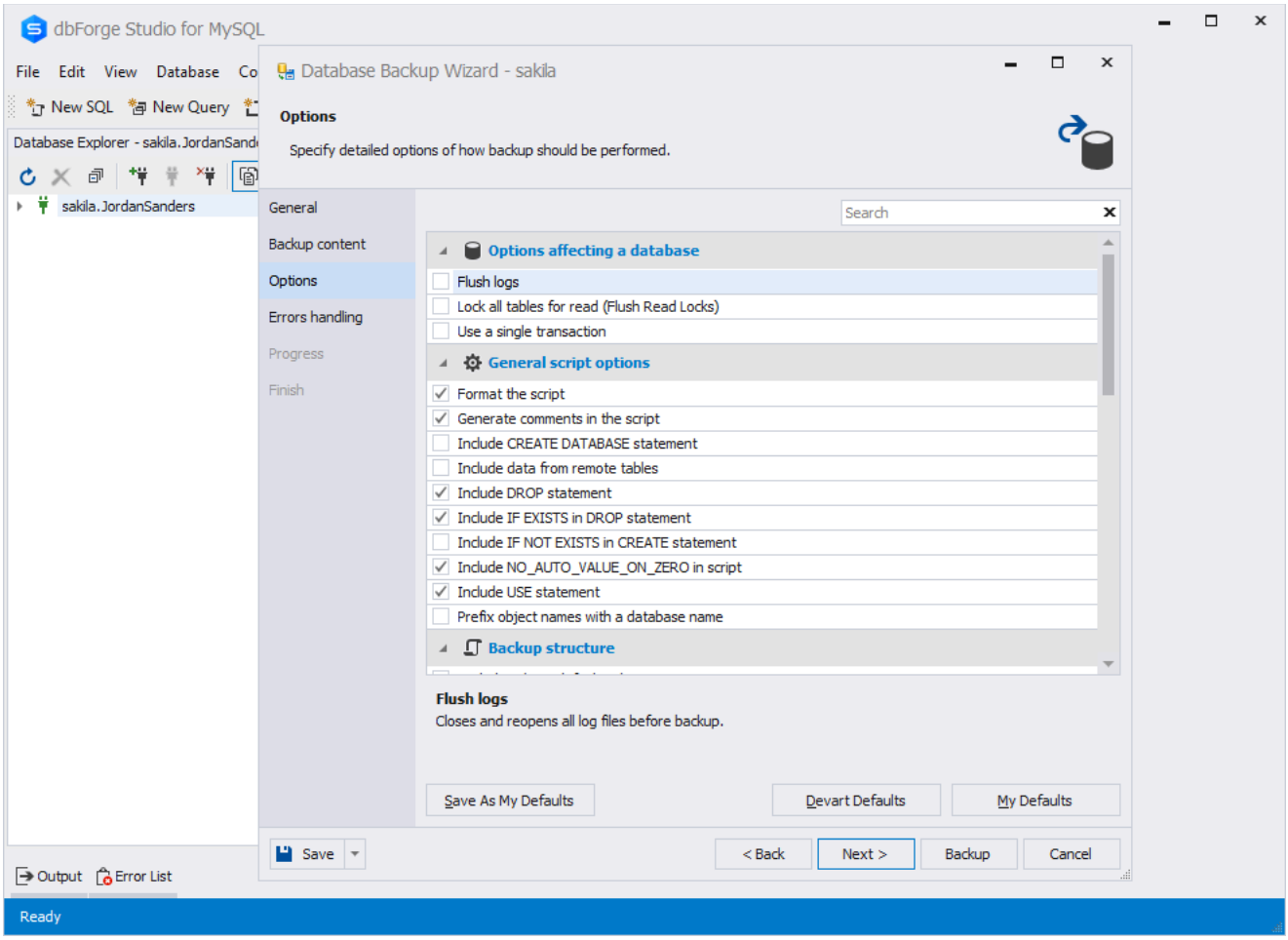
b1

3. On the **Backup content** page, select the content for your backup and click **Next**.



b2

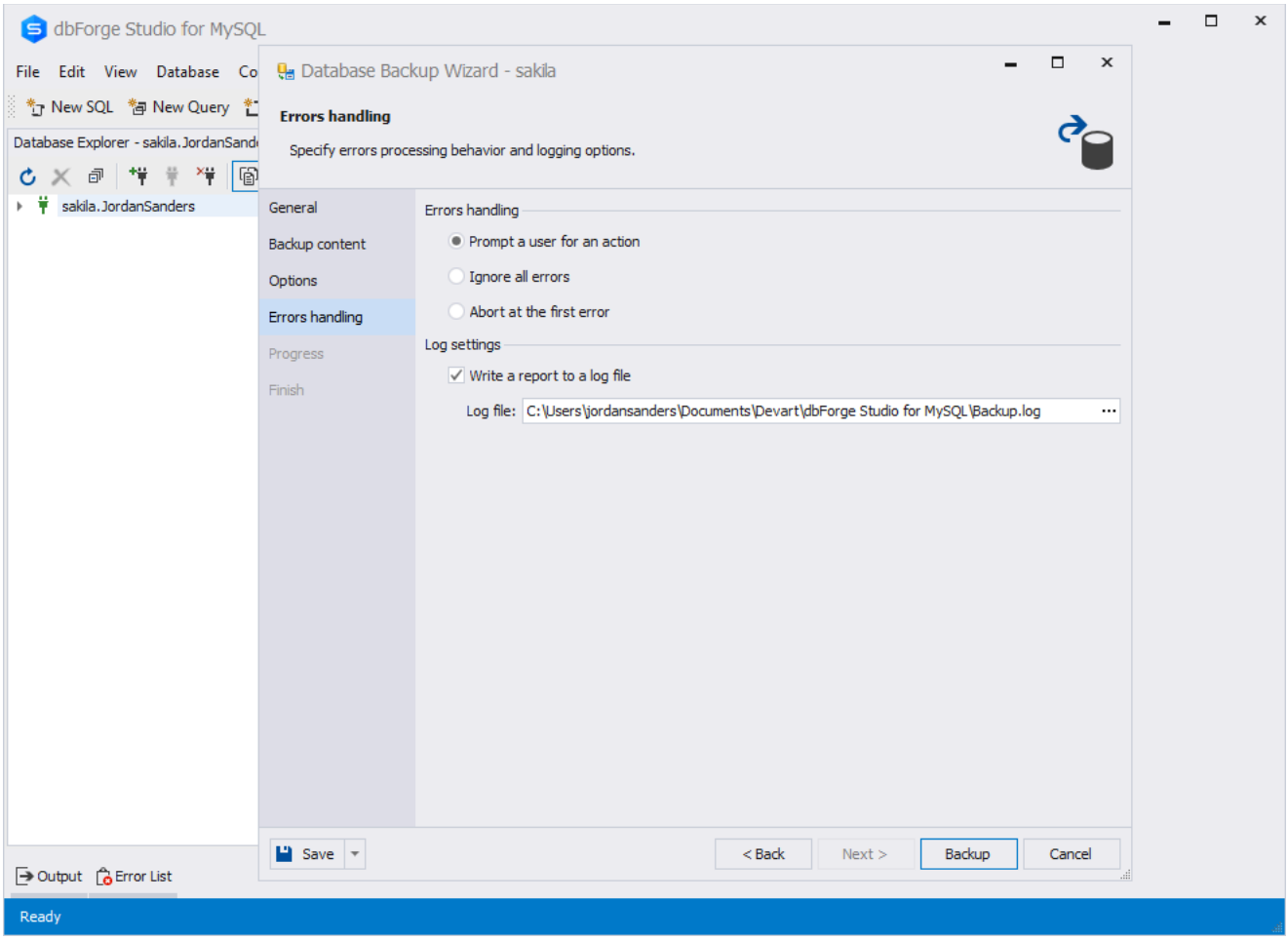
4. On the **Options** page, configure your detailed backup options—there are quite a few of those to match your requirements most precisely. Then click **Next**.



b3

5. On the **Errors handling** page, configure the **Errors handling** and **Log settings** options. Afterwards, click **Backup** to run the backup process.

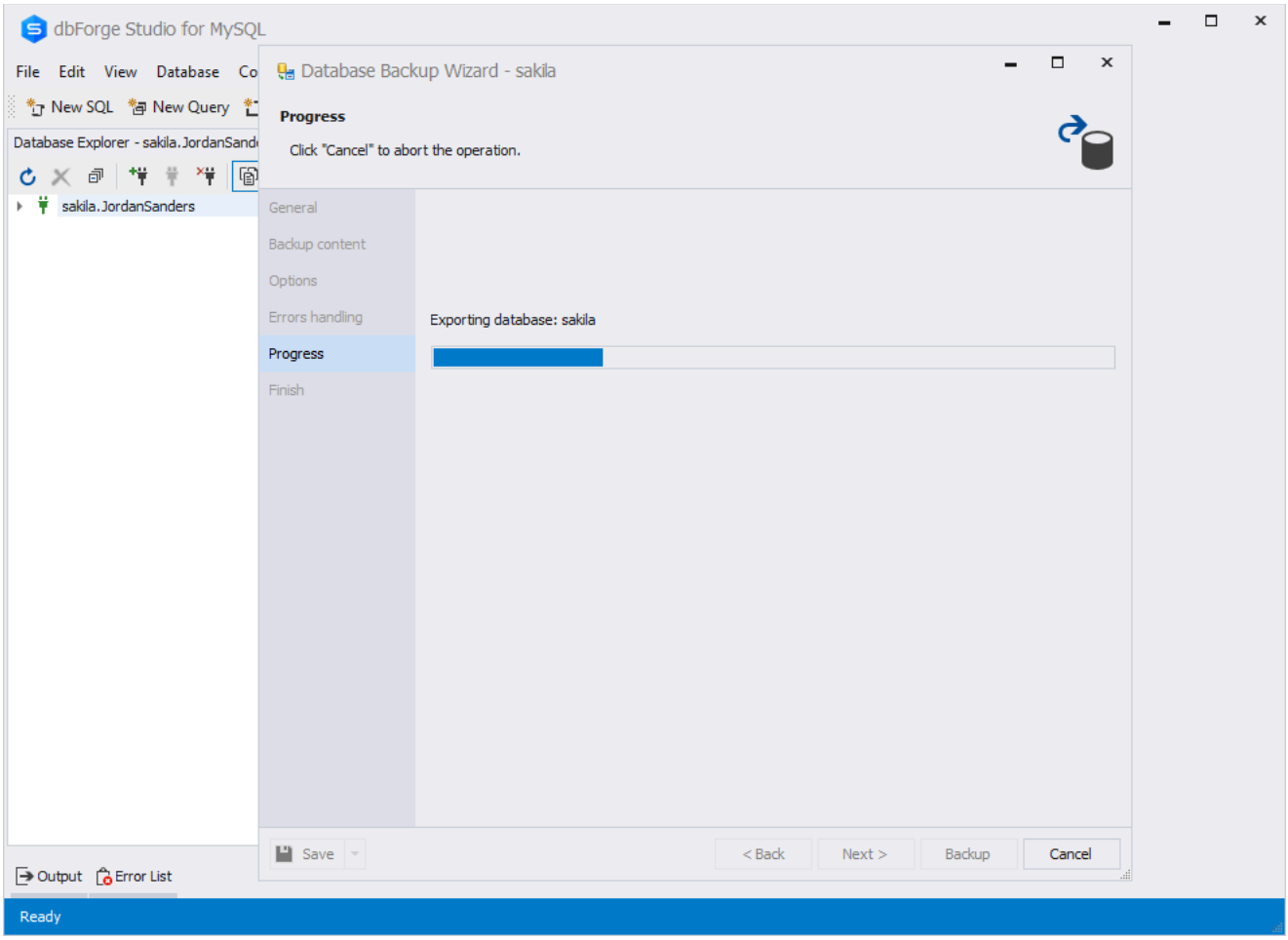
Note that you have two more options here: you can select **Save Project** to save your current backup project with all the settings—or you can select **Save Command Line** to save a backup script that you can execute from the command line whenever you need.



b4

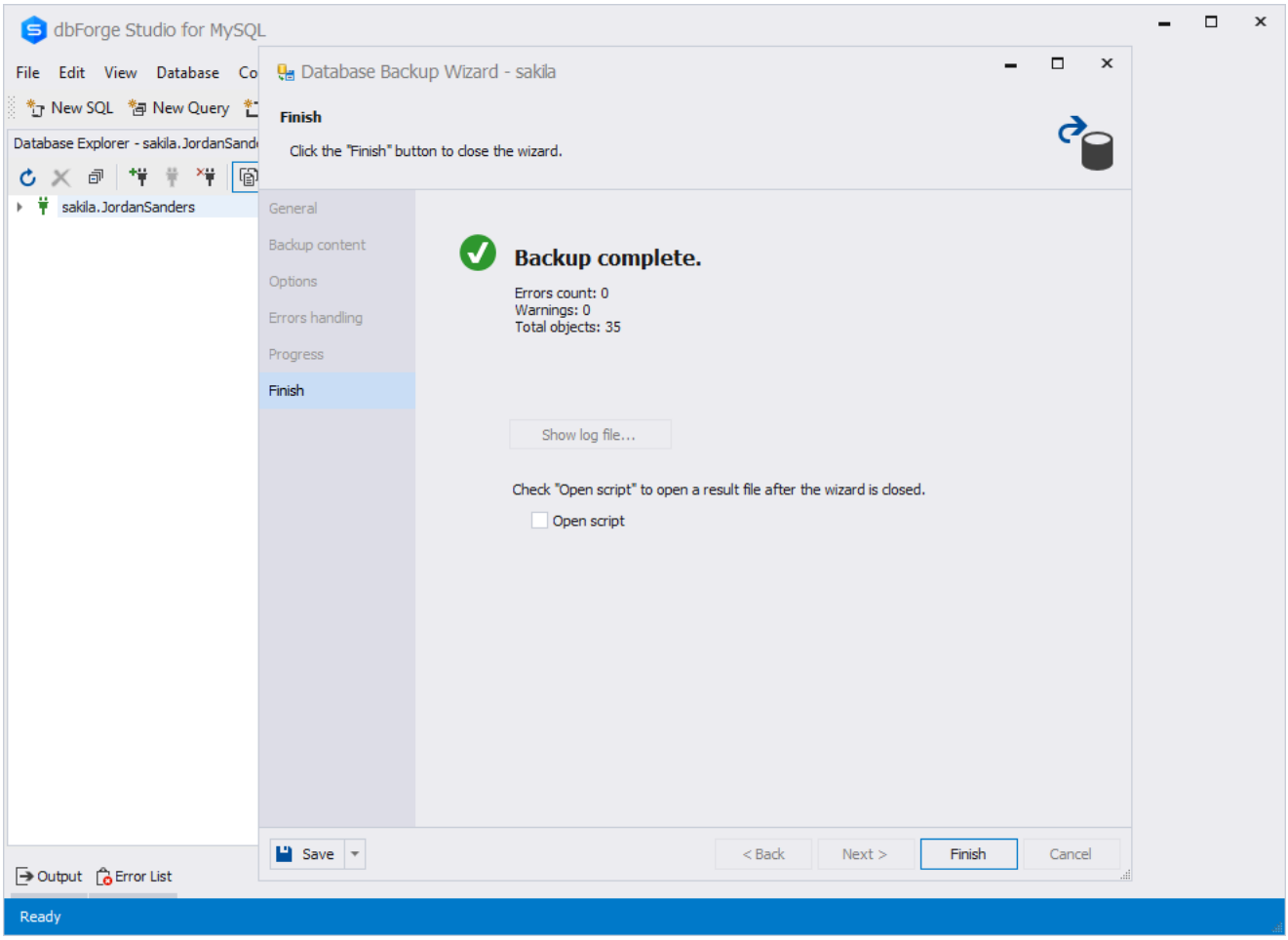
6. After you click **Backup**, wait for the backup process to be completed.

Note that you don't have to go through every wizard page to click **Backup**. You can do it whenever you've finished configuring your settings.



b5

7. Finally, confirm the successful completion by clicking **Finish**.



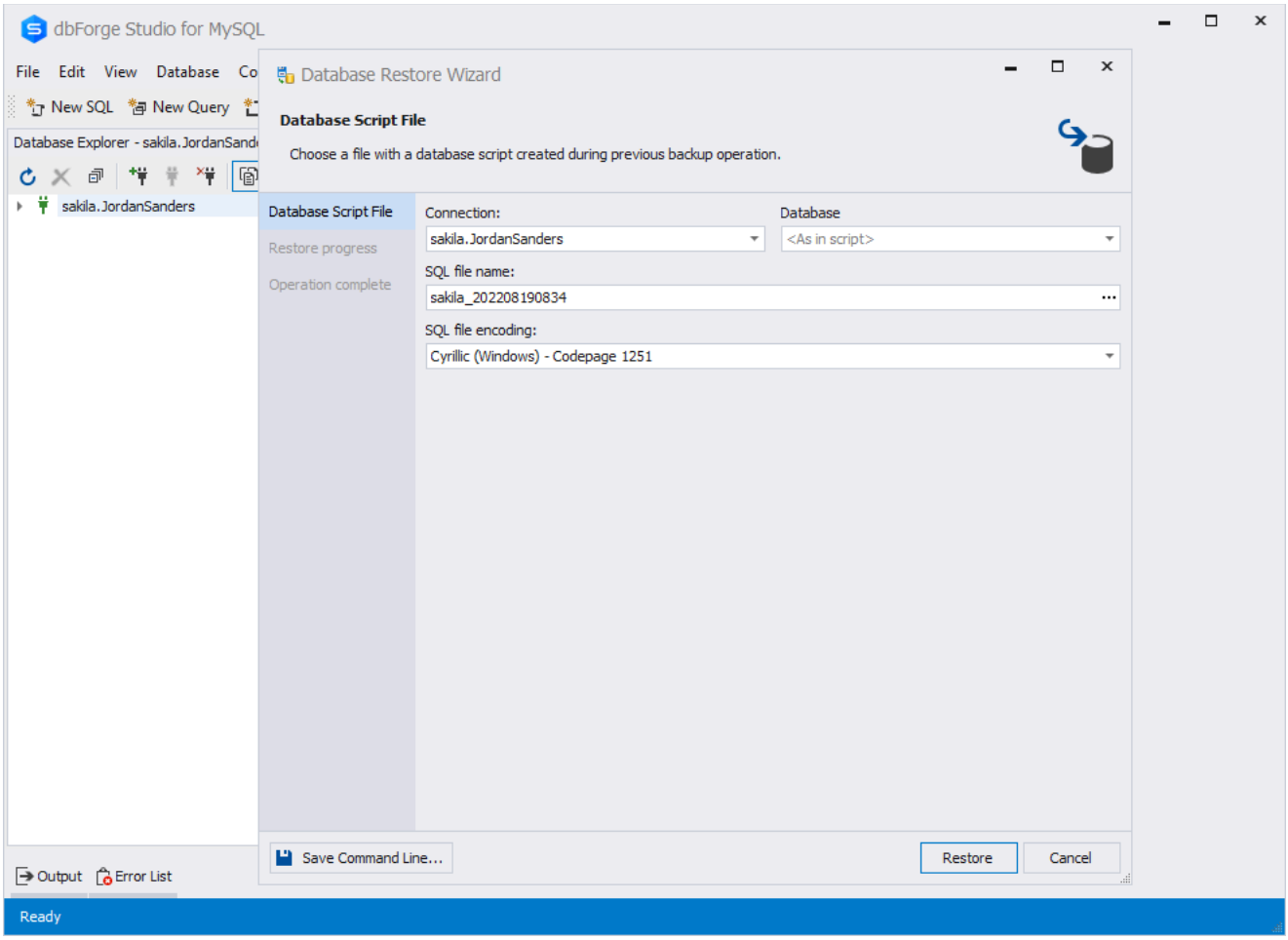
b6

As you can see, it's very easy. Furthermore, you can schedule to run regular backups using **Action > Create Basic Task** in **Windows Task Scheduler**.

Restore a MariaDB backup

This is an even faster task, done in half as many steps.

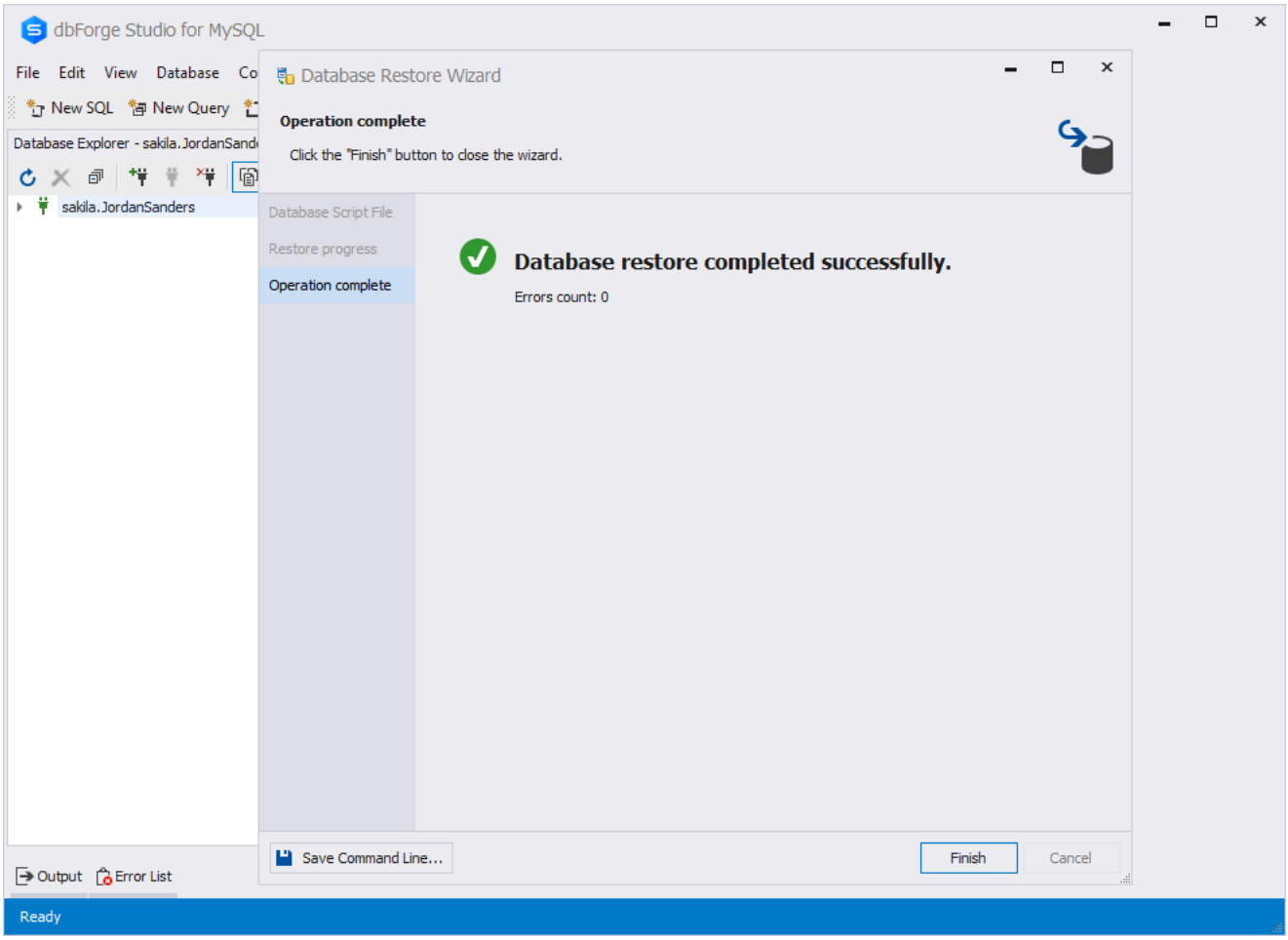
1. On the **Database** menu, go to ***Backup and Restore**, and click **Restore Database** to open the **Database Restore Wizard**.
2. On the **Database Script File** page, specify the required connection and database, as well as the path to the previously saved backup file.



r1

3. After that, click **Restore**, and let the Studio do the rest for you.

And when it's done, click **Finish**, and there you have it.



r2

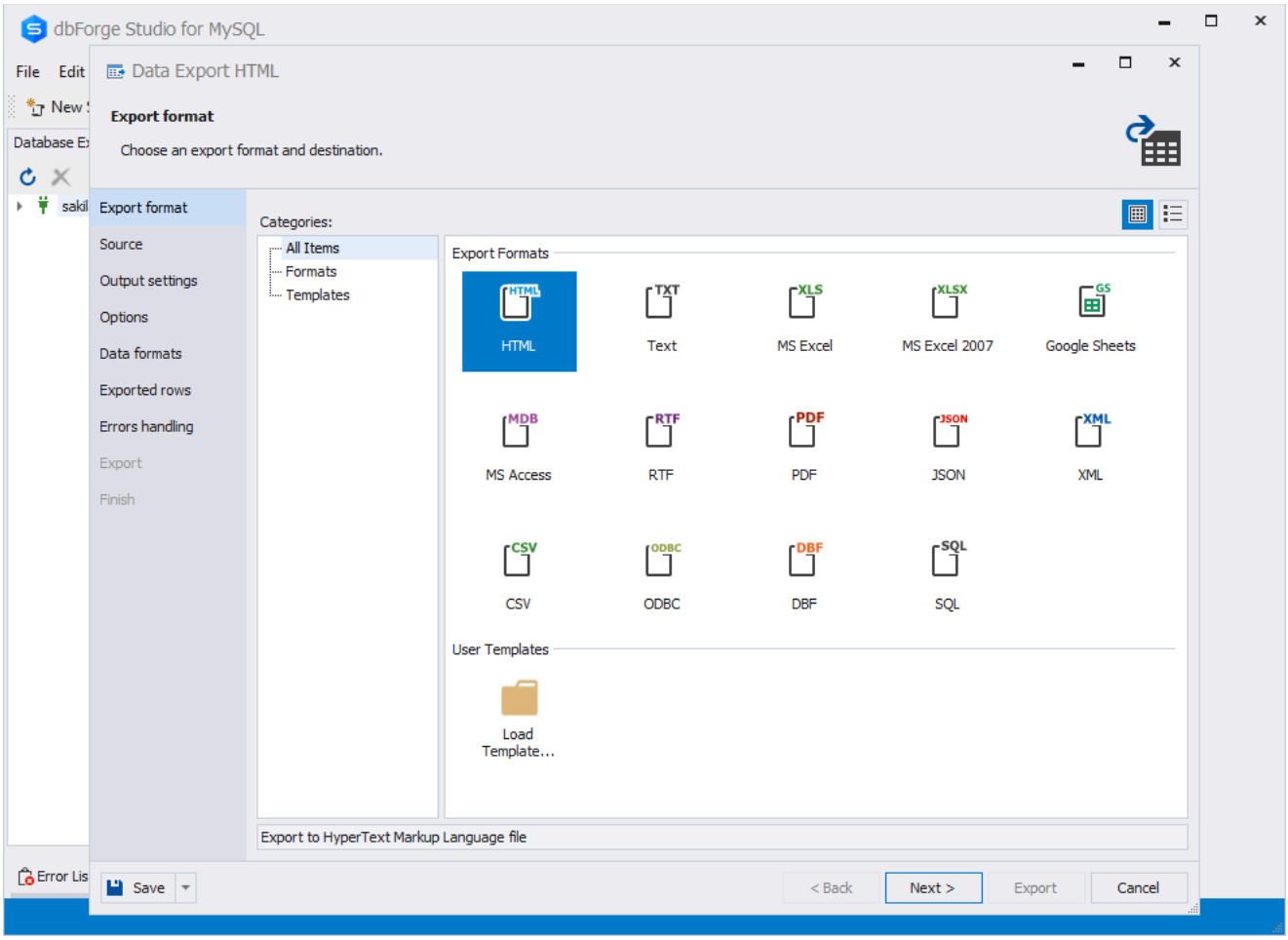
You can learn more about this functionality on the dedicated backup/restore page. Please note: while the page focuses on MySQL databases, everything that's described there is just as perfectly applicable to MariaDB from the same Studio with the same workflow.

Export data from MariaDB

With dbForge Studio, you can export data to 14 most popular formats: HTML, TXT, XLS, XLSX, MDB, RTF, PDF, JSON, XML, CSV, OBSC, DBF, SQL, and Google Sheets. You can do it with an easy-to-follow wizard that guides you through the entire process and delivers quite a few customization options.

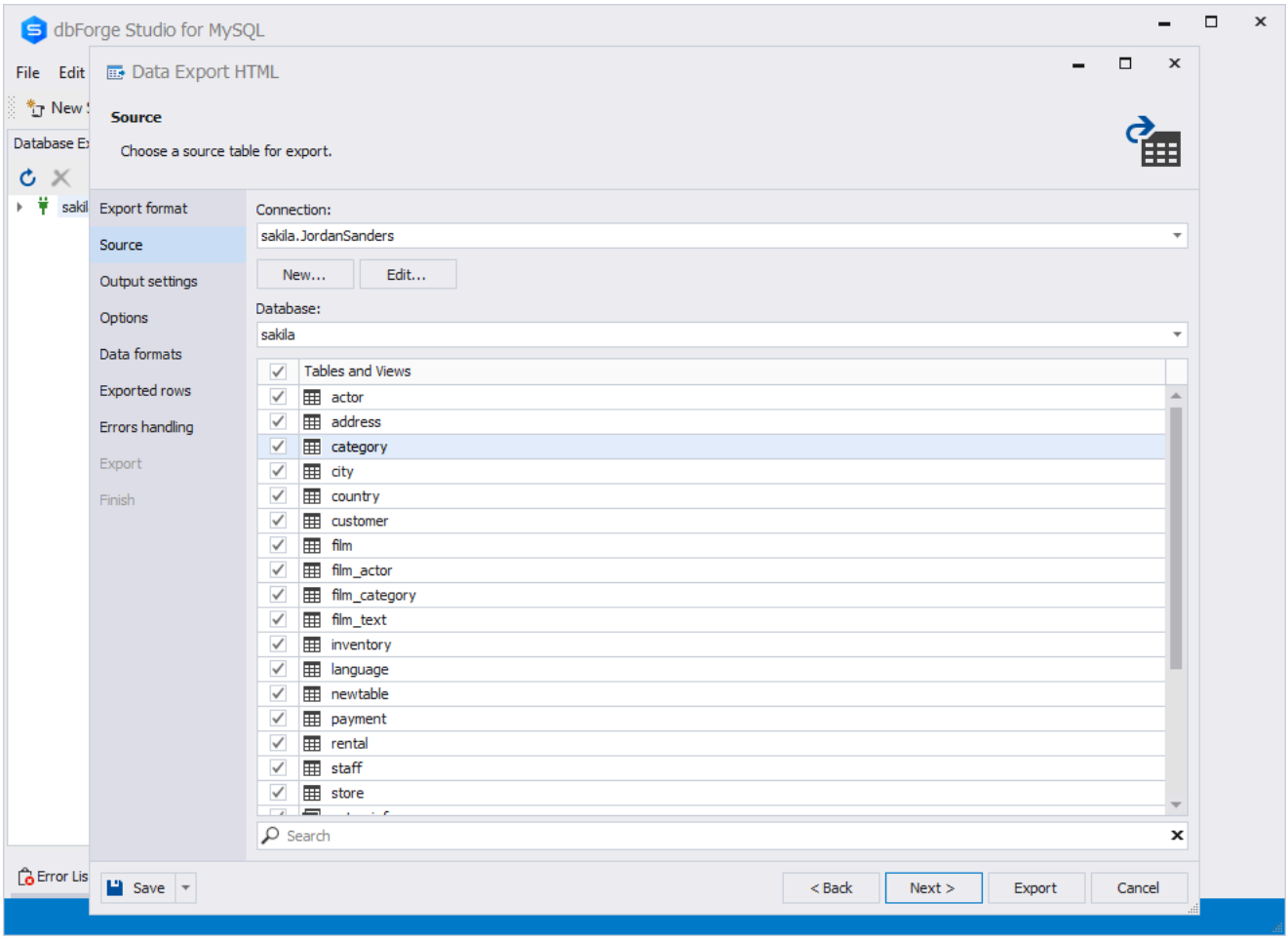
Let's see how it works. And before we start, note that different formats may have slightly different wizard pages. In our walkthrough, we'll take the HTML format as an example.

1. To open the export wizard, on the **Database** menu, click **Export Data**.
2. On the **Export format** page, pick the required format and click **Next**.



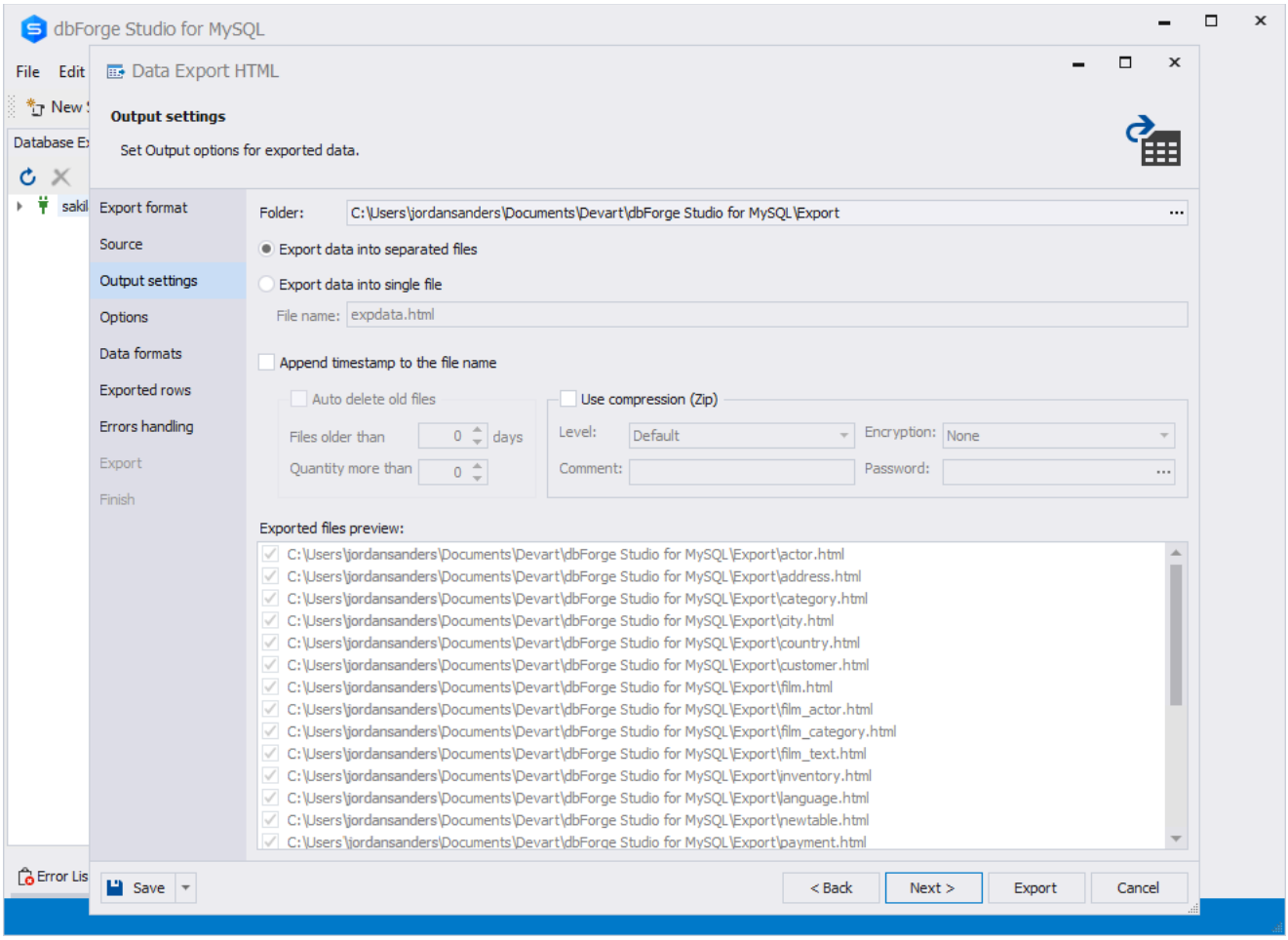
e1

3. On the **Source** page, select the required connection, database, as well as tables and views to be exported. Then click **Next**.



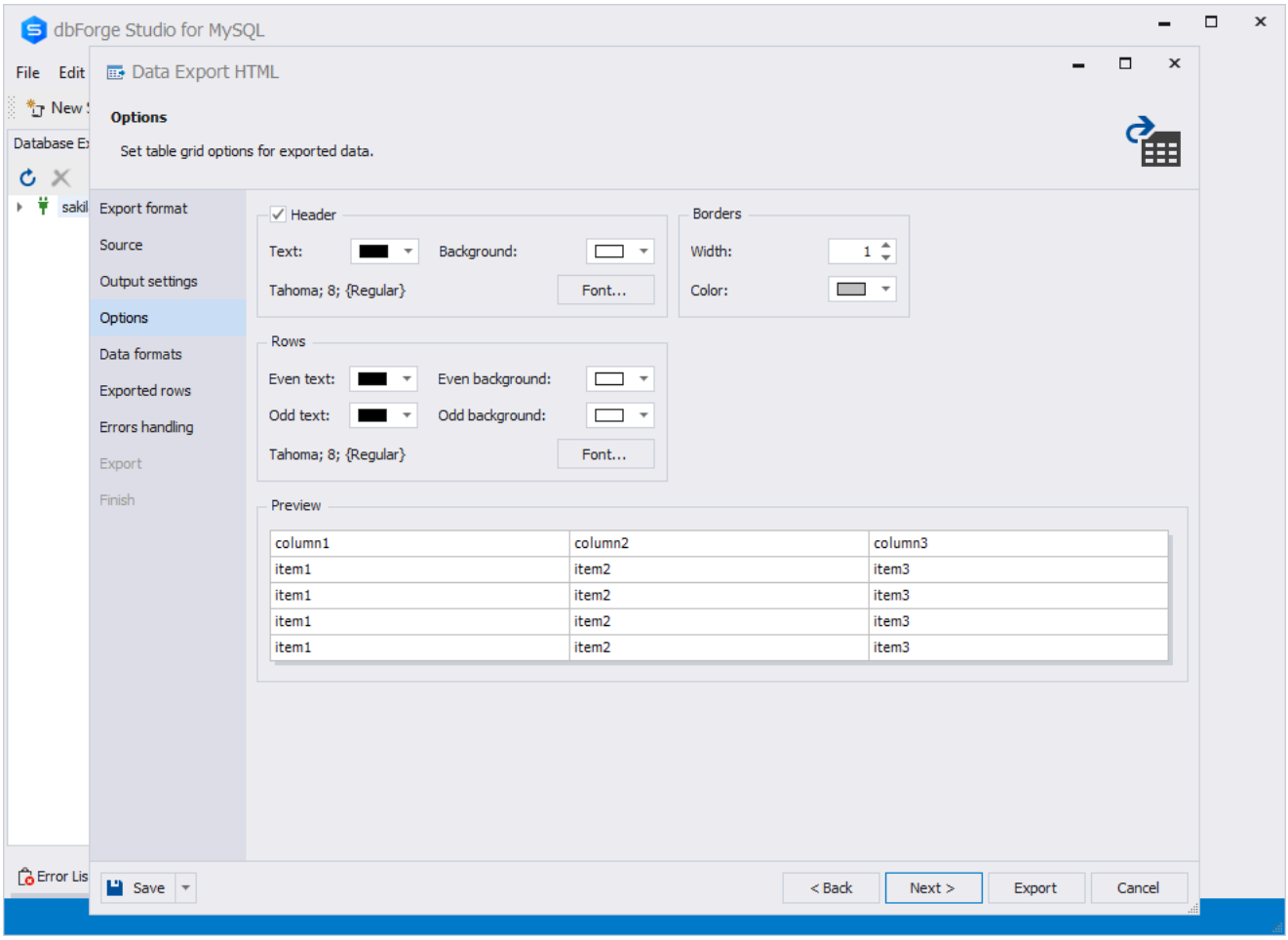
e2

4. On the **Output settings** page, specify the path for the output, select to export data into a single or several separate files, and configure a few other settings, such as timestamps and compression. Then click **Next**.



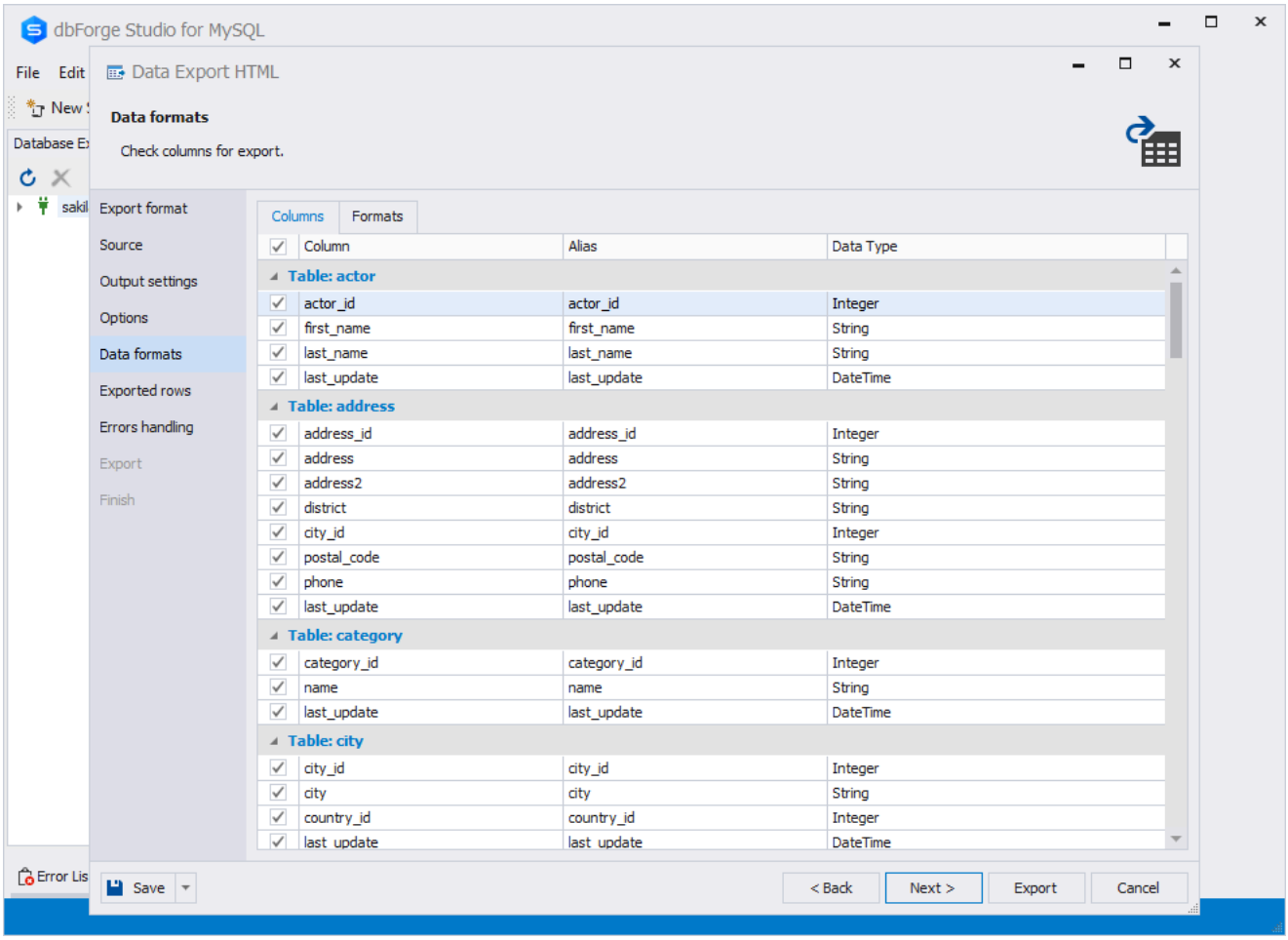
e3

5. On the **Options** page, configure and preview table grid options for exported data. Click **Next**.



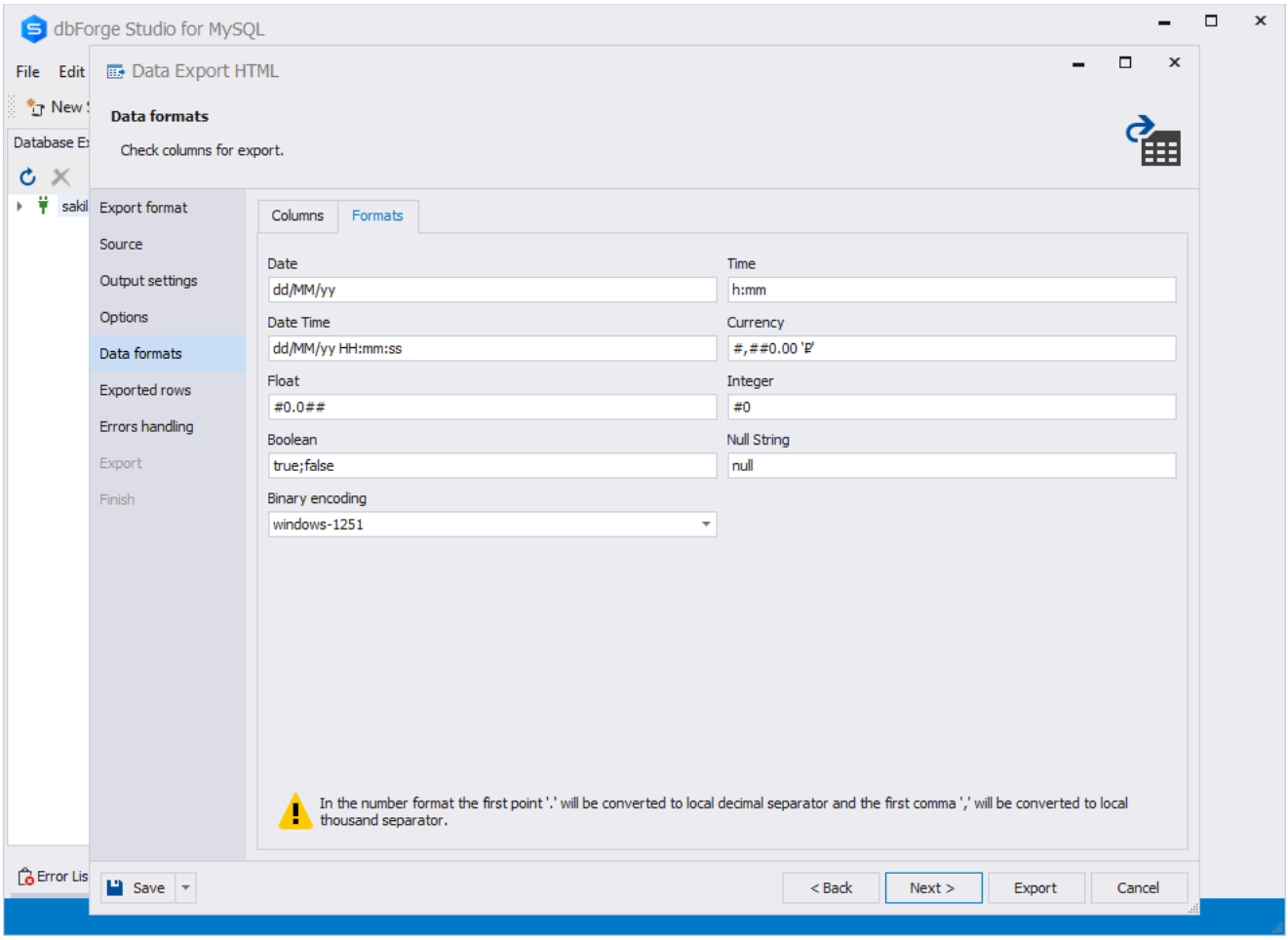
e4

6. On the **Data formats** page, you have two tabs. On the **Columns** tab, you can check the list of columns to be exported.



e5

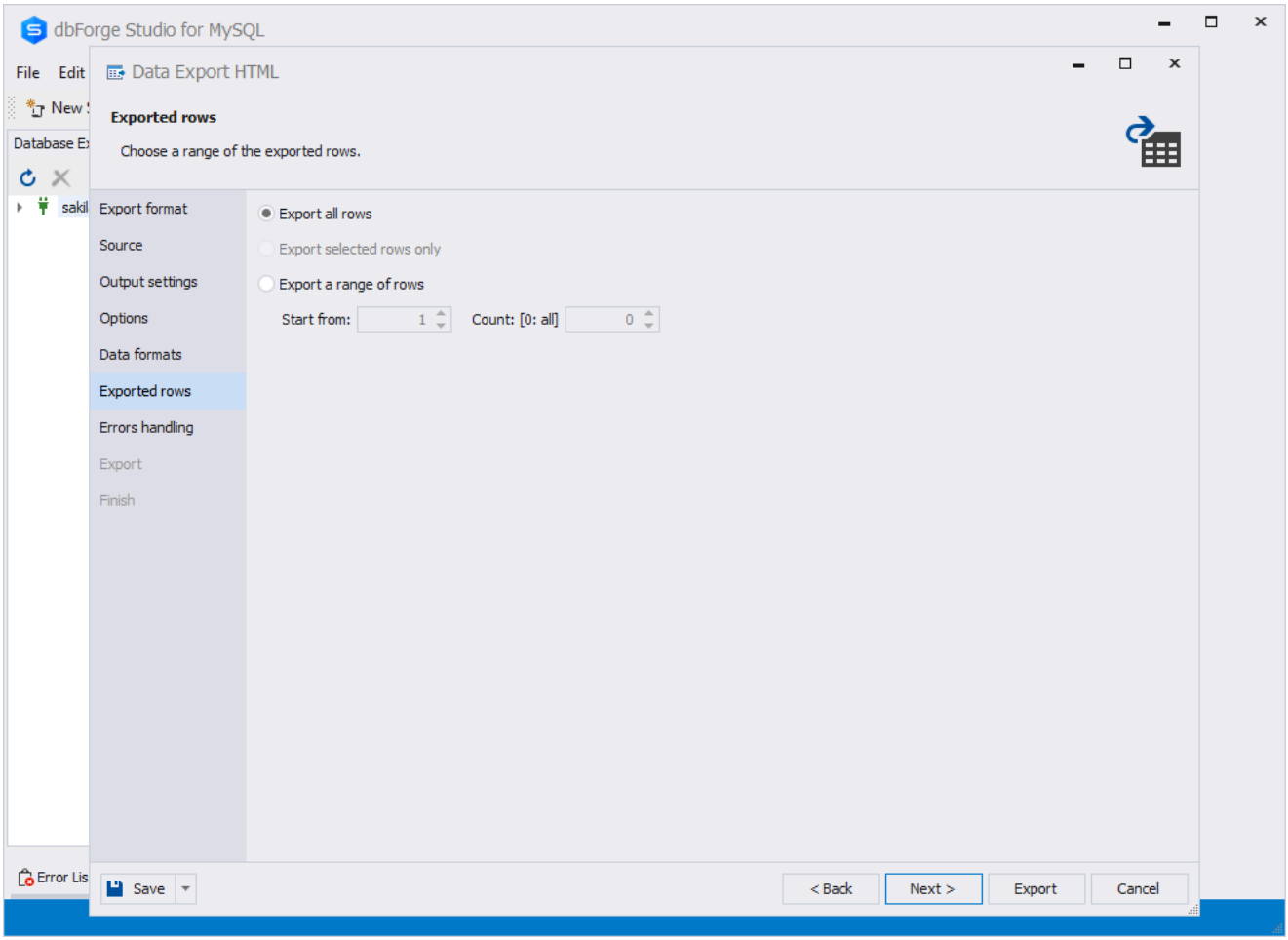
Then, on the **Formats** tab, you can adjust the default format settings for Date, Time, Date Time, Currency, Float, Integer, Boolean, Null String, as well as select the required binary encoding.



e6

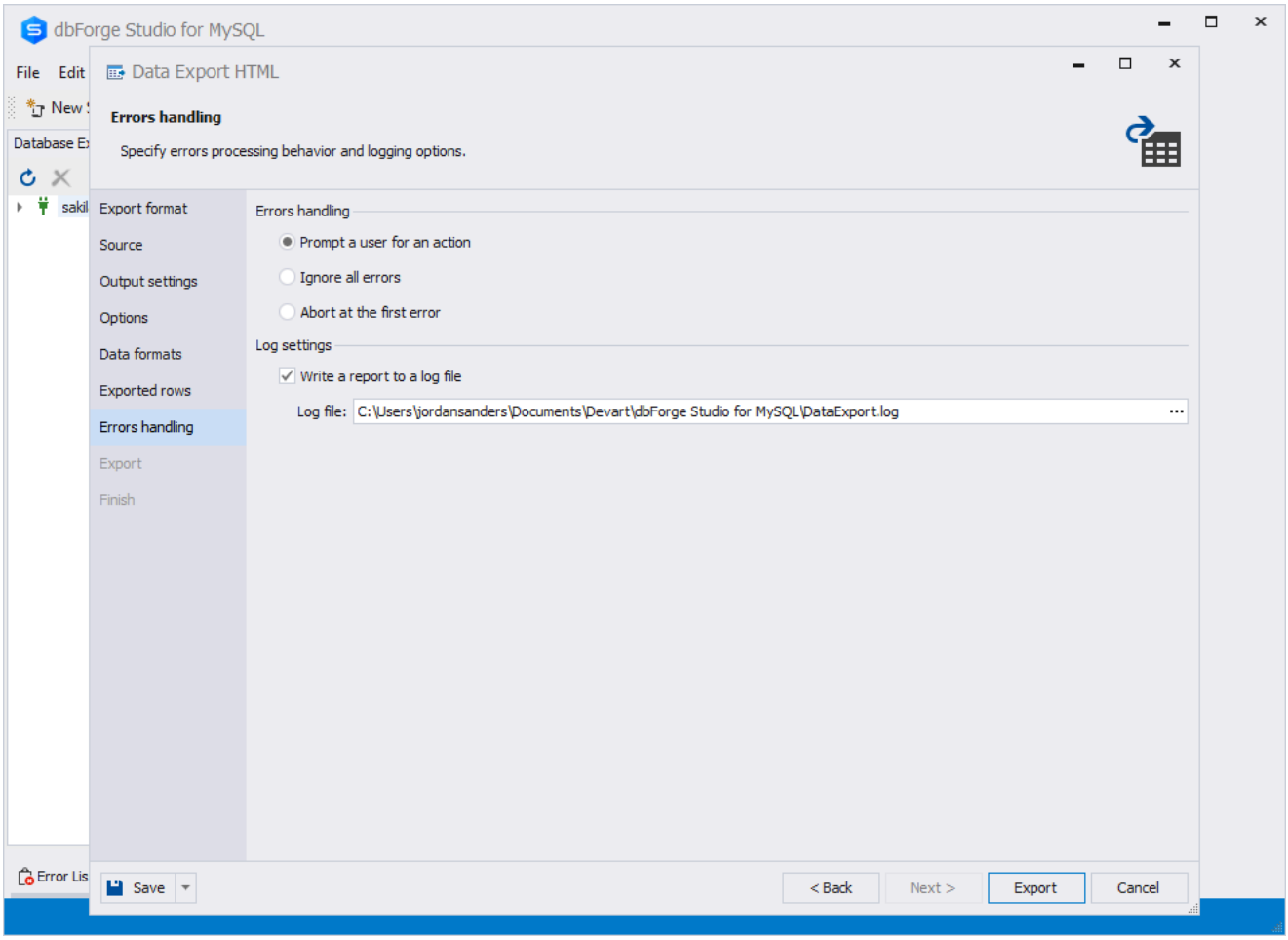
Once you make sure everything is correct, click **Next**.

7. On the **Exported rows** page, select to export all rows or define a certain range of rows, and then click *Next*.**



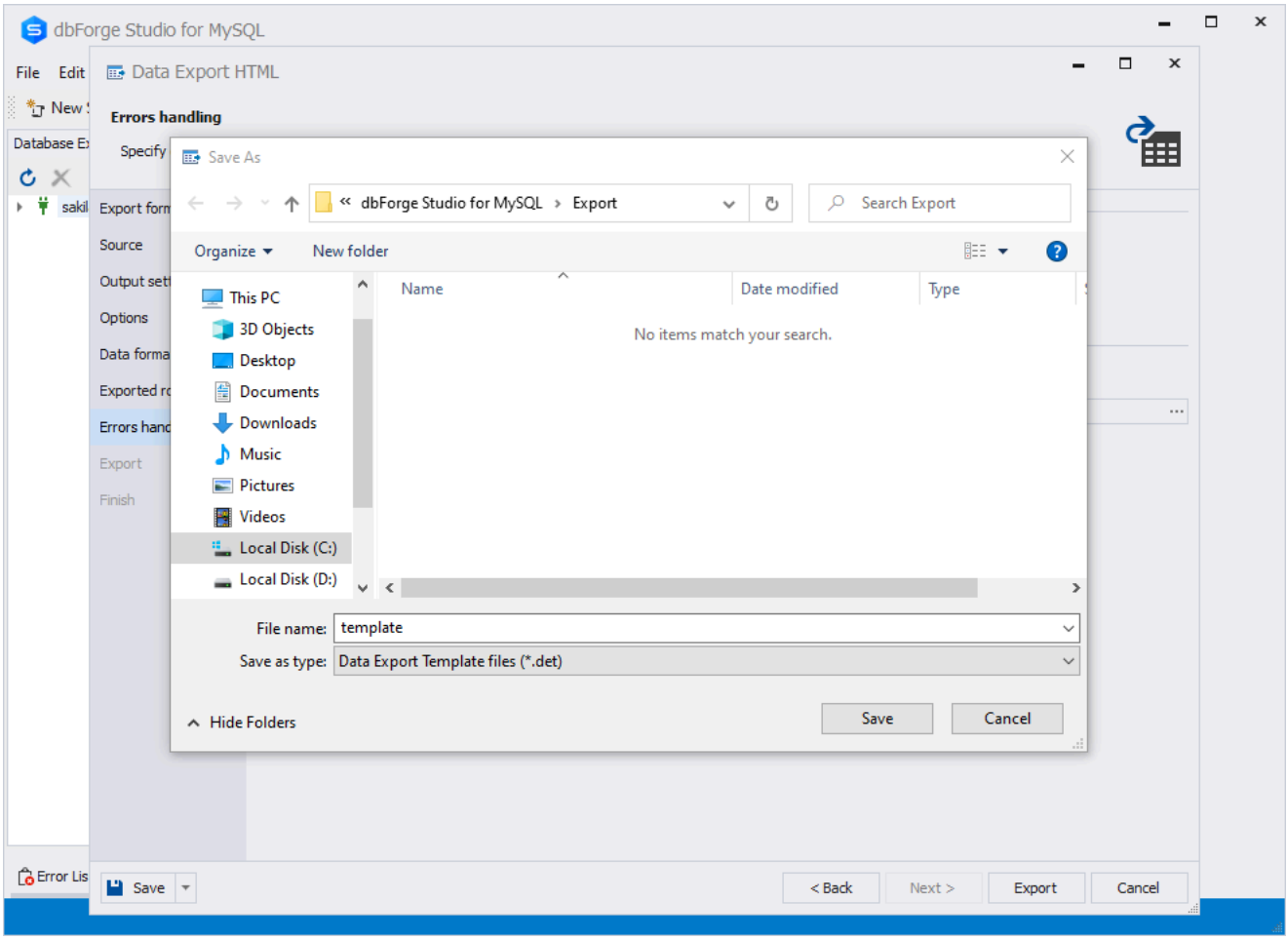
e7

8. On the **Errors handling** page, configure the errors handling behavior and select to keep a log file, if necessary.



e8

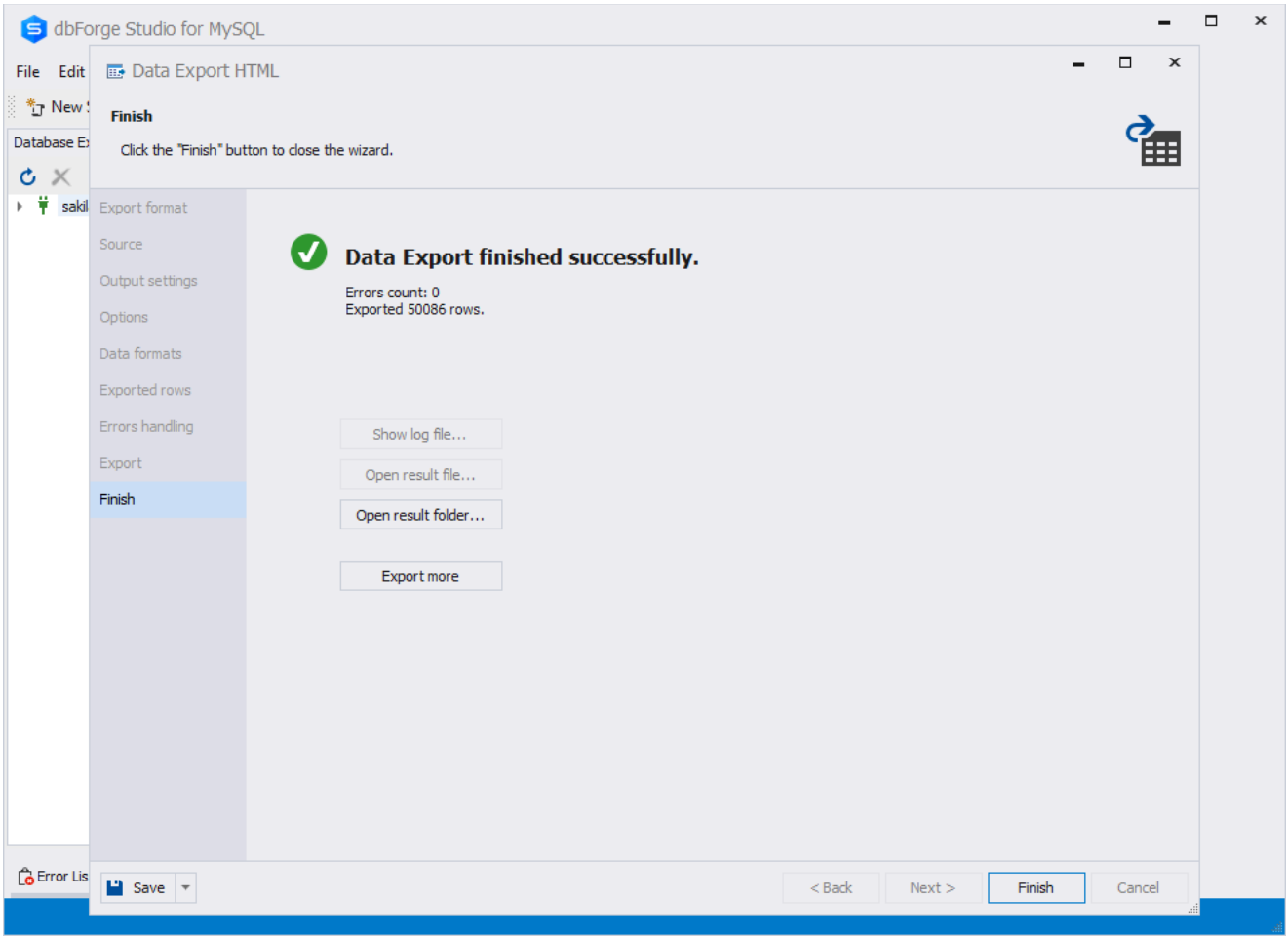
But before you click **Export**, note that you can save templates with your settings for recurring export operations. To do that, click **Save** in the lower left corner of the wizard, specify a name and a destination for the template file to be saved to, and then click **Save**.



e9

Also note that you don’t have to go through every wizard page to click **Export**. You can do it whenever you’ve finished configuring your settings.

9. Finally, after you click **Export**, watch the progress and click **Finish** upon completion.



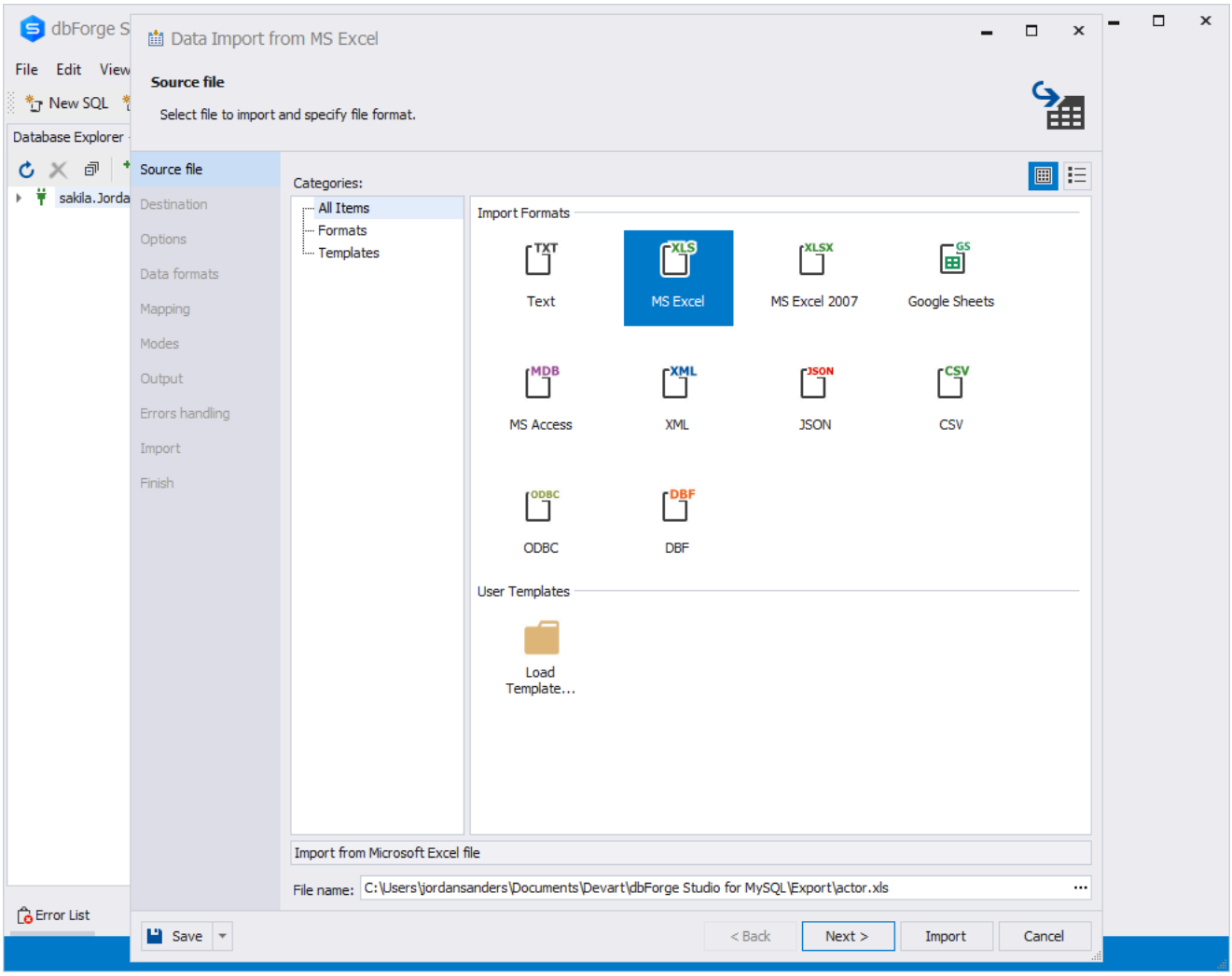
e10

Done! Now, if you want, you can open the folder with the output file right away.

Import data into MariaDB

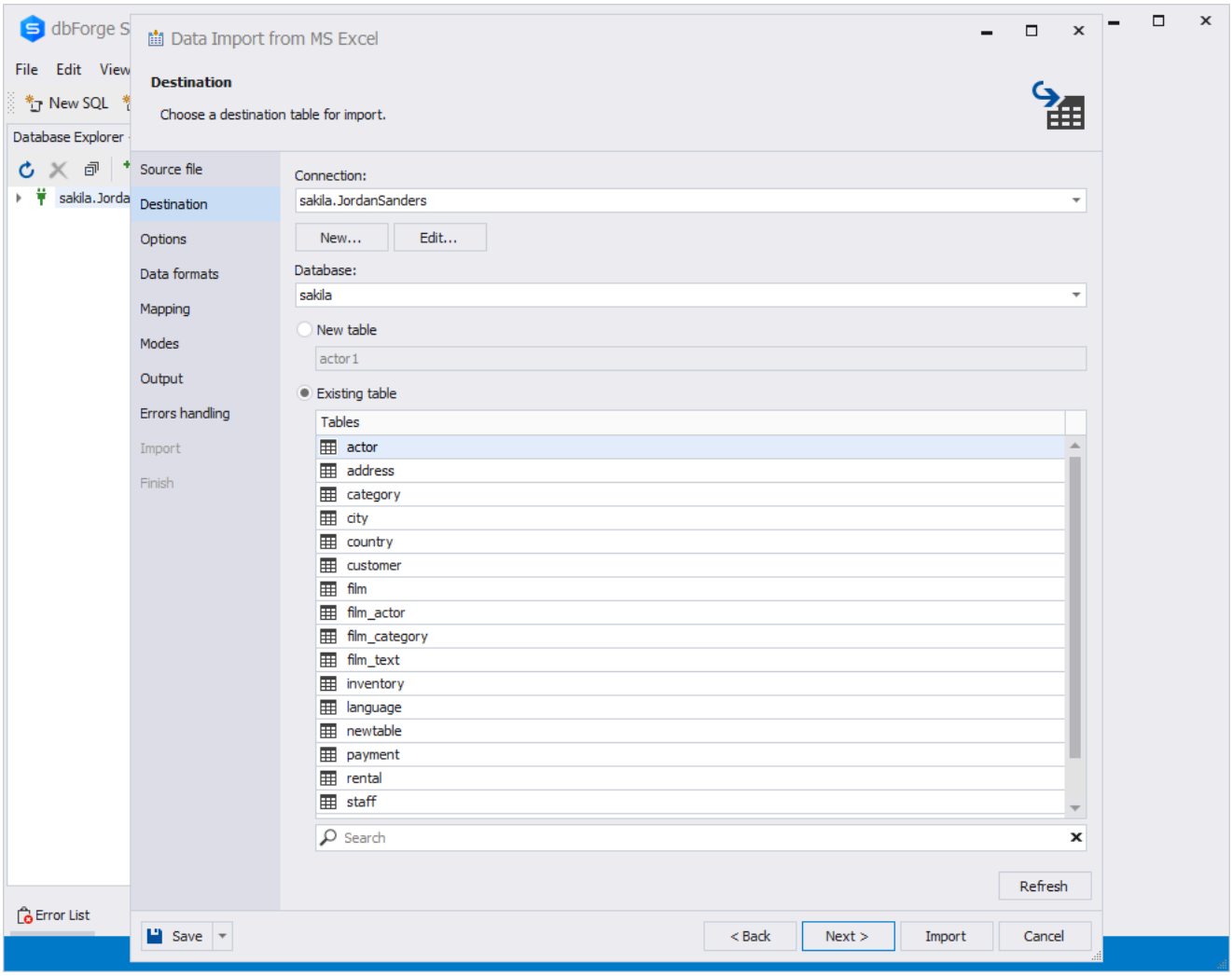
dbForge Studio supports 10 data formats for import, including TXT, XLS, XLSX, MDB, XML, JSON, CSV, ODBC, DBF, and Google Sheets. Just like with export, you have a helpful wizard at hand, whose pages may have differences, depending on the format. And let's pick a different format this time, say, the Microsoft Excel format (XLS).

1. To open the wizard, on the **Database** menu, click **Import Data**.
2. On the **Source file** page, choose the required format, select the file to import data from, and click **Next**.



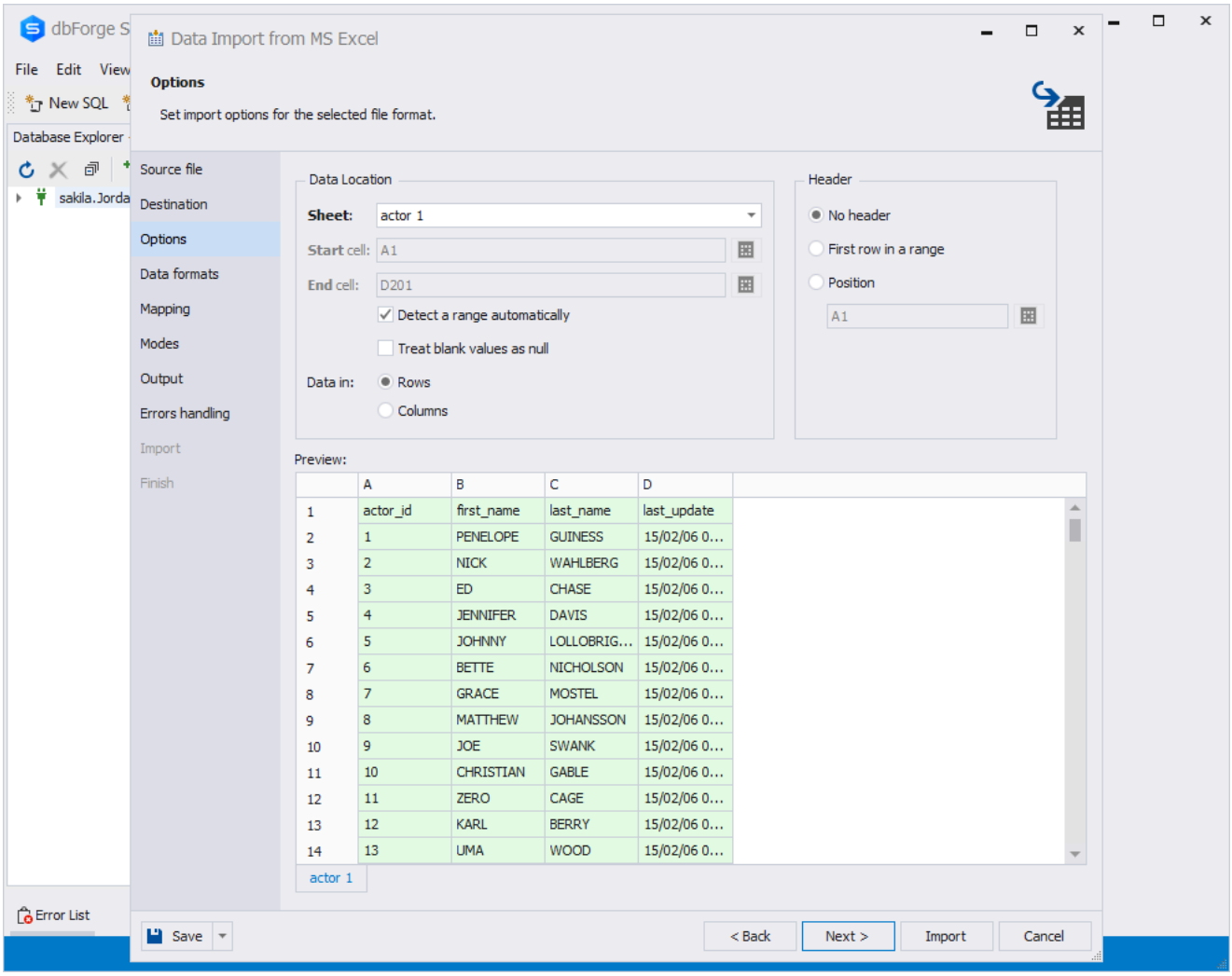
i1

3. On the **Destination** page, select the target connection and database. Then you can select to import data either to a new table or to an existing table. Click **Next**.



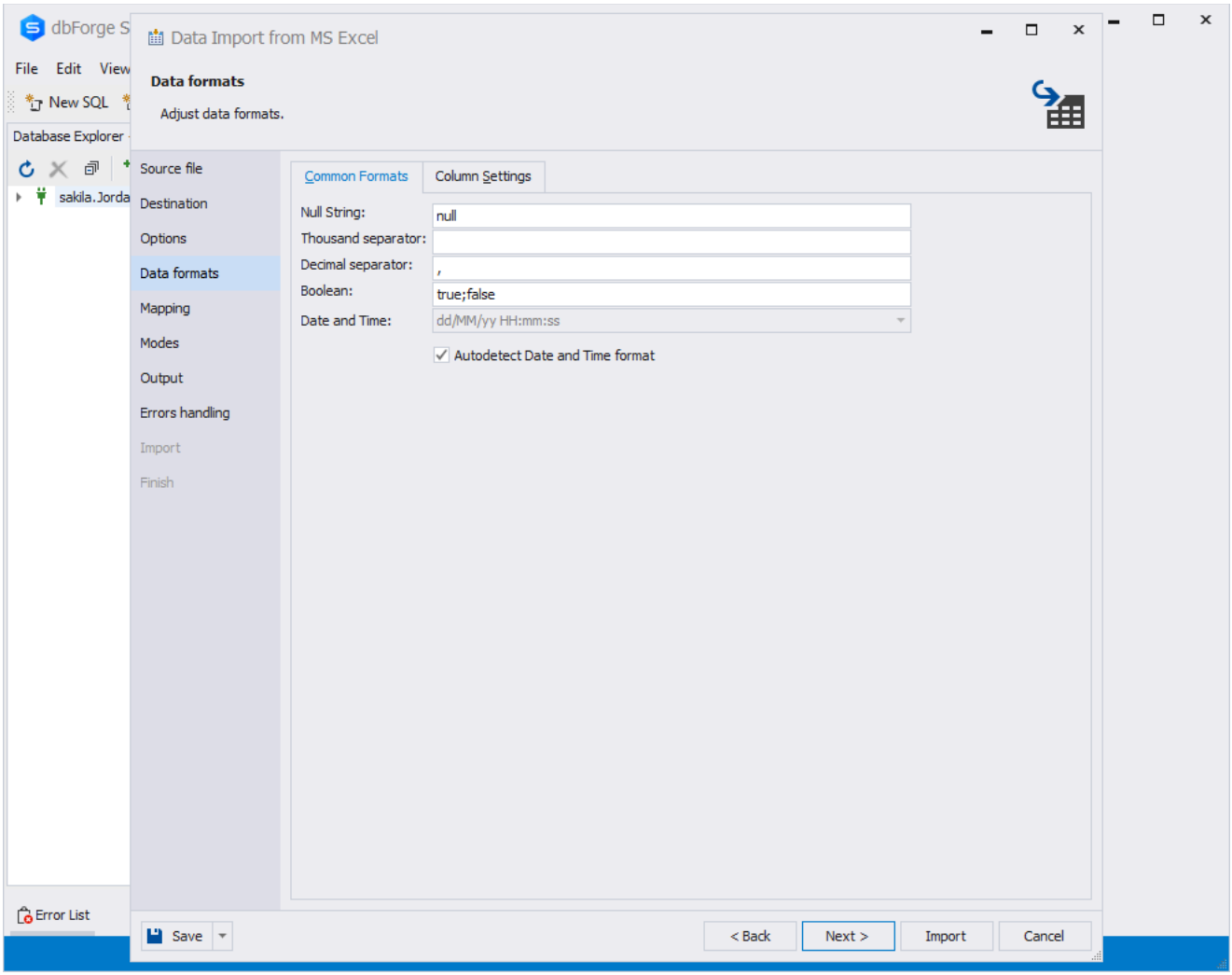
i2

4. On the **Options** page, configure and preview table grid options for imported data. Click **Next**.



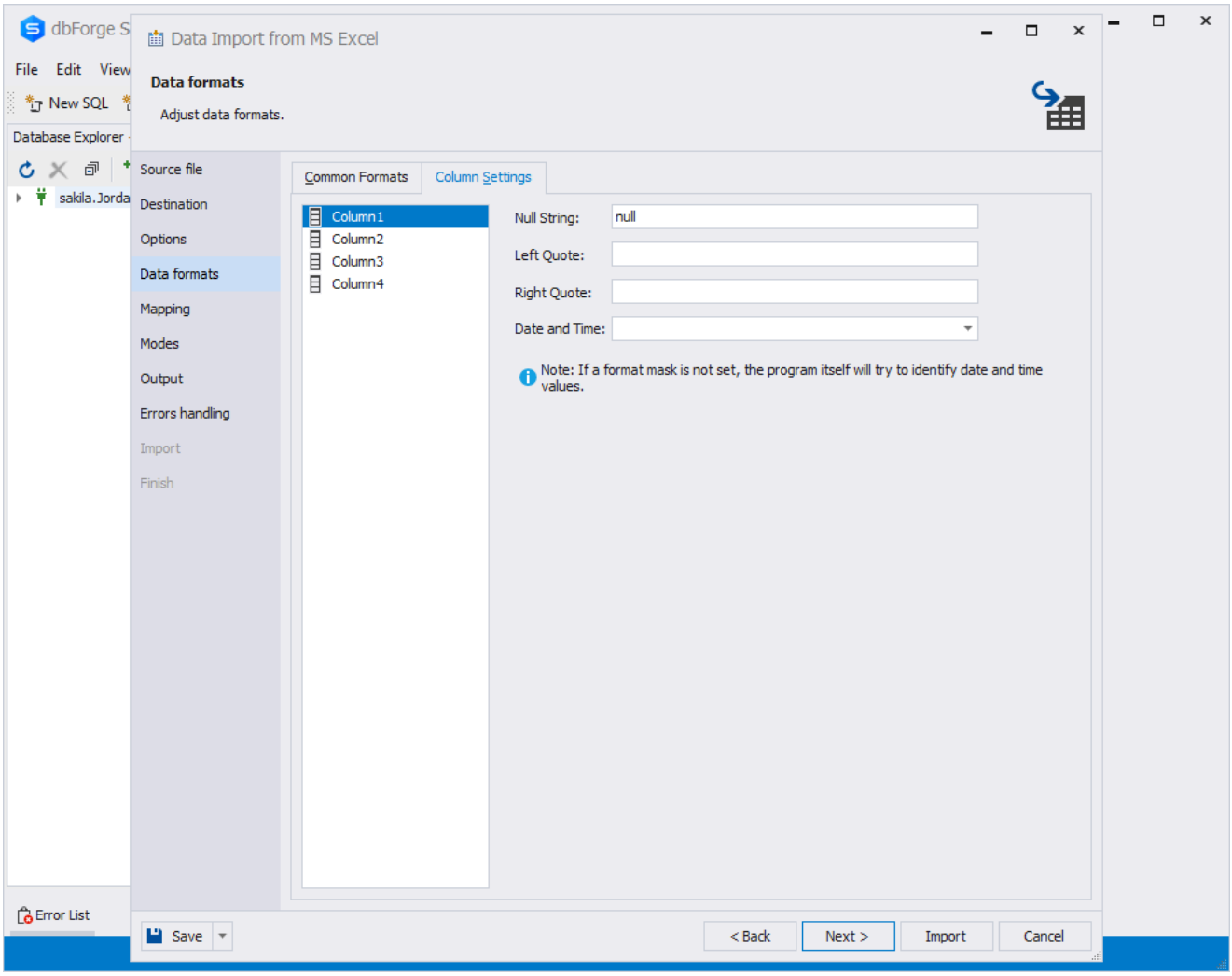
i3

5. On the **Data formats** page, you have two tabs. The first tab is called **Common Formats**, where you can specify the required formats for null strings, thousand and decimal separators, boolean variables, date and time.



i4

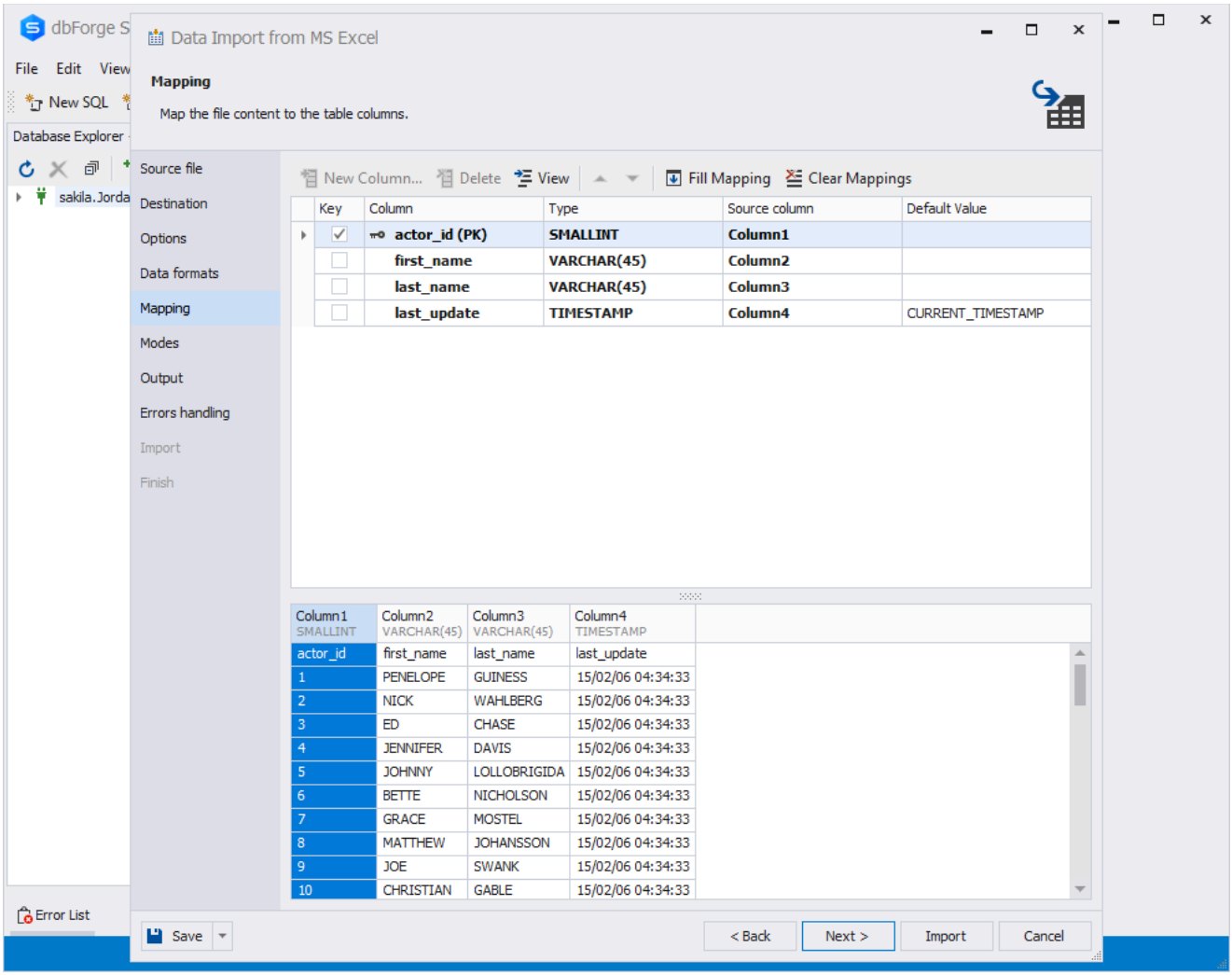
The second tab is called **Column Settings**, where you can configure format settings for separate columns.



i5

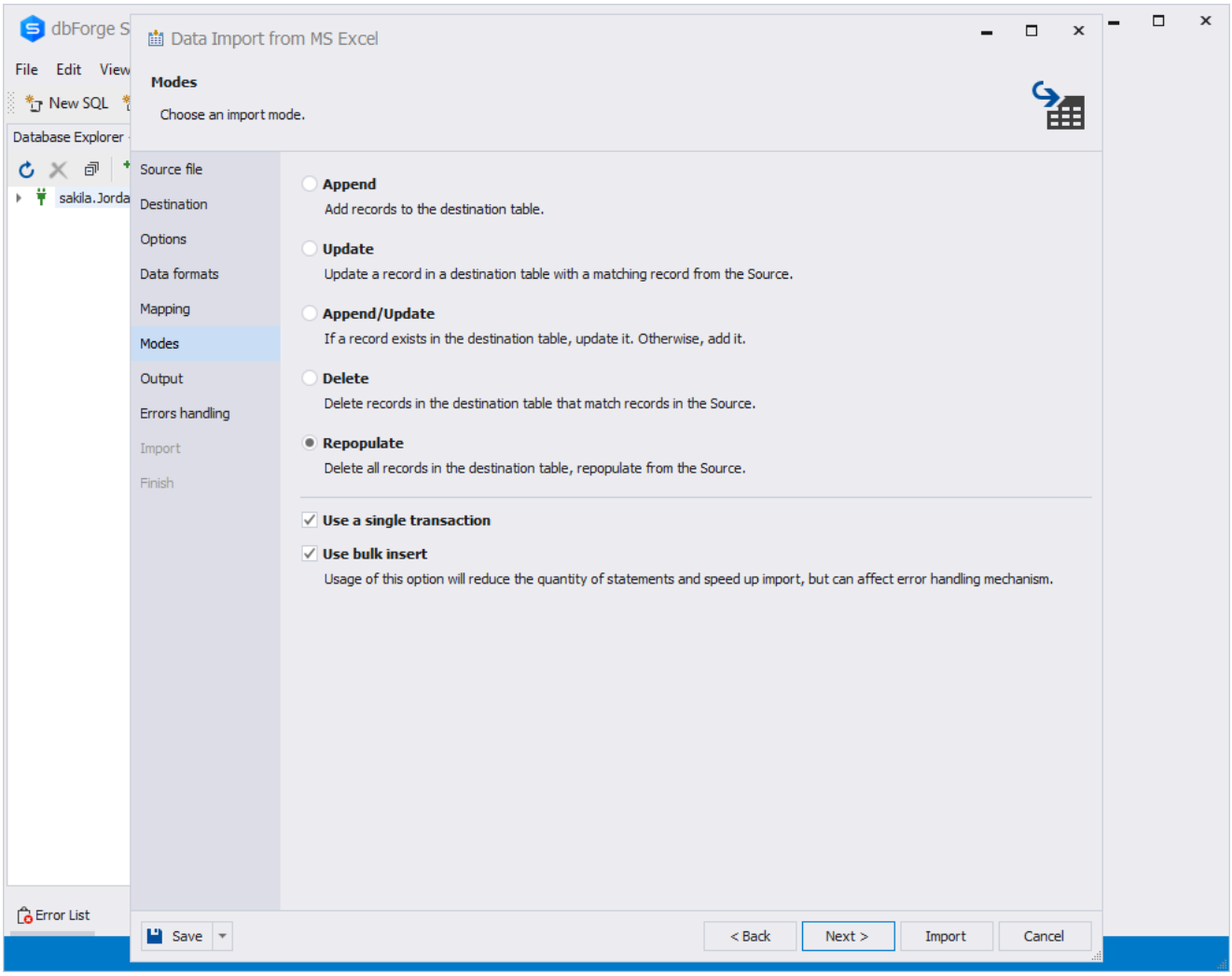
Once you make sure everything is correct, click **Next**.

6. On the **Mapping** page, you can map the source columns to the target ones and preview the results. If you're importing data into a new table, the Studio will automatically create and map all the columns, so you will only have to make adjustments if you wish. Then click **Next**.



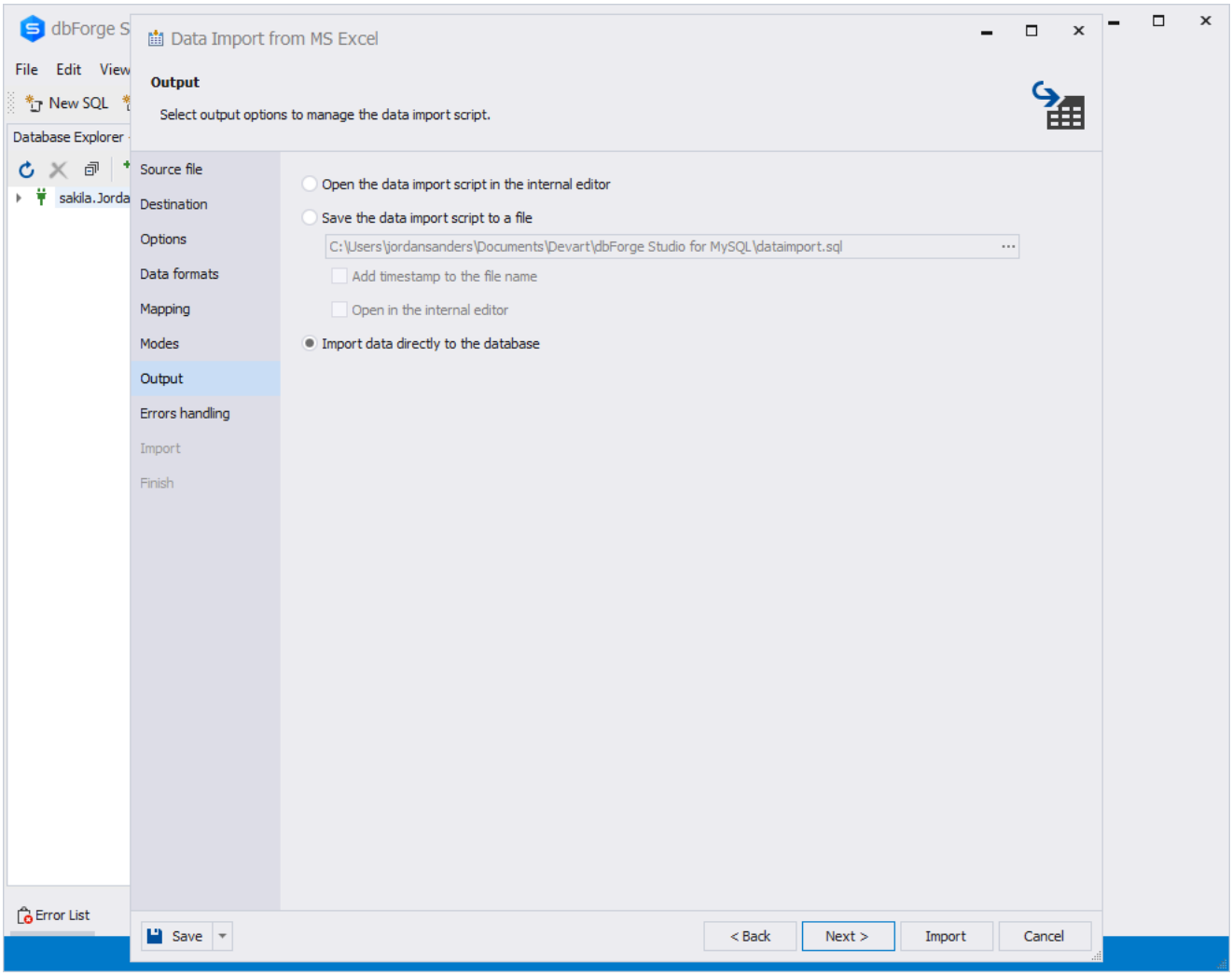
i6

7. On the **Modes** page, select one of the 5 available import modes and click **Next**.



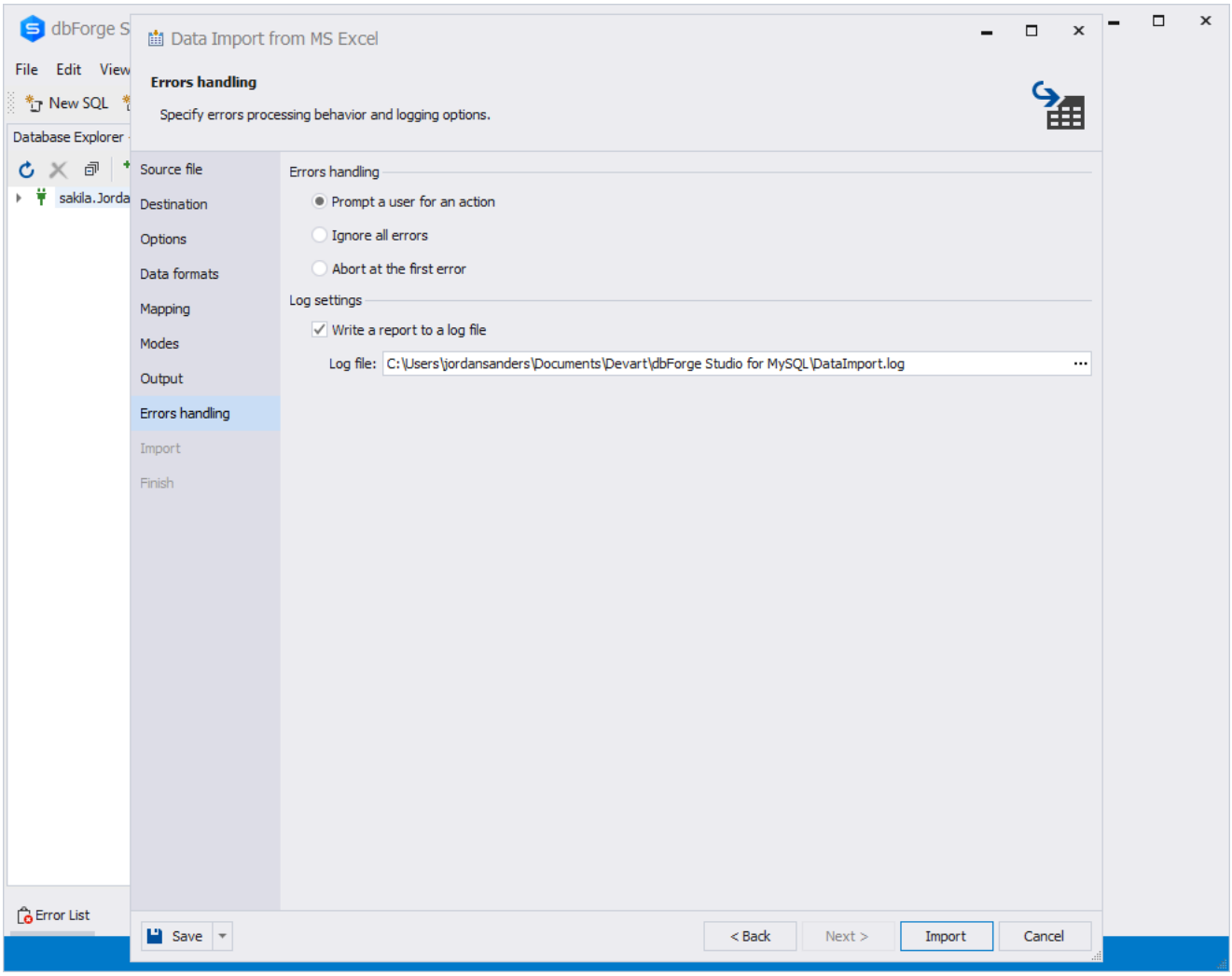
i7

8. On the **Output** page, select the preferred output option and click **Next**.



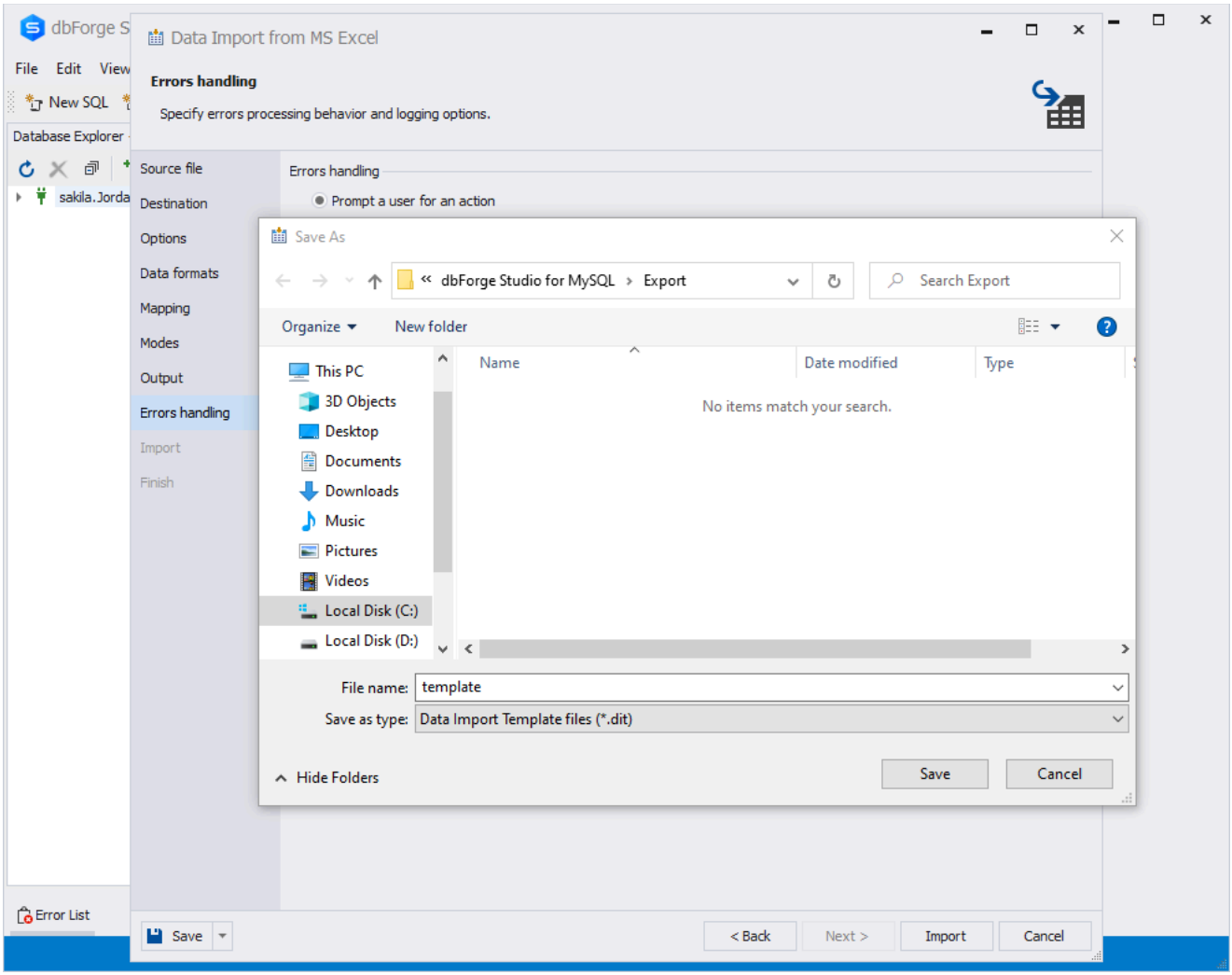
i8

9. On the **Errors handling** page, configure the errors handling behavior and select to keep a log file, if necessary.



i9

Similarly to export, you can save templates with your settings for recurring import operations. To do that, click **Save** in the lower left corner of the wizard, specify a name and a destination for the template file to be saved to, and then click **Save**.



i10

Also note that you don't have to go through every wizard page to click **Import**. You can do it whenever you've finished configuring your settings.

10. After you click **Import**, wait for the process to be completed. Then click **Finish** to confirm the successful completion, and check the results if you wish. That's it!

You can learn more about this functionality on the dedicated data export/import page. Please note: while the page focuses on MySQL databases, everything that's described there is just as perfectly applicable to MariaDB from the same Studio with the same workflow.

There is much more to dbForge Studio when it comes to MariaDB development and management. You can have a brief overview of its features and capabilities on the Features page.

That said, if you'd love to have a single IDE that doesn't need any 3rd-party extensions because it can perfectly deal with nearly any task on its own, feel free to download dbForge Studio for a free 30-day trial and give it a go in your daily work.

This page is licensed: CC BY-SA / Gnu FDL

Deployment Tools

See mariadb-install-db.

Definitive mariadb-install-db guide: initialize data directory, –user/–datadir syntax, –auth-root-authentication-method=socket, and –skip-test-db.

See mariadb-upgrade.

Complete mariadb-upgrade reference: update mysql system tables, CHECK TABLE FOR UPGRADE, ALTER TABLE FORCE, and –upgrade-system-tables option.

See mariadb-secure-installation.

Complete MariaDB installation guide. Complete setup instructions for Linux, Windows, and macOS with configuration and verification for production use.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-install-db

> **Warning:**
>
> **This page is for the `mariadb-install-db` script for Linux/Unix only.**
>
> For the Windows-specific tool of similar name and purpose, see Installing System Tables on Windows.
>
> The Windows version shares the common theme (creating system tables), yet has a lot of functionality specific to Windows systems, for example creating a Windows service. The Windows version does *not* share command line parameters with the Unix shell script.

`mariadb-install-db` initializes the MariaDB data directory and creates the system tables in the mysql database, if they do not exist.

> **Note:**
>
> Previously, the client was called `mysql_install_db` . It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

MariaDB uses these tables to manage privileges, roles, and plugins. It also uses them to provide the data for the help command in the mariadb client.

`mariadb-install-db` works by starting MariaDB Server's `mariadbd` process in –bootstrap mode and sending commands to create the system tables and their content.

Usage

Invoke `mariadb-install-db` using the following syntax:

```
$ mariadb-install-db [options]
```

Because the MariaDB server (`mariadbd`) needs to access the data directory when it runs, you should either run `mariadb-install-db` from the same account that is used for running `mariadbd` , or run it as `root` and use the `--user` option to indicate the username that `mariadbd` runs as. It might be necessary to specify other options such as `--basedir` or `--datadir` if `mariadb-install-db` does not use the correct locations for the installation directory or data directory. Here is a typical invocation:

```
$ scripts/mariadb-install-db --user=mysql \
   --basedir=/opt/mysql/mysql \
   --datadir=/opt/mysql/mysql/data
```

Options

`mariadb-install-db` supports the following options:

–auth-root-authentication-method={ `normal` | `socket` }

If set to `normal`, it creates a `root@localhost` account that authenticates with the `mysql_native_password` authentication plugin and that has no initial password set, which can be insecure. If set to `socket`, it creates a `root@localhost` account that authenticates with the `unix_socket` authentication plugin. Set to `socket` by default.

–auth-root-socket-user=*username*

Used with `--auth-root-authentication-method=socket`. It specifies the name of the second account to create with SUPER privileges in addition to root, as well as of the system account allowed to access it. Defaults to the value of `--user`.

–basedir=*path*

The *path* to the MariaDB installation directory.

–builddir=*directory*

If using `--srcdir` with out-of-directory builds, you will need to set this to the location of the *build directory* where built files reside.

–catalogs=["*list*"]

Initialize MariaDB for catalogs. Argument is a *list of the catalogs to create*, separated with space or comma. The def catalog is created automatically. This option is available from MariaDB 11.7.

–catalog-user=*username*

User when adding catalogs to running server. This option is available from MariaDB 11.7.

–catalog-password[=*password*]

Password for catalog-user. This option is available from MariaDB 11.7.

–catalog-client-arg=*argument*

Other *arguments* to `mariadb` when adding new catalogs. This option is available from MariaDB 11.7.

–cross-bootstrap

For internal use. Used when building the MariaDB system tables on a different host than the target.

–datadir=*path*, –ldata=*path*

The *path* to the MariaDB data directory.

–client-debug

Write commands to-be executed in `/tmp/mariadb_install_db.log`. This option is available from MariaDB 11.6.

–server-debug

Start mariadbd (server) with `--debug`.

–defaults-extra-file=*file*

Read this *file* after the global files are read. Must be given as the first option.

–defaults-file=*file*

Only read default options from the given *file* name. Must be given as the first option.

–defaults-group-suffix=*suffix*

In addition to the given groups, read also groups with this *suffix*.

–force

Causes mariadb-install-db to run even if DNS does not work. In that case, grant table entries that normally use host names will use IP addresses.

–no-defaults

Don't read default options from any option file. Must be given as the first option.

–print-defaults

Print the program argument list and exit. Must be given as the first option.

–rpm

For internal use. This option is used by RPM files during the MariaDB installation process.

–skip-name-resolve

Uses IP addresses rather than host names when creating grant table entries. This option can be useful if your DNS does not work.

–skip-test-db

Don't install the test database.

–srcdir=*path*

For internal use. The *path* to the MariaDB source directory. This option uses the compiled binaries and support files within the source tree, useful for if you don't want to install MariaDB yet and just want to create the system tables. The directory under which mariadb-install-db looks for support files such as the error message file and the file for populating the help tables.

–user=*username*

The login *username* to use for running mariadbd. Files and directories created by mariadbd is owned by this user. You must be root to use this option. By default, mariadbd runs using your current login name and files and directories that it creates is owned by you.

–verbose

Verbose mode. Print more information about what the program does.

–windows

For internal use. This option is used for creating Windows distributions.

Option Files

In addition to reading options from the command line, `mariadb-install-db` can also read options from option files. If an unknown option is provided to `mariadb-install-db` in an option file, then it is ignored.

The following options relate to how MariaDB command line tools handles option files. They must be given as the first argument on the command line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

Option Groups

`mariadb-install-db` reads options from the following option groups from option files:

Group	Description
[mysql_install_db]	Options read by mysqld_safe, which includes both MariaDB Server and MySQL Server.

`mariadb-install-db` also reads options from the following server option groups from option files:

Group	Description
[mysqld]	Options read by mysqld, which includes both MariaDB Server and MySQL Server.
[server]	Options read by MariaDB Server.
[mysqld-X.Y]	Options read by a specific version of mysqld, which includes both MariaDB Server and MySQL Server. For example, [mysqld-5.5].
[mariadb]	Options read by MariaDB Server.
[mariadb-X.Y]	Options read by a specific version of MariaDB Server.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[galera]	Options read by a galera-capable MariaDB Server. Available on systems compiled with Galera support.

Installing System Tables

Installing System Tables From a Source Tree

If you have just compiled MariaDB from source, and if you want to use `mariadb-install-db` from your source tree, then that can be done without having to actually install MariaDB. This is very useful if you want to test your changes to MariaDB without disturbing any existing installations of MariaDB.

To do so, you would have to provide the `--srcdir` option. For example:

```
./scripts/mariadb-install-db --srcdir=. --datadir=path-to-temporary-data-dir
```

Installing System Tables From a Binary Tarball

If you install a binary tarball package in a non standard path, like your home directory, and if you already have a MariaDB / MySQL package installed, then you may get conflicts with the default `/etc/my.cnf`. This often results in permissions errors.

One possible solution is to use the `--no-defaults` option, so that it does not read any option files. For example:

```
./scripts/mariadb-install-db --no-defaults --basedir=. --datadir=data
```

Another possible solution is to use the `defaults-file` option, so that you can specify your own option file. For example:

```
./scripts/mariadb-install-db --defaults-file=~/.my.cnf
```

User Accounts Created by Default

`mariadb-install-db` sets `--auth-root-authentication-method=socket` by default. When this is set, the default `root@localhost` user account is created with the ability to use two authentication plugins:

- First, it is configured to try to use the unix_socket authentication plugin. This allows the `root@localhost` user to login without a password via the local Unix socket file defined by the socket system variable, as long as the login is attempted from a process owned by the operating system `root` user account.
- Second, if authentication fails with the unix_socket authentication plugin, then it is configured to try to use the mysql_native_password authentication plugin.

The definition of the default `root@localhost` user account is:

```
CREATE USER 'root'@'localhost' IDENTIFIED VIA unix_socket
  OR mysql_native_password USING 'invalid';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION;
GRANT PROXY ON ''@'%' TO 'root'@'localhost' WITH GRANT OPTION;
```

Since `mariadb-install-db` sets `--auth-root-authentication-method=socket` by default, the following additional user accounts are **not** created by default:

- `root@127.0.0.1`
- `root@::1`
- `root@${current_hostname}`

However, an additional user account that is defined by the `--auth-root-socket-user` option is created. If this option is not set, then the value defaults to the value of the `--user` option. On most systems, the `--user` option will use the value of `mysql` by default, so this additional user account would be called `mysql@localhost`.

The definition of this `mysql@localhost` user account is similar to the `root@localhost` user account:

```
CREATE USER 'mysql'@'localhost' IDENTIFIED VIA unix_socket
  OR mysql_native_password USING 'invalid';
GRANT ALL PRIVILEGES ON *.* TO 'mysql'@'localhost' WITH GRANT OPTION;
```

An invalid password is initially set for both of these user accounts. This means that before a password can be used to authenticate as either of these user accounts, the accounts must first be given a valid password by executing the SET PASSWORD statement.

For example, here is an example of setting the password for the `root@localhost` user account immediately after installation:

```
$ sudo yum install MariaDB-server
$ sudo systemctl start mariadb
$ sudo mariadb
...
MariaDB> SET PASSWORD = PASSWORD('XH4VmT3_jt');
```

You may notice in the above example that the mariadb command-line client is executed via sudo. This allows the `root@localhost` user account to successfully authenticate via the unix_socket authentication plugin.

Troubleshooting Issues

Checking the Error Log

If `mariadb-install-db` fails, you should examine the error log in the data directory, which is the directory specified with `--datadir` option. This should provide a clue about what went wrong.

Testing With mariadbd

You can also test that this is not a general fault of MariaDB Server by trying to start the `mariadbd` process. The `--skip-grant-tables` option will tell it to ignore the system tables. Enabling the general query log can help you determine what queries are being run on the server. For example:

```
mariadbd --skip-grant-tables --general-log
```

At this point, you can use the mariadb client to connect to the mysql database and look at the system tables. For example:

```
$ /usr/local/mysql/bin/mysql -u root mysql
MariaDB [mysql]> show tables
```

Using a Server Compiled With –disable-grant-options

The following only apply in the exceptional case that you are using a mariadbd server which is configured with the `--disable-grant-options` option:

`mariadb-install-db` needs to invoke `mariadbd` with the `--bootstrap` and `--skip-grant-tables` options.
A MariaDB configured with the `--disable-grant-options` option has `--bootstrap` and `--skip-grant-tables` disabled. To handle this case, set the `MYSQLD_BOOTSTRAP` environment variable to the full path name of a mariadbd server that is configured without `--disable-grant-options`. `mariadb-install-db` will use that server.

The test and test_% Databases

When calling the `mariadb-install-db` script, a new folder called `test` is created in the data directory.
It only has the single `db.opt` file, which sets the client options `default-character-set` and `default-collation` only.

If you run `mysql` as an anonymous user, `mysql -u''@localhost`, and look for the grants and databases you are able to work with, you will get the following:

```
SELECT current_user;
+--------------+
| current_user |
+--------------+
| @localhost   |
+--------------+

SHOW GRANTS FOR current_user;
+---------------------------------------+
| Grants for @localhost                 |
+---------------------------------------+
| GRANT USAGE ON *.* TO ``@`localhost` |
+---------------------------------------+

SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
+--------------------+
```

Shown are the `information_schema` as well as `test` databases that are built in databases.
But looking from SHOW GRANTS appears to be a paradox; how can the current user see something if they don't have privileges for that?

Let's go a step further.
Now, use the `root` / `unix` user, which has all rights, in order to create a new database with the prefix `test_` , something like:

```
CREATE DATABASE test_electricity;
```

With the above change, a new directory is created in the data directory.
Now login again with the anonymous user and run SHOW DATABASES:

```
SHOW DATABASES
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
| test_electricity   |
+--------------------+
```

Again we are able to see the newly created database, without any rights?
We have an anonymous user that has no privileges, but still can see the `test` and `test_electricity` databases.
Where does this come from?

Log in with the `root` / `unix` user to find out all privileges that the anonymous user has:

```
SELECT * FROM mysql.user WHERE user='' AND host='localhost'\G
*************************** 1. row ***************************
                 Host: localhost
                 User:
             Password:
           Select_priv: N
           Insert_priv: N
           Update_priv: N
           Delete_priv: N
           Create_priv: N
             Drop_priv: N
           Reload_priv: N
         Shutdown_priv: N
          Process_priv: N
             File_priv: N
            Grant_priv: N
       References_priv: N
            Index_priv: N
            Alter_priv: N
          Show_db_priv: N
            Super_priv: N
 Create_tmp_table_priv: N
      Lock_tables_priv: N
          Execute_priv: N
       Repl_slave_priv: N
      Repl_client_priv: N
      Create_view_priv: N
        Show_view_priv: N
   Create_routine_priv: N
    Alter_routine_priv: N
      Create_user_priv: N
            Event_priv: N
          Trigger_priv: N
Create_tablespace_priv: N
   Delete_history_priv: N
              ssl_type:
            ssl_cipher:
           x509_issuer:
          x509_subject:
         max_questions: 0
           max_updates: 0
       max_connections: 0
  max_user_connections: 0
                plugin:
 authentication_string:
      password_expired: N
               is_role: N
          default_role:
    max_statement_time: 0.000000
```

As seen above from the mysql.user table, the anonymous user doesn't have any global privileges.
Still, the anonymous user can see databases, so there must be a way so that anonymous user can see the `test` and `test_electricity` databases.

Let's check for grants on the database level. That information can be found in the mysql.db table.
Looking at the `mysql.db` table, it already contains 2 rows created when the `mariadb-install-db` script was invoked.

The anonymous user has database privileges (without `grant`, `alter_routine` and `execute`) on `test` and `test_%` databases:

```
SELECT * FROM mysql.db\G
*************************** 1. row ***************************
                 Host: %
                   Db: test
                 User:
          Select_priv: Y
          Insert_priv: Y
          Update_priv: Y
          Delete_priv: Y
          Create_priv: Y
            Drop_priv: Y
           Grant_priv: N
      References_priv: Y
           Index_priv: Y
           Alter_priv: Y
Create_tmp_table_priv: Y
     Lock_tables_priv: Y
     Create_view_priv: Y
       Show_view_priv: Y
  Create_routine_priv: Y
   Alter_routine_priv: N
         Execute_priv: N
           Event_priv: Y
         Trigger_priv: Y
  Delete_history_priv: Y
*************************** 2. row ***************************
                 Host: %
                   Db: test\_%
                 User:
          Select_priv: Y
          Insert_priv: Y
          Update_priv: Y
          Delete_priv: Y
          Create_priv: Y
            Drop_priv: Y
           Grant_priv: N
      References_priv: Y
           Index_priv: Y
           Alter_priv: Y
Create_tmp_table_priv: Y
     Lock_tables_priv: Y
     Create_view_priv: Y
       Show_view_priv: Y
  Create_routine_priv: Y
   Alter_routine_priv: N
         Execute_priv: N
           Event_priv: Y
         Trigger_priv: Y
  Delete_history_priv: Y
```

The first row is reserved for explicit usage for the `test` database, which is automatically created with `mariadb-install-db`.

Since database `test_electricity` satisfies the `test_%` pattern where `test_` is a prefix, we can understand why the user has the right to work with the newly-created database.

As long as records in `mysql.db` for the anonymous user exists, each new user created will have the privileges for the `test` and `test_%` databases.

Other databases privileges **are not automatically granted** for the newly created user. We have to grant privileges, which is visible in `mysql.db` table.

Not Creating the test Database and Anonymous User

If you run `mariadb-install-db` with the `--skip-test-db` option, no `test` database is created, which we can see as follows:

```
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+

SELECT * FROM mysql.db;
Empty set (0.001 sec)
```

Also, no anonymous user is created (only `unix` / `mariadb.sys` / `root` users):

```
SELECT user,host FROM mysql.user;
+-------------+-----------+
| User        | Host      |
+-------------+-----------+
| anel        | localhost |
| mariadb.sys | localhost |
| root        | localhost |
+-------------+-----------+
```

See Also

- Configure MariaDB with catalog support
- Installing system tables (mariadb-install-db)
- The Windows version of `mariadb-install-db` : mysql_install_db.exe

This page is licensed: CC BY-SA / Gnu FDL

mariadb-upgrade

`mariadb-upgrade` is a tool that checks and updates your tables to the latest version.

> **Note:**
>
> Previously, the client was called `mysql_upgrade` . It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Overview

You should run `mariadb-upgrade` after upgrading from one major MariaDB release to another, such as from MariaDB 10.11 to MariaDB 11.4. You also have to use `mariadb-upgrade` after migrating to MariaDB from MySQL. It's also safe to run `mariadb-upgrade` for minor upgrades, as, if there are no incompatibilities, nothing is changed.

> **CURRENT**
>
> `mariadb-upgrade` needs to be run as a user with write access to the data directory.
>
> Starting from mariadb-upgrade 2.0, the user running the upgrade tool must have write access to `datadir/mysql_upgrade_info` , so that the tool can write the current MariaDB version into the file.
>
> **< 10.7.2 / 10.6.6 / 10.5.14**

> `mariadb-upgrade` needs to be run as a user with write access to the data directory.

`mariadb-upgrade` is run after starting the new MariaDB server. Running it before you shut down the old version will not hurt anything and will allow you to make sure it works and figure out authentication for it ahead of time.

> **Note:**
>
> The MariaDB server never performs the upgrade itself. On startup, if it detects an out-of-date system table or a version mismatch, it only writes a message to the error log recommending that you run `mariadb-upgrade`.
>
> On Debian and Ubuntu, the server packages run `mariadb-upgrade` for you automatically after a package upgrade: once the new server has started, the startup script runs it in the background, using `--version-check` so the work is only done once per major version. On RPM-based distributions, and whenever you start the server directly (for example with `systemd` or by running `mariadbd`), `mariadb-upgrade` is not run automatically and you must run it yourself.

> **Tip:**
>
> It is recommended to make a backup of all databases before running `mariadb-upgrade`.

In most cases, `mariadb-upgrade` should just take a few seconds. The main work of `mariadb-upgrade` is to:

- Update the system tables in the `mysql` database to the latest version (normally just add new fields to a few tables).
- Check that all tables are up to date (runs CHECK TABLE table_name FOR UPGRADE). For tables that are not up to date, runs ALTER TABLE table_name FORCE on the table to update it. A table is not up to date in the following cases:
  - The table uses an index for which there has been a collation change (rare).
  - A format change in the storage engine requires an update (very rare).

Why run mariadb-upgrade?

If you skip running `mariadb-upgrade`, issues can arise, including these:

- Errors in the error log that some system tables don't have all needed columns.
- Updates or searches may not find the record they are attempting to update or search for.
- CHECKSUM TABLE may report the wrong checksum for MyISAM or Aria tables.
- The error message "Cannot load from mysql.proc. The table is probably corrupted."

To fix issues like this, run `mariadb-upgrade`, mariadb-check, CHECK TABLE and, if needed, REPAIR TABLE on the faulty table.

Upgrading to a Minor Version, or to Enterprise Server

Starting from mariadb-upgrade 2.0, mysql-upgrade doesn't run when upgrading to a new minor version (for instance, from MariaDB 10.6.3 to 10.6.4).

> **Warning:**
>
> This includes updating from Community Server to Enterprise Server.

In those cases, mariadb-upgrade terminates with a message like this:

```
This installation of MariaDB is already upgraded to 10.11.4-MariaDB.
There is no need to run mysql_upgrade again for 10.11.11-MariaDB.
You can use --force if you still want to run mysql_upgrade.
```

As the message indicates, use the –force option to override this behavior.

> **Warning:**
>
> When upgrading from, say, Community Server 10.11.4 to Enterprise Server 10.11.11, mariadb-upgrade considers that a minor-version upgrade, so you must use the `--force` option to make it run.

Usage

```
mariadb-upgrade [--force] [--user=# --password=#
  --host=hostname --port=# --socket=#
  --protocol=tcp|socket|pipe|memory
  --verbose] [OTHER_OPTIONS]
```

`mariadb-upgrade` is mainly a framework to call mariadb-check. `mariadb-upgrade` works by doing the following operations:

```
# Find out path to datadir
echo "show variables like 'datadir'" | mysql
mariadb-check --no-defaults --check-upgrade --auto-repair --databases mysql
mysql_fix_privilege_tables
mariadb-check --no-defaults --all-databases --fix-db-names --fix-table-names
mariadb-check --no-defaults --check-upgrade --all-databases --auto-repair
```

The connect options given to `mariadb-upgrade` are passed along to mariadb-check and mariadb command-line client.

The `mysql_fix_privilege_tables` script is not called; it's included as part of `mariadb-upgrade` .

If you have a problem with `mariadb-upgrade` , try running it in very verbose mode:

```
mariadb-upgrade --verbose --verbose other-options
```

`mariadb-upgrade` also saves the MariaDB version number in a file named `mysql_upgrade_info` in the data directory. This is used to quickly check whether all tables have been checked for this release so that table-checking can be skipped. For this reason, `mariadb-upgrade` needs to be run as a user with write access to the data directory. To ignore this file and perform the check regardless, use the `--force` option.

> **Warning:**
>
> Because this check relies on the version recorded in `mysql_upgrade_info` , restoring a `mariadb-dump` (`mysqldump`) backup taken from an older server version can leave the restored system tables out of date while the file still shows the current version. In that case, `mariadb-upgrade` reports that no upgrade is needed, so run it with the `--force` option to upgrade the restored tables.

Options

`mariadb-upgrade` supports the following options:

-?, –help

Display this help message and exit.

–basedir=*path*

Old option accepted for backward compatibility but ignored.

–character-sets-dir=*path*

Old option accepted for backward compatibility but ignored.

–check-if-upgrade-is-needed

Do a quick check if upgrade is needed. Returns `0` if an upgrade is needed, `1` if not. Available from mariadb-upgrade 2.0.

–compress=*name*

Old option accepted for backward compatibility but ignored.

–datadir=*name*

Old option accepted for backward compatibility but ignored.

–debug-check

Check memory and open file usage at exit.

-T, –debug-info

Print some debug info at exit.

–default-character-set=*name*

Old option accepted for backward compatibility but ignored.

-f, –force

Force execution of mariadb-check even if mariadb-upgrade has already been executed for the current version of MariaDB. Ignores `mysql_upgrade_info`.

-h, –host=*host*

Connect to MariaDB on the given *host*.

-p, –password=*password*

Password to use when connecting to server. If password is not given, it's solicited on the command line (which should be considered insecure). You can use an option file to avoid giving the password on the command line.

-P, –port=*port-number*

Port number to use for connection or 0 for default to, in order of preference, my.cnf, the `MYSQL_TCP_PORT` environment variable, `/etc/services`, built-in default (`3306`).

–protocol=*protocol*

The *protocol* to use for connection (`tcp`, `socket`, `pipe`, `memory`).

–silent

Print less information.

-S, –socket=*file*

For connections to localhost, the Unix socket *file* to use, or, on Windows, the name of the *named pipe* to use.

–ssl

Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The `--ssl` option does not enable verifying the server certificate by default. In order to verify the server certificate, you must specify the `--ssl-verify-server-cert` option.

–ssl-ca=*path*

Defines a *path* to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option implies the `--ssl` option.

–ssl-capath=*path*

Defines a *path* to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the `--ssl` option.

–ssl-cert=*path*

Defines a *path* to the X509 certificate file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

–ssl-cipher=*ciphers*

List of permitted ciphers or cipher suites to use for TLS. This option implies the `--ssl` option.

–ssl-crl=*path*

Defines a *path* to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL or Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

–ssl-crlpath=*path*

Defines a *path* to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

–ssl-key=*file*

Defines a path to a *private key file* to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

–ssl-verify-server-cert

Enables server certificate verification. This option is disabled by default.

-t, –tmpdir=*directory*

Directory for temporary files.

-s, –upgrade-system-tables

Only upgrade the system tables in the mysql database. Tables in other databases are not checked or touched.

-u, –user=name

User for login if not current user.

-v, –verbose

Display more output about the process, using it twice will print connection arguments; using it 3 times will print out all CHECK, RENAME and ALTER TABLE commands used during the check phase; using it 4 times will also write out all mariadb-check commands used.

-V, –version

Output version information and exit.

-k, –version-check

Run this program only if its 'server version' matches the version of the server to which it's connecting check. Note: the 'server version' of the program is the version of the MariaDB server with which it was built/distributed. (Defaults to on; use `--skip-version-check` to disable.)

–write-binlog

> **CURRENT**
>
> All commands, including those run by mariadb-check, are written to the binary log. Disabled by default.
>
> **< 10.0.6**
>
> All commands, including those run by mariadb-check, are written to the binary log. Enabled by default. The `--skip-write-binlog` option must be used when commands should not be sent to replication replicas.

Option Files

In addition to reading options from the command line, `mariadb-upgrade` can also read options from option files. If an unknown option is provided to `mariadb-upgrade` in an option file, then it is ignored.

The following options relate to how MariaDB command line tools handles option files. They must be given as the first argument on the command line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

`mariadb-upgrade` is linked with MariaDB Connector/C. However, MariaDB Connector/C does not handle the parsing of option files for `mariadb-upgrade`. That is still performed by the server option file parsing code. See MDEV-19035 for more information.

Option Groups

`mariadb-upgrade` reads options from the following option groups from option files:

Group	Description
[mysql_upgrade]	Options read by mariadb-upgrade, which includes both MariaDB Server and MySQL Server.
[mariadb-upgrade]	Options read by mariadb-upgrade. Available starting with MariaDB 10.4.6.
[client]	Options read by all MariaDB and MySQL client programs, which includes both MariaDB and MySQL clients. For example, mysqldump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

Differences Between mysql_upgrade in MariaDB and MySQL

- MariaDB converts long table names properly.
- MariaDB converts InnoDB tables (no need to do a dump/restore or ALTER TABLE).
- MariaDB converts old archive tables to the new 5.1 format.
- `mysql_upgrade --verbose` runs `mariadb-check --verbose`, so that you get more information of what is happening. Running with 3 times –verbose prints out all `CHECK`, `RENAME` and `ALTER TABLE` statements executed.
- The mysql.event table is upgraded live. There is no need to restart the server to use events if the event table has changed.
- More descriptive output.

Speeding up mariadb-upgrade

- If you are sure that all your tables are up to date with the current version, then you can run `mariadb-upgrade --upgrade-system-tables`, which will only fix your system tables in the mysql database to be compatible with the latest version.

The main reason to run `mariadb-upgrade` on all your tables is to allow it to check that:

- There has not been any change in table formats between versions.
- This has not happened since MariaDB 5.1.
- If some of the tables are using an index for which we have changed sort order.
- This has not happened since MariaDB 5.5.

> **Warning:**
>
> If you are sure this applies to your situation, you can just run `mariadb-upgrade` with the `--upgrade-system-tables` option.

Other Uses

- `mariadb-upgrade` recreates any missing tables in the mysql database. It doesn't touch any data in existing tables.

mariadb-upgrade 2.0

> **Note:**
>
> `mariadb-upgrate/mysql_upgrade 2.0` was introduced in MariaDB 10.5.14 / 10.6.6 / 10.7.2.

Previously, the tool first ran the upgrade process and then created the `datadir/mysql_upgrade_info` file. If the file could not be created because of permissions (`mariadb-upgrade` did not have rights to create the file), `mariadb-upgrad` gave an error, but this was often ignored. One effect of not being able to create the `mysql_upgrade_info` file was that every new `mariadb-upgrade` run would have to do a full upgrade check, which can take a while if there are a lot of tables.

`mariadb-upgrade` 2.0 fixes the following issues:

- The `datadir/mysql_upgrade_info` is now created at the start of the upgrade process and locked. This ensures that two `mariadb-upgrade` processes cannot be run in parallel, which can cause deadlocks (MDEV-27068). One side effect of this is that `mariadb-upgrade` has to have write access to `datadir`, which means it has to be run as the user that installed MariaDB, normally 'mysql' or 'root' .
- One can use `mariadb-upgrade --force --force` to force the upgrade to be run, even if there was no version change or if one doesn't have write access to `datadir`. Note that if this option is used, the next `mariadb-upgrade` run will assume that there is a major version change and the upgrade must be done (again).
- The upgrade is done only if there is a major server version change (for instance, from MariaDB 10.5 to 10.6). This avoids unnecessary upgrades.
- New option added: `--check-if-upgrade-is-needed`. If this is used, `mariadb-upgrade` will return `0` if there has been a major version change and you should run `mariadb-upgrade`. If not upgrade is needed, `1` is returned.
- `--verbose` writes more information, including from which version to which version the upgrade is done.
- Better messages when there is no need to run `mariadb-upgrade`.

See Also

- mariadb-check
- CHECK TABLE
- REPAIR TABLE
- Downgrading between Major Versions of MariaDB

This page is licensed: CC BY-SA / Gnu FDL

mariadb-secure-installation

> **Note:**
>
> Note that many of the reasons for the existence of this script no longer apply (and therefore the guidelines in many online tutorials. In particular, from MariaDB 10.4, Unix socket authentication is applied by default, and there is usually no need to create a root password. See Authentication from MariaDB 10.4.

> **Note:**
>
> Previously, the client was called `mysql_secure_installation`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Description

`mariadb-secure-installation` is a shell script available on Unix systems, and enables you to improve the security of your MariaDB installation in the following ways:

- You can set a password for root accounts.
- You can remove root accounts that are accessible from outside the local host.
- You can remove anonymous-user accounts.

- You can remove the test database, which by default can be accessed by anonymous users.

`mariadb-secure-installation` can be invoked without arguments:

```
shell> mariadb-secure-installation
```

The script will prompt you to determine which actions to perform.

```
Example:
localhost:# mariadb-secure-installation
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!
In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password is blank,
so you should just press enter here.
Enter current password for root (enter for none):
OK, successfully used password, moving on...
Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.
You already have a root password set, so you can safely answer 'n'.
Change the root password? [Y/n] n
 ... skipping.
By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.
Remove anonymous users? [Y/n] y
 ... Success!
Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.
Disallow root login remotely? [Y/n] y
 ... Success!
By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.
Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!
Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.
Reload privilege tables now? [Y/n] y
 ... Success!
Cleaning up...
All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.
Thanks for using MariaDB!
```

Options

`mariadb-secure-installation` accepts some options:

Option	Description
–basedir=dir	Base directory.
–no-defaults	Don’t read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.

Other unrecognized options is passed on to the server.

Option Files

In addition to reading options from the command line, `mariadb-secure-installation` can also read options from option files. If an unknown option is provided to `mariadb-secure-installation` in an option file, it is ignored.

The following options relate to how MariaDB command-line tools handle option files. They must be given as the first argument on the command line:

Option	Description
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

Option Groups

`mariadb-secure-installation` reads options from the following option groups from option files:

Group	Description
[client]	Options read by all MariaDB and MySQL client programs, which includes both MariaDB and MySQL clients like mysqldump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

Use With Galera Cluster

> **Warning:**
>
> This script is not completely safe for use with Galera Cluster as it directly manipulates the mysql.user/mysql.global_priv table, which is not propagated by Galera to the other nodes.

You should run this script on the first node in the cluster before adding more nodes.

If you want to run this after the cluster is up and running, you should find alternative ways.

Anyone can vote for this to be fixed at MDEV-10112.

This page is licensed: CC BY-SA / Gnu FDL

Graphical and Enhanced Clients

> **Note:**
>
> The third-party tools described in this section are not developed or maintained by MariaDB and are not included with MariaDB Server. MariaDB doesn't test, validate, or support them. Refer to each tool's own documentation and license terms.

See Adminer.

See AdminNeo.

See Beekeeper Studio.

See SQL Diagnostic Manager & SQLyog.

See Database Workbench.

See dbForge Data Compare.

See dbForge Data Generator.

See dbForge Documenter.

See dbForge Edge.

See dbForge Fusion.

See dbForge Query Builder.

See dbForge Schema Compare.

See dbForge Studio.

See DbSchema.

See DbVisualizer.

See DeZign for Databases.

See ERBuilder Data Modeler.

See DBeaver.

See OmniDB.

See Sequel Pro.

See HeidiSQL.

See KS DB Merge Tools.

See LibreOffice Base.

See Luna Modeler.

See mycli.

See Navicat.

See ocelotgui.

See PgManage.

See phpMyAdmin.

See Querious.

See SB Data Generator.

See SQLPro Studio.

See SQLyog.

See TablePlus.

See TOAD Edge.

See Valentina Studio.

See MariaDB Direct Query Adapter For Microsoft Power BI.

This page is licensed: CC BY-SA / Gnu FDL

Adminer

> **Note:**
>
> Adminer is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Adminer is a database administration web interface. It is usable via a web browser. It is written in PHP and requires a web server.

Adminer mainly supports MySQL, but it also supports MariaDB and other database management systems. Adminer has a wide range of plugins, some of which have been developed by the community.

Adminer is distributed with a dual license: Apache License 2.0 and GPL 2. Adminer does not have a commercial edition, but it accepts donations.

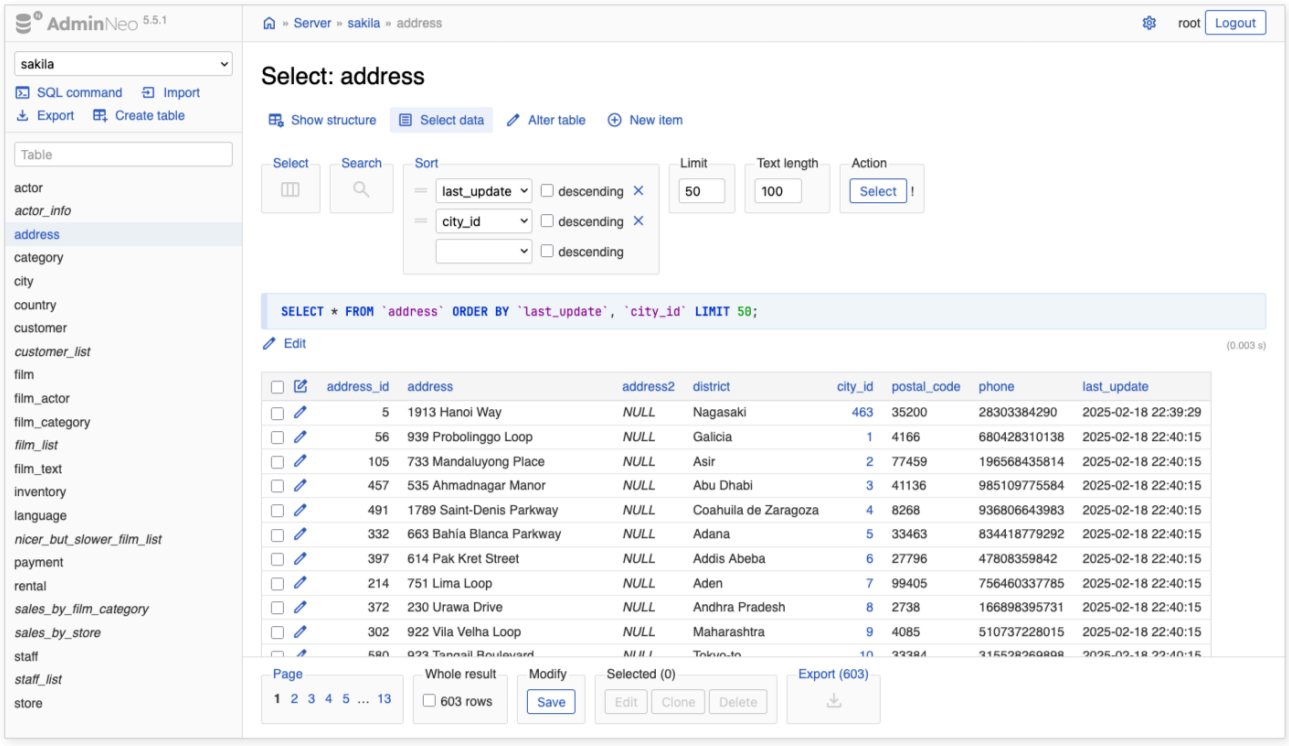
This page is licensed: CC BY-SA / Gnu FDL

AdminNeo

> **Note:**
>
> AdminNeo is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

AdminNeo is a database management tool written in PHP and used through a web browser. It consists of a single file that you deploy to a web server with PHP.

AdminNeo is based on the Adminer project, with a redesigned user interface and a refactored code base.



AdminNeo — selecting data

Key features

- Clean, modern user interface.

- Managing the structure of databases and tables.
- Data manipulation and searching.
- Exporting and importing databases and data.
- Executing batch SQL commands.
- Extensibility through plugins and customizations.

Supported databases

AdminNeo supports MariaDB through the MySQL driver. Other database systems: PostgreSQL, MS SQL, SQLite, Oracle, MongoDB, SimpleDB, Elasticsearch (beta) and ClickHouse (alpha).

License

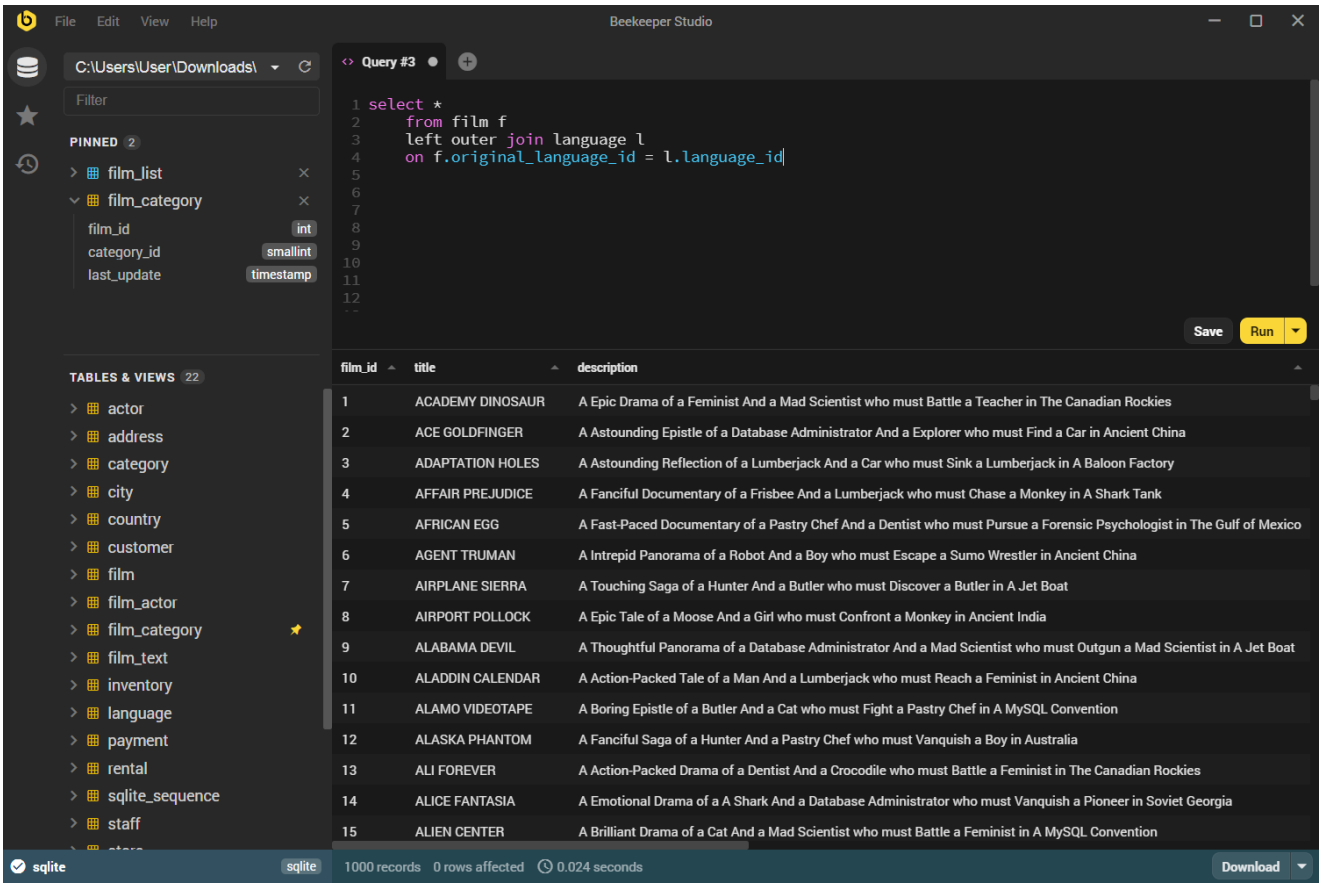
AdminNeo is dual-licensed under the Apache License 2.0 and GPL 2.

This page is licensed: CC BY-SA / Gnu FDL

Beekeeper Studio

> **Note:**
>
> Beekeeper Studio is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Beekeeper Studio is an open-source database GUI available for Linux, MacOS, and Windows.



SQL Gui Screenshot

Beekeeper Studio works with MariaDB, but also works with other databases.

Beekeeper Studio includes the following features:

- Tabbed interface.
- SSH tunneling.
- SQL autocomplete.
- Multi-connection.

The Community Edition is free and open source, and is privacy respecting, with no usage tracking or similar behavior.

Links

- Homepage - beekeeperstudio.io
- GitHub - beekeeper-studio

This page is licensed: CC BY-SA / Gnu FDL

SQL Diagnostic Manager & SQLyog

> **Note:**
>
> SQL Diagnostic Manager and SQLyog are third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support them. Refer to their own documentation and license terms.

SQL Diagnostic Manager is a monitoring tool that gives database administrators real-time insights for optimizing the performance of MariaDB servers.

Key Features:

1. Agentless monitoring.
2. Fully customizable.
3. Affordable.

Webyog's SQL DM is fully compatible with Amazon Aurora databases, and Webyog is an Amazon launch partner.

Features specifically for RDS and Aurora instances also available. SQL Diagnostic Manager is browser based and is a proud MariaDB Monitor and Advisor.

SQLyog Ultimate is a MySQL administration tool for DBAs, developers, and database architects.
This tool enables database developers, administrators, and architects to visually compare, optimize, and document schemas.

Key Features:

1. Automatically synchronize data.
2. Visually compare data.
3. Import external data.

SQLyog runs on Windows and is a graphical MariaDB manager and admin tool, combining the features of MySQL Administrator, phpMyAdmin and other MariaDB frontend tools.

This page is licensed: CC BY-SA / Gnu FDL

Database Workbench

> **Note:**
>
> Database Workbench is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

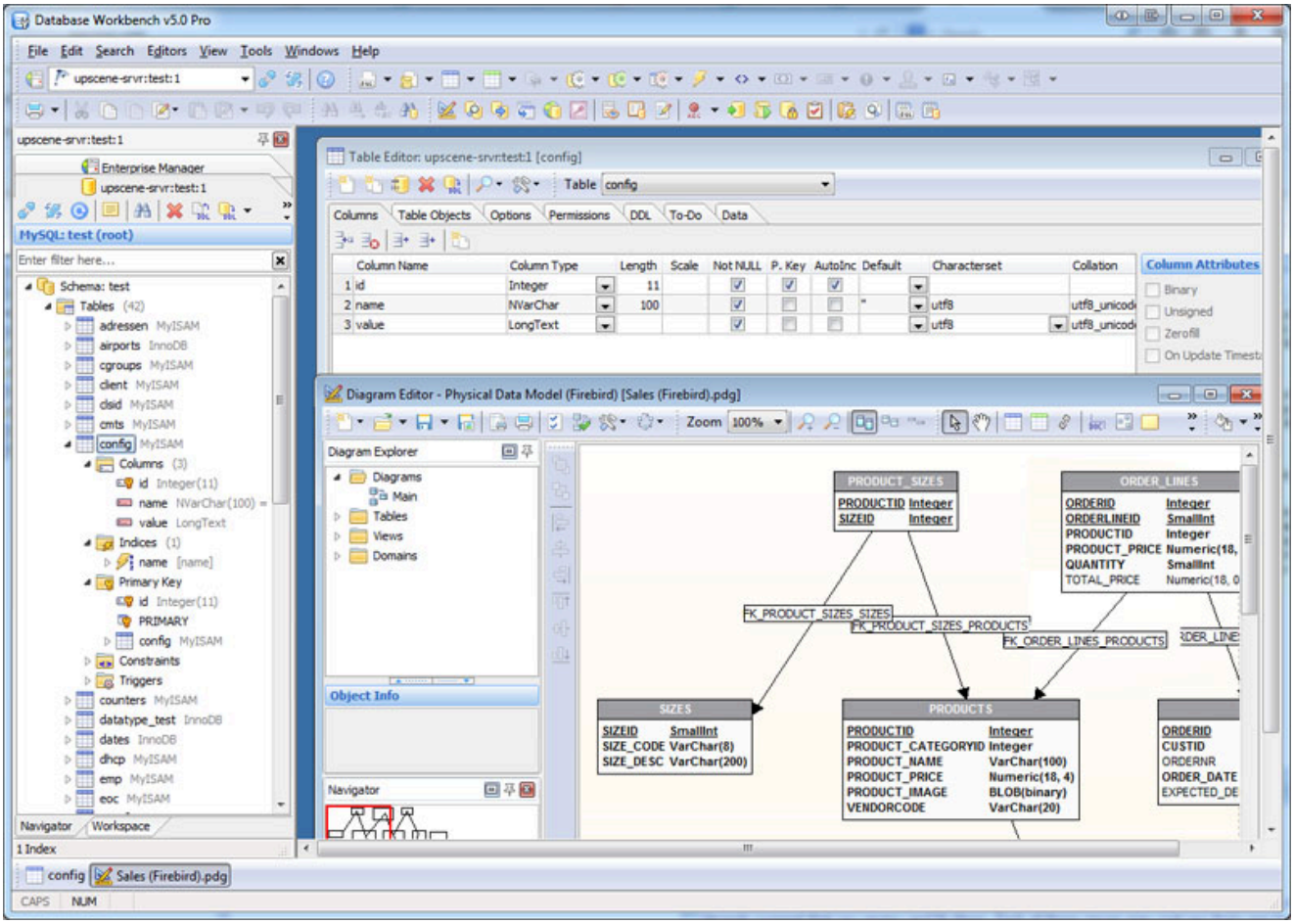
Database Workbench is a Windows application (*which works fine under Wine on Linux*) for database design, development, maintenance and testing for several database systems, including MySQL and MariaDB.

With Database Workbench you can:

- design a database;
- connect to multiple databases;
- view and modify meta data;
- view and modify data;
- import and export data;
- connect to any ODBC/ADO data source;
- transfer data between database systems;
- migrate meta data;
- compare meta data;
- manage privileges;
- debug stored procedure;
- print meta data or data.

It includes numerous other tools.

Database Workbench comes in multiple editions and is licensed on a per-module basis.



dbwoverview

This page is licensed: CC BY-SA / Gnu FDL

dbForge Data Compare

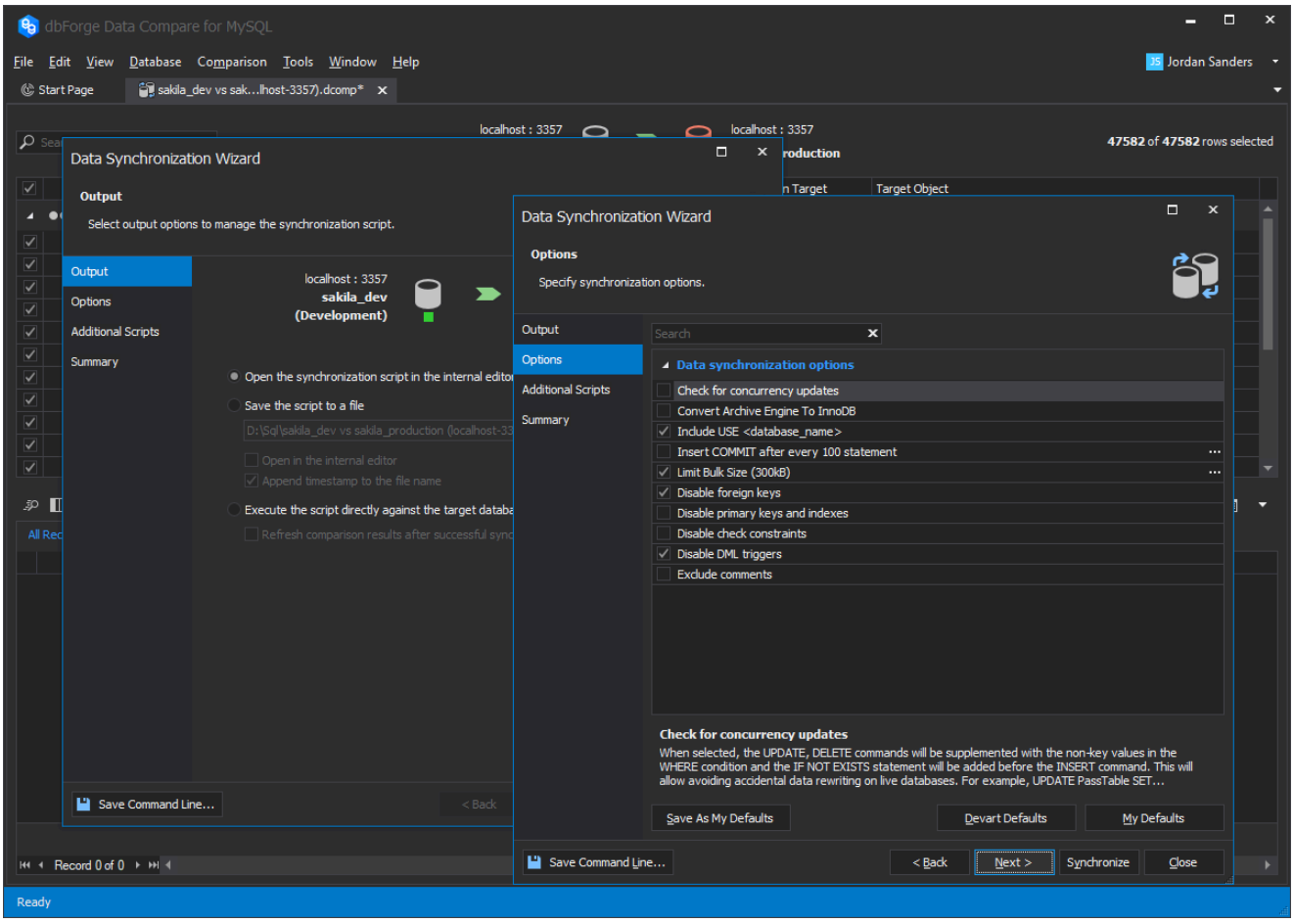
> **Note:**
>
> dbForge Data Compare is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

dbForge Data Compare helps compare and synchronize data in MariaDB, MySQL, and Percona databases and scripts folders. You can find differences between your data, as it helps analyze comparison results, creates a synchronization script, and applies changes. Additionally, MariaDB data can be compared with command-line support.

Features are described in the following.

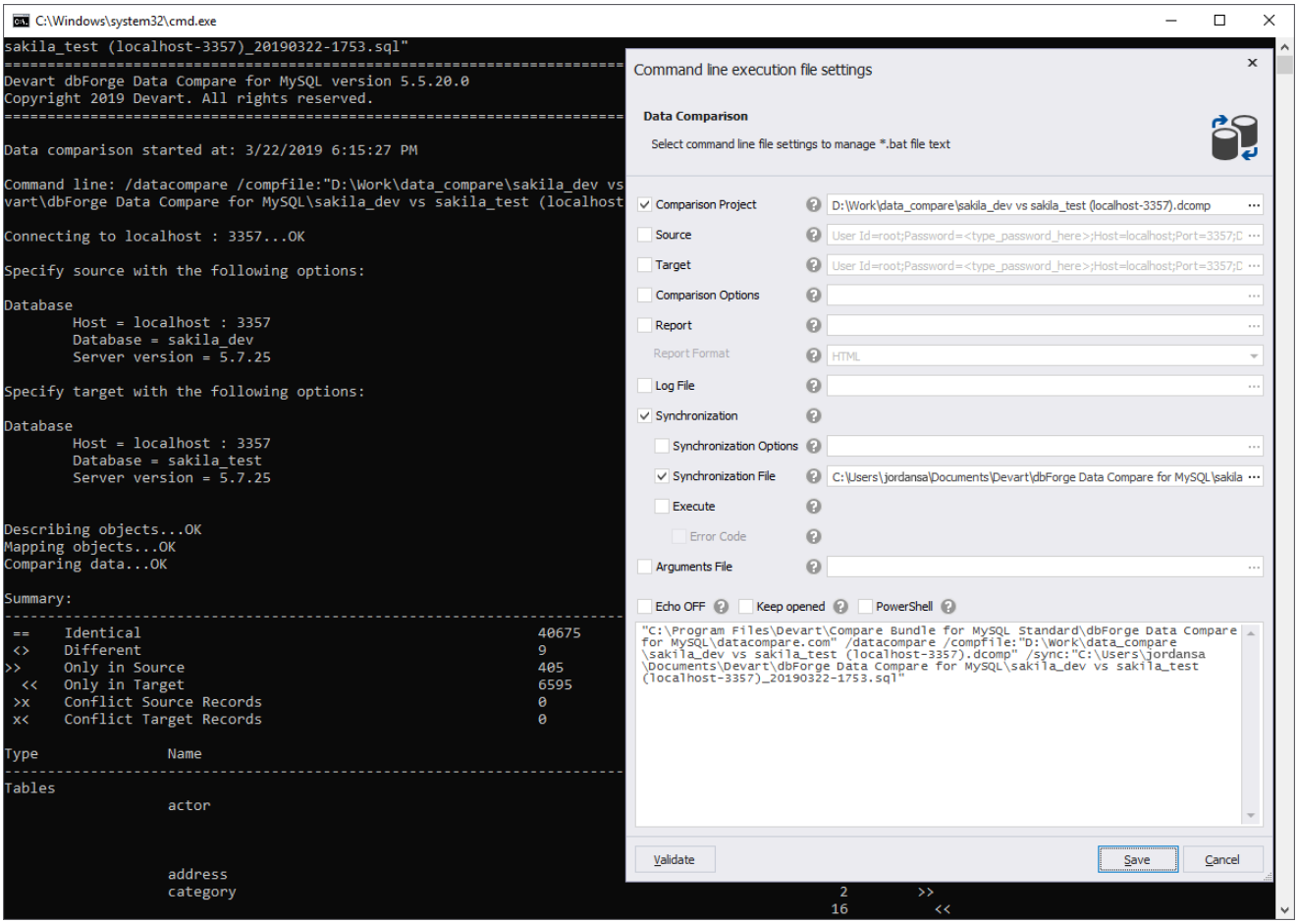
Database Synchronization Features

- Create accurate synchronization scripts.
- Synchronize asynchronous data.
- Create custom scripts with execution before/after the process of data synchronization.
- Get warning messages on errors during synchronization.



MySQL & MariaDB Task Automation

- Automate routine synchronizations with a command line interface.
- Plan routine execution with Windows Scheduler.
- Support for PowerShell compatibility.

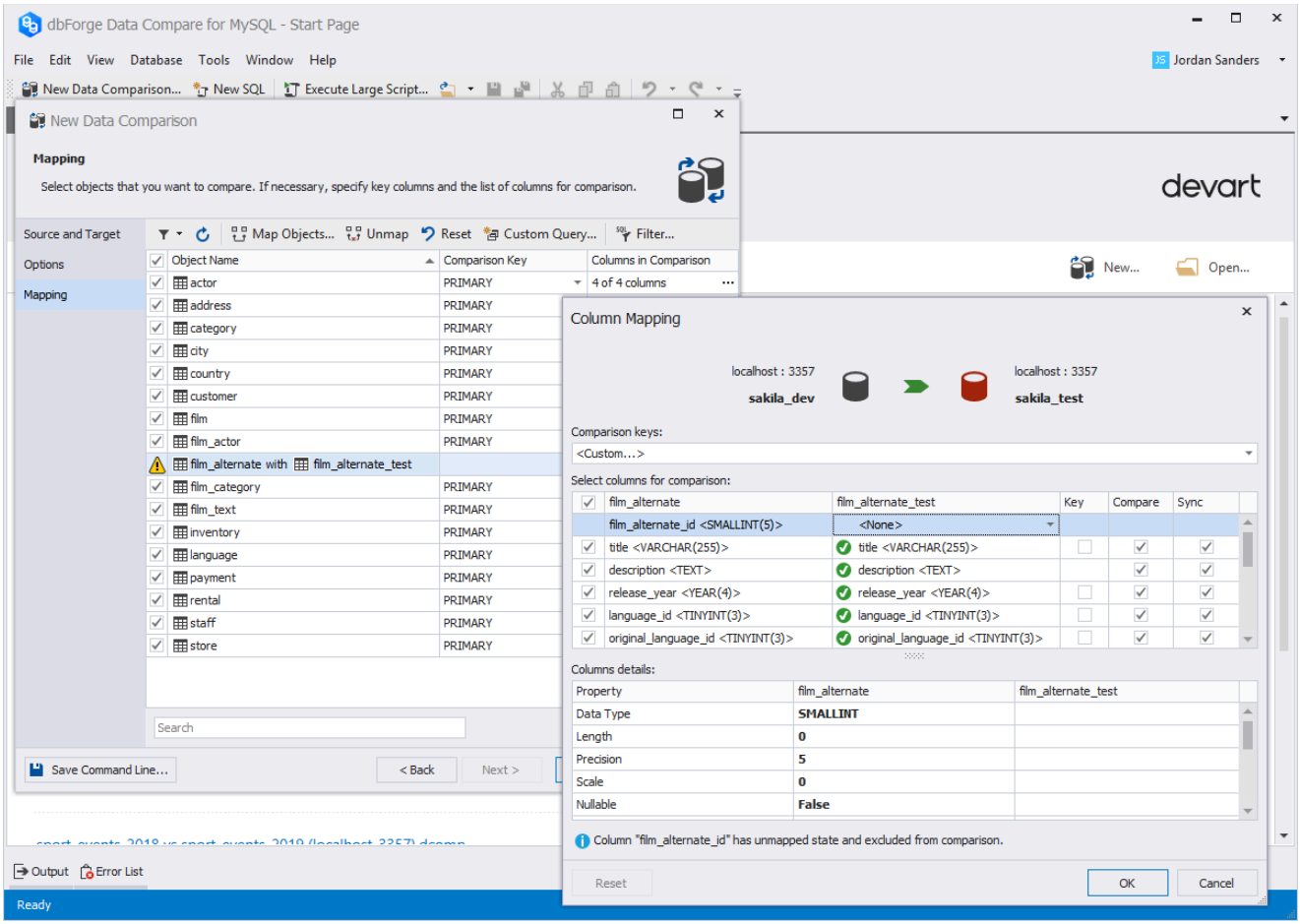


Script for Folder Data Comparison

- Compare data from your database with the data placed in the script folder locally.
- Choose any database, connection, or script folder with the new Data Comparison Wizard.
- See the data comparison results in a data result grid.

Configurations for Data Comparison

- Various options for automatic mapping.
- Compare customized queries.
- Compare custom keys.



Supported Data

- Tables.
- Temporal tables.
- Views.
- `BLOB` data.
- Compare databases and sync different MariaDB versions.

Compatibility Options

- All popular MariaDB server versions.
- MariaDB setup on Windows, Linux, FreeBSD, etc.
- Various cloud services: Amazon RDS, Amazon Aurora, Google Cloud, Oracle MySQL Cloud, Alibaba Cloud.
- Secure Socket Layer (SSL), Secure Shell (SSH), HTTP Tunneling, PAM Percona security connections.

Manage Comparison Results

- Find tables by name.
- Filter the comparison process results.
- Enter the values of default comparison options.
- Group records by type in each compared object.

Export Comparison Results

- Use only relevant data in your reports.
- Generate comparison reports in various formats.

Download a free 30-day trial of dbForge Data Compare here.

Documentation

Version	Introduced
dbForge Data Compare 10.1	Support for the rds-ca-rsa2048-g1 SSL/TLS certificate for connecting to MariaDB servers on AWS and enhanced cloud compatibility features, new automation capabilities.
dbForge Data Compare 10.0	Support for MariaDB 11.4, Added support for temporal tables in MariaDB.
dbForge Data Compare 5.9	Support for MariaDB 11.3.
dbForge Data Compare 5.8	Support for MariaDB 10.9 and 10.10.
dbForge Data Compare 5.7	Connectivity support for MariaDB 10.5.

This page is licensed: CC BY-SA / Gnu FDL

dbForge Data Generator

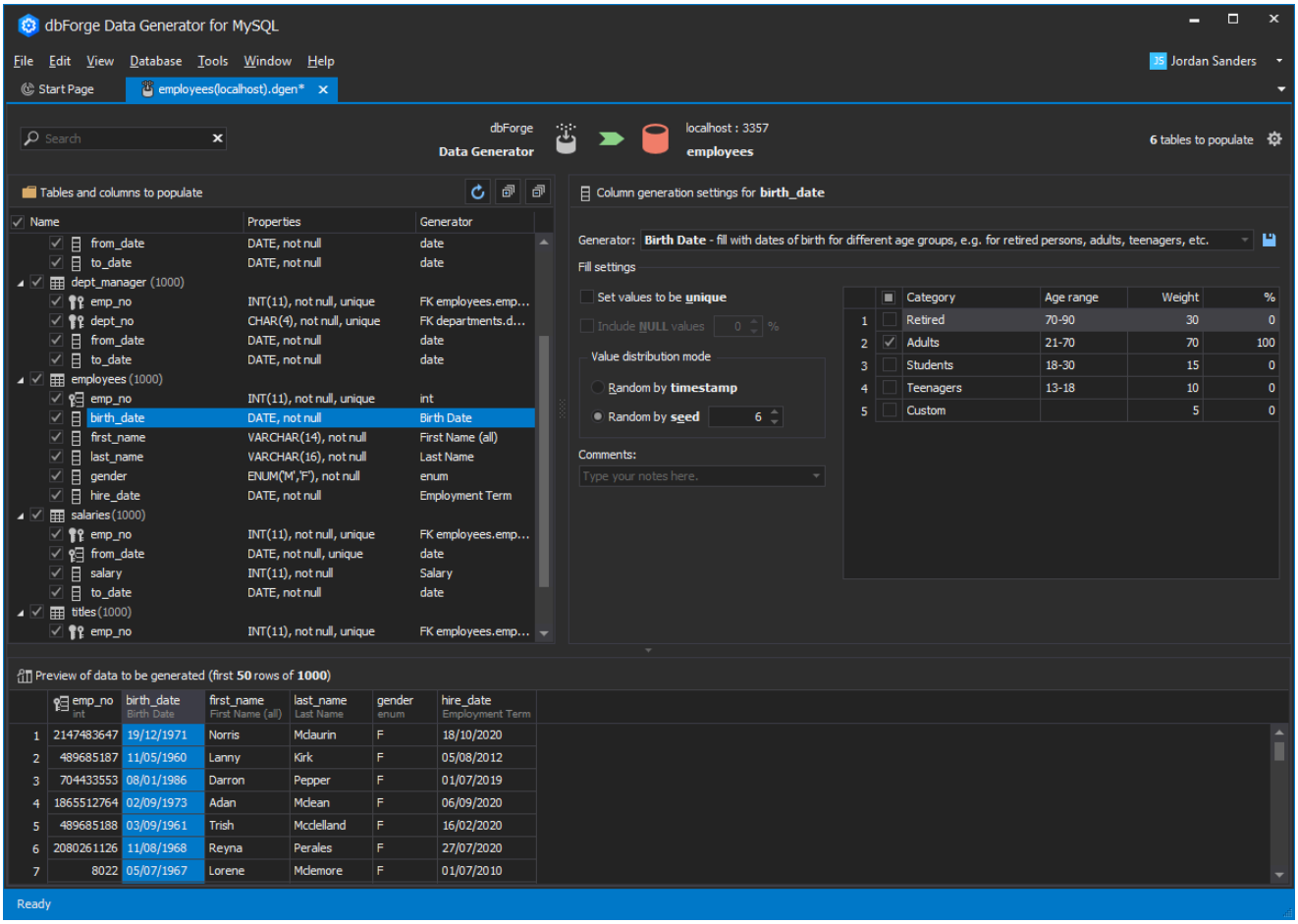
> **Note:**
>
> dbForge Data Generator is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

dbForge Data Generator for MariaDB and MySQL is a powerful solution that helps create massive volumes of meaningful and realistic data. This tool performs various predefined data generators with customizable options.

Data Generator Key Features:

1. Extensive support for all column data types

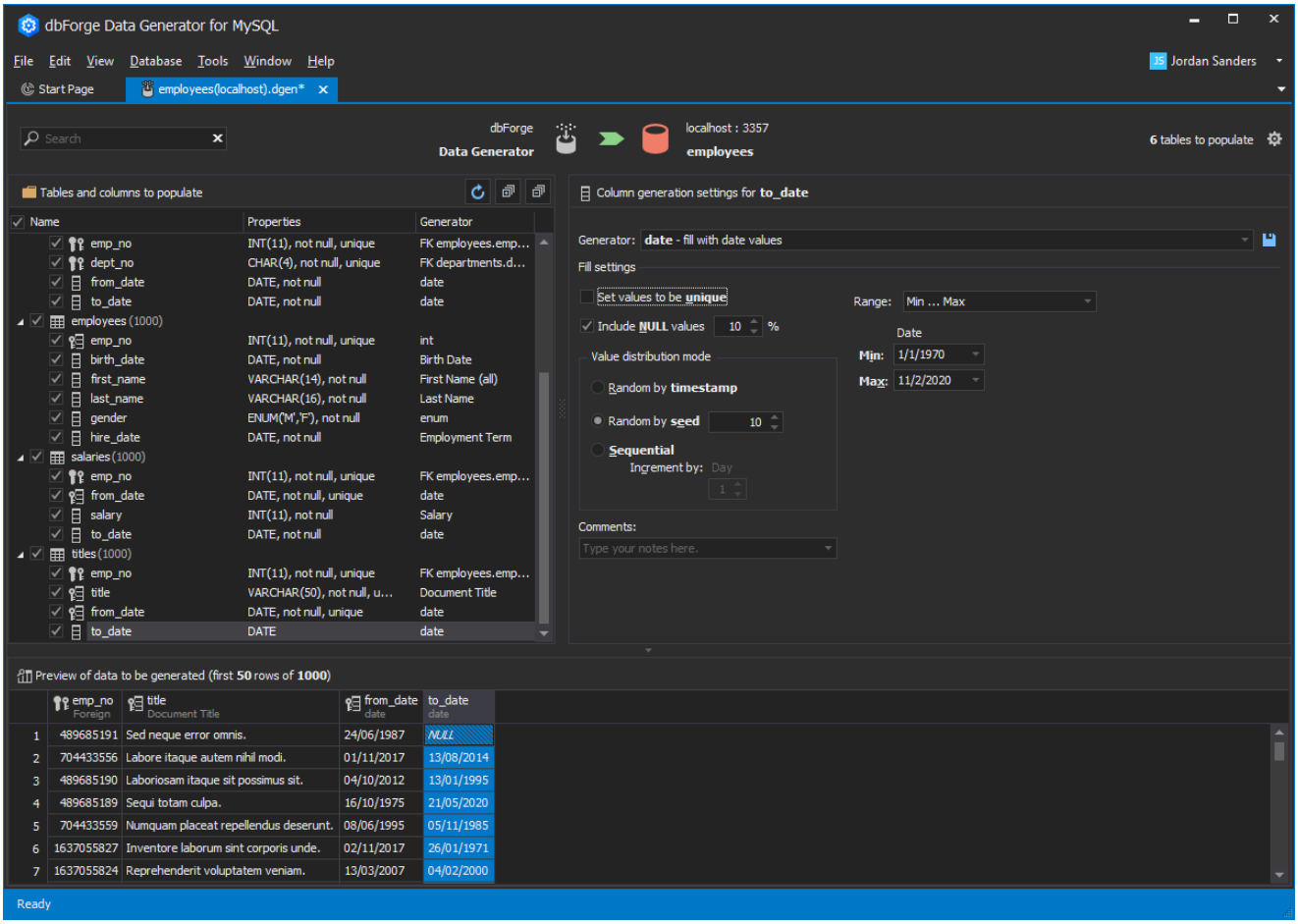
Enjoy the advantage of various data types support



column-data-types

2. Multiple generators with unique test data

Enjoy multiple customization options with individual generators for every data type supported



custom-options

3. MariaDB data generation and data integrity support

Generate consistent random data through multiple tables

Disable triggers and constraints to avoid interference with database logic

4. MariaDB data distribution modes

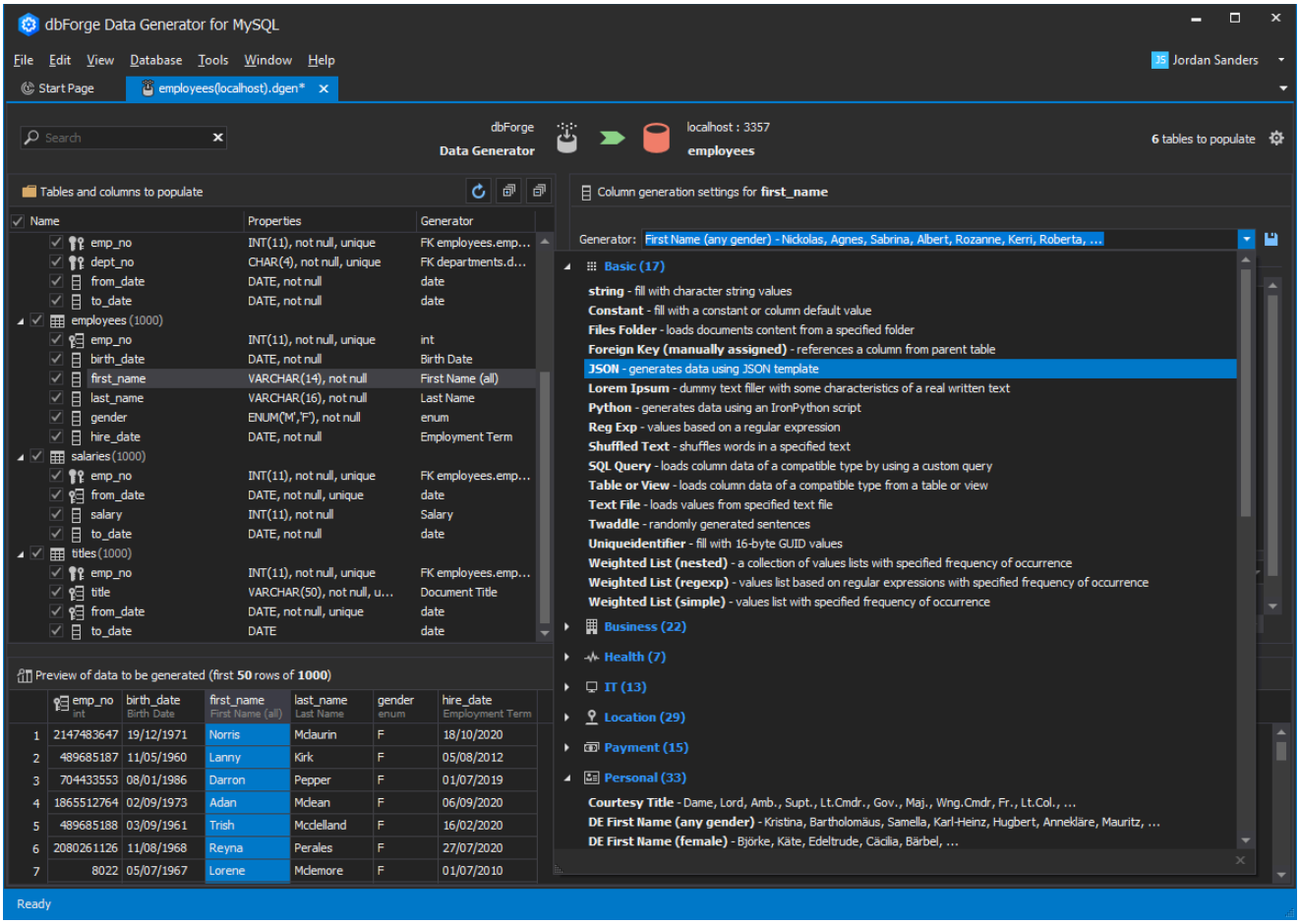
Fine-tune the way you want your data to be generated

5. Multiple ways to generate data in MySQL

Generate a data population script, execute it against a MariaDB database, save or edit it later

6. Basic generators

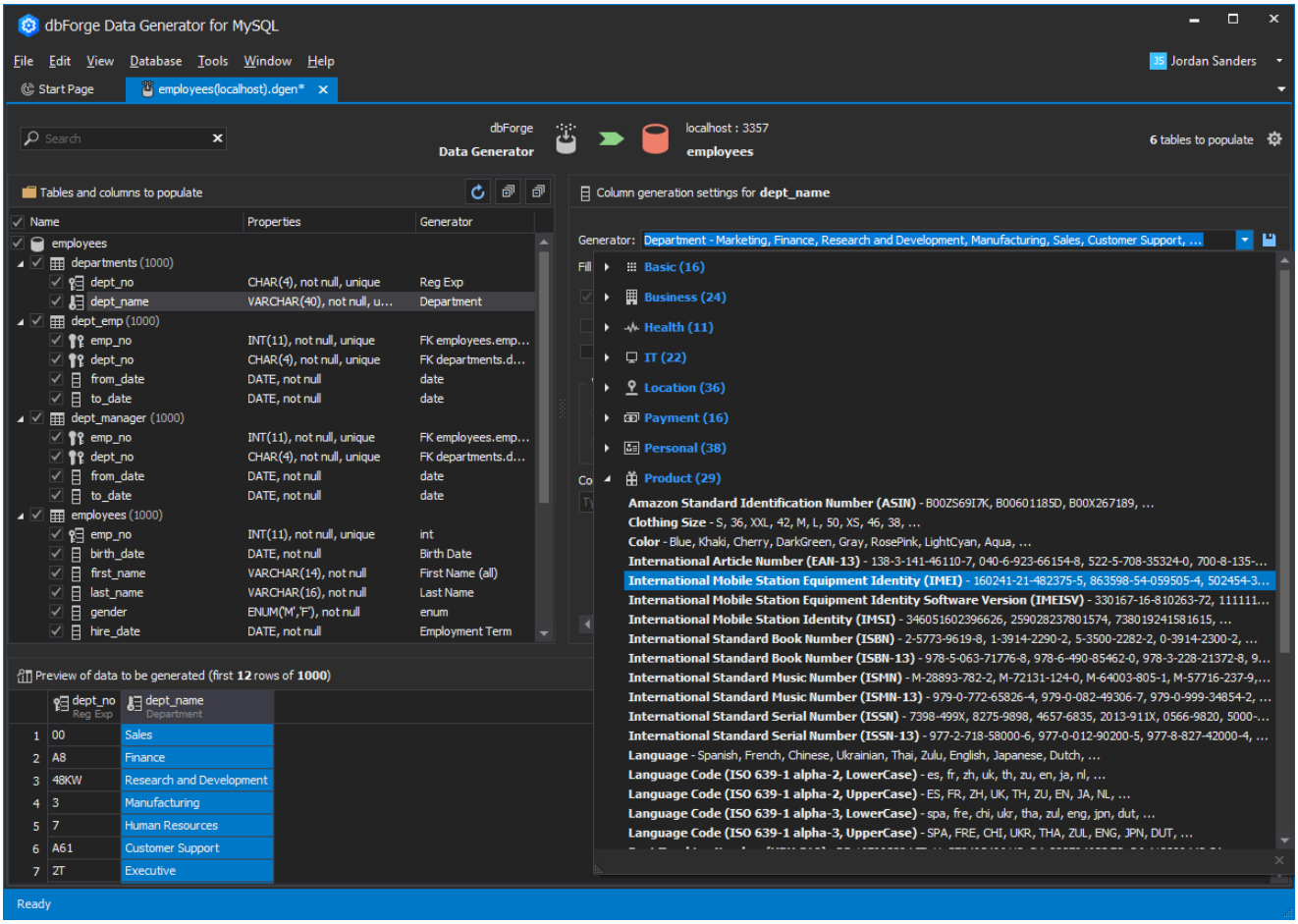
Populate tables with a great variety of values types, like JSON, Python, XML, etc.



basic-generators-1

7. Meaningful test data generators

Select any of 200+ real-world generators and populate tables with realistic data related to various spheres



meaningful-generators-1

8. User-defined generators

Create, save and use your own data generators tailored for your needs

9. Real-time data generation preview

Enjoy real-time visualization of the alterations you make

Visually assess the data to be generated

10. Command-line interface

Schedule your routine data generation tasks

Create a command line execution file for running database documentation tasks

11. Broad compatibility options

MariaDB server versions 5.5-11.4

Various cloud services: Amazon RDS, Amazon Aurora, Google Cloud, Oracle MySQL Cloud, Alibaba Cloud

Security connections: Secure Socket Layer (SSL), Secure Shell (SSH), HTTP Tunneling, PAM Percona

Download a free 30-day trial of dbForge Data Generator here.

Documentation

Version	Introduced
dbForge Data Generator 10.1	Support for the rds-ca-rsa2048-g1 SSL/TLS certificate for connecting to MariaDB servers on AWS and enhanced cloud compatibility features, new automation capabilities
dbForge Data Generator 10.0	Support for MariaDB 11.4, Added support for temporal tables in MariaDB
dbForge Data Generator 2.6	Support for MariaDB 11.3
dbForge Data Generator 2.5	Support for MariaDB 10.9, Support for MariaDB 10.10
dbForge Data Generator 2.4	Connectivity support for MariaDB 10.5 is added
dbForge Data Generator 2.2	Support for MariaDB 10.4
dbForge Data Generator 2.1	Support for MariaDB 10.3
dbForge Data Generator 2.0	Support for MariaDB 10.2, Support for MariaDB 10.1

This page is licensed: CC BY-SA / Gnu FDL

dbForge Documenter

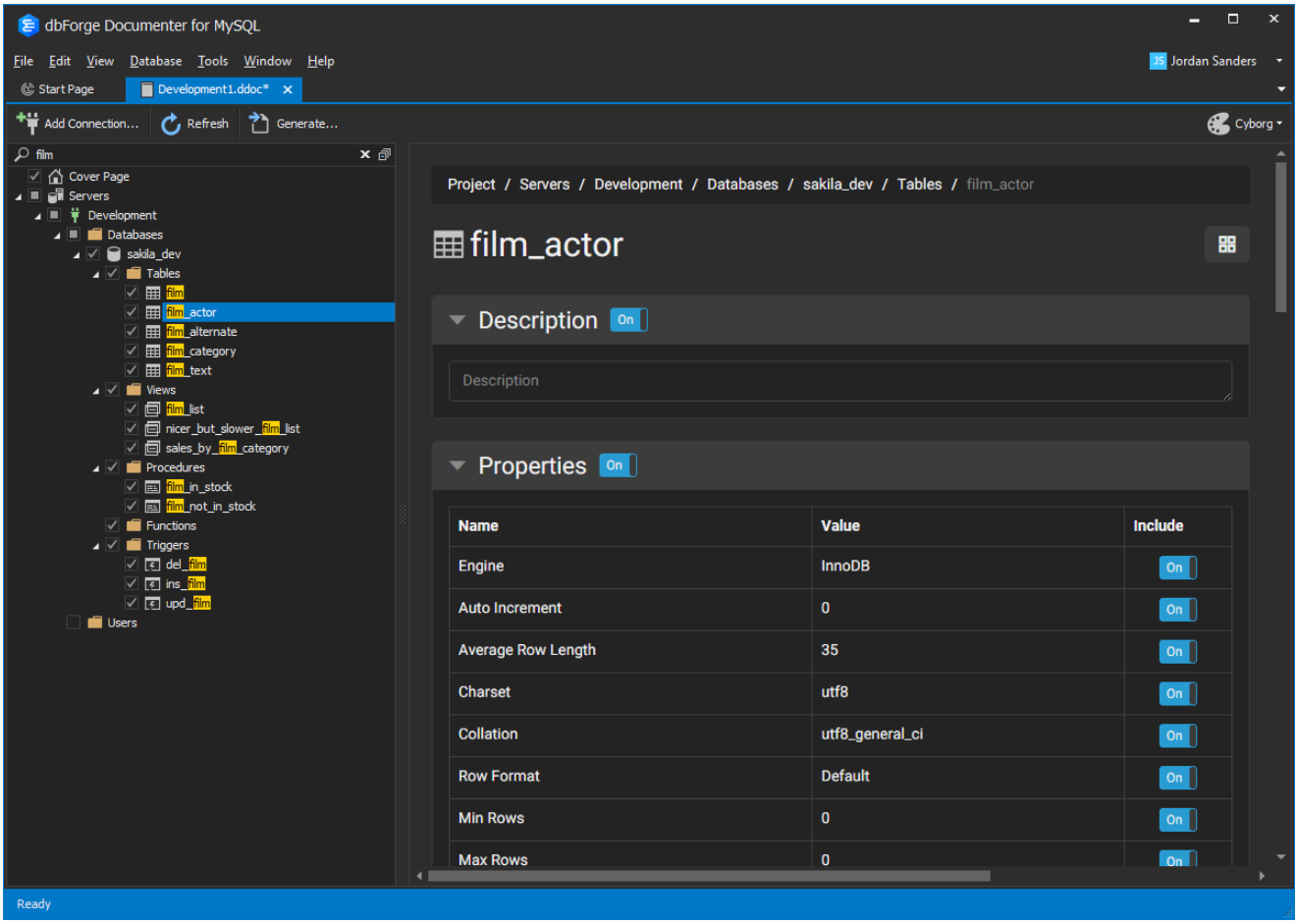
> **Note:**
>
> dbForge Documenter is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

dbForge Documenter is a useful tool for the MariaDB database that allows for the automatic generation of database documentation in such formats as HTML, PDF, and Markdown. Users can adjust the created documentation with a great variety of options.

Features are described in the following.

Searchable MariaDB & MySQL Documentation

Search as you type. View the highlighted matching text after entering the name of a required object in the search field.



Compatibility Options

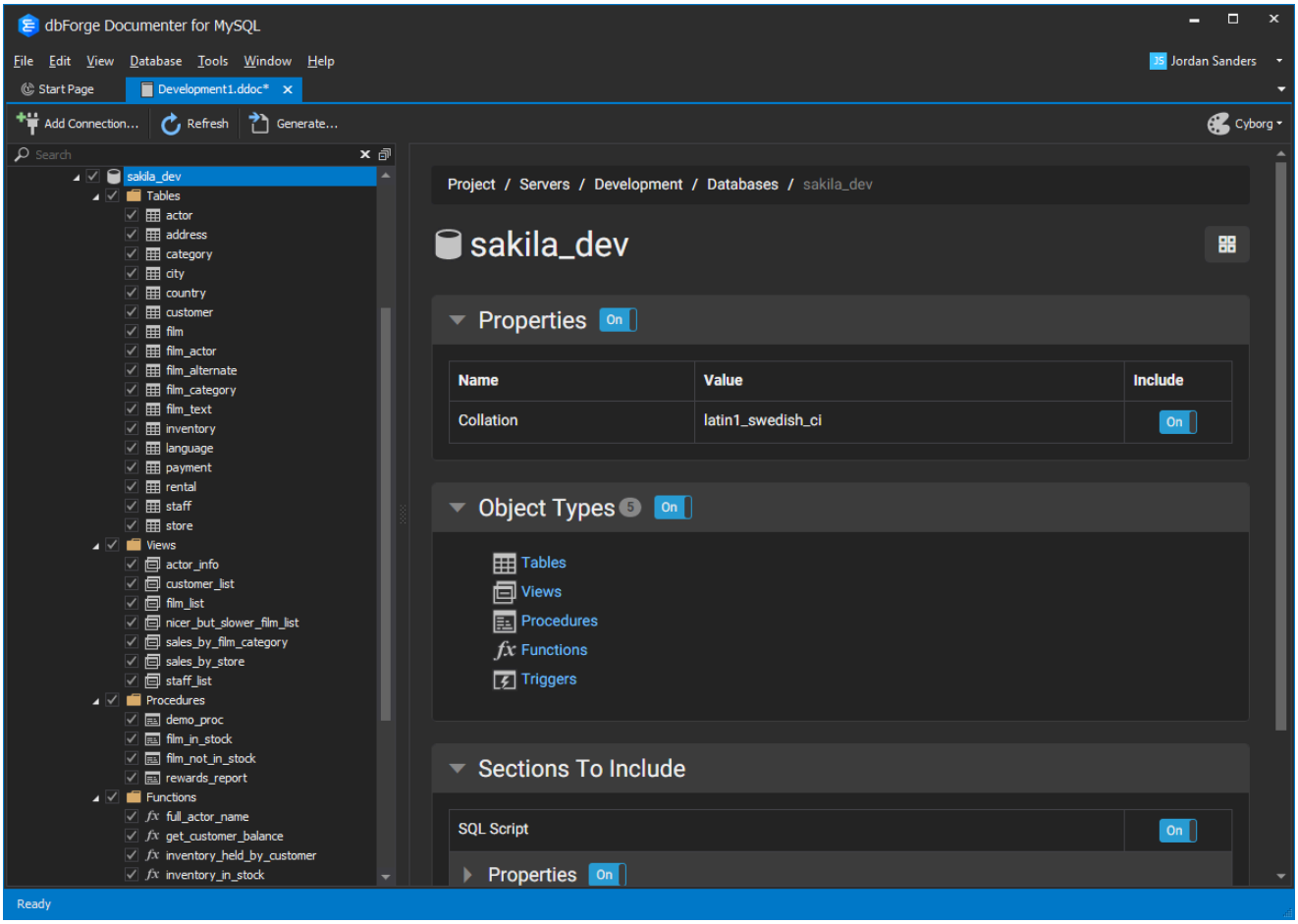
Various cloud services: Amazon RDS, Amazon Aurora, Google Cloud, Oracle MySQL Cloud, Alibaba Cloud,

Security connections: Secure Socket Layer (SSL), Secure Shell (SSH), HTTP Tunneling, PAM Percona.

compatibility

Database Structure

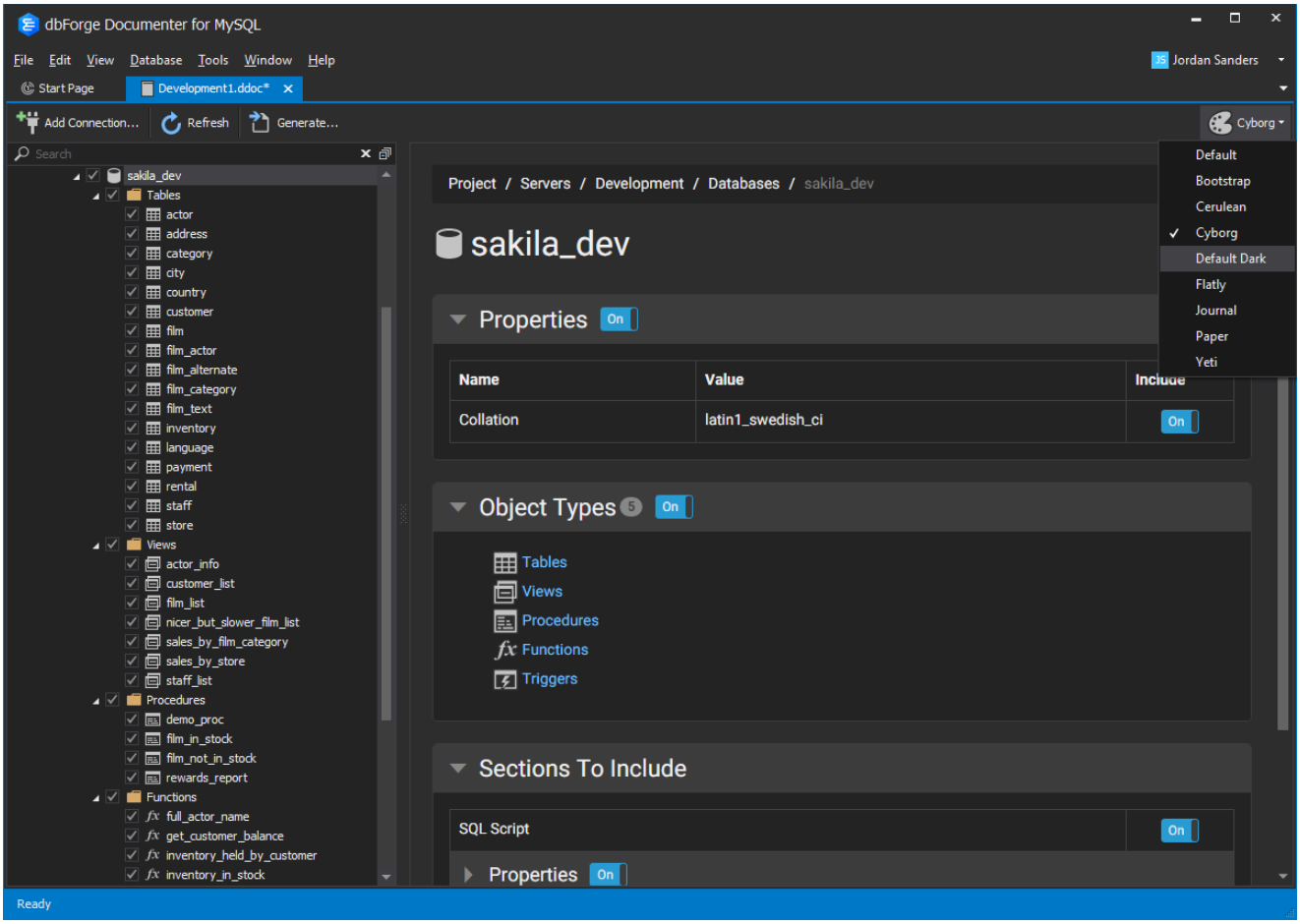
Detailed MariaDB database information, such as types, details, and properties of the objects, inter-object dependencies, and DDL codes.



database-structure

Customization of Document Management

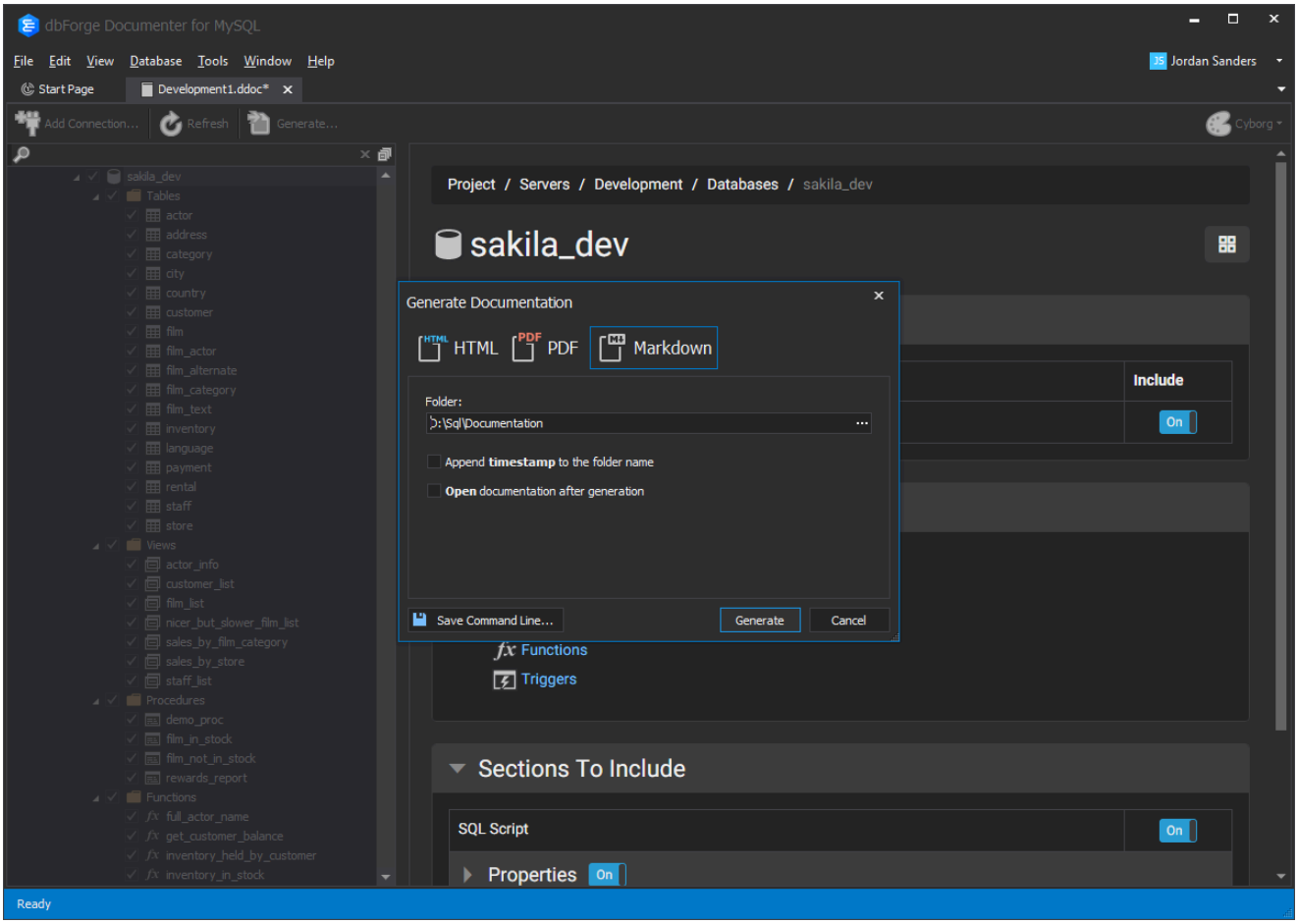
Various style templates allowing for alteration of documentation layout.



custom-features

Document Formats

- HTML and PDF (searchable formats).
- Markdown.



formats

Extended Properties Support

Edit or add an object description.

Download a free 30-day trial of dbForge Documenter for MariaDB and MySQL here.

Documentation is available, too.

Version	Introduced
dbForge Documenter 10.1	Support for the rds-ca-rsa2048-g1 SSL/TLS certificate for connecting to MariaDB servers on AWS and enhanced cloud compatibility features, new automation capabilities.
dbForge Documenter 10.0	Support for MariaDB 11.4. Added support for temporal tables in MariaDB.
dbForge Documenter 2.2	Support for MariaDB 11.3.
dbForge Documenter 2.1	Support for MariaDB 10.9 and 10.10.
dbForge Documenter 2.0	Connectivity support for MariaDB 10.5.

This page is licensed: CC BY-SA / Gnu FDL

dbForge Edge

> **Note:**
>
> dbForge Edge is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

dbForge Edge is a versatile software solution designed to meet the needs of full-stack database specialists. It offers a wide range of features to address database challenges across major providers: MySQL/MariaDB, SQL Server, Oracle, and PostgreSQL. When it comes to MySQL and MariaDB, dbForge Edge provides a highly functional IDE that covers almost all aspects of database development and management.

Features are described in the following.

Design

dbForge Edge includes a built-in Database Designer equipped with ER diagrams, allowing users to construct and edit databases. An intuitive interface facilitates the creation of database structures and generation of reusable SQL scripts for creating actual databases. Table Designer is a separate feature for visual creation and management of tables that covers the setup of columns, data types, constraints, etc.

SQL Development

dbForge Edge offers a powerful set of features to simplify SQL coding tasks. The code assistance module offers context-based keyword and object suggestions, code completion, instant syntax validation, customizable code formatting profiles, debugging, and code refactoring capabilities. Another feature worth mentioning is Query Builder – a tool that allows users to construct even the most complex SQL queries effortlessly in a visual mode.

Database Management

The solution incorporates schema and data comparison tools, enabling users to identify and manage differences in database schemas and table data. With automatically generated synchronization scripts, changes can be seamlessly deployed to target platforms. Comparison and synchronization jobs can also be automated, improving user efficiency.

dbForge Edge supports data migration (import and export) in over 10 major data formats. Both the import and export processes can be tailored to meet any specific requirements and automated for efficient workflows.

Also, dbForge Edge delivers Source Control, an integrated tool that enables version control of database schemas and static data, available for MariaDB, MySQL, and SQL Server databases. With its help, users can compare database versions, commit and revert changes with dependency awareness, view and resolve conflicts, and examine the history of changes. Source Control supports all popular version control systems, such as Git (including GitHub, GitLab, Azure DevOps, and Bitbucket), Apache Subversion (SVN), TFVC, Mercurial (Hg), Perforce (P4), and SourceGear Vault.

Testing

The solution features a robust Data Generator capable of producing high-quality test data in varying volumes and formats. Additionally, the Query Profiler tool enables comprehensive analysis and optimization of queries, thereby saving valuable time and resources.

Security Management

The dedicated module deals with user management configurations. The functionality provided by the software allows easy creation, editing, and management of user accounts and privileges.

Documenting and Reporting

With dbForge Edge, users can create detailed database documentation in various formats and with convenient navigation. The reporting module enables the summarization and analysis of data, presenting comprehensive reports with informative graphs, diagrams, and pivot tables.

dbForge Edge is not just a comprehensive IDE for MySQL and MariaDB. It also serves as an all-in-one solution for database experts who work with multiple database management systems, eliminating the need to assemble toolsets for each. With its extensive feature set, dbForge Edge empowers professionals to efficiently perform their tasks across diverse database environments.

Download a free 30-day trial of dbForge Edge.

This page is licensed: CC BY-SA / Gnu FDL

dbForge Fusion

> **Note:**
>
> dbForge Fusion is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

dbForge Fusion is an add-in for Visual Studio. It provides automatic and simple MariaDB database development, and boosts data management capacity. With this tool integrated, it is easy to work with database development and administration tasks from Visual Studio.

> **Note:**
>
> Note: dbForge Fusion is officially discontinued. However, you can access the features available in dbForge Fusion for MySQL/MariaDB through dbForge Studio for MySQL, which is actively supported and regularly updated.

This page is licensed: CC BY-SA / Gnu FDL

dbForge Query Builder

> **Note:**
>
> dbForge Query Builder is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

dbForge Query Builder is a visual tool that helps create any sort of MariaDB queries. Queries can be drawn on a diagram.

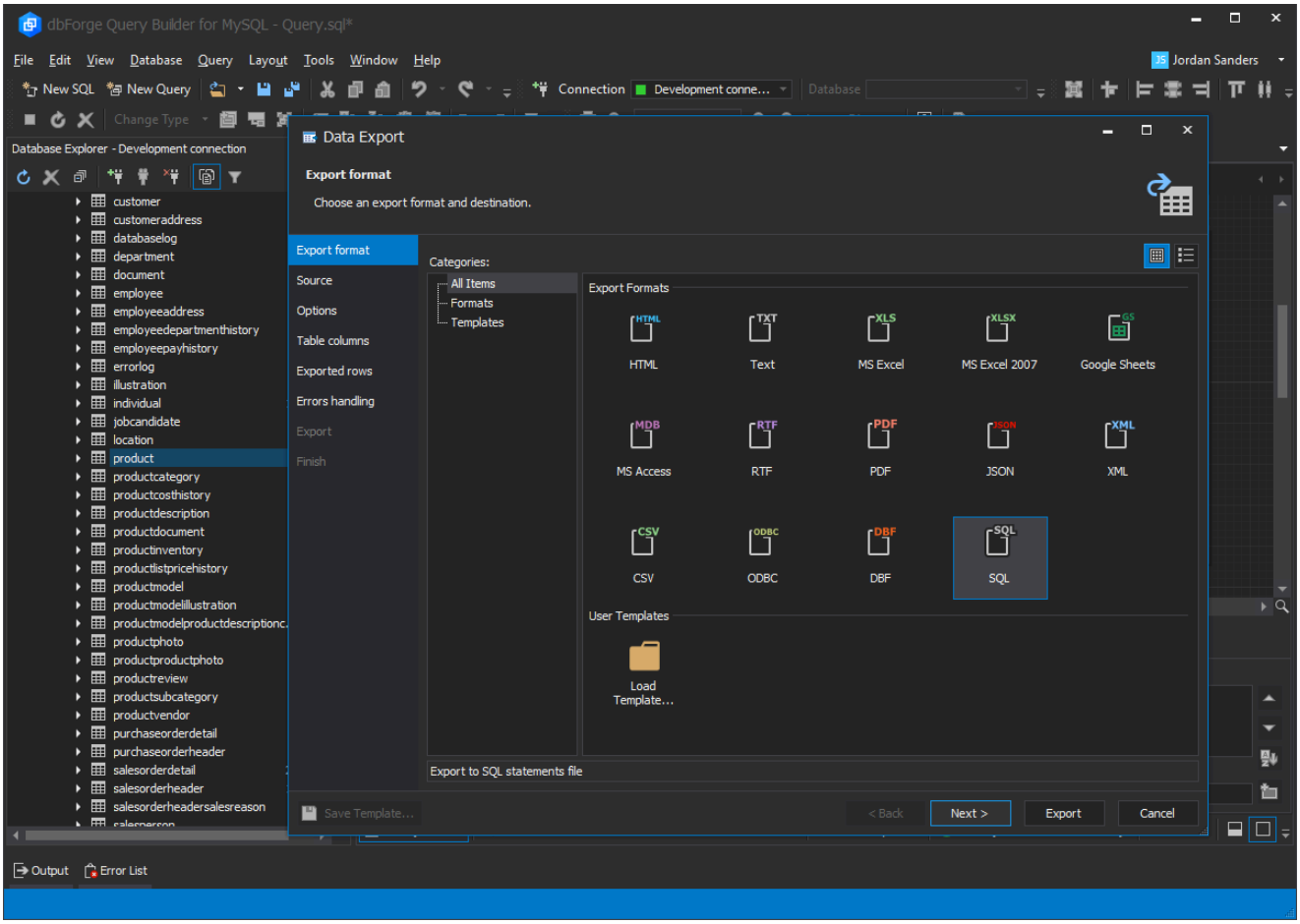
Features are described in the following.

Exporting Data

Export data to many popular formats.

Export data from several tables simultaneously.

Create templates for repeated operations.

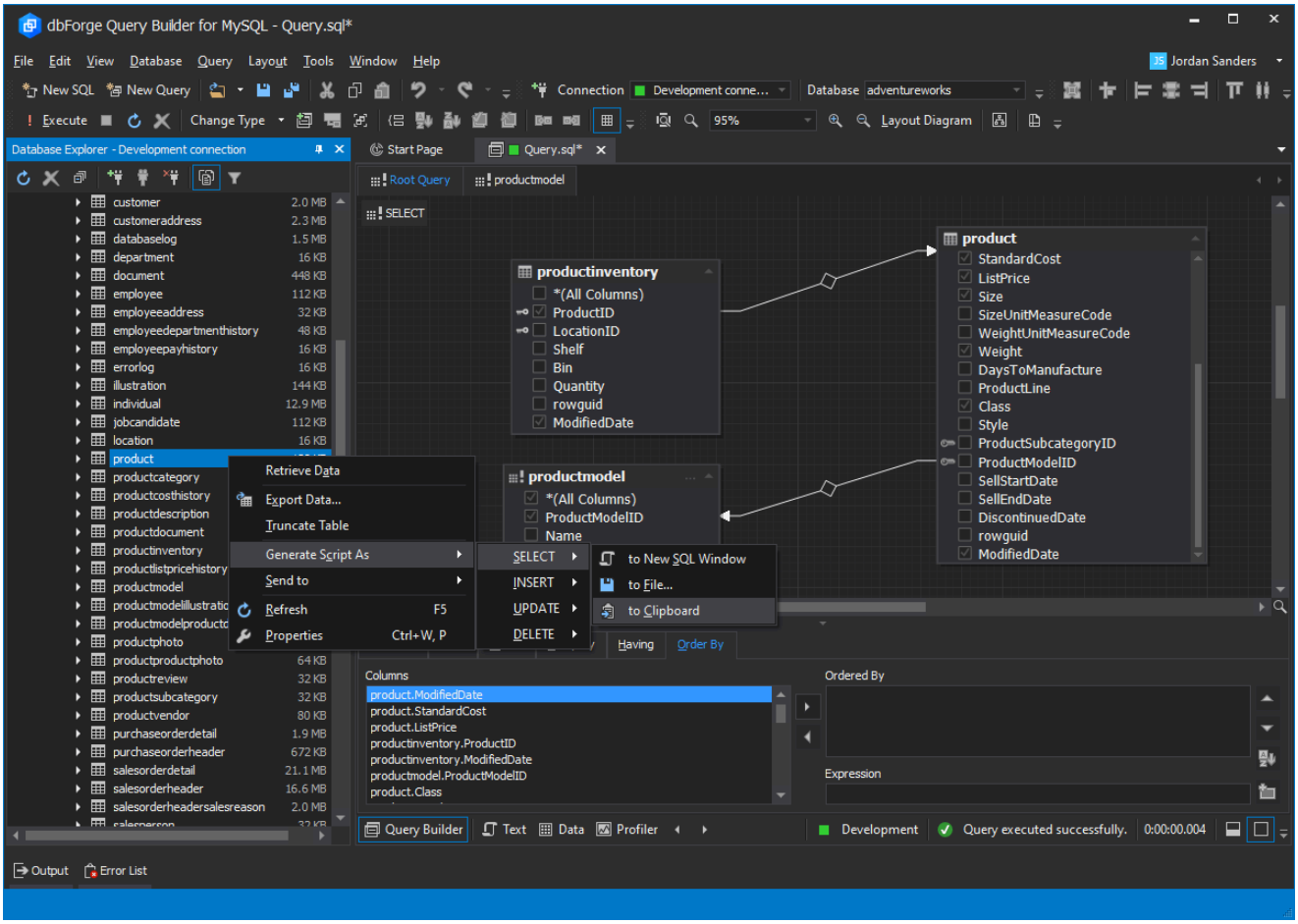


Database Explorer

Work with multiple database connections.

MariaDB query management with drag&drop functionality.

Quick script generation.



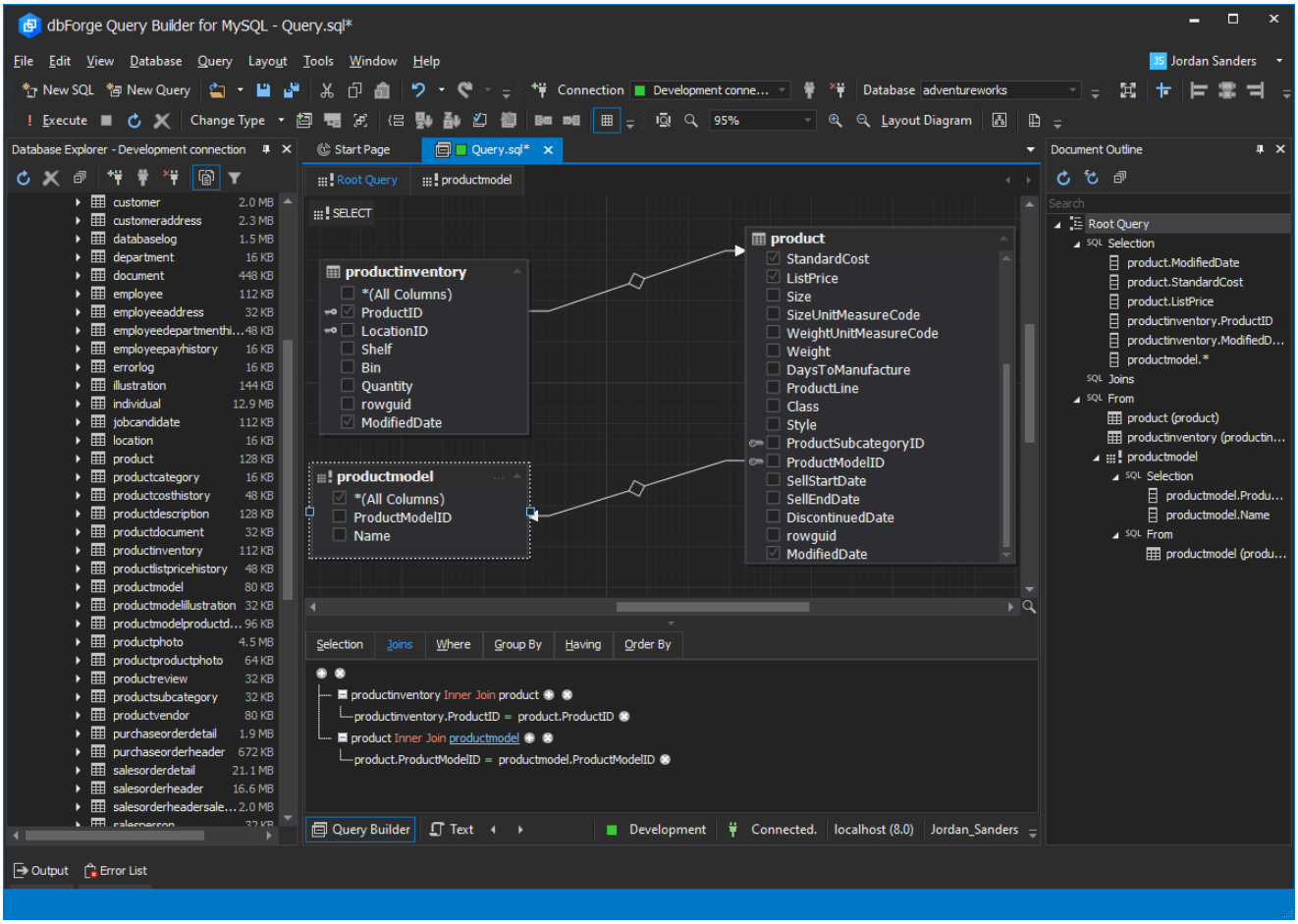
Visual Drag and Drop Query Builder

Zooming and keyboard support in a sophisticated diagram.

Preview and print diagrams.

Add and edit sub-queries visually.

MariaDB JOINs generator.



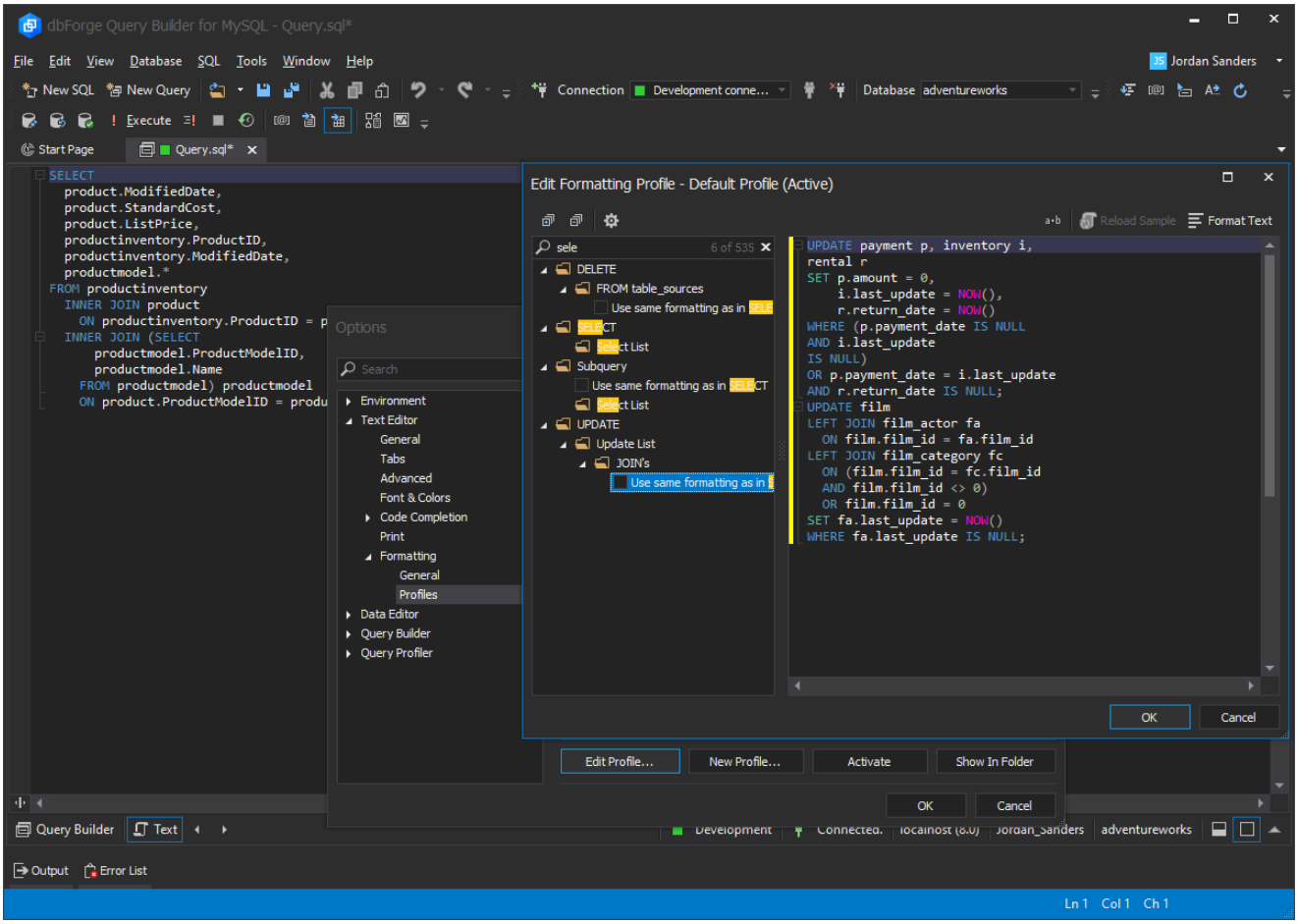
SQL Editing and Execution

Check the syntax automatically.

Get quick information about schema objects.

Customize formatting profiles.

Execute scripts, statements and MariaDB fragments seamlessly.

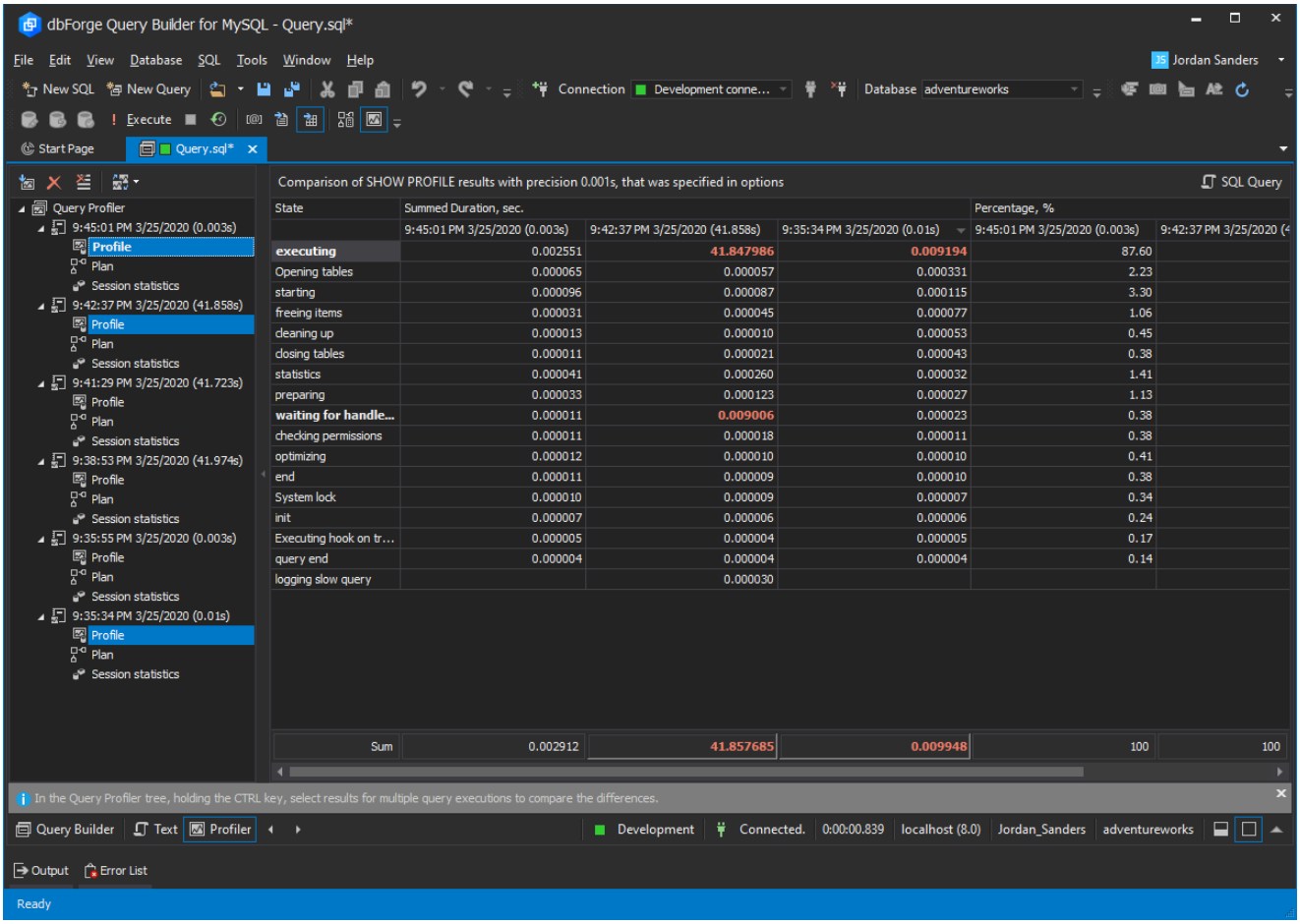


Query Profiler

Compare the results of the query visually.

Review and evaluate a query plan shown in the tree view.

Get the detailed query information.

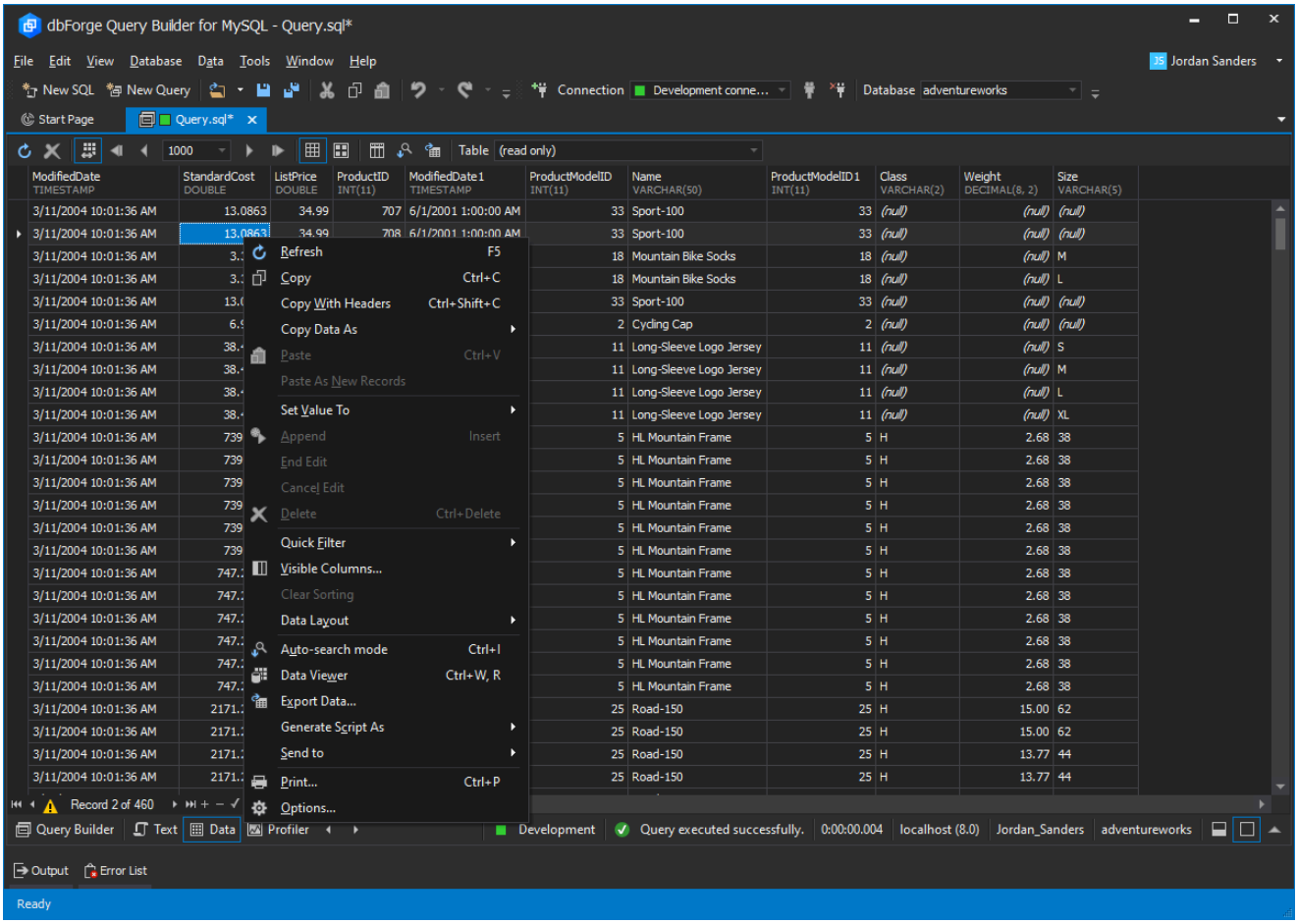


Data Editor

Advanced data filters.

Browse and edit large objects in Data Viewer and Editor windows.

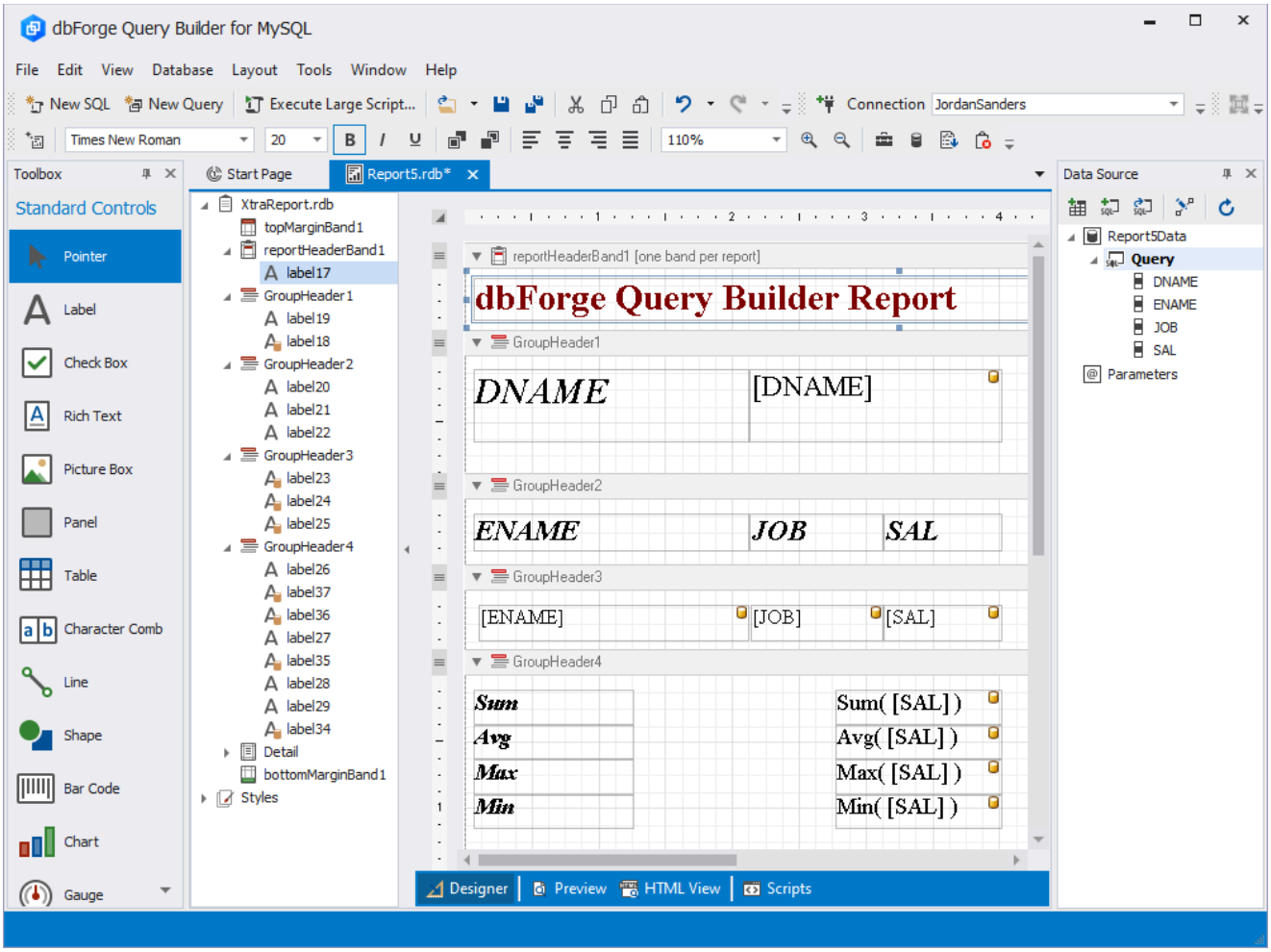
Fetch asynchronous data.



slider-data-editor

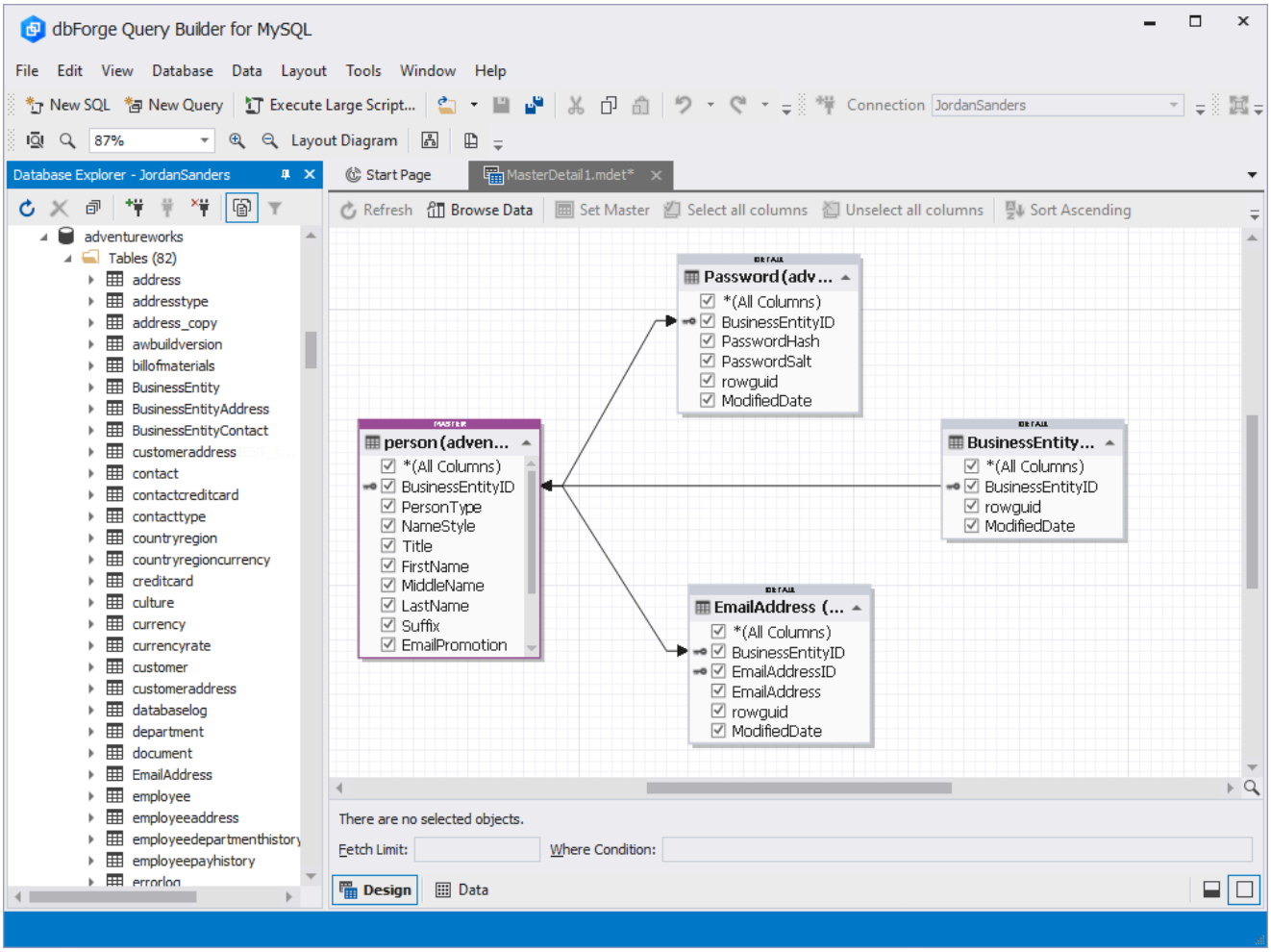
Data Reports

Data Report wizard supports building Data Reports and customizing them with a rich set of controls.



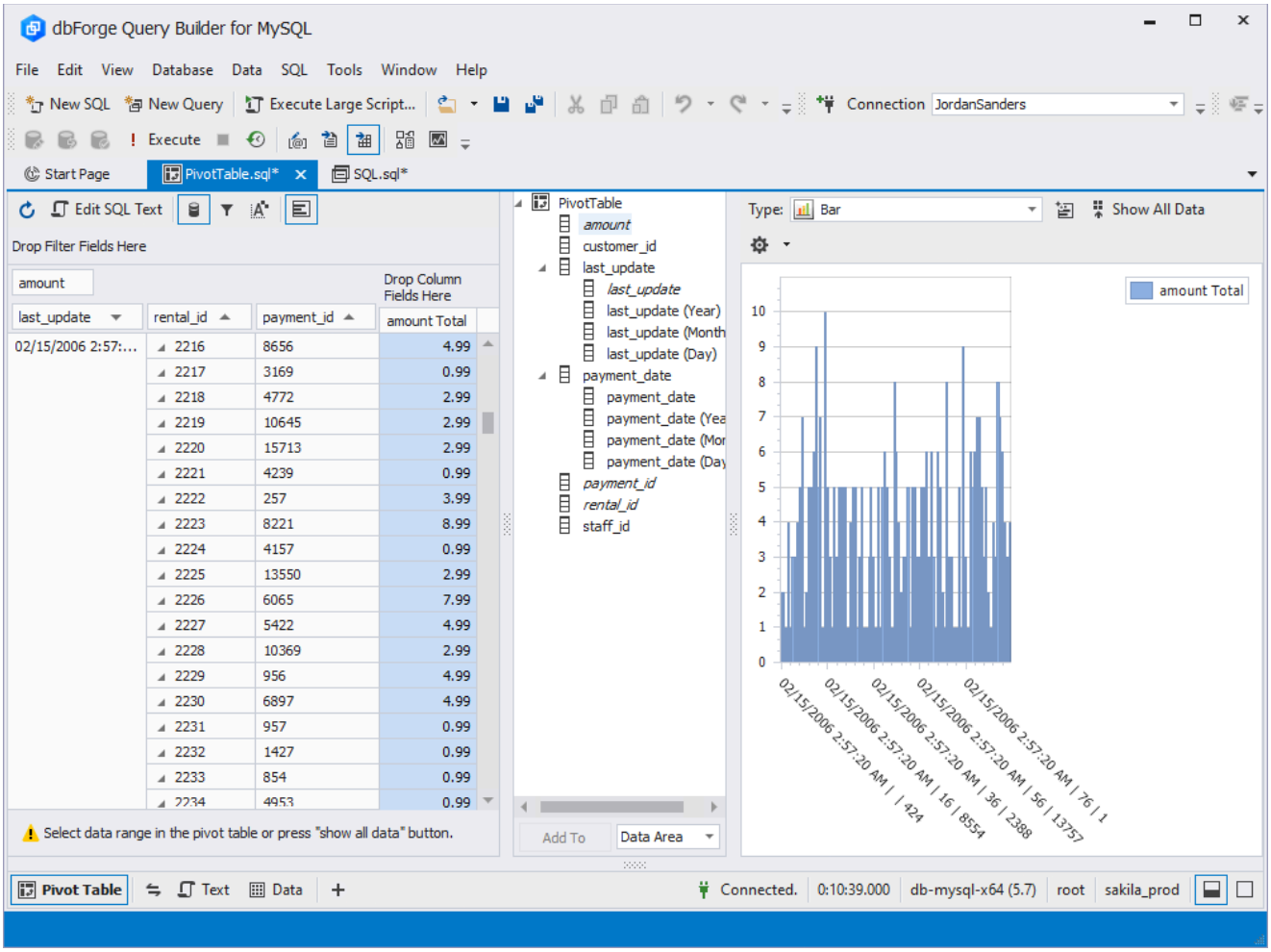
Master-Detail Browser

Allows users to view and analyze the data of several related database objects in a master-detail structured document. Edit data, build relationships, filter data based on specific criteria, sort data, etc. in the Design and Data views.



Pivot Tables

Let users manipulate data visually, including sorting, grouping, or filtering fields, building charts, and calculating totals.



Download a free 30-day trial of dbForge Query Builder for MariaDB here.

Documentation is available, too.

Version	Introduced
dbForge Query Builder 10.1	Support for the rds-ca-rsa2048-g1 SSL/TLS certificate for connecting to MariaDB servers on AWS and enhanced cloud compatibility features, new automation capabilities.
dbForge Query Builder 10.0	MariaDB 11.4, Implemented support for the `JSON_TABLE()` table function in MariaDB.
dbForge Query Builder 5.2	MariaDB 11.3.
dbForge Query Builder 5.1	MariaDB 10.9, MariaDB 10.10.
dbForge Query Builder 4.5	MariaDB 10.5.

This page is licensed: CC BY-SA / Gnu FDL

dbForge Schema Compare

> **Note:**
>
> dbForge Schema Compare is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

dbForge Schema Compare is a solution that allows for the comparison of the MariaDB database structure. With this tool, you can find the differences in MariaDB database schemas.

Features are described in the following.

MariaDB & MySQL Schema Synchronization

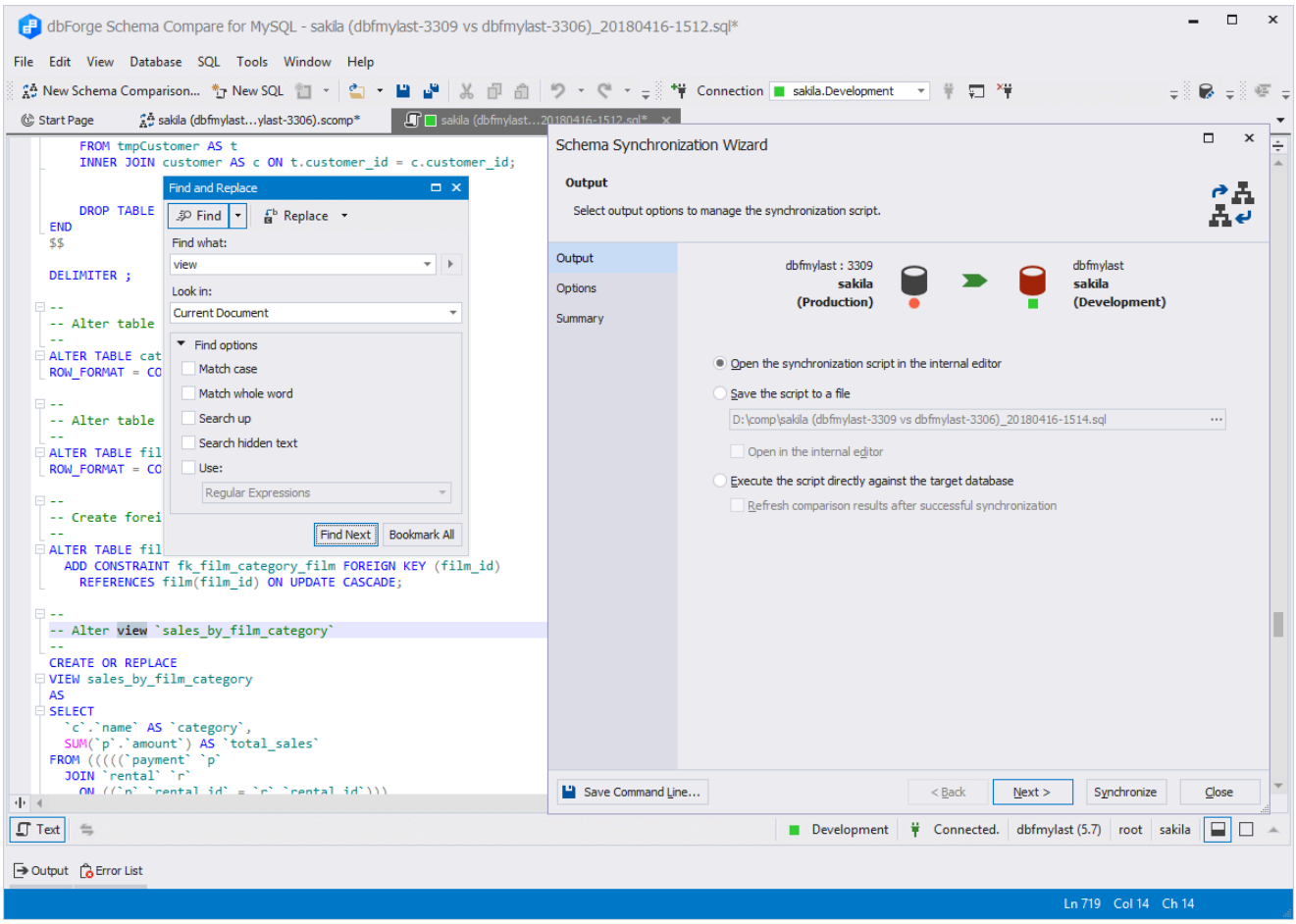
Update MariaDB schemas by generating accurate synchronization scripts.

Preview sync scripts for any schema objects.

Customize the synchronization output with various options.

Get warnings on possible errors while synchronization.

Sync asynchronous structure.



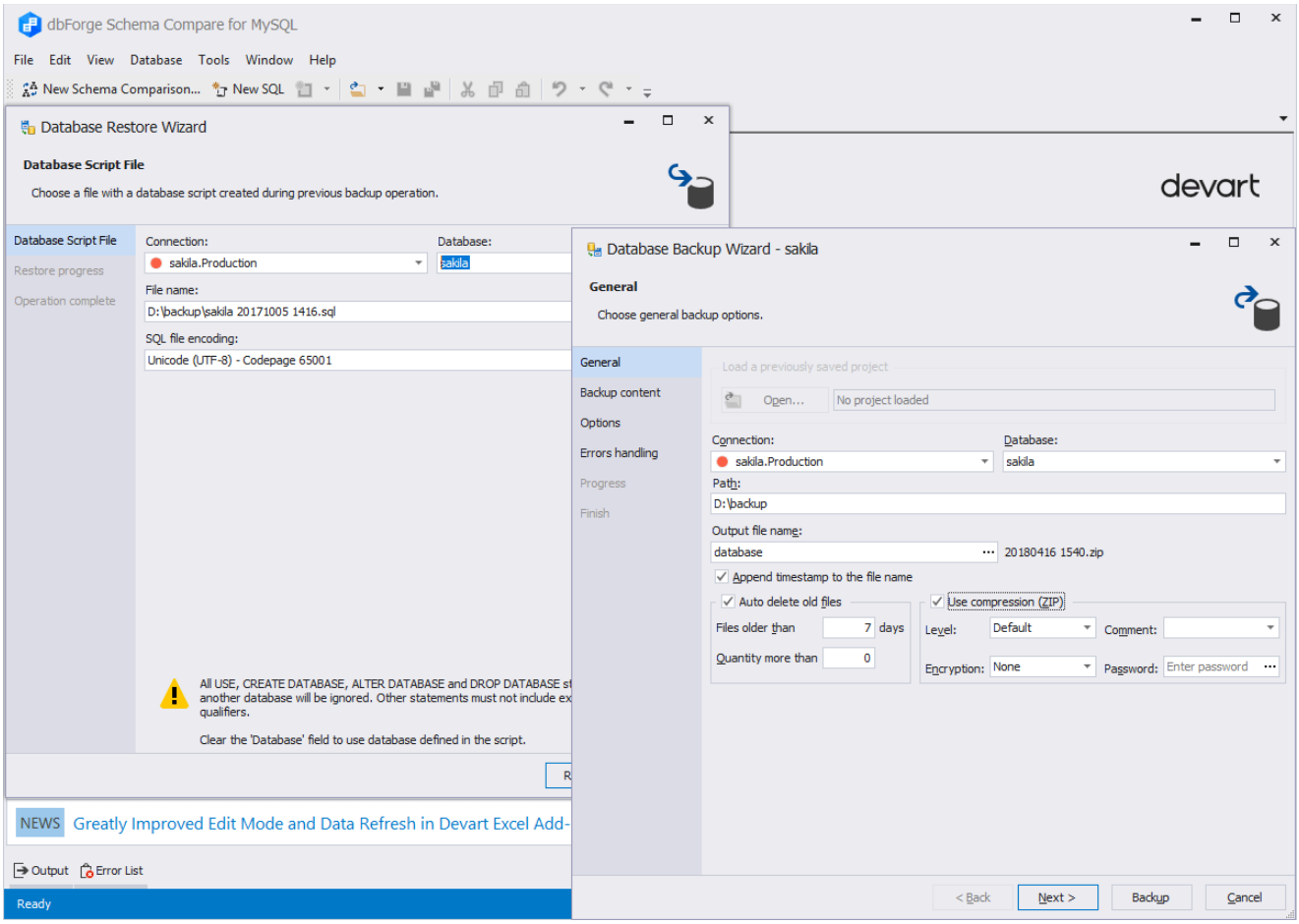
MariaDB Backup and Restore

Use the advantages of MariaDB backup and restore Wizards.

Create a full or partial backup.

Restore database and schedule backups with a command line interface.

Support for databases over 2Gb.

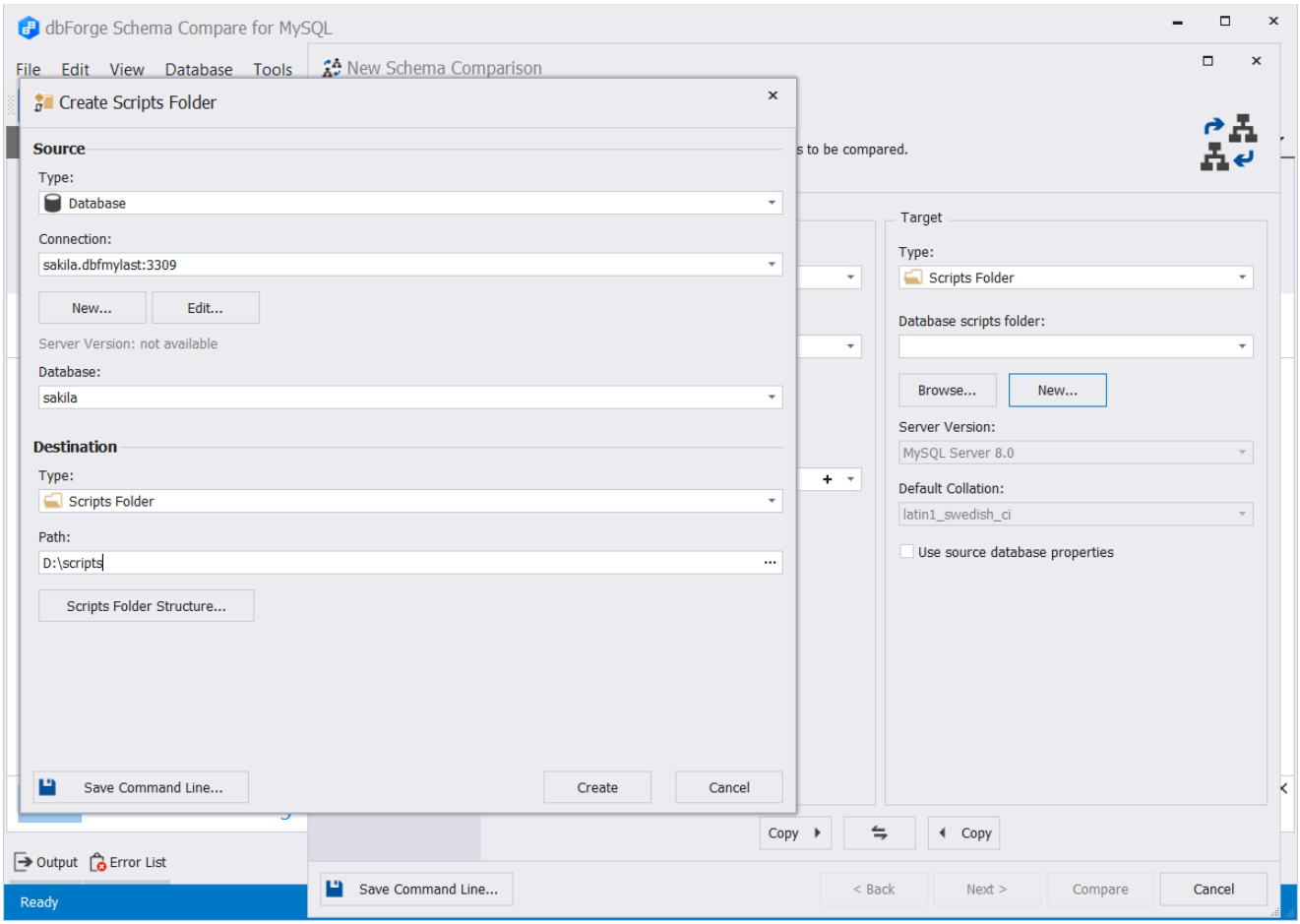


Scripts Folder Comparison

Compare the data both in two different databases and the database data with the data placed locally.

Choose any connection and script folder stored locally with the new Data Comparison Wizard.

View the comparison process results in a data grid.

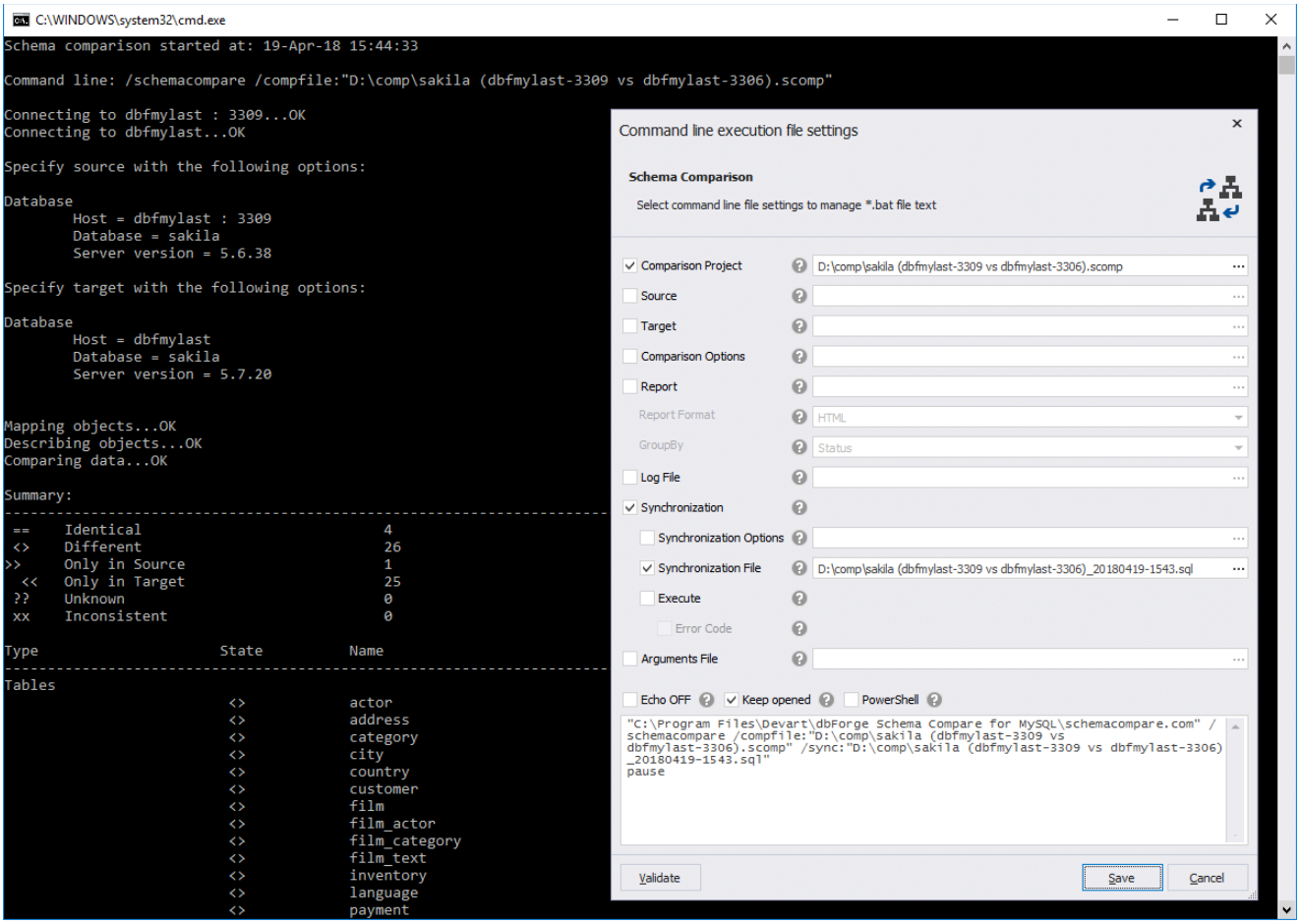


Routine Tasks Automation

Sync and compare database schemas with command line support.

Use the Windows Scheduler to automate the sync processes.

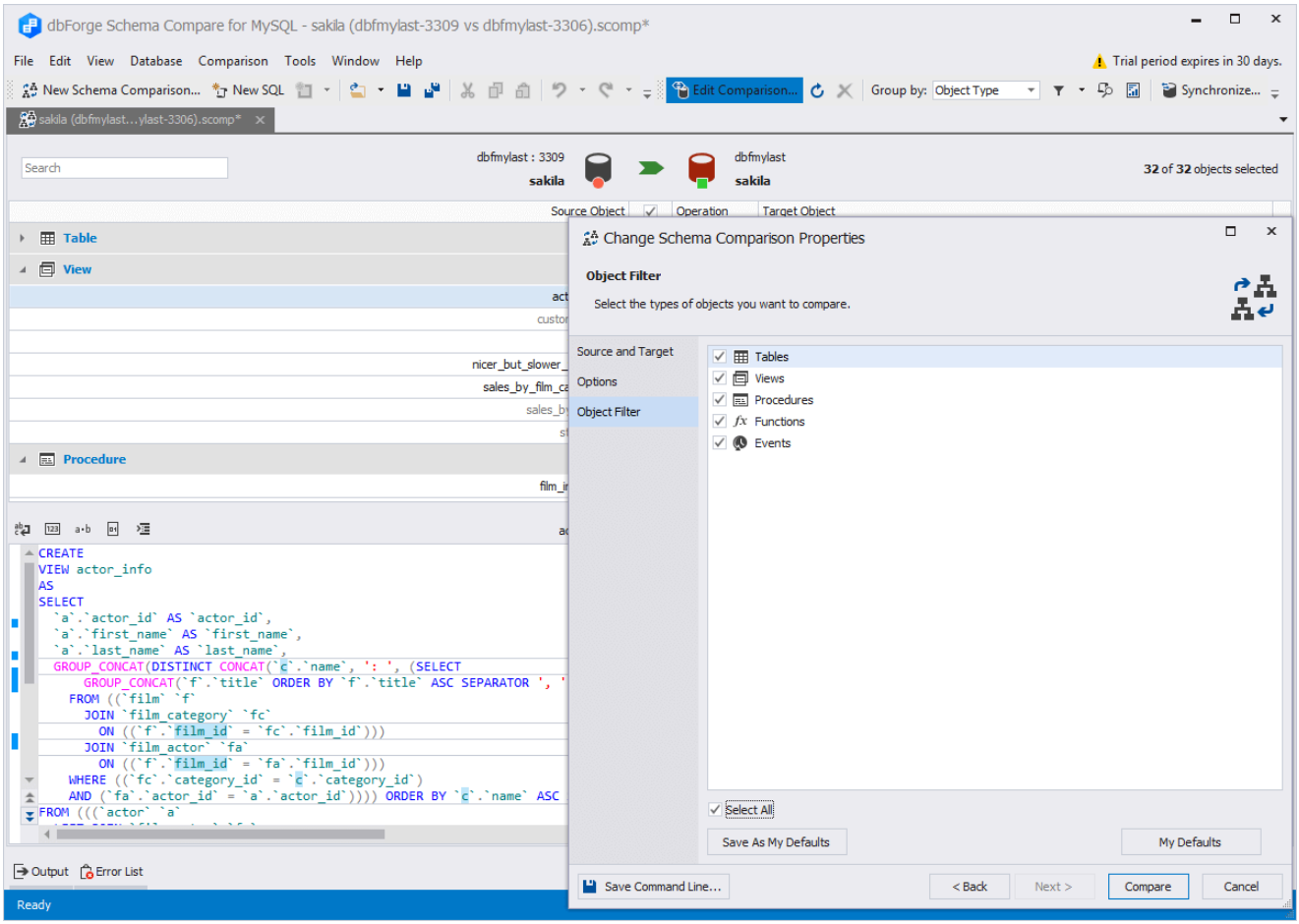
Run synchronization and schema comparison tasks with command line execution files.



Supported Schema Objects

Packages.

Sequences.

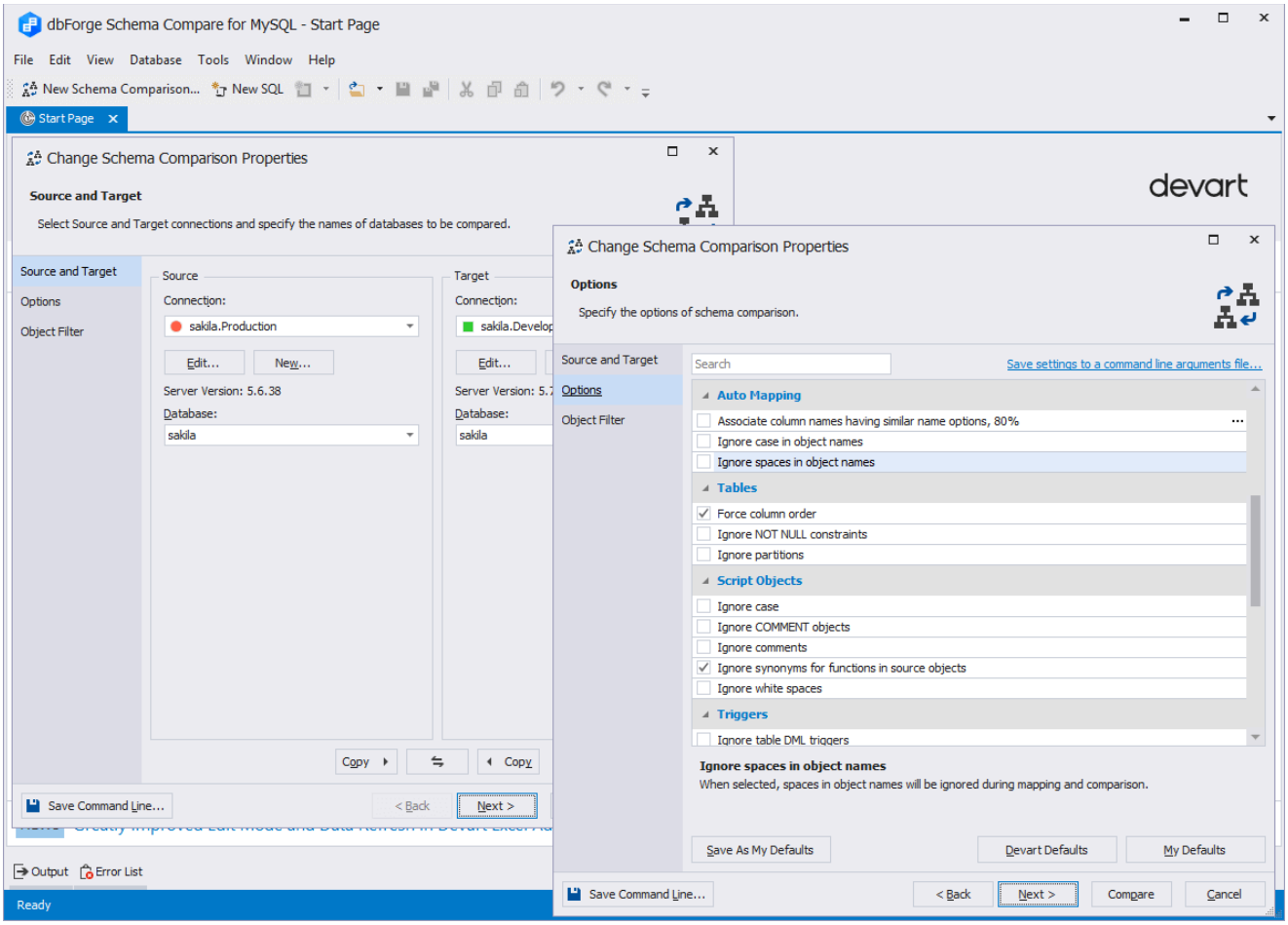


Database Structure Comparison Configuration

Filter the types of objects before the comparison.

Dismiss particular table options while comparing.

Use the Schema Comparison Wizard for switching between the Target and Source databases.

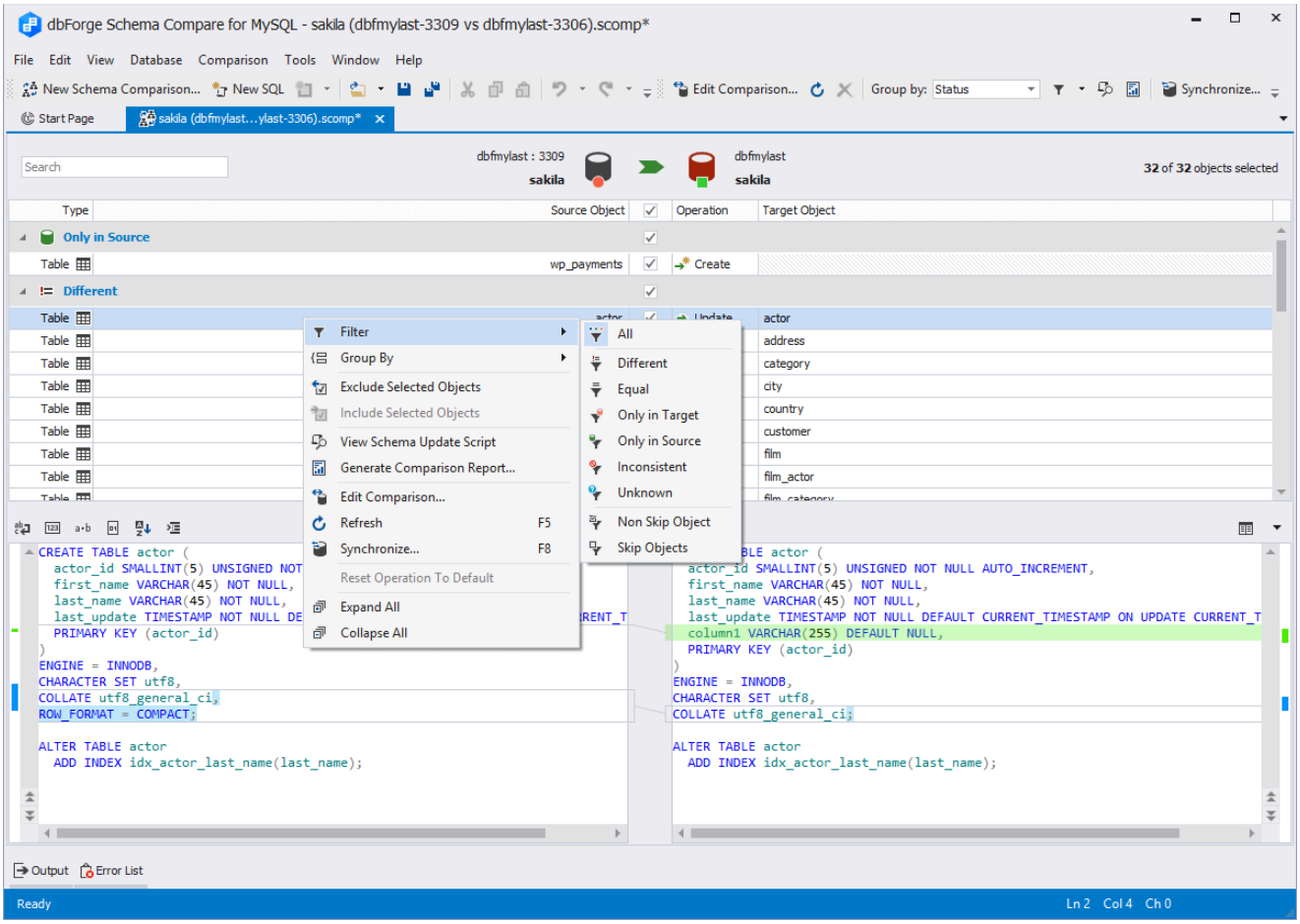


Comparison Results Analysis

Sort and filter the compared objects.

Get a clear view of MariaDB and MySQL schema diffs in the grid.

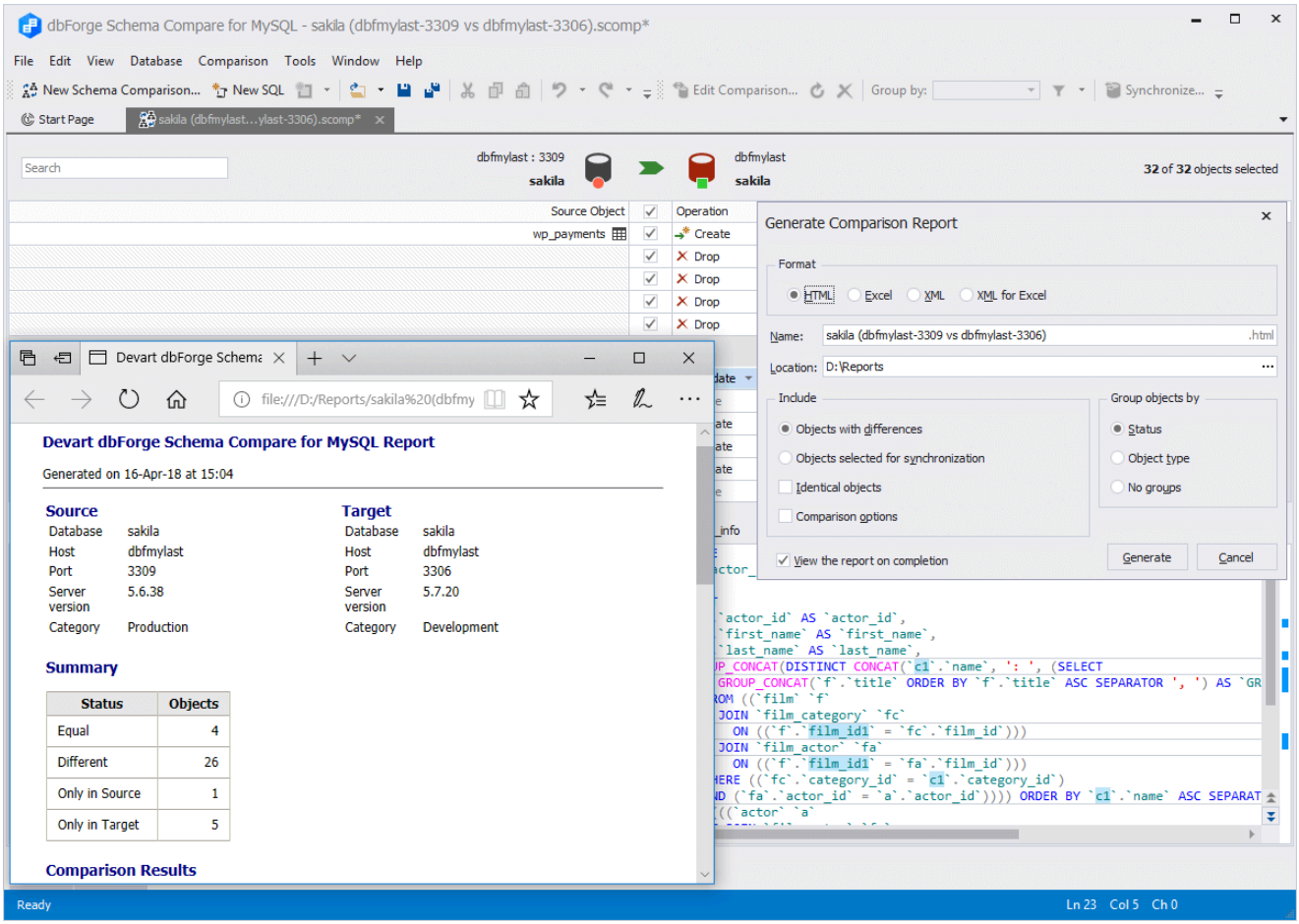
Manage groups of compared objects.



Comparison Results Export

Generate comparison results reports in HTML, Excel XML, XML for Excel.

Include specifically selected objects in your comparison report.



Compatibility

All MariaDB instances set up on Windows, Linux, FreeBSD, etc.

Various cloud services: Amazon RDS, Amazon Aurora, Google Cloud, Oracle MySQL Cloud, Alibaba Cloud.

Security connections: Secure Socket Layer (SSL), Secure Shell (SSH), HTTP Tunneling, PAM Percona.

Download a free 30-day trial of dbForge MariaDB and MySQL Schema Compare here.

Documentation is available, too.

Version	Introduced
dbForge Schema Compare 10.1	Support for the rds-ca-rsa2048-g1 SSL/TLS certificate for connecting to MariaDB servers on AWS and enhanced cloud compatibility features, new automation capabilities.
dbForge Schema Compare 10.0	MariaDB 11.4, Added support for temporal tables in MariaDB.
dbForge Schema Compare 5.2	MariaDB 11.3.
dbForge Schema Compare 5.1	MariaDB 10.9, MariaDB 10.10.
dbForge Schema Compare 5.0	MariaDB 10.5.
dbForge Schema Compare 4.4	MariaDB 10.4.

This page is licensed: CC BY-SA / Gnu FDL

dbForge Studio

> **Note:**
>
> dbForge Studio is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

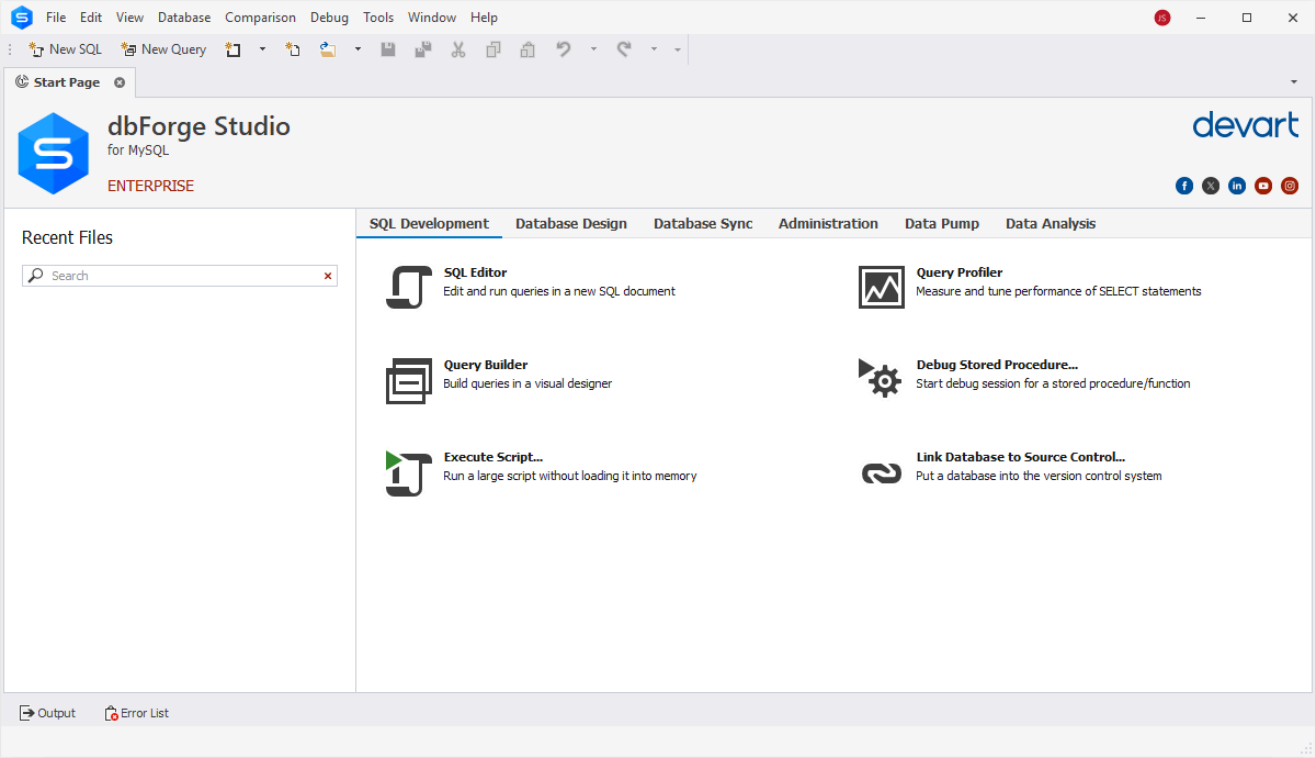
dbForge Studio for MariaDB is a universal IDE with GUI tools that has all the necessary built-in capabilities to work with MariaDB and MySQL databases for their development, management, and administration. It allows for creating, managing, and editing the data without the need to store them locally.

This GUI tool offers a rich visual design that is perfect for working with large scripts, preparing data reports, and database projects. At any time, you can back up or restore your data, export or import them to and from the most commonly used formats, and compare or synchronize the MariaDB databases.

Features are described in the following.

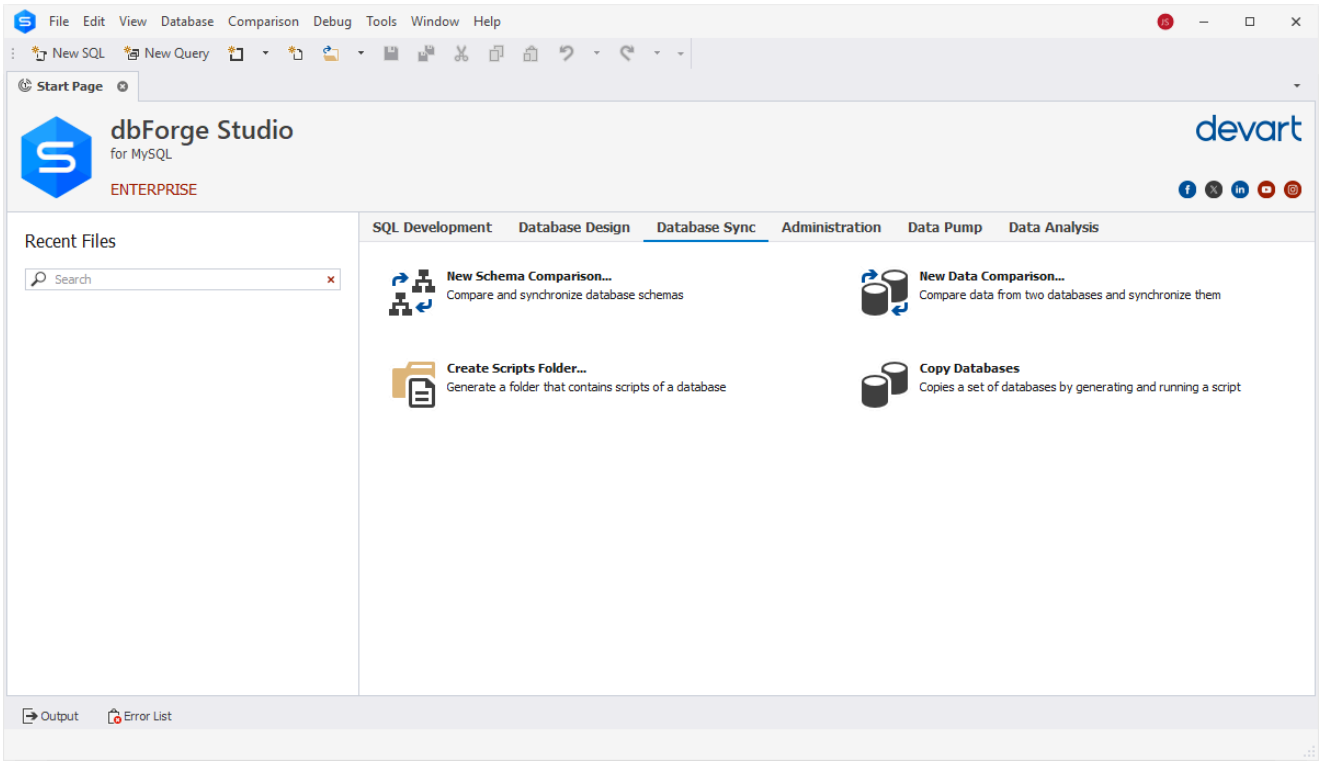
Intelligent SQL Coding

- Automatic code completion.
- MariaDB syntax highlighting.
- Code refactoring and formatting.
- CRUD generation.



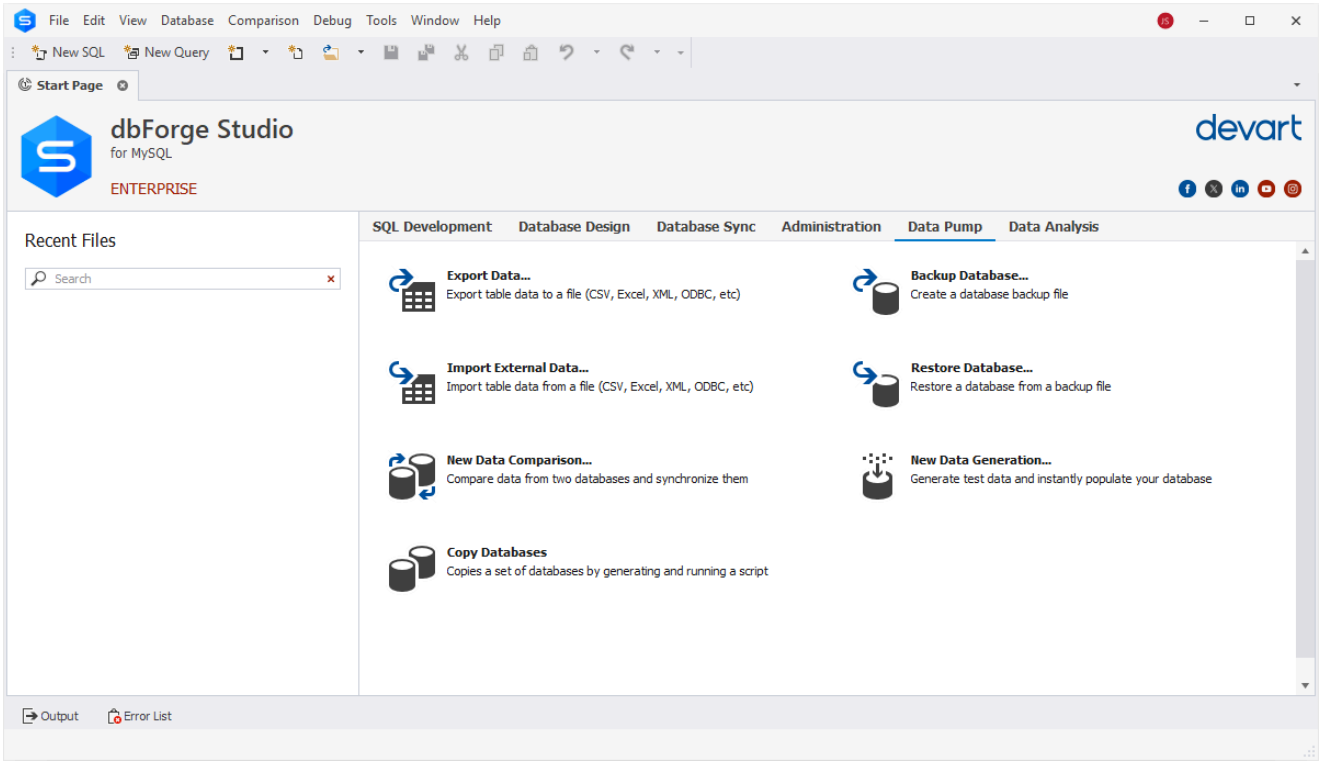
Database Compare and Sync

- Data and schema synchronization and comparison.
- Recurring database sync tasks planning.
- Comparison report generation.



Import and Export Data

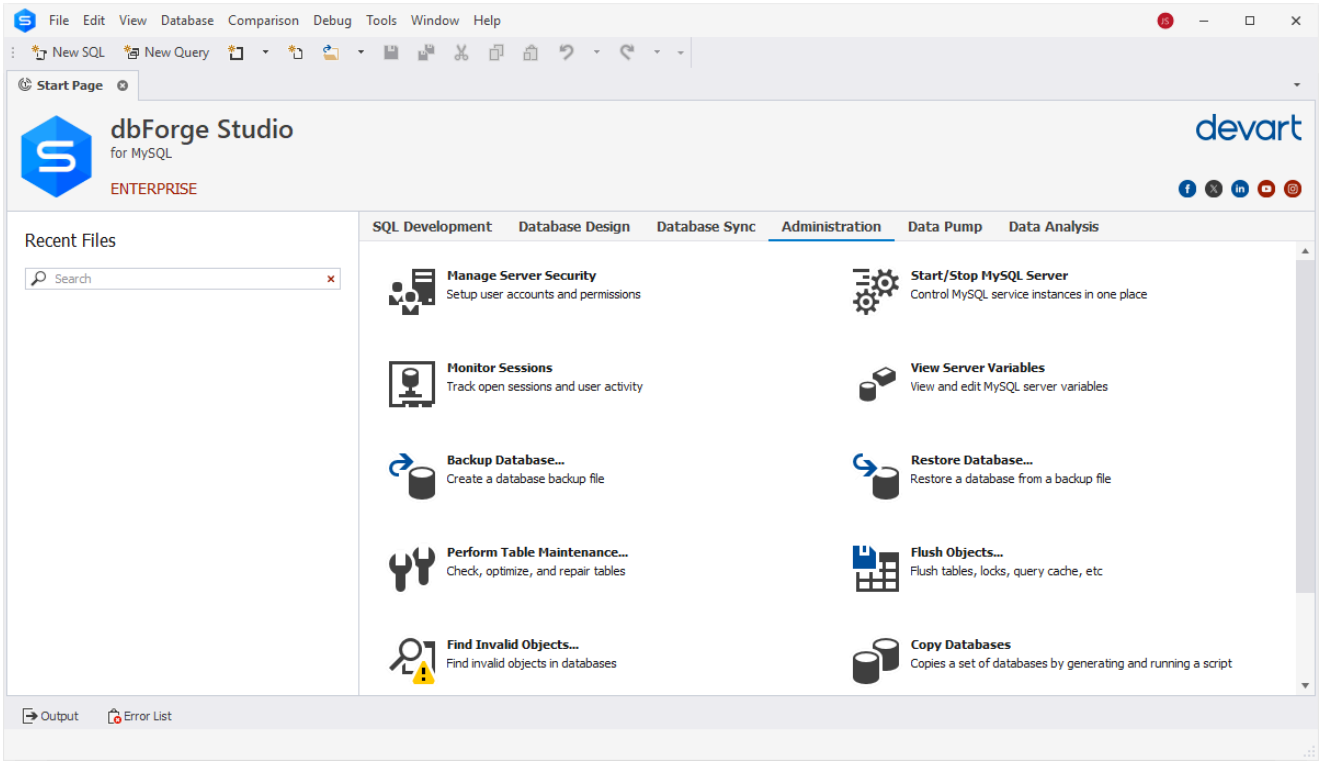
- MariaDB data export to commonly used formats.
- Data import from popular formats into MariaDB table.
- Rich customization ability.
- Command-line automation for data tasks.



MariaDB Administration Tools

- MariaDB databases backing up and restoring.

- User accounts and permissions configuration.
- Table maintenance.
- Database scripts generation.

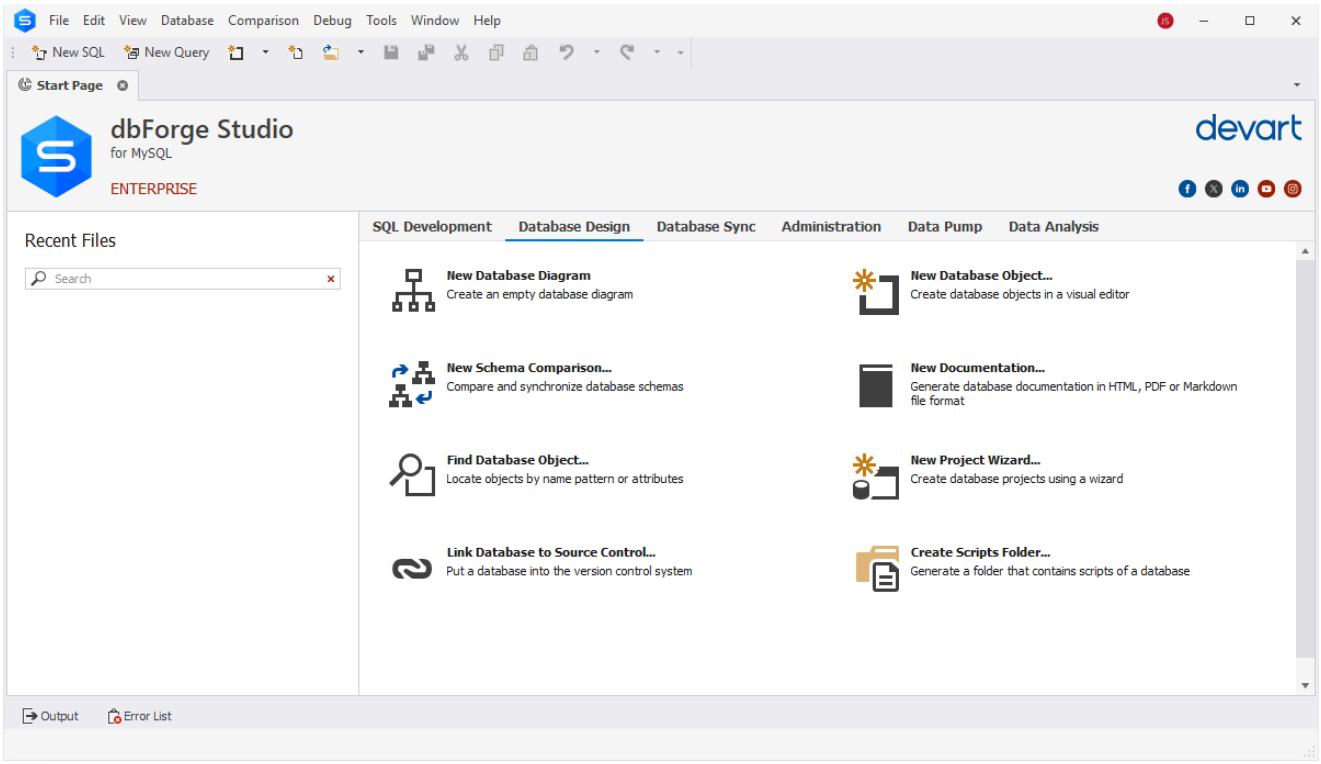


Copy Database

- Source and Target server selection.
- Database to be copied or dropped selection.
- Configuration of the database copying parameters.
- Progress of copying tracing and abortion of the copying.

Database Designer

- Schema diagrams generation.
- Visual database construction.
- Foreign key relations between tables.



Data Generator

- Data customization supported by multiple generators.
- All kinds of generators: basic, meaningful, and user-defined.
- Real-time preview of generated data.
- Command-line interface.

Query Profiler

- Visual query profiling.
- Profiling results comparison.

Visual Query Builder

- Automatic generation of the script template for queries.
- Easy navigation through the database objects.
- Support for different query types.
- Support for sequence tables.

Table Designer

- Table creation and editing in MariaDB with no code.
- Automatic data type setting for frequently-used column values.
- Errors prevention while working with tables.
- MariaDB script automatic generation according to the changes made by a user.

Database Refactoring

- Renaming database objects with preview.
- Refactoring script.

Source Control

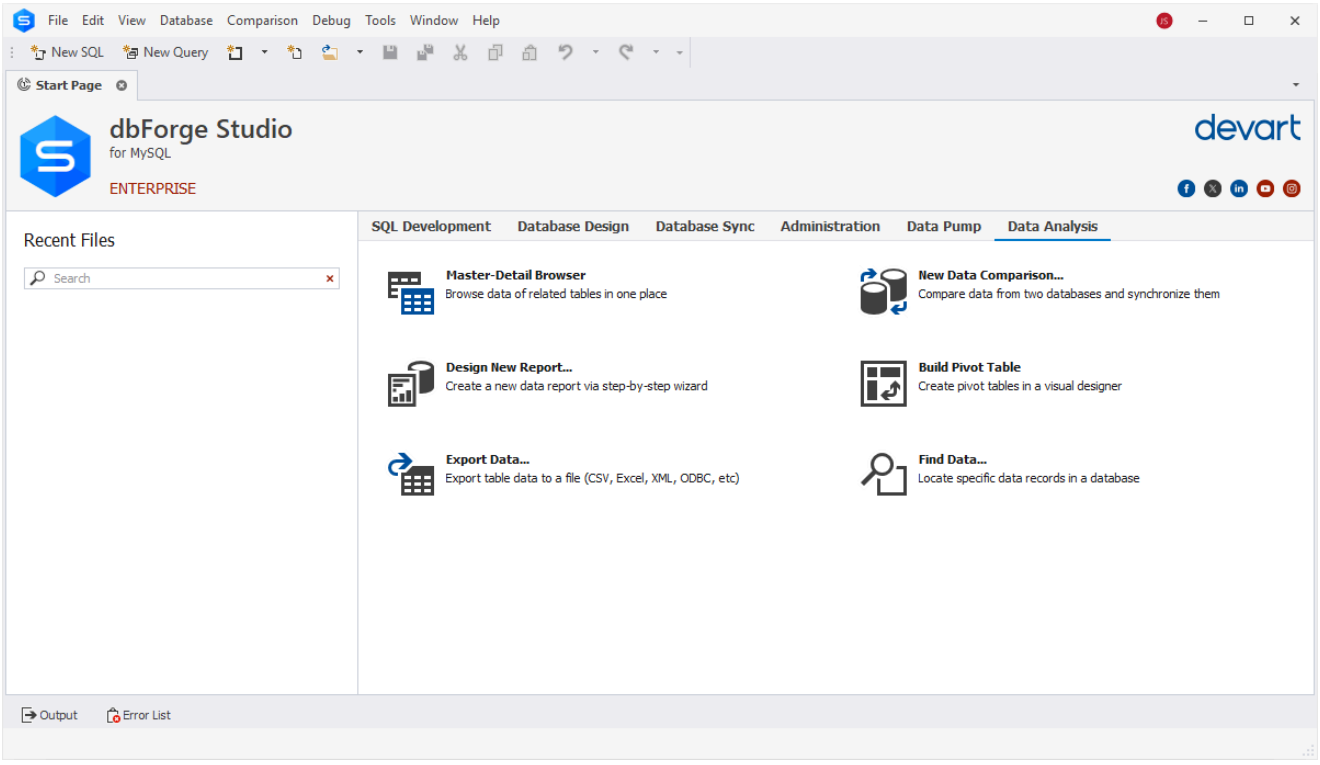
- Version control of database schemas and static table data.
- Compatibility with all major version control systems.
- Dedicated and shared development models.
- Quick comparison of database versions.
- Examination and resolution of conflicts.
- History of changes.

Database Projects

- MariaDB version control system providing efficient database team development.
- Scripts folder for exporting a database project to a MariaDB script file.

Report and Analysis

- Pivot tables for summarizing and viewing the data.
- A powerful wizard with robust features.
- Full command line support.
- 9 formats for reports' delivery.



Database Documenter

- A comprehensive overview of the database structure.
- Examine the internal dependencies between objects and databases.
- Customize the style and generate the docs in several formats.

Support for MariaDB Objects

- Packages.
- Sequences.
- Temporal tables.

This database tool is one of the most appreciated worldwide and trusted by many people. Devart's dbForge Studio has received a lot of awards that confirm its overwhelming popularity.

Download a free 30-day trial of dbForge Studio for MariaDB and MySQL here.

Documentation is available, too.

Version	Introduced
dbForge Studio for MySQL 10.1	Support for the rds-ca-rsa2048-g1 SSL/TLS certificate for connecting to MariaDB servers on AWS and enhanced cloud compatibility features, new automation capabilities.
dbForge Studio for MySQL 10.0	Support for MariaDB 11.4, added Source Control, support for temporal tables in MariaDB.
dbForge Studio for MySQL 9.2	Support for MariaDB 11.3, enhanced code completion for application-period temporal tables, support for sequence tables in MariaDB databases
dbForge Studio for MySQL 9.1	Connectivity support for MariaDB 10.9 and 10.10.
dbForge Studio for MySQL 9.0	Connectivity support for MariaDB 10.5 and 10.6.
dbForge Studio for MySQL 8.1	Support for MariaDB 10.4.

This page is licensed: CC BY-SA / Gnu FDL

DbSchema

> **Note:**
>
> DbSchema is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

DbSchema helps you design, document and manage databases. Design new tables, generate HTML5 documentation, explore and edit the database data, compare and synchronize the schema over multiple databases, edit and execute SQL, and generate random data.

DbSchema is compatible with all relational and some No-SQL databases. It works on all major operating systems, including Windows, Linux and macOS.

You can download and evaluate DbSchema 15 days for free.

DBSchema Webpage

This page is licensed: CC BY-SA / Gnu FDL

DbVisualizer

> **Note:**
>
> DbVisualizer is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

DbVisualizer is a cross-platform database tool for developers, DBAs, analysts and SQL programmers. Both Free and paid Pro versions are available. Supported databases include MySQL, MariaDB, and PostgreSQL.

DbVisualizer has been carefully developed by a small and dedicated team. Development decisions are based on user feedback.

Example of features:

- Simple navigation of database objects and their properties.
- Integrated AI Assistant, opt-in to ask questions, explain errors and analyze your code using a simple chat interface.
- Version control using Git, track and share database scripts, collaborate, clone, switch branches, push changes and explore history.
- Visual rendering of primary/foreign keys.
- Table data editing in spreadsheet with Export and import database schema.
- Flexible user interface in light and dark themes.
- Query optimization with an explain plan feature.
- Visual query builder using drag and drop.
- Flexible SQL scripts execution with parameter support.
- SQL formatting and Command-line interface for headless execution.
- All features.

This page is licensed: CC BY-SA / Gnu FDL

DeZign for Databases

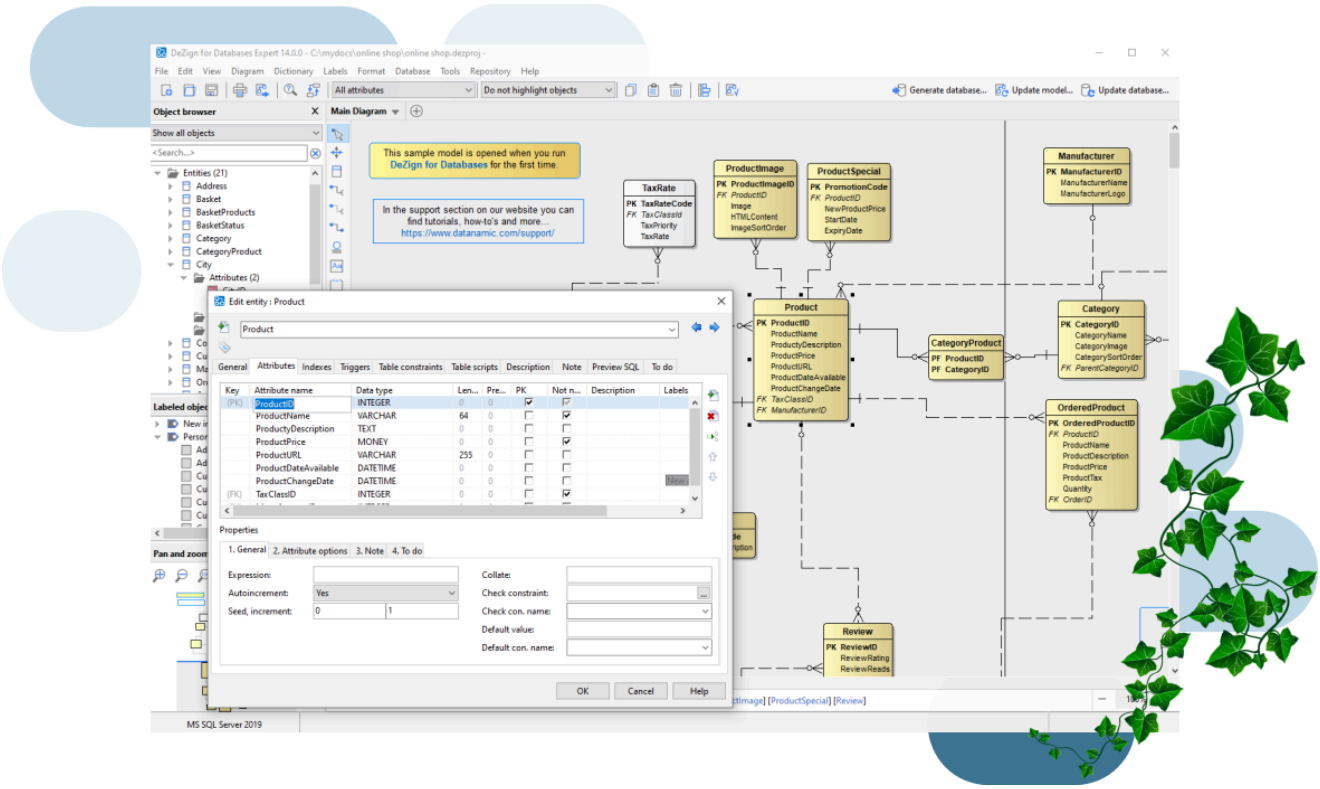
> **Note:**
>
> DeZign for Databases is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

DeZign for Databases is a data modeling tool for developers and DBA's that helps model, create, and maintain MariaDB databases. It runs on Windows only.

Supported database management systems include MariaDB, MySQL, and SQL Server.

Key features:

- Visual database modeling (Entity relationship diagrams).
- Generate database schemas directly from the data model.
- Incorporate changes from live databases into your existing models.
- Create change scripts by comparing your model with the live database structure.
- Publish and maintain data definitions to ensure consistency and accessibility.
- Convert models into SQL, streamlining database creation.
- Create clear and visually appealing ER diagrams, including sub-diagrams for complex systems.
- Support version control to manage different iterations of your model.



See www.datanamic.com for more information.

This page is licensed: CC BY-SA / Gnu FDL

ERBuilder Data Modeler

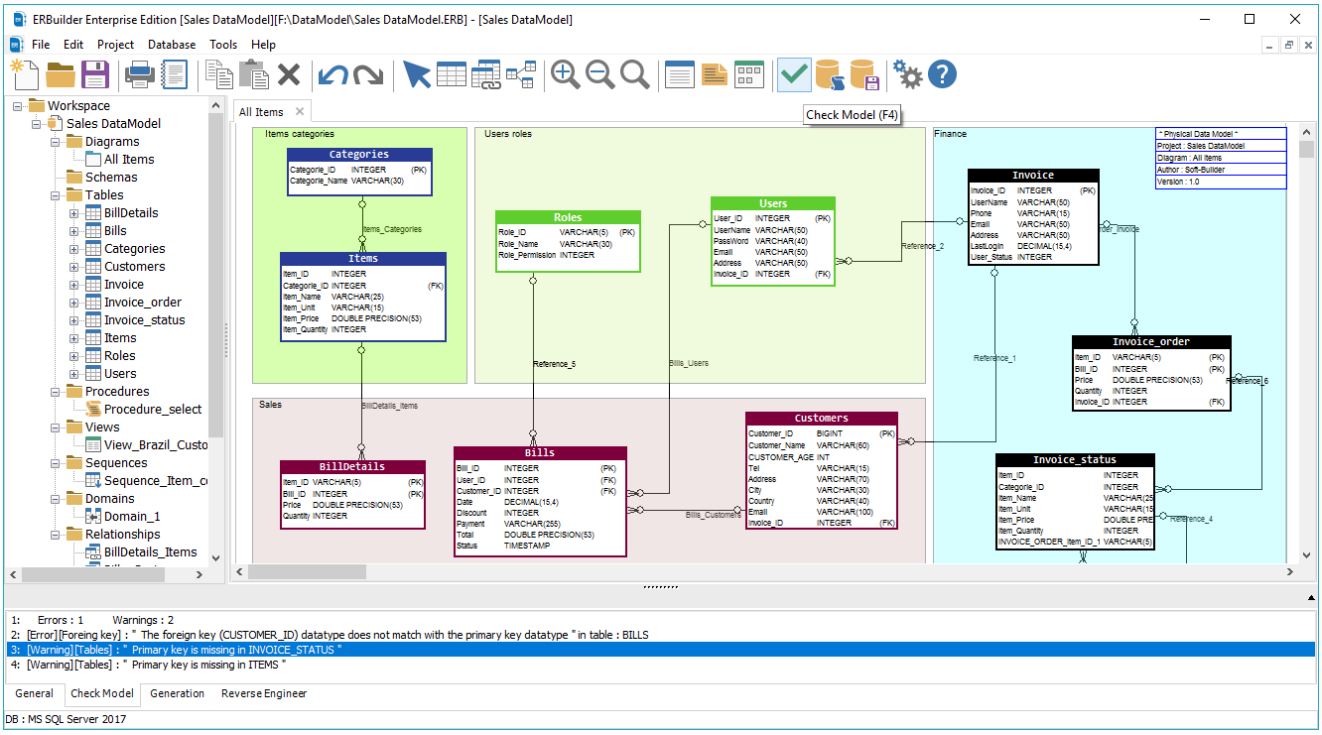
> **Note:**
>
> ERBuilder Data Modeler is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

ERBuilder Data Modeler is a GUI data modeling tool that allows developers to visualize, design, and model databases by using entity relationship diagrams and automatically generate the most popular SQL databases, and to generate and share the data model documentation. Optimize your data model by using advanced features such as test data generation, schema compare, and schema synchronization.

Supported DBMS include MariaDB, MySQL, and Microsoft SQL Server.

Key features:

- Visual data modeling.
- Forward and Reverse Engineering.
- Data Model Validation.
- Data model documentation.
- Schema Comparison and Synchronization.
- Test data generation.
- Change Database Platform.
- Generate web user interface for CRUD.
- Version management.



ERBuilder-main-screen

A feature-limited free version is available for download. Advanced features are available in the commercial edition. A feature comparison matrix is available for more details about features, pricing, and trial versions.

See www.soft-builder.com for more information.

This page is licensed: CC BY-SA / Gnu FDL

DBeaver

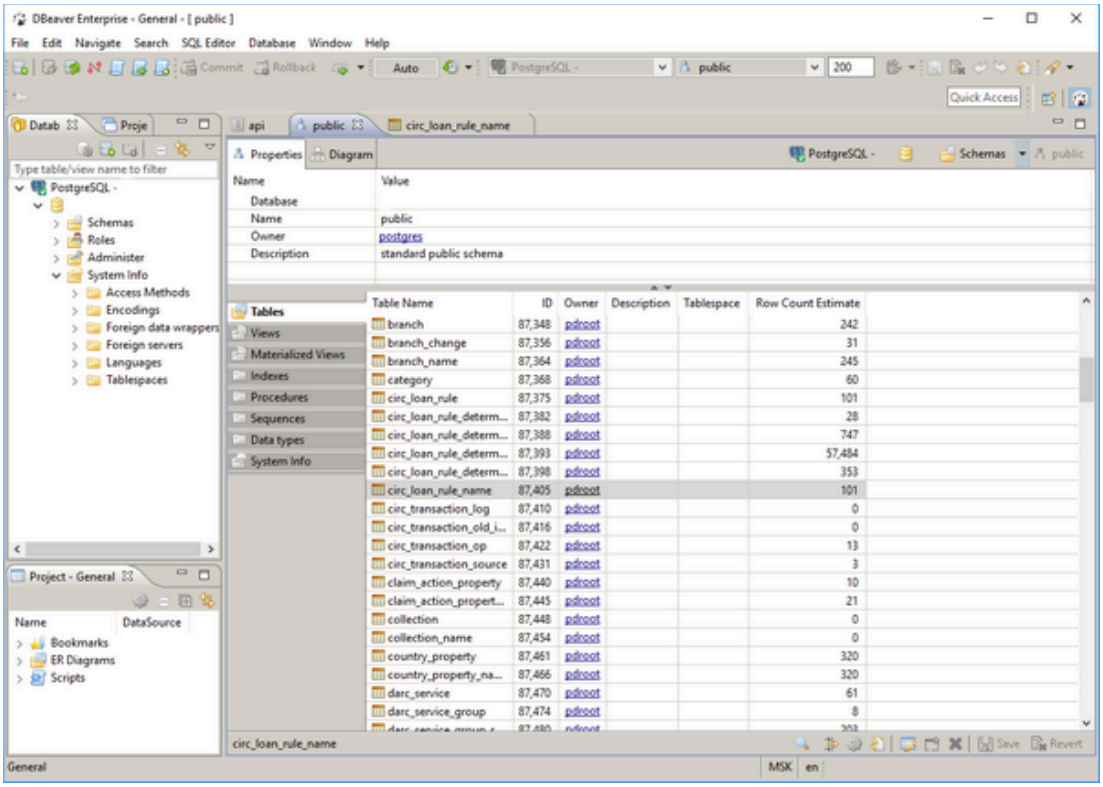
> **Note:**
>
> DBeaver is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

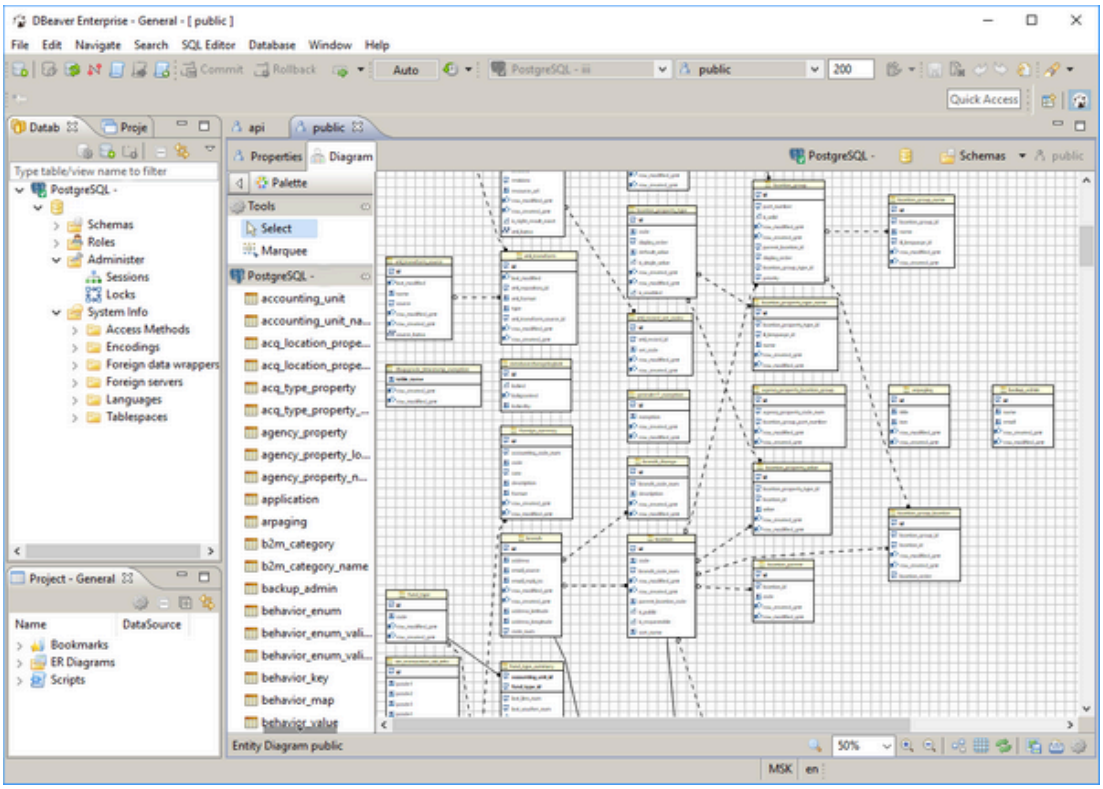
DBeaver is a free multi-platform database tool for developers, SQL programmers, database administrators and analysts. It supports many popular relational databases like MySQL, MariaDB, and PostgreSQL.

A list of basic features:

- Data viewer and editor: sorting, filtering, image displaying, export of selected data and much more.
- Metadata browser: possibility to view and edit existing tables, views, columns, indexes, procedures, triggers, storage entities (tablespaces, partitions, etc), security entities (users, roles, etc).
- Data transfer: export and import for files in various formats (CSV, HTML, XML, XLS, XLSX).
- ER diagrams: possibility to automatically generate ER diagrams for a database/schema (diagram will contain all schema tables) or for a single table and export the diagram in a suitable format.
- SQL editor: possibility to organize all your scripts in folders, reassign database connections for particular scripts.
- Data and metadata search: full-text data search using against all chosen tables/views.
- Database structure comparing: possibility to perform objects structure compare.

Usability is the main goal of this project, so the program UI is carefully designed and implemented. Users can send bug report and feature request on the GitHub page.





This page is licensed: CC BY-SA / Gnu FDL

OmniDB

> **Note:**
>
> OmniDB is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

OmniDB is a browser-based tool that simplifies MariaDB database management focusing on interactivity. It is designed to be powerful and lightweight.

Characteristics:

- Browser-based Tool: Accessible from any platform, using a browser as a medium.
- Responsive Interface: All available functions in a single page.
- Unified workspace: Different technologies managed in a single workspace.
- Simplified Editing: Easy to add and remove connections.
- Safety: Multi-user support with encrypted personal information.
- Interactive Tables: All functionalities use interactive tables, allowing copying and pasting in blocks.
- Smart SQL Editor: Contextual SQL code completion.
- Beautiful SQL Editor: You can choose between many available color themes.
- Tabbed SQL Editor: Easily add, rename or delete editor tabs.

More information can be found here.

This page is licensed: CC BY-SA / Gnu FDL

Sequel Pro

> **Note:**
>
> Sequel Pro is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Sequel Pro is a fast, database management application for working with MySQL and MariaDB databases. It runs on macOS only.

Sequel Pro is open source, so it's easy to be involved and enhance it for MariaDB.

If you have any issues with Sequel Pro on MariaDB, please use the issue tracking for that!

See Also

- Source code
- Issue tracking

This page is licensed: CC BY-SA / Gnu FDL

HeidiSQL

> **Note:**
>
> HeidiSQL is third-party software, not developed or maintained by MariaDB. It ships as an optional component of the MariaDB Windows installer, but MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

HeidiSQL is a Windows client for MariaDB and MySQL, and is bundled with the Windows version of MariaDB.

HeidiSQL Version	Introduced
HeidiSQL 12.6	MariaDB 11.3.2, MariaDB 11.2.3, MariaDB 11.1.4, MariaDB 11.0.5, MariaDB 10.11.7, MariaDB 10.6.17, MariaDB 10.5.24, MariaDB 10.4.33
HeidiSQL 12.3	MariaDB 11.0.1, MariaDB 10.11.3, MariaDB 10.10.4, MariaDB 10.9.6, MariaDB 10.8.8, MariaDB 10.6.13, MariaDB 10.5.20, MariaDB 10.4.29

HeidiSQL Version	Introduced
HeidiSQL 11.3	MariaDB 10.6.2, MariaDB 10.5.11, MariaDB 10.4.20, MariaDB 10.3.30, MariaDB 10.2.39
HeidiSQL 11.0	MariaDB 10.5.2, MariaDB 10.4.13, MariaDB 10.3.23, MariaDB 10.2.32, MariaDB 10.1.45, MariaDB 5.5.68
HeidiSQL 10.2	MariaDB 10.4.7, MariaDB 10.3.17, MariaDB 10.2.26, MariaDB 10.1.41, MariaDB 5.5.65
HeidiSQL 9.5	MariaDB 10.3.13, MariaDB 10.2.22, MariaDB 10.1.38, MariaDB 10.0.38, MariaDB 5.5.63
HeidiSQL 9.4	MariaDB 10.1.20, MariaDB 10.0.29, MariaDB 5.5.54
HeidiSQL 9.3	MariaDB 10.1.15, MariaDB 10.0.26, MariaDB 5.5.50
HeidiSQL 9.1	MariaDB 10.1.0, MariaDB 10.0.17, MariaDB 5.5.42
HeidiSQL 8.3	MariaDB 10.0.12, MariaDB 5.5.38
HeidiSQL 8.0	MariaDB 10.0.2, MariaDB 5.5.31
HeidiSQL 7.0	MariaDB 10.0.0, MariaDB 5.5.20

HeidiSQL can:

- Connect to multiple servers in one window.
- Connect to servers via command line.
- Create and edit tables, views, stored routines, triggers and scheduled events.
- Generate nice SQL-exports.
- Export from one server/database directly to another server/database.
- Manage user-privileges.
- Import text-files.
- Export table rows as CSV, HTML, XML, SQL, LaTeX and Wiki Markup.
- Browse and edit table-data using a comfortable grid.
- Bulk edit tables (move to db, change engine, collation etc.).
- Batch-insert ascii or binary files into tables.
- Write queries with customizable syntax-highlighting and code-completion.
- Pretty reformat disordered SQL.
- Monitor and kill client-processes.
- Find specific text in all tables of all databases of one server.
- Optimize and repair tables in a batch manner.

More information, including Screenshots of HeidiSQL are available at the HeidiSQL Website.

This page is licensed: CC BY-SA / Gnu FDL

KS DB Merge Tools

> **Note:**
>
> KS DB Merge Tools is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Overview

KS DB Merge Tools for MySQL and MariaDB is a diff & merge tool for MySQL and MariaDB, allowing to compare and sync both schema and data. It is a Freemium application - many features are exposed in the Free version (available for commercial use), some extended features are available only in the paid Standard version (in many cases can be provided at no cost for open source developers).

Product home page

Download

Use cases

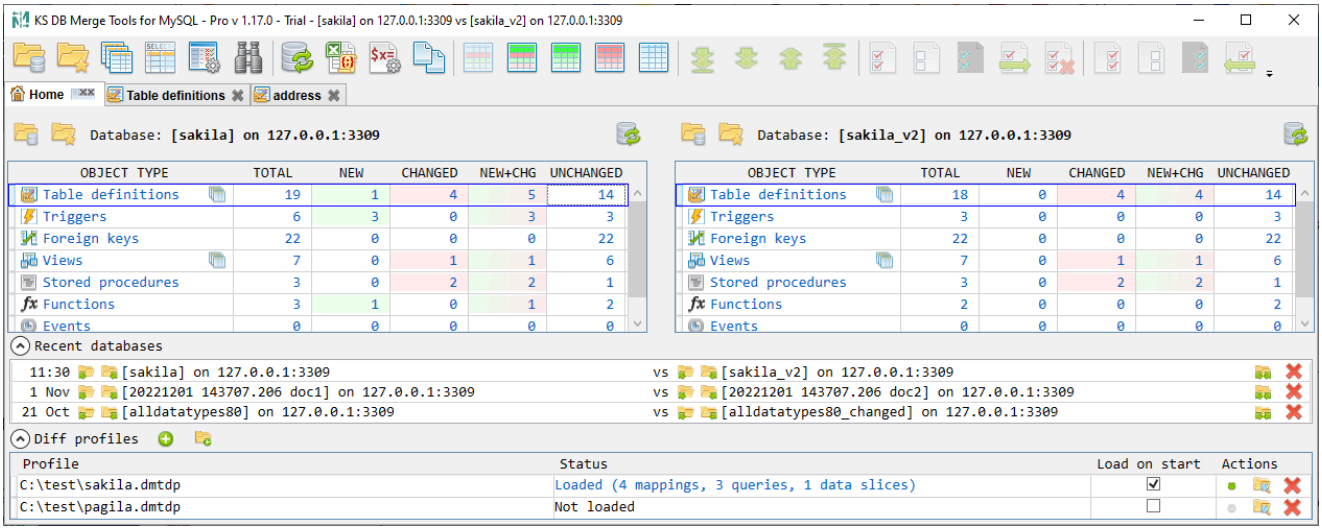
Documentation

Application has tabbed UI, there are several types of tabs responsible for particular application features and scope of tasks. Here are the primary application tabs:

Home Tab

It is a starting point to open databases. Shows summary about all database objects. Note that it does not provide information about data/content changes, only about object definitions.

This tab is also used as a starting point to manage diff profiles, making it easy to customize the tool for your specific database project. Here, you can save and reuse custom queries, mappings, and data slices, allowing you to create tailored data diff summaries.



Object List

Lists all objects of some particular type - tables, views, etc. Allows to identify whether some object is new, changed or unchanged (note that for tables and views it does not provide information about data/content changes, only about object definitions). Quick filters available to show only new/changed/new+changed objects. Here we can select required objects on one side and generate a synchronization script to merge these changes to the other side.

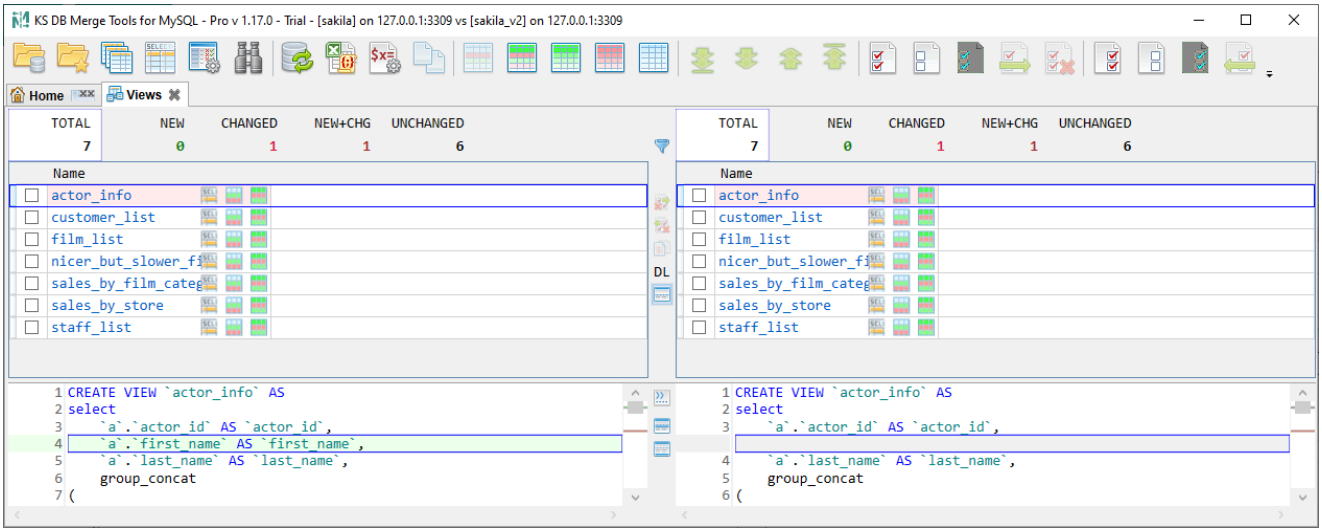
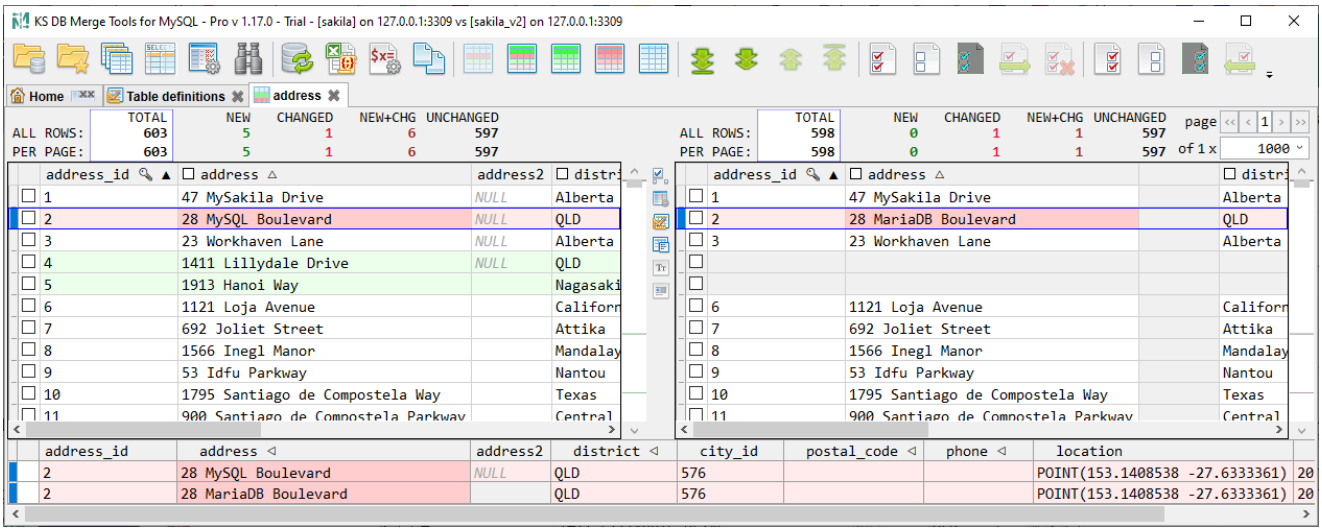


Table Data Diff

Compares data for particular table or view. Quick filters available to show only new, changed, or new and changed rows. We can select required rows on one side and generate synchronization script to merge these changes to the other side.



Text Diff

Compares definition of some particular database programming object like view or stored procedure.

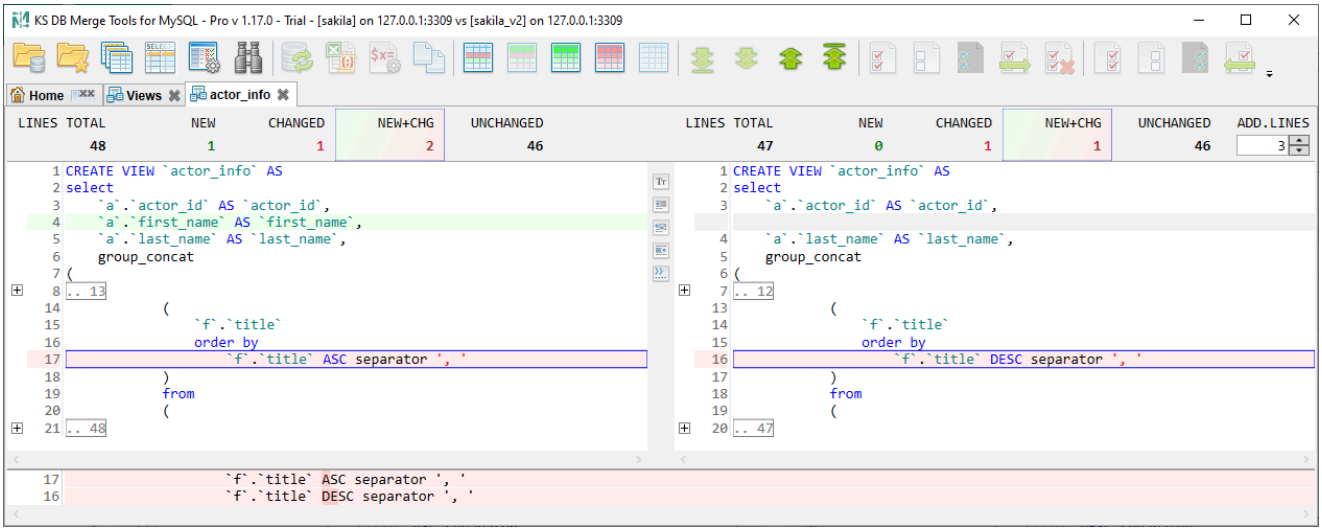
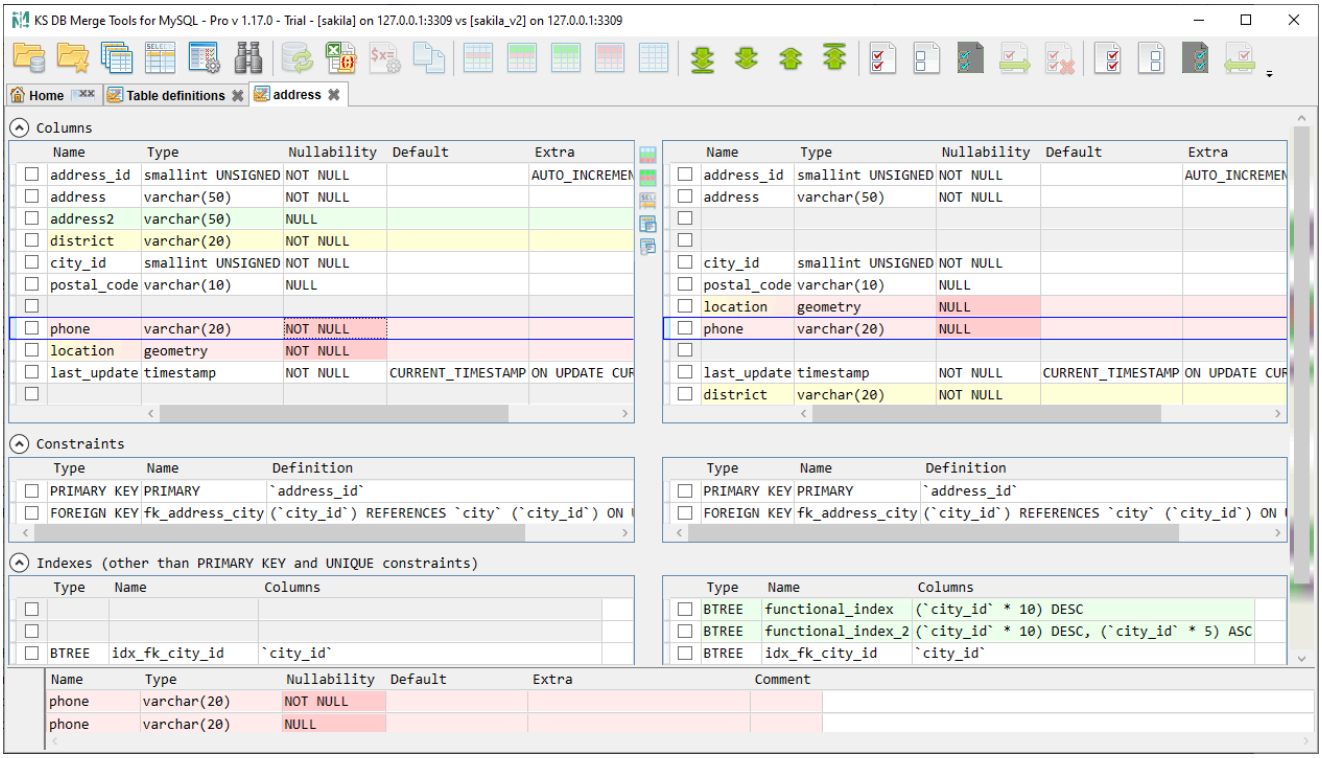


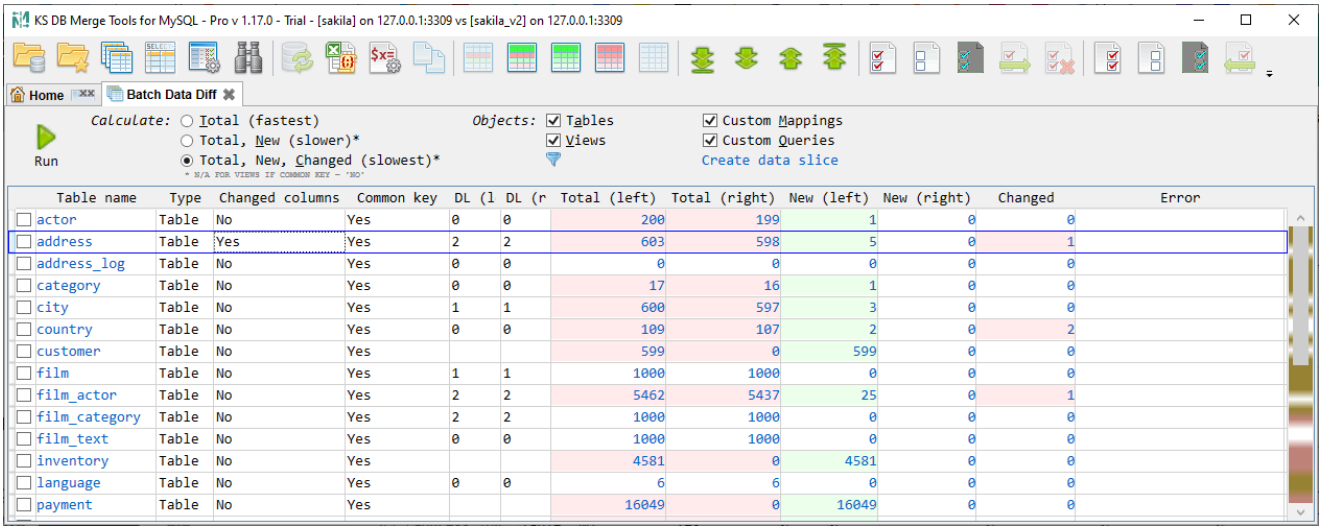
Table Structure Diff

Compares definition of particular table. Here we can select required changes and generate synchronization script for them. This tab is available only in the Standard version, the Free version is using Text diff tab to compare table definitions.



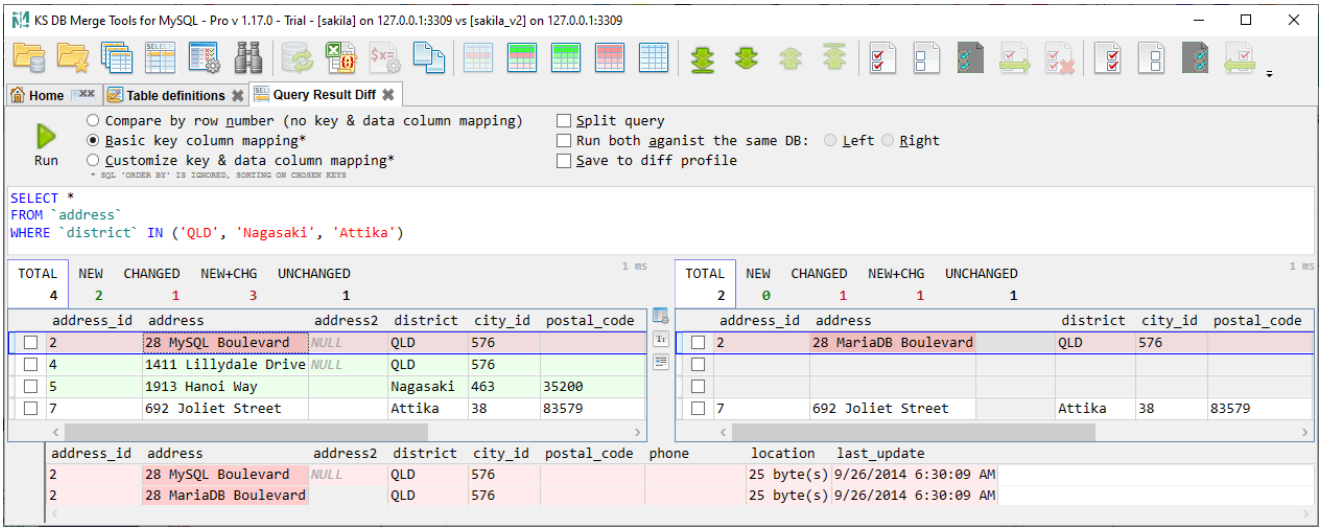
Batch Data Diff

Allows to compare data for multiple tables and views, providing summary of data changes for the whole database. This tab is available only in the Standard version.



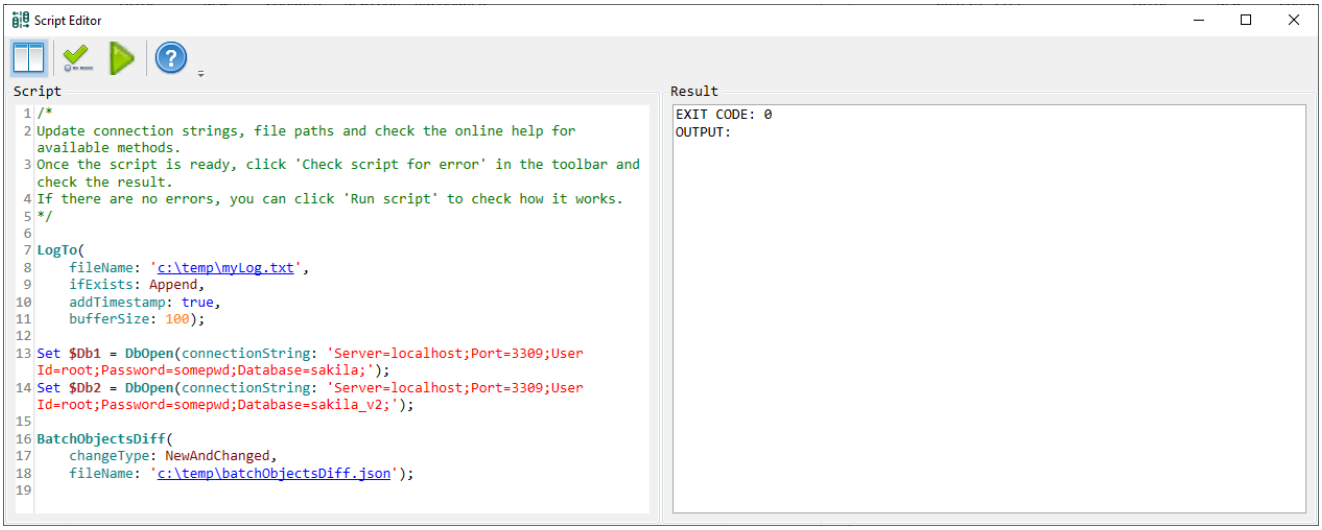
Query Result Diff

Compares arbitrary query results, it can be the same query running on both databases or different queries running on the same or different databases. This tab is available only in the Standard version.



Automation and Scripting

The Standard version has support of own domain-specific scripting language designed to automate diff and merge tasks provided by GUI. In addition to the primary GUI, there is a Script Editor application designed to help in writing and troubleshooting scripts and a separate command line utility that is used to run these scripts without user interaction. For the most typical tasks the scripts can be generated just with a single button click on GUI which produces a script relevant for the data or objects you currently observe on UI.



This page is licensed: CC BY-SA / Gnu FDL

LibreOffice Base

> **Note:**
>
> LibreOffice Base is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

LibreOffice Base is an open source RDBMS (relational database management system) frontend tool to create and manage various databases.

Preparing the ODBC Connection

First, make sure to prepare MariaDB Connector/ODBC as explained in MariaDB Connector/ODBC.

This includes

- downloading the latest MariaDB Connector/ODBC,
- copying the shared library libmaodbc.so to /usr/lib/[multi-arch],
- installing the unixodbc, unixodbc-dev, openssh-client, odbcinst packages, and
- creating a template file for the ODBC driver.
- Install the ODBC driver from the template file by running:

```
$ sudo odbcinst -i -d -f MariaDB_odbc_driver_template.ini
odbcinst: Driver installed. Usage count increased to 1.
    Target directory is /etc
```

Verify the installation was successful in `/etc/odbcinst.ini` file (this path is obtained by the config info */-j/* option, where drivers are installed in that predefined location).

```
$ odbcinst -j
unixODBC 2.3.6
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
USER DATA SOURCES..: /home/anel/.odbc.ini
SQLULEN Size.......: 8
SQLLEN Size........: 8
SQLSETPOSIROW Size.: 8

$ cat /etc/odbcinst.ini
[MariaDB ODBC 3.1 Driver]
Description=MariaDB Connector/ODBC v.3.1
Driver=/usr/lib/x86_64-linux-gnu/libmaodbc.so
UsageCount=1
```

- Create a template file for the Data Source Name (DSN). A sample “MariaDB_odbc_data_source_template.ini” could be:

[MariaDB-server]
Description=MariaDB server
Driver=MariaDB ODBC 3.1 Driver
SERVER=localhost
USER=anel
PASSWORD=
DATABASE=test
PORT=3306

- Install data source:

```
odbcinst -i -s -h -f MariaDB_odbc_data_source_template.ini
```

- Verify successful installation in the /.odbc.ini file

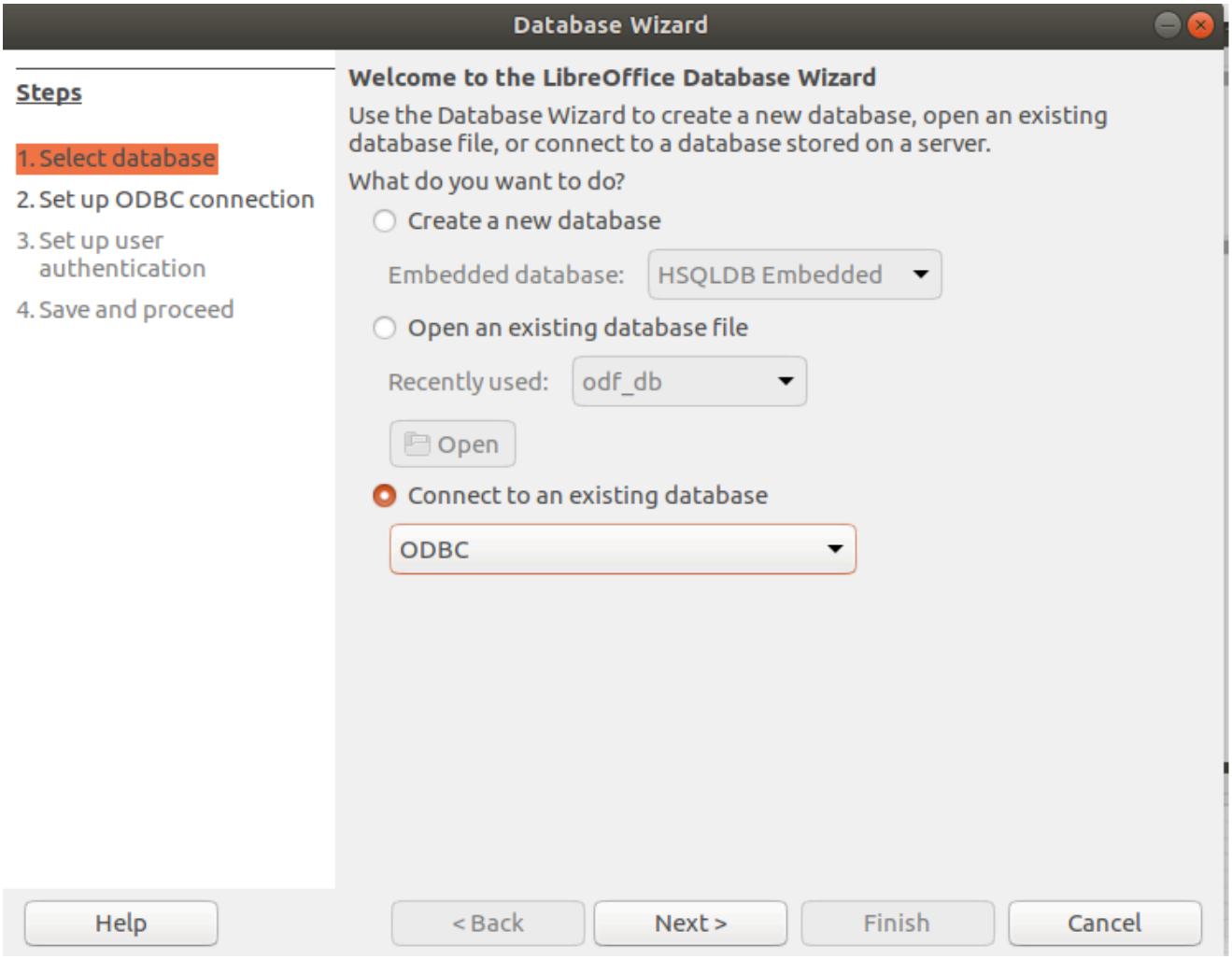
```
$ cat ~/.odbc.ini
[MariaDB-server]
Description=MariaDB server
Driver=MariaDB ODBC 3.1 Driver
SERVER=MariaDB
USER=anel
PASSWORD=
DATABASE=test
PORT=3306
```

- Verify successful installation also using the isql utility:

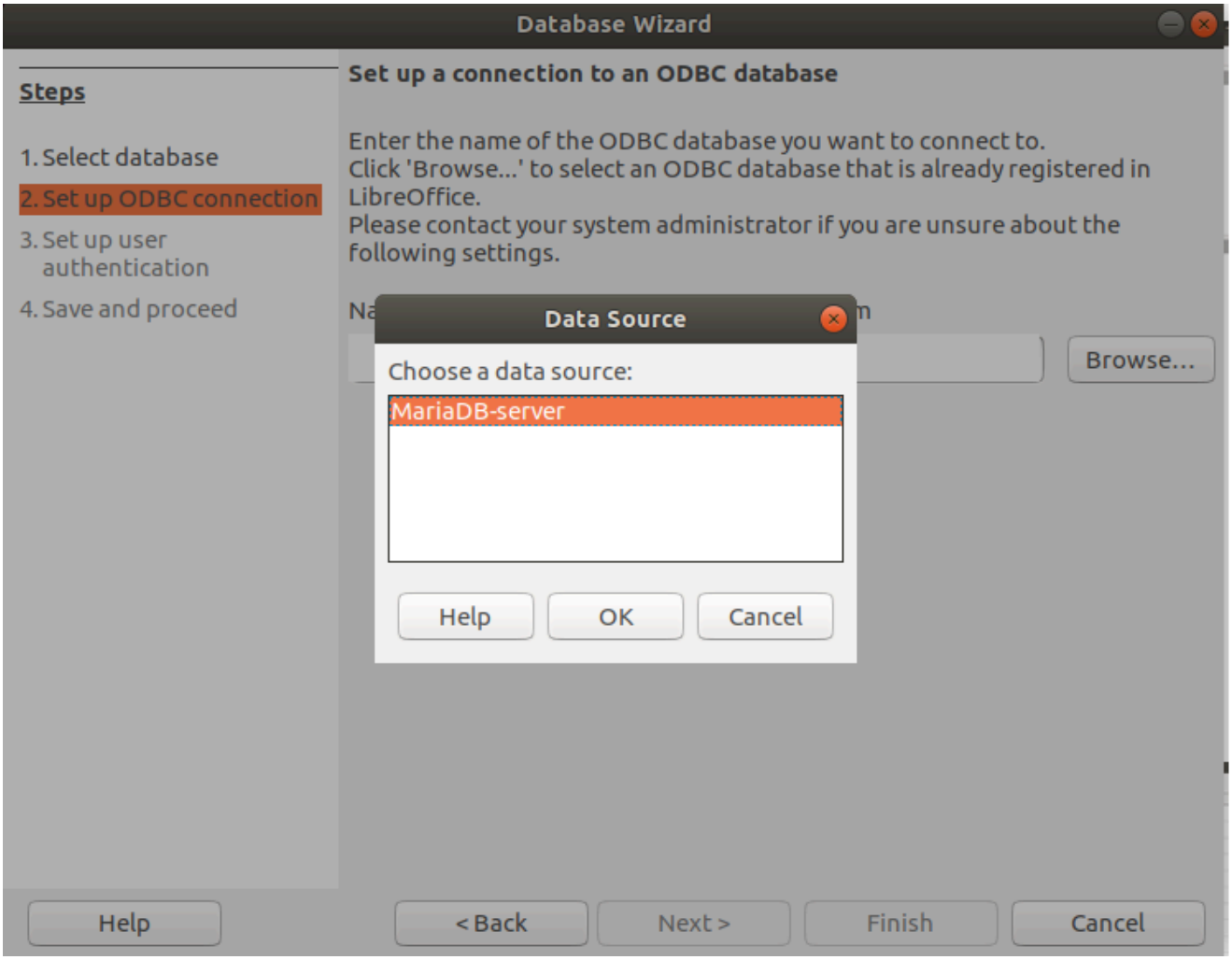
```
$ isql MariaDB-server
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL> show tables;
+---------------------------------------------------------------------------+
| Tables_in_test                                                            |
+---------------------------------------------------------------------------+
| Authors                                                                   |
| tbl_names                                                                 |
| webposts                                                                  |
| webusers                                                                  |
+---------------------------------------------------------------------------+
SQLRowCount returns 4
4 rows fetched
```

Start with LibreOffice Base

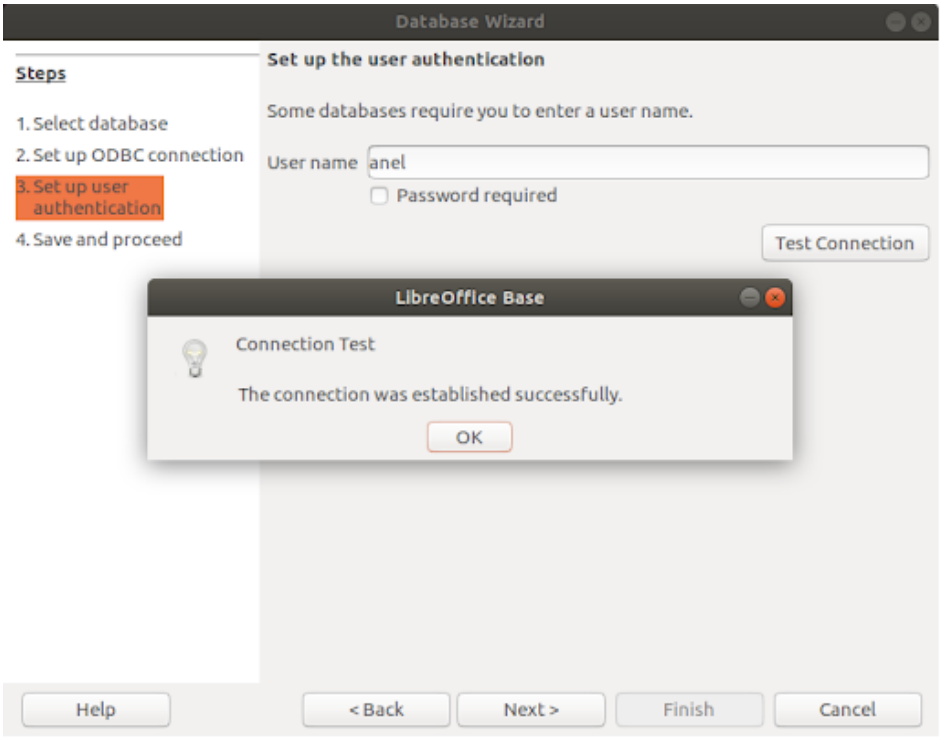
Start Libreoffice Base from the terminal by running *lobase* (make sure to install the *libreoffice-base* package if needed). The default option is to create a new database, which is *HSQLDB*. In order to connect to a running MariaDB server, choose *“Connect to an existing database”* and choose *“ODBC”* driver as shown below:



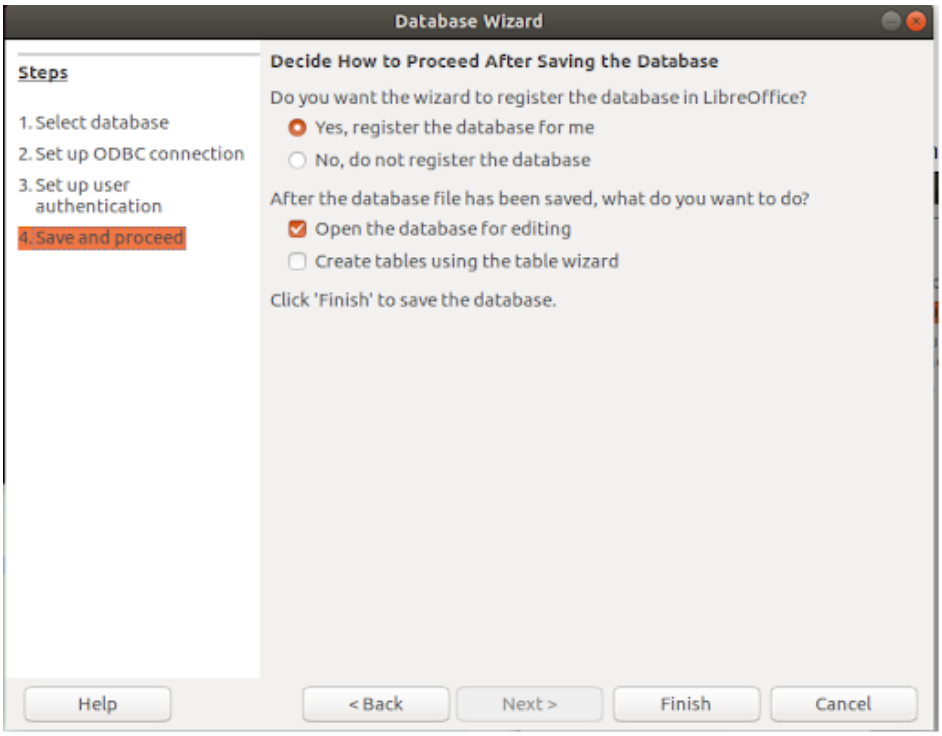
After that, choose DSN (the one that we created in the previous step) and click *“Next”*:



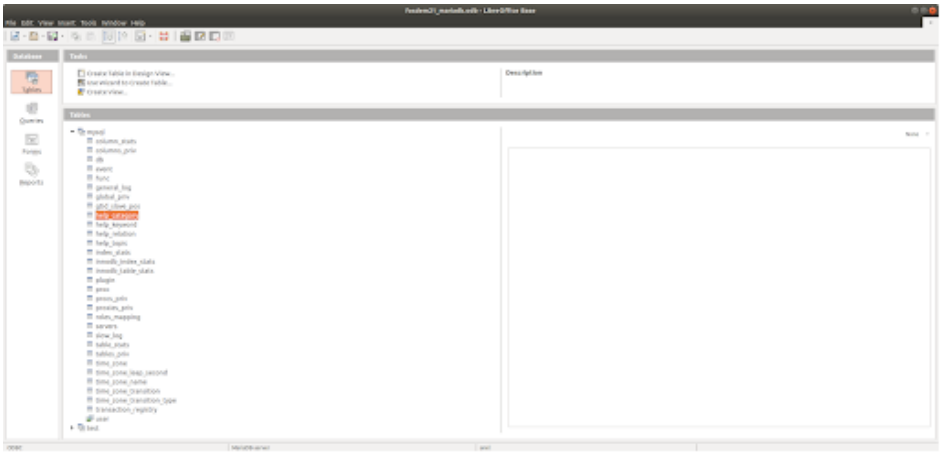
Provide a user name (and password if needed) and again check the connection (with the *"Test Connection"* button) and click *"Next"*:



After that, we have options to register the database. Registration in this sense means that the database is viewable by other LibreOffice modules (like *LibreOffice Calc* and *LibreOffice Writer*). So this step is optional. In this example, we will save as *"fosdem21_mariadb.odb"*. See Using a Registered Database.



It opens the following window:



It consists of three panels:

1. "Database" window with the options,
2. "Tables",
3. "Queries",
4. "Forms",
5. "Reports".
6. "Tasks window (dependent on what is selected in the "Database" window). When "Tables" is selected, the options are:
7. "Create Table in Design View",
8. "Use Wizard to Create Table" and
9. "Create View".

10. “Tables” window - shows list of tables that are created.

As we can see, there are system tables in the *“mysql”* database as well as *“test”* database.

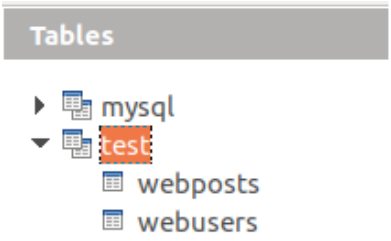
Let’s say we create a table using the REST API from JSON data from posts, and another table using the same mechanism from users, and let’s call them *webposts* and *webusers*. In order to do so, we have to enable the **CONNECT** storage engine plugin and start with REST_API. See more in the CONNECT - Files Retrieved Using Rest Queries article.

The queries we need to run in MariaDB are:

```
CREATE TABLE webusers ENGINE=CONNECT TABLE_TYPE=JSON
  HTTP='http://jsonplaceholder.typicode.com/users';

CREATE TABLE webposts ENGINE=CONNECT TABLE_TYPE=JSON
  HTTP='http://jsonplaceholder.typicode.com/posts';
```

The result in LibreOffice Base is as shown below:



librebase_6

Double clicking on the table opens a new window with the data displayed to inspect:

id	name	username	email	address	phone	website	company
1	Leanne Graham	Bret	Sincere@april.biz	Kulas Light Apt. 556 Gwenborough	1-770-736-8031 x56442	hildegard.org	Romaguera-Crona Multi-layered clien
2	Ervin Howell	Antonette	Shanna@melissa.tv	Victor Plains Suite 879 Wisokyburgl	010-692-6593 x09125	anastasia.net	Deckow-Crist Proactive didactic conti
3	Clementine Bauch	Samantha	Nathan@yesenia.net	Douglas Extension Suite 847 McKer	1-463-123-4447	ramiro.info	Romaguera-Jacobson Face to face bif
4	Patricia Lebsack	Karianne	Julianne.OConner@kory.org	Hoeger Mall Apt. 692 South Elvis 53	493-170-9623 x156	kale.biz	Robel-Corkery Multi-tiered zero tolera
5	Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca	Skiles Walks Suite 351 Roscoeview :	(254)954-1289	demarco.info	Keebler LLC User-centric fault-toleran
6	Mrs. Dennis Schulist	Leopoldo_Corkery	Karley_Dach@jasper.info	Norberto Crossing Apt. 950 South C	1-477-935-8478 x6430	ola.org	Considine-Lockman Synchronised bot
7	Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger@billy.biz	Rex Trail Suite 280 Howemouth 588	210.067.6132	elvis.io	Johns Group Configurable multimedi
8	Nicholas Runolfsdottir V	Maxime_Nienow	Sherwood@rosamond.me	Ellsworth Summit Suite 729 Aliyavie	586.493.6943 x140	jacynthe.com	Abernathy Group Implemented secon
9	Glenna Reichert	Delphine	Chaim_McDermott@dana.io	Dayna Park Suite 449 Bartholomebu	(775)976-6794 x41206	conrad.com	Yost and Sons Switchable contextuall
10	Clementina DuBuque	Moriah.Stanton	Rey.Padberg@karina.biz	Kattie Turnpike Suite 198 Lebsackbu	024-648-3804	ambrose.net	Hoeger LLC Centralized empowering

To create the table from the *“Tasks”* window, use the option *“Create Table in Design View”*, where one can specify specific field names and types as shown:

From the “Tasks” window one can create a table using the option *“Use Wizard to Create Table”* to create some sample tables.

One can fill the data in the existing table, or create and define the new table from the *LibreOffice Calc* module with simple copy-paste (in the *“Tasks”* window).

Using a Registered Database

Other modules can use the registered database, for example, open *“LibreOffice Calc”* and go to *“Tools”*, *“Options”* and you will see the *“odb”* file we registered when starting *“LibreOffice Base”*.

This page is licensed: CC BY-SA / Gnu FDL

Luna Modeler

> **Note:**
>
> Luna Modeler is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn’t test, validate, or support it. Refer to its own documentation and license terms.

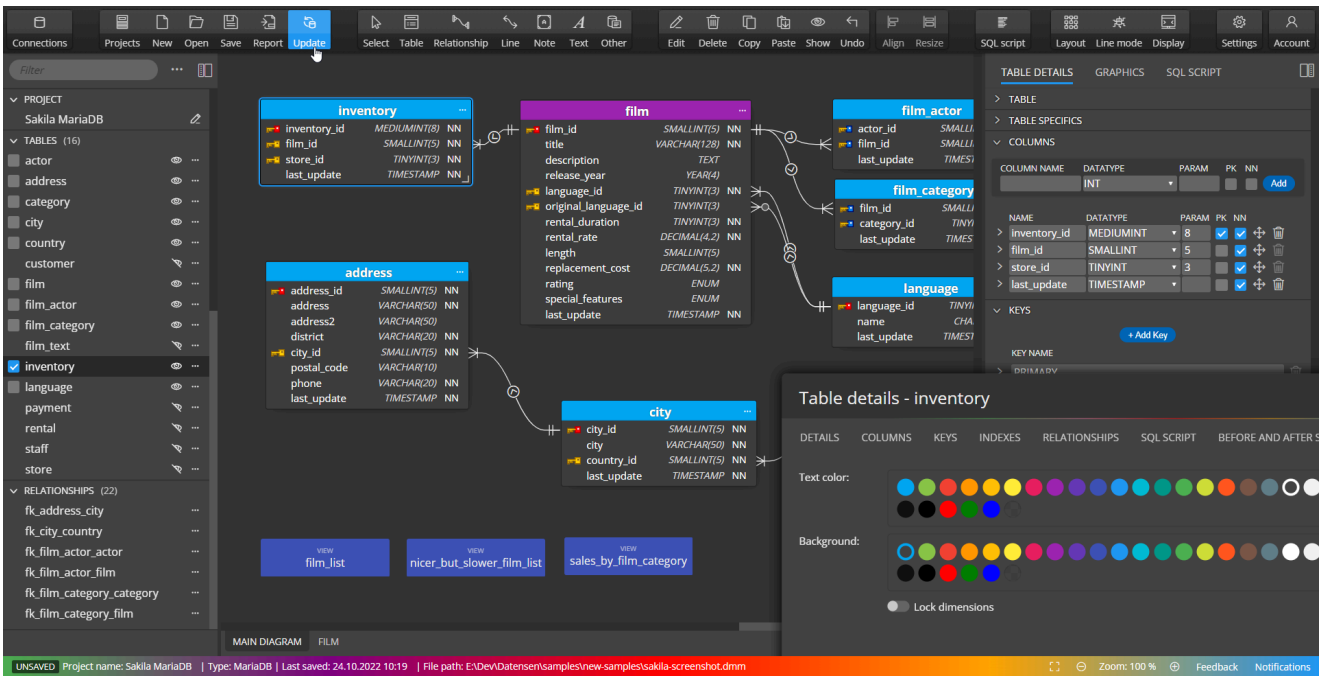
Luna Modeler is a database design tool for MariaDB and other relational databases.

Draw diagrams, reverse engineer existing database structures and generate SQL code.

Supported platforms include MariaDB, PostgreSQL, and SQLite.

Key features:

- Database modeling & schema design.
- Reverse engineering.
- Support for database-specific settings.
- Three display modes: metadata, sample data or descriptions.
- Default values for newly created objects.
- Export to PDF.
- Dark and Light themes.
- SQL script generation.



Supported OS

Luna Modeler is available for Windows, Linux and macOS.

More information

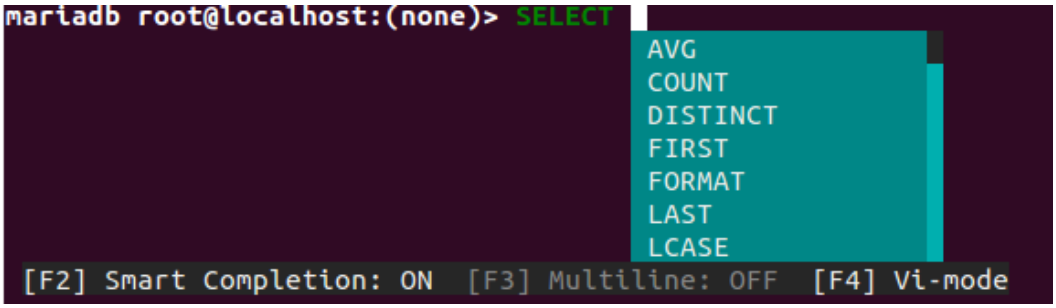
See www.datensen.com for more information.

This page is licensed: CC BY-SA / Gnu FDL

mycli

> **Note:**
>
> mycli is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

mycli is a command-line interface for MariaDB, MySQL, and Percona, with auto-completion and syntax highlighting. It is written in Python, and runs on Linux, Mac and Windows.



Read more at mycli.net.

This page is licensed: CC BY-SA / Gnu FDL

Navicat

> **Note:**
>
> Navicat is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Navicat is a graphical frontend for MariaDB. It is a commercial product with several different versions (Navicat Premium, Navicat for MySQL, etc…) and different "editions" within those versions (Non-commercial, Standard, and Enterprise). Certain features are only available in certain editions/versions.

Navicat is available for Windows, MacOS, and Linux.

In addition to standard client features, it includes

- An SQL Builder/Editor,
- A Data Modeling Tool (Enterprise version only),
- Data/Structure Synchronization,
- Report Builder (Windows-only),
- "Premium" version which can connect to MariaDB, MySQL, SQL Server, SQLite, Oracle, and PostgreSQL databases simultaneously.

A feature comparison matrix is available. More information, including pricing and trial versions, is available on the Navicat website.

This page is licensed: CC BY-SA / Gnu FDL

ocelotgui

> **Note:**
>
> ocelotgui is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

The Ocelot GUI (ocelotgui), a database client, allows to connect to a MySQL or MariaDB server, enter SQL statements, and receive results. Features include syntax highlighting, user-settable colors and fonts for each part of the screen, resultset displays with multiline rows and resizable columns, and a debugger.

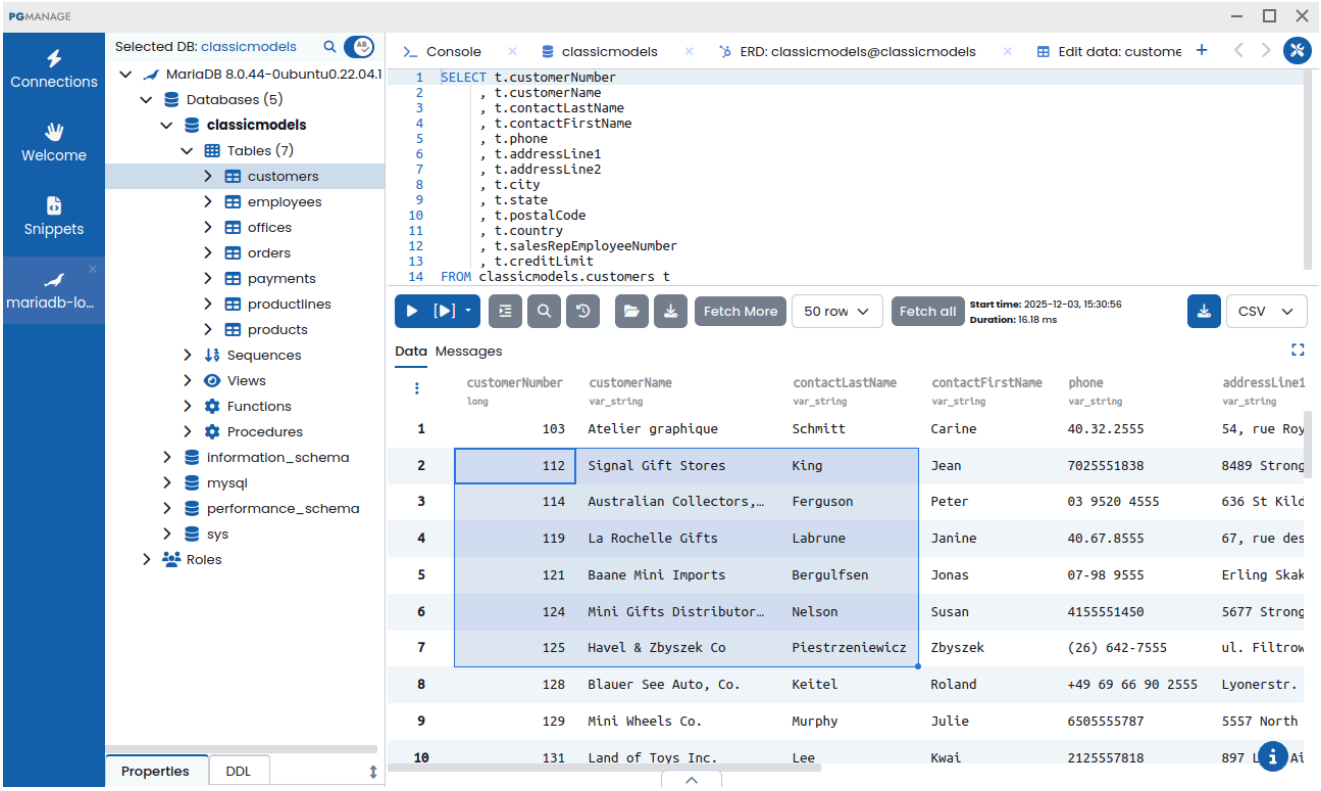
Visit ocelot.ca for more information and to download.

This page is licensed: CC BY-SA / Gnu FDL

PgManage

> **Note:**
>
> PgManage is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

PgManage is a modern SQL editor database management toolkit.



App Screenshot

PgManage supports MariaDB along many other databases.

Features:

Desktop or Browser-based: Runs as desktop application or can be installed as a webapp and used in a browser.

Modern Interface: The UI is optimized to be easy to use efficient to navigate.

Multiple Workspaces: Work with many database sessions at once and easily switch between them

Secure: Sensitive information like DB passwords, SSH keys etc. is stored in an encrypted form and protected by Master Password.

Schema and Table Editors: Create/alter tables and manage data using intuitive and clean UI.

Smart SQL Editor: Context-aware code completion, error annotations, powerful search and replace.

Light or Dark: Different themes to choose from.

Tabbed Workspace: Keep multiple editors and other tools open and easily switch between them.

Quick Search: Easily find an object in DB Explorer using a fuzzy search dialog.

Tunneling: Connect to your databases via SSH tunnel if it is not directly accessible.

Links:

Product Page www.commandprompt.com/products/pgmanage
Github Project https://github.com/commandprompt/pgmanage

This page is licensed: CC BY-SA / Gnu FDL

phpMyAdmin

> **Note:**
>
> phpMyAdmin is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

phpMyAdmin is a web-based tool for administering MariaDB and MySQL.

It requires a web server, PHP, and a browser.

Read more at phpmyadmin.net.

This page is licensed: CC BY-SA / Gnu FDL

Querious

> **Note:**
>
> Querious is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Querious is a database administration tool for macOS.

It can be purchased and downloaded at querious.

This page is licensed: CC BY-SA / Gnu FDL

SB Data Generator

> **Note:**
>
> SB Data Generator is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Generate and populate databases with large volumes of realistic test data.

SB Data Generator is a simple and powerful GUI tool for creating large volumes of realistic test data to populate selected tables or entire databases. The tool reverses your database and displays tables and columns, so you can

assign to them multiple data generator templates. It includes multiple built-in generators that allow populating MariaDB database tables with realistic data of various types.

Key Benefits:

- Understand your database structure by visualizing the ER Diagram (tables, columns, relationships, keys, sequences, indexes, and triggers).
- Ability to generate a large volume of realistic test data.
- Preview what data is generated before populating the database with test data.
- Multiple built-in generators available with the ability to create custom data generators.
- Generate test data for most popular DBMS including MariaDB and MySQL.

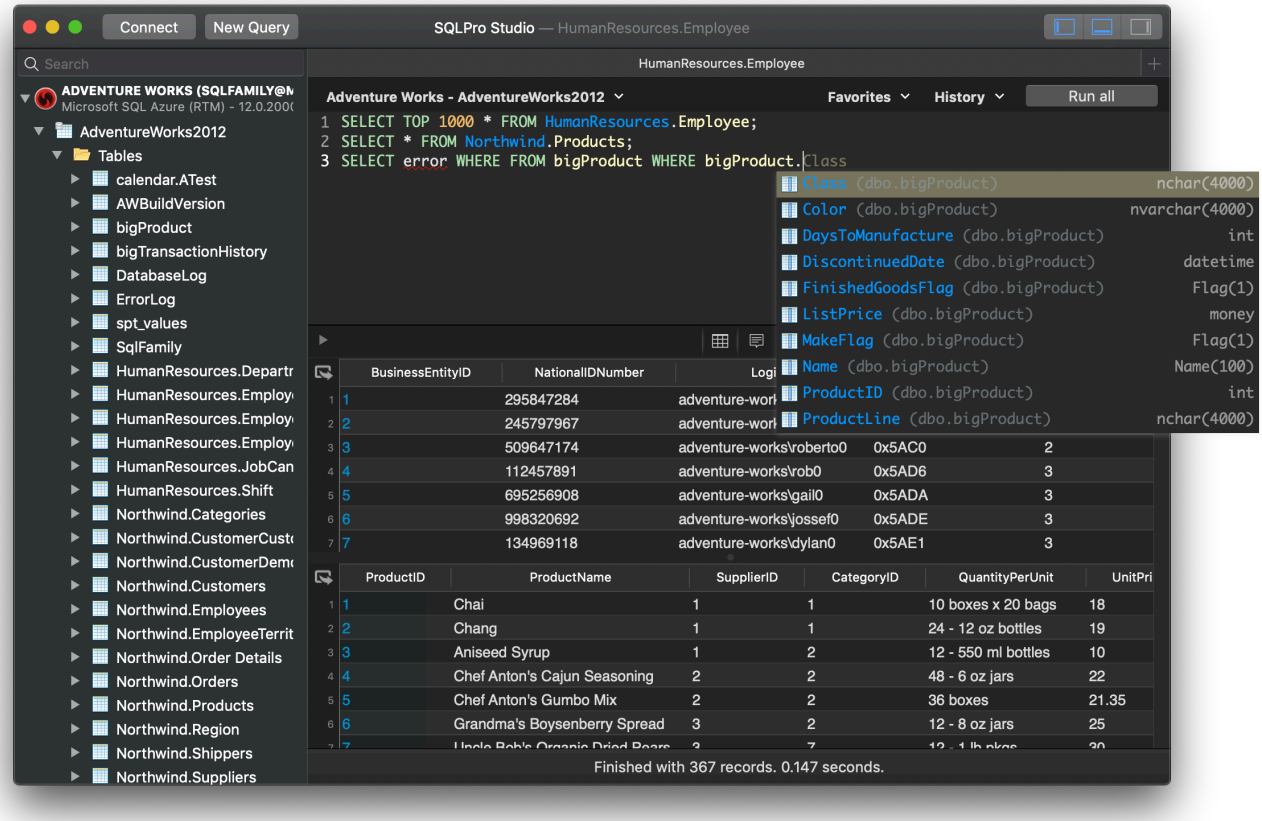
Learn more about SB Data Generator.

This page is licensed: CC BY-SA / Gnu FDL

SQLPro Studio

> **Note:**
>
> SQLPro Studio is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

SQLPro Studio is a fully native database client for macOS macOS and iOS. It supports database management systems such as MariaDB, MySQL, and Postgres.



SQLPro UI

Features include:

- Syntax highlighting & Autocomplete (sometimes called intellisense).
- Customizable themes allowing the query editor to be completely customized.
- The ability to run multiple queries at one time (and a customizable run query shortcut).
- Improved query error detection (errors in queries is underlined red).
- Full dark mode & light mode support, along with custom themes.

Other features include:

- Complete SSH Tunnel support.
- Inline data filtering.
- Exporting & Exporting options to CSV, JSON & XML.

This page is licensed: CC BY-SA / Gnu FDL

SQLyog

> **Note:**
>
> SQLyog is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

SQLyog is a GUI tool to manage MySQL and MariaDB servers and databases in physical, virtual, and cloud environments. DBAs, developers, and database architects alike, use SQLyog to visually compare, optimize, and document schemas.

Key Features:

- Automatically synchronize data,
- Visually compare data, and
- Import external data.

Additional Highlights:

- Runs on Microsoft Windows with no dependencies on runtimes (such as Microsoft .NET and Java) and database abstraction layers (such as ODBC and JDBC).
- Distributed as a free Community edition and as a paid, proprietary Ultimate edition.

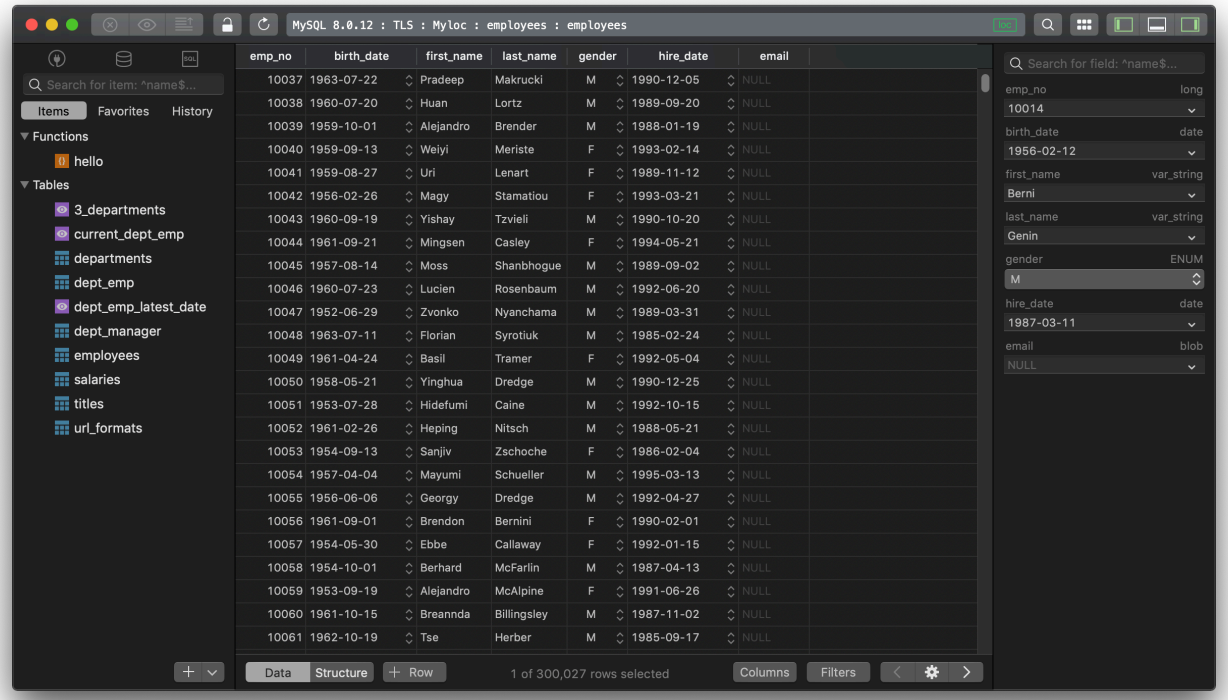
More information can be found here.

This page is licensed: CC BY-SA / Gnu FDL

TablePlus

> **Note:**
>
> TablePlus is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

TablePlus is an application with a clean user interface that allows developers to simultaneously manage databases in a very fast and secure way. It supports many popular database management systems like MariaDB, MySQL, and Postgres.



TablePlus is available for macOS, Windows, iOS, and Linux.

Some notable features:

- Native build.
- Convenient query editor.
- Multi Tabs & Code Review.
- Can connect to multiples databases simultaneously.

TablePlus is available for free, but users can purchase a license to remove some limitations and customize the tool for higher needs on TablePlus website.

This page is licensed: CC BY-SA / Gnu FDL

TOAD Edge

> **Note:**
>
> Toad Edge is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Toad Edge is a database management application that allows you to perform database administration tasks with ease.

Toad Edge allows you to:

- Connect to your MySQL database, view, explore and edit database structure, database objects and properties.
- Manage database objects, easily add, edit or drop objects in Object Explorer.
- Manage data stored in your database, add, edit or remove records.
- Write complex SQL code comfortably in Worksheet.
- Compare and synchronize databases using powerful Schema Compare.
- Obtain detailed information about your databases.

More information can be found here.

This page is licensed: CC BY-SA / Gnu FDL

Valentina Studio

> **Note:**
>
> Valentina Studio is third-party software, not developed or maintained by MariaDB and not included with MariaDB Server. MariaDB doesn't test, validate, or support it. Refer to its own documentation and license terms.

Valentina Studio is a graphical front end for MariaDB with a free version, and a Pro version that adds advanced features.

Free Valentina Studio offers the following features:

- natively available on macOS, Windows & Linux,
- a feature rich, visual SQL Editor with powerful search functionality,
- SQLDIFF,
- a highly productive Data Editor,
- support for many data sources,
- ODBC import,
- Schema Editor with powerful visual diagramming features, Diagram Editor; import diagrams from MySQL Workbench,
- Report Viewer with printing, export to PDF and HTML (local reports and from Valentina Server),
- Forms Client for working with visual forms and MariaDB (locally and from Valentina Server),
- Free supports most popular data sources, MariaDB, MySQL, SQL Server, SQLite, PostgreSQL and more.

Use the Valentina Studio FREE vs Valentina Studio PRO feature matrix to see which version is best for your needs or download to immediately evaluate.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Direct Query Adapter For Microsoft Power BI

MariaDB Direct Query Adapter for Power BI enables Microsoft Power BI Desktop users to remotely connect to and query their MariaDB database including on MariaDB Cloud without downloading the entire data set to their local machine.

MariaDB Direct Query Adapter is a Microsoft certified connector for Microsoft Power BI Desktop.

About

Microsoft Power BI Desktop is a free business intelligence tool available as a downloadable desktop client. Built for analysts, Microsoft Power BI provides state-of-the-art interactive visualizations, with industry-leading data query and modeling built-in. It is one of the top business intelligence tools in use globally.

MariaDB database products support analytics and data warehousing in addition to transactional databases. Customers can meet the challenges from complex workloads using a single stack. MariaDB Enterprise Server includes pluggable smart storage engines to meet specific workload needs and connectors for high-performance data access by applications.

MariaDB Enterprise ColumnStore is a columnar storage engine designed to handle the extremely large data sets and ad hoc queries of analytical workloads. Data is written to disk by column rather than row and is automatically partitioned. Columnar data storage is designed to deliver real time analysis on billions of rows. MariaDB Enterprise ColumnStore is a plugin storage engine and is included with MariaDB Enterprise Server.

MariaDB Direct Query Adapter enables Power BI users to work with remote data without the need to download it onto the local machine. Downloading analytical data sets can present challenges such as download speed and inadequate memory on a local machine, and even a powerful laptop or desktop is outperformed by the processing power of a high end server or MPP system. MariaDB's Direct Query Adapter enables users to avoid these limitations and provides robust querying and data visualization from any location.

Requirements

MariaDB Direct Query Adapter for Microsoft Power BI requires:

- Microsoft Power BI Desktop
- MariaDB ODBC Connector v3.1.10 or later.

Download and Install

To use MariaDB Direct Query Adapter with Microsoft Power BI Desktop:

1. Check the system requirements for Microsoft Power BI Desktop.
2. Download Microsoft Power BI Desktop using the Microsoft instructions.
3. Download MariaDB ODBC Connector.
4. Select MariaDB ODBC Connector v3.1.10 or later for Windows.
5. Click on the download or choose "Open" to start the MariaDB ODBC Connector Setup Wizard.
6. Click "Next".
7. Read and accept the terms and agreement.
8. Click "Next".
9. Choose "Typical" for installation type and click "Install".
10. When asked if you want to allow this app to makes changes to your device, click "Yes".
11. If you have older versions of MariaDB ODBC Connector 64-bit, select "Make User DSN's for older Connector version to use this version" and click "Finish". Otherwise, just click "Finish".

Connect

Connect to your remote MariaDB database from Power BI Desktop using the MariaDB Direct Query Adapter.

1. Open Microsoft Power BI Desktop.
2. From the Power BI home screen, choose Get Data->More. Enter "MariaDB". Select MariaDB from the menu and click "Connect".
3. Enter the data source. This is an IP address and port number, or in the case of MariaDB Cloud, it is the qualified domain name and port number.
4. Select "DirectQuery" and click "OK".
5. Enter your user name and password and click "Connect".

You are now connected to your remote MariaDB database and can use Microsoft Power BI to run queries without the need to download the data locally.

Queries

The MariaDB Direct Query Adapter for Power BI interacts with the SQL layer. It is upstream of the storage engine, so queries can be done regardless of the underlying storage engine.

For example, MariaDB Direct Query Adapter for Microsoft Power BI can be used to query data on a row-based transactional database using the InnoDB storage engines or a column-based data warehouse using the MariaDB Enterprise ColumnStore storage engine to deliver real time analysis of billions of rows of data.

Relationships

Power BI depends on relationships for data analysis. These relationships are automatically established for row-based databases, for example those using the InnoDB storage engine.

Because column-based storage engines such as MariaDB Enterprise ColumnStore do not use indexes, relationships must be set up manually. For more about creating relationships in Power BI see Create and Manage Relationships in Power BI.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Logging Tools

See mariadb-binlog.

Use mariadb-binlog to process binary log files. This utility is essential for examining replication events, recovering from data loss, and auditing changes in your MariaDB Server.

See mariadb-dumpslow.

Summarize and analyze the MariaDB slow query log with mariadb-dumpslow (formerly mysqldumpslow), grouping similar queries to surface the slowest statements.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-binlog

See Using mariadb-binlog.

How to use the mariadb-binlog client (formerly mysqlbinlog) to read and process binary log files.

See mariadb-binlog Options.

Command-line options for the mariadb-binlog client (formerly mysqlbinlog).

See Annotate_rows_log_event.

The ANNOTATE_ROWS_EVENT, which records the original SQL statement alongside row-based binary log events.

See mysqlbinlog.

mysqlbinlog is the former name of the mariadb-binlog client, still available as a symlink or alternate binary.

This page is licensed: CC BY-SA / Gnu FDL

Using mariadb-binlog

> **Note:**
>
> Previously, the client was called `mysqlbinlog`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Overview

The MariaDB server's binary log is a set of files containing "events" which represent modifications to the contents of a MariaDB database.

Events are written in a binary (that is, non-human-readable) format. The `mariadb-binlog` utility is used to view these events in plain text.

Run mariadb-binlog from a command line:

```
mariadb-binlog [options] binlog-filename [binlog-filename ...]
```

See mariadb-binlog Options for details on the available options.

Usage

Display the contents of a binary log file named `mariadb-bin.000152` like this:

```
mariadb-binlog mariadb-bin.000152
```

Processing a Single Log File

The logging format is determined by the value of the binlog_format system variable. If you are using statement-based logging, the output includes the SQL statement, the ID of the server the statement was executed on, a timestamp, and how much time the statement took to execute. If you are using row-based logging, the output of an event will not include an SQL statement, but will instead output how individual rows were changed.

The output from `mariadb-binlog` can be used as input to the `mariadb` client to redo the statements contained in a binary log. This is useful for recovering after a server crash (replace *binlog-filename* with the name of a binary log file):

```
mariadb-binlog binlog-filename | mysql -u root -p
```

If you would like to view and possibly edit the file before applying it to your database, use the `-r` flag to redirect the output to a file (replace *outputfile* with the name of a file to store the output, and *binlog-filename* with the name of a binary log file):

```
mariadb-binlog -r outputfile binlog-filename
```

In the output file, delete any statements you don't want executed (such as an accidental `DROP DATABASE`). Once you are satisfied with the contents, you can execute it:

```
mariadb -u root -p --binary-mode < outputfilename
```

Be careful to process multiple log files in a single connection, especially if one or more of them have any `CREATE TEMPORARY TABLE ...` statements. Temporary tables are dropped when the mariadb client terminates, so if you are processing multiple log files one at a time (i.e. multiple connections), and one log file creates a temporary table and then a subsequent log file refers to the table, you get an 'unknown table' error.

Processing Multiple Log Files

To execute multiple log files using a single connection, list them all on the `mariadb-binlog` command line:

```
mariadb-binlog mariadb-bin.000001 mariadb-bin.000002 | mariadb -u root -p --binary-mode
```

If you need to manually edit the binlogs before executing them, combine them all into a single file before processing:

```
mariadb-binlog mariadb-bin.000001 > /tmp/mariadb-bin.sql
mariadb-binlog mariadb-bin.000002 >> /tmp/mariadb-bin.sql
# make any edits
mariadb -u root -p -e "source /tmp/mariadb-bin.sql"
```

See Also

- mariadb-binlog
- mariadb-binlog Options

This page is licensed: GPLv2, originally from fill_help_tables.sql

mariadb-binlog Options

> **Note:**
>
> Previously, the client was called `mysqlbinlog`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Options

mariadb-binlog is a utility included with MariaDB for processing binary log and relay log files.

The following options are supported by mariadb-binlog. They can be specified on the command line or in option files.

-?, –help

Display a help statement.

> **CURRENT**
>
> **–base64-output=_name_**
>
> Determine when the output statements should be base64-encoded `BINLOG` statements. Options (case-insensitive) include `auto`, `unspec`, `never` ,and `decode-rows`. `never` neither prints base64 encodings nor verbose event data, and exits on error if a row-based event is found. This option is useful for binlogs that are entirely statement-based. `decode-rows` decodes row events into commented SQL statements if the `--verbose` option is also given. It can enhance the debugging experience with large binary log files, as the raw data is omitted. Unlike `never`, mariadb-binlog does not exit with an error if a row event is found. `auto` (synonymous with `unspec`) outputs base64 encoded entries for row-based and format description events; it should be used when `ROW`-format events are processed for re-executing on the MariaDB server. This behavior is presumed, such that `auto` is the default value when no option specification is provided. The other option values are intended only for debugging or testing purposes because they may produce output that does not include all events in executable form.
>
> **< 10.6 / 10.5.10**
>
> **–base64-output[=_name_]**
>
> Determine when the output statements should be base64-encoded BINLOG statements. Options (case-insensitive) include `auto`, `unspec`, `always` (deprecated), `never` , and `decode-rows`. `never` disables it and works only for binlogs without row-based events; `decode-rows` decodes row events into commented SQL statements if the `--verbose` option is also given. Unlike `never`, `mariadb-binlog` does not exit with an error if a row event is found `auto` or `unspec`, the default, prints base64 only when necessary (for instance, for row-based events and format description events), and is the only safe behavior if you intend to use the output of `mariadb-binlog` to re-execute binary log file contents. The other option values are intended only for debugging or testing purposes, because they may produce output that does not include all events in executable form. `always` prints base64 whenever possible, and is for debugging only and should not be used in a production system. If this option is not given, the default is `auto`; if it is given with no argument, `always` is used.

Default value: `auto`.

–binlog-row-event-max-size=*val*

The maximum size in bytes of a row-based binary log event. Should be a multiple of 256. Minimum 256, maximum `18446744073709547520`. Default value: `4294967040` (4GB)

–character-sets-dir=*name*

Directory where the character sets are.

-d, –database=*name*

Output entries from the binary log (local log only) that occur while the name has been selected as the default database by USE. Only one database can be specified. The effects depend on whether the statement-based or row-based logging format is in use. For statement-based logging, the server only logs statements where the default database is *name*. The default database is set with the USE statement. For row-based logging, the server logs any updates to any tables in the named database, irrespective of the current database. Ignored in `--raw` mode.

[options], –debug[=options]

In a debug build, write a debugging log. A typical debug options string is `d:t:o,file\_name`. Default value: `d:t:o,/tmp/mariadb-binlog.trace`

–debug-check

Print some debug info at exit. Default value: `FALSE`

–debug-info

Print some debug info and memory and CPU info at exit. Default value: `FALSE`

–default-auth=*name*

Default authentication client-side plugin to use.

–defaults-extra-file=*name*

Read the file name, which can be the full path or the path relative to the current directory, after the global files are read.

–defaults-file=*name*

Only read default options from the given file name, which can be the full path or the path relative to the current directory.

–defaults-group-suffix=*str*

Also read groups with a suffix of *str*. For example, since `mariadb-binlog` normally reads the `[client]` and `[mysqlbinlog]` groups, `--defaults-group-suffix=x` would cause it to also read the groups `\[mysqlbinlog\_x]` and `\[client\_x]`.

-D, –disable-log-bin

Disable binary log. This is useful if you enabled `--to-last-log` and are sending the output to the same MariaDB server. This way, you could avoid an endless loop. You would also like to use it when restoring after a crash to avoid duplication of the statements you already have. The `SUPER` privilege is needed to use this option. Default value: `FALSE`

–do-domain-ids=*name*

A list of positive integers, separated by commas, that form a whitelist of domain ids. Any log event with a GTID that originates from a domain id specified in this list is displayed. Cannot be used with `--ignore-domain-ids`. When used with `--ignore-server-ids` or `--do-server-ids`, the result is the intersection between the two datasets. Available from MariaDB 10.9.

–do-server-ids=*name*

A list of positive integers, separated by commas, that form a whitelist of server ids. Any log event originating from a server id specified in this list is displayed. Cannot be used with `--ignore-server-ids`. When used with `--ignore-domain-ids` or `do-domain-ids`, the result is the intersection between the two datasets. Alias for `--server-id`. Available from MariaDB 10.9.

-B, –flashback

Support flashback mode. Default value: `FALSE`

-F, –force-if-open

Force if binlog was not closed properly. Defaults to `ON`; use `--skip-force-if-open` to disable. Default value: `TRUE`

-f, –force-read

If `mariadb-binlog` reads a binary log event that it does not recognize, it prints a warning, ignores the event, and continues. Without this option, `mariadb-binlog` stops if it reads such an event. Default value: `FALSE`

–gtid-strict-mode

Process binlog according to `gtid-strict-mode` specification. The start, stop positions are verified to satisfy the start < stop comparison condition. Sequence numbers of any GTID domain must comprise a monotonically growing sequence. Defaults to `ON`; use `--skip-gtid-strict-mode` to disable. Available from MariaDB 10.8. Default value: `TRUE`

-H, –hexdump

Augment output with hexadecimal and ASCII event dump. Default value: `FALSE`

-h, –host=*name*

Get the binlog from the MariaDB server on the given host.

–ignore-domain-ids=*name*

A list of positive integers, separated by commas, that form a blacklist of domain ids. Any log event with a GTID that originates from a domain id specified in this list is hidden. Cannot be used with `--do-domain-ids`. When used with `--ignore-server-ids` or `--do-server-ids`, the result is the intersection between the two datasets. Available from MariaDB 10.9.

–ignore-server-ids=*name*

A list of positive integers, separated by commas, that form a blacklist of server ids. Any log event originating from a server id specified in this list is hidden. Cannot be used with `--do-server-ids`. When used with `--ignore-domain-ids` or `--do-domain-ids`, the result is the intersection between the two datasets. Available from MariaDB 10.9.

-l path, –local-load=*path*

Prepare local temporary files for LOAD DATA INFILE in the specified directory. The temporary files are not automatically removed.

–no-defaults

Don't read default options from any option file.

-o value, –offset=*value*

Skip the first value entries in the log. Default value: `0`

–open_files_limit=#

Reserve file descriptors for usage by mariadb-binlog. Default value: `64`

-p[*passwd*], –password[=*passwd*]

Password to connect to the remote server. The password can be omitted, allow it to be entered from the prompt, or an option file can be used to avoid the security risk of passing a password on the command line.

–plugin-dir=*dir_name*

Directory for client-side plugins.

-P *num*, –port=*num*

Port number to use for connection or `0` for default to, in order of preference, `my.cnf`, `$MYSQL_TCP_PORT`, `/etc/services`, built-in default (`3306`). Default value: `0`

–position=#

Removed. Use `--start-position` instead.

–print-defaults

Print the program argument list from all option files and exit.

–print-row-count

Print row counts for each row of events. (Defaults to `ON`; use `--skip-print-row-count` to disable.) Default value: `TRUE`

–print-row-event-positions

Print row event positions. Defaults to on; use `--skip-print-row-event-positions` to disable.) Default value: `TRUE`

print-table-metadata

Print metadata stored in `Table_map_log_event`.

–protocol=*name*

The protocol of the connection (`tcp`, `socket`, `pipe`, `memory`).

–raw

Requires `-R`. Output raw binlog data instead of SQL statements. Output files are named after server logs.

-R, –read-from-remote-server

Read binary logs from a remote MariaDB server rather than reading a local log file. Any connection parameter options are ignored unless this option is given as well. These options are `--host`, `--password`, `--port`, `--protocol`, `--socket`, and `--user`. This option requires that the remote server be running. It works only for binary log files on the remote server, not relay log files. Default value: `FALSE`

-r name, –result-file=*name*

Direct output to a given file. With `--raw`, this is a prefix for the file names.

–rewrite-db=*name*

Updates to a database with a different name than the original. Example: `rewrite-db='from->to'`. For events that are logged as statements, rewriting the database constitutes changing a statement's default database from `db1` to `db2`. There is no statement analysis or rewrite of any kind; that is, if you specify `db1.tbl` in the statement explicitly, that occurrence won't be changed to `db2.tbl`. Row-based events are rewritten correctly to use the new database name. Filtering (for instance, with `--database=name`) happens before the database rewrites have been performed. If you use this option on the command line and `>` has a special meaning to your command interpreter, quote the value (for instance, `--rewrite-db="oldname->newname"`).

–server-id=*idnum*

Extract only binlog entries created by the server having the given id. From MariaDB 10.9, an alias for `--do-server-ids`. Default value: `0`

–set-charset=*character_set*

Add `SET NAMES character_set` to the output to specify the character set to be used for processing log files.

–shared-memory-base-name=*name*

Shared-memory *name* to use for Windows connections using shared memory to a local server (started with the `--shared-memory` option). Case-sensitive. Default value: `MYSQL`

-s, –short-form

Just show regular queries: no extra info and no row-based events. This is for testing only, and should not be used in production systems. If you want to suppress base64-output, consider using `--base64-output=never` instead. Default value: `FALSE`

–skip-annotate-row-events

Skip all Annotate_rows events in the mariadb-binlog output (by default, `mariadb-binlog` prints `Annotate_rows` events if the binary log contains them).

-S, –socket=*name*

For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

–ssl

Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The `--ssl` option does not enable verifying the server certificate by default. In order to verify the server certificate, you must specify the `--ssl-verify-server-cert` option. Default value: `FALSE`

–ssl-ca=*name*

Defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option implies the `--ssl` option.

–ssl-capath=*name*

Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the `--ssl` option.

–ssl-cert=*name*

Defines a path to the X509 certificate file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

–ssl-cipher=*name*

List of permitted ciphers or cipher suites to use for TLS. This option implies the `--ssl` option.

–ssl-crl=*name*

Defines a path to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL or Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

–ssl-crlpath=*name*

Defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

–ssl-key=*name*

Defines a path to a private key file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

–ssl-verify-server-cert

Enables server certificate verification. This option is disabled by default. Default value: `FALSE`

–start-datetime=*datetime*

If specified, start reading the binlog at the first event having a datetime equal to or later than the argument; the argument must be a date and time in the local time zone, in any format accepted by the MariaDB server for DATETIME and TIMESTAMP types, for example: `2014-12-25 11:25:56` (you should probably use quotes for your shell to set it properly). This option is useful for point-in-time recovery.

-j pos, –start-position=*pos*

Start reading the binlog at this position. Type can either be a positive integer or, from MariaDB 10.8, a GTID list. When using a positive integer, the value only applies to the first binlog passed on the command line. In GTID mode, multiple GTIDs can be passed as a comma-separated list, where each must have a unique domain id. The list represents the GTID binlog state that the client (another "replica" server) is aware of. Therefore, each GTID is exclusive; only events after a given sequence number are printed to allow users to receive events after their current state. Default value: `4`

> **Warning:**
>
> Options `--start-position` and `--stop-position` currently compare Sequence Numbers only per Domain ID and ignore Server IDs. This is incorrect, as it is different from the Replication design for GTIDs, which compares per Domain–Server ID pair. MDEV-37231 tracks this bug.

–stop-datetime=*name*

If specified, stop reading the binlog at the first event having a datetime equal to or posterior to the argument; the argument must be a date and time in the local time zone, in any format accepted by the MariaDB server for `DATETIME` and `TIMESTAMP` types, for example: `2014-12-25 11:25:56` (you should probably use quotes for your shell to set it properly). Ignored in `--raw` mode.

> **≥ 10.11.14**
>
> Emit a warning if the specified time is not found within the specified binlogs.

–stop-never

Wait for more data from the server (and thus requires `-R` or `--read-from-remote-server`) instead of stopping at the end of the last log. Implies `--to-last-log`.

–stop-never-slave-server-id

The replica server_id used for `--read-from-remote-server` `--stop-never`.

–stop-position=*position*

If specified, stop reading the binlog at this *position*. Type can either be a positive integer or, from MariaDB 10.8, a GTID list. When using a positive integer, the value only applies to the last binlog passed on the command line. In GTID mode, multiple GTIDs can be passed as a comma-separated list, where each must have a unique domain id. Each GTID is inclusive; only events up to the given sequence numbers are printed. Ignored in `--raw` mode.

> **≥ 10.11.14**
>
> Emit a warning if the specified position or GTID is not within the specified binlogs.
>
> **≥ 10.5.27**
>
> Emit a warning if the specified position is beyond the end of the last binlog.

> **Warning:**
>
> Options `--start-position` and `--stop-position` currently compare Sequence Numbers only per Domain ID and ignore Server IDs. This is incorrect, as it is different from the Replication design for GTIDs, which compares per Domain–Server ID pair. MDEV-37231 tracks this bug.

-T, –table

List entries for just this table (affects only row events).

–tls-version=*name*

This option accepts a comma-separated list of TLS protocol versions. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted. See Secure Connections Overview: TLS Protocol Versions for more information. Default value: `TLSv1.1,TLSv1.2,TLSv1.3`

-t, –to-last-log

Requires `-R` or `--read-from-remote-server`. Does not stop at the end of the requested binlog but rather continues printing until the end of the last binlog of the MariaDB server. If you send the output to the same MariaDB server, that may lead to an endless loop. Default value: `FALSE`

-u, –user=*username*

Connect to the remote server as *username*.

-v, –verbose

Reconstruct SQL statements out of row events. `-v -v` adds comments on column data types.

-V, –version

Print version and exit.

–verify-binlog-checksum

Verify binlog event checksums when reading a binlog file.

Option Files

In addition to reading options from the command line, `mariadb-binlog` can also read options from option files. If an unknown option is provided to `mariadb-binlog` in an option file, it is ignored.

The following options relate to how MariaDB command line tools handle option files. They must be given as the first argument on the command line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

`mariadb-binlog` is linked with MariaDB Connector/C. However, MariaDB Connector/C does not handle the parsing of option files for this client. That is performed by the server's option file parsing code. See MDEV-19035 for more information.

Option Groups

`mariadb-binlog` reads options from the following option groups from the option files:

Group	Description
[mysqlbinlog]	Options read by mariadb-binlog, which includes both MariaDB Server and MySQL Server.
[mariadb-binlog]	Options read by mariadb-binlog.
[client]	Options read by all MariaDB and MySQL client programs, which include both MariaDB and MySQL clients. For example, mysqldump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which are common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

This page is licensed: CC BY-SA / Gnu FDL

Annotate_rows_log_event

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Overview

`Annotate_rows` events accompany `row` events and describe the query which caused the row event.

In the binary log, each `Annotate_rows` event precedes the corresponding Table map event or the first of the Table map events, if there are more than one (for instance, in a case of multi-delete or insert delayed).

Example

```
master> DROP DATABASE IF EXISTS test;
master> CREATE DATABASE test;
master> USE test;
master> CREATE TABLE t1(a int);
master> INSERT INTO t1 VALUES (1), (2), (3);
master> CREATE TABLE t2(a int);
master> INSERT INTO t2 VALUES (1), (2), (3);
master> CREATE TABLE t3(a int);
master> INSERT DELAYED INTO t3 VALUES (1), (2), (3);
master> DELETE t1, t2 FROM t1 INNER JOIN t2 INNER JOIN t3
    ->        WHERE t1.a=t2.a AND t2.a=t3.a;

master> SHOW BINLOG EVENTS IN 'master-bin.000001';
+------------------+------+--------------+-----------+-------------+---------------------------------
-----------------------------------------------+
| Log_name         | Pos  | Event_type   | Server_id | End_log_pos | Info
|
+------------------+------+--------------+-----------+-------------+---------------------------------
-----------------------------------------------+
| master-bin.000001 |    4 | Format_desc  |       100 |         240 | Server ver: 5.5.20-MariaDB-
mariadb1~oneiric-log, Binlog ver: 4                |
| master-bin.000001 |  240 | Query        |       100 |         331 | DROP DATABASE IF EXISTS test
|
| master-bin.000001 |  331 | Query        |       100 |         414 | CREATE DATABASE test
|
| master-bin.000001 |  414 | Query        |       100 |         499 | use `test`; CREATE TABLE t1(a int)
|
| master-bin.000001 |  499 | Query        |       100 |         567 | BEGIN
|
| master-bin.000001 |  567 | Annotate_rows |       100 |         621 | INSERT INTO t1 VALUES (1), (2),
(3)                                              |
| master-bin.000001 |  621 | Table_map    |       100 |         662 | table_id: 16 (test.t1)
|
| master-bin.000001 |  662 | Write_rows   |       100 |         706 | table_id: 16 flags: STMT_END_F
|
| master-bin.000001 |  706 | Query        |       100 |         775 | COMMIT
|
| master-bin.000001 |  775 | Query        |       100 |         860 | use `test`; CREATE TABLE t2(a int)
|
| master-bin.000001 |  860 | Query        |       100 |         928 | BEGIN
|
| master-bin.000001 |  928 | Annotate_rows |       100 |         982 | INSERT INTO t2 VALUES (1), (2),
(3)                                              |
| master-bin.000001 |  982 | Table_map    |       100 |        1023 | table_id: 17 (test.t2)
|
| master-bin.000001 | 1023 | Write_rows   |       100 |        1067 | table_id: 17 flags: STMT_END_F
|
| master-bin.000001 | 1067 | Query        |       100 |        1136 | COMMIT
|
| master-bin.000001 | 1136 | Query        |       100 |        1221 | use `test`; CREATE TABLE t3(a int)
|
| master-bin.000001 | 1221 | Query        |       100 |        1289 | BEGIN
|
| master-bin.000001 | 1289 | Annotate_rows |       100 |        1351 | INSERT DELAYED INTO t3 VALUES (1),
(2), (3)                                           |
| master-bin.000001 | 1351 | Table_map    |       100 |        1392 | table_id: 18 (test.t3)
|
| master-bin.000001 | 1392 | Write_rows   |       100 |        1426 | table_id: 18 flags: STMT_END_F
|
| master-bin.000001 | 1426 | Table_map    |       100 |        1467 | table_id: 18 (test.t3)
|
| master-bin.000001 | 1467 | Write_rows   |       100 |        1506 | table_id: 18 flags: STMT_END_F
|
| master-bin.000001 | 1506 | Query        |       100 |        1575 | COMMIT
|
| master-bin.000001 | 1575 | Query        |       100 |        1643 | BEGIN
|
```

```
| master-bin.000001 | 1643 | Annotate_rows |       100 |        1748 | DELETE t1, t2 FROM t1 INNER JOIN
t2 INNER JOIN t3 WHERE t1.a=t2.a AND t2.a=t3.a |
| master-bin.000001 | 1748 | Table_map     |       100 |        1789 | table_id: 16 (test.t1)
|
| master-bin.000001 | 1789 | Table_map     |       100 |        1830 | table_id: 17 (test.t2)
|
| master-bin.000001 | 1830 | Delete_rows   |       100 |        1874 | table_id: 16
|
| master-bin.000001 | 1874 | Delete_rows   |       100 |        1918 | table_id: 17 flags: STMT_END_F
|
| master-bin.000001 | 1918 | Query         |       100 |        1987 | COMMIT
|
+-------------------+------+---------------+-----------+-------------+---------------------------------
-----------------------------------------------+
```

Options

The following options control the behavior of `Annotate_rows_log_event`.

Master Option: `--binlog-annotate-row-events`

This option tells the master to write `Annotate_rows` events to the binary log. See binlog_annotate_row_events for a detailed description of the variable.

Session values allow to annotate only some selected statements:

```
...
SET SESSION binlog_annotate_row_events=ON;
... statements to be annotated ...
SET SESSION binlog_annotate_row_events=OFF;
... statements not to be annotated ...
```

Replica Option: `--replicate-annotate-row-events`

This option tells the replica to reproduce `Annotate_row` events received from the master in its own binary log (sensible only when used in tandem with the `log-slave-updates` option).

See replicate_annotate_row_events for a detailed description of the variable.

mariadb-binlog Option: `--skip-annotate-row-events`

This option tells mariadb-binlog to skip all `Annotate_row` events in its output (by default, mariadb-binlog prints `Annotate_row` events, if the binary log contains them).

Example of mariadb-binlog Output

```
...> mariadb-binlog.exe -vv -R --user=root --port=3306 --host=localhost master-bin.000001

/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#100516 15:36:00 server id 100  end_log_pos 240         Start: binlog v 4, server v 5.1.44-debug-log
created 100516
 15:36:00 at startup
ROLLBACK/*!*/;
BINLOG '
oNjvSw9kAAAA7AAAAPAAAAAAAAQANS4xLjQ0LWRlYnVnLWxvZwAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAACg2O9LEzgNAAgAEgAEBAQEEgAA2QAEGggAAAAICAgCAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAA=
'/*!*/;
# at 240
#100516 15:36:18 server id 100  end_log_pos 331         Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
SET @@session.pseudo_thread_id=1/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=1, @@session.unique_checks=1,
@@session.autocommit=1
/*!*/;
SET @@session.sql_mode=0/*!*/;
SET @@session.auto_increment_increment=1, @@session.auto_increment_offset=1/*!*/;
/*!\C latin1 *//*!*/;
SET @@session.character_set_client=8,@@session.collation_connection=8,@@session.collation_server=8/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
DROP DATABASE IF EXISTS test
/*!*/;
# at 331
#100516 15:36:18 server id 100  end_log_pos 414         Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
CREATE DATABASE test
/*!*/;
# at 414
#100516 15:36:18 server id 100  end_log_pos 499         Query   thread_id=1     exec_time=0
error_code=0
use test/*!*/;
SET TIMESTAMP=1274009778/*!*/;
CREATE TABLE t1(a int)
/*!*/;
# at 499
#100516 15:36:18 server id 100  end_log_pos 567         Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
BEGIN
/*!*/;
# at 567
# at 621
# at 662
#100516 15:36:18 server id 100  end_log_pos 621         Annotate_rows:
#Q> INSERT INTO t1 VALUES (1), (2), (3)
#100516 15:36:18 server id 100  end_log_pos 662         Table_map: `test`.`t1` mapped to number 16
#100516 15:36:18 server id 100  end_log_pos 706         Write_rows: table id 16 flags: STMT_END_F

BINLOG '
stjvSxNkAAAAKQAAAJYCAAAAABAAAAAAAAAABHRlc3QAAnQxAAEDAAE=
stjvSxdkAAAALAAAAMICAAAQABAAAAAAAAAEAAf/+AQAAAP4CAAAA/gMAAAA=
'/*!*/;
### INSERT INTO test.t1
### SET
###   @1=1 /* INT meta=0 nullable=1 is_null=0 */
```

```
### INSERT INTO test.t1
### SET
###   @1=2 /* INT meta=0 nullable=1 is_null=0 */
### INSERT INTO test.t1
### SET
###   @1=3 /* INT meta=0 nullable=1 is_null=0 */
# at 706
#100516 15:36:18 server id 100  end_log_pos 775         Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
COMMIT
/*!*/;
# at 775
#100516 15:36:18 server id 100  end_log_pos 860         Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
CREATE TABLE t2(a int)
/*!*/;
# at 860
#100516 15:36:18 server id 100  end_log_pos 928         Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
BEGIN
/*!*/;
# at 928
# at 982
# at 1023
#100516 15:36:18 server id 100  end_log_pos 982         Annotate_rows:
#Q> INSERT INTO t2 VALUES (1), (2), (3)
#100516 15:36:18 server id 100  end_log_pos 1023        Table_map: `test`.`t2` mapped to number 17
#100516 15:36:18 server id 100  end_log_pos 1067        Write_rows: table id 17 flags: STMT_END_F

BINLOG '
stjvSxNkAAAAKQAAAP8DAAAAABEAAAAAAAAABHRlc3QAAnQyAAEDAAE=
stjvSxdkAAAALAAAACsEAAAQABEAAAAAAAEAAf/+AQAAAP4CAAAA/gMAAAA=
'/*!*/;
### INSERT INTO test.t2
### SET
###   @1=1 /* INT meta=0 nullable=1 is_null=0 */
### INSERT INTO test.t2
### SET
###   @1=2 /* INT meta=0 nullable=1 is_null=0 */
### INSERT INTO test.t2
### SET
###   @1=3 /* INT meta=0 nullable=1 is_null=0 */
# at 1067
#100516 15:36:18 server id 100  end_log_pos 1136        Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
COMMIT
/*!*/;
# at 1136
#100516 15:36:18 server id 100  end_log_pos 1221        Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
CREATE TABLE t3(a int)
/*!*/;
# at 1221
#100516 15:36:18 server id 100  end_log_pos 1289        Query   thread_id=2     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
BEGIN
/*!*/;
# at 1289
# at 1351
# at 1392
#100516 15:36:18 server id 100  end_log_pos 1351        Annotate_rows:
#Q> INSERT DELAYED INTO t3 VALUES (1), (2), (3)
#100516 15:36:18 server id 100  end_log_pos 1392        Table_map: `test`.`t3` mapped to number 18
```

```
#100516 15:36:18 server id 100  end_log_pos 1426        Write_rows: table id 18 flags: STMT_END_F

BINLOG '
stjvSxNkAAAAKQAAAHAFAAAAABIAAAAAAAAABHRlc3QAAnQzAAEDAAE=
stjvSxdkAAAAIgAAAJIFAAAQABIAAAAAAAEAAf/+AQAAAA==
'/*!*/;
### INSERT INTO test.t3
### SET
###   @1=1 /* INT meta=0 nullable=1 is_null=0 */
# at 1426
# at 1467
#100516 15:36:18 server id 100  end_log_pos 1467        Table_map: `test`.`t3` mapped to number 18
#100516 15:36:18 server id 100  end_log_pos 1506        Write_rows: table id 18 flags: STMT_END_F

BINLOG '
stjvSxNkAAAAKQAAALsFAAAAABIAAAAAAAAABHRlc3QAAnQzAAEDAAE=
stjvSxdkAAAAJwAAAOIFAAAQABIAAAAAAAEAAf/+AgAAAP4DAAAA
'/*!*/;
### INSERT INTO test.t3
### SET
###   @1=2 /* INT meta=0 nullable=1 is_null=0 */
### INSERT INTO test.t3
### SET
###   @1=3 /* INT meta=0 nullable=1 is_null=0 */
# at 1506
#100516 15:36:18 server id 100  end_log_pos 1575        Query   thread_id=2     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
COMMIT
/*!*/;
# at 1575
#100516 15:36:18 server id 100  end_log_pos 1643        Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
BEGIN
/*!*/;
# at 1643
# at 1748
# at 1789
# at 1830
# at 1874
#100516 15:36:18 server id 100  end_log_pos 1748        Annotate_rows:
#Q> DELETE t1, t2 FROM t1 INNER JOIN t2 INNER JOIN t3
#Q>        WHERE t1.a=t2.a AND t2.a=t3.
#100516 15:36:18 server id 100  end_log_pos 1789        Table_map: `test`.`t1` mapped to number 16
#100516 15:36:18 server id 100  end_log_pos 1830        Table_map: `test`.`t2` mapped to number 17
#100516 15:36:18 server id 100  end_log_pos 1874        Delete_rows: table id 16
#100516 15:36:18 server id 100  end_log_pos 1918        Delete_rows: table id 17 flags: STMT_END_F

BINLOG '
stjvSxNkAAAAKQAAAP0GAAAAABAAAAAAAAAABHRlc3QAAnQxAAEDAAE=
stjvSxNkAAAAKQAAACYHAAAAABEAAAAAAAAABHRlc3QAAnQyAAEDAAE=
stjvSxlkAAAALAAAAFIHAAAAABAAAAAAAAAAAf/+AQAAAP4CAAAA/gMAAAA=
### DELETE FROM test.t1
### WHERE
###   @1=1 /* INT meta=0 nullable=1 is_null=0 */
### DELETE FROM test.t1
### WHERE
###   @1=2 /* INT meta=0 nullable=1 is_null=0 */
### DELETE FROM test.t1
### WHERE
###   @1=3 /* INT meta=0 nullable=1 is_null=0 */
stjvSxlkAAAALAAAAH4HAAAQABEAAAAAAAEAAf/+AQAAAP4CAAAA/gMAAAA=
'/*!*/;
### DELETE FROM test.t2
### WHERE
###   @1=1 /* INT meta=0 nullable=1 is_null=0 */
### DELETE FROM test.t2
### WHERE
```

```
###   @1=2 /* INT meta=0 nullable=1 is_null=0 */
### DELETE FROM test.t2
### WHERE
###   @1=3 /* INT meta=0 nullable=1 is_null=0 */
# at 1918
#100516 15:36:18 server id 100  end_log_pos 1987        Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1274009778/*!*/;
COMMIT
/*!*/;
DELIMITER ;
# End of log file
ROLLBACK /* added by mariadb-binlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
```

See Also

- mariadb-binlog Options
- Replication and Binary Log Server System Variables
- Full List of MariaDB Options, System and Status Variables
- mariadbd Options

This page is licensed: CC BY-SA / Gnu FDL

mysqlbinlog

`mysqlbinlog` is a client tool which is called `mariadb-binlog` now. It can still be accessed under its original name via a symlink in Linux, or an alternate binary in Windows.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-dumpslow

> **Note:**
>
> Previously, the client was called `mysqldumpslow`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

`mariadb-dumpslow` is a tool to examine the slow query log.

It parses the slow query log files, printing a summary result. Normally, `mariadb-dumpslow` groups queries that are similar, except for the particular values of number and string data values. It “abstracts” these values to N and ´S´ when displaying summary output. The `-a` and `-n` options can be used to modify value abstracting behavior.

Usage

```
mariadb-dumpslow [ options... ] [ logs... ]
```

Options

`-a`

Don’t abstract all numbers to N and strings to ‘S’.

`-d`, **`--debug`**

Debug mode.

`-g`` *pattern*

Grep: only consider statements that include this string *pattern*.

`--help`

Display help.

`-h`` *hostname*

Hostname of db server for -slow.log filename (can be a wildcard). The Default is `''`, that is, match all hosts.

`-i`` *name*

Name of server instance (if using mysql.server startup script).

`-l`

Don't subtract lock time from total time.

`-j`, **`--json`**

Stores the dumped data in JSON format. Available from MariaDB 12.1.

`-n`` *number*

Abstract numbers with at least this *number* of digits within names.

`-r`

Reverse the sort order (largest last instead of first).

`-s`` *order*

What to sort by (`aa`, `ae`, `al`, `ar`, `at`, `a`, `c`, `e`, `l`, `r`, `t`). The default is `at`. The meaning of the abbreviations is:

- `aa` – average rows affected
- `ae` – aggregated number of rows examined
- `al` – average lock time
- `ar` – average rows sent
- `at` – average query time
- `a` – rows affected
- `c` – count
- `e` – rows examined
- `l` – lock time
- `r` – rows sent
- `t` – query time:

`-t`` *number*

Just show the top *number of queries*.

`-v`, **`--verbose`**

Verbose mode.

This page is licensed: CC BY-SA / Gnu FDL

mariadb Client

See mariadb Command-Line Client.

Complete MariaDB command-line client guide. Complete reference for server connections, query execution, data import, and batch processing for production use.

See mysql Command-Line Client.

The mysql command-line client is the former name of the mariadb SQL shell, still available as a symlink or alternate binary for compatibility.

This page is licensed: CC BY-SA / Gnu FDL

mariadb Command-Line Client

`mariadb` is a simple SQL shell with GNU readline capabilities.

> **CURRENT**
>
> The command-line client is called `mariadb`. On Unix system, a symlink named `mysql` is available. On Windows, an alternative binary named `mysql.exe` is available.
>
> **< 10.5**
>
> The command-line client is called `mysql`.

Overview

`mariadb` supports interactive and non-interactive use. When used interactively, query results are presented in an ASCII-table format. When used non-interactively (for example, as a filter), the result is presented in tab-separated format. The output format can be changed using command options.

If you have problems due to insufficient memory for large result sets, use the `--quick` option. This forces mariadb to retrieve results from the server a row at a time rather than retrieving the entire result set and buffering it in memory before displaying it. This is done by returning the result set using the `mysql_use_result()` C API function in the client/server library rather than `mysql_store_result()`.

Invoke the client from the prompt of your command interpreter (for instance, `cmd` on Windows, `Terminal` on macOS) as follows:

```
mariadb db_name
```

Alternatively, start the client and log on to MariaDB server:

```
mariadb --user=user_name --password=your_password db_name
```

Type an SQL statement, end it with a semicolon (`;`), `\g`, or `\G` , and press `Enter`.

Typing `Control-C` causes `mariadb` to attempt to kill the current statement. If this cannot be done, or `Control-C` is typed again before the statement is killed, `mariadb` exits.

You can execute SQL statements in a script file (batch file) like this:

```
mariadb db_name < script.sql > output.tab
```

Usage

The client's general syntax is:

```
mariadb <options>
```

Options

`mariadb` supports the following options:

`-?, --help`

Display help and exit.

`-I, --help`

Synonym for `-?`

`--abort-source-on-error`

Abort ‘source filename’ operations in case of errors.

`--auto-rehash`

Enable automatic rehashing. This option is on by default, which enables database, table, and column name completion. Use `--disable-auto-rehash`, `--no-auto-rehash` or `skip-auto-rehash` to disable rehashing. That causes mariadb to start faster, but you must issue the rehash command if you want to use name completion. To complete a name, enter the first part and press Tab. If the name is unambiguous, mariadb completes it. Otherwise, you can press Tab again to see the possible names that begin with what you have typed so far. Completion does not occur if there is no default database.

`-A, --no-auto-rehash`

No automatic rehashing. One has to use ‘rehash’ to get table and field completion. This gives a quicker start of mariadb and disables rehashing on reconnect.

`--auto-vertical-output`

Automatically switch to vertical output mode if the result is wider than the terminal width.

`-B, --batch`

Print results using tab as the column separator, with each row on a new line. With this option, mariadb does not use the history file. Batch mode results in nontabular output format and escaping of special characters. Escaping may be disabled by using raw mode; see the description for the `--raw` option. (Enables `--silent`.)

`--binary-mode`

By default, `ASCII '\0'` is disallowed and `\r\n` is translated to `\n`. This switch turns off both features, and also turns off parsing of all client commands except `\C` and `DELIMITER`, in non-interactive mode (for input piped to mariadb or loaded using the `source` command). This is necessary when processing output from mariadb-binlog that may contain blobs.

`--character-sets-dir=name`

Directory for character set files.

`--column-names`

Write column names in results. (Defaults to `ON`; use `--skip-column-names` to disable.)

`--column-type-info`

Display column type information.

`-c, --comments`

Preserve comments. Send comments to the server. The default is `--skip-comments` (discard comments), enable with `--comments`.

`-C, --compress`

Compress all information sent between the client and the server if both support compression.

`--connect-expired-password`

Notify the server that this client is prepared to handle expired password sandbox mode even if `--batch` was specified.

`--connect-timeout=`*`num`*

Number of seconds before connection timeout. Defaults to `0`.

`-D, --database=`*`name`*

Database to use.

`-#[`*`options`*`], --debug[=`*`options`*`]`

On debugging builds, write a debugging log. A typical `debug_options` string is `d:t:o,file_name`. The default is `d:t:o,/tmp/mysql.trace`.

`--debug-check`

Check memory and open file usage at exit.

`-T, --debug-info`

Print some debug info at exit.

`--default-auth=`*`plugin`*

Default authentication client-side plugin to use.

`--default-character-set=`*`name`*

Set the default character set. A common issue that can occur when the operating system uses utf8 or another multibyte character set is that output from the mariadb client is formatted incorrectly, due to the fact that the MariaDB client uses the latin1 character set by default. You can usually fix such issues by using this option to force the client to use the system character set instead. If set to `auto` the character set is taken from the client environment (`LC_CTYPE` on Unix).

`--defaults-extra-file=`*`file`*

Read this *file* after the global files are read. Must be given as the first option.

`--defaults-file=`*`file`*

Only read default options from the given *file*. Must be given as the first option.

`--defaults-group-suffix=`*`suffix`*

In addition to the given groups, also read groups with this *suffix*.

`--delimiter=`*`name`*

Delimiter to be used. The default is the semicolon (`;`).

`--enable-cleartext-plugin`

Obsolete option. Exists only for MySQL compatibility.

`-e, --execute=`*`name`*

Execute statement and quit. Disables `--force` and history file. The default output format is like that produced with `--batch`.

`-f, --force`

Continue even if we get an SQL error. Sets `--abort-source-on-error` to `0`.

`-h, --host=`*`host`*

Connect to *host*.

`-H, --html`

Produce HTML output.

`-U, --i-am-a-dummy`

Synonym for option `--safe-updates`, `-U`.

`-i, --ignore-spaces`

Ignore space after function names. Allows one to have spaces (including tab characters and new line characters) between function name and '('. The drawback is that this causes built in functions to become reserved words.

`--init-command=str`

SQL Command to execute when connecting to the MariaDB server. Will automatically be re-executed when reconnecting.

`--line-numbers`

Write line numbers for errors. (Defaults to `ON`; use `--skip-line-numbers` to disable.)

`--local-infile`

Enable or disable LOCAL capability for LOAD DATA INFILE. With no value, the option enables `LOCAL`. The option may be given as `--local-infile=0` or `--local-infile=1` to explicitly disable or enable `LOCAL`. Enabling `LOCAL` has no effect if the server does not also support it.

`--max-allowed-packet=`*`num`*

The maximum packet length to send to or receive from server. The default is 16MB, the maximum 1GB.

`--max-join-size=`*`num`*

Automatic limit for rows in a join when using `--safe-updates`. Default is `1000000`.

`-G, --named-commands`

Enable named commands. Named commands mean mariadb's internal commands (see below) . When enabled, the named commands can be used from any line of the query, otherwise only from the first line, before an enter. Long-format commands are allowed, not just short-format commands. For example, `quit` and `\q` are both recognized. Disable with `--disable-named-commands`. This option is disabled by default.

`--net-buffer-length=`*`num`*

The buffer size for TCP/IP and socket communication. Default is 16KB.

`-b, --no-beep`

Turn off beep on error.

`--no-defaults`

Don't read default options from any option file. Must be given as the first option.

`-o, --one-database`

Ignore statements except those that occur while the default database is the one named on the command line. This filtering is limited, and based only on USE statements. This is useful for skipping updates to other databases in the binary log.

`--pager[=name]`

Pager to use to display results (Unix only). If you don't supply an option, the default pager is taken from your ENV variable PAGER. Valid pagers are *less*, *more*, *cat [> filename]*, etc. See interactive help (\h) also. This option does not work in batch mode. Disable with `--disable-pager`. This option is disabled by default.

`-p, --password[=`*`password`*`]`

Password to use when connecting to server. If you use the short option form (`-p`), you cannot have a space between the option and the password. If you omit the password value following the `--password` or `-p` option on the command line, mariadb prompts for one. Specifying a password on the command line should be considered insecure. You can use an option file to avoid giving the password on the command line.

`--plugin-dir=`*`name`*

Directory for client-side plugins.

`-P, --port=`*`num`*

Port number to use for connection or `0` for default to, in order of preference, `my.cnf`, `$MYSQL_TCP_PORT`, `/etc/services`, built-in default (`3306`).

`--print-defaults`

Print the program argument list and exit. Must be given as the first option.

`--progress-reports`

Get progress reports for long running commands (such as ALTER TABLE). (Defaults to `ON`; use `--skip-progress-reports` to disable.)

`--prompt=`*`name`*

Set the mariadb prompt to this value. See prompt command for options.

`--protocol=`*`name`*

The protocol to use for connection (`tcp`, `socket`, `pipe`, `memory`).

`-q, --quick`

Don't cache result, print it row by row. This may slow down the server if the output is suspended. Doesn't use history file.

`--quick-max-column-width=N`

> **CURRENT**
>
> Maximal field length limit in case of `--quick`.
>
> **< 11.7.1 / 11.6.2 / 11.4.4 / 10.11.10 / 10.6.20 / 10.5.27**
>
> This option is not available.

`-r, --raw`

For tabular output, the "boxing" around columns enables one column value to be distinguished from another. For nontabular output (such as is produced in batch mode or when the `--batch` or `--silent` option is given), special characters are escaped in the output so they can be identified easily. Newline, tab, NUL, and backslash are written as `\n`, `\t`, `\0`, and `\\`. The `--raw` option disables this character escaping.

`--reconnect`

Reconnect if the connection is lost. This option is enabled by default. Disable with `--disable-reconnect` or `skip-reconnect`.

`-U, --safe-updates`

Allow only those UPDATE and DELETE statements that specify which rows to modify by using key values. If you have set this option in an option file, you can override it by using `--safe-updates` on the command line. See using the –safe-updates option for more.

`--sandbox`

> **CURRENT**
>
> Disallow commands that access the file system (except `\P` without an argument and `\e`). Disabled commands include system (`\!`), tee (`\T`), pager with an argument(`\P`` ``` *foo*), source (`\.`). Using a disabled command is an error, which can be ignored with `--force`. A sandbox command (`\-`) enables the sandbox mode until EOF (current file or the session, if interactive).
>
> **< 11.4.2 / 11.2.4 / 11.1.5 / 11.0.6 / 10.11.8 / 10.6.18 / 10.5.25**
>
> This option is not available.

`--script-dir=`*`name`*

> **CURRENT**
>
> Sets an alternative directory path for searching scripts invoked via the source command.
>
> **< 12.0**

> This option is not available.

`--secure-auth`

Refuse client connecting to server if it uses old (pre-MySQL4.1.1) protocol. Defaults to `FALSE`.

`--select-limit=`*`num`*

Automatic limit for `SELECT` when using `--safe-updates`. Default `1000`.

`--server-arg=`*`name`*

Send embedded server this as a parameter.

`--shared-memory-base-name=`*`name`*

Shared-memory name to use for Windows connections using shared memory to a local server (started with the `--shared-memory` option). Case-sensitive.

`--show-serveroutput`

Enables displaying of `DBMS_OUTPUT` messages. See this page for details.

`--show-warnings`

Show warnings after every statement. Applies to interactive and batch mode.

`--sigint-ignore`

Ignore `SIGINT` signals (usually `CTRL-C`).

`-s, --silent`

Be more silent. This option can be given multiple times to produce less and less output. This option results in nontabular output format and escaping of special characters. Escaping may be disabled by using raw mode; see the description for the `--raw` option.

`--skip-auto-rehash`

Disable automatic rehashing. See `--auto-rehash`.

`-N, --skip-column-names`

Don't write column names in results. See `--column-names`.

`--skip-comments`

Discard comments. Set by default, see `--comments` to enable.

`-L`, `--skip-line-numbers`

Don't write line number for errors. See `--line-numbers`.

`--skip-progress-reports`

Disables getting progress reports for long running commands. See `--progress-reports`.

`--skip-reconnect`

Don't reconnect if the connection is lost. See `--reconnect`.

`-S, --socket=name`

For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use. The socket file can exist in different locations depending on setup. Common locations include:

- Debian-based, Ubuntu: `/var/run/mysqld/mysqld.sock`
- SUSE: `/var/run/mysql/mysql.sock`
- Red Hat: `/var/lib/mysql/mysql.sock`
- Other: `/tmp/mysql.sock`

`--ssl`

> **CURRENT**
>
> Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The `--ssl` option does not enable verifying the server certificate by default. In order to verify the server certificate, the user must specify the `--ssl-verify-server-cert` option.
>
> TLS with `--ssl` is enabled by default.
>
> **< 10.10**
>
> Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The `--ssl` option does not enable verifying the server certificate by default. In order to verify the server certificate, the user must specify the `--ssl-verify-server-cert` option.

`--ssl-ca=name`

Defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option implies the `--ssl` option.

`--ssl-capath=name`

Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the `--ssl` option.

`--ssl-cert=name`

Defines a path to the X509 certificate file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

`--ssl-cipher=name`

List of permitted ciphers or cipher suites to use for TLS. This option implies the `--ssl` option.

`--ssl-crl=name`

Defines a path to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL or

Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

`--ssl-crlpath=`*`name`*

Defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

`--ssl-key=`*`name`*

Defines a path to a private key file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

`--ssl-verify-server-cert`

> **CURRENT**
>
> Enables server certificate verification. This option is enabled by default. Use `--disable-ssl` or `--disable-ssl-verify-server-cert` to revert this behavior.
>
> **< 11.4**
>
> Enables server certificate verification. This option is disabled by default.

`-t, --table`

Display output in table format. This is the default for interactive use, but can be used to produce table output in batch mode.

`--tee=`*`name`*

Append everything into outfile. See interactive help (`\h`) also. Does not work in batch mode. Disable with `--disable-tee`. This option is disabled by default.

`--tls-version=`*`name`*

This option accepts a comma-separated list of TLS protocol versions. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted. See Secure Connections Overview: TLS Protocol Versions for more information.

`--ssl-fp=`*`name`*

Server certificate fingerprint (implies `--ssl`).

`--ssl-fplist=`*`name`*

File with accepted server certificate fingerprints, one per line (implies `--ssl`).

`-n, --unbuffered`

Flush buffer after each query.

`-u`, `--user=` ***`name`***

User for login if not current user.

`-v, --verbose`

Write more. (`-v -v -v` gives the table output format).

`-V, --version`

Output version information and exit.

`-E, --vertical`

Print the output of a query (rows) vertically. Use the `\G` delimiter to apply to a particular statement if this option is not enabled.

`-w, --wait`

If the connection cannot be established, wait and retry instead of aborting.

`-X`, `--xml`

Produce XML output. See the mariadb-dump –xml option for more.

Option Files

In addition to reading options from the command line, `mariadb` can also read options from option files. If an unknown option is provided to `mariadb` in an option file, then it is ignored.

The following options relate to how MariaDB command line tools handles option files. They must be given as the first argument on the command line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix. If not set, MariaDB checks the $MYSQL_GROUP_SUFFIX environment variable. From MariaDB 12.0.1, MariaDB 11.8.2, MariaDB 11.4.6, MariaDB 10.11.12, `$MARIADB_GROUP_SUFFIX` is also checked, and takes precedence if both are set.

`mariadb` is linked with MariaDB Connector/C. However, MariaDB Connector/C does not handle the parsing of option files for this client. That is performed by the server option file parsing code. See MDEV-19035 for more information.

Option Groups

`mariadb` reads options from the following option groups from option files:

Group	Description
[mysql]	Options read by mysql, which includes both MariaDB Server and MySQL Server.

Group	Description
[mariadb-client]	Options read by mariadb.
[client]	Options read by all MariaDB and MySQL client programs, which includes both MariaDB and MySQL clients. For example, mysqldump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

Delimiters

The default delimiter in the mariadb client is the semicolon.

When creating stored programs from the command line, it is likely you will need to differentiate between the regular delimiter and a delimiter inside a BEGIN END block. Consider the following example:

```
CREATE FUNCTION FortyTwo() RETURNS TINYINT DETERMINISTIC
BEGIN
 DECLARE x TINYINT;
 SET x = 42;
 RETURN x;
END;
```

If you enter the above line by line, the mariadb client will treat the first semicolon, at the end of the `DECLARE x TINYINT` line, as the end of the statement. Since that's only a partial definition, it will throw a syntax error, as follows:

```
CREATE FUNCTION FortyTwo() RETURNS TINYINT DETERMINISTIC
BEGIN
DECLARE x TINYINT;
ERROR 1064 (42000): You have an error in your SQL syntax;
check the manual that corresponds to your MariaDB server version
 for the right syntax to use near '' at line 3
```

The solution is to specify a distinct delimiter for the duration of the process, using the `DELIMITER`. The delimiter can be any set of characters you choose, but it needs to be a distinctive set of characters that won't cause further confusion. `//` is a common choice, and used throughout the documentation.

Here's how the function can be entered, using the new delimiter:

```
DELIMITER //

CREATE FUNCTION FortyTwo() RETURNS TINYINT DETERMINISTIC
BEGIN
  DECLARE x TINYINT;
  SET x = 42;
  RETURN x;
END

//

DELIMITER ;
```

At the end, the delimiter is restored to the default semicolon. The `\g` and `\G` delimiters can always be used, even when a custom delimiter is specified.

Specify Which Protocol to Use

You can force which protocol are used to connect to the `mariadbd` server, by giving the `protocol` option one of the following values: `tcp`, `socket`, `pipe`, or `memory`.

> **CURRENT**
>
> A connection property specified via the command line (e.g. `--port=3306`) forces its type. The protocol that matches the respective connection property is used. For instance, a TCP/IP connection is created when `--port` is specified.
>
> **< 10.6.1**
>
> If `protocol` is not specified, command-line connection properties that do not force protocol are ignored.

If multiple or no connection properties are specified via the command line, the following happens on Unix and Windows systems:

Unix

- If `hostname` is not specified or `hostname` is `localhost`, Unix sockets are used.
- In other cases (`hostname` is given and it's not `localhost`) then a TCP/IP connection through the `port` option is used.

> **Warning:**
>
> `localhost` is a special value. Using `127.0.0.1` is not the same thing. The latter will connect to the `mariadbd` server through TCP/IP.

Windows

- If `shared-memory-base-name` is specified and `hostname` is not specified or `hostname` is `localhost`, then the connection will happen through shared memory.
- If `shared-memory-base-name` is not specified and `hostname` is not specified or `hostname` is `localhost`, then the connection will happen through windows named pipes.
- Named pipes will also be used if the `libmysql` / `libmariadb` client library detects that the client doesn't support TCP/IP.
- In other cases, a TCP/IP connection through the `port` option is used.

Test Which Protocol is Used

The `status` command shows you information about which protocol is used:

```
shell> mariadb test

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 11.4.1-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [test]> status;
--------------
mysql  Ver 15.1 Distrib 10.0.25-MariaDB, for Linux (x86_64) using readline 5.2

Connection id:          10
Current database:       test
Current user:           monty@localhost
...
Connection:             Localhost via UNIX socket
...
UNIX socket:            /tmp/mysql-dbug.sock
```

mariadb Commands

There are also a number of commands that can be run inside the client. Note that all text commands must be first on the line and end with a semicolon (`;`).

Command	Description
-	Enables sandbox mode until EOF (current file or the session, if interactive). From MariaDB 10.5.25, MariaDB 10.6.18, MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5, MariaDB 11.2.4, MariaDB 11.4.2.
?, ?	Synonym for `help'.
clear, \c	Clear the current input statement.
connect,	Reconnect to the server. Optional arguments are db and host.
delimiter, \d	Set statement delimiter.
edit, \e	Edit command with $EDITOR.
ego, \G	Send command to mariadb server, display result vertically.
exit, \q	Exit mariadb. Same as quit.
go, \g	Send command to mariadb server.
help, \h	Display this help.
nopager,	Disable pager, print to stdout.
noserveroutput, \o	Disables displaying of `DBMS_OUTPUT` messages. See this page for details.
notee,	Don't write into outfile.
pager, \P	Set PAGER [to_pager]. Print the query results via PAGER.
print, \p	Print current command.
prompt,	Change your mariadb prompt.
quit, \q	Quit mariadb.
rehash, #	Rebuild completion hash.
serveroutput, \O	Enables displaying of `DBMS_OUTPUT` messages. See this page for details.
source, .	Execute an SQL script file. Takes a file name as an argument. Usually looks in the working directory, unless, from MariaDB 12.0, a path is given with `--script-dir`.

Command	Description
status, \s	Get status information from the server.
system, !	Execute a system shell command. Only works in Unix-like systems.
tee, \T	Set outfile [`to_outfile`]. Append everything into given outfile.
use, \u	Use another database. Takes database name as argument.
charset, \C	Switch to another charset. Might be needed for processing binlog with multi-byte charsets.
warnings, \W	Show warnings after every statement.
nowarning, \w	Don't show warnings after every statement.

The mysql_history File

On Unix, the mariadb client writes a record of executed statements to a history file. By default, this file is named `.mysql_history` and is created in your home directory. To specify a different file, set the value of the `MYSQL_HISTFILE` environment variable.

The `.mysql_history` file should be protected with a restrictive access mode because sensitive information might be written to it, such as the text of SQL statements that contain passwords.

If you do not want to maintain a history file, first remove `.mysql_history` if it exists, and use either of the following techniques:

- Set the `MYSQL_HISTFILE` variable to `/dev/null`. To cause this setting to take effect each time you log in, put the setting in one of your shell's startup files.
- Create `.mysql_history` as a symbolic link to `/dev/null`:

```
shell> ln -s /dev/null $HOME/.mysql_history
```

You need do this only once.

prompt Command

The prompt command reconfigures the default prompt `\N [\d]>`. The string for defining the prompt can contain the following special sequences.

Option	Description
\c	A counter that increments for each statement you issue.
\D	The full current date.
\d	The default database.
\h	The server host.
\l	The current delimiter.
\m	Minutes of the current time.
\n	A newline character.
\O	The current month in three-letter format (Jan, Feb, …).
\o	The current month in numeric format.
\P	am/pm.
\p	The current TCP/IP port or socket file.
\R	The current time, in 24-hour military time (0–23).

Option	Description
\r	The current time, standard 12-hour time (1–12).
\S	Semicolon.
\s	Seconds of the current time.
\t	A tab character.
\U	Your full user_name@host_name account name.
\u	Your user name.
\v	The server version.
\w	The current day of the week in three-letter format (Mon, Tue, …).
\Y	The current year, four digits.
\y	The current year, two digits.
_	A space.
\	A space (a space follows the backslash).
’	Single quote.
”	Double quote.
\\	A literal “” backslash character.
\x	x, for any “x” not listed above.

mariadb Tips

This section describes some techniques that can help you use `mariadb` more effectively.

Displaying Query Results Vertically

Some query results are much more readable when displayed vertically, instead of in the usual horizontal table format. Queries can be displayed vertically by terminating the query with `\G` instead of a semicolon (`;`). For example, longer text values that include newlines often are much easier to read with vertical output:

```
mariadb> SELECT * FROM mails WHERE LENGTH(txt) < 300 LIMIT 300,1\G
*************************** 1. row ***************************
  msg_nro: 3068
    date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Monty
    reply: monty@no.spam.com
  mail_to: "Thimble Smith" <tim@no.spam.com>
      sbj: UTF-8
      txt: >>>>> "Thimble" == Thimble Smith writes:
Thimble> Hi.  I think this is a good idea.  Is anyone familiar
Thimble> with UTF-8 or Unicode? Otherwise, I´ll put this on my
Thimble> TODO list and see what happens.
Yes, please do that.
Regards,
Monty
    file: inbox-jani-1
    hash: 190402944
1 row in set (0.09 sec)
```

Using the –safe-updates Option

For beginners, a useful startup option is `--safe-updates` (or `--i-am-a-dummy`, which has the same effect). It is helpful for cases when you might have issued a `DELETE FROM tbl_name` statement but forgotten the `WHERE` clause.

Normally, such a statement deletes all rows from the table. With `--safe-updates`, you can delete rows only by specifying the key values that identify them. This helps prevent accidents.

When you use the `--safe-updates` option, mariadb issues the following statement when it connects to the MariaDB server:

```
SET sql_safe_updates=1, sql_select_limit=1000, sql_max_join_size=1000000;
```

The SET statement has the following effects:

- You are not allowed to execute an UPDATE or DELETE statement unless you specify a key constraint in the `WHERE` clause or provide a `LIMIT` clause (or both). For example:

```
UPDATE tbl_name SET not_key_column=val WHERE key_column=val;
UPDATE tbl_name SET not_key_column=val LIMIT 1;
```

- The server limits all large `SELECT` results to 1,000 rows unless the statement includes a `LIMIT` clause.
- The server aborts multiple-table `SELECT` statements that probably need to examine more than 1,000,000 row combinations.

To specify limits different from 1,000 and 1,000,000, you can override the defaults by using the `--select_limit` and `--max_join_size` options:

```
mariadb --safe-updates --select_limit=500 --max_join_size=10000
```

Disabling mariadb Auto-Reconnect

If the mariadb client loses its connection to the server while sending a statement, it immediately and automatically tries to reconnect once to the server and send the statement again. However, even if mariadb succeeds in reconnecting, your first connection has ended and all your previous session objects and settings are lost: temporary tables, the autocommit mode, and user-defined and session variables. Also, any current transaction rolls back. This behavior may be dangerous for you, as in the following example where the server was shut down and restarted between the first and second statements without you knowing it:

```
mariadb> SET @a=1;
Query OK, 0 rows affected (0.05 sec)
mariadb> INSERT INTO t VALUES(@a);
ERROR 2006: MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    1
Current database: test
Query OK, 1 row affected (1.30 sec)
mariadb> SELECT * FROM t;
+------+
| a    |
+------+
| NULL |
+------+
```

The `@a` user variable has been lost with the connection, and after that, the reconnection it is undefined. If it is important to have mariadb terminate with an error if the connection has been lost, you can start the mariadb client with the `--skip-reconnect` option.

See Also

- Troubleshooting Connection Issues
- Readline commands and configuration

This page is licensed: CC BY-SA / Gnu FDL

mysql Command-Line Client

> **Note:**
>
> The client is now called `mariadb`. The old name, `mysql`, can still be used, via a symlink in Linux, or an alternate binary in Windows.

`mysql` is a simple SQL shell with GNU readline capabilities.

This page is licensed: GPLv2

Networking Tools

See resolveip.

Resolve host names to IP addresses and the reverse with the resolveip utility, useful when configuring host-based privileges.

This page is licensed: CC BY-SA / Gnu FDL

resolveip

resolveip is a utility for resolving IP addresses to host names and vice versa.

Usage

```
resolveip [OPTIONS] hostname or IP-address
```

Options

Option	Description
Option	Description
-?, –help	Display help and exit.
-I, –info	Synonym for –help.
-s, –silent#	Be more silent.
-V, –version	Display version information and exit.

Examples

```
shell> resolveip mariadb.org
IP address of mariadb.org is 166.78.144.191
```

```
resolveip 166.78.144.191
Host name of 166.78.144.191 is mariadb.org
```

This page is licensed: CC BY-SA / Gnu FDL

Table Tools

See mariadb-check.

Complete mariadb-check reference: CHECK/REPAIR/ANALYZE/OPTIMIZE TABLE operations, –all-databases/–databases flags, –auto-repair, and option files.

See mariadb-convert-table-format.

Convert every table in a database to a given storage engine (MyISAM by default) with mariadb-convert-table-format (formerly mysql_convert_table_format).

See mariadb-fix-extensions.

Fix the file extensions of MyISAM table files with mariadb-fix-extensions (formerly mysql_fix_extensions).

This page is licensed: CC BY-SA / Gnu FDL

mariadb-check

Overview

`mariadb-check` is a maintenance tool that allows you to check, repair, analyze and optimize multiple tables from the command line.

> **Note:**
>
> Previously, the client was called `mysqlcheck`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

It is essentially a command-line interface to the CHECK TABLE, REPAIR TABLE, ANALYZE TABLE, and OPTIMIZE TABLE statements, and so, unlike myisamchk and aria_chk, requires the server to be running.

> **Warning:**
>
> This tool does not work with partitioned tables.

Usage

```
./client/mariadb-check [OPTIONS] database [tables]
```

OR

```
./client/mariadb-check [OPTIONS] --databases DB1 [DB2 DB3...]
```

OR

```
./client/mariadb-check [OPTIONS] --all-databases
```

`mariadb-check` can be used to `CHECK` (`-c`, `-m`, `-C`), `REPAIR` (`-r`), `ANALYZE` (`-a`), or `OPTIMIZE` (`-o`) tables. Some of the options (like `-e` or `-q`) can be used in combination.

> **Warning:**
>
> Not all options are supported by all storage engines.

The `-c`, `-r`, `-a`, and `-o` options are exclusive to each other.

The `--check` option is used by default, if no other options were specified. You can change the default behavior by making a symbolic link to the binary, or copying it somewhere with another name, the alternatives are:

Tool	Default Option
mysqlrepair	The default option is `-r` (`--repair`).
mysqlanalyze	The default option is `-a` (`--analyze`).
mysqloptimize	The default option is `-o` (`--optimize`).

Options

`mariadb-check` supports the following options:

Option	Description
-A, –all-databases	Check all the databases. This is the same as `--databases` with all databases selected.
-1, –all-in-1	Instead of issuing one query for each table, use one query per database, naming all tables in the database in a comma-separated list.
-a, –analyze	Analyze given tables.
–auto-repair	If a checked table is corrupted, automatically fix it. Repairing is done after all tables have been checked.
–character-sets-dir=name	Directory where character set files are installed.
-c, –check	Check table for errors.
-C, –check-only-changed	Check only tables that have changed since last check or haven't been closed properly.
-g, –check-upgrade	Check tables for version-dependent changes. May be used with `--auto-repair` to correct tables requiring version-dependent updates. Automatically enables the `--fix-db-names` and `--fix-table-names` options. Used when upgrading.
–compress	Compress all information sent between the client and server if both support compression.
-B, –databases	Check several databases. Note that normally mariadb-check treats the first argument as a database name, and following arguments as table names. With this option, no tables are given, and all name arguments are regarded as database names.
## , –debug[=name]	Output debug log. Often this is `d:t:o,filename`.
–debug-check	Check memory and open file usage at exit.
–debug-info	Print some debug info at exit.
–default-auth=plugin	Default authentication client-side plugin to use.

Option	Description
–default-character-set=name	Set the default character set.
-e, –extended	Used with `--check`, it ensures that the table is fully consistent, but this takes a long time. Used with `--repair`, it forces the usage of the old, slow, repair mechanism with keycache method, instead of the much faster repair by sorting.
-F, –fast	Check only tables that haven't been closed properly.
–fix-db-names	Convert database names to the format used since MySQL 5.1. Only database names that contain special characters are affected. Used when upgrading from an old MySQL version.
–fix-table-names	Convert table names (including views) to the format used since MySQL 5.1. Only table names that contain special characters are affected. Used when upgrading from an old MySQL version.
–flush	Flush each table after check. This is useful if you don't want to have the checked tables take up space in the caches after the check.
-f, –force	Continue even if we get an SQL error.
-?, –help	Display this help message and exit.
-h name, –host=name	Connect to the given host.
-m, –medium-check	Faster than extended-check, but only finds 99.99 percent of all errors. This should be good enough for most cases.
-o, –optimize	Optimize tables.
-p, –password[=name]	Password to use when connecting to the server. If you use the short option form (`-p`), you cannot have a space between the option and the password. If you omit the password value following the `--password` or `-p` option on the command line, mariadb-check prompts for one. Specifying a password on the command line should be considered insecure. You can use an option file to avoid giving the password on the command line.
-Z, –persistent	When using `ANALYZE TABLE` (`--analyze`), it uses the `PERSISTENT FOR ALL` option, which forces Engine-independent Statistics for this table to be updated.
-W, –pipe	On Windows, connect to the server via a named pipe. This option applies only if the server supports named-pipe connections.
–plugin-dir	Directory for client-side plugins.

Option	Description
-P num, –port=num	Port number to use for connection or `0` for default to, in order of preference, `my.cnf`, `$MYSQL_TCP_PORT`, `/etc/services`, built-in default (`3306`).
–process-tables	Perform the requested operation (check, repair, analyze, optimize) on tables. Enabled by default. Use `--skip-process-tables` to disable.
–process-views[=val]	Perform the requested operation (only CHECK VIEW or REPAIR VIEW). Possible values are `NO`, `YES` (correct the checksum, if necessary, add the mariadb-version field), `UPGRADE_FROM_MYSQL` (same as `YES` and toggle the algorithm `MERGE<->TEMPTABLE`.
–protocol=name	The connection protocol (tcp, socket, pipe, memory) to use for connecting to the server. Useful when other connection parameters would cause a protocol to be used other than the one you want.
-q, –quick	If you are using this option with `CHECK TABLE`, it prevents the check from scanning the rows to check for wrong links. This is the fastest check. If you are using this option with `REPAIR TABLE`, it tries to repair only the index tree. This is the fastest repair method for a table.
-r, –repair	Can fix almost anything except unique keys that aren't unique.
–shared-memory-base-name	Shared-memory name to use for Windows connections using shared memory to a local server (started with the `--shared-memory` option). Case sensitive.
-s, –silent	Print only error messages.
–skip-database	Don't process the database (case sensitive) specified as argument.
-S name, –socket=name	For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.
–ssl	Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The `--ssl` option does not enable verifying the server certificate by default. In order to verify the server certificate, you must specify the `--ssl-verify-server-cert` option.
–ssl-ca=name	Defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option implies the `--ssl` option.
–ssl-capath=name	Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the `--ssl` option.
–ssl-cert=name	Defines a path to the X509 certificate file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.

Option	Description
–ssl-cipher=name	List of permitted ciphers or cipher suites to use for TLS. This option implies the `--ssl` option.
–ssl-crl=name	Defines a path to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL or Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
–ssl-crlpath=name	Defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
–ssl-key=name	Defines a path to a private key file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the `--ssl` option.
–ssl-verify-server-cert	Enables server certificate verification. This option is disabled by default.
–tables	Overrides the `--databases` or `-B` option such that all name arguments following the option are regarded as table names.
–use-frm	For repair operations on MyISAM tables, get table structure from .frm file, so the table can be repaired even if the .MYI header is corrupted.
-u, –user=name	User for login if not current user.
-v, –verbose	Print info about the various stages. You can give this option several times to get even more information. See mariadb-check and verbose, below.
-V, –version	Output version information and exit.
–write-binlog	Write `ANALYZE`, `OPTIMIZE`, and `REPAIR TABLE` statements to the binary log. Enabled by default; use `--skip-write-binlog` when commands should not be sent to replication slaves.

Option Files

In addition to reading options from the command line, `mariadb-check` can also read options from option files. If an unknown option is provided to `mariadb-check` in an option file, it is ignored.

The following options relate to how MariaDB command line tools handles option files. They must be given as the first argument on the command line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file `#`.

Option	Description
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

`mariadb-check` is linked with MariaDB Connector/C. However, MariaDB Connector/C does not yet handle the parsing of option files for this client. That is still performed by the server option file parsing code. See MDEV-19035 for more information.

Option Groups

`mariadb-check` reads options from the following option groups from option files:

Group	Description
[mysqlcheck]	Options read by mysqlcheck, which includes both MariaDB Server and MySQL Server.
[mariadb-check]	Options read by mariadb-check.
[client]	Options read by all MariaDB and MySQL client programs, which includes both MariaDB and MySQL clients. For example, mysqldump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

Notes

Default Values

To see the default values for the options and the arguments you get from configuration files, you can do:

```
./client/mariadb-check --print-defaults
./client/mariadb-check --help
```

mariadb-check and auto-repair

When running `mariadb-check` with `--auto-repair` (as done by mariadb-upgrade), `mariadb-check` will first check all tables and then in a separate phase repair those that failed the check.

mariadb-check and all-databases

`mariadb-check --all-databases` will ignore the internal log tables general_log and slow_log as these can't be checked, repaired or optimized.

mariadb-check and verbose

Using one `--verbose` option will give you more information about what mariadb-check is doing.

Using two `--verbose` options will also give you connection information.

If you use three `--verbose` options you will also get, on stdout, all ALTER, RENAME, and CHECK commands that mariadb-check executes.

Sequences

From MariaDB 12.0, `mariadb-check` supports sequences.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-convert-table-format

`mariadb-convert-table-format` converts the tables in a database to use a particular storage engine (MyISAM by default).

> **Note:**
>
> Previously, the client was called `mysql_convert_table_format`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

Usage

```
mariadb-convert-table-format [options] db_name
```

Description

`mariadb-convert-table-format` is written in Perl and requires that the DBI and DBD::mysql Perl modules be installed. Invoke `mariadb-convert-table-format` like this:

```
shell> mariadb-convert-table-format [options]db_name
```

The `db_name` argument indicates the database containing the tables to be converted.

Options

`mariadb-convert-table-format` supports the options described in the following list:

Option	Description
-?, –help	Display help and exit.
-e, –engine=ENGINE	Specify the storage engine that the tables should be converted to use. The default is MyISAM if this option is not given.
-f, –force	Continue even if errors occur.
-h, –host=host_name	Connect to the MariaDB server on the given host. Default localhost.
-p, –password=password	The password to use when connecting to the server. Note that the password value is not optional for this option, unlike for other client programs. Specifying the password on the command line is generally considered insecure.
-P, –port=port_num	The TCP/IP port number to use for the connection.
-S, –socket=path	For connections to localhost, the Unix socket file to use.
-u, –user=user_name	The MariaDB user name to use when connecting to the server.
-v, –verbose	Verbose mode. Print more information about what the program does.
-V, –version	Display version information and exit.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-fix-extensions

`mariadb-fix-extensions` converts the extensions for MyISAM (or ISAM) table files to their canonical forms.

> **Note:**
>
> Previously, the client was called `mysql_fix_extensions`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

It looks for files with extensions matching any lettercase variant of `.frm`, `.myd`, `.myi`, `.isd`, and `.ism` and renames them to have extensions of `.frm`, `.MYD`, `.MYI`, `.ISD`, and `.ISM`, respectively. This can be useful after transferring the files from a system with case-insensitive file names (such as Windows) to a system with case-sensitive file names.

Invoke `mariadb-fix-extensions` as follows, where `data_dir` is the path name to the MariaDB data directory:

```
mariadb-fix-extensions data_dir
```

This page is licensed: CC BY-SA / Gnu FDL

Testing Tools

See mariadb-slap.

Load-test MariaDB with mariadb-slap by emulating many concurrent clients running a set of queries repeatedly (formerly mysqlslap).

See mariadb-stress-test.

Stress-test a MariaDB server by running test scenarios repeatedly with mariadb-stress-test (a symlink to mysql-stress-test).

See mariadb-test.

MariaDB uses mariadb-test to test functionality. It is an all-in-one test framework doing unit, regression, and conformance testing

This page is licensed: CC BY-SA / Gnu FDL

mariadb-slap

`mariadb-slap` is a tool for load-testing MariaDB. It allows you to emulate multiple concurrent connections, and run a set of queries multiple times.

Prior to MariaDB 10.5, the client was called `mysqlslap`. It can still be accessed under this name, via a symlink in Linux, or an alternate binary in Windows.

It returns a benchmark including the following information:

- Average number of seconds to run all queries;
- Minimum number of seconds to run all queries;
- Maximum number of seconds to run all queries;
- Number of clients running queries;
- Average number of queries per client.

Using mariadb-slap

The command to use `mariadb-slap` and the general syntax is:

```
mariadb-slap [options]
```

Options

`mariadb-slap` supports the following options:

Option	Description
-a, –auto-generate-sql	Generate SQL statements automatically when they are not supplied in files or via command options.
–auto-generate-sql-add-autoincrement	Add an AUTO_INCREMENT column to auto-generated tables.
–auto-generate-sql-execute-number=num	Specify how many queries to generate automatically.
–auto-generate-sql-guid-primary	Add GUID based primary keys to auto-generated tables.
–auto-generate-sql-load-type=name	Specify the test load type. The allowable values are read (scan tables), write (insert into tables), key (read primary keys), update (update primary keys), or mixed (half inserts, half scanning selects). The default is mixed.
–auto-generate-sql-secondary-indexes=num	Number of secondary indexes to add to auto-generated tables. By default, none are added.
–auto-generate-sql-unique-query-number=num	Number of unique queries to generate for automatic tests. For example, if you run a key test that performs 1000 selects, you can use this option with a value of 1000 to run 1000 unique queries, or with a value of 50 to perform 50 different selects. The default is 10.
–auto-generate-sql-unique-write-number=num	Number of unique queries to generate for auto-generate-sql-write-number.
–auto-generate-sql-write-number=num	Number of row inserts to perform for each thread. The default is 100.

Option	Description
–commit=num	Number of statements to execute before committing. The default is 0.
-C, –compress	Use compression in server/client protocol if both support it.
-c name, –concurrency=name	Number of clients to simulate for query to run.
–create=name	File or string containing the statement to use for creating the table.
–create-schema=name	Schema to run tests in.
–csv[=name]	Generate comma-delimited output to named file or to standard output if no file is named.
-d, –debug[=options]	For debug builds, write a debugging log. A typical debug_options string is d:t:o,file_name. The default is d:t:o,/tmp/mariadb-slap.trace.
–debug-check	Check memory and open file usage at exit.
-T, –debug-info	Print some debug info at exit.
–default-auth=name	Default authentication client-side plugin to use.
–defaults-extra-file=name	Read this file after the global files are read. Must be given as the first option.
–defaults-file=name	Only read default options from the given file name Must be given as the first option.
-F name, –delimiter=name	Delimiter to use in SQL statements supplied in file or command line.
–detach=num	Detach (close and reopen) connections after the specified number of requests. The default is 0 (connections are not detached).
-e name, –engine=name	Comma separated list of storage engines to use for creating the table. The test is run for each engine. You can also specify an option for an engine after a #:#, for example memory:max_row=2300.
-?, –help	Display help and exit.
-h name, –host=name	Connect to the MariaDB server on the given host.
–init-command=name	SQL Command to execute when connecting to the MariaDB server. Will automatically be re-executed when reconnecting.

Option	Description
-i num, –iterations=num	Number of times to run the tests.
–no-defaults	Don’t read default options from any option file. Must be given as the first option.
–no-drop	Do not drop any schema created during the test after the test is complete.
-x name, –number-char-cols=name	Number of VARCHAR columns to create in table if specifying –auto-generate-sql.
-y name, –number-int-cols=name	Number of INT columns to create in table if specifying –auto-generate-sql.
–number-of-queries=num	Limit each client to approximately this number of queries. Query counting takes into account the statement delimiter. For example, if you invoke as follows, mariadb-slap –delimiter=“;” –number-of-queries=10 –query=“use test;insert into t values(null)”, the #; delimiter is recognized so that each instance of the query string counts as two queries. As a result, 5 rows (not 10) are inserted.
–only-print	Do not connect to the databases, but instead print out what would have been done.
-p[password], –password[=password]	Password to use when connecting to server. If password is not given it’s asked from the command line. Specifying a password on the command line should be considered insecure. You can use an option file to avoid giving the password on the command line.
-W, –pipe	On Windows, connect to the server via a named pipe. This option applies only if the server supports named-pipe connections.
–plugin-dir=name	Directory for client-side plugins.
-P num, –port=num	Port number to use for connection.
–post-query=name	Query to run or file containing query to execute after tests have completed. This execution is not counted for timing purposes.
–post-system=name	system() string to execute after tests have completed. This execution is not counted for timing purposes.
–pre-query=name	Query to run or file containing query to execute before running tests. This execution is not counted for timing purposes.
–pre-system=name	system() string to execute before running tests. This execution is not counted for timing purposes.
–print-defaults	Print the program argument list and exit. Must be given as the first option.
–protocol=name	The protocol to use for connection (tcp, socket, pipe, memory).

Option	Description
-q name, –query=name	Query to run or file containing query to run.
–shared-memory-base-name	Shared-memory name to use for Windows connections using shared memory to a local server (started with the –shared-memory option). Case-sensitive.
-s, –silent	Run program in silent mode - no output.
-S, –socket=name	For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.
–ssl	Enables TLS. TLS is also enabled even without setting this option when certain other TLS options are set. The –ssl option will not enable verifying the server certificate by default. In order to verify the server certificate, the user must specify the –ssl-verify-server-cert option.
–ssl-ca=name	Defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option implies the –ssl option.
–ssl-capath=name	Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Authorities (CAs) for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms. This option implies the –ssl option.
–ssl-cert=name	Defines a path to the X509 certificate file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the –ssl option.
–ssl-cipher=name	List of permitted ciphers or cipher suites to use for TLS. This option implies the –ssl option.
–ssl-crl=name	Defines a path to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL or Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
–ssl-crlpath=name	Defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command. See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.
–ssl-key=name	Defines a path to a private key file to use for TLS. This option requires that you use the absolute path, not a relative path. This option implies the –ssl option.
–ssl-verify-server-cert	Enables server certificate verification. This option is disabled by default.
-u, –user=name	User for login if not current user.
-v, –verbose	More verbose output; you can use this multiple times to get even more verbose output.

Option	Description
-V, –version	Output version information and exit.

Option Files

In addition to reading options from the command line, `mariadb-slap` can also read options from option files. If an unknown option is provided to `mariadb-slap` in an option file, then it is ignored.

The following options relate to how MariaDB command line tools handles option files. They must be given as the first argument on the command line:

Option	Description
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=#	Only read default options from the given file #.
–defaults-extra-file=#	Read this file after the global files are read.
–defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

`mariadb-slap` is linked with MariaDB Connector/C. However, MariaDB Connector/C does not yet handle the parsing of option files for this client. That is still performed by the server option file parsing code. See MDEV-19035 for more information.

Option Groups

`mariadb-slap` reads options from the following option groups from option files:

Group	Description
[mysqlslap]	Options read by mariadb-slap, which includes both MariaDB Server and MySQL Server.
[mariadb-slap]	Options read by mariadb-slap. Available starting with MariaDB 10.4.6.
[client]	Options read by all MariaDB and MySQL client programs, which includes both MariaDB and MySQL clients. For example, mysqldump.
[client-server]	Options read by all MariaDB client programs and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB client programs.

Examples

Create a table with data, and then query it with 40 simultaneous connections 100 times each.

```
mariadb-slap
 --delimiter=";"
 --create="CREATE TABLE t (a int);INSERT INTO t VALUES (5)"
 --query="SELECT * FROM t"
 --concurrency=40
 --iterations=100

Benchmark
    Average number of seconds to run all queries: 0.010 seconds
    Minimum number of seconds to run all queries: 0.009 seconds
    Maximum number of seconds to run all queries: 0.020 seconds
    Number of clients running queries: 40
    Average number of queries per client: 1
```

Using files to store the create and query SQL. Each file can contain multiple statements separated by the specified delimiter.

```
mariadb-slap
 --create=define.sql
 --query=query.sql
 --concurrency=10
 --iterations=20
 --delimiter=";"

Benchmark
    Average number of seconds to run all queries: 0.002 seconds
    Minimum number of seconds to run all queries: 0.002 seconds
    Maximum number of seconds to run all queries: 0.004 seconds
    Number of clients running queries: 10
    Average number of queries per client: 1
```

CC BY-SA / Gnu FDL

This page is licensed: CC BY-SA / Gnu FDL

mariadb-stress-test

`mariadb-stress-test` is a symlink to `mysql-stress-test`, the script for assisting with adding users or databases or changing passwords in MariaDB.

> **CURRENT**
>
> `mysql-stress-test` is the symlink, and `mariadb-stress-test` the binary name.
>
> **< 10.5.2**
>
> `mysql-stress-test` is the binary name.

mariadb-stress-test.pl is a Perl script that performs stress-testing of the MariaDB server. It requires a version of Perl that has been built with threads support.

Syntax

```
mariadb-stress-test.pl [options]
```

Options

Option	Description
–help	Display a help message and exit.
–abort-on-error=N	Causes the program to abort if an error with severity less than or equal to N was encountered. Set to 1 to abort on any error.
–check-tests-file	Periodically check the file that lists the tests to be run. If it has been modified, reread the file. This can be useful if you update the list of tests to be run during a stress test.
–cleanup	Force cleanup of the working directory.
–log-error-details	Log error details in the global error log file.
–loop-count=N	In sequential test mode, the number of loops to execute before exiting.
–mysqltest=path	The path name to the mysqltest program.

Option	Description
–server-database=db_name	The database to use for the tests. The default is test.
–server-host=host_name	he host name of the local host to use for making a TCP/IP connection to the local server. By default, the connection is made to localhost using a Unix socket file.
–server-logs-dir=path	This option is required. path is the directory where all client session logs is stored. Usually this is the shared directory that is associated with the server used for testing.
–server-password=password	The password to use when connecting to the server.
–server-port=port_num	The TCP/IP port number to use for connecting to the server. The default is 3306.
–server-socket=file_name	For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use. The default is /tmp/mysql.sock.
–server-user=user_name	The MariaDB user name to use when connecting to the server. The default is root.
–sleep-time=N	The delay in seconds between test executions.
–stress-basedir=path	This option is required and specified the path is the working directory for the test run. It is used as the temporary location for result tracking during testing.
–stress-datadir=path	The directory of data files to be used during testing. The default location is the data directory under the location given by the –stress-suite-basedir option.
–stress-init-file[=path]	file_name is the location of the file that contains the list of tests to be run once to initialize the database for the testing. If missing, the default file is stress_init.txt in the test suite directory.
–stress-mode=mode	This option indicates the test order in stress-test mode. The mode value is either random to select tests in random order or seq to run tests in each thread in the order specified in the test list file. The default mode is random.
–stress-suite-basedir=path	This option is required and specifies the directory that has the t and r subdirectories containing the test case and result files. This directory is also the default location of the stress-test.txt file that contains the list of tests. (A different location can be specified with the –stress-tests-file option.)
–stress-tests-file[=file_name]	Use this option to run the stress tests. file_name is the location of the file that contains the list of tests. If omitted, the default file is stress-test.txt in the stress suite directory. (See –stress-suite-basedir.)
–suite=suite_name	Run the named test suite. The default name is main (the regular test suite located in the mysql-test directory).
–test-count=N	The number of tests to execute before exiting.
–test-duration=N	The duration of stress testing in seconds.
–threads=N	The number of threads. The default is 1.
–verbose	Verbose mode. Print more information about what the program does

CC BY-SA / Gnu FDL

This page is licensed: CC BY-SA / Gnu FDL

mariadb-test

See mariadb-test Overview.

How mariadb-test runs a test file and compares its output against a result file to decide pass or fail.

See Installing MinIO for Usage With mariadb-test-run.

Install MinIO to provide S3-compatible storage for testing the S3 storage engine with mariadb-test-run.

See mariadb-test Auxiliary Files.

The auxiliary files the mariadb-test framework uses to control test runs.

See mariadb-test-run.pl Options.

Command-line options for the mariadb-test-run.pl test-runner script.

See New Features for mysqltest in MariaDB.

MariaDB enhancements to the mariadb-test (mysqltest) framework.

See Pausing mariadb-test-run.pl.

How to pause and resume mariadb-test-run.pl while a test run is in progress.

See The Debug Sync Facility.

The Debug Sync facility for placing synchronization points to coordinate tests.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-test Overview

Basics

The client program `mariadb-test` executes a *test file* and compares the produced output with the *result file*. If the files match, the test is passed; otherwise, the test has failed. This approach can be used to test any SQL statement, as well as other executables (with the `exec` command).

The complete process of testing is governed and monitored by the *mariadb-test-run.pl* driver script, or *mtr* for short (for convenience, `mtr` is created as a symbolic link to `mariadb-test-run.pl`). The `mtr` script is responsible for preparing the test environment, creating a list of all tests to run, running them, and producing the report at the end. It can run many tests in parallel, execute tests in an order which minimizes server restarts (as they are slow), run tests in a debugger or under `valgrind` or `strace`, and so on.

Test files are located in *suites*. A *suite* is a directory which contains test files, result files, and optional configuration files. The `mtr` script looks for suites in the `mariadb-test/suite` directory, and in the `mariadb-test` subdirectories of plugins and storage engine directories. For example, the following are all valid suite paths:

```
mariadb-test/suite/rpl
```

```
mariadb-test/suite/handler
```

```
storage/example/mariadb-test/demo
```

```
plugin/auth_pam/mariadb-test/pam
```

In almost all cases, the suite directory name is the suite name. A notable historical exception is the *main* suite, which is located directly in the `mariadb-test` directory.

Test files have a `.test` extension and can be placed directly in the suite directory (for example, `mariadb-test/suite/handler/interface.test`) or in the `t` subdirectory (e.g. `mariadb-test/suite/rpl/t/rpl_alter.test` or `mariadb-test/t/grant.test`). Similarly, result files have the `.result` extension and can be placed either in the suite directory or in the `r` subdirectory.

A test file can include other files (with the `source` command). These included files can have any name and may be placed anywhere, but customarily they have a `.inc` extension and are located either in the suite directory or in the `inc` or `include` subdirectories (for example, `mariadb-test/suite/handler/init.inc` or `mariadb-test/include/start_slave.inc`).

Other files which affect testing, while not being tests themselves, are:

- `disabled.def`
- `suite.opt`
- other `*.opt` files
- `my.cnf`
- other `*.cnf` files
- `combinations`
- other `*.combinations` files
- `suite.pm`
- `*.sh` files
- `*.require` files
- `*.rdiff` files
- `valgrind.supp`

See Auxiliary files for details on these.

Overlays

In addition to regular suite directories, `mtr` supports *overlays*. An *overlay* is a directory with the same name as an existing suite, but which is located in a storage engine or plugin directory. For example, `storage/myisam/mariadb-test/rpl` could be a *myisam* overlay of the *rpl* suite in `mariadb-test/suite/rpl`.
And `plugin/daemon_example/mariadb-test/demo` could be a *daemon_example* overlay of the *demo* suite in `storage/example/mariadb-test/demo`. As a special exception, an overlay of the main suite, should be called `main`, as in `storage/pbxt/mariadb-test/main`.

An overlay is like a second transparent layer in a graphics editor. It can obscure, extend, or modify the background image. Also, one may notice that an overlay is very close to a *UnionFS*, but implemented in perl inside `mtr`.

An overlay can replace almost any file in the overlaid suite, or add new files. For example, if some overlay of the main suite contains a `include/have_innodb.inc` file, then all tests that include it will see and use the overlaid version. Or, an overlay can create a `t/create.opt` file (even though the main suite does not have such a file), and `create.test` is executed with the specified additional options.

But adding an overlay never affects how the original suite is executed. That is, `mtr` always executes the original suite as if no overlay was present. Additionally, it executes a combined “union” of the overlay and the original suite. When doing that, `mtr` takes care to avoid re-executing tests that are not changed in the overlay. For example, creating `t/create.opt` in the overlay of the main suite will only cause `create.test` to be executed in the overlay. But creating `suite.opt` affects all tests — and it will cause all tests to be re-executed with the new options.

Combinations

In certain cases it makes sense to run a specific test or a group of tests several times with different server settings. This can be done using so-called *combinations*. Combinations are groups of settings that are used alternatively. A

combinations file defines these alternatives using `my.cnf` syntax, for example:

```
[row]
binlog-format=row
[stmt]
binlog-format=statement
[mix]
binlog-format=mixed
```

All tests where this combinations file applies is run three times: Once for the combination called “row”, and `--binlog-format=row` on the server command line, once for the “stmt” combination, and once for the “mix” combination.

More than one combinations file may be applicable to a given test file. In this case, `mtr` runs the test for all possible combinations of the given combinations. A test that uses replication (three combinations as above) and InnoDB (two combinations - innodb and xtradb), is run six times.

Sample Output

The typical `mtr` output looks like this:

```
==========================================================================
TEST                                WORKER RESULT   TIME (ms) or COMMENT
--------------------------------------------------------------------------
rpl.rpl_row_find_row_debug               [ skipped ]  Requires debug build
main-pbxt.connect                        [ skipped ]  No PBXT engine
main-pbxt.mysqlbinlog_row                [ disabled ]  test expects a non-transactional engine
rpl.rpl_savepoint 'mix,xtradb'           w2 [ pass ]    238
rpl.rpl_stm_innodb 'innodb_plugin,row'   w1 [ skipped ]  Neither MIXED nor STATEMENT binlog format
binlog.binlog_sf 'stmt'                  w2 [ pass ]      7
unit.dbug                                w2 [ pass ]      1
maria.small_blocksize                    w1 [ pass ]     23
sys_vars.autocommit_func3 'innodb_plugin' w1 [ pass ]      5
sys_vars.autocommit_func3 'xtradb'       w1 [ pass ]      6
main.ipv6                                w1 [ pass ]    131
...
```

Every test is printed as “suitename.testname”, and a suite name may include an overlay name (like in `main-pbxt`). After the test name, `mtr` prints combinations that were applied to this test, if any.

A similar syntax can be used on the `mtr` command line to specify what tests to run:

$ ./mtr innodb	**$ ./mtr main.innodb**
$ ./mtr innodb	search for innodb test in every suite from the default list, and run all that was found.
$ ./mtr main.innodb	run the innodb test from the main suite
$ ./mtr main-pbxt.innodb	run the innodb test from the pbxt overlay of the main suite
$ ./mtr main-.innodb	run the innodb test from the main suite and all its overlays.
$ ./mtr main.innodb,xtradb	run the innodb test from the main suite, only in the xtradb combination
$ ./mtr –suite=rpl	Run all test in the rpl suite (found in the suite/rpl directory)
$ ./mtr –suite=mroonga/storage	Run all test in the mroonga/wrapper suite (found in the storage/mroonga/mysql-test/mroonga/storage directory)

Plugin Support

The `mtr` driver has special support for MariaDB plugins.

First, on startup it copies or symlinks all dynamically-built plugins into `var/plugins`. This allows one to have many plugins loaded at the same time. For example, you can load Federated and InnoDB engines together. Also, `mtr`

creates environment variables for every plugin with the corresponding plugin name. For example, if the InnoDB engine was built, `$HA_INNODB_SO` is set to `ha_innodb.so` (or `ha_innodb.dll` on Windows). The test can safely use the corresponding environment variable on all platforms to refer to a plugin file; it always has the correct platform-dependent extension.

Second, when combining server command line options (which may come from many different sources) into one long list before starting `mariadbd`, mtr treats `--plugin-load` specially. Normal server semantics is to use the latest value of any particular option on the command line. If one starts the server with, for example, `--port=2000 --port=3000`, the server will use the last value for the port, that is 3000. To allow different `.opt` files to require different plugins, mtr goes through the assembled server command line, and joins all `--plugin-load` options into one. Additionally it removes all empty `--plugin-load` options. For example, suppose a test is affected by three `.opt` files which contain, respectively:

```
--plugin-load=$HA_INNODB_SO
```

```
--plugin-load=$AUTH_PAM_SO
```

```
--plugin-load=$HA_EXAMPLE_SO
```

Assuming the Example engine was not built (`$HA_EXAMPLE_SO` is empty), the server gets this:

```
--plugin-load=ha_innodb.so:auth_pam.so
```

Instead of this:

```
--plugin-load=ha_innodb.so --plugin-load=auth_pam.so --plugin-load=
```

Third, to allow plugin sources to be simply copied into the `plugin/` or `storage/` directories, and still not affect existing tests (even if new plugins are statically linked into the server), mtr automatically disables all optional plugins on server startup. A plugin is optional if it can be disabled with the corresponding `--skip-XXX` server command line option. Mandatory plugins, like MyISAM or MEMORY, do not have `--skip-XXX` options (for instance, there is no `--skip-myisam` option). This `mtr` behavior means that no plugin, statically or dynamically built, has any effect on the server unless it was explicitly enabled. A convenient way to enable a given plugin *XXX* for specific tests is to create a `have_XXX.opt` file which contains the necessary command line options, and a `have_XXX.inc` file which checks whether a plugin was loaded. Then any test that needs this plugin can source the `have_XXX.inc` file and have the plugin loaded automatically.

mtr Communication Procedure

`mtr` is first creating the server socket (`master`). After that, `workers` are created using `fork()`.

For each worker, the `run_worker()` function is called, which is executing the following:

- Creates a new socket to connect to `server_port` obtained from the `master`.
- Initiate communication with the `master` using `START` command.
- `master` sends first test from list of tests supplied by the user and immediately sends command `TESTCASE` to the `worker`.
- `worker` gets command `TESTCASE` and processes test case, by calling `run_testcase()` function which starts(/restarts if needed) the server and sends `TESTRESULT` (in case of restart `WARNINGS` command is issued to the `master` in case some warnings/error logs are found).
- `master` accepts `TESTRESULT` command and run `mtr_report_test()` function which check does the test fail and also generates the new command `TESTCASE` if some new test case exist.

- If there is no other test case `master` sends `BYE` command which gets accepted by the `worker` which is properly closing the connection.

Options

From MariaDB Server 11.8.3, you can start `mtr` with the `--enable_serveroutput` option to enable `DBMS_OUTPUT` messages. Disable displaying those messages with `--disable_serveroutput`. See this page for details.

A new package procedure was added, `DBMS_OUTPUT.GET_LINES_RESULT()`. It reuses the SQL code fetching lines from the `DBMS_OUTPUT` buffer. All buffer lines are returned in a single result set. This procedure is MariaDB-specific; it does not exist in Oracle. This workaround is needed in MariaDB, instead of using the `GET_LINES()` function, because MariaDB does not support fetching arrays in the client-server protocol.

This page is licensed: CC BY-SA / Gnu FDL

Installing MinIO for Usage With mariadb-test-run

When testing the S3 storage engine with the s3 test suite, mariadb-test-run needs access to Amazon S3 compatible storage.

The easiest way to achieve this is to install MinIO, an open source S3 compatible storage.

Here is a shell script that you can use to install MinIO with the right credentials for mariadb-test-run.
This should work on most Linux systems as the binaries are statically linked.
You can alternatively download MinIO binaries directly from here.

```
# Where to install the MinIO binaries and where to store the data
install=/my/local/minio
data=/tmp/shared

# Get the MinIO binaries. You can skip this test if you already have MinIO installed.
mkdir -p $install
wget https://dl.min.io/server/minio/release/linux-amd64/minio -O $install/minio
wget https://dl.min.io/client/mc/release/linux-amd64/mc -O $install/mc
chmod a+x $install/minio $install/mc

# Setup MinIO for usage with mariadb-test-run
MINIO_ACCESS_KEY=minio MINIO_SECRET_KEY=minioadmin $install/minio server $data 2>&1 &
$install/mc config host add local http://127.0.0.1:9000 minio minioadmin
$install/mc mb --ignore-existing local/storage-engine
```

Now you can run the S3 test suite:

```
cd "mysql-source-dir"/mariadb-test
./mariadb-test-run --suite=s3
```

If there is an issue while running the test suite, you can see the files created by MinIO with:

```
$install/mc ls -r local/storage-engine
```

or

```
ls $data/storage-engine
```

If you want to use MinIO with different credentials or you want to run the test against another S3 storage you ave to update the update the following files:

```
mariadb-test/suite/s3/my.cnf
mariadb-test/suite/s3/slave.cnf
```

This page is licensed: CC BY-SA / Gnu FDL

mariadb-test Auxiliary Files

The mariadb-test framework utilizes many other files that affect the testing process, in addition to test and result files.

`disabled.def` file

This file can be used to disable certain tests temporarily. For example, if one test fails and you are working on that, you may want to push the changeset that disables the test into the test suite so that other tests won't be disturbed by this failure.

The file contains test names and a comment (that should explain why the test was disabled), separated by a colon. Lines that start with a hash sign
(`#`) are ignored. A typical `disabled.def` may look like this (note that a hash sign in the middle of a line does not start a comment):

```
# List of disabled tests
# test name : comment
rpl_redirect : Fails due to bug#49978
events_time_zone  : need to fix the timing
```

During testing, mtr will print disabled tests like this:

```
...
rpl.rpl_redirect              [ disabled ]  Fails due to bug#49978
rpl.events_time_zone          [ disabled ]  need to fix the timing
...
```

This file should be located in the suite directory.

`suite.opt` file

This file lists server options that is added to the `mariadbd` command line for every test of this suite. It can refer to environment variables with the `$NAME` syntax. Shell meta-characters should be quoted. For example

```
--plugin-load=$AUTH_PAM_SO
--max-connections=40 --net_read_timeout=5
"--replicate-rewrite-db=test->rewrite"
```

Note that options may be put either on one line or on separate lines. It is a good idea to start an option name with the `--loose-` prefix if the server may or may not recognize the option depending on the configuration.
An unknown option in the `.opt` file will stop the server from starting, and the test is aborted.

This file should be located in the suite directory.

other `*.opt` files

For every test or include file `somefile.test` or `somefile.inc`, mtr will look for `somefile.opt`, `somefile-master.opt` and `somefile-slave.opt`.
These files have exactly the same syntax as the `suite.opt` above. Options from these files will also be added to the server command line (all servers started for this test, only master, or only slave respectively) for all affected tests, for example, for all tests that include `somefile.inc` directly or indirectly.

A typical usage example is `include/have_blackhole.inc` and `include/have_blackhole.opt`. The latter contains the necessary command line options to load the Blackhole storage engine, while the former verifies that the engine

was really loaded. Any test that needs the Blackhole engine needs only to start from `source include/have_blackhole.inc;` and the engine will be automatically loaded for the test.

`my.cnf` file

This is not the `my.cnf` file that tests from this suite will use, but rather a *template* of it. It is converted later to an actual `my.cnf`. If a suite contains no `my.cnf` template, a default template,
— `include/default_my.cnf`
— is used. Or `suite/rpl/my.cnf` if the test includes `master-slave.inc` (it's one of the few bits of the old MySQL `mysql-test-run` magic that we have not removed yet). Typically a suite template will not contain a complete server configuration, but rather start from

```
!include include/default_my.cnf
```

and then add the necessary modifications.

The syntax of `my.cnf` template is the same of a normal `my.cnf` file, with a few extensions and assumptions. They are:

- For any group with the name `[mysqld.N]`, where N is a number, mtr will start one `mysqld` process. Usually one needs to have only `[mysqld.1]` group, and `[mysqld.2]` group for replication tests.
- There can be groups with non-standard names (`[foo]`, `[bar]`, whatever), not used by `mysqld`. The `suite.pm` files (see below) may use them somehow.
- Values can refer to each other using the syntax `@groupname.optionname`
  — these references be expanded as needed. For example

```
[mysqld.2]
master-port= @mysqld.1.port
```

- it sets the value of the `master-port` in the `[mysqld.2]` group to the value of `port` in the `[mysqld.1]` group.
- An option name may start with a hash sign `#`. In the resulting `my.cnf` it will look like a comment, but it still can be referred to. For example:

```
[example]
#location = localhost:@mysqld.1.port
bar = server:@example.#location/data
```

- There is the `[ENV]` group. It sets values for the environment variables.
  For example

```
[ENV]
MASTER_MYPORT = @mysqld.1.port
```

- Also, one can refer to values of environment variables via this group:

```
[mysqld.1]
user = @ENV.LOGNAME
```

- There is the `[OPT]` group. It allows to invoke functions and generate values. Currently it contains only one option
  — `@OPT.port`. Every time this option is referred to in some other group in the `my.cnf` template, a new unique port number is generated. It will not match any other port number used by this test run.
  For example

```
[ENV]
SPHINXSEARCH_PORT = @OPT.port
```

This file should be located in the suite directory.

other `*.cnf` files

For every test file `somefile.test` (but for not included files) mtr will look for `somefile.cnf` file. If such a file exists, it is used as a template instead of suite `my.cnf` or a default `include/default_my.cnf` templates.

`combinations` file

The `combinations` file defines few sets of alternative configurations, and every test in this suite is run many times - once for every configuration.
This can be used, for example, to run all replication tests in the *rpl* suite for all three binlog format modes (row, statement, and mixed). A corresponding `combinations` file would look as following:

```
[row]
binlog-format=row

[stmt]
binlog-format=statement

[mix]
binlog-format=mixed
```

It uses `my.cnf` file syntax, with groups (where group names define combination names) and options. But, despite the similarity, it is not a `my.cnf` template, and it cannot use the templating extensions. Instead, options from the `combinations` file are added to the server command line. In this regard, combination file is closer to `suite.opt` file. And just like it, combination file can use environment variables using the `$NAME` syntax.

Not all tests will necessarily run for all combinations. A particular test may require to be run only in one specific combination. For example, in replication, if a test can only be run with the row binlog format, it will have `--binlog-format=row` in one of the `.opt` files. In this case, mtr will notice that server command line already has an option that matches one of the combinations, and will skip all other combinations for this particular test.

The `combinations` file should be located in the suite directory.

other `*.combinations` files

Just like with the `*.opt` files, mtr will use `somefile.combinations` file for any `somefile.test` and `somefile.inc` that is used in testing. These files have exactly the same format as a suite `combinations` file.

This can cause many combination files affecting one test file (if a test includes two `.inc` files, and both of them have corresponding `.combinations` files). In this case, mtr will run the test for all combinations of combinations from both files. In MariaDB 5.5, for example, `rpl_init.inc` adds combinations for row/statement/mixed, and `have_innodb.inc` adds combinations for innodb/xtradb. Thus any replication test that uses innodb is run six times.

`suite.pm` file

This (optional) file is a perl module. It must declare a package that inherits from `My::Suite`.

This file must normally end with `bless {}` — that is it must return an object of that class. It can also return a string — in this case all tests in the suite is skipped, with this string being printed as a reason (for example “PBXT engine was not compiled”).

A suite class can define the following methods:

- `config_files()`
- `is_default()`
- `list_cases()`
- `servers()`
- `skip_combinations()`
- `start_test()`

A `config_files()` method returns a list of additional config files (besides `my.cnf`), that this suite needs to be created. For every file it specifies a function that will create it, when given a `My::Config` object. For example:

```
sub config_files {(
    'config.ini' => \&write_ini,
    'new.conf'   => \&do_new
)}
```

A `servers()` method returns a list of processes that needs to be started for this suite. A process is specified as a [regex, hash] pair. The regular expression must match a section in the `my.cnf` template (for example, `qr/mysqld\./` corresponds to all `mysqld` processes), the hash contains these options:

SORT	a number. Processes are started in the order of increasing SORT values (and stopped in the reverse order). mysqld has number 300.
START	a function to start a process. It takes two arguments, My::Config::Group and My::Test. If START is undefined a process will not be started.
WAIT	a function to wait for the process to be started. It takes My::Config::Group as an argument. Internally mtr first invokes START for all processes, then WAIT for all started processes.

```
sub servers {(
    qr/^foo$/ => { SORT => 200,  # start foo before mysqld
                   START => \&start_foo,
                   WAIT => \&wait_foo }
)}
```

See the *sphinx* suite for a working example.

A `list_cases()` method returns a complete list of tests for this suite. By default it is the list of files that have `.test` extension, but without the extension. This list is filtered by mtr, subject to different mtr options (`--big-test`, `--start-from`, etc), the suite object does not have to do it.

A `start_test()` method starts one test process, by default it is `mariadb-test`. See the *unit* suite for a working example of `list_cases()` and `start_test()` methods.

A `skip_combinations()` method returns a hash that maps file names (where combinations are defined) to a list of combinations that should be skipped. As a special case, it can disable a complete file by using a string instead of a hash. For example

```
sub skip_combinations {(
    'combinations' => [ 'mix', 'rpl' ],
    'inc/many.combinations' => [ 'a', 'bb', 'c' ],
    'windows.inc' => "Not on windows",
)}
```

The last line will cause all tests of this suite that include `windows.inc` to be skipped with the reason being “Not on windows”.

An `is_default()` method returns 1 if this particular suite should be run by default, when the `mariadb-test-run.pl` script is run without explicitly specified test suites or test cases.

`*.sh` files

For every test file `sometest.test` mtr looks for `sometest-master.sh` and `sometest-slave.sh`. If either of these files is found, it is run before the test itself.

`*.require` files

These files are obsolete. Do not use them anymore. If you need to skip a test use the `skip` command instead.

`*.rdiff` files

These files also define what the test result should be. But unlike `*.result` files, they contain a patch that should be applied to one result file to create a new result file. This is very useful when a result of some test in one combination differs slightly from the result of the same test, but in another combination. Or when a result of a test in an overlay differs from the test result in the overlaid suite.

It is quite difficult to edit `.rdiff` files to update them after the test file has changed. But luckily, it is never needed. When a test fails, mtr creates a `.reject` file. Having it, one can create `.rdiff` file as easy as (for example)

```
diff -u main/foo.result main/foo.reject > main/foo,comb.rdiff
or
diff -u main/foo.result main/foo,comb.reject > main/foo,comb.rdiff
```

Some example:

```
diff -u main/innodb_ext_key.result main/innodb_ext_key,off.reject > main/innodb_ext_key,off.rdiff

diff -u suite/sys_vars/r/sysvars_server_notembedded.result
suite/sys_vars/r/sysvars_server_notembedded,32bit.reject >
suite/sys_vars/r/sysvars_server_notembedded,32bit.rdiff
```

Note: This will also add a timestamp in the .rdiff file, so if you are submitting a patch you could remove it manually. If the same .rdiff file is used for multiple combinations, then it would be good to omit in the header that would identify the combination, to allow git to pack the repository better. Example:

```
--- testname.result
+++ testname.reject
```

Because a combination can be part of the `.result` or `.rdiff` file name, mtr has to look in many different places for a test result. For example, consider a test `foo.test` in the combination pair `aa,bb`, that is run in the overlay *rty* of the suite *qwe*, in other words, for the test that mtr prints as

```
qwe-rty.foo 'aa,bb'                    [ pass ]
```

For this test a result can be in

- either `.rdiff` or `.result` file
- either in the overlay "`rty/`" or in the overlaid suite "`qwe/`"
- with or without combinations in the file name ("`,a`", "`,b`", "`,a,b`", or nothing)

which means any of the following 15 file names can be used:

1. `rty/r/foo,aa,bb.result`
2. `rty/r/foo,aa,bb.rdiff`
3. `qwe/r/foo,aa,bb.result`
4. `qwe/r/foo,aa,bb.rdiff`

5. `rty/r/foo,aa.result`
6. `rty/r/foo,aa.rdiff`
7. `qwe/r/foo,aa.result`
8. `qwe/r/foo,aa.rdiff`
9. `rty/r/foo,bb.result`
10. `rty/r/foo,bb.rdiff`
11. `qwe/r/foo,bb.result`
12. `qwe/r/foo,bb.rdiff`
13. `rty/r/foo.result`
14. `rty/r/foo.rdiff`
15. `qwe/r/foo.result`

They are listed, precisely, in the order of preference, and mtr will walk that list from top to bottom and the first file that is found is used.

If this found file is a `.rdiff`, mtr continues walking down the list until the first `.result` file is found. A `.rdiff` is applied to that `.result`.

`valgrind.supp` file

This file defines valgrind suppressions, and it is used when mtr is started with a `--valgrind` option.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-test-run.pl Options

Syntax

```
./mariadb-test-run.pl [ OPTIONS ] [ TESTCASE ]
```

Where the test case can be specified as: `testcase[.test]` Runs the test case named 'testcase' from all suites.

```
path-to-testcase
[suite.]testcase[,combination]
```

Examples

`aliasmain.alias` 'main' is the name of the suite for the 't' directory.

```
rpl.rpl_invoked_features,mix,xtradb_plugin
suite/rpl/t/rpl.rpl_invoked_features
```

Options

Options to Control What Engine/Variation to Run

Option	Description
–embedded-server	Use the embedded server, i.e., no mysqld daemons.

Option	Description
–ps-protocol	Use the binary protocol between client and server.
–cursor-protocol	Use the cursor protocol between client and server (implies –ps-protocol).
–view-protocol	Create a view to execute all non- updating queries.
–sp-protocol	Create a stored procedure to execute all queries.
–compress	Use the compressed protocol between client and server if both support it.
–ssl	If mariadb-test-run.pl is started with the –ssl option, it sets up a secure connection for all test cases. In this case, if mysqld does not support TLS, mariadb-test-run.pl exits with an error message: Couldn't find support for SSL.
–skip-ssl	Don't start the server with support for TLS connections.
–vs-config	Visual Studio configuration used to create executables (default: MTR_VS_CONFIG environment variable).
–parallel=num	How many parallel tests should be run. Default is 1, use –parallel=auto for auto-setting of num.
–defaults-file=	Use a fixed config template for all tests.
–defaults_extra_file=	Extra config template to add to all generated configs.
–combination=	Extra options to pass to mysqld. The value should consist of one or more comma-separated mysqld options. This option is similar to –mysqld but should be given two or more times. mariadb-test-run.pl executes multiple test runs, using the options for each instance of –combination in successive runs. If –combination is given only once, it has no effect. For test runs specific to a given test suite, an alternative to the use of –combination is to create a combinations file in the suite directory. The file should contain a section of options for each test run.
–dry-run	Don't run any tests; print the list of tests that were selected for execution.

Options to Control Directories to Use

Option	Description
–tmpdir=DIR	The directory where temporary files are stored (default: ./var/tmp). The environment variable MYSQL_TMP_DIR is set to the path for this directory, whether it has the default value or has been set explicitly. This may be referred to in tests.
–vardir=DIR	The directory where files generated from the test run is stored (default: ./var). Specifying a ramdisk or tmpfs will speed up tests. The environment variable MYSQLTEST_VARDIR is set to the path for this directory, whether it has the default value or has been set explicitly. This may be referred to in tests.

Option	Description
–mem	Run testsuite in “memory” using tmpfs or ramdisk. This can decrease test times significantly, in particular if you would otherwise be running over a remote file system. Attempts to find a suitable location using a built-in list of standard locations for tmpfs (/dev/shm). The option can also be set using the environment variable MTR_MEM=[DIR]. If DIR is given, it is added to the beginning of the list of locations to search, so it takes precedence over any built-in locations. Once you have run tests with –mem within a mariadb-testdirectory, a soflink var will have been set up to the temporary directory, and this is reused the next time, until the soflink is deleted. Thus, you do not have to repeat the –mem option next time.
–client-bindir=PATH	Path to the directory where client binaries are located.
–client-libdir=PATH	Path to the directory where client libraries are located.

Options to Control What Test Suites or Cases to Run

Option	Description
–force	Normally, mariadb-test-run.pl exits if a test case fails. –force causes execution to continue regardless of test case failure.
–with-ndbcluster-only	Run only tests that include “ndb” in the filename.
–skip-ndb[cluster]	Skip all tests that need a cluster. Default.
–do-test=PREFIX or REGEX	Run test cases with names prefixed with PREFIX or that fulfil the REGEX. For example, –do-test=testa matches tests that begin with testa, –do-test=main.testa matches tests in the main test suite that begin with testa, and –do-test=main.*testa matches test names that contain main followed by testa with anything in between. In the latter case, the pattern match is not anchored to the beginning of the test name, so it also matches names such as xmainytestz.
–skip-test=PREFIX or REGEX	Skip test cases with names prefixed with PREFIX or that fulfil the REGEX. See -do-test for examples.
–start-from=PREFIX	Sorts the list of names of the test cases to be run, and then starts with the test prefixed with PREFIX, where the prefix may be suite.testname or just testname.
–suite[s]=NAME1,...,NAMEN	Comma separated list of suite names to run. The default, as of MariaDB 10.4.5, is:“main-, archive-, binlog-, binlog_encryption-, csv-, compat/oracle-, encryption-, federated-, funcs_1-, funcs_2-, gcol-, handler-, heap-, innodb-, innodb_fts-, innodb_gis-, innodb_zip-, json-, maria-, mariadb-backup-, multi_source-, optimizer_unfixed_bugs-, parts-, perfschema-, plugins-, roles-, rpl-, sys_vars-, sql_sequence-, unit-, vcol-, versioning-,period-”.
–skip-rpl	Skip the replication test cases.
–big-test	Allow tests marked as “big” to run. Tests can be thus marked by including the line –source include/big_test.inc, and they will only be run if this option is given, or if the environment variable BIG_TEST is set to 1. Repeat this option twice to run only “big” tests. This is typically used for tests that take a very long time to run, or that use many resources, so that they are not suitable for running as part of a normal test suite run.
–staging-run	Run a limited number of tests (no slow tests). Used for running staging trees with valgrind.
–enable-disabled	Ignore any disabled.def file, and also run tests marked as disabled. Success or failure of those tests is reported the same way as other tests.

Option	Description
–print-testcases	Don't run the tests, but print details about all the selected tests, in the order they would be run.
–skip-test-list=FILE	Skip the tests listed in FILE. Each line in the file is an entry and should be formatted as: <TESTNAME>:<COMMENT>.

Options That Specify Ports

Option	Description
–[mtr-]port-base=num	Base for port numbers. Ports from this number to number+9 are reserved. Should be divisible by 10; if not it is rounded down. May be set with the environment variable MTR_PORT_BASE. If this value is set and is not "auto", it overrides build-thread.
–[mtr-]build-thread=num	Specify a unique number to calculate port number(s) from. Can be set in the environment variable MTR_BUILD_THREAD. Set MTR_BUILD_THREAD="auto" to automatically acquire a build thread id that is unique to the current host. The more logical –port-base is supported as an alternative.

Options For Test Case Authoring

Option	Description
–record TESTNAME	(Re)generate the result file for TESTNAME.
–check-testcases	Check testcases for side-effects. This is done by checking the system state before and after each test case; if there is any difference, a warning to that effect is written, but the test case will not be marked as failed because of it. This check is enabled by default. Use –nocheck-testcases to disable.
mark-progress	Log line number and elapsed time to <testname>.progress

Options That Pass On Options

Option	Description
–mysqld=ARGS	Specify additional arguments to "mysqld"
–mysqltest=ARGS	Specify additional arguments to "mariadb-test". Use additional –mysqld-env options to set more than one variable.

Options to Run Test On Running Server

Option	Description
extern option=value	Use an already running server. The option/value pair is what is needed by the mariadb client to connect to the server. Each –extern option can only take one option/value pair as an argument, so you need to repeat –extern for each pair needed. Example: ./mariadb-test-run.pl –extern socket=var/tmp/mysqld.1.sock alias. Note: If a test case has an .opt file that requires the server to be restarted with specific options, the file will not be used. The test case likely will fail as a result.

Options For Debugging the Product

In `mariadb-test-run.pl`, there is a concept of a "debugger". A "debugger" is a tool that `mariadb-test-run.pl` will execute instead of `mariadbd`. This tool will then start `mariadbd` and can control its execution as it wants. The following "debuggers" are supported:

name	Description
gdb	GNU debugger
ddd	GUI frontend for gdb

name	Description
dbx	Dbx_(debugger)
devenv	Visual Studio debugger
windbg	WinDbg
lldb	Debugger from LLVM project
valgrind	Detects memory management problems and more
strace	syscall tracer
rr	“record and replay” — record the program execution and then replay it forward, backward, or in any other direction

Every “debugger” from the list above supports the following set of options (replace XXX below with a debugger name)

Option	Description
–XXX	Start mariadbd process under a debugger
–client-XXX	Start mariadb-test process under a debugger
–boot-XXX	Before running tests, mariadb-test-run executes mariadbd to bootstrap, prepare the datadir. This option causes this bootstrapping mariadbd process to be run under a debugger.
–manual-XXX	Don’t start anything; instead, print the command that the user needs to run to start mariadbd under a debugger. Then wait.

Every option from the above accepts an optional argument. It can be used to specify additional command-line options to pass to the tool. Or additional commands that the tool will run on startup. Or both. Commands are separated from each other and from options with a semicolon. For example:

```
./mtr 1st --strace
./mtr 1st --client-rr=--chaos
./mtr 1st --manual-gdb='b mysql_parse;r'
./mtr 1st --boot-gdb='--quiet --tui;b mysql_parse;r'
```

Misc Debugging Related Options

Option	Description
–debug	Dump trace output for all servers and client programs.
–debug-common	Same as –debug, but sets the ‘d’ debug flags to “query,info,error,enter,exit”
–debug-server	Use the debug version of the server, but without turning on tracing.
–max-save-core	Limit the number of core files saved (to avoid filling up disks for a heavily crashing server). Defaults to 5, set to 0 for no limit. Set its default with MTR_MAX_SAVE_CORE.
–max-save-datadir	Limit the number of datadir saved (to avoid filling up disks for a heavily crashing server). Defaults to 20, set to 0 for no limit. Set its default with MTR_MAX_SAVE_DATDIR.
–max-test-fail	Limit the number of test failures before aborting the current test run. Defaults to 10, set to 0 for no limit. Set its default with MTR_MAX_TEST_FAIL.

Misc Options

Option	Description
–user=USER	User for connecting to mysqld (default: root)

Option	Description
–comment=STR	Write STR to the output within lines filled with #, as a form of banner.
–timer	Show test case execution time. Use no-timer to disable.
–verbose	More verbose output (use multiple times for even more)
–verbose-restart	Write when and why servers are restarted between test cases.
–start	Only initialize and start the servers, using the startup settings for the first specified test case. Example: ./mariadb-test-run.pl –start alias & start-dirty. Only start the servers (without initialization) for the first specified test case
–start-and-exit	Same as –start, but mariadb-test-run terminates and leaves just the server running.
–start-dirty	This is similar to –start, but will skip the database initialization phase and assume that database files are already available. Usually, this means you must have run another test first.
–user-args	In combination with start* and no test name, drops arguments to mysqld except those specified with –mysqld (if any).
–wait-all	If –start or –start-dirty option is used, wait for all servers to exit before finishing the process. Otherwise, it will terminate if one (of several) servers is restarted.
–fast	Do not perform a controlled shutdown when servers need to be restarted or at the end of the test run. This is equivalent to using –shutdown-timeout=0.
–force-restart	Always restart servers between tests.
–parallel=N	Run tests in N parallel threads (default 1). Use parallel=auto for auto-setting of N.
–repeat=N	Run each test N number of times.
–retry=N	If a test fails, it is retried up to a maximum of N runs (default 1). Retries are also limited by the maximum number of failures before stopping, set with the –retry-failure option. This option has no effect unless –force is also used; without it, test execution will terminate after the first failure. The –retry and –retry-failure options do not affect how many times a test repeated with –repeat may fail in total, as each repetition is considered a new test case, which may in turn be retried if it fails.
–retry-failure=N	When using the –retry option to retry failed tests, stop when N failures have occurred (default 2). Setting it to 0 or 1 effectively turns off retries.
–reorder	Reorder tests to get fewer server restarts. This is the default behavior. There is no guarantee that a particular set of tests will always end up in the same order. Use -no-reorder to disable.
–help	Display help text.
–testcase-timeout=MINUTES	Max test case run time in minutes (default 15).
–suite-timeout=MINUTES	Max test suite run time in minutes (default 360).
–shutdown-timeout=SECONDS	Max number of seconds to wait for server shutdown before killing servers (default 10).
–warnings	Scan the log files for warnings and report any suspicious ones; if any are found, the test is marked as failed. Use –nowarnings to turn off.

Option	Description
–stop-file=file	If this file is detected, mariadb-test will not start new tests until the file is removed (also the MTR_STOP_FILE environment variable).
–stop-keep-alive=sec	Works with –stop-file, print messages every second when mariadb-test is waiting to remove the file (for buildbot) (also MTR_STOP_KEEP_ALIVE environment variable).
–sleep=SECONDS	Passed to mariadb-test; it is used as a fixed sleep time.
–debug-sync-timeout=NUM	Set default timeout for WAIT_FOR debug sync actions. Disable facility with NUM=0.
–gcov	Collect coverage information after the test. The result is a dgcov file per source and header file, and a last_changes.dgcov file in the vardir with the coverage for the uncommitted changes, if any (or the last commit).
–gprof	Collect profiling information using the gprof profiling tool.
–experimental=	Specify a file that contains a list of test cases that should be displayed with the [exp-fail] code rather than [fail] if they fail. For an example of a file that might be specified via this option, see mariadb-test/collections/default.experimental.
–report-features	First, run a “test” that reports MariaDB features, displaying the output of SHOW ENGINES and SHOW VARIABLES. This can be used to verify that binaries are built with all required features.
–timestamp	Print the timestamp before each test report line, showing when the test ended.
–timediff	Used with –timestamp, also print the time passed since the previous test started.
–max-connections=N	Maximum number of simultaneous server connections that may be used per test. Default is 128. Minimum is 8, maximum is 5120. Corresponds to the same option for mariadb-test.
–default-myisam	Set default storage engine to MyISAM for non-innodb tests. This is needed after switching the default storage engine to InnoDB.
–report-times	Report how much time has been spent on different phases of test execution.
–stress=ARGS	Run stress test, providing options to mysql-stress-test.pl. Options are separated by a comma.
xml-report=	Output jUnit xml file of the results. From MariaDB 10.1.45, MariaDB 10.2.32, MariaDB 10.3.23, MariaDB 10.4.13, MariaDB 10.5.3
tail-lines=N	Number of lines of the result to include in a failure report. From MariaDB 10.3.4.

This page is licensed: GPLv2

New Features for mysqltest in MariaDB

> **Note:**
>
> Note that not all MariaDB-enhancements are listed on this page. See mariadb-test and mariadb-test-embedded for a full set of options.

Startup Option –connect-timeout

```
--connect-timeout=N
```

This can be used to set the MYSQL_OPT_CONNECT_TIMEOUT parameter of mysql_options, to change the number of seconds before an unsuccessful connection attempt times out.

Test Commands for Handling Warnings During Prepare Statements

- `enable_prepare_warnings;`
- `disable_prepare_warnings;`

Normally, when running with the prepared statement protocol with warnings enabled and executing a statement that returns a result set (like SELECT), warnings that occur during the execute phase are shown, but warnings that occur during the prepare phase are ''not'' shown. The reason for this is that some warnings are returned both during prepare and execute; if both copies of warnings were shown, then test cases would show different number of warnings between prepared statement execution and normal execution (where there is no prepare phase).

The `enable_prepare_warnings` command changes this so that warnings from both the prepare and execute phase are shown, regardless of whether the statement produces a result set in the execute phase.
The `disable_prepare_warnings` command reverts to the default behaviour.

These commands only have effect when running with the prepared statement protocol (–ps-protocol) *and* with warnings enabled (enable_warnings). Furthermore, they only have effects for statements that return a result set (as for statements without result sets, warnings from are always shown when warnings are enabled).

The `replace_regex` command supports paired delimiters (like in perl, etc). If the first non-space character in the `replace_regex` argument is one of `(`, `[`, `{`, `<`, then the pattern should end with `)`, `]`, `}`, `>` accordingly. The replacement string can use its own pair of delimiters, not necessarily the same as the pattern. If the first non-space character in the `replace_regex` argument is not one of the above, then it should also separate the pattern and the replacement string and it should end the replacement string. Backslash can be used to escape the current terminating character as usual. The examples below demonstrate valid usage of `replace_regex`:

```
--replace_regex (/some/path)</another/path>
--replace_regex !/foo/bar!foobar!
--replace_regex {pat\}tern}/replace\/ment/i
```

Dumping “exec” output on errors only

Sometimes it is only interesting to see the output of a utility to stdout/stderr, if utility failed. In case of success, the output might be unpredictable, and contain timestamps, startup messages etc. mariadb-backup can be a good example of such utility.

mysqltest in MariaDB can helps in this situation. In the example below, the output of $XTRABACKUP is suppressed, however if `exec` fails, stdout and stderr both is dumped , to aid diagnostics:

```
--disable_result_log
exec $XTRABACKUP  --param1 --param2  # Note, do not use output redirection here
--enable_result_log
```

This page is licensed: CC BY-SA / Gnu FDL

Pausing mariadb-test-run.pl

Sometimes you need to work when your computer is busy running mariadb-test-run.pl. The mariadb-test-run.pl script allows you to stop it temporarily so you can use your computer and then restart the tests when you’re ready.

There are two ways to enable this:

1. Command-line: The `--stop-file` and `--stop-keep-alive` options.
2. Environment Variables: If you are calling mariadb-test-run.pl indirectly (i.e from a script or program such as buildbot) you can set `MTR_STOP_FILE` and `MTR_STOP_KEEP_ALIVE`.

Keep Alive

If you plan on using this feature with other programs, such as buildbot, you should set the `MTR_STOP_KEEP_ALIVE` environment variable or the `--stop-keep-alive` command line option with a value in seconds. This will make the script print messages to whatever program is calling mariadb-test-run.pl at the interval you set to prevent timeouts.

If you are calling mariadb-test-run.pl directly, you do not need to specify a timeout.

The mariadb-test-run Stop File

The stop file is a temporary file that you create on your system when you want to pause the execution of mariadb-test-run. When enabled via the command line or environment variable options, mariadb-test-run will periodically check for the existence of the file and if it exists it will stop until the file is no longer present.

Examples

Command-line:

```
mariadb-test-run.pl --stop-file="/path/to/stop/file" --stop-keep-alive=120
```

Environment Variables:

```
export MTR_STOP_FILE="/path/to/stop/file"
export MTR_STOP_KEEP_ALIVE=120
mariadb-test-run.pl
```

This page is licensed: CC BY-SA / Gnu FDL

The Debug Sync Facility

The Debug Sync Facility allows placement of synchronization points in the server code by using the DEBUG_SYNC macro:

```
open_tables(...)

DEBUG_SYNC(thd, "after_open_tables");

lock_tables(...)
```

When activated, a sync point can

- Emit a signal and/or
- Wait for a signal

Nomenclature	Description
signal	A value of a global variable that persists until overwritten by a new signal. The global variable can also be seen as a “signal post” or “flag mast”. Then the signal is what is attached to the “signal post” or “flag mast”.
emit a signal	Assign the value (the signal) to the global variable (“set a flag”) and broadcast a global condition to wake those waiting for a signal.
wait for a signal	Loop over waiting for the global condition until the global value matches the wait-for signal.

By default, all sync points are inactive. They do nothing (except to burn a couple of CPU cycles for checking if they are active).

A sync point becomes active when an action is requested for it.
To do so, put a line like this in the test case file:

```
SET DEBUG_SYNC= 'after_open_tables SIGNAL opened WAIT_FOR flushed';
```

This activates the sync point `'after_open_tables'`. It requests it to emit the signal `'opened'` and wait for another thread to emit the signal `'flushed'` when the thread's execution runs through the sync point.

For every sync point there can be one action per thread only. Every thread can request multiple actions, but only one per sync point. In other words, a thread can activate multiple sync points.

Here is an example how to activate and use the sync points:

```
--connection conn1
SET DEBUG_SYNC= 'after_open_tables SIGNAL opened WAIT_FOR flushed';
send INSERT INTO t1 VALUES(1);
    --connection conn2
    SET DEBUG_SYNC= 'now WAIT_FOR opened';
    SET DEBUG_SYNC= 'after_abort_locks SIGNAL flushed';
    FLUSH TABLE t1;
```

When `conn1` runs through the `INSERT` statement, it hits the sync point `'after_open_tables'`. It notices that it is active and executes its action. It emits the signal `'opened'` and waits for another thread to emit the signal `'flushed'`.

`conn2` waits immediately at the special sync point `'now'` for another thread to emit the `'opened'` signal.

A signal remains in effect until it is overwritten. If `conn1` signals `'opened'` before `conn2` reaches `'now'`, `conn2` will still find the `'opened'` signal. It does not wait in this case.

When `conn2` reaches `'after_abort_locks'`, it signals `'flushed'`, which lets `conn1` awake.

Normally the activation of a sync point is cleared when it has been executed. Sometimes it is necessary to keep the sync point active for another execution. You can add an execute count to the action:

```
SET DEBUG_SYNC= 'name SIGNAL sig EXECUTE 3';
```

This sets the signal point's activation counter to 3. Each execution decrements the counter. After the third execution the sync point becomes inactive.

One of the primary goals of this facility is to eliminate sleeps from the test suite. In most cases it should be possible to rewrite test cases so that they do not need to sleep. (But this facility cannot synchronize multiple processes.) However, to support test development, and as a last resort, sync point waiting times out. There is a default timeout, but it can be overridden:

```
SET DEBUG_SYNC= 'name WAIT_FOR sig TIMEOUT 10 EXECUTE 2';
```

`TIMEOUT 0` is special: If the signal is not present, the wait times out immediately.

When a wait timed out (even on `TIMEOUT 0`), a warning is generated so that it shows up in the test result.

You can throw an error message and kill the query when a synchronization point is hit a certain number of times:

```
SET DEBUG_SYNC= 'name HIT_LIMIT 3';
```

Or combine it with signal and/or wait:

```
SET DEBUG_SYNC= 'name SIGNAL sig EXECUTE 2 HIT_LIMIT 3';
```

Here the first two hits emit the signal, the third hit returns the error message and kills the query.

For cases where you are not sure that an action is taken and thus cleared in any case, you can force to clear (deactivate) a sync point:

```
SET DEBUG_SYNC= 'name CLEAR';
```

If you want to clear all actions and clear the global signal, use:

```
SET DEBUG_SYNC= 'RESET';
```

This is the only way to reset the global signal to an empty string.

For testing of the facility itself you can execute a sync point just as if it had been hit:

```
SET DEBUG_SYNC= 'name TEST';
```

Formal Syntax

The string to "assign" to the DEBUG_SYNC variable can contain:

```
RESET |
<sync point name> TEST |
<sync point name> CLEAR |
<sync point name> {{SIGNAL <signal name> |
                   WAIT_FOR <signal name> [TIMEOUT <seconds>]}
                   [EXECUTE <count>] &| HIT_LIMIT <count>}
```

Here '&|' means 'and/or'. This means that one of the sections separated by '&|' must be present or both of them.

Activation/Deactivation

With a MariaDB for debug build, it can be enabled by a mysqld command line option:

```
--debug-sync-timeout[=default_wait_timeout_value_in_seconds]
```

`'default_wait_timeout_value_in_seconds'` is the default timeout for the `WAIT_FOR` action. If set to zero, the facility stays disabled.

The facility is enabled by default in the test suite, but can be disabled with:

```
mariadb-test-run.pl ... --debug-sync-timeout=0 ...
```

Likewise the default wait timeout can be set:

```
mariadb-test-run.pl ... --debug-sync-timeout=10 ...
```

The command line option influences the readable value of the debug_sync system variable.

- If the facility is not compiled in, the system variable does not exist.
- If `--debug-sync-timeout=0` the value of the variable reads as `"OFF"`.
- Otherwise the value reads as `"ON - current signal: "` followed by the current signal string, which can be empty.

The readable variable value is the same, regardless if read as a global or session value.

Setting the debug_sync system variable requires the `'SUPER'` privilege. You can never read back the string that you assigned to the variable, unless you assign the value that the variable already has. But that would give a parse error. A syntactically correct string is parsed into a debug sync action and stored apart from the variable value.

Implementation

Pseudo code for a sync point:

```
#define DEBUG_SYNC(thd, sync_point_name)
        if (unlikely(opt_debug_sync_timeout))
          debug_sync(thd, STRING_WITH_LEN(sync_point_name))
```

The sync point performs a binary search in a sorted array of actions for this thread.

The `SET DEBUG_SYNC` statement adds a requested action to the array or overwrites an existing action for the same sync point. When it adds a new action, the array is sorted again.

A typical synchronization pattern

There are quite a few places in MariaDB and MySQL where we use a synchronization pattern like this:

```
mysql_mutex_lock(&mutex);
thd->enter_cond(&condition_variable, &mutex, new_message);
#if defined(ENABLE_DEBUG_SYNC)
if (!thd->killed && !end_of_wait_condition)
   DEBUG_SYNC(thd, "sync_point_name");
#endif
while (!thd->killed && !end_of_wait_condition)
  mysql_cond_wait(&condition_variable, &mutex);
thd->exit_cond(old_message);
```

Here are some explanations:

`thd->enter_cond()` is used to register the condition variable and the mutex in `thd->mysys_var`. This is done to allow the thread to be interrupted (killed) from its sleep. Another thread can find the condition variable to signal and mutex to use for synchronization in this thread's `THD::mysys_var`.

`thd->enter_cond()` requires the mutex to be acquired in advance.

`thd->exit_cond()` unregisters the condition variable and mutex and releases the mutex.

If you want to have a Debug Sync point with the wait, please place it behind `enter_cond()`. Only then you can safely decide, if the wait is taken. Also you will have `THD::proc_info` correct when the sync point emits a signal. `DEBUG_SYNC` sets its own proc_info, but restores the previous one before releasing its internal mutex. As soon as another thread sees the signal, it does also see the proc_info from before entering the sync point. In this case it is "new_message", which is associated with the wait that is to be synchronized.

In the example above, the wait condition is repeated before the sync point. This is done to skip the sync point, if no wait takes place.
The sync point is before the loop (not inside the loop) to have it hit once only. It is possible that the condition variable is signaled multiple times without the wait condition to be true.

A bit off-topic: At some places, the loop is taken around the whole synchronization pattern:

```
while (!thd->killed && !end_of_wait_condition)
{
  mysql_mutex_lock(&mutex);
  thd->enter_cond(&condition_variable, &mutex, new_message);
  if (!thd->killed [&& !end_of_wait_condition])
  {
    [DEBUG_SYNC(thd, "sync_point_name");]
    mysql_cond_wait(&condition_variable, &mutex);
  }
  thd->exit_cond(old_message);
}
```

Note that it is important to repeat the test for thd->killed after `enter_cond()`. Otherwise the killing thread may kill this thread after it tested `thd->killed` in the loop condition and before it registered the condition variable and mutex in `enter_cond()`. In this case, the killing thread does not know that this thread is going to wait on a condition variable. It would just set `THD::killed`. But if we would not test it again, we would go asleep though we are killed. If the killing thread would kill us when we are after the second test, but still before sleeping, we hold the mutex, which is registered in mysys_var.
The killing thread would try to acquire the mutex before signaling the condition variable. Since the mutex is only released implicitly in `mysql_cond_wait()`, the signaling happens at the right place. We have a safe synchronization.

Co-work with the DBUG facility

When running the MariaDB test suite with the `--debug-dbug` command line option, the Debug Sync Facility writes trace messages to the DBUG trace. The following shell commands proved very useful in extracting relevant information:

```
egrep 'query:|debug_sync_exec:' mysql-test/var/log/mysqld.1.trace
```

It shows all executed SQL statements and all actions executed by synchronization points.

Sometimes it is also useful to see, which synchronization points have been run through (hit) with or without executing actions. Then add `"|debug_sync_point:"` to the egrep pattern.

Synchronizing DEBUG_SYNC Actions

Tests may need additional synchronization mechanisms between `DEBUG_SYNC` actions, because certain combinations of actions can result in lost signals. More specifically, once a `SIGNAL` action is issued, it is stored in a global variable for any waiting threads to determine if they are depending on that signal for continuing. However, if a subsequent action overwrites that variable before a waiting thread is able to check against it, the original signal is lost. Examples of actions which would change the variable state are another `SIGNAL` or a `RESET`. Therefore, before issuing these commands, the test writer should verify the previous signal has been acknowledged. The following code snippets show an example of a problematic pattern and a potential solution.

```
SET DEBUG_SYNC='now SIGNAL sig';
SET DEBUG_SYNC='RESET'; # Problematic because sig can be cleared before a waiting thread can acknowledge
it
```

```
SET DEBUG_SYNC='now SIGNAL sig';

# Don't issue the RESET until we have proven the waiting thread has received the signal
let $wait_condition= select count(*)=0 from information_schema.processlist where state like "debug sync
point%";
source include/wait_condition.inc;

SET DEBUG_SYNC='RESET'; # Now this is safe
```

This page is licensed: CC BY-SA / Gnu FDL

MyISAM Clients and Utilities

See Memory and Disk Use With myisamchk.

Tune myisamchk memory and disk usage to speed up checking and repairing large MyISAM tables.

See MyISAM Database Management using GUI Client.

An overview of managing MyISAM tables through graphical (GUI) client tools.

See myisam_ftdump.

Display information about MyISAM FULLTEXT indexes with the myisam_ftdump utility.

See myisamchk Table Information.

Use myisamchk options to report information about MyISAM tables and their indexes.

See myisamchk.

Check, repair, and optimize non-partitioned MyISAM tables from the command line with myisamchk.

See myisamlog.

Process and display the contents of a MyISAM log file with the myisamlog utility.

See myisampack.

Compress MyISAM tables into read-only form that is typically 40 to 70 percent smaller with myisampack.

This page is licensed: CC BY-SA / Gnu FDL

Memory and Disk Use With myisamchk

myisamchk's performance can be dramatically enhanced for larger tables by making sure that its memory-related variables are set to an optimum level.

By default, myisamchk will use very little memory (about 3MB is allocated), but can temporarily use a lot of disk space. If disk space is a limitation when repairing, the *–safe-recover* option should be used instead of *–recover*. However, if TMPDIR points to a memory file system, an out of memory error can easily be caused, as myisamchk places temporary files in TMPDIR. The *–tmpdir=path* option should be used in this case to specify a directory on disk.

myisamchk has the following requirements for disk space:

- When repairing, space for twice the size of the data file, available in the same directory as the original file. This is for the original file as well as a copy. This space is not required if the –quick option is used, in which case only the index file is re-created.
- Disk space in the temporary directory (TMPDIR or the tmpdir=path option) is needed for sorting if the –recover or –sort-recover options are used when not using –safe-recover). The space required is approximately (largest_key + row_pointer_length) * number_of_rows * 2. To get information about the length of the keys as well as the row pointer length, use myisamchk -dv table_name.
- Space for a new index file to replace the existing one. The old index is first truncated, so unless the old index file is not present or is smaller for some reason, no significant extra space is needed.

There are a number of system variables that are useful to adjust when running myisamchk. They will increase memory usage, and since some are per-session variables, you don't want to increase the general value, but you can either pass an increased value to myisamchk as a command line option, or with a [myisamchk] section in your my.cnf file.

- sort_buffer_size. By default this is 4M, but it's very useful to increase to make myisamchk sorting much faster. Since the server won't be running when you run myisamchk, you can increase substantially. 16M is usually a minimum, but values such as 256M are not uncommon if memory is available.
- key_buffer_size (which particularly helps with the –extend-check and –safe-recover options.
- read_buffer_size
- write_buffer_size

For example, if you have more than 512MB available to allocate to the process, the following settings could be used:

```
myisamchk
  --myisam_sort_buffer_size=256M
  --key_buffer_size=512M
  --read_buffer_size=64M
  --write_buffer_size=64M
...
```

This page is licensed: CC BY-SA / Gnu FDL

MyISAM Database Management using GUI Client

Navigating the complexities of database management requires the right tools and know-how, especially when dealing with specialized storage engines like MyISAM in MySQL and MariaDB environments. This comprehensive guide demystifies the role and functionality of MyISAM clients, offering you a deep dive into their essential features and operations. From initiating CRUD operations to optimizing table performance, we cover it all. We also spotlight dbForge Studio for MySQL, a feature-rich IDE that simplifies your interactions with MyISAM tables and databases. Whether you're a seasoned DBA or a developer looking to sharpen your database skills, this article serves as your go-to resource for understanding MyISAM clients and maximizing their capabilities.

What is a MyISAM client?

A MyISAM client refers to a specialized application or module designed to interact with the MyISAM database storage engine. Its primary function is to facilitate essential database operations such as creating, reading, updating, and deleting data (commonly referred to as CRUD operations) through specific protocols, APIs, or query languages.

In the context of MyISAM, it's important to note that we are essentially referring to a MySQL/MariaDB client, as the MyISAM storage engine is an integral part of these database management systems. A MyISAM client plays a pivotal role in initiating database management processes, sending queries to databases, and handling subsequent responses.

In simpler terms, any MyISAM client serves as the interface that allows users to interact with MySQL/MariaDB. There is a wide range of available solutions catering to different needs, including lightweight command line utilities like **myisamchk**, versatile options like the MySQL Command-Line Client, and feature-rich IDEs based on graphical user interfaces, such as MySQL Workbench or dbForge Studio for MySQL.

The choice of the right client largely depends on user preferences, the specific work environment, and the tasks they need to accomplish.

How it works: a MyISAM client in action

To better understand the functionality of a MyISAM client, let's delve into a practical scenario where a user needs to compose and execute a query against the database and subsequently retrieve the results. Whether you opt for a command line or GUI client, the process unfolds as follows:

Step 1: Connection Setup

The client initiates a connection to a MySQL/MariaDB server using the necessary library or connector. During this setup, it provides authentication credentials and specifies the database to be accessed.

Step 2: Query Submission

The client sends SQL queries to the MySQL/MariaDB server, utilizing the appropriate database engine—in our case, MyISAM.

Note: Modern GUI-based database clients, like dbForge Studio for MySQL and MariaDB, enhance query writing by offering essential features such as code auto-completion, formatting, debugging, code snippet libraries, and visual query builders for constructing intricate queries through diagrams.

Step 3: Query Execution

The database system takes charge of parsing and executing the SQL query. When the query involves a table managed by the MyISAM storage engine, MyISAM handles the data and its storage particulars.

Step 4: Result Retrieval

Upon completing query processing, the results are transmitted back to the client. The client then handles these results as needed—whether it's displaying them to the user or employing them for further operations.

Step 5: Connection Closure

Once the interaction is finished, the client responsibly closes the connection.

Throughout this process, it's essential to note that the client does not directly engage with MyISAM or any other storage engines. Instead, it interacts with the MySQL/MariaDB server, which, in turn, utilizes MyISAM to manage data at its level.

dbForge Studio for MySQL and MariaDB as the MyISAM client

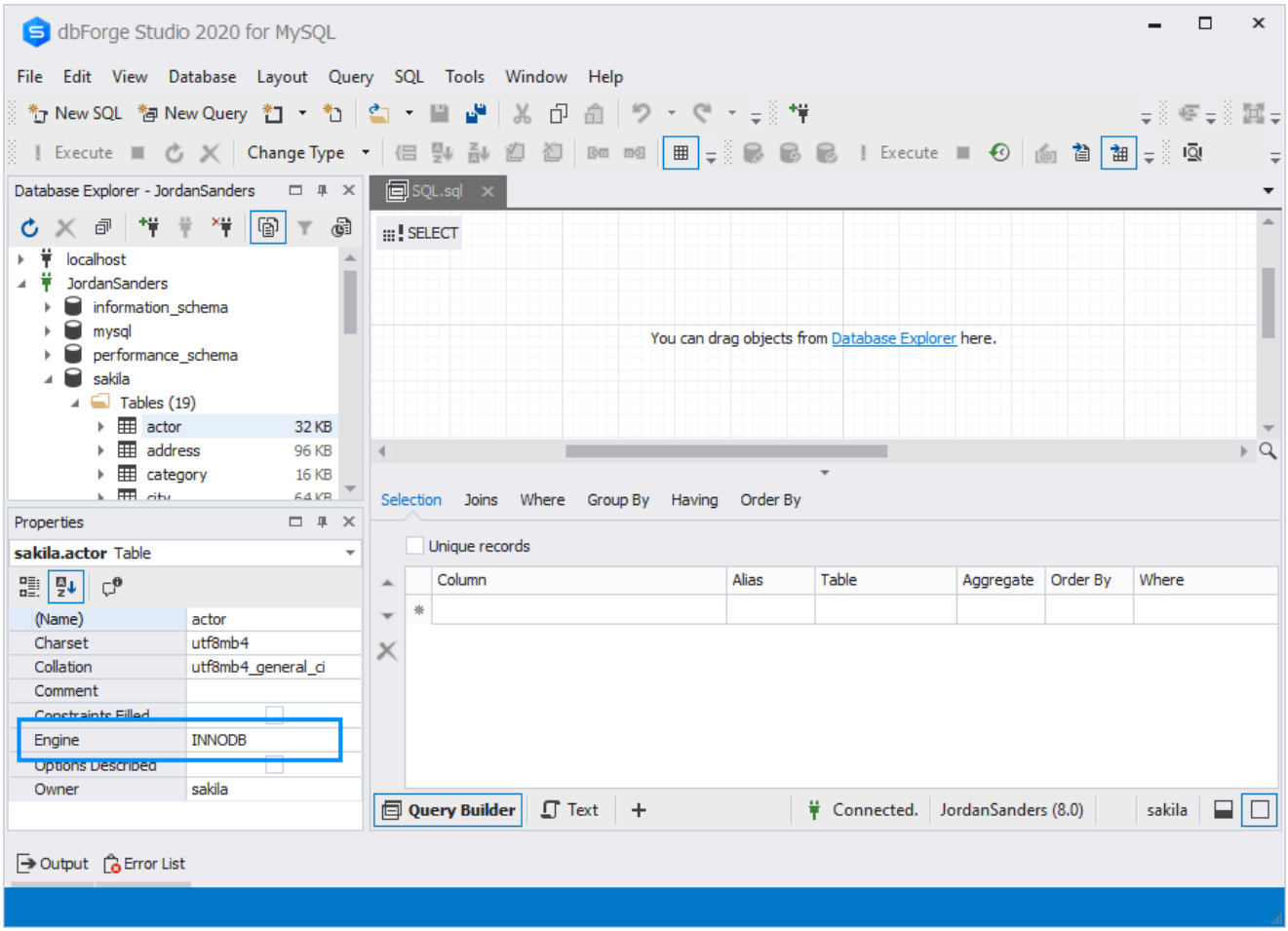
dbForge Studio for MySQL that is also a viable MariaDB tool offers full support for all MySQL storage engines, including MyISAM, and a user-friendly graphical interface that simplifies interactions with MyISAM tables. The functionality of the Studio makes it the common choice for professionals who prefer GUI tools over command line utilities.

It includes options available in the **myisamchk** utility and provides many more features to cover all database-related tasks on MySQL and MariaDB, no matter which storage engine those databases use.

Speaking of MyISAM tables, it is worth mentioning some basic operations.

Check what storage engine is used in the database

In dbForge Studio for MySQL, you can check the storage engine of the current database in a couple of clicks. Select the database in Database Explorer, right-click the necessary table, and select **Properties** from the shortcut menu. The output will present the information of the database engine among other data.



Check storage engine

Convert a table from InnoDB to MyISAM (and vice versa)

If you need to switch between database engines for some tables, you can use the standard ALTER TABLE command. dbForge Studio for MySQL allows executing SQL queries directly against the database.

To convert an InnoDB table to MyISAM, use the following command:

ALTER TABLE database_name.table_name ENGINE=MyISAM;

Img 1 – InnoDB, img 2 – MyISAM

The same ALTER TABLE command serves in the opposite situation where you want to convert MyISAM tables to InnoDB. You will need to specify ENGINE=InnoDB in that case.

MyISAM does not support foreign key constraints that are supported by InnoDB. In case you want to convert an InnoDB table having foreign keys into MyISAM, you will encounter an error. Make sure to drop those constraints before converting. There are more differences between MyISAM and InnoDB that should be considered when you deal with these two storage engines.

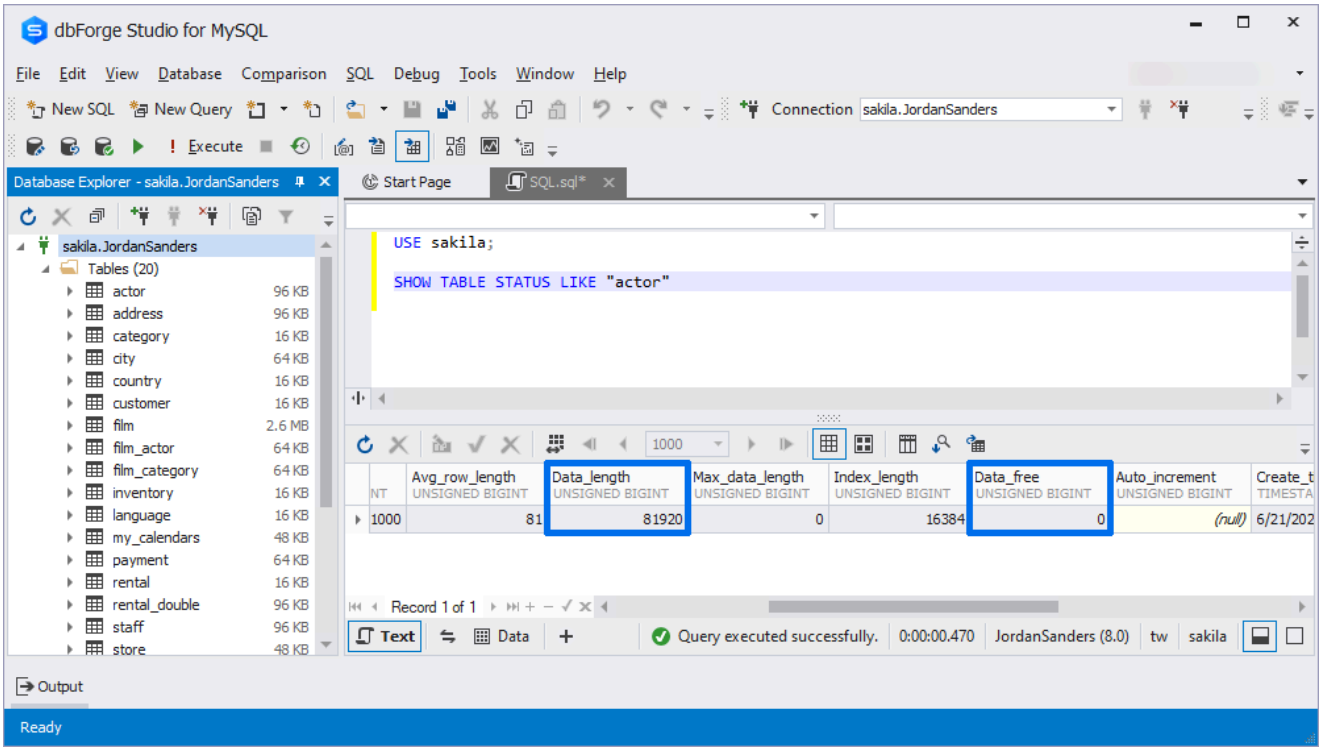
Optimize MyISAM tables

The primary function of myisamchk is to check, optimize, and repair MyISAM tables. This is crucial because frequent CRUD operations (CREATE, READ, DELETE, UPDATE) can lead to table corruption and a gradual decline in performance over time. dbForge Studio for MySQL includes a table maintenance tool that simplifies the process of identifying, analyzing, optimizing, and repairing tables.

To locate tables that require the administrator's attention, you can run the standard command in dbForge Studio for MySQL:

SHOW TABLE STATUS LIKE table_name;

This command provides detailed information about the table properties, including the total space occupied by the table and the total unused space (Data_length and Data_free columns, respectively). This helps users pinpoint tables that may be consuming excessive space and may be corrupted.



showtablestatus

This way, users can identify those tables that may take up too much space and be corrupted.

After detecting the MyISAM tables that require maintenance, you can apply the Studio's integrated table maintenance tool to those tables to fix the issues. To launch it, right-click the necessary table in Database Explorer and select **Table Maintenance**.

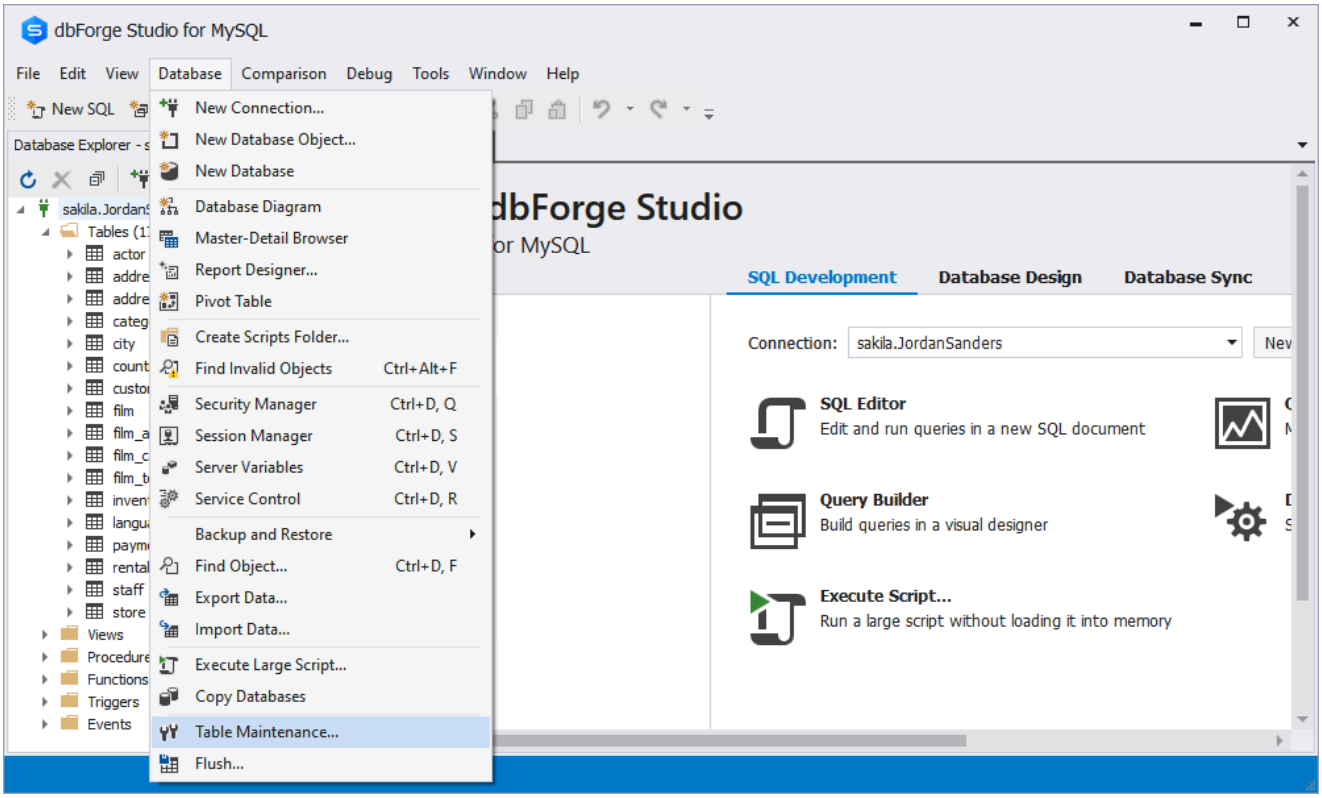
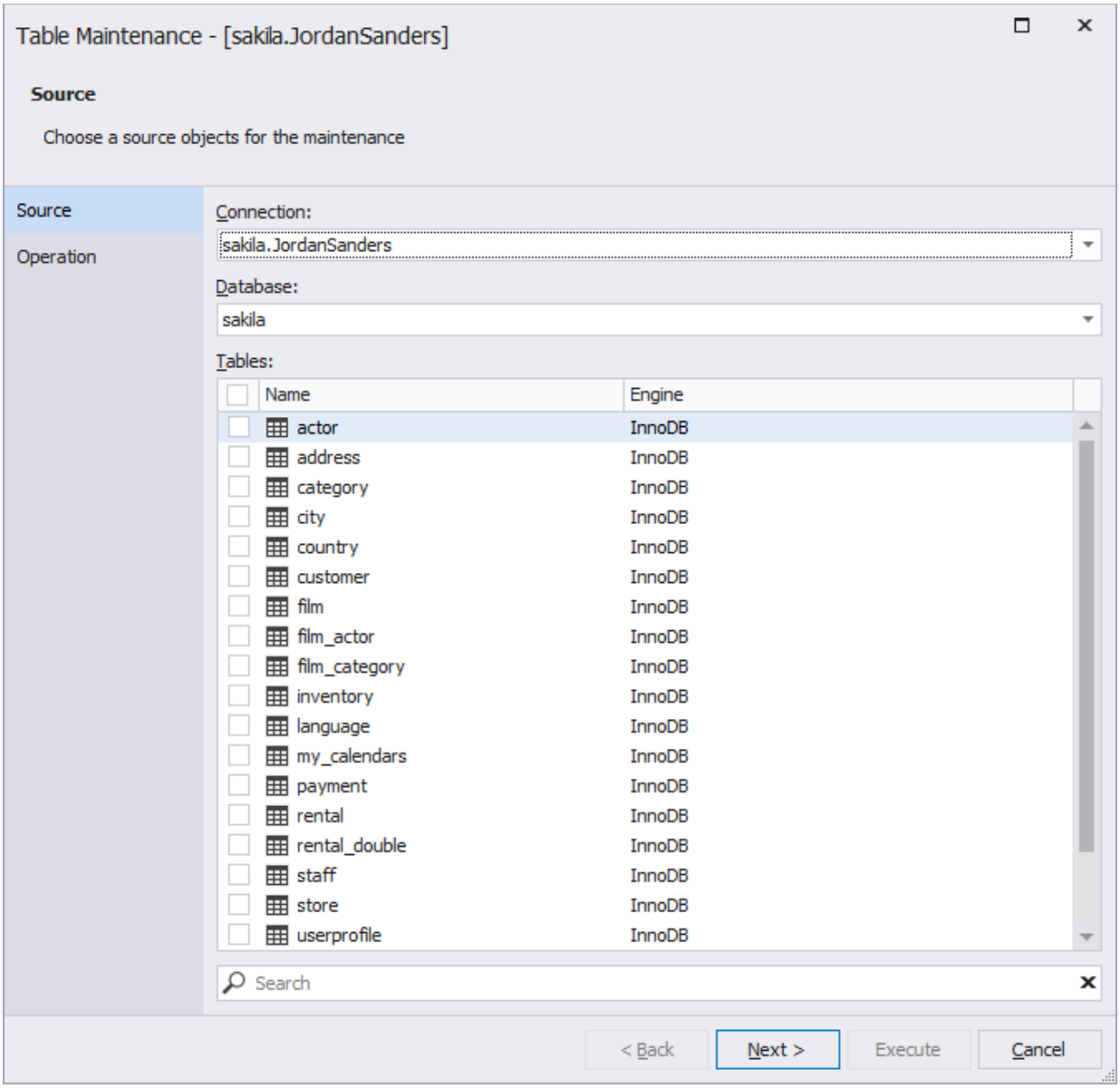


Table Maintenance

The table maintenance module offers the following options:



tablemaintenancewindow

- Analyze the table: This tool examines the key statistics of the table, helping to create more efficient query plans for optimal performance.
- Optimize the table: This option allows optimizing MyISAM tables for defragmentation, improving their performance.
- Check table errors: With this option, you can run diagnostic tests on tables, identifying and reporting all errors and corruption. It's worth noting that dbForge Studio provides flexible configuration for error checking, allowing users to determine the level of detail required. While a thorough scan for all possible errors may take time, it ensures that every issue is detected and reports 100% consistency.
- Checksum the table: This feature checks if the table has been modified, helping to maintain data integrity.
- Repair the table: This command is used to fix corrupted MyISAM tables. Applying this option may also recommend upgrading tables if necessary to rectify errors.

dbForge Studio streamlines all of these tasks with a single click. Users only need to specify the desired option and click **Execute**. This simplifies the process of maintaining MyISAM tables in MySQL databases.

More features of dbForge Studio for MySQL that can be applied to MyISAM tables and databases

The functionality of myisamchk is covered by one of many features available in dbForge Studio for MySQL. This multi-featured IDE offers a comprehensive toolset with flexible configuration and robust customization capacities to perform all database-related tasks in MySQL and MariaDB, no matter which database storage engine is used there. The following aspects are just several examples of the Studio's work:

- SQL Development: The functionality of this software client allows the users to construct and execute SQL queries and scripts, no matter the complexity, on databases using the MyISAM storage engine.
- Database Design: Users can both build database structures from scratch and modify existing ones, including creating and managing MyISAM tables and their respective indexes and keys.
- Database Administration: DBAs have access to essential tools for server connection management, performance monitoring, user account and privilege administration, and optimization and repair of tables using various storage engines.
- Data Management: Beyond basic data export and import tasks, dbForge Studio allows users to browse and edit data directly in the grid. You can also sort and filter data according to your specific needs.
- Backup and Restore: With support for MyISAM point-in-time backup and restore, the Studio can create backups of all types and restore them as needed. This ensures the safety, consistency, and stable performance of your databases.

Conclusion

Navigating the complexities of MyISAM storage engines doesn't have to be a daunting task. This article has provided you with an in-depth understanding of MyISAM clients and their integral role in MySQL and MariaDB database management. We've also showcased the unparalleled utility of dbForge Studio for MySQL, a one-stop IDE that simplifies everything from CRUD operations to advanced table maintenance. If you're keen on optimizing your database management processes, there's no better time to act.

Download a 30-day free trial of dbForge Studio to experience firsthand how this comprehensive tool can elevate your database operations to new heights.

This page is licensed: CC BY-SA / Gnu FDL

myisam_ftdump

myisam_ftdump is a utility for displaying information about MyISAM FULLTEXT indexes. It will scan and dump the entire index, and can be a lengthy process.

If the server is running, make sure you run a FLUSH TABLES statement first.

Usage

```
myisam_ftdump <table_name> <index_num>
```

The table_name can be specified with or without the .MYI index extension.

The index number refers to the number of the index when the table was defined, starting at zero. For example, take the following table definition:

```
CREATE TABLE IF NOT EXISTS `employees_example` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(30) NOT NULL,
  `last_name` VARCHAR(40) NOT NULL,
  `position` VARCHAR(25) NOT NULL,
  `home_address` VARCHAR(50) NOT NULL,
  `home_phone` VARCHAR(12) NOT NULL,
  `employee_code` VARCHAR(25) NOT NULL,
  `bio` text NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `employee_code` (`employee_code`),
  FULLTEXT (`bio`)
) ENGINE=MyISAM;
```

The fulltext index is `2`. The primary key is index `0`, and the unique key index `1`.

You can use *myisam_ftdump* to generate a list of index entries in order of frequency of occurrence as follows:

```
myisam_ftdump -c mytexttable 1 | sort -r
```

Options

Option	Description
-h, –help	Display help and exit.
-?, –help	Synonym for -h.
-c, –count	Calculate per-word stats (counts and global weights).
-d, –dump	Dump index (incl. data offsets and word weights).
-l, –length	Report length distribution.
-s, –stats	Report global stats.
-v, –verbose	Be verbose.

This page is licensed: CC BY-SA / Gnu FDL

myisamchk Table Information

myisamchk can be used to obtain information about MyISAM tables, particularly with the *-d*, *-e*, *-i* and *-v* options.

Common options for gathering information include:

- myisamchk -d
- myisamchk -dv
- myisamchk -dvv
- myisamchk -ei
- myisamchk -eiv

The *-d* option returns a short description of the table and its keys. Running the option while the table is being updated, and with external locking disabled, may result in an error, but no damage is done to the table. Each extra *v* adds more output. *-e* checks the table thoroughly (but slowly), and the *-i* options adds statistical information,

-dvv output

The following table describes the output from the running myisamchk with the *-dvv* option:

Heading	Description
MyISAM file	Name and path of the MyISAM index file (without the extension)

Heading	Description
Record format	Storage format. One of packed (dynamic), fixed or compressed.
Character set	Default character set for the table.
File-version	Always 1.
Creation time	Time the data file was created
Recover time	Most recent time the file was reconstructed.
Status	Table status. One or more of analyzed, changed, crashed, open, optimized keys and sorted index pages.
Auto increment key	Index number of the table's auto-increment column. Not shown if no auto-increment column exists.
Last value	Most recently generated auto-increment value. Not shown if no auto-increment column exists.
Data records	Number of records in the table.
Deleted blocks	Number of deleted blocks that are still reserving space. Use OPTIMIZE TABLE to defragment.
Datafile parts	For dynamic tables, the number of data blocks. If the table is optimized, this will match the number of data records.
Deleted data	Number of bytes of unreclaimed deleted data, Use OPTIMIZE TABLE to reclaim the space.
Datafile pointer	Size in bytes of the data file pointer. The size of the data file pointer, in bytes.
Keyfile pointer	Size in bytes of the index file pointer.
Max datafile length	Maximum length, in bytes, that the data file could become.
Max keyfile length	Maximum length, in bytes, that the index file could become.
Recordlength	Space, in bytes, that each row takes.
table description	Description of all indexes in the table, followed by all columns
Key	Index number, starting with one. If not shown, the index is part of a multiple-column index.
Start	Where the index part starts in the row.
Len	Length of the index or index part. The length of a multiple-column index is the sum of the component lengths. Indexes of string columns is shorter than the full column length if only a string prefix is indexed.
Index	Whether an index value is unique or not. Either multip. or unique.
Type	Data type of the index of index part.
Rec/key	Record of the number of rows per value for the index or index part. Used by the optimizer to calculate query plans. Can be updated with myisamchk-a. If not present, defaults to 30.
Root	Root index block address.
Blocksize	Index block size, in bytes.
Field	Column number, starting with one. The first line will contain the position and number of bytes used to store NULL flags, if any (see Nullpos and Nullbit, below).

Heading	Description
Start	Column's byte position within the table row.
Length	Column length, in bytes.
Nullpos	Byte containing the flag for NULL values. Empty if column cannot be NULL.
Nullbit	Bit containing the flag for NULL values. Empty if column cannot be NULL.
Type	Data type - see the table below for a list of possible values.
Huff tree	Only present for packed tables, contains the Huffman tree number associated with the column.
Bits	Only present for packed tables, contains the number of bits used in the Huffman tree.

Data type	Description
constant	All rows contain the same value.
no endspace	No endspace is stored.
no endspace, not_always	No endspace is stored, and endspace compression is not always performed for all values.
no endspace, no empty	No endspace is stored, no empty values are stored.
table-lookup	Column was converted to an ENUM.
zerofill(N)	Most significant N bytes of the value are not stored, as they are always zero.
no zeros	Zeros are not stored.
always zero	Zero values are stored with one bit.

-eiv output

The following table describes the output from the running myisamchk with the *-eiv* option:

Heading	Description
Data records	Number of records in the table.
Deleted blocks	Number of deleted blocks that are still reserving space. Use OPTIMIZE TABLE to defragment.
Key	Index number, starting with one.
Keyblocks used	Percentage of the keyblocks that are used. Percentages is higher for optimized tables.
Packed	Percentage space saved from packing key values with a common suffix.
Max levels	Depth of the b-tree index for the key. Larger tables and longer key values result in higher values.
Records	Number of records in the table.
M.recordlength	Average row length. For fixed rows, is the actual length of each row.
Packed	Percentage saving from stripping spaces from the end of strings.
Recordspace used	Percentage of the data file used.
Empty space	Percentage of the data file unused.
Blocks/Record	Average number of blocks per record. Values higher than one indicate fragmentation. Use OPTIMIZE TABLE to defragment.
Recordblocks	Number of used blocks. Will match the number of rows for fixed or optimized tables.
Deleteblocks	Number of deleted blocks
Recorddata	Used bytes in the data file.
Deleted data	Unused bytes in the data file.
Lost space	Total bytes lost, such as when a row is updated to a shorter length.
Linkdata	Sum of the bytes used for pointers linking disconnected blocks. Each is four to seven bytes in size.

Examples

```
myisamchk -d /var/lib/mysql/test/posts

MyISAM file:         /var/lib/mysql/test/posts
Record format:       Compressed
Character set:       utf8mb4_unicode_ci (224)
Data records:                1680  Deleted blocks:                0
Recordlength:                2758
Using only keys '0' of 5 possibly keys

table description:
Key Start Len Index   Type
1   1     8   unique  ulonglong
2   2265  80  multip. varchar prefix
    63    80          varchar
    17    5           binary
    1     8           ulonglong
3   1231  8   multip. ulonglong
4   9     8   multip. ulonglong
5   387   764 multip. ? prefix
```

```
myisamchk -dvv /var/lib/mysql/test/posts

MyISAM file:         /var/lib/mysql/test/posts
Record format:       Compressed
Character set:       utf8mb4_unicode_ci (224)
File-version:        1
Creation time:       2015-08-10 16:26:54
Recover time:        2015-08-10 16:26:54
Status:              checked,analyzed,optimized keys
Auto increment key:              1  Last value:                  1811
Checksum:               2299272165
Data records:                 1680  Deleted blocks:                 0
Datafile parts:               1680  Deleted data:                   0
Datafile pointer (bytes):        6  Keyfile pointer (bytes):        6
Datafile length:           4298092  Keyfile length:            156672
Max datafile length: 281474976710654  Max keyfile length: 288230376151710719
Recordlength:                 2758
Using only keys '0' of 5 possibly keys

table description:
Key Start Len Index   Type                     Rec/key         Root  Blocksize
1   1     8   unique  ulonglong                      1                    1024
2   2265  80  multip. varchar prefix               336                    1024
    63    80          varchar                      187
    17    5           binary                         1
    1     8           ulonglong                      1
3   1231  8   multip. ulonglong                     10                    1024
4   9     8   multip. ulonglong                    840                    1024
5   387   764 multip. ? prefix                       1                    4096

Field Start Length Nullpos Nullbit Type                          Huff tree  Bits
1     1     8                     zerofill(6)                           1     9
2     9     8                     zerofill(7)                           1     9
3     17    5                                                           1     9
4     22    5                                                           1     9
5     27    12                    blob                                  2     9
6     39    10                    blob                                  3     9
7     49    4                     always zero                           1     9
8     53    10                    blob                                  1     9
9     63    81                    varchar                               4     9
10    144   81                    varchar                               5     5
11    225   81                    varchar                               5     5
12    306   81                    varchar                               1     9
13    387   802                   varchar                               6     9
14    1189  10                    blob                                  1     9
15    1199  10                    blob                                  7     9
16    1209  5                                                           1     9
17    1214  5                                                           1     9
18    1219  12                    blob                                  1     9
19    1231  8                     no zeros, zerofill(6)                 1     9
20    1239  1022                  varchar                               7     9
21    2261  4                     always zero                           1     9
22    2265  81                    varchar                               8     8
23    2346  402                   varchar                               2     9
24    2748  8                     no zeros, zerofill(7)                 1     9
```

```
myisamchk -eiv /var/lib/mysql/test/posts
Checking MyISAM file: /var/lib/mysql/test/posts
Data records:    1680   Deleted blocks:       0
- check file-size
- check record delete-chain
No recordlinks
- check key delete-chain
block_size 1024:
block_size 2048:
block_size 3072:
block_size 4096:
- check index reference
- check data record references index: 1
Key:  1:  Keyblocks used:  92%  Packed:    0%  Max levels:  2
- check data record references index: 2
Key:  2:  Keyblocks used:  93%  Packed:   90%  Max levels:  2
- check data record references index: 3
Key:  3:  Keyblocks used:  92%  Packed:    0%  Max levels:  2
- check data record references index: 4
Key:  4:  Keyblocks used:  92%  Packed:    0%  Max levels:  2
- check data record references index: 5
Key:  5:  Keyblocks used:  88%  Packed:   97%  Max levels:  2
Total:    Keyblocks used:  91%  Packed:   91%

- check records and index references
Records:              1680    M.recordlength:     4102   Packed:             0%
Recordspace used:      100%   Empty space:           0%  Blocks/Record:   1.00
Record blocks:        1680    Delete blocks:         0
Record data:       6892064    Deleted data:          0
Lost space:           1284    Linkdata:           6264

User time 0.11, System time 0.00
Maximum resident set size 3036, Integral resident set size 0
Non-physical pagefaults 925, Physical pagefaults 0, Swaps 0
Blocks in 0 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 0, Involuntary context switches 74
```

This page is licensed: CC BY-SA / Gnu FDL

myisamchk

myisamchk

myisamchk is a command line tool for checking, repairing and optimizing non-partitioned MyISAM tables.

myisamchk is run from the command line as follows:

```
myisamchk [OPTIONS] tables[.MYI]
```

The full list of options are listed below. One or more MyISAM tables can be specified. MyISAM tables have an associated .MYI index file, and the table name can either be specified with or without the .MYI extension. Referencing it with the extension allows you to use wildcards, so it's possible to run myisamchk on *all* the MyISAM tables in the database with `*.MYI`.

The path to the files must also be specified if they're not in the current directory.

> **Important:**
>
> myisamchk should not be run while anyone is accessing any of the affected tables. It is also best to make a backup before running.

With no options, myisamchk simply checks your table as the default operation.

The following options can be set while passed as command line options to myisamchk, or set with a [myisamchk] section in your my.cnf file.

General Options

Option	Description
-H, –HELP	Display help and exit. Options are presented in a single list.
-?, –help	Display help and exit. Options are grouped by type of operation.
-# options, –debug=options	Write a debugging log. A typical debug options string is `d:t:o,file_name`. The default is `d:t:o,/tmp/myisamchk.trace`. (Available in debug builds only)
-t path, –tmpdir=path	Path for temporary files. Multiple paths can be specified, separated by colon (:) on Unix and semicolon (;) on Windows. They are used in a round-robin fashion. If not set, the TMPDIR environment variable is used.
-s, –silent	Only print errors. One can use two -s (-ss) to make myisamchk very silent.
-v, –verbose	Print more information. This can be used with –description and –check. Use many -v for more verbosity.
-V, –version	Print version and exit.
-w, –wait	If table is locked, wait instead of returning an error.
–print-defaults	Print the program argument list and exit.
–no-defaults	Don't read default options from any option file.
–defaults-file=filename	Only read default options from the given file filename, which can be the full path, or the path relative to the current directory.
–defaults-extra-file=filename	Read the file filename, which can be the full path, or the path relative to the current directory, after the global files are read.
–defaults-group-suffix=str	Also read groups with a suffix of str. For example, –defaults-group-suffix=x would read the groups [myisamchk] and [myisamchk_x]

The following variables can also be set by using *–var_name=value*, for example *–ft_min_word_len=5*

Variable	Default Value
decode_bits	9
ft_max_word_len	version-dependent
ft_min_word_len	4
ft_stopword_file	built-in list
key_buffer_size	1044480
key_cache_block_size	1024
myisam_block_size	1024
myisam_sort_buffer_size	134216704
myisam_sort_key_blocks	16
read_buffer_size	262136
sort_buffer_size	134216704
sort_key_blocks	16
stats_method	nulls_unequal
write_buffer_size	262136

Checking Tables

If no option is provided, myisamchk will perform a check table. It is possible to check MyISAM tables without shutting down or restricting access to the server by using CHECK TABLE instead.

The following check options are available:

Option	Description
-c, –check	Check table for errors. This is the default operation if you specify no option that selects an operation type explicitly.
-e, –extend-check	Check the table VERY thoroughly. Only use this in extreme cases as it may be slow, and myisamchk should normally be able to find out if the table has errors even without this switch. Increasing the key_buffer_size can help speed the process up.
-F, –fast	Check only tables that haven't been closed properly.
-C, –check-only-changed	Check only tables that have changed since last check.
-f, –force	Restart with '-r' (recover) if there are any errors in the table. States is updated as with '–update-state'.
-i, –information	Print statistics information about the table that is checked.
-m, –medium-check	Faster than extend-check, but only finds 99.99% of all errors. Should be good enough for most cases.
-U –update-state	Mark tables as crashed if you find any errors. This should be used to get the full benefit of the –check-only-changed option, but you shouldn´t use this option if the mariadbd server is using the table and you are running it with external locking disabled.
-T, –read-only	Don't mark table as checked. This is useful if you use myisamchk to check a table that is in use by some other application that does not use locking, such as mariadbd when run with external locking disabled.

Repairing Tables

It is also possible to repair MyISAM tables by using REPAIR TABLE.

The following repair options are available, and are applicable when using '-r' or '-o':

Option	Description
-B, –backup	Make a backup of the .MYD file as 'filename-time.BAK'.
–correct-checksum	Correct the checksum information for table.
-D len, –data-file-length=#	Max length of data file (when recreating data file when it's full).
-e, –extend-check	Try to recover every possible row from the data file. Normally this will also find a lot of garbage rows; Don't use this option if you are not totally desperate.
-f, –force	Overwrite old temporary files. Add another –force to avoid 'myisam_sort_buffer_size is too small' errors. In this case we will attempt to do the repair with the given myisam_sort_buffer_size and dynamically allocate as many management buffers as needed.
-k val, –keys-used=#	Specify which keys to update. The value is a bit mask of which keys to use. Each binary bit corresponds to a table index, with the first index being bit 0. 0 disables all index updates, useful for faster inserts. Deactivated indexes can be reactivated by using myisamchk -r.
–create-missing-keys	Create missing keys. This assumes that the data file is correct and that the number of rows stored in the index file is correct. Enables –quick
–max-record-length=#	Skip rows larger than this if myisamchk can't allocate memory to hold them.
-r, –recover	Can fix almost anything except unique keys that aren't unique (a rare occurrence). Usually this is the best option to try first. Increase myisam_sort_buffer_size for better performance.
-n, –sort-recover	Forces recovering with sorting even if the temporary file would be very large.

Option	Description
-p, –parallel-recover	Uses the same technique as '-r' and '-n', but creates all the keys in parallel, in different threads.
-o, –safe-recover	Uses old recovery method; Slower than '-r' but uses less disk space and can handle a couple of cases where '-r' reports that it can't fix the data file. Increase key_buffer_size for better performance.
–character-sets-dir=directory_name	Directory where the character sets are installed.
–set-collation=name	Change the collation (and by implication, the character set) used by the index.
-q, –quick	Faster repair by not modifying the data file. One can give a second '-q' to force myisamchk to modify the original datafile in case of duplicate keys. NOTE: Tables where the data file is corrupted can't be fixed with this option.
-u, –unpack	Unpack file packed with myisampack.

Other Actions

Option	Description
-a, –analyze	Analyze distribution of keys. Will make some joins faster as the join optimizer can better choose the order in which to join the tables and which indexes to use. You can check the calculated distribution by using '–description –verbose table_name' or SHOW INDEX FROM table_name.
–stats_method=name	Specifies how index statistics collection code should treat NULLs. Possible values of name are "nulls_unequal" (default), "nulls_equal" (emulate MySQL 4.0 behavior), and "nulls_ignored".
-d, –description	Print some descriptive information about the table. Specifying the –verbose option once or twice produces additional information.
-A [value], –set-auto-increment[=value]	Force auto_increment to start at this or higher value. If no value is given, then sets the next auto_increment value to the highest used value for the auto key + 1.
-S, –sort-index	Sort the index tree blocks in high-low order. This optimizes seeks and makes table scans that use indexes faster.
-R index_num, –sort-records=#	Sort records according to the given index (as specified by the index number). This makes your data much more localized and may speed up range-based SELECTs and ORDER BYs using this index. It may be VERY slow to do a sort the first time! To see the index numbers, SHOW INDEX displays table indexes in the same order that myisamchk sees them. The first index is 1.
-b offset, –block-search=offset	Find the record to which a block at the given offset belongs.

For more, see Memory and Disk Use With myisamchk.

Examples

Check all the MyISAM tables in the current directory:

```
myisamchk *.MYI
```

If you are not in the database directory, you can check all the tables there by specifying the path to the directory:

```
myisamchk /path/to/database_dir/*.MYI
```

Check all tables in all databases by specifying a wildcard with the path to the MariaDB data directory:

```
myisamchk /path/to/datadir/*/*.MYI
```

The recommended way to quickly check all MyISAM tables:

```
myisamchk --silent --fast /path/to/datadir/*/*.MYI
```

Check all MyISAM tables and repair any that are corrupted:

```
myisamchk --silent --force --fast --update-state \
  --key_buffer_size=64M --sort_buffer_size=64M \
  --read_buffer_size=1M --write_buffer_size=1M \
/path/to/datadir/*/*.MYI
```

See Also

- Memory and Disk Use With myisamchk

This page is licensed: CC BY-SA / Gnu FDL

myisamlog

`myisamlog` processes and returns the contents of a MyISAM log file.

Invoke `myisamlog` like this:

```
shell> myisamlog [options] [log_file [tbl_name] ...]
shell> isamlog [options] [log_file [tbl_name] ...]
```

The default operation is update (`-u`). If a recovery is done (`-r`), all writes and possibly updates and deletes are done and errors are only counted.
The default log file name is `myisam.log` for `myisamlog` and `isam.log` for `isamlog` if no `log_file` argument is given. If tables are named on the command line, only those tables are updated.

`myisamlog` supports the following options:

Option	Description
-?, -I	Display a help message and exit.
-c N	Execute only N commands.
-f N	Specify the maximum number of open files.
-i	Display extra information before exiting.
-o offset	Specify the starting offset.
-p N	Remove N components from path.
-r	Perform a recovery operation.
-R record_pos_file record_pos	Specify record position file and record position.
-u	Displays update operations.
-v	Verbose mode. Print more output about what the program does. This option can be given multiple times (-vv, -vvv) to produce more and more output.
-w write_file	Specify the write file.
-V	Display version information.

This page is licensed: CC BY-SA / Gnu FDL

myisampack

myisampack

`myisampack` is a tool for compressing MyISAM tables. The resulting tables are read-only, and usually about 40% to 70% smaller. It is run as follows:

```
myisampack [options] file_name [file_name2...]
```

The `file_name` is the `.MYI` index file. The extension can be omitted, although keeping it permits wildcards, such as:

```
myisampack *.MYI
```

…to compress all the files.

`myisampack` compresses each column separately, and, when the resulting data is read, only the individual rows and columns required need to be decompressed, allowing for quicker reading.

Once a table has been packed, use myisamchk -rq (the quick and recover options) to rebuild its indexes.

`myisampack` does not support partitioned tables or vector indexes.

> **Important:**
>
> Do not run myisampack if the tables could be updated during the operation, and skip_external_locking has been set.

Options

The following variables can be set while passed as command line options to `myisampack`, or set with a `[myisampack]` section in your my.cnf file.

Option	Description
-b, –backup	Make a backup of the table as table_name.OLD.
–character-sets-dir=name	Directory where character sets are.
-	
## , –debug[=name]	Output debug log. Often this is ‘d:t:o,filename’.
-f, –force	Force packing of table even if it gets bigger or if tempfile exists.
-j, –join=name	Join all given tables into ‘new_table_name’. All tables must have identical layouts.
-?, –help	Display help and exit.
-s, –silent	Only write output when an error occurs
-T, –tmpdir=name	Use temporary directory to store temporary table.
-t, –test	Don’t pack table, only test packing it.
-v, –verbose	Write info about progress and packing result. Use multiple -v flags for more verbosity.
-V, –version	Output version information and exit.
-w, –wait	Wait and retry if table is in use.

Uncompressing

To uncompress a table compressed with `myisampack`, use the myisamchk -u option.

Examples

```
> myisampack /var/lib/mysql/test/posts
Compressing /var/lib/mysql/test/posts.MYD: (1680 records)
- Calculating statistics
- Compressing file
37.71%
> myisamchk -rq /var/lib/mysql/test/posts
- check record delete-chain
- recovering (with sort) MyISAM-table '/var/lib/mysql/test/posts'
Data records: 1680
- Fixing index 1
- Fixing index 2
```

See Also

- FLUSH TABLES FOR EXPORT
- myisamchk

This page is licensed: CC BY-SA / Gnu FDL

Server & Client Software

See About MariaDB Software.

An overview of MariaDB, the open-source relational database — its background, licensing, and widespread adoption.

See Applications Supporting MariaDB.

A list of projects and applications that officially support MariaDB or its enhanced features.

See About the MariaDB Foundation.

The MariaDB Foundation is the custodian of the MariaDB code and steward of its open, vendor-neutral development.

See Client Libraries.

Explore server and client software for MariaDB. This section introduces various tools and applications for connecting to, managing, and interacting with your MariaDB Server instances effectively.

See Downloads.

Where to download MariaDB — tarballs, source, and Linux and Windows binaries, plus distribution packages.

This page is licensed: CC BY-SA / Gnu FDL

About MariaDB Software

MariaDB is an open-source, multi-threaded, relational database management system, released under the GNU Public License (GPL). MariaDB's lead developer is Michael "Monty" Widenius, who is one of the founders of MySQL AB.

There are millions of installations of MariaDB worldwide, and thousands of downloads a day of MariaDB installation software. The success of MariaDB as a leading database is due not only to open source, but also its reliability, performance, and features.

Features

Many features contribute to MariaDB's standing as a database system. Its speed is one of its most prominent features. MariaDB is remarkably scalable, and is able to handle tens of thousands of tables and billions of rows of

data. It can also manage small amounts of data quickly and smoothly, making it convenient for small businesses or personal projects.

Another feature that sets it apart from its predecessors is a particular focus on security. When critical security issues are discovered, our engineers immediately prepare and distribute a new release of MariaDB, as quickly as possible. The MariaDB team works closely with the Mitre Organization to ensure all security issues are promptly reported and documented in sufficient detail. These details are typically released after the new release has been published.

MariaDB's built-in functions include ones for manipulating and formatting text, business and statistical calculations, recording chronological information, as well as specialty features such ones related to GPS mapping.

Usage

We strive to develop a database system that works easily on the web and has one of the fastest connectors. Thanks to the fact that we use threads, we can handle much higher loads than other database systems. We have always striven to adapt to new hardware and to optimize the software for all commonly used systems and methods of deployment. Because we're continuously improving the software, we're able to provide a new release each month and a new version every year. That's an indication that things are happening and improving regularly.

In addition to web usage, MariaDB can be used for stand-alone applications ranging from enterprise transactional and analytical systems down to mobile devices, embedded with other software. MariaDB works in the cloud or on premise.

Due to its focus on performance and based on its threading model, MariaDB features the scale out technologies necessary for when your site or application experiences major spikes in traffic or rapid growth in business.

Security

With encryption that we've introduced in recent versions of MariaDB, you can be assured that your databases is secure — by default. Most other database systems don't have encryption by default.

Many organizations are leery about using commercial database software because they don't know for sure if the compiled code contains back-doors for accessing the data or some special way in which the software is using encryption that could allow a hacker to get at their data. Since MariaDB is open-source, governments and other organizations know that their databases are secure. Nothing is hidden.

Integration

The critical software in any database management system is its storage engine, which manages queries and interfaces between a user's SQL statements and the database's back-end storage. MariaDB offers several storage engines with different advantages. Some are transaction-safe storage engines that allow for rollback of data.

MariaDB also offers full integration with Galera cluster — an add-on for MariaDB for running multiple database servers for better performance and high availability. This is just one of many examples of how MariaDB integrates well with other software and systems. Such integration and MariaDB new encryption features have made it the premier database.

Version & Release Schedule

MariaDB is known for rapid and stable improvements. Each new release comes with speed and stability improvements, as well as new features. Current and development versions of MariaDB (and many previous versions) can be downloaded from the MariaDB Downloads Directory. There is also a yum repository for installing easily MariaDB and related software. Use the yum repository tool to configure your server.

See also the MariaDB maintenance policy.

MariaDB makes updates to active releases about once a month. The MariaDB Foundation provides a road map, which will give you information on when to expect the next release of MariaDB.

Learning MariaDB

If you're developing a career as a computer programmer, a web developer, or a similar role in computer technology, learning MariaDB will prove useful in your career. Many businesses develop and maintain custom software with MariaDB and the similar software. Additionally, many of the most popular web sites and software use MariaDB for their database — or they use another SQL database system that you can learn easily once you understand MariaDB.

It's highly likely that, for any job to which you apply or take as a database developer or administrator or as a web site developer, you is required to know or you isnefit from knowing MariaDB. Therefore, learning MariaDB is a good foundation for your career in computer technology. These documentation pages are your best resource for learning MariaDB in detail.

Related Pages

- MariaDB versus MySQL - Features
- MariaDB versus MySQL - Compatibility
- MariaDB Release Notes

This page is licensed: CC BY-SA / Gnu FDL

Applications Supporting MariaDB

> **Note:**
>
> The projects listed on this page are third-party software, not developed or maintained by MariaDB. MariaDB doesn't test, validate, or support them. Refer to each project's own documentation and license terms.

This page lists projects which officially implement MariaDB's enhanced features, or state that they work with MariaDB. If you know of a project which officially supports MariaDB and it isn't listed here, please let us know in the comments.

> **Note:**
>
> Note: Every project we know of which works with MySQL also works with MariaDB. This page is just to point out projects which officially support MariaDB (e.g. by mentioning MariaDB in their documentation or setup instructions).

Supports Enhanced MariaDB Features

- dbForge Studio is a versatile and feature-rich IDE designed for MySQL and MariaDB professionals. It fully supports all individual features of MariaDB, such as support for Packages and Sequences, and other specificities.
- Dynamic Active Record - The yii2-dynamic-ar extension uses MariaDB dynamic columns to add NoSQL-like documents to Yii 2 Framework's Active Record ORM.
- ocelotgui - supports MariaDB 10.2 window functions.
- Shard-query has special support for MariaDB 10.0 features
- SQL Maestro for MySQL is a database management tool that provides direct support for a number of MariaDB features like roles, check constraints, and virtual columns.

Officially Supports MariaDB

- Alfresco - Alfresco is for Enterprise Content Management and Business Process Management and it supports running with MariaDB Server.
- Bitbucket by Atlassian
- Codeship

- Dataedo - Database documentation tool with rich-text fields and ERD creator.
- DBeaver - free convenient cross-platform and cross-database Java GUI client which supports special MySQL/MariaDB features.
- dbForge Edge is a multidatabase solution for MariaDB/MySQL, SQL Server, Oracle, and PostgreSQL. It includes a full-featured IDE for MariaDB, covering all key tasks in database development, design, management, and administration.
- dbForge Compare Bundle - contains two separate tools, dbForge Schema Compare and dbForge Data Compare.
- dbForge Data Compare - data comparison and synchronization tool.
- dbForge Data Generator - a visual tool for generation of large volumes of meaningful test table data.
- dbForge Documenter - a handy MariaDB documentation tool that automatically generates documentation in HTML, PDF, and MARKDOWN file formats.
- dbForge Query Builder is a GUI tool designed to create any MySQL and MariaDB queries in a visual mode. The drag-and-drop feature ensures the easy and accurate building of even the most sophisticated statements.
- dbForge Schema Compare - a tool for comparison and sync of DDL differences between database objects
- DbVisualizer - Cross-platform universal database tool, supports MariaDB/MySQL specific features
- dotConnect for MySQL - data provider for MariaDB and MySQL built over ADO.NET with support for .NET Core 1.x and 2.0
- Devart ODBC Driver for MySQL - ODBC driver for direct access to MariaDB or MySQL database via TCP/IP without database client
- Devart Excel Add-ins for MySQL - connect Microsoft Excel to MariaDB and MySQL and work with data like with usual Excel worksheets
- Drupal
- ERBuilder Data Modeler - Data modeling tool for multiple databases platforms including MariaDB, MySQL, and more. Free and commercial versions available.
- eZ
- Flyway
- Friendica
- Geeklog (installation requirements)
- ImpressCMS,a community-developed content-management system
- Jelastic - Java in the cloud
- Kajona³ - Open Source Content Management Framework (support announcement)
- Matomo – Matomo is an open-source web analytics platform that prioritizes user privacy and data ownership, providing a powerful, GDPR-compliant alternative to Google Analytics.
- MediaWiki (installation guide)
- MicroStrategy
- Moodle
- Moon Modeler - database design tool for MariaDB
- Mroonga
- mycli - command line interface for MariaDB with auto-completion and syntax highlighting.
- MySQL Data Access Components - the library of components for direct access to MariaDB and MySQL from Delphi
- Neor Profile SQL
- ownCloud (documentation)
- phpMyAdmin
- Plone (documentation)

- Querious - Mac OS X database administration tool
- RAD Studio (Embarcadero) supports MariaDB from RAD Studio 10.2 Tokyo
- Replication Manager for MariaDB and MySQL - High Availability solution to manage MariaDB Server 10.0+ GTID-based replication topologies
- SafeNet ProtectFile rom Gemalto
- SaltOS
- sequelize - Dialect object-relationship-mapper for node.js.
- Skyvia - Cloud service for data integration, management and backup
- SQLTool Pro
- SSIS Data Flow Components for MySQL - set of components for SQL Server Integration Services packages that allow loading data from MariaDB and MySQL in SSIS Data Flows
- Talend
- TransProCloud
- 12PRESS - an online FLOSS meeting and events schedule presentation platform for 12-Step recovery fellowships (AA, NA, Alanon, etc.), their regions and area service committees, etc.
- Travis CI
- UI Bakery - Low-code development platform for building web applications on top of MariaDB data (integration page, documentation)
- Universal Data Access Components - the library of non-visual cross-database data access components for Delphi that includes support for MariaDB and MySQL
- Valentina Reports & Reports Server - Reports Server on macOS, Windows and Linux and deployable app solution using MariaDB as data source
- Valentina Studio - Database GUI on macOS, Windows and Linux. Free & Pro versions.
- WordPress (glossary from install documentation)
- WPИ-XM
- XAMPP (announcement)
- Wercker
- Yii (Yii 1.1 guide and Yii 2.0 guide)
- Zarafa 7.2 onwards comes with MariaDB support release notes
- Zend Framework (Zend_Db_Adapter)

See Also

- Graphical clients
- Distributions which include MariaDB

This page is licensed: CC BY-SA / Gnu FDL

About the MariaDB Foundation

The MariaDB Foundation ensures that there is not just one person or one company driving MariaDB/MySQL development. It is the custodian of the MariaDB code and guardian of the MariaDB community.

The MariaDB Foundation is the owner of the main MariaDB server project and owns mariadb.org. This ensures that the official MariaDB development tree will always be open for the MariaDB developer community.

Goals

That MariaDB be actively developed in the community and to:

- Increase adoption of MariaDB.

- Ensure sustainable high-quality efforts to build, test and distribute MariaDB.
- Ensure that community patches are reviewed and adopted.
- Guarantee a community voice and define development rules.
- Keep MariaDB compatible with MySQL.
- Maintain mariadb.org.

If you are using MariaDB in production and want to ensure that MariaDB is always actively developed, then you should sponsor or become a member of the MariaDB foundation!.

It's the MariaDB Foundation that ensures that all community patches, including MySQL source code, are merged into MariaDB. It also does the builds and QA of MariaDB and provides a lot of the MariaDB documentation.

We are very grateful to our current members and sponsors who are making this work possible!

More information about the Foundation can be found on the MariaDB foundation website.

This page is licensed: CC BY-SA / Gnu FDL

Client Libraries

See Connect and Query.

Connect to and query MariaDB using a connector (from an application) or a client (interactively or from scripts).

See Proxy Protocol Support.

Proxy protocol support, which lets proxy programs relay client IP addresses to MariaDB Server.

See Client/Server Protocol.

Pointers to the MariaDB client/server protocol reference documentation.

This page is licensed: CC BY-SA / Gnu FDL

Connect and Query

Database connections are made using a **connector** (from an application) or a **client** (interactively or from scripts).

Clients and connectors listed here are supported under MariaDB Corporation Engineering Policies.

Clients and connectors listed here are compatible with:

- MariaDB database products (including Enterprise Server and MaxScale)
- MariaDB Community Server

MariaDB Connectors

MariaDB Connectors are available for many popular programming languages.

Programming Language / Interface	Connector
C	MariaDB Connector/C
C++	MariaDB Connector/C++
Java - JDBC	MariaDB Connector/J
Java - R2DBC	MariaDB Connector/R2DBC
JavaScript	MariaDB Connector/Node.js
ODBC	MariaDB Connector/ODBC
Python	MariaDB Connector/Python

MariaDB Client

MariaDB Client can be used interactively or within scripts.

MariaDB Client is included with distributions of MariaDB database products.

Compatible third-party clients exist but are not listed here.

Executable Filename	Purpose
mariadb, mysql	Connect from the command line

For additional information about MariaDB Client, see MariaDB Client.

Tools and Utilities

Tools and utilities listed here are included with distributions of MariaDB database products and make a client connection.

Command-Line Executable	Purpose
mariadb-admin, mysqladmin	Check configuration and current status
mariadb-backup, mariadb-backup	Create and restore physical backups (including Aria, InnoDB, MyISAM, MyRocks)
mariadb-binlog, mysqlbinlog	Read binary logs or relay logs
mariadb-check, mysqlcheck	Perform table maintenance operations
mariadb-dump, mysqldump	Create logical backups
mariadb-import, mysqlimport	Load table data from CSV, TSV, and other text file formats
mariadb-show, mysqlshow	Display databases, tables, table columns, indexes
mariadb-slap, mysqlslap	Generate client load for testing

Business Intelligence (BI)

MariaDB database products are accessible from business intelligence (BI) platforms, including:

BI Platform	Detail
Microsoft Power BI	MariaDB Direct Query Adapter for Microsoft Power BI enables Microsoft Power BI Desktop users to remotely connect to and query their MariaDB database, including on MariaDB MariaDB Cloud, without downloading the entire data set to their local machine.

This page is: Copyright © 2026 MariaDB. All rights reserved.

Proxy Protocol Support

The proxy protocol allows proxy programs to relay the IP addresses of the clients to the server programs. It is important in the case of MariaDB, since IP information is actually a part of user identity.

How Proxy Protocol Works

As per the proxy protocol specification, the connecting client can prefix its first packet with a proxy protocol header. The server will parse the header and assume the client's IP address is the one set in the proxy header.

For example, if a client sends the proxy header (V1, text) which is "PROXY TCP4 192.168.0.1 192.168.0.11 56324 443\r\n", the server, after parsing, assumes the client's IP is 192.168.0.1

MariaDB server understands both Version 1 (text) and Version 2 (binary) of the proxy header.

If the protocol specified by the version 1 header is "UNKNOWN", the MariaDB server will treat the connection as if the proxy protocol was disabled. This can be used when a valid proxy protocol header is needed, but there is no

client to proxy, for example, when a proxy does a health check.

Enabling Proxy Protocol in MariaDB Server

To enable use of the proxy protocol, it is necessary to specify subnetworks that are allowed to send proxy headers, using the proxy-protocol-networks server variable.

proxy-protocol-networks is either a comma-separated list of (sub) networks or IP addresses. One can also use 'localhost' in this list, which means Unix domain socket/named pipe/shared memory connections are allowed as well. Or proxy-protocol-networks can be set to *, meaning that proxy headers are allowed from any client.

> **Warning:**
>
> Note that a client running on a host within an allowed proxy network or an IP address can itself pretend to be connected from any IP address whatsoever and thus can possibly impersonate other users. Generally, you should limit shell access to proxy hosts to a minimum. And remember, with `proxy-protocol-networks=*` every host is a proxy host.

Example in my.ini/my.cnf

```
proxy-protocol-networks=::1, 192.168.0.0/16, localhost
```

allows IPv6 connections from local machine ::1, from IP addresses starting with 192.168, and from connections made with Unix domain sockets or named pipes.

Client-Side Support for Proxy Protocol

Since the functionality is suited only to very specific proxy-like programs, most client APIs do not provide support for sending proxy headers. One exception is Connector/C version 3 or later. One can now use mysql_optionsv():

```
mysql_optionsv(mysql, MARIADB_OPT_PROXY_HEADER, header,  header_size)
```

prior to mysql_real_connect() or mysql_connect(), to send the header. In the call above, `_header_` is the proxy header with the type `void *`, and `_header_size_` is its size in bytes (type is `size_t`).

Example

```
const char *hdr="PROXY TCP4 192.168.0.1 192.168.0.11 56324 443\r\n";
mysql_optionsv(mysql, MARIADB_OPT_PROXY_HEADER, hdr,  strlen(hdr));
```

Using Proxy Protocol with MariaDB MaxScale

If you want to use proxy protocol with MaxScale:

- Add the IP address of the MaxScale server to proxy-protocol-networks
- In `maxscale.cnf`, add the proxy_protocol parameter for all configured servers

Once configured, MaxScale will proxy the credentials from the client to the server.

See Also

- Proxy Protocol

This page is licensed: CC BY-SA / Gnu FDL

Client/Server Protocol

Find detailed descriptions in the Reference section:

See Client/Server Protocol.

This page is licensed: CC BY-SA / Gnu FDL

Downloads

Latest Packages

Tarballs, source and binaries (Linux and Windows), and packages for some Linux distributions are available at mariadb.org/download.

We hope that interested community package maintainers will step forward, as others already have, to build packages for their distributions. We ask for strict adherence to your packaging system's best practices and invite you to create a bug report if our project impedes this in any way.

Instructions how to install the packages can be found here. See the Includes MariaDB documentation page for a list of which Linux distributions that include MariaDB.

Pre-Release Binaries

Binaries from our Buildbot system at buildbot.mariadb.org are available as containers on quay.io/mariadb-foundation/mariadb-devel:{major version}. Tags available are listed here.

These haven't passed through a full release process; however changes there have been reviewed and are considered complete by the server developers. Its recommend to use these for testing.

Getting the Source

The mariadb.org/download download page contains the source for all released binaries. You can find the latest source code at server

To retrieve the code, the Git source control software offers the path of least resistance. If you are unfamiliar with git, please refer to the git documentation for an understanding of version control with git.

For instructions on creating a local branch of MariaDB, see the Getting the MariaDB Source Code page.

See the Generic Build Instructions page for general instructions on compiling MariaDB from the source.
The source page has links to platform and distribution-specific information, including information on how we build the release packages.

Old Versions

Running the most recent MariaDB version is generally the best choice. Note that there are long-term releases, maintained for five years, short-term releases, maintained for one year, and rolling releases. However, some organizations still use old or very old versions of MariaDB. An upgrade would probably require important changes in their applications, and sometimes they don't even have the sources of those applications.

mariadb.org/download/ contains all historical releases.

There are source and repositories there. To use repositories see the following link:

- RHEL and RPM distros
- Debian/Ubuntu based distos

It is also possible to access Docker Official Images of MariaDB back to 5.5.40 when it became an official image (available tags).

See MariaDB Source Code.

Check out the MariaDB Server source code from GitHub with Git, including how to select a specific branch or release.

See MariaDB RPM Packages.

Where to find MariaDB RPM packages for Fedora and other distributions, plus the YUM repository configuration tool.

See Mirror Sites for MariaDB.

How MariaDB is distributed through mirror sites, and how to volunteer your server as an official MariaDB mirror.

See Where to Download MariaDB.

Where to download MariaDB tarballs, binaries, and distribution packages, covering the latest releases and historical versions.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Source Code

Checking out the Source with Git

The MariaDB source is hosted on GitHub: server. You can get a copy with:

```
git clone https://github.com/MariaDB/server mariadb-server
```

If you want a source tarball for a specific released MariaDB version, you can find it at downloads.mariadb.org.

At any given time, developers is working on their own branches locally or on GitHub, with the main MariaDB branches receiving pushes less often.

The main MariaDB branches are:

- main
- 13.0
- 12.3
- 11.8
- 11.4
- 10.11
- 10.6

Source repositories for the MariaDB Connectors are:

- MariaDB Connector/C
- MariaDB Connector/C++
- MariaDB Connector/J
- MariaDB Connector/Node.js
- MariaDB Connector/ODBC
- MariaDB Connector/Python

See also:

- Using git page for instructions on how to use git to check out the source code and switch between the various branches.
- Compiling MariaDB from source

> **Important:**
>
> The rest of this page contains information on checking out the MariaDB source from Launchpad. As the current source is now on GitHub, the information is mainly of historical interest and not useful for current development.

The instructions on this page will help you download your own local branch of the MariaDB source code repository with the full revision history. If you want a tarball of the source without the revision history, see the MariaDB download page.

Checking out the Source with Bazaar

If you simply want to browse the source code, you can do so from maria.

Prerequisites

You need Bazaar for revision control.

Instructions

1. Prepare a directory to keep your MariaDB code in:

```
mkdir $repo

# where $repo is some directory (ex: ~/repos)

cd $repo
bzr init-repo maria

# this creates ~/repos/maria
```

1. Get a clean local copy of the Maria repo with:

```
cd $repo/maria

# (ex: ~/repos/maria)

bzr branch lp:maria trunk
```

The above will give you the latest stable MariaDB version.
If you want to get another release use lp:maria/5.2, lp:maria/5.3…
For a complete list, go to Launchpad and choose 'Code' from the top menu on the page.

- Note: The initial branch operation can take a long time depending on the speed of your Internet connection and the load on launchpad. For this initial branch you need to download over a gigabyte of data.
- Note: Bzr is written in Python and very slow for initial checkout. Even on a fast connection, expect to need 1-2 GByte of RAM and possibly several CPU hours for the checkout.

1. If you get an error like:

```
bzr: ERROR: Unknown repository format: 'Bazaar RepositoryFormatKnitPack6 (bzr 1.9)'
```

then the version of bzr you are using is too old. Using version 1.12 or higher will fix this error.

1. If you have upgraded your bzr and are unable to successfully branch from launchpad, try using the source tree tarball (below) instead.
2. You can see the current history with:

```
cd $repo/maria/trunk
bzr log | less
```

1. If you are going to be hacking on the MariaDB source code. See the Contributing Code page for help.
2. If you just want to compile MariaDB at this point, see the Compiling MariaDB page.

Source Tree Tarball

For those that have trouble branching MariaDB from Launchpad we have created a tarball of a complete repository of the MariaDB tree.

Prerequisites

You need Bazaar to work with the repository.

Using the Source Tree Tarball

1. Download the mariadb-shared-repo.tgz file from one of the MariaDB mirrors.

- The file is 267MB, so the download may take a long time to complete depending on your Internet connection.

1. The .tgz file contains a .bzr directory. The parent directory of this .bzr directory is (or becomes) a shared repository containing the
   MariaDB source code. It is recommended to create a new directory for this, so the next step is to create a directory to house the repository. Call this directory anything you like (“maria”, “mariadb”,
   “my”, “src”, etc…). Once created, cd into the directory and untar the file. Here is an example using the name “mariadb” for the new directory, with the directory located in a directory called “src” in the home directory of the current user, and the mariadb-shared-repo.tgz file located in a directory named Downloads (also in the current user’s home directory):

```
mariadbdir="mariadb"
downloadsdir="${HOME}/Downloads"
sourcecodedir="${HOME}/src"
cd ${sourcecodedir}
mkdir ${mariadbdir}
cd ${mariadbdir}
tar -zxvf ${downloadsdir}/mariadb-shared-repo.tgz
```

1. After the untar step you will have a bzr shared repository, but not a working tree. While in the shared repository directory, use the `bzr branch` command to branch the MariaDB trees you are interested in. For example:

- `bzr branch lp:maria/5.2`
- `bzr branch lp:maria`

1. Thanks to the repository, either of the above commands will complete very fast.
2. Before working with the code, make sure you pull down the latest version of the source code:

```
cd mariadb-5.2 #or to wherever your MariaDB tree is
bzr pull
```

1. You can now use this source tree as if you had branched it from launchpad directly.

Alternate Bazaar Instructions

The following alternative instructions are what we have used for setting up repositories on our build machines in buildbot.

Shell Variables

To streamline later steps, we start by setting several shell variables. Set the values of the BZR and WORK_DIR variables to the appropriate values for your
Linux distribution. The rest of the variables in this section should not need to be changed.

Binaries:

```
BZR="/usr/bin/bzr"
```

Directories

```
WORK_DIR="${HOME}/work/monty_program"

MARIA_DIR=${WORK_DIR}/mariadb
MARIA_REPO="lp:maria"
MARIA_MASTER="${MARIA_DIR}/maria-local-master"
MARIA_WORK="${MARIA_DIR}/maria"

PACKAGING_DIR=${WORK_DIR}/packaging
PACKAGING_REPO="lp:~maria-captains/ourdelta/ourdelta-montyprogram-fixes"
PACKAGING_MASTER="${PACKAGING_DIR}/ourdelta-montyprogram-fixes-local-master"
PACKAGING_WORK="${PACKAGING_DIR}/ourdelta-montyprogram-fixes"
```

Source Checkout

Initialize your bzr working directories, if not done already:

```
$BZR init-repo $MARIA_DIR
$BZR init-repo $PACKAGING_DIR
```

Check out MariaDB sources:

```
$BZR branch $MARIA_REPO $MARIA_MASTER
$BZR branch $MARIA_MASTER $MARIA_WORK
```

Check out packaging sources (only for MariaDB 5.3 and below):

```
$BZR branch $PACKAGING_REPO $PACKAGING_MASTER
$BZR branch $PACKAGING_MASTER $PACKAGING_WORK
```

This page is licensed: CC BY-SA / Gnu FDL

MariaDB RPM Packages

Packages for Fedora Linux are included with our other packages available for download on .

We have also added YUM repositories to our online Repository Configuration Tool.

This page is licensed: CC BY-SA / Gnu FDL

Mirror Sites for MariaDB

We rely on mirrors to distribute MariaDB to the world through the official download site at download. If you would like to volunteer to become a mirror, thank you! Getting you set up is easy.

> **Tip:**
>
> The state of MariaDB mirrors is monitored at https://mirmon.mariadb.org/

How to Become a MariaDB Mirror

The only requirements for becoming a mirror are: 1. A willingness to mirror MariaDB 1. Mirroring is done using rsync 1. An https mirror should be utilised (http mirrors are not accepted any more) 1. Available bandwidth and disk space

- Each release of MariaDB takes about between 110-140GB of disk space for all of the various packages we create. There are several MariaDB releases per year, but only the most recent two from each main MariaDB series (as of March 2025 this is 10.5, 10.6, 10.11, 11.2, 11.4, 11.7 and 11.8) are kept on the mirrors
- We recommend our mirrors have at least a 100Mbit/sec connection to the Internet

1. The ability to do directory listings on mirrored sub-directories on your mirror host (for some platforms the download page links directly to mirrored sub-directories)
2. The mirror should update itself, preferably every 6 hours, but at the very least, twice a day. (See Updating Notes for more information)
3. Not mandatory: provide `rsync` or `ftp` access to the mirror for our mirrorbits system (See: Mirrorbits)

Updating Notes

MariaDB will normally have no more than one release per series (10.4, 10.5, etc…) per month. There may be times when this is not the case (such as if a major bug is found), but those times are few.

Timing for release announcements is generally oriented to the U.S. Eastern time zone. So if your mirror only updates less frequently than every 6 hours, have it update after midnight, but before 06:00, U.S. Eastern Time.

With rsync, it is no problem to update several times per day, and doing so is encouraged.

rsync Information

The two primary rsync URLs for mirrors to use are:

USA:

```
rsync.osuosl.org::mariadb
```

Europe:

```
mirror.netcologne.de::mariadb
```

Here are suggested rsync commands for a mirror to use:

```
rsync -a --partial --delete-after --delay-updates mirror.netcologne.de::mariadb
/path/to/local/download/directory/rsync_test
```

```
rsync -a --partial --delete-after --delay-updates rsync.osuosl.org::mariadb
/path/to/local/download/directory/rsync_test
```

Be aware that when using the above rsync command, a “mariadb” directory **will not** be created. Instead, the contents of the mariadb module is rsynced into whatever local folder you specify. We suggest creating a “mariadb” folder for this.

It is also a good idea to test your rsync command before adding it to a cron job by adding the `-v --dry-run` flags to the command temporarily so you can see what it will do (without actually doing anything).

As of June 2025, running the above command will use around 750GB of disk space in total.

See the rsync documentation for details on what each of the arguments in the command above does.

Secondary rsync mirrors

We also have several secondary rsync mirrors you can pull from. In many instances pulling from the secondary mirror is faster than pulling from the primary rsync mirrors. Here they are, with the country they are located in. Simply replace the rsync URI in the example above with one from this list to use it.

Country	Rsync URL
Australia	rsync://mirror.aarnet.edu.au/pub/MariaDB/
Bulgaria	rsync://mirrors.neterra.net/MariaDB/
Canada	rsync://mariadb.mirror.globo.tech/mariadb/
China	rsync://mirrors.tuna.tsinghua.edu.cn/mariadb/
Denmark	rsync://mirror.one.com/mariadb/
Estonia	rsync://mirrors.xtom.ee/mariadb/
Germany	rsync://archive.mariadb.org/mariadb/
Germany	rsync://mirror.netcologne.de/mariadb/
Kenya	rsync://mariadb.mirror.liquidtelecom.com/MariaDB/
Morocco	rsync://mirror.marwan.ma/mariadb/
Netherlands	rsync://ftp.nluug.nl/mariadb/
Netherlands	rsync://mirror.nl.mirhosting.net/mariadb/
Norway	rsync://rsync.fjordos.com/mariadb/
Poland	rsync://ftp.icm.edu.pl/pub/unix/database/mariadb/
Portugal	rsync://mirrors.ptisp.pt/mariadb/
Romania	rsync://mirrors.nxthost.com/mariadb/
Russia	rsync://mirror.mephi.ru/mariadb/
Russia	rsync://mirror.truenetwork.ru/MariaDB/
Spain	rsync://mirror.raiolanetworks.com/pub/mariadb/
Taiwan	rsync://ftp.ubuntu-tw.org/mariadb/
Thailand	rsync://mirror.kku.ac.th/mariadb/
United Kingdom	rsync://mariadb.gb.ssimn.org/mariadb/
United States	rsync://mirror.nodesdirect.com/mariadb/
United States	rsync://mirror.us.mirhosting.net/mariadb/
United States	rsync://rsync.osuosl.org/mariadb/
Uruguay	rsync://espejito.fder.edu.uy/mariadb/

The MariaDB Foundation mirrorbits system

The MariaDB Foundation maintains a mirror manager system (see: mirrorbits). It is hosted by Hetzner in Germany, it is **not** available via `rsync` so you can't use it as a source for your mirror but you are encouraged to use it for your MariaDB installation and deployments. You can check the status of the service at https://mirror.mariadb.org/mirrorstats.

Getting Added to the Mirror List

Once you have initially mirrored the MariaDB release tree, please send an email to (mirror (at) mariadb [dot] org) with the following information:

1. The name of the company or organization sponsoring the mirror (so we can give credit where credit is due).
2. Contact name (and email address) for your mirror (so we know who to contact if there are any issues)
3. The general physical location (e.g. "Paris, France", "Hong Kong", or "Austin, Texas, USA") of the mirror (we use geolocation to try and send people to a mirror close to them).
4. The base public URL for the mirror (e.g. `'http://mirror.example.net/pub/mariadb'`).

5. The mirror you are using to rsync from

Once we receive your email and verify your mirror is working we'll add you to the list of mirrors. Apologies in advance in case of slow responses - mirror updates are usually batched around release time.

Thank you for volunteering to mirror MariaDB!

Credits

The primary MariaDB mirrors have been generously provided by the Oregon State University Open Source Lab (at rsync.osuosl.org) and NetCologne (at mirror.netcologne.de).

Other mirrors (list not necessarily complete) are listed below and can be selected on the mariadb.org/download/ site:

Global (CDN)

- CICKU (Global - CloudFlare)

Algeria

- Hostarts (Algiers)

Australia

- AARNet (Brisbane)
- Digital Pacific (Sydney)
- Real World Group (Sydney)

Austria

- Alwyzon (Vienna)
- Digital Nova (Graz)
- Kumi Systems e.U (Vienna)
- next layer GmbH (Vienna)

Azerbaijan

- YER Hosting

Bulgaria

- Neterra Telecommunications

Chile

- Insacom (Santiago de Chile)

China - □□

- □□□□ TUNA □□ - Tsinghua University TUNA Association (Beijing)
- □□□□□□□□, □□ - USTC (Hefei)
- eScience Center, Nanjing University (□□□□)

Czech Republic

- HOSTING90 systems (Prague)
- vpsFree.cz

Denmark

- dotsrc.org - Aalborg University (Aalborg)
- Group.one (Copenhagen)

Estonia

- xTom GmbH (Tallinn)

France

- IRCAM, Institut de recherche et coordination acoustique/musique (Paris)
- OVH (Roubaix)

Germany

- 23media (Frankfurt)
- AG DSN (Dresden)
- creoline GmbH (Frankfurt am Main)
- dogado GmbH (Leipzig)
- Host Europe GmbH
- IPHH Internet Port Hamburg GmbH (Hamburg)
- NetCologne GmbH
- wilhelm.tel GmbH (Hamburg)
- www.n-ix.net (Nuremberg)
- xTom GmbH (Düsseldorf)

Greece

- University of Crete / Computer Center (Crete, Heraklion)

Hong Kong

- □□□□□□□□ - Nethub Online Limited
- xTom GmbH (Hong Kong)

Hungary

- Budapest University of Technology and Economics (Budapest)

India

- Indian Institute Of Technology Delhi (New Delhi)

Indonesia

- Universitas Surabaya (Surabaya)
- Citrahost By Citranet (Jakarta)
- Heru Nugroho (Jakarta)
- Nevacloud (Jakarta)
- Domainesia (Jakarta)

Iran

- Kernel.ir (Tehran)
- Mobinhost (Tehran)

- Parsvds (Tehran)

Ireland

- HEAnet Ltd. (Dublin)

Israel

- SPD Hosting LTD (Rosh Haayin)

Italy

- EvoWise (Milan)

Japan

- □□□□, □□□ - Yamagata University (Yonezawa)
- xTom GmbH (Osaka)
- KuronekoServer (Tokyo)
- S-TECH (Tokyo)

Kenya

- Liquid Telecom (Nairobi)

Lithuania

- UAB “Interneto Vizija” (Vilnius)

Moldova

- ihost.md (Chisinau)
- MangoHost.NET (Chisinau)

Morocco

- Marwan

The Netherlands

- bouwhuis.network (Amsterdam)
- NLUUG (Amsterdam)
- Triple IT B.V.
- xTom GmbH (Amsterdam)
- Mirhosting (Dronten)

Norway

- FjordOS (Oslo)

Philippines

- RISE

Poland

- ICM UW (Interdisciplinary Centre for Modelling, Warsaw University)

Portugal

- PTISP (Lisbon)
- WebHS (Lisbon)
- Universidade do Porto

Romania

- Chroot Network (Bucharest)
- nxtHost.com (Bucharest)

Russia

- docker.ru (Moscow)
- МИФИ - National Research Nuclear University “MEPhI” (Moscow)
- Truenetwork (Novosibirsk)

Singapore

- Daan van Gorkum
- Quape (operated by MariaDB Foundation on Quape’s infra)

South Korea

- zzunipark (Gwangju, South Korea)

Spain

- Raiola Networks (Madrid)

Switzerland

- MvA Internet Services GmbH (Zurich)

Taiwan

- OSSPlanet + Ubuntu-TW - Ubuntu □□□□□□□
- Blendbyte (Taipei)

Thailand

- Khon Kaen University (Khon Kaen)
- THZHost.com (Bangkok)

Ukraine

- Distrohub (Kyiv)
- mirohost.net (Kyiv)

The United States of America (USA)

- Accretive Networks (Washington, Seattle)
- EvoWise (California, Los Angeles)
- EvoWise (New York, New York)
- Gigenet (Chicago, Illinois)
- KnownHost (Atlanta, Georgia)
- Limestone Networks (Texas, Dallas)

- Nodes Direct (Florida, Jacksonville)
- Rackspace (Dallas, Virginia, Chicago, London, Sydney, Hong Kong)
- xTom GmbH (San Jose, California)
- Mirhosting (New York, New York)

Uruguay

- Universidad de la República - Facultad de Derecho (Montevideo)

The MariaDB Archive

The distributed mirrors above only hold the most recent 3-4 releases of each MariaDB series (10.6, 10.11, etc…). If you need to access older MariaDB releases, there is an archive server that contains every MariaDB release. This server is available via the web at archive.mariadb.org and via rsync at:

```
archive.mariadb.org::mariadb
```

Mirroring the MariaDB Archive with rsync

If you want to mirror the archive mirror, please contact us at mirror@mariadb.org in advance.

To mirror the archive with rsync, something like the following will get you everything:

```
rsync -avP archive.mariadb.org::mariadb/ /path/to/local/dir/
```

…or you can specify a specific release such as:

```
rsync -avP archive.mariadb.org::mariadb/mariadb-10.4.13/ /path/to/local/dir/
```

Warning, as of June 2025, the entire archive will use around 14 TB of disk space in total, and if you want to keep your mirror up to date (via cron routine for instance), please **first compare** the STATUS file **before** doing the rsync. This file is updated anytime a new content is added in the archive tree.

You can use something like the following:

```
rsync archive.mariadb.org::mariadb/STATUS /tmp/STATUS
diff -q /tmp/STATUS /path/to/local/dir/STATUS || rsync -avP archive.mariadb.org::mariadb/
/path/to/local/dir/
```

We reserve the right to block IPs that continually rsync the archive mirror when no new content was added, since it uselessly overloads the archive server, which uses a slow remote storage system.

To get a listing of the top level items in the archive including a list of every MariaDB release, just do:

```
rsync archive.mariadb.org::mariadb
```

If you would like to become a full archive mirror of MariaDB and be listed here, just let us know via the contact information above. Thanks!

This page is licensed: CC BY-SA / Gnu FDL

Where to Download MariaDB

The Latest Packages

Tarballs, binaries (Linux, Solaris, and Windows), and packages for some Linux distributions are available at mariadb.com/downloads/ or mariadb.org/download/ (which also contains a PDF version of the MariaDB Server

documentation).

We hope that interested community package maintainers will step forward, as others already have, to build packages for their distributions. We ask for strict adherence to your packaging system's best practices and invite you to create a bug report if our project impedes this in any way.

Instructions how to install the packages can be found here.

Distributions Which Include MariaDB

- Most distributions already include MariaDB. See Distributions Which Include MariaDB.

Pre-Release Binaries

Binaries from our Buildbot system (see also the Buildbot page), are available at archive. They are not suitable for use in production systems but may be of use for debugging.

Once at the above URL you will need to click on the MariaDB tree you are interested in, and then the build. The build number corresponds to the `tarbuildnum` variable in Buildbot.

For example, if you were interested in the bsd9-64 build of the MariaDB 5.5 tree, revision 3497, the `tarbuildnum` is listed in the "Build Properties" table of the Buildbot build report.
In this case, the value is "2434".

Getting the Source

You can find all the source code at server

To retrieve the code, the Git source control software offers the path of least resistance. If you are unfamiliar with git, please refer to the git documentation for an understanding of version control with git.

For instructions on creating a local branch of MariaDB, see the Getting the MariaDB Source Code page.

See the Generic Build Instructions page for general instructions on compiling MariaDB from the source.
The source page has links to platform and distribution-specific information, including information on how we build the release packages.

This page is licensed: CC BY-SA / Gnu FDL

Analyzing Tools

See EXPLAIN Analyzer.

The EXPLAIN Analyzer was an online tool for analyzing and sharing EXPLAIN output. It is no longer active.

See EXPLAIN Analyzer API.

The EXPLAIN Analyzer API let applications submit EXPLAIN output to the online analyzer. It is no longer active.

This page is licensed: CC BY-SA / Gnu FDL

EXPLAIN Analyzer

> **Warning:**
>
> The `EXPLAIN` Analyzer is no longer active.

The EXPLAIN Analyzer was an online tool for analyzing and optionally sharing the output of both `EXPLAIN` and `EXPLAIN EXTENDED`.

Using the Analyzer

Using the analyzer is very simple.

In the mariadb client, run `EXPLAIN` on a query and copy the output. For example:

```
EXPLAIN SELECT * FROM t1 INNER JOIN t2 INNER JOIN t3 WHERE t1.a=t2.a AND t2.a=t3.a;
+------+-------------+-------+------+---------------+------+---------+------+------+--------------------
-----------------------------------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra
|
+------+-------------+-------+------+---------------+------+---------+------+------+--------------------
-----------------------------------+
|    1 | SIMPLE      | t1    | ALL  | NULL          | NULL | NULL    | NULL |    3 |
|
|    1 | SIMPLE      | t2    | ALL  | NULL          | NULL | NULL    | NULL |    3 | Using where; Using
join buffer (flat, BNL join)        |
|    1 | SIMPLE      | t3    | ALL  | NULL          | NULL | NULL    | NULL |    3 | Using where; Using
join buffer (incremental, BNL join) |
+------+-------------+-------+------+---------------+------+---------+------+------+--------------------
-----------------------------------+
3 rows in set (0.00 sec)
```

1. Paste the output into the EXPLAIN Analyzer input box and click the “Analyze Explain” button.
2. The formatted `EXPLAIN` is shown. You can now click on various part to get more information about them.

Some Notes:

- As you can see in the example above, you don’t need to chop off the query line or the command prompt.
- To save the EXPLAIN, so you can share it, or just for future reference, click the “Save Explain for analysis and sharing” button and then click the “Analyze Explain” button. You is given a link which leads to your saved `EXPLAIN` . For example, the above explain can be viewed here:
- Some of the elements in the formatted `EXPLAIN` are clickable. Clicking on them will show pop-up help related to that element.

Clients which integrate with the Explain Analyzer

The Analyzer has an API that client programs can use to send EXPLAINs. If you are a client application developer, see the EXPLAIN Analyzer API page for details.

The following clients have support for the EXPLAIN Analyzer built in:

HeidiSQL

HeidiSQL has a button when viewing a query that sends the query to the explain analyzer.

CC BY-SA / Gnu FDL

This page is licensed: CC BY-SA / Gnu FDL

EXPLAIN Analyzer API

> **Warning:**
>
> `EXPLAIN` Analyzer is no longer active.

The online `EXPLAIN` Analyzer tool has an open API to allow client applications to send it EXPLAINs.

Sending EXPLAINs to the EXPLAIN Analyzer

To send an `EXPLAIN` to the `EXPLAIN` Analyzer, simply `POST` or `GET` to the following address:

```
mariadb.org/explain_analyzer/api/1/?raw_explain=EXPLAIN&client=CLIENT
```

Replace “EXPLAIN” with the output of the EXPLAIN command and “CLIENT” with the name of your client.

Client Banner

If you like, you can have a banner promoting your client appear at the bottom of the page. Once you’ve added support for the `EXPLAIN` Analyzer to your client application, just send a logo, the name of your client, and what you want the name and logo to link to bryan AT montyprogram DOT com.

CC BY-SA / Gnu FDL

This page is licensed: CC BY-SA / Gnu FDL

HA & Performance

See Optimization and Tuning.

Optimize and tune MariaDB Server. This section covers buffers and caches, indexing, the query optimizer and hints, compression, and the system variables that control performance.

See Replication.

Replicate data with MariaDB Server. This section covers primary and replica setup, binary log formats, global transaction IDs, and multi-source, parallel, and semisynchronous replication.

See Benchmarking.

Benchmark MariaDB Server performance using published benchmark results and SystemTap scripts for measuring throughput and behavior under load.

See MariaDB Memory Allocation.

Plan MariaDB memory usage by sizing global caches, per-connection buffers, and engine-specific settings to avoid swapping and out-of-memory conditions.

See Hardware Optimization.

Optimize MariaDB Server performance through hardware. This section covers selecting appropriate CPU, memory, and storage configurations to maximize your database's speed and throughput.

See Connection Redirection Mechanism in the MariaDB Client/Server Protocol.

Explore connection redirection. This section details how the client/server protocol handles redirection for high availability and load balancing, ensuring seamless database access and failover.

This page is licensed: CC BY-SA / Gnu FDL

Optimization and Tuning

See Buffers, Caches and Threads.

Covers essential configurations to maximize throughput and responsiveness for your database workloads.

See MariaDB Internal Optimizations.

Delves into how the database engine enhances query execution, data storage, and overall performance through its core architecture.

See Operating System Optimizations.

Covers configuring your OS for improved I/O, memory management, and network settings to maximize database efficiency.

See Optimization and Indexes.

Covers index types, creation, and best practices for leveraging them to significantly improve query performance and data retrieval speed.

See Optimizer Hints.

Optimizer hints are options available that affect the execution plan.

See Compression.

Details how to apply data compression at various levels to reduce disk space and improve I/O efficiency.

See Optimizing Data Structure.

Covers schema design, data types, and normalization techniques to improve query efficiency and storage utilization.

See Optimizing Tables.

Covers various techniques, including proper indexing, data types, and storage engine choices, to improve query speed and efficiency.

See Optimizing Queries.

Provides techniques for writing efficient SQL, understanding query execution plans, and leveraging indexes effectively to speed up your queries.

See System and Status Variables.

Optimize MariaDB Server with system variables, configuring various parameters to fine-tune performance, manage resources, and adapt the database to your specific workload requirements.

This page is licensed: CC BY-SA / Gnu FDL

Buffers, Caches and Threads

See Query Cache.

Complete MariaDB performance optimization guide. Complete reference for query tuning, indexing strategies, and configuration improvements for production use.

See Thread Command Values.

The COMMAND values a thread can show in SHOW PROCESSLIST, and what each one means.

See Thread Pool.

Optimize MariaDB Server with the thread pool. This section explains how to manage connections and improve performance by efficiently handling concurrent client requests, reducing resource overhead.

See Thread States.

Understand MariaDB Server thread states. This section explains the different states a thread can be in, helping you monitor and troubleshoot query execution and server performance.

This page is licensed: CC BY-SA / Gnu FDL

Query Cache

The query cache stores results of `SELECT` queries so that if the identical query is received in future, the results can be quickly returned.

This is extremely useful in high-read, low-write environments (such as most websites). It does not scale well in environments with high throughput on multi-core machines, so it is disabled by default.

Note that the query cache cannot be enabled in certain environments. See Limitations.

Setting Up the Query Cache

Unless MariaDB has been specifically built without the query cache, the query cache will always be available, although inactive. The have_query_cache server variable will show whether the query cache is available.

```
SHOW VARIABLES LIKE 'have_query_cache';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| have_query_cache | YES   |
+------------------+-------+
```

If this is set to `NO`, you cannot enable the query cache unless you rebuild or reinstall a version of MariaDB with the cache available.

To see if the cache is enabled, view the query_cache_type server variable. It is disabled by default — enable it by setting `query_cache_type` to `1` :

```
SET GLOBAL query_cache_type = 1;
```

The query_cache_size is set to 1MB by default. Set the cache to a larger size if needed, for example:

```
SET GLOBAL query_cache_size = 2000000;
```

The `query_cache_type` is automatically set to `ON` if the server is started with the `query_cache_size` set to a non-zero (and non-default) value.

See Limiting the size of the Query Cache below for details.

How the Query Cache Works

When the query cache is enabled and a new `SELECT` query is processed, the query cache is examined to see if the query appears in the cache.

Queries are considered identical if they use the same database, same protocol version and same default character set. Prepared statements are always considered as different to non-prepared statements, see Query cache internal structure for more info.

If the identical query is not found in the cache, the query will be processed normally and then stored, along with its result set, in the query cache. If the query is found in the cache, the results will be pulled from the cache, which is much quicker than processing it normally.

Queries are examined in a case-sensitive manner, so :

```
SELECT * FROM t
```

Is different from :

```
SELECT * from t
```

Comments are also considered and can make the queries differ, so :

```
/* retry */SELECT * FROM t
```

Is different from :

```
/* retry2 */SELECT * FROM t
```

See the query_cache_strip_comments server variable for an option to strip comments before searching.

Each time changes are made to the data in a table, all affected results in the query cache are cleared. It is not possible to retrieve stale data from the query cache.

When the space allocated to query cache is exhausted, the oldest results will be dropped from the cache.

When using `query_cache_type=ON`, and the query specifies `SQL_NO_CACHE` (case-insensitive), the server will not cache the query and will not fetch results from the query cache.

When using `query_cache_type=DEMAND` and the query specifies `SQL_CACHE`, the server will cache the query.

Queries Stored in the Query Cache

If the query_cache_type system variable is set to `1`, or `ON`, all queries fitting the size constraints will be stored in the cache unless they contain a `SQL_NO_CACHE` clause, or are of a nature that caching makes no sense, for example making use of a function that returns the current time. Queries with `SQL_NO_CACHE` will not attempt to acquire query cache lock.

If any of the following functions are present in a query, it will not be cached. Queries with these functions are sometimes called 'non-deterministic' — don't get confused with the use of this term in other contexts.

BENCHMARK()	CONNECTION_ID()
CONVERT_TZ()	CURDATE()
CURRENT_DATE()	CURRENT_TIME()
CURRENT_TIMESTAMP()	CURTIME()
DATABASE()	ENCRYPT() (one parameter)
FOUND_ROWS()	GET_LOCK()
LAST_INSERT_ID()	LOAD_FILE()
MASTER_POS_WAIT()	NOW()
RAND()	RELEASE_LOCK()
SLEEP()	SYSDATE()
UNIX_TIMESTAMP() (no parameters)	USER()
UUID()	UUID_SHORT()

A query will also not be added to the cache if:

- It is of the form:
  - SELECT SQL_NO_CACHE …
  - SELECT … INTO OUTFILE …
  - SELECT … INTO DUMPFILE …
  - SELECT … FOR UPDATE
  - SELECT * FROM … WHERE autoincrement_column IS NULL
  - SELECT … LOCK IN SHARE MODE
- It uses TEMPORARY table
- It uses no tables at all
- It generates a warning
- The user has a column-level privilege on any table in the query
- It accesses a table from INFORMATION_SCHEMA, mysql or the performance_schema database
- It makes use of user or local variables
- It makes use of stored functions
- It makes use of user-defined functions
- It is inside a transaction with the SERIALIZABLE isolation level
- It is querying a table inside a transaction after the same table executed a query cache invalidation using `INSERT`, `UPDATE` or `DELETE`

The query itself can also specify that it is not to be stored in the cache by using the `SQL_NO_CACHE` attribute. Query-level control is an effective way to use the cache more optimally.

It is also possible to specify that *no* queries must be stored in the cache unless the query requires it. To do this, the query_cache_type server variable must be set to `2`, or `DEMAND`. Then, only queries with the `SQL_CACHE` attribute are cached.

Limiting the Size of the Query Cache

There are two main ways to limit the size of the query cache. First, the overall size in bytes is determined by the query_cache_size server variable. About 40KB is needed for various query cache structures.

The query cache size is allocated in 1024 byte-blocks, thus it should be set to a multiple of 1024.

The query result is stored using a minimum block size of query_cache_min_res_unit. Check two conditions to use a good value of this variable: Query cache insert result blocks with locks, each new block insert lock query cache, a small value will increase locks and fragmentation and waste less memory for small results, a big value will increase memory use wasting more memory for small results but it reduce locks. Test with your workload for fine tune this variable.

If the strict mode is enabled, setting the query cache size to an invalid value will cause an error. Otherwise, it will be set to the nearest permitted value, and a warning will be triggered.

```
SHOW VARIABLES LIKE 'query_cache_size';
+------------------+----------+
| Variable_name    | Value    |
+------------------+----------+
| query_cache_size | 67108864 |
+------------------+----------+

SET GLOBAL query_cache_size = 8000000;
Query OK, 0 rows affected, 1 warning (0.03 sec)

SHOW VARIABLES LIKE 'query_cache_size';
+------------------+---------+
| Variable_name    | Value   |
+------------------+---------+
| query_cache_size | 7999488 |
+------------------+---------+
```

The ideal size of the query cache is very dependent on the specific needs of each system. Setting a value too small will result in query results being dropped from the cache when they could potentially be reused later. Setting a value too high could result in reduced performance due to lock contention, as the query cache is locked during updates.

The second way to limit the cache is to have a maximum size for each set of query results. This prevents a single query with a huge result set taking up most of the available memory and knocking a large number of smaller queries out of the cache. This is determined by the query_cache_limit server variable.

If you attempt to set a query cache that is too small (the amount depends on the architecture), the resizing will fail and the query cache will be set to zero, for example :

```
SET GLOBAL query_cache_size=40000;
Query OK, 0 rows affected, 2 warnings (0.03 sec)

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------+
| Level   | Code | Message                                                         |
+---------+------+-----------------------------------------------------------------+
| Warning | 1292 | Truncated incorrect query_cache_size value: '40000'             |
| Warning | 1282 | Query cache failed to set size 39936; new query cache size is 0 |
+---------+------+-----------------------------------------------------------------+
```

Examining the Query Cache

A number of status variables provide information about the query cache.

```
SHOW STATUS LIKE 'Qcache%';
+-------------------------+----------+
| Variable_name           | Value    |
+-------------------------+----------+
| Qcache_free_blocks      | 1158     |
| Qcache_free_memory      | 3760784  |
| Qcache_hits             | 31943398 |
| Qcache_inserts          | 42998029 |
| Qcache_lowmem_prunes    | 34695322 |
| Qcache_not_cached       | 652482   |
| Qcache_queries_in_cache | 4628     |
| Qcache_total_blocks     | 11123    |
+-------------------------+----------+
```

`Qcache_inserts` contains the number of queries added to the query cache, `Qcache_hits` contains the number of queries that have made use of the query cache, while `Qcache_lowmem_prunes` contains the number of queries that were dropped from the cache due to lack of memory.

The above example could indicate a poorly performing cache. More queries have been added, and more queries have been dropped, than have actually been used.

Results returned by the query cache count towards `Com_select` (see MDEV-4981).

The QUERY_CACHE_INFO plugin creates the QUERY_CACHE_INFO table in the INFORMATION_SCHEMA, allowing you to examine the contents of the query cache.

Query Cache Fragmentation

The Query Cache uses blocks of variable length, and over time may become fragmented. A high `Qcache_free_blocks` relative to `Qcache_total_blocks` may indicate fragmentation. FLUSH QUERY CACHE will defragment the query cache without dropping any queries :

```
FLUSH QUERY CACHE;
```

After this, there will only be one free block :

```
SHOW STATUS LIKE 'Qcache%';
+-------------------------+----------+
| Variable_name           | Value    |
+-------------------------+----------+
| Qcache_free_blocks      | 1        |
| Qcache_free_memory      | 6101576  |
| Qcache_hits             | 31981126 |
| Qcache_inserts          | 43002404 |
| Qcache_lowmem_prunes    | 34696486 |
| Qcache_not_cached       | 655607   |
| Qcache_queries_in_cache | 4197     |
| Qcache_total_blocks     | 8833     |
+-------------------------+----------+
```

Emptying and disabling the Query Cache

To empty or clear all results from the query cache, use RESET QUERY CACHE. FLUSH TABLES will have the same effect.

Setting either query_cache_type or query_cache_size to `0` will disable the query cache, but to free up the most resources, set both to `0` when you wish to disable caching.

Limitations

- The query cache needs to be disabled in order to use OQGRAPH.
- The query cache is not used by the Spider storage engine (amongst others).

LOCK TABLES and the Query Cache

The query cache can be used when tables have a write lock (which may seem confusing since write locks should avoid table reads). This behaviour can be changed by setting the query_cache_wlock_invalidate system variable to `ON`, in which case each write lock will invalidate the table query cache. Setting to `OFF`, the default, means that cached queries can be returned even when a table lock is being held. For example:

```
1> SELECT * FROM T1
+---+
| a |
+---+
| 1 |
+---+
-- Here the query is cached

-- From another connection execute:
2> LOCK TABLES T1 WRITE;

-- Expected result with: query_cache_wlock_invalidate = OFF
1> SELECT * FROM T1
+---+
| a |
+---+
| 1 |
+---+
-- read from query cache

-- Expected result with: query_cache_wlock_invalidate = ON
1> SELECT * FROM T1
-- Waiting Table Write Lock
```

Transactions and the Query Cache

The query cache handles transactions. Internally a flag (`FLAGS_IN_TRANS`) is set to `0` when a query was executed outside a transaction, and to 1 when the query was inside a transaction (begin / COMMIT / ROLLBACK). This flag is part of the “query cache hash”, in others words one query inside a transaction is different from a query outside a transaction.

Queries that change rows (INSERT / UPDATE / DELETE / TRUNCATE) inside a transaction will invalidate all queries from the table, and turn off the query cache to the changed table. Transactions that don’t end with `COMMIT` / `ROLLBACK` check that even without `COMMIT` / `ROLLBACK`, the query cache is turned off to allow row level locking and consistency level.

Examples:

```
SELECT * FROM T1 <first INSERT TO query cache, USING FLAGS_IN_TRANS=0>
+---+
| a |
+---+
| 1 |
+---+
```

```
BEGIN;
SELECT * FROM T1 <first INSERT TO query cache, USING FLAGS_IN_TRANS=1>
+---+
| a |
+---+
| 1 |
+---+
```

```
SELECT * FROM T1 <result FROM query cache, USING FLAGS_IN_TRANS=1>
+---+
| a |
+---+
| 1 |
+---+
```

```
INSERT INTO T1 VALUES(2);  <invalidate queries FROM TABLE T1 AND disable query cache TO TABLE T1>
```

```
SELECT * FROM T1 <don't USE query cache, a normal query FROM innodb TABLE>
+---+
| a |
+---+
| 1 |
| 2 |
+---+
```

```
SELECT * FROM T1 <don't USE query cache, a normal query FROM innodb TABLE>
+---+
| a |
+---+
| 1 |
| 2 |
+---+
```

```
COMMIT;  <query cache IS now turned ON TO T1 TABLE>
```

```
SELECT * FROM T1 <first INSERT TO query cache, USING FLAGS_IN_TRANS=0>
+---+
| a |
+---+
| 1 |
+---+
```

```
SELECT * FROM T1 <result FROM query cache, USING FLAGS_IN_TRANS=0>
+---+
| a |
+---+
| 1 |
+---+
```

Query Cache Internal Structure

Internally, each flag that can change a result using the same query is a different query. For example, using the latin1 charset and using the utf8 charset with the same query are treated as different queries by the query cache.

Some fields that differentiate queries are (from “Query_cache_query_flags” internal structure) :

- query (string)
- current database schema name (string)
- client long flag (0/1)

- client protocol 4.1 (0/1)
- protocol type (internal value)
- more results exists (protocol flag)
- in trans (inside transaction or not)
- autocommit (autocommit session variable)
- pkt_nr (protocol flag)
- character set client (character_set_client session variable)
- character set results (character_set_results session variable)
- collation connection (collation_connection session variable)
- limit (sql_select_limit session variable)
- time zone (time_zone session variable)
- sql_mode (sql_mode session variable)
- max_sort_length (max_sort_length session variable)
- group_concat_max_len (group_concat_max_len session variable)
- default_week_format (default_week_format session variable)
- div_precision_increment (div_precision_increment session variable)
- lc_time_names (lc_time_names session variable)

Timeout and Mutex Contention

When searching for a query inside the query cache, a `try_lock` function waits with a timeout of 50ms. If the lock fails, the query isn't executed via the query cache. This timeout is hard-coded (MDEV-6766 include two variables to tune this timeout).

From the sql_cache.cc, function "try_lock" using `TIMEOUT` :

```
struct timespec waittime;
        set_timespec_nsec(waittime,(ulong)(50000000L));  /* Wait for 50 msec */
        int res= mysql_cond_timedwait(&COND_cache_status_changed,
                                      &structure_guard_mutex, &waittime);
        if (res == ETIMEDOUT)
          break;
```

When inserting a query inside the query cache or aborting a query cache insert (using the KILL command for example), a try_lock function waits until the query cache returns; no timeout is used in this case.

When two processes execute the same query, only the last process stores the query result. All other processes increase the Qcache_not_cached status variable.

SQL_NO_CACHE and SQL_CACHE

There are two aspects to the query cache: placing a query in the cache, and retrieving it from the cache.

1. Adding a query to the query cache. This is done automatically for cacheable queries (see (Queries Stored in the Query Cache) when the query_cache_type system variable is set to `1`, or `ON` and the query contains no `SQL_NO_CACHE` clause, or when the query_cache_type system variable is set to `2`, or `DEMAND`, and the query contains the `SQL_CACHE` clause.
2. Retrieving a query from the cache. This is done after the server receives the query and before the query parser. In this case one point should be considered:

When using `SQL_NO_CACHE`, it should be after the first `SELECT` hint:

```
SELECT SQL_NO_CACHE .... FROM (SELECT SQL_CACHE ...) AS temp_table
```

Don’t use it like this:

```
SELECT SQL_CACHE .... FROM (SELECT SQL_NO_CACHE ...) AS temp_table
```

The second query will be checked. The query cache only checks if `SQL_NO_CACHE` / `SQL_CACHE` exists after the first `SELECT`. (More info at MDEV-6631)

This page is licensed: CC BY-SA / Gnu FDL

Thread Command Values

A thread can have any of the following `COMMAND` values (displayed by the `COMMAND` field listed by the SHOW PROCESSLIST statement or in the Information Schema PROCESSLIST Table, as well as the `PROCESSLIST_COMMAND` value listed in the Performance Schema threads Table). These indicate the nature of the thread’s activity.

Value	Description
Binlog Dump	Master thread for sending binary log contents to a slave.
Change user	Executing a change user operation.
Close stmt	Closing a prepared statement.
Connect	Replication slave is connected to its master.
Connect Out	Replication slave is in the process of connecting to its master.
Create DB	Executing an operation to create a database.
Daemon	Internal server thread rather than for servicing a client connection.
Debug	Generating debug information.
Delayed insert	A delayed-insert handler.
Drop DB	Executing an operation to drop a database.
Error	Error.
Execute	Executing a prepared statement.
Fetch	Fetching the results of an executed prepared statement.
Field List	Retrieving table column information.
Init DB	Selecting default database.
Kill	Killing another thread.
Long Data	Retrieving long data from the result of executing a prepared statement.
Ping	Handling a server ping request.
Prepare	Preparing a prepared statement.
Processlist	Preparing processlist information about server threads.
Query	Executing a statement.
Quit	In the process of terminating the thread.
Refresh	Flushing a table, logs or caches, or refreshing replication server or status variable information.
Register Slave	Registering a slave server.
Reset stmt	Resetting a prepared statement.
Set option	Setting or resetting a client statement execution option.
Sleep	Waiting for the client to send a new statement.
Shutdown	Shutting down the server.

Value	Description
Statistics	Preparing status information about the server.
Table Dump	Sending the contents of a table to a slave.
Time	Not used.

This page is licensed: CC BY-SA / Gnu FDL

Thread Pool

See Thread Pool in MariaDB.

Explains MariaDB's dynamic, adaptive thread pool: the problems it solves, its features, when it helps or hurts, and how to configure it via thread_handling and related system variables.

See Thread Groups in the Unix Implementation of the Thread Pool.

Describes thread groups in the Unix thread pool implementation, how thread_pool_size sets their number, and how client connections are distributed among them.

See Thread Pool System and Status Variables.

Documents the system and status variables used to configure and monitor the MariaDB thread pool, including extra_max_connections, extra_port, and thread_handling.

See Thread Pool in MariaDB 5.1 - 5.3.

Documents the old pool-of-threads implementation in MariaDB up to version 5.3, retained for reference by older compatibility pages.

This page is licensed: CC BY-SA / Gnu FDL

Thread Pool in MariaDB

Problems That Thread Pools Solve

The task of scalable server software (and a DBMS like MariaDB is an example of such software) is to maintain top performance with an increasing number of clients. MySQL traditionally assigned a thread for every client connection, and as the number of concurrent users grows this model shows performance drops. Many active threads are a performance killer, because increasing the number of threads leads to extensive context switching, bad locality for CPU caches, and increased contention for hot locks. An ideal solution that would help to reduce context switching is to maintain a lower number of threads than the number of clients. But this number should not be too low either, since we also want to utilize CPUs to their fullest, so ideally, there should be a single active thread for each CPU on the machine.

MariaDB Thread Pool Features

MariaDB has a dynamic and adaptive thread pool, aimed at optimizing resource utilization and preventing deadlocks.

For example, a thread may depend on another thread's completion, and they may block each other via locks and/or I/O. It is hard, and sometimes impossible, to predict how many threads are ideal or even sufficient to prevent deadlocks in every situation. MariaDB implements a dynamic and adaptive pool that takes care of creating new threads in times of high demand, and retiring threads if they have nothing to do. This is a complete reimplementation of the legacy `pool-of-threads` scheduler, with the following goals:

- Make the pool dynamic, so that it will grow and shrink whenever required.
- Minimize the amount of overhead that is required to maintain the thread pool itself.

- Make the best use of underlying OS capabilities. For example, if a native thread pool implementation is available, it should be used. If not, the best I/O multiplexing method should be used.
- Limit the resources used by threads.

There are currently two different low-level implementations – depending on OS. One implementation is designed specifically for Windows which utilizes a native CreateThreadpool API. The second implementation is primarily intended to be used in Unix-like systems. Because the implementations are different, some system variables differ between Windows and Unix.

When to Use the Thread Pool

Thread pools are most efficient in situations where queries are relatively short and the load is CPU-bound, such as in OLTP workloads. If the workload is not CPU-bound, then you might still benefit from limiting the number of threads to save memory for the database memory buffers.

When the Thread Pool is Less Efficient

There are special, rare cases where the thread pool is likely to be less efficient.

- If you have a very bursty workload, then the thread pool may not work well for you. These tend to be workloads in which there are long periods of inactivity followed by short periods of very high activity by many users. These also tend to be workloads in which delays cannot be tolerated, so the throttling of thread creation that the thread pool uses is not ideal. Even in this situation, performance can be improved by tweaking how often threads are retired. For example, with thread_pool_idle_timeout on Unix, or with thread_pool_min_threads on Windows.
- If you have many concurrent, long, non-yielding queries, then the thread pool may not work well for you. In this context, a “non-yielding” query is one that never waits or which does not indicate waits to the thread pool. These kinds of workloads are mostly used in data warehouse scenarios. Long-running, non-yielding queries will delay execution of other queries. However, the thread pool has stall detection to prevent them from totally monopolizing the thread pool. See Thread Groups in the Unix Implementation of the Thread Pool: Thread Group Stalls for more information. Even when the whole thread pool is blocked by non-yielding queries, you can still connect to the server through the extra-port TCP/IP port.
- If you rely on the fact that simple queries always finish quickly, no matter how loaded your database server is, then the thread pool may not work well for you. When the thread pool is enabled on a busy server, even simple queries might be queued to be executed later. This means that even if the statement itself doesn’t take much time to execute, even a simple `SELECT 1`, might take a bit longer when the thread pool is enabled than with `one-thread-per-connection` if it gets queued.

Configuring the Thread Pool

The thread_handling system variable is the primary system variable that is used to configure the thread pool.

There are several other system variables as well, which are described in the sections below. Many of the system variables documented below are dynamic, meaning that they can be changed with SET GLOBAL on a running server.

Generally, there is no need to tweak many of these system variables. The goal of the thread pool was to provide good performance out-of-the box. However, the system variable values can be changed, and we intended to expose as many knobs from the underlying implementation as we could. Feel free to tweak them as you see fit.

If you find any issues with any of the default behavior, then we encourage you to submit a bug report.

See Thread Pool System and Status Variables for the full list of the thread pool’s system variables.

Configuring the Thread Pool on Unix

On Unix, if you would like to use the thread pool, then you can use the thread pool by setting the thread_handling system variable to `pool-of-threads` in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
thread_handling=pool-of-threads
```

The following system variables can also be configured on Unix:

- thread_pool_size – The number of thread groups in the thread pool, which determines how many statements can execute simultaneously. The default value is the number of CPUs on the system. When setting this system variable's value at system startup, the max value is 100000. However, it is not a good idea to set it that high. When setting this system variable's value dynamically, the max value is either 128 or the value that was set at system startup–whichever value is higher. See Thread Groups in the Unix Implementation of the Thread Pool for more information.
- thread_pool_max_threads – The maximum number of threads in the thread pool. Once this limit is reached, no new threads will be created in most cases. In rare cases, the actual number of threads can slightly exceed this, because each thread group needs at least two threads (i.e. at least one worker thread and at least one listener thread) to prevent deadlocks. The default value is `65536`.
- thread_pool_stall_limit – The number of milliseconds between each stall check performed by the timer thread. The default value is `500`. Stall detection is used to prevent a single client connection from monopolizing a thread group. When the timer thread detects that a thread group is stalled, it wakes up a sleeping worker thread in the thread group, if one is available. If there isn't one, then it creates a new worker thread in the thread group. This temporarily allows several client connections in the thread group to run in parallel. However, note that the timer thread will not create a new worker thread if the number of threads in the thread pool is already greater than or equal to the maximum defined by the thread_pool_max_threads variable, unless the thread group does not already have a listener thread. See Thread Groups in the Unix Implementation of the Thread Pool: Thread Group Stalls for more information.
- thread_pool_oversubscribe – Determines how many worker threads in a thread group can remain active at the same time once a thread group is oversubscribed due to stalls. The default value is `3`. Usually, a thread group only has one active worker thread at a time. However, the timer thread can add more active worker threads to a thread group if it detects a stall. There are trade-offs to consider when deciding whether to allow only one thread per CPU to run at a time, or whether to allow more than one thread per CPU to run at a time. Allowing only one thread per CPU means that the thread can have unrestricted access to the CPU while its running, but it also means that there is additional overhead from putting threads to sleep or waking them up more frequently. Allowing more than one thread per CPU means that the threads have to share the CPU, but it also means that there is less overhead from putting threads to sleep or waking them up. This is primarily for internal use, and it is not meant to be changed for most users. See Thread Groups in the Unix Implementation of the Thread Pool: Thread Group Oversubscription for more information.
- thread_pool_idle_timeout – The number of seconds before an idle worker thread exits. The default value is `60`. If there is currently no work to do, how long should an idle thread wait before exiting?

Configuring the Thread Pool on Windows

The Windows implementation of the thread pool uses a native thread pool created with the CreateThreadpool API.

On Windows, if you would like to use the thread pool, then you do not need to do anything, because the default for the thread_handling system variable is already preset to `pool-of-threads`.

However, if you would like to use the old one thread per-connection behavior on Windows, then you can use that by setting the thread_handling system variable to `one-thread-per-connection` in a server option group in an option

file prior to starting up the server. For example:

```
[mariadb]
...
thread_handling=one-thread-per-connection
```

On older versions of Windows, such as XP and 2003, `pool-of-threads` is not implemented, and the server will silently switch to using the legacy `one-thread-per-connection` method.

The native CreateThreadpool API allows applications to set the minimum and maximum number of threads in the pool. The following system variables can be used to configure those values on Windows:

- thread_pool_min_threads – The minimum number of threads in the pool. Default is 1. This applicable in a special case of very "bursty" workloads. Imagine having longer periods of inactivity after periods of high activity. While the thread pool is idle, Windows would decide to retire pool threads (based on experimentation, this seems to happen after thread had been idle for 1 minute). Next time high load comes, it could take some milliseconds or seconds until the thread pool size stabilizes again to optimal value. To avoid thread retirement, one could set the parameter to a higher value.
- thread_pool_max_threads – The maximum number of threads in the pool. Threads are not created when this value is reached. The default is 1000. This parameter can be used to prevent the creation of new threads if the pool can have short periods where many or all clients are blocked (for example, with `FLUSH TABLES WITH READ LOCK`, high contention on row locks, or similar). New threads are created if a blocking situation occurs (such as after a throttling interval), but sometimes you want to cap the number of threads, if you're familiar with the application and need to, for example, save memory. If your application constantly pegs at 500 threads, it might be a strong indicator for high contention in the application, and the thread pool does not help much.

Configuring Priority Scheduling

It is possible to configure connection prioritization. The priority behavior is configured by the thread_pool_priority system variable.

By default, if thread_pool_priority is set to `auto`, then queries would be given a higher priority, in case the current connection is inside a transaction. This allows the running transaction to finish faster, and has the effect of lowering the number of transactions running in parallel. The default setting will generally improve throughput for transactional workloads. But it is also possible to explicitly set the priority for the current connection to either 'high' or 'low'.

There is also a mechanism in place to ensure that higher priority connections are not monopolizing the worker threads in the pool (which would cause indefinite delays for low priority connections). On Unix, low priority connections are put into the high priority queue after the timeout specified by the thread_pool_prio_kickup_timer system variable.

Configuring the Extra Port

MariaDB allows you to configure an extra port for administrative connections. This is primarily intended to be used in situations where all threads in the thread pool are blocked, and you still need a way to access the server. However, it can also be used to ensure that monitoring systems (including MaxScale's monitors) always have access to the system, even when all connections on the main port are used. This extra port uses the old `one-thread-per-connection` thread handling.

You can enable this and configure a specific port by setting the extra_port system variable.

You can configure a specific number of connections for this port by setting the extra_max_connections system variable.

These system variables can be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
extra_port = 8385
extra_max_connections = 10
```

Once you have the extra port configured, you can use the mariadb client with the `-P` option to connect to the port.

```
$ mariadb -u root -P 8385 -p
```

Monitoring Thread Pool Activity

Currently there are two status variables exposed to monitor pool activity.

Variable	Description
Threadpool_threads	Number of threads in the thread pool. In rare cases, this can be slightly higher than thread_pool_max_threads, because each thread group needs at least two threads (i.e. at least one worker thread and at least one listener thread) to prevent deadlocks.
Threadpool_idle_threads	Number of inactive threads in the thread pool. Threads become inactive for various reasons, such as by waiting for new work. However, an inactive thread is not necessarily one that has not been assigned work. Threads are also considered inactive if they are being blocked while waiting on disk I/O, or while waiting on a lock, etc. This status variable is only meaningful on Unix.

Thread Groups in the Unix Implementation of the Thread Pool

On Unix, the thread pool implementation uses objects called thread groups to divide up client connections into many independent sets of threads. See Thread Groups in the Unix Implementation of the Thread Pool for more information.

Fixing a Blocked Thread Pool

When using global locks, even with a high value on the thread_pool_max_threads system variable, it is still possible to block the entire pool.

Imagine the case where a client performs FLUSH TABLES WITH READ LOCK then pauses. If then the number of other clients connecting to the server to start write operations exceeds the maximum number of threads allowed in the pool, it can block the Server. This makes it impossible to issue the UNLOCK TABLES statement. It can also block MaxScale from monitoring the Server.

To mitigate the issue, MariaDB allows you to configure an extra port for administrative connections. See Configuring the Extra Port for information on how to configure this.

Once you have the extra port configured, you can use the mariadb client with the `-P` option to connect to the port.

```
$ mariadb -u root -P 8385 -p
```

This ensures that your administrators can access the server in cases where the number of threads is already equal to the configured value of the thread_pool_max_threads system variable, and all threads are blocked. It also ensures that MaxScale can still access the server in such situations for monitoring information.

Once you are connected to the extra port, you can solve the issue by increasing the value on the thread_pool_max_threads system variable, or by killing the offending connection, (that is, the connection that holds the global lock, which would be in the `sleep` state).

Information Schema

The following Information Schema tables relate to the thread pool:

- Information Schema THREAD_POOL_GROUPS Table
- Information Schema THREAD_POOL_QUEUES Table

- Information Schema THREAD_POOL_STATS Table
- Information Schema THREAD_POOL_WAITS Table

MariaDB Thread Pool vs Oracle MySQL Enterprise Thread Pool

Commercial editions of MySQL since 5.5 include an Oracle MySQL Enterprise thread pool implemented as a plugin, which delivers similar functionality. A detailed discussion about the design of the feature is at Mikael Ronstrom's blog. Here is the summary of similarities and differences, based on the above materials.

Similarities

- On Unix, both MariaDB and Oracle MySQL Enterprise Thread Pool will partition client connections into groups. The thread_pool_size parameter thus has the same meaning for both MySQL and MariaDB.
- Both implementations use similar schema checking for thread stalls, and both have the same parameter name for thread_pool_stall_limit (though in MariaDB it is measured using millisecond units, not 10ms units like in Oracle MySQL).

Differences

- The Windows implementation is completely different – MariaDB's uses native Windows thread pooling, while Oracle's implementation includes a convenience function WSAPoll() (provided for convenience to port Unix applications). As a consequence of relying on WSAPoll(), Oracle's implementation does not work with named pipes and shared memory connections.
- MariaDB uses the most efficient I/O multiplexing facilities for each operating system: Windows (the I/O completion port is used internally by the native thread pool), Linux (epoll), Solaris (event ports), FreeBSD and OSX (kevent). Oracle uses optimized I/O multiplexing only on Linux, with epoll, and uses poll() otherwise.
- Unlike Oracle MySQL Enterprise Thread Pool, MariaDB's one is built in, not a plugin.

MariaDB Thread Pool vs Percona Thread Pool

Percona's implementation is a port of the MariaDB's thread pool with some added features. In particular, Percona added priority scheduling to its 5.5-5.7 releases. MariaDB and Percona priority scheduling works in a similar fashion, but there are some differences in details.

- MariaDB's `thread_pool_priority=auto,high, low` correspond to Percona's `thread_pool_high_prio_mode=transactions,statements,none`.
- Percona has a `thread_pool_high_prio_tickets` connection variable to allow every nth low priority query to be put into the high priority queue. MariaDB does not have a corresponding setting.
- MariaDB has a `thread_pool_prio_kickup_timer` setting, which Percona does not have.

Running Benchmarks

When running `sysbench` and maybe other benchmarks, that create many threads on the same machine as the server, it is advisable to run benchmark driver and server on different CPUs to get the realistic results. Running lots of driver threads and only several server threads on the same CPUs will have the effect that OS scheduler will schedule benchmark driver threads to run with much higher probability than the server threads, that is driver will preempt the server. Use "taskset –c" on Linuxes, and "set /affinity" on Windows to separate benchmark driver and server CPUs, as the preferred method to fix this situation.

A possible alternative on Unix (if taskset or a separate machine running the benchmark is not desired for some reason) would be to increase thread_pool_size to make the server threads more "competitive" against the client threads.

When running sysbench, a good rule of thumb could be to give 1/4 of all CPUs to the sysbench, and 3/4 of CPUs to mariadbd. It is also good idea to run sysbench and mariadbd on different NUMA nodes, if possible.

Notes

The thread_cache_size system variable is not used when the thread pool is used and the Threads_cached status variable will have a value of 0.

See Also

- Thread Pool System and Status Variables
- Threadpool Benchmarks

This page is licensed: CC BY-SA / Gnu FDL

Thread Groups in the Unix Implementation of the Thread Pool

This article does not apply to the thread pool implementation on Windows. On Windows, MariaDB uses a native thread pool created with the CreateThreadpool API, which has its own methods to distribute threads between CPUs.

On Unix, the thread pool implementation uses objects called thread groups to divide up client connections into many independent sets of threads. The thread_pool_size system variable defines the number of thread groups on a system. Generally speaking, the goal of the thread group implementation is to have one running thread on each CPU on the system at a time. Therefore, the default value of the thread_pool_size system variable is auto-sized to the number of CPUs on the system.

When setting the thread_pool_size system variable's value at system startup, the max value is `100000`. However, it is not a good idea to set it that high. When setting its value dynamically, the max value is either `128` or the value that was set at system startup–whichever value is higher. It can be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL thread_pool_size=32;
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
..
thread_handling=pool-of-threads
thread_pool_size=32
```

If you do not want MariaDB to use all CPUs on the system for some reason, then you can set it to a lower value than the number of CPUs. For example, this would make sense if the MariaDB Server process is limited to certain CPUs with the taskset utility on Linux.

If you set the value to the number of CPUs and if you find that the CPUs are still underutilized, then try increasing the value.

The thread_pool_size system variable tends to have the most visible performance effect. It is roughly equivalent to the number of threads that can run at the same time. In this case, run means use CPU, rather than sleep or wait. If a client connection needs to sleep or wait for some reason, then it wakes up another client connection in the thread group before it does so.

One reason that CPU underutilization may occur in rare cases is that the thread pool is not always informed when a thread is going to wait. For example, some waits, such as a page fault or a miss in the OS buffer cache, cannot be detected by MariaDB.

Distributing Client Connections Between Thread Groups

When a new client connection is created, its thread group is determined using the following calculation:

```
thread_group_id = connection_id % thread_pool_size
```

The `connection_id` value in the above calculation is the same monotonically increasing number that you can use to identify connections in SHOW PROCESSLIST output or the information_schema.PROCESSLIST table.

This calculation should assign client connections to each thread group in a round-robin manner. In general, this should result in an even distribution of client connections among thread groups.

Types of Threads

Thread Group Threads

Thread groups have two different kinds of threads: a **listener thread** and **worker threads**.

- A thread group's worker threads actually perform work on behalf of client connections. A thread group can have many worker threads, but usually, only one will be actively running at a time. This is not always the case. For example, the thread group can become oversubscribed if the thread pool's timer thread detects that the thread group is stalled. This is explained more in the sections below.
- A thread group's listener thread listens for I/O events and distributes work to the worker threads. If it detects that there is a request that needs to be worked on, then it can wake up a sleeping worker thread in the thread group, if any exist. If the listener thread is the only thread in the thread group, then it can also create a new worker thread. If there is only one request to handle, and if the thread_pool_dedicated_listener system variable is not enabled, then the listener thread can also become a worker thread and handle the request itself. This helps decrease the overhead that may be introduced by excessively waking up sleeping worker threads and excessively creating new worker threads.

Global Threads

The thread pool has one global thread: a **timer thread**. The **timer thread** performs tasks, such as:

- Checks each thread group for stalls.
- Ensures that each thread group has a listener thread.

Thread Creation

A new thread is created in a thread group in the scenarios listed below.

In all of the scenarios below, the thread pool implementation prefers to wake up a sleeping **worker thread** that already exists in the thread group, rather than to create a new thread.

Worker Thread Creation by Listener Thread

A thread group's **listener thread** can create a new **worker thread** when it has more client connection requests to distribute, but no pre-existing **worker threads** are available to work on the requests. This can help to ensure that the thread group always has enough threads to keep one **worker thread** active at a time.

A thread group's **listener thread** creates a new **worker thread** if all of the following conditions are met:

- The listener thread receives a client connection request that needs to be worked on.
- There are more client connection requests in the thread group's work queue that the listener thread still needs to distribute to worker threads, so the listener thread should not become a worker thread.
- There are no active worker threads in the thread group.
- There are no sleeping worker threads in the thread group that the listener thread can wake up.
- And one of the following conditions is also met:
  - The entire thread pool has fewer than thread_pool_max_threads.
  - There are fewer than two threads in the thread group. This is to guarantee that each thread group can have at least two threads, even if thread_pool_max_threads has already been reached or exceeded.

Thread Creation by Worker Threads During Waits

A thread group's **worker thread** can create a new **worker thread** when the thread has to wait on something, and the thread group has more client connection requests queued, but no pre-existing **worker threads** are available to work on them. This can help to ensure that the thread group always has enough threads to keep one **worker thread** active at a time. For most workloads, this tends to be the primary mechanism that creates new **worker threads**.

A thread group's **worker thread** creates a new thread if all of the following conditions are met:

- The worker thread has to wait on some request. For example, it might be waiting on disk I/O, or it might be waiting on a lock, or it might just be waiting for a query that called the SLEEP() function to finish.
- There are no active worker threads in the thread group.
- There are no sleeping worker threads in the thread group that the worker thread can wake up.
- And one of the following conditions is also met:
  - The entire thread pool has fewer than thread_pool_max_threads.
  - There are fewer than two threads in the thread group. This is to guarantee that each thread group can have at least two threads, even if thread_pool_max_threads has already been reached or exceeded.
- And one of the following conditions is also met:
  - There are more client connection requests in the thread group's work queue that the listener thread still needs to distribute to worker threads. In this case, the new thread is intended to be a worker thread.
  - There is currently no listener thread in the thread group. For example, if the thread_pool_dedicated_listener system variable is not enabled, then the thread group's listener thread can became a worker thread, so that it could handle some client connection request. In this case, the new thread can become the thread group's listener thread.

Listener Thread Creation by Timer Thread

The thread pool's **timer thread** can create a new **listener thread** for a thread group when the thread group has more client connection requests that need to be distributed, but the thread group does not currently have a **listener thread** to distribute them. This can help to ensure that the thread group does not miss client connection requests because it has no **listener thread**.

The thread pool's **timer thread** creates a new **listener thread** for a thread group if all of the following conditions are met:

- The thread group has not handled any I/O events since the last check by the timer thread.
- There is currently no listener thread in the thread group. For example, if the thread_pool_dedicated_listener system variable is not enabled, then the thread group's listener thread can became a worker thread, so that it could handle some client connection request. In this case, the new thread can become the thread group's listener thread.
- There are no sleeping worker threads in the thread group that the timer thread can wake up.
- And one of the following conditions is also met:
  - The entire thread pool has fewer than thread_pool_max_threads.
  - There are fewer than two threads in the thread group. This is to guarantee that each thread group can have at least two threads, even if thread_pool_max_threads has already been reached or exceeded.
- If the thread group already has active worker threads, then the following condition also needs to be met:
  - A worker thread has not been created for the thread group within the throttling interval.

Worker Thread Creation by Timer Thread during Stalls

The thread pool's **timer thread** can create a new **worker thread** for a thread group when the thread group is stalled. This can help to ensure that a long query can't monopole its thread group.

The thread pool's **timer thread** creates a new **worker thread** for a thread group if all of the following conditions are met:

- The timer thread thinks that the thread group is stalled. This means that the following conditions have been met:
  - There are more client connection requests in the thread group's work queue that the listener thread still needs to distribute to worker threads.
  - No client connection requests have been allowed to be dequeued to run since the last stall check by the timer thread.
- There are no sleeping worker threads in the thread group that the timer thread can wake up.
- And one of the following conditions is also met:
  - The entire thread pool has fewer than thread_pool_max_threads.
  - There are fewer than two threads in the thread group. This is to guarantee that each thread group can have at least two threads, even if thread_pool_max_threads has already been reached or exceeded.
- A worker thread has not been created for the thread group within the throttling interval.

Thread Creation Throttling

In some of the scenarios listed above, a thread is only created within a thread group if no new threads have been created for the thread group within the *throttling interval*. The throttling interval depends on the number of threads that are already in the thread group.

In MariaDB 10.5 and later, thread creation is not throttled until a thread group has more than 1 + thread_pool_oversubscribe threads:

Number of Threads in Thread Group	Throttling Interval (milliseconds)
0-(1 + thread_pool_oversubscribe)	0
4-7	50 * `THROTTLING_FACTOR`
8-15	100 * `THROTTLING_FACTOR`
16-65536	20 * `THROTTLING_FACTOR`

The throttling factor is calculated like this (see thread_pool_stall_limit for more information):

```
THROTTLING_FACTOR = thread_pool_stall_limit / MAX (500,thread_pool_stall_limit)
```

Thread Group Stalls

The thread pool has a feature that allows it to detect if a client connection is executing a long-running query that may be monopolizing its thread group. If a client connection were to monopolize its thread group, then that could prevent other client connections in the thread group from running their queries. In other words, the thread group would appear to be *stalled*.

This stall detection feature is implemented by creating a **timer thread** that periodically checks if any of the thread groups are stalled. There is only a single **timer thread** for the entire thread pool. The thread_pool_stall_limit system variable defines the number of milliseconds between each stall check performed by the timer thread. The default value is `500`. It can be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL thread_pool_stall_limit=300;
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
..
thread_handling=pool-of-threads
thread_pool_size=32
thread_pool_stall_limit=300
```

The **timer thread** considers a thread group to be stalled if the following is true:

- There are more client connection requests in the thread group's work queue that the listener thread still needs to distribute to worker threads.
- No client connection requests have been allowed to be dequeued to run since the last stall check by the timer thread.

This indicates that the one or more client connections currently using the active **worker threads** may be monopolizing the thread group, and preventing the queued client connections from performing work. When the **timer thread** detects that a thread group is stalled, it wakes up a sleeping **worker thread** in the thread group, if one is available. If there isn't one, then it creates a new **worker thread** in the thread group. This temporarily allows several client connections in the thread group to run in parallel.

The thread_pool_stall_limit system variable essentially defines the limit for what a "fast query" is. If a query takes longer than thread_pool_stall_limit, then the thread pool is likely to think that it is too slow, and it will either wake up a sleeping worker thread or create a new worker thread to let another client connection in the thread group run a query in parallel.

In general, changing the value of the thread_pool_stall_limit system variable has the following effect:

- Setting it to higher values can help avoid starting too many parallel threads if you expect a lot of client connections to execute long-running queries.
- Setting it to lower values can help prevent deadlocks.

Thread Group Oversubscription

If the **timer thread** were to detect a stall in a thread group, then it would either wake up a sleeping **worker thread** or create a new **worker thread** in that thread group. At that point, the thread group would have multiple active **worker threads**. In other words, the thread group would be *oversubscribed*.

You might expect that the thread pool would shutdown one of the **worker threads** when the stalled client connection finished what it was doing, so that the thread group would only have one active **worker thread** again. However, this does not always happen. Once a thread group is oversubscribed, the thread_pool_oversubscribe system variable defines the upper limit for when **worker threads** start shutting down after they finish work for client connections. The default value is `3`. It can be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL thread_pool_oversubscribe=10;
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
..
thread_handling=pool-of-threads
thread_pool_size=32
thread_pool_stall_limit=300
thread_pool_oversubscribe=10
```

To clarify, the thread_pool_oversubscribe system variable does not play any part in the creation of new **worker threads**. The thread_pool_oversubscribe system variable is only used to determine how many **worker threads** should remain active in a thread group, once a thread group is already oversubscribed due to stalls.

In general, the default value of `3` should be adequate for most users. Most users should not need to change the value of the thread_pool_oversubscribe system variable.

This page is licensed: CC BY-SA / Gnu FDL

Thread Pool System and Status Variables

This article describes the system and status variables used by the MariaDB thread pool. For a full description, see Thread Pool in MariaDB.

System variables

`extra_max_connections`

- Description: The number of connections on the extra_port.
  - See Thread Pool in MariaDB: Configuring the Extra Port for more information.
- Command line: `--extra-max-connections=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `100000`

`extra_port`

- Description: Extra port number to use for TCP connections in a `one-thread-per-connection` manner. If set to `0`, then no extra port is used.
  - See Thread Pool in MariaDB: Configuring the Extra Port for more information.
- Command line: `--extra-port=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`

`thread_handling`

- Description: Determines how the server handles threads for client connections. In addition to threads for client connections, this also applies to certain internal server threads, such as Galera slave threads. On Windows, if you would like to use the thread pool, then you do not need to do anything, because the default for the thread_handling system variable is already preset to `pool-of-threads`.
  - When the default `one-thread-per-connection` mode is enabled, the server uses one thread to handle each client connection.
  - When the `pool-of-threads` mode is enabled, the server uses the thread pool for client connections.
  - When the `no-threads` mode is enabled, the server uses a single thread for all client connections, which is really only usable for debugging.
- Command line: `--thread-handling=name`
- Scope: Global
- Dynamic: No
- Data Type: `enumeration`
- Default Value: `one-thread-per-connection` (non-Windows), `pool-of-threads` (Windows)

- Valid Values: `no-threads`, `one-thread-per-connection`, `pool-of-threads`.
- Documentation: Using the thread pool.
- Notes: In MySQL, the thread pool is only available in MySQL Enterprise. In MariaDB it's available in all versions.

`thread_pool_dedicated_listener`

- Description: If set to 1, then each group will have its own dedicated listener, and the listener thread will not pick up work items. As a result, the queueing time in the Information Schema Threadpool_Queues and the actual queue size in the Information Schema Threadpool_Groups table will be more exact, since IO requests are immediately dequeued from poll, without delay.
  - This system variable is only meaningful on Unix.
- Command line: `thread-pool-dedicated-listener={0|1}`
- Scope:
- Dynamic:
- Data Type: `boolean`
- Default Value: `0`
- Introduced: MariaDB 10.5.0

`thread_pool_exact_stats`

- Description: If set to 1, provides better queueing time statistics by using a high precision timestamp, at a small performance cost, for the time when the connection was added to the queue. This timestamp helps calculate the queuing time shown in the Information Schema Threadpool_Queues table.
  - This system variable is only meaningful on Unix.
- Command line: `thread-pool-exact-stats={0|1}`
- Scope:
- Dynamic:
- Data Type: `boolean`
- Default Value: `0`
- Introduced: MariaDB 10.5.0

`thread_pool_idle_timeout`

- Description: The number of seconds before an idle worker thread exits. The default value is `60`. If there is currently no work to do, how long should an idle thread wait before exiting?
  - This system variable is only meaningful on Unix.
  - The thread_pool_min_threads system variable is comparable for Windows.
- Command line: `thread-pool-idle-timeout=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `60`
- Documentation: Using the thread pool.

`thread_pool_max_threads`

- Description: The maximum number of threads in the thread pool. Once this limit is reached, no new threads will be created in most cases.

    - On Unix, in rare cases, the actual number of threads can slightly exceed this, because each thread group needs at least two threads (i.e. at least one worker thread and at least one listener thread) to prevent deadlocks.

- Scope:
- Command line: `thread-pool-max-threads=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `65536`
- Range: `1` to `65536`
- Documentation: Using the thread pool.

`thread_pool_min_threads`

- Description: Minimum number of threads in the thread pool. In bursty environments, after a period of inactivity, threads would normally be retired. When the next burst arrives, it would take time to reach the optimal level. Setting this value higher than the default would prevent thread retirement even if inactive.
    - This system variable is only meaningful on Windows.
    - The thread_pool_idle_timeout system variable is comparable for Unix.
- Command line: `thread-pool-min-threads=#`
- Data Type: `numeric`
- Default Value: `1`
- Documentation: Using the thread pool.

`thread_pool_oversubscribe`

- Description: Determines how many worker threads in a thread group can remain active at the same time once a thread group is oversubscribed due to stalls. The default value is `3`. Usually, a thread group only has one active worker thread at a time. However, the timer thread can add more active worker threads to a thread group if it detects a stall. There are trade-offs to consider when deciding whether to allow only one thread per CPU to run at a time, or whether to allow more than one thread per CPU to run at a time. Allowing only one thread per CPU means that the thread can have unrestricted access to the CPU while its running, but it also means that there is additional overhead from putting threads to sleep or waking them up more frequently. Allowing more than one thread per CPU means that the threads have to share the CPU, but it also means that there is less overhead from putting threads to sleep or waking them up.
    - See Thread Groups in the Unix Implementation of the Thread Pool: Thread Group Oversubscription for more information.
    - This is primarily for internal use, and it is not meant to be changed for most users.
    - This system variable is only meaningful on Unix.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `3`
- Range: `1` to `65536`
- Documentation: Using the thread pool.

`thread_pool_prio_kickup_timer`

- Description: Time in milliseconds before a dequeued low-priority statement is moved to the high-priority queue.

    - This system variable is only meaningful on Unix.
- Command line: `thread-pool-kickup-timer=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1000`
- Range: `0` to `4294967295`
- Introduced: MariaDB 10.2.2
- Documentation: Using the thread pool.

`thread_pool_priority`

- Description: Thread pool priority. High-priority connections usually start executing earlier than low-priority. If set to 'auto' (the default), the actual priority (low or high) is determined by whether or not the connection is inside a transaction.
- Command line: `--thread-pool-priority=#`
- Scope: Global,Connection
- Data Type: `enum`
- Default Value: `auto`
- Valid Values: `high`, `low`, `auto`.
- Introduced: MariaDB 10.2.2
- Documentation: Using the thread pool.

`thread_pool_size`

- Description: The number of thread groups in the thread pool, which determines how many statements can execute simultaneously. The default value is the number of CPUs on the system. When setting this system variable's value at system startup, the max value is 100000. However, it is not a good idea to set it that high. When setting this system variable's value dynamically, the max value is either 128 or the value that was set at system startup–whichever value is higher.
    - See Thread Groups in the Unix Implementation of the Thread Pool for more information.
    - This system variable is only meaningful on Unix.
- Command line: `--thread-pool-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: Based on the number of processors (but see MDEV-7806).
- Range: `1` to `128`
- Documentation: Using the thread pool.

`thread_pool_stall_limit`

- Description: The number of milliseconds between each stall check performed by the timer thread. The default value is `500`. Stall detection is used to prevent a single client connection from monopolizing a thread group. When the timer thread detects that a thread group is stalled, it wakes up a sleeping worker thread in the thread group, if one is available. If there isn't one, then it creates a new worker thread in the thread group. This temporarily allows several client connections in the thread group to run in parallel. However, note that the timer thread will not create a new worker thread if the number of threads in the thread pool is already greater than or

equal to the maximum defined by the thread_pool_max_threads variable, unless the thread group does not already have a listener thread.
    - See Thread Groups in the Unix Implementation of the Thread Pool: Thread Group Stalls for more information.
    - This system variable is only meaningful on Unix.
    - Note that if you are migrating from the MySQL Enterprise thread pool plugin, then the unit used in their implementation is 10ms, not 1ms.

- Command line: `--thread-pool-stall-limit=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `500`
- Range: `1` to `4294967295`
- Documentation: Using the thread pool.

Status variables

`Threadpool_idle_threads`

- Description: Number of inactive threads in the thread pool. Threads become inactive for various reasons, such as by waiting for new work. However, an inactive thread is not necessarily one that has not been assigned work. Threads are also considered inactive if they are being blocked while waiting on disk I/O, or while waiting on a lock, etc.
    - This status variable is only meaningful on Unix.
- Scope: Global, Session
- Data Type: `numeric`

`Threadpool_threads`

- Description: Number of threads in the thread pool. In rare cases, this can be slightly higher than thread_pool_max_threads, because each thread group needs at least two threads (i.e. at least one worker thread and at least one listener thread) to prevent deadlocks.
- Scope: Global, Session
- Data Type: `numeric`

See Also

- Thread Pool in MariaDB

This page is licensed: CC BY-SA / Gnu FDL

Thread Pool in MariaDB 5.1 - 5.3

> **Note:**
>
> This page describes the old thread pool implementation in MariaDB up to version 5.3.
>
> It's left here because some compatibility pages in the release notes refer to it.
>
> For the current implementation, refer to the Thread Pool in MariaDB page.

About Pool of Threads

This is an extended version of the pool-of-threads code from MySQL 6.0. This allows you to use a limited set of threads to handle all queries, instead of the old 'one-thread-per-connection' style. In recent times, its also been referred to as "thread pool" or "thread pooling" as this feature (in a different implementation) is available in Enterprise editions of MySQL (not in the Community edition).

This can be a very big win if most of your queries are short running queries and there are few table/row locks in your system.

Instructions

To enable pool-of-threads you must first run configure with the `--with-libevent` option. (This is automatically done if you use any 'max' scripts in the BUILD directory):

```
./configure --with-libevent
```

When starting mysqld with the pool of threads code you should use:

```
mysqld --thread-handling=pool-of-threads --thread-pool-size=20
```

Default values are:

```
thread-handling=  one-thread-per-connection
thread-pool-size= 20
```

One issue with pool-of-threads is that if all worker threads are doing work (like running long queries) or are locked by a row/table lock no new connections can be established and you can't login and find out what's wrong or login and kill queries.

To help this, we have introduced two new options for mysqld; extra_port and extra_max_connections:

```
--extra-port=#             (Default 0)
--extra-max-connections=#  (Default 1)
```

If extra-port is <> 0 then you can connect max_connections number of normal threads + 1 extra SUPER user through the 'extra-port' TCP/IP port. These connections use the old one-thread-per-connection method.

To connect with through the extra port, use:

```
mysql --port='number-of-extra-port' --protocol=tcp
```

This allows you to freely use, on connection bases, the optimal connection/thread model.

See also

- Thread-handling and thread-pool-size variables
- How MySQL Uses Threads for Client Connections

This page is licensed: CC BY-SA / Gnu FDL

Thread States

See Delayed Insert Connection Thread States.

Lists the thread states for the connection thread that processes INSERT DELAYED statements, as shown by SHOW PROCESSLIST and related tables.

See Delayed Insert Handler Thread States.

Lists the thread states for the handler thread that inserts the results of INSERT DELAYED statements, as shown by SHOW PROCESSLIST and related tables.

See Event Scheduler Thread States.

Lists the thread states related to event scheduling and execution, as shown by SHOW PROCESSLIST and related tables.

See Galera Cluster Thread States.

Lists the Galera Cluster-specific thread states shown in SHOW PROCESSLIST during cluster operations, such as certification, flow control, and Total Order Isolation DDL waits.

See General Thread States.

Documents the major general thread states shown by SHOW PROCESSLIST and related tables, with links to the more specific state lists for replication, the query cache, and other subsystems.

See Master Thread States.

Lists the thread states related to replication primary (master) threads, as shown by SHOW PROCESSLIST and related tables.

See Query Cache Thread States.

Lists the thread states related to the query cache, as shown by SHOW PROCESSLIST and related tables.

See Replica I/O Thread States.

Lists the replica I/O thread states shown by Slave_IO_State in SHOW REPLICA STATUS and by SHOW PROCESSLIST and related tables.

See Replica Connection Thread States.

Lists the thread states for connection threads on a replication replica, as shown by SHOW PROCESSLIST and related tables.

See Replica SQL Thread States.

Lists the replica SQL thread states shown by Slave_SQL_State in SHOW REPLICA STATUS and by SHOW PROCESSLIST and related tables, including parallel replication states.

This page is licensed: CC BY-SA / Gnu FDL

Delayed Insert Connection Thread States

This article documents thread states that are related to the connection thread that processes INSERT DELAYED statements.

These correspond to the `STATE` values listed by the SHOW PROCESSLIST statement or in the Information Schema PROCESSLIST Table as well as the `PROCESSLIST_STATE` value listed in the Performance Schema threads Table.

Value	Description
allocating local table	Preparing to allocate rows to the delayed-insert handler thread. Follows from the got handler lock state.
Creating delayed handler	Creating a handler for the delayed-inserts.
got handler lock	Lock to access the delayed-insert handler thread has been received. Follows from the waiting for handler lock state and before the allocating local table state.
got old table	The initialization phase is over. Follows from the waiting for handler open state.

Value	Description
storing row into queue	Adding new row to the list of rows to be inserted by the delayed-insert handler thread.
waiting for delay_list	Initializing (trying to find the delayed-insert handler thread).
waiting for handler insert	Waiting for new inserts, as all inserts have been processed.
waiting for handler lock	Waiting for delayed insert-handler lock to access the delayed-insert handler thread.
waiting for handler open	Waiting for the delayed-insert handler thread to initialize. Follows from the Creating delayed handler state and before the got old table state.

This page is licensed: CC BY-SA / Gnu FDL

Delayed Insert Handler Thread States

This article documents thread states that are related to the handler thread that inserts the results of INSERT DELAYED statements.

These correspond to the `STATE` values listed by the SHOW PROCESSLIST statement or in the Information Schema PROCESSLIST Table as well as the `PROCESSLIST_STATE` value listed in the Performance Schema threads Table.

Value	Description
insert	About to insert rows into the table.
reschedule	Sleeping in order to let other threads function, after inserting a number of rows into the table.
upgrading lock	Attempting to get lock on the table in order to insert rows.
Waiting for INSERT	Waiting for the delayed-insert connection thread to add rows to the queue.

This page is licensed: CC BY-SA / Gnu FDL

Event Scheduler Thread States

This article documents thread states that are related to event scheduling and execution. These include the Event Scheduler thread, threads that terminate the Event Scheduler, and threads for executing events.

These correspond to the `STATE` values listed by the SHOW PROCESSLIST statement or in the Information Schema PROCESSLIST Table as well as the `PROCESSLIST_STATE` value listed in the Performance Schema threads Table

Value	Description
Clearing	Thread is terminating.
Initialized	Thread has be initialized.
Waiting for next activation	The event queue contains items, but the next activation is at some time in the future.
Waiting for scheduler to stop	Waiting for the event scheduler to stop after issuing SET GLOBAL event_scheduler=OFF.
Waiting on empty queue	Sleeping, as the event scheduler's queue is empty.

This page is licensed: CC BY-SA / Gnu FDL

Galera Cluster Thread States

This article documents thread states that are specific to MariaDB Enterprise Cluster and appear in the SHOW PROCESSLIST when nodes are performing cluster-specific operations. These states enable monitoring of session status within clustered environments.

The following table lists the thread states present in Galera Cluster and their descriptions:

Thread State	Description
waiting to execute in isolation	The thread is waiting to execute a transaction in isolation mode.
waiting for TOI DDL	The thread is waiting for a Total Order Isolation Data Definition Language (DDL) operation to complete.
waiting for flow control	The thread is paused due to Galera flow control, which regulates replication throughput.
waiting for certification	The transaction is waiting for Galera transaction certification before it can be applied to avoid conflict with other transactions.

These states contribute in the identification of synchronization delays and performance bottlenecks in a Galera Cluster framework.

> These states were introduced in MariaDB Enterprise Server 11.4 and are also available in MariaDB Community Server 11.0.3 and later.

See Also

- SHOW PROCESSLIST
- Galera Cluster Flow Control
- MariaDB Galera Cluster Overview
- What's New in MariaDB Enterprise Server 11.4

This page is licensed: CC BY-SA / Gnu FDL

General Thread States

This article documents the major general thread states. More specific lists related to delayed inserts, replication, the query cache and the event scheduler are listed in:

- Event Scheduler Thread States
- Query Cache Thread States
- Master Thread States
- Slave Connection Thread States
- Slave I/O Thread States
- Slave SQL Thread States

These correspond to the `STATE` values listed by the SHOW PROCESSLIST statement or in the Information Schema PROCESSLIST Table as well as the `PROCESSLIST_STATE` value listed in the Performance Schema threads Table

Value	Description
After create	The function that created (or tried to create) a table (temporary or non-temporary) has just ended.
Analyzing	Calculating table key distributions, such as when running an ANALYZE TABLE statement.
checking permissions	Checking to see whether the permissions are adequate to perform the statement.
Checking table	Checking the table.
cleaning up	Preparing to reset state variables and free memory after executing a command.

Value	Description
closing tables	Flushing the changes to disk and closing the table. This state will only persist if the disk is full or under extremely high load.
converting HEAP to Aria	Converting an internal MEMORY temporary table into an on-disk Aria temporary table.
converting HEAP to MyISAM	Converting an internal MEMORY temporary table into an on-disk MyISAM temporary table.
copy to tmp table	A new table has been created as part of an ALTER TABLE statement, and rows are about to be copied into it.
Copying to group table	Sorting the rows by group and copying to a temporary table, which occurs when a statement has different GROUP BY and ORDER BY criteria.
Copying to tmp table	Copying to a temporary table in memory.
Copying to tmp table on disk	Copying to a temporary table on disk, as the resultset is too large to fit into memory.
Creating index	Processing an ALTER TABLE … ENABLE KEYS for an Aria or MyISAM table.
Creating sort index	Processing a SELECT statement resolved using an internal temporary table.
creating table	Creating a table (temporary or non-temporary).
Creating tmp table	Creating a temporary table (in memory or on-disk).
deleting from main table	Deleting from the first table in a multi-table delete, saving columns and offsets for use in deleting from the other tables.
deleting from reference tables	Deleting matched rows from secondary reference tables as part of a multi-table delete.
discard_or_import_tablespace	Processing an ALTER TABLE … IMPORT TABLESPACE or ALTER TABLE … DISCARD TABLESPACE statement.
end	State before the final cleanup of an ALTER TABLE, CREATE VIEW, DELETE, INSERT, SELECT, or UPDATE statement.
executing	Executing a statement.
Execution of init_command	Executing statements specified by the –init_command mariadb client option.

Value	Description
filling schema table	A table in the information_schema database is being built.
freeing items	Freeing items from the query cache after executing a command. Usually followed by the cleaning up state.
Flushing tables	Executing a FLUSH TABLES statement and waiting for other threads to close their tables.
FULLTEXT initialization	Preparing to run a full-text search. This includes running the fulltext search (MATCH … AGAINST) and creating a list of the result in memory
init	About to initialize an ALTER TABLE, DELETE, INSERT, SELECT, or UPDATE statement. Could be performaing query cache cleanup, or flushing the binary log or InnoDB log.
Killed	Thread will abort next time it checks the kill flag. Requires waiting for any locks to be released.
Locked	Query has been locked by another query.
logging slow query	Writing statement to the slow query log.
NULL	State used for SHOW PROCESSLIST.
login	Connection thread has not yet been authenticated.
manage keys	Enabling or disabling a table index.
Opening table[s]	Trying to open a table. Usually very quick unless the limit set by table_open_cache has been reached, or an ALTER TABLE or LOCK TABLE is in progress.
optimizing	Server is performing initial optimizations in for a query.
preparing	State occurring during query optimization.
Purging old relay logs	Relay logs that are no longer needed are being removed.
query end	Query has finished being processed, but items have not yet been freed (the freeing items state.
Reading file	Server is reading the file (for example during LOAD DATA INFILE).
Reading from net	Server is reading a network packet.
Removing duplicates	Duplicated rows being removed before sending to the client. This happens when SELECT DISTINCT is used in a way that the distinct operation could not be optimized at an earlier point.
removing tmp table	Removing an internal temporary table after processing a SELECT statement.
rename	Renaming a table.
rename result table	Renaming a table that results from an ALTER TABLE statement having created a new table.

Value	Description
Reopen tables	Table is being re-opened after thread obtained a lock but the underlying table structure had changed, so the lock was released.
Repair by sorting	Indexes are being created with the use of a sort. Much faster than the related Repair with keycache.
Repair done	Multi-threaded repair has been completed.
Repair with keycache	Indexes are being created through the key cache, one-by-one. Much slower than the related Repair by sorting.
Rolling back	A transaction is being rolled back.
Saving state	New table state is being saved. For example, after, analyzing a MyISAM table, the key distributions, rowcount etc. are saved to the .MYI file.
Searching rows for update	Finding matching rows before performing an UPDATE, which is needed when the UPDATE would change the index used for the UPDATE
Sending data	Sending data to the client as part of processing a SELECT statement or other statements that returns data like INSERT … RETURNING . Often the longest-occurring state as it also include all reading from tables and disk read activities. Where an aggregation or un-indexed filtering occurs there is significantly more rows read than what is sent to the client.
setup	Setting up an ALTER TABLE operation.
Sorting for group	Sorting as part of a GROUP BY
Sorting for order	Sorting as part of an ORDER BY
Sorting index	Sorting index pages as part of a table optimization operation.
Sorting result	Processing a SELECT statement using a non-temporary table.
statistics	Calculating statistics as part of deciding on a query execution plan. Usually a brief state unless the server is disk-bound.
System lock	Requesting or waiting for an external lock for a specific table. The storage engine determines what kind of external lock to use. For example, the MyISAM storage engine uses file-based locks. However, MyISAM's external locks are disabled by default, due to the default value of the skip_external_locking system variable. Transactional storage engines such as InnoDB also register the transaction or statement with MariaDB's transaction coordinator while in this thread state. See MDEV-19391 for more information about that.
Table lock	About to request a table's internal lock after acquiring the table's external lock. This thread state occurs after the System lock thread state.
update	About to start updating table.
Updating	Searching for and updating rows in a table.
updating main table	Updating the first table in a multi-table update, and saving columns and offsets for use in the other tables.
updating reference tables	Updating the secondary (reference) tables in a multi-table update

Value	Description
updating status	This state occurs after a query's execution is complete. If the query's execution time exceeds long_query_time, then Slow_queries is incremented, and if the slow query log is enabled, then the query is logged. If the SERVER_AUDIT plugin is enabled, then the query is also logged into the audit log at this stage. If the userstats plugin is enabled, then CPU statistics are also updated at this stage.
User lock	About to request or waiting for an advisory lock from a GET LOCK() call. For SHOW PROFILE, means requesting a lock only.
User sleep	A SLEEP() call has been invoked.
Waiting for commit lock	FLUSH TABLES WITH READ LOCK is waiting for a commit lock, or a statement resulting in an explicit or implicit commit is waiting for a read lock to be released. This state was called Waiting for all running commits to finish in earlier versions.
Waiting for global read lock	Waiting for a global read lock.
Waiting for table level lock	External lock acquired,and internal lock about to be requested. Occurs after the System lock state. In earlier versions, this was called Table lock.
Waiting for xx lock	Waiting to obtain a lock of type xx.
Waiting on cond	Waiting for an unspecified condition to occur.
Writing to net	Writing a packet to the network.

This page is licensed: CC BY-SA / Gnu FDL

Master Thread States

This article documents thread states that are related to replication master threads. These correspond to the `STATE` values listed by the SHOW PROCESSLIST statement or in the Information Schema PROCESSLIST Table as well as the `PROCESSLIST_STATE` value listed in the Performance Schema threads Table.

Value	Description
Finished reading one binlog; switching to next binlog	After completing one binary log, the next is being opened for sending to the slave.
Master has sent all binlog to slave; waiting for binlog to be updated	All events have been read from the binary logs and sent to the slave. Now waiting for the binary log to be updated with new events.
Sending binlog event to slave	An event has been read from the binary log, and is now being sent to the slave.
Waiting to finalize termination	State that only occurs very briefly while the thread is terminating.

This page is licensed: CC BY-SA / Gnu FDL

Query Cache Thread States

This article documents thread states that are related to the Query Cache. These correspond to the `STATE` values listed by the SHOW PROCESSLIST statement or in the Information Schema PROCESSLIST Table as well as the `PROCESSLIST_STATE` value listed in the Performance Schema threads Table.

Value	Description
checking privileges on cached query	Checking whether the user has permission to access a result in the query cache.
checking query cache for query	Checking whether the current query exists in the query cache.
invalidating query cache entries	Marking query cache entries as invalid as the underlying tables have changed.
sending cached result to client	A result found in the query cache is being sent to the client.
storing result in query cache	Saving the result of a query into the query cache.
Waiting for query cache lock	Waiting to take a query cache lock.

This page is licensed: CC BY-SA / Gnu FDL

Replica I/O Thread States

This article documents thread states that are related to replica I/O threads. These correspond to the `Slave_IO_State` shown by SHOW REPLICA STATUS and the `STATE` values listed by the SHOW PROCESSLIST statement or in the Information Schema PROCESSLIST Table as well as the `PROCESSLIST_STATE` value listed in the Performance Schema threads Table.

Value	Description
Checking master version	Checking the primary's version, which only occurs very briefly after establishing a connection with the primary.
Connecting to master	Attempting to connect to primary.
Queueing master event to the relay log	Event is being copied to the relay log after being read, where it can be processed by the SQL thread.
Reconnecting after a failed binlog dump request	Attempting to reconnect to the primary after a previously failed binary log dump request.
Reconnecting after a failed master event read	Attempting to reconnect to the primary after a previously failed request. After successfully connecting, the state will change to Waiting for master to send event.
Registering slave on master	Registering the replica on the primary, which only occurs very briefly after establishing a connection with the primary.
Requesting binlog dump	Requesting the contents of the binary logs from the given log file name and position. Only occurs very briefly after establishing a connection with the primary.
Waiting for master to send event	Waiting for binary log events to arrive after successfully connecting. If there are no new events on the primary, this state can persist for as many seconds as specified by the slave_net_timeout system variable, after which the thread will reconnect. Prior to MariaDB 10.6.18, MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5, MariaDB 11.2.4 and MariaDB 11.4.2, the time was from SLAVE START. From these versions, the time is since reading the last event.
Waiting for slave mutex on exit	Waiting for replica mutex while the thread is stopping. Only occurs very briefly.
Waiting for the slave SQL thread to free enough relay log space.	Relay log has reached its maximum size, determined by relay_log_space_limit (no limit by default), so waiting for the SQL thread to free up space by processing enough relay log events.
Waiting for master update	State before connecting to primary.

Value	Description
Waiting to reconnect after a failed binlog dump request	Waiting to reconnect after a binary log dump request has failed due to disconnection. The length of time in this state is determined by the MASTER_CONNECT_RETRY clause of the CHANGE MASTER TO statement.
Waiting to reconnect after a failed master event read	Sleeping while waiting to reconnect after a disconnection error. The time in seconds is determined by the MASTER_CONNECT_RETRY clause of the CHANGE MASTER TO statement.

This page is licensed: CC BY-SA / Gnu FDL

Replica Connection Thread States

This article documents thread states that are related to connection threads that occur on a replicatioin replica. These correspond to the `STATE` values listed by the SHOW PROCESSLIST statement or in the Information Schema PROCESSLIST Table as well as the `PROCESSLIST_STATE` value listed in the Performance Schema threads Table.

Value	Description
Changing master	Processing a CHANGE MASTER TO statement.
Killing slave	Processing a STOP SLAVE statement.
Opening master dump table	A table has been created from a master dump and is now being opened.
Reading master dump table data	After the table created by a master dump (the Opening master dump table state) the table is now being read.
Rebuilding the index on master dump table	After the table created by a master dump has been opened and read (the Reading master dump table data state), the index is built.

This page is licensed: CC BY-SA / Gnu FDL

Replica SQL Thread States

This article documents thread states that are related to replication slave SQL threads. These correspond to the `Slave_SQL_State` shown by SHOW SLAVE STATUS as well as the `STATE` values listed by the SHOW PROCESSLIST statement and the Information Schema PROCESSLIST as well as the `PROCESSLIST_STATE` value listed in the Performance Schema threads Table.

Value	Description
Apply log event	Log event is being applied.
Making temp file	Creating a temporary file containing the row data as part of a LOAD DATA INFILE statement.
Reading event from the relay log	Reading an event from the relay log in order to process the event.
Slave has read all relay log, waiting for the slave I/O thread to update it	All relay log events have been processed, now waiting for the I/O thread to write new events to the relay log.
Waiting for work from SQL thread	In parallel replication the worker thread is waiting for more things from the SQL thread.
Waiting for prior transaction to start commit before starting next transaction	In parallel replication the worker thread is waiting for conflicting things to end before starting executing.
Waiting for worker threads to be idle	Happens in parallel replication when moving to a new binary log after a master restart. All slave temporary files are deleted, and worker threads are restarted.
Waiting due to global read lock	In parallel replication when worker threads are waiting for a global read lock to be released.

Value	Description
Waiting for worker threads to pause for global read lock	FLUSH TABLES WITH READ LOCK is waiting for worker threads to finish what they are doing.
Waiting while replication worker thread pool is busy	Happens in parallel replication during a FLUSH TABLES WITH READ LOCK or when changing number of parallel workers.
Waiting for other master connection to process GTID received on multiple master connections	A worker thread noticed that there is already another thread executing the same GTID from another connection and it's waiting for the other to complete.
Waiting for slave mutex on exit	Thread is stopping. Only occurs very briefly.
Waiting for the next event in relay log	State before reading next event from the relay log.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Internal Optimizations

See Fair Choice Between Range and Index_merge Optimizations.

How the optimizer chooses fairly between the range and index_merge access methods.

See Multi Range Read Optimization.

The Multi-Range Read (MRR) optimization, which improves performance for I/O-bound queries scanning many rows.

This page is licensed: CC BY-SA / Gnu FDL

Fair Choice Between Range and Index_merge Optimizations

`index_merge` is a method used by the optimizer to retrieve rows from a single table using several index scans. The results of the scans are then merged.

When using EXPLAIN, if `index_merge` is the plan chosen by the optimizer, it will show up in the "type" column. For example:

```
MariaDB [ontime]> SELECT COUNT(*) FROM ontime;
+--------+
|count(*)|
+--------+
| 1578171|
+--------+

MySQL [ontime]> EXPLAIN SELECT * FROM ontime WHERE (Origin='SEA' OR Dest='SEA');
+--+-----------+------+-----------+-------------+-----------+-------+----+-----+-------------------------
-------------+
|id|select_type|table |type       |possible_keys|key        |key_len|ref |rows |Extra
|
+--+-----------+------+-----------+-------------+-----------+-------+----+-----+-------------------------
-------------+
| 1|SIMPLE     |ontime|index_merge|Origin,Dest  |Origin,Dest|6,6    |NULL|92800|Using union
(Origin,Dest); Using where|
+--+-----------+------+-----------+-------------+-----------+-------+----+-----+-------------------------
-------------+
```

The "rows" column gives us a way to compare efficiency between `index_merge` and other plans.

It is sometimes necessary to discard index_merge in favor of a different plan to avoid a combinatorial explosion of possible range and/or index_merge strategies. But, the old logic in MySQL for when index_merge was rejected caused some good index_merge plans to not even be considered. Specifically, additional `AND` predicates in `WHERE` clauses could cause an index_merge plan to be rejected in favor of a less efficient plan. The slowdown could be anywhere from 10x to over 100x. Here are two examples (based on the previous query) using MySQL:

```
MySQL [ontime]> EXPLAIN SELECT * FROM ontime WHERE (Origin='SEA' OR Dest='SEA') AND securitydelay=0;
+--+-----------+------+----+------------------------+-------------+-------+-----+------+-----------+
|id|select_type|table |type|possible_keys           |key          |key_len|ref  |rows  |Extra      |
+--+-----------+------+----+------------------------+-------------+-------+-----+------+-----------+
| 1|SIMPLE     |ontime|ref |Origin,Dest,SecurityDelay|SecurityDelay|5      |const|791546|Using where|
+--+-----------+------+----+------------------------+-------------+-------+-----+------+-----------+

MySQL [ontime]> EXPLAIN SELECT * FROM ontime WHERE (Origin='SEA' OR Dest='SEA') AND depdelay < 12*60;
+--+-----------+------+----+--------------------+----+-------+----+-------+-----------+
|id|select_type|table |type|possible_keys       |key |key_len|ref |rows   |Extra      |
+--+-----------+------+----+--------------------+----+-------+----+-------+-----------+
| 1|SIMPLE     |ontime|ALL |Origin,DepDelay,Dest|NULL|NULL   |NULL|1583093|Using where|
+--+-----------+------+----+--------------------+----+-------+----+-------+-----------
```

In the above output, the “rows” column shows that the first is almost 10x less efficient and the second is over 15x less efficient than `index_merge`.

Starting in MariaDB 5.3, the optimizer will delay discarding potential `index_merge` plans until the point where it is really necessary.

By not discarding potential `index_merge` plans until absolutely necessary, the two queries stay just as efficient as the original:

```
MariaDB [ontime]> EXPLAIN SELECT * FROM ontime WHERE (Origin='SEA' OR Dest='SEA');
+--+-----------+------+-----------+-------------+-----------+-------+----+-----+-------------------------
------------+
|id|select_type|table |type       |possible_keys|key        |key_len|ref |rows |Extra
|
+--+-----------+------+-----------+-------------+-----------+-------+----+-----+-------------------------
------------+
| 1|SIMPLE     |ontime|index_merge|Origin,Dest  |Origin,Dest|6,6    |NULL|92800|Using union(Origin,Dest);
Using where|
+--+-----------+------+-----------+-------------+-----------+-------+----+-----+-------------------------
------------+

MariaDB [ontime]> EXPLAIN SELECT * FROM ontime WHERE (Origin='SEA' OR Dest='SEA') AND securitydelay=0;
+--+-----------+------+-----------+-------------------------+-----------+-------+----+-----+-------------
------------------------+
|id|select_type|table |type       |possible_keys            |key        |key_len|ref |rows |Extra
|
+--+-----------+------+-----------+-------------------------+-----------+-------+----+-----+-------------
------------------------+
| 1|SIMPLE     |ontime|index_merge|Origin,Dest,SecurityDelay|Origin,Dest|6,6    |NULL|92800|Using
union(Origin,Dest); Using where|
+--+-----------+------+-----------+-------------------------+-----------+-------+----+-----+-------------
------------------------+

MariaDB [ontime]> EXPLAIN SELECT * FROM ontime WHERE (Origin='SEA' OR Dest='SEA') AND depdelay < 12*60;
+--+-----------+------+-----------+--------------------+-----------+-------+----+-----+------------------
--------------------+
|id|select_type|table |type       |possible_keys       |key        |key_len|ref |rows |Extra
|
+--+-----------+------+-----------+--------------------+-----------+-------+----+-----+------------------
--------------------+
| 1|SIMPLE     |ontime|index_merge|Origin,DepDelay,Dest|Origin,Dest|6,6    |NULL|92800|Using
union(Origin,Dest); Using where|
+--+-----------+------+-----------+--------------------+-----------+-------+----+-----+------------------
--------------------+
```

This new behavior is always on and there is no need to enable it. There are no known issues or gotchas with this new optimization.

This page is licensed: CC BY-SA / Gnu FDL

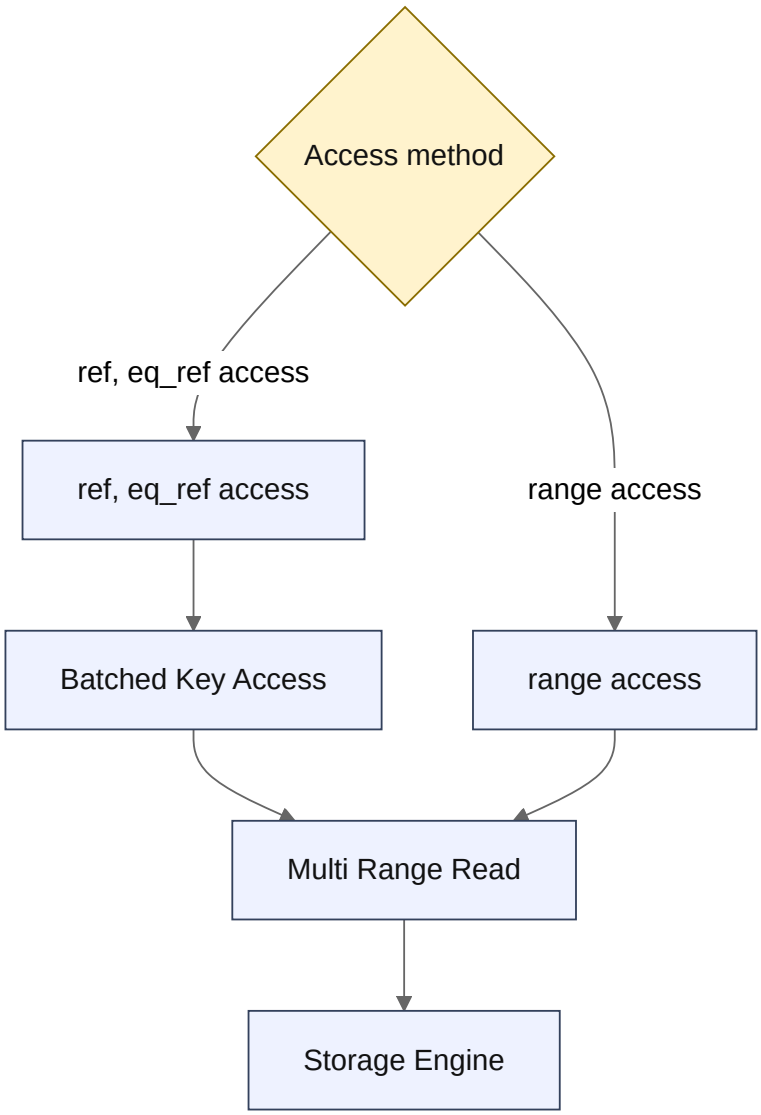
Multi Range Read Optimization

Multi Range Read is an optimization aimed at improving performance for IO-bound queries that need to scan lots of rows.

Multi Range Read can be used with

- `range` access
- `ref` and `eq_ref` access, when they are using Batched Key Access

as shown in this diagram:



Possible ways to use MRR: range access, or ref/eq_ref access via Batched Key Access, both feed into Multi Range Read.

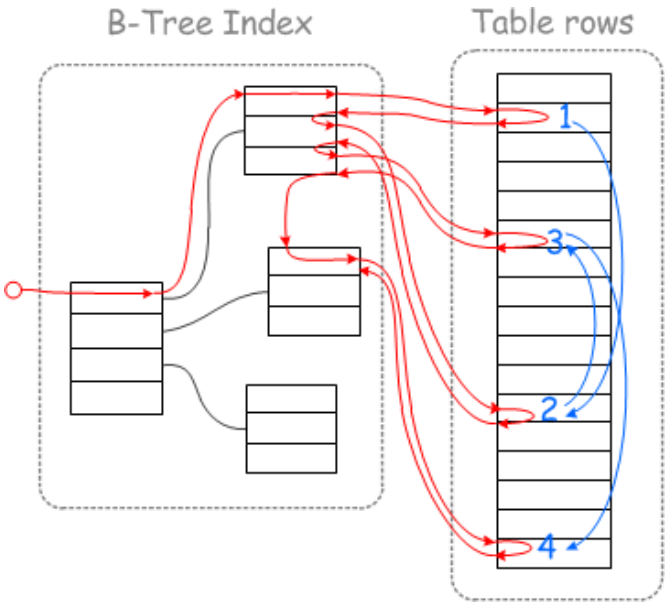
The Idea

Case 1: Rowid Sorting for Range Access

Consider a range query:

```
EXPLAIN SELECT * FROM tbl WHERE tbl.key1 BETWEEN 1000 AND 2000;
+----+-------------+-------+-------+---------------+------+---------+------+------+----------------------
-+
| id | select_type | table | type  | possible_keys | key  | key_len | ref  | rows | Extra
|
+----+-------------+-------+-------+---------------+------+---------+------+------+----------------------
-+
|  1 | SIMPLE      | tbl   | range | key1          | key1 | 5       | NULL |  960 | Using index condition
|
+----+-------------+-------+-------+---------------+------+---------+------+------+----------------------
-+
```

When this query is executed, disk IO access pattern will follow the red line in this figure:



no-mrr-access-pattern

Execution will hit the table rows in random places, as marked with the blue line/numbers in the figure.

When the table is sufficiently big, each table record read will need to actually go to disk (and be served from buffer pool or OS cache), and query execution will be too slow to be practical. For example, a 10,000 RPM disk drive is able to make 167 seeks per second, so in the worst case, query execution will be capped at reading about 167 records per second.

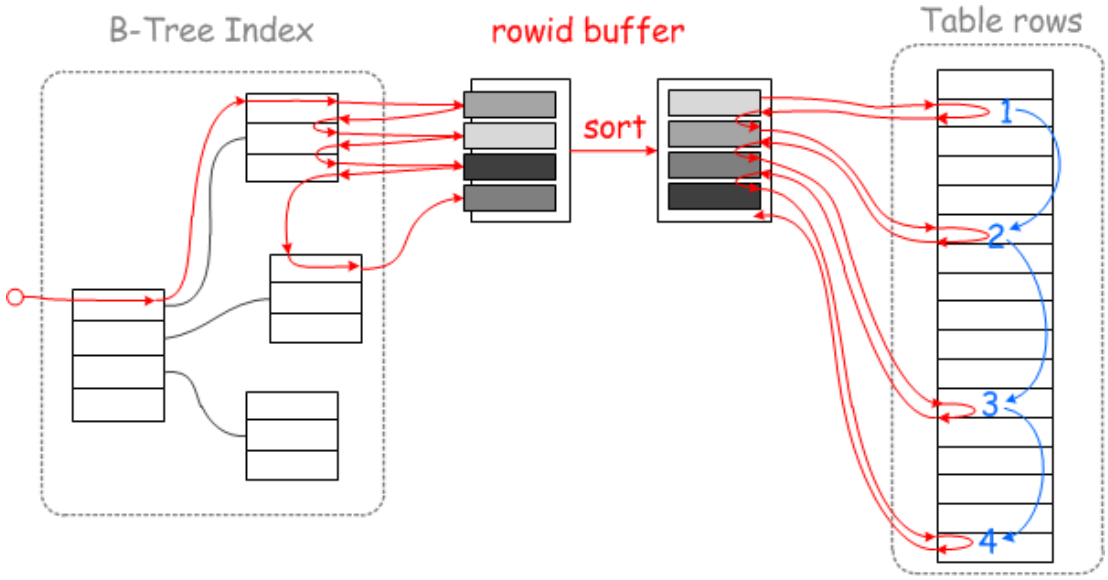
SSD drives do not need to do disk seeks, so they will not be hurt as badly, however the performance will still be poor in many cases.

Multi-Range-Read optimization aims to make disk access faster by sorting record read requests and then doing one ordered disk sweep. If one enables Multi Range Read, `EXPLAIN` will show that a "`Rowid-ordered scan`" is used:

```
SET optimizer_switch='mrr=ON';
Query OK, 0 rows affected (0.06 sec)

EXPLAIN SELECT * FROM tbl WHERE tbl.key1 BETWEEN 1000 AND 2000;
+----+-------------+-------+-------+---------------+------+---------+------+------+----------------------
----------------------+
| id | select_type | table | type  | possible_keys | key  | key_len | ref  | rows | Extra
|
+----+-------------+-------+-------+---------------+------+---------+------+------+----------------------
----------------------+
|  1 | SIMPLE      | tbl   | range | key1          | key1 | 5       | NULL |  960 | Using index
condition; Rowid-ordered scan |
+----+-------------+-------+-------+---------------+------+---------+------+------+----------------------
----------------------+
1 row in set (0.03 sec)
```

and the execution will proceed as follows:



mrr-access-pattern

Reading disk data sequentially is generally faster, because

- Rotating drives do not have to move the head back and forth
- One can take advantage of IO-prefetching done at various levels
- Each disk page will be read exactly once, which means we won't rely on disk cache (or buffer pool) to save us from reading the same page multiple times.

The above can make a huge difference on performance. There is also a catch, though:

- If you're scanning small data ranges in a table that is sufficiently small so that it completely fits into the OS disk cache, then you may observe that the only effect of MRR is that extra buffering/sorting adds some CPU overhead.
- `LIMIT n` and `ORDER BY ... LIMIT n` queries with small values of `n` may become slower. The reason is that MRR reads data in disk order, while `ORDER BY ... LIMIT n` wants first `n` records in index order.

Case 2: Rowid Sorting for Batched Key Access

Batched Key Access can benefit from rowid sorting in the same way as range access does. If one has a join that uses index lookups:

```
EXPLAIN SELECT * FROM t1,t2 WHERE t2.key1=t1.col1;
+----+-------------+-------+------+---------------+------+---------+--------------+------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref          | rows | Extra       |
+----+-------------+-------+------+---------------+------+---------+--------------+------+-------------+
|  1 | SIMPLE      | t1    | ALL  | NULL          | NULL | NULL    | NULL         | 1000 | Using where |
|  1 | SIMPLE      | t2    | ref  | key1          | key1 | 5       | test.t1.col1 |    1 |             |
+----+-------------+-------+------+---------------+------+---------+--------------+------+-------------+
2 rows in set (0.00 sec)
```

Execution of this query will cause table `t2` to be hit in random locations by lookups made through `t2.key1=t1.col`. If you enable Multi Range and Batched Key Access, you will get table `t2` to be accessed using a `Rowid-ordered scan`:

```
SET optimizer_switch='mrr=ON';
Query OK, 0 rows affected (0.06 sec)

SET join_cache_level=6;
Query OK, 0 rows affected (0.00 sec)

EXPLAIN SELECT * FROM t1,t2 WHERE t2.key1=t1.col1;
+----+-------------+-------+------+---------------+------+---------+--------------+------+--------------
----------------------------------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref          | rows | Extra
|
+----+-------------+-------+------+---------------+------+---------+--------------+------+--------------
----------------------------------------+
|  1 | SIMPLE      | t1    | ALL  | NULL          | NULL | NULL    | NULL         | 1000 | Using where
|
|  1 | SIMPLE      | t2    | ref  | key1          | key1 | 5       | test.t1.col1 |    1 | Using join
buffer (flat, BKA join); Rowid-ordered scan |
+----+-------------+-------+------+---------------+------+---------+--------------+------+--------------
----------------------------------------+
2 rows in set (0.00 sec)
```

The benefits will be similar to those listed for `range` access.

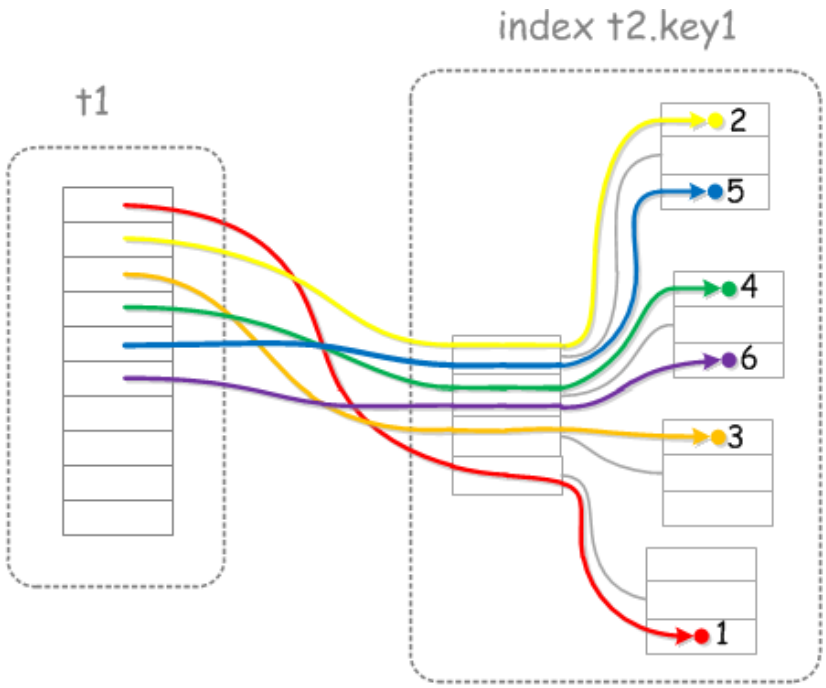
An additional source of speedup is this property: if there are multiple records in `t1` that have the same value of `t1.col1`, then regular Nested-Loops join will make multiple index lookups for the same value of `t2.key1=t1.col1`. The lookups may or may not hit the cache, depending on how big the join is. With Batched Key Access and Multi-Range Read, no duplicate index lookups will be made.

Case 3: Key Sorting for Batched Key Access

Let us consider again the nested loop join example, with `ref` access on the second table:

```
EXPLAIN SELECT * FROM t1,t2 WHERE t2.key1=t1.col1;
+----+-------------+-------+------+---------------+------+---------+--------------+------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref          | rows | Extra       |
+----+-------------+-------+------+---------------+------+---------+--------------+------+-------------+
|  1 | SIMPLE      | t1    | ALL  | NULL          | NULL | NULL    | NULL         | 1000 | Using where |
|  1 | SIMPLE      | t2    | ref  | key1          | key1 | 5       | test.t1.col1 |    1 |             |
+----+-------------+-------+------+---------------+------+---------+--------------+------+-------------+
```

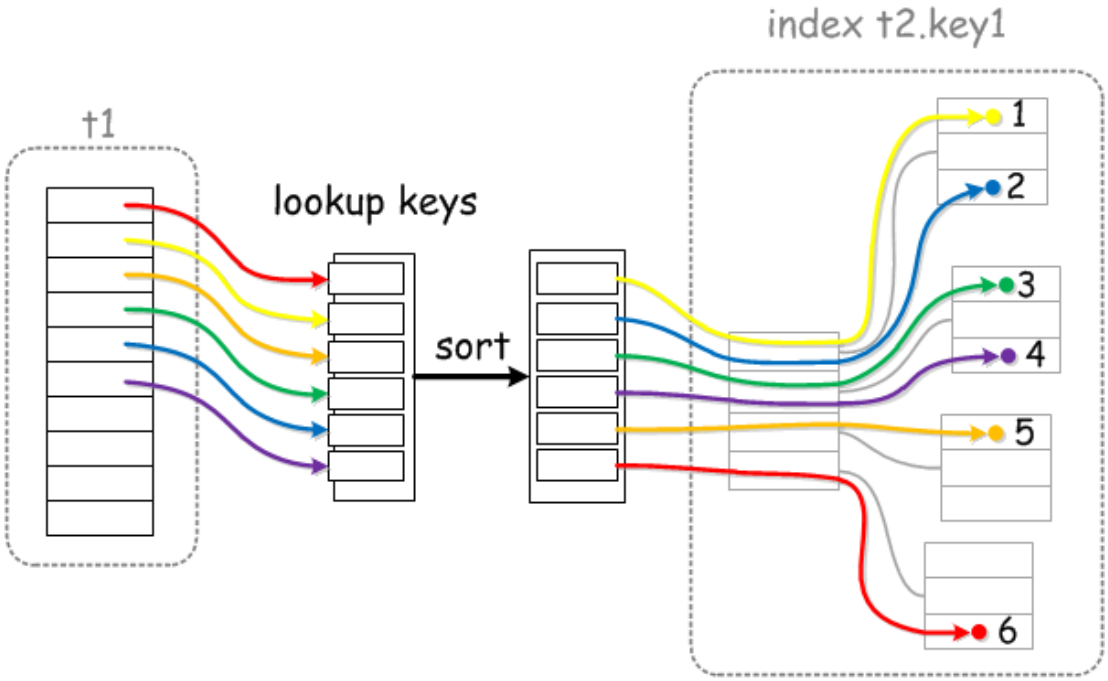
Execution of this query plan will cause random hits to be made into the index `t2.key1`, as shown in this picture:



Regular Nested Loops Join will hit the index at random

key-sorting-regular-nl-join

In particular, on step #5 we'll read the same index page that we've read on step #2, and the page we've read on step #4 will be re-read on step#6. If all pages you're accessing are in the cache (in the buffer pool, if you're using InnoDB, and in the key cache, if you're using MyISAM), this is not a problem. However, if your hit ratio is poor and you're going to hit the disk, it makes sense to sort the lookup keys, like shown in this figure:



With *Batched Nested Loops Join* and *Key-Ordered retrieval*, index lookups are done in one "sweep"

key-sorting-join

This is roughly what `Key-ordered scan` optimization does. In EXPLAIN, it looks as follows:

```
SET optimizer_switch='mrr=ON,mrr_sort_keys=ON';
Query OK, 0 rows affected (0.00 sec)

SET join_cache_level=6;
Query OK, 0 rows affected (0.02 sec)
EXPLAIN SELECT * FROM t1,t2 WHERE t2.key1=t1.col1\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        TABLE: t1
         type: ALL
possible_keys: a
          KEY: NULL
      key_len: NULL
          ref: NULL
         ROWS: 1000
        Extra: USING WHERE
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        TABLE: t2
         type: ref
possible_keys: key1
          KEY: key1
      key_len: 5
          ref: test.t1.col1
         ROWS: 1
        Extra: USING JOIN buffer (flat, BKA JOIN); KEY-ordered Rowid-ordered scan
2 rows in set (0.00 sec)
```

((TODO: a note about why sweep-read over InnoDB's clustered primary index scan (which is, actually the whole InnoDB table itself) will use `Key-ordered scan` algorithm, but not `Rowid-ordered scan` algorithm, even though conceptually they are the same thing in this case))

Buffer Space Management

As was shown above, Multi Range Read requires sort buffers to operate. The size of the buffers is limited by system variables. If MRR has to process more data than it can fit into its buffer, it will break the scan into multiple passes. The more passes are made, the less is the speedup though, so one needs to balance between having too big buffers (which consume lots of memory) and too small buffers (which limit the possible speedup).

Range Access

When MRR is used for `range` access, the size of its buffer is controlled by the mrr_buffer_size system variable. Its value specifies how much space can be used for each table. For example, if there is a query which is a 10-way join and MRR is used for each table, `10*@@mrr_buffer_size` bytes may be used.

Batched Key Access

When Multi Range Read is used by Batched Key Access, then buffer space is managed by BKA code, which will automatically provide a part of its buffer space to MRR. You can control the amount of space used by BKA by setting

- join_buffer_size to limit how much memory BKA uses for each table, and
- join_buffer_space_limit to limit the total amount of memory used by BKA in the join.

Status Variables

There are three status variables related to Multi Range Read:

Variable name	Meaning
Handler_mrr_init	Counts how many Multi Range Read scans were performed

Variable name	Meaning
Handler_mrr_key_refills	Number of times key buffer was refilled (not counting the initial fill)
Handler_mrr_rowid_refills	Number of times rowid buffer was refilled (not counting the initial fill)

Non-zero values of `Handler_mrr_key_refills` and/or `Handler_mrr_rowid_refills` mean that Multi Range Read scan did not have enough memory and had to do multiple key/rowid sort-and-sweep passes. The greatest speedup is achieved when Multi Range Read runs everything in one pass, if you see lots of refills it may be beneficial to increase sizes of relevant buffers mrr_buffer_size join_buffer_size and join_buffer_space_limit

Effect on Other Status Variables

When a Multi Range Read scan makes an index lookup (or some other "basic" operation), the counter of the "basic" operation, e.g. Handler_read_key, will also be incremented. This way, you can still see total number of index accesses, including those made by MRR. Per-user/table/index statistics counters also include the row reads made by Multi Range Read scans.

Why Using Multi Range Read Can Cause Higher Values in Status Variables

Multi Range Read is used for scans that do full record reads (i.e., they are not "Index only" scans). A regular non-index-only scan will read

1. an index record, to get a rowid of the table record
2. a table record
   Both actions will be done by making one call to the storage engine, so the effect of the call will be that the relevan `Handler_read_XXX` counter will be incremented BY ONE, and Innodb_rows_read will be incremented BY ONE.

Multi Range Read will make separate calls for steps #1 and #2, causing TWO increments to `Handler_read_XXX` counters and TWO increments to `Innodb_rows_read` counter. To the uninformed, this looks as if Multi Range Read was making things worse. Actually, it doesn't - the query will still read the same index/table records, and actually Multi Range Read may give speedups because it reads data in disk order.

Multi Range Read Factsheet

- Multi Range Read is used by
  - `range` access method for range scans.
  - Batched Key Access for joins
- Multi Range Read can cause slowdowns for small queries over small tables, so it is disabled by default.
- There are two strategies:
  - Rowid-ordered scan
  - Key-ordered scan
- : and you can tell if either of them is used by checking the `Extra` column in `EXPLAIN` output.
- There are three optimizer_switch flags you can switch ON:
  - `mrr=on` - enable MRR and rowid ordered scans
  - `mrr_sort_keys=on` - enable Key-ordered scans (you must also set `mrr=on` for this to have any effect)
  - `mrr_cost_based=on` - enable cost-based choice whether to use MRR. Currently not recommended, because cost model is not sufficiently tuned yet.

Differences from MySQL

- MySQL supports only `Rowid ordered scan` strategy, which it shows in `EXPLAIN` as `Using MRR`.
- EXPLAIN in MySQL shows `Using MRR`, while in MariaDB it may show

    - `Rowid-ordered scan`
    - `Key-ordered scan`
    - `Key-ordered Rowid-ordered scan`
- MariaDB uses mrr_buffer_size as a limit of MRR buffer size for `range` access, while MySQL uses read_rnd_buffer_size.
- MariaDB has three MRR counters: Handler_mrr_init, `Handler_mrr_extra_rowid_sorts`, `Handler_mrr_extra_key_sorts`, while MySQL has only `Handler_mrr_init`, and it will only count MRR scans that were used by BKA. MRR scans used by range access are not counted.

This page is licensed: CC BY-SA / Gnu FDL

Operating System Optimizations

See Filesystem Optimizations.

Filesystem choices and settings that affect MariaDB Server performance.

See Storage I/O: Buffering and Persistence.

Defines the OS-agnostic terms MariaDB documentation uses for disk I/O — unbuffered I/O, write-through, persist — and shows how each one maps to Unix-like and Windows mechanisms.

This page is licensed: CC BY-SA / Gnu FDL

Filesystem Optimizations

Suitability of Filesystems

The filesystem is not the most important aspect of MariaDB performance. More important are the available memory (RAM), the drive speed, and the system variable settings (see Hardware Optimization and System Variables).

Optimizing the filesystem can, however, make a noticeable difference in some cases. Among the best suited Linux filesystems are ext4, XFS and Btrfs. They are all included in the mainline Linux kernel and are widely supported and available on most Linux distributions.

The following theoretical file size and filesystem size limits apply to those filesystems:

Limit	ext4	XFS	Btrfs
Links	ext4	XFS	Btrfs
Max file size	16-256 TB	8 EB	16 EB
Max filesystem size	1 EB	8 EB	16 EB

Each has unique characteristics that are worth understanding to get the most from their usage.

Disabling Access Time

It's unlikely you'll need to record file access time on a database server, and mounting your filesystem with this disabled can give an easy improvement in performance. To do so, use the `noatime` option.

If you want to keep access time for log files or other system files, these can be stored on a separate drive.

Using NFS

Generally, we recommend not to use NFS (Network File System) with MariaDB, for these reasons:

- MariaDB data and log files on NFS volumes can become locked and unavailable for use. Locking issues may occur in cases where multiple instances of MariaDB access the same data directory, or when MariaDB is shut down improperly, for instance, due to a power outage. In particular, sharing a data directory among MariaDB instances is not recommended.
- Data inconsistencies due to messages received out of order or lost network traffic. To avoid this issue, use TCP with `hard` and `intr` mount options.

Using NFS within a professional SAN[29] environment or other storage system tends to offer greater reliability than using NFS outside of such an environment. However, NFS within a SAN environment may be slower than directly attached or bus-attached non-rotational storage.

This page is licensed: CC BY-SA / Gnu FDL

Storage I/O: Buffering and Persistence

This page defines the I/O behavior terms that other MariaDB documentation pages use — *unbuffered I/O*, *write-through*, *persist to non-volatile storage* — and shows how each one maps to the underlying mechanism on Unix-like systems and Windows. Other pages refer here for the OS-level details rather than embedding system-specific flag names inline.

Why This Matters

The MariaDB server, and especially the InnoDB storage engine, needs precise control over when writes actually reach non-volatile storage. The operating system normally buffers writes in memory (the *page cache*) and flushes them to disk on its own schedule. That is fast, but unsafe for a transactional database: a power loss between the application write and the OS flush would lose committed data.

MariaDB therefore exposes configuration that lets administrators choose between OS-buffered performance and database-controlled durability. The configuration uses general I/O terms — buffering, write-through, persistence — that describe the behavior independent of the operating system. This page is the reference for what those terms mean and how each OS implements them.

Unbuffered I/O

Behavior: Reads and writes bypass the operating system's page cache. Each read fetches directly from the storage device; each write is handed to the device without keeping a copy in OS memory.

Trade-off: Avoids "double buffering" — that is, caching the same data both in MariaDB's own buffer pool and in the OS page cache, which costs memory and CPU. The trade-off is the loss of the OS's read-ahead and write-coalescing optimizations. Unbuffered I/O is most appropriate when MariaDB itself manages most of the memory available for caching data (for example, when the InnoDB buffer pool is sized to most of system RAM).

Operating system	Underlying mechanism
Unix-like	File opened with `O_DIRECT`. Supported on Linux, FreeBSD, NetBSD, Dragonfly BSD, Illumos / Solaris, and IBM AIX; not supported on OpenBSD or Apple macOS.
Windows	File opened with `FILE_FLAG_NO_BUFFERING`

Write-Through

Behavior: Every individual write call returns only after the storage layer has accepted the data for non-volatile storage. Equivalent to issuing a write followed by an immediate persist, on every single write.

Trade-off: Maximum durability per write, at the cost of write latency. The application does not need to issue a separate persist call to know that data is safe.

Operating system	Underlying mechanism
Unix-like	File opened with `O_DSYNC`. Supported on Linux, FreeBSD, NetBSD, Solaris, and IBM AIX; not supported on Dragonfly BSD (which has `O_FSYNC`, not used by MariaDB), OpenBSD, or Apple macOS.
Windows	File opened with `FILE_FLAG_WRITE_THROUGH`

Persist to Non-Volatile Storage

Behavior: After one or more buffered writes, force any data still in the OS page cache or the storage device's own write cache out to non-volatile media. The call returns only once the data is durable.

Trade-off: Lets the application batch many writes cheaply and persist them with a single explicit call, instead of paying the durability cost on every write (as write-through does). This is the foundation of MariaDB's group-commit and binary-log durability behavior.

MariaDB does not validate file access or modification timestamps anywhere, so on each platform it prefers the lighter-weight call that omits those metadata updates, falling back to the full-persist call only when the lighter one is not available.

Operating system	Underlying mechanism
Unix-like	`fdatasync()` when available (Linux, FreeBSD, Dragonfly BSD, NetBSD, OpenBSD, Solaris, IBM AIX); `fsync()` otherwise (notably Apple macOS).
Windows	`NtFlushBuffersFileEx()` with the `FLUSH_FLAGS_FILE_DATA_SYNC_ONLY` flag when available — documented for NTFS, and observed to work on ReFS as well. `FlushFileBuffers()` otherwise (older Windows versions, non-NTFS filesystems, or if the lighter call fails at run time).

See Also

- `innodb_flush_method` — selects InnoDB's combination of buffering and persistence strategies.
- `sync_binlog` — controls how often the binary log is persisted.
- `innodb_doublewrite` — InnoDB's protection against torn writes.
- Filesystem Optimizations — choosing and tuning the filesystem MariaDB writes to.

This page is licensed: CC BY-SA / Gnu FDL

Optimization and Indexes

See Building the best INDEX for a given SELECT.

A cookbook for designing the best index for a given SELECT query.

See Compound (Composite) Indexes.

How compound (composite) indexes work in MariaDB and how to design them effectively.

See Foreign Keys.

Complete MariaDB performance optimization guide. Complete reference for query tuning, indexing strategies, and configuration improvements for production use.

See Ignored Indexes.

Ignored indexes allow indexes to be visible and maintained without being used by the optimizer. This feature is comparable to MySQL 8's "invisible indexes."

See Index Statistics.

Index statistics provide crucial insights to the MariaDB query optimizer, guiding it in executing queries efficiently. Up-to-date index statistics ensure optimized query performance.

See Latitude/Longitude Indexing.

Techniques for indexing latitude/longitude data to speed up nearest-location queries.

See Primary Keys with Nullable Columns.

How MariaDB handles primary keys over nullable columns, following the SQL standard.

See Storage Engine Index Types.

The index types available when creating an index: BTREE, HASH, and RTREE.

See Full-Text Indexes.

Implement full-text indexes in MariaDB Server for efficient text search. This section guides you through creating and utilizing these indexes to optimize queries on large text datasets.

This page is licensed: CC BY-SA / Gnu FDL

Building the best INDEX for a given SELECT

The problem

You have a SELECT and you want to build the best INDEX for it. This blog is a “cookbook” on how to do that task.

- A short algorithm that works for many simpler SELECTs and helps in complex queries.
- Examples of the algorithm, plus digressions into exceptions and variants
- Finally a long list of “other cases”.

The hope is that a newbie can quickly get up to speed, and his/her INDEXes will no longer smack of “newbie”.

Many edge cases are explained, so even an expert may find something useful here.

Algorithm

Here’s the way to approach creating an INDEX, given a SELECT. Follow the steps below, gathering columns to put in the INDEX in order. When the steps give out, you usually have the ‘perfect’ index.

1. Given a WHERE with a bunch of expressions connected by AND: Include the columns (if any), in any order, that are compared to a constant and not hidden in a function.
2. You get one more chance to add to the INDEX; do the first of these that applies:

- 2a. One column used in a ‘range’ – BETWEEN, ‘>’, LIKE w/o leading wildcard, etc.
- 2b. All columns, in order, of the GROUP BY.
- 2c. All columns, in order, of the ORDER BY if there is no mixing of ASC and DESC.

Digression

This blog assumes you know the basic idea behind having an INDEX. Here is a refresher on some of the key points.

Virtually all INDEXes in MySQL are structured as BTrees
BTrees allow very efficient for

- Given a key, find the corresponding row(s);
- “Range scans” – That is start at one value for the key and repeatedly find the “next” (or “previous”) row.

A PRIMARY KEY is a UNIQUE KEY; a UNIQUE KEY is an INDEX. (“KEY” == “INDEX”.)

InnoDB “clusters” the PRIMARY KEY with the data. Hence, given the value of the PK (“PRIMARY KEY”), after drilling down the BTree to find the index entry, you have all the columns of the row when you get there. A “secondary key” (any UNIQUE or INDEX other than the PK) in InnoDB first drills down the BTree for the secondary index, where it finds a copy of the PK. Then it drills down the PK to find the row.

Every InnoDB table has a PRIMARY KEY. While there is a default if you do not specify one, it is best to explicitly provide a PK.

For completeness: MyISAM works differently. All indexes (including the PK) are in separate BTrees. The leaf node of such BTrees have a pointer (usually a byte offset) into the data file.

All discussion here assumes InnoDB tables, however most statements apply to other Engines.

First, some examples

Think of a list of names, sorted by last_name, then first_name. You have undoubtedly seen such lists, and they often have other information such as address and phone number. Suppose you wanted to look me up. If you remember my full name (‘James’ and ‘Rick’), it is easy to find my entry. If you remembered only my last name (‘James’) and first initial (‘R’). You would quickly zoom in on the Jameses and find the Rs in them. There, you might remember ‘Rick’ and ignore ‘Ronald’. But, suppose you remembered my first name (‘Rick’) and only my last initial (‘J’). Now you are in trouble. You would be scanning all the Js – Jones, Rick; Johnson, Rick; Jamison, Rick; etc, etc. That’s much less efficient.

Those equate to

```
INDEX(last_name, first_name) -- the order of the list.
    WHERE last_name = 'James' AND first_name = 'Rick'  -- best case
    WHERE last_name = 'James' AND first_name LIKE 'R%' -- pretty good
    WHERE last_name LIKE 'J%' AND first_name = 'Rick'  -- pretty bad
```

Think about this example as I talk about “=” versus “range” in the Algorithm, below.

Algorithm, step 1 (WHERE “column = const”)

- `WHERE aaa = 123 AND ...` : an INDEX starting with aaa is good.
- `WHERE aaa = 123 AND bbb = 456 AND ...` : an INDEX starting with aaa and bbb is good. In this case, it does not matter whether aaa or bbb comes first in the INDEX.
- `xxx IS NULL` : this acts like “= const” for this discussion.
- `WHERE t1.aa = 123 AND t2.bb = 456` – You must only consider columns in the current table.

Note that the expression must be of the form of `column_name` = (constant). These do not apply to this step in the Algorithm: DATE(dt) = ‘…’, LOWER(s) = ‘…’, CAST(s …) = ‘…’, x=‘…’ COLLATE…

(If there are no “=” parts AND’d in the WHERE clause, move on to step 2 without any columns in your putative INDEX.)

Algorithm, step 2

Find the first of 2a / 2b / 2c that applies; use it; then quit. If none apply, then you are through gathering columns for the index.

In some cases it is optimal to do step 1 (all equals) plus step 2c (ORDER BY).

Algorithm, step 2a (one range)

A “range” shows up as

- `aaa >= 123` – any of <, <=, >=, >; but not <>, !=

- `aaa BETWEEN 22 AND 44`
- `sss LIKE 'blah%'` – but not `sss LIKE '%blah'`
- `xxx IS NOT NULL`
  Add the column in the range to your putative INDEX.

If there are more parts to the WHERE clause, you must stop now.

Complete examples (assume nothing else comes after the snippet)

- `WHERE aaa >= 123 AND bbb = 1 ⇒ INDEX(bbb, aaa)` (WHERE order does not matter; INDEX order does)
- `WHERE aaa >= 123 ⇒ INDEX(aaa)`
- `WHERE aaa >= 123 AND ccc > 'xyz' ⇒ INDEX(aaa) or INDEX(ccc)` (only one range)
- `WHERE aaa >= 123 ORDER BY aaa ⇒ INDEX(aaa)` – Bonus: The ORDER BY will use the INDEX.
- `WHERE aaa >= 123 ORDER BY aaa ⇒ INDEX(aaa) DESC` – Same Bonus.

Algorithm, step 2b (GROUP BY)

If there is a GROUP BY, all the columns of the GROUP BY should now be added, in the specified order, to the INDEX you are building. (I do not know what happens if one of the columns is already in the INDEX.)

If you are GROUPing BY an expression (including function calls), you cannot use the GROUP BY; stop.

Complete examples (assume nothing else comes after the snippet)

- `WHERE aaa = 123 AND bbb = 1 GROUP BY ccc ⇒ INDEX(bbb, aaa, ccc) or INDEX(aaa, bbb, ccc)` (='s first, in any order; then the GROUP BY)
- `WHERE aaa >= 123 GROUP BY xxx ⇒ INDEX(aaa)` (You should have stopped with Step 2a)
- `GROUP BY x,y ⇒ INDEX(x,y)` (no WHERE)
- `WHERE aaa = 123 GROUP BY xxx, (a+b) ⇒ INDEX(aaa)` – expression in GROUP BY, so no use including even xxx.

Algorithm, step 2c (ORDER BY)

If there is a ORDER BY, all the columns of the ORDER BY should now be added, in the specified order, to the INDEX you are building.

If there are multiple columns in the ORDER BY, and there is a mixture of ASC and DESC, do not add the ORDER BY columns; they won't help; stop.

If you are ORDERing BY an expression (including function calls), you cannot use the ORDER BY; stop.

Complete examples (assume nothing else comes after the snippet)

- `WHERE aaa = 123 GROUP BY ccc ORDER BY ddd ⇒ INDEX(aaa, ccc)` – should have stopped with Step 2b
- `WHERE aaa = 123 GROUP BY ccc ORDER BY ccc ⇒ INDEX(aaa, ccc)` – the ccc will be used for both GROUP BY and ORDER BY
- `WHERE aaa = 123 ORDER BY xxx ASC, yyy DESC ⇒ INDEX(aaa)` – mixture of ASC and DESC.

The following are especially good. Normally a LIMIT cannot be applied until after lots of rows are gathered and then sorted according to the ORDER BY. But, if the INDEX gets all they way through the ORDER BY, only (OFFSET + LIMIT) rows need to be gathered. So, in these cases, you win the lottery with your new index:

- `WHERE aaa = 123 GROUP BY ccc ORDER BY ccc LIMIT 10 ⇒ INDEX(aaa, ccc)`
- `WHERE aaa = 123 ORDER BY ccc LIMIT 10 ⇒ INDEX(aaa, ccc)`
- `ORDER BY ccc LIMIT 10 ⇒ INDEX(ccc)`
- `WHERE ccc > 432 ORDER BY ccc LIMIT 10 ⇒ INDEX(ccc)` – This "range" is compatible with ORDER BY

(It does not make much sense to have a LIMIT without an ORDER BY, so I do not discuss that case.)

Algorithm end

You have collected a few columns; put them in INDEX and ADD that to the table. That will often produce a “good” index for the SELECT you have. Below are some other suggestions that may be relevant.

An example of the Algorithm being ‘wrong’:

```
SELECT ... FROM t WHERE flag = true;
```

This would (according to the Algorithm) call for INDEX(flag). However, indexing a column that has two (or a small number of) values is almost always useless. This is called ‘low cardinality’. The Optimizer would prefer to do a table scan than to bounce between the index BTree and the data.

On the other hand, the Algorithm is ‘right’ with

```
SELECT ... FROM t WHERE flag = true AND date >= '2015-01-01';
```

That would call for a compound index starting with a flag: INDEX(flag, date). Such an index is likely to be very beneficial. And it is likely to be more beneficial than INDEX(date).

If your resulting INDEX include column(s) that are likely to be UPDATEd, note that the UPDATE will have extra work to remove a ‘row’ from one place in the INDEX’s BTree and insert a ‘row’ back into the BTree. For example:

```
INDEX(x)
UPDATE t SET x = ... WHERE ...;
```

There are too many variables to say whether it is better to keep the index or to toss it.

In this case, shortening the index may be beneficial:

```
INDEX(z, x)
UPDATE t SET x = ... WHERE ...;
```

Changing to INDEX(z) would make for less work for the UPDATE, but might hurt some SELECT. It depends on the frequency of each, plus many more factors.

Limitations

(There are exceptions to some of these.)

- You may not create an index bigger than 3KB.
- You may not include a column that equates to bigger than some value (767 bytes – VARCHAR(255) CHARACTER SET utf8).
- You can deal with big fields using “prefix” indexing; but see below.
- You should not have more than 5 columns in an index. (This is just a Rule of Thumb; nothing prevents having more.)
- You should not have redundant indexes. (See below.)

Flags and low cardinality

`INDEX(flag)` is almost never useful if `flag` has very few values. More specifically, when you say WHERE flag = 1 and “1” occurs more than 20% of the time, such an index will be shunned. The Optimizer would prefer to scan the table instead of bouncing back and forth between the index and the data for more than 20% of the rows.

(“20%” is really somewhere between 10% and 30%, depending on the phase of the moon.)

“Covering” indexes

A “Covering” index is an index that contains all the columns in the SELECT. It is special in that the SELECT can be completed by looking only at the INDEX BTree. (Since InnoDB’s PRIMARY KEY is clustered with the data, “covering” is of no benefit when considering at the PRIMARY KEY.)

Mini-cookbook:

1. Gather the list of column(s) according to the “Algorithm”, above.
2. Add to the end of the list the rest of the columns seen in the SELECT, in any order.

Examples:

- `SELECT x FROM t WHERE y = 5; ⇒ INDEX(y,x)` – The algorithm said just INDEX(y)
- `SELECT x,z FROM t WHERE y = 5 AND q = 7; ⇒ INDEX(y,q,x,z)` – y and q in either order (Algorithm), then x and z in either order (covering).
- `SELECT x FROM t WHERE y > 5 AND q > 7; ⇒ INDEX(y,q,x)` – y or q first (that’s as far as the Algorithm goes), then the other two fields afterwards.

The speedup you get might be minor, or it might be spectacular; it is hard to predict.

But…

- It is not wise to build an index with lots of columns. Let’s cut it off at 5 (Rule of Thumb).
- Prefix indexes cannot ‘cover’, so don’t use them anywhere in a ‘covering’ index.
- There are limits (3KB?) on how ‘wide’ an index can be, so “covering” may not be possible.

Redundant/excessive indexes

INDEX(a,b) can find anything that INDEX(a) could find. So you don’t need both. Get rid of the shorter one.

If you have lots of SELECTs and they generate lots of INDEXes, this may cause a different problem. Each index must be updated (sooner or later) for each INSERT. More indexes ⇒ slower INSERTs. Limit the number of indexes on a table to about 6 (Rule of Thumb).

Notice in the cookbook how it says “in any order” in a few places. If, for example, you have both of these (in different SELECTs):

- `WHERE a=1 AND b=2` begs for either INDEX(a,b) or INDEX(b,a)
- `WHERE a>1 AND b=2` begs only for INDEX(b,a)
  Include only INDEX(b,a) since it handles both cases with only one INDEX.

Suppose you have a lot of indexes, including (a,b,c,dd) and (a,b,c,ee). Those are getting rather long. Consider either picking one of them, or having simply (a,b,c). Sometimes the selectivity of (a,b,c) is so good that tacking on ‘dd’ or ‘ee’ does make enough difference to matter.

Optimizer picks ORDER BY

The main cookbook skips over an important optimization that is sometimes used. The optimizer will sometimes ignore the WHERE and, instead, use an INDEX that matches the ORDER BY. This, of course, needs to be a perfect match – all columns, in the same order. And all ASC or all DESC.

This becomes especially beneficial if there is a LIMIT.

But there is a problem. There could be two situations, and the Optimizer is sometimes not smart enough to see which case applies:

- If the WHERE does very little filtering, fetching the rows in ORDER BY order avoids a sort and has little wasted effort (because of ‘little filtering’). Using the INDEX matching the ORDER BY is better in this case.

- If the WHERE does a lot of filtering, the ORDER BY is wasting a lot of time fetching rows only to filter them out. Using an INDEX matching the WHERE clause is better.

What should you do? If you think the “little filtering” is likely, then create an index with the ORDER BY columns in order and hope that the Optimizer uses it when it should.

OR

Cases…

- `WHERE a=1 OR a=2` – This is turned into WHERE a IN (1,2) and optimized that way.
- `WHERE a=1 OR b=2` usually cannot be optimized.
- `WHERE x.a=1 OR y.b=2` This is even worse because of using two different tables.

A workaround is to use UNION. Each part of the UNION is optimized separately. For the second case:

```
( SELECT ... WHERE a=1 )   -- and have INDEX(a)
   UNION DISTINCT -- "DISTINCT" is assuming you need to get rid of dups
   ( SELECT ... WHERE b=2 )   -- and have INDEX(b)
   GROUP BY ... ORDER BY ...  -- whatever you had at the end of the original query
```

Now the query can take good advantage of two different indexes. Note: “Index merge” might kick in on the original query, but it is not necessarily any faster than the UNION. Sister blog on compound indexes, including ‘Index Merge’

The third case (OR across 2 tables) is similar to the second.

If you originally had a LIMIT, UNION gets complicated. If you started with ORDER BY z LIMIT 190, 10, then the UNION needs to be

```
( SELECT ... LIMIT 200 )   -- Note: OFFSET 0, LIMIT 190+10
   UNION DISTINCT -- (or ALL)
   ( SELECT ... LIMIT 200 )
   LIMIT 190, 10              -- Same as originally
```

TEXT / BLOB

You cannot directly index a TEXT or BLOB or large VARCHAR or large BINARY column. However, you can use a “prefix” index: INDEX(foo(20)). This says to index the first 20 characters of `foo`. But… It is rarely useful.

Example of a prefix index:

```
INDEX(last_name(2), first_name)
```

The index for me would contain ‘Ja’, ‘Rick’. That’s not useful for distinguishing between ‘Jamison’, ‘Jackson’, ‘James’, etc., so the index is so close to useless that the optimizer often ignores it.

Probably never do UNIQUE(foo(20)) because this applies a uniqueness constraint on the first 20 characters of the column, not the whole column!

More on prefix indexing

Dates

DATE, DATETIME, etc. are tricky to compare against.

Some tempting, but inefficient, techniques:

`date_col LIKE '2016-01%'` – must convert date_col to a string, so acts like a function `LEFT(date_col, 4) = '2016-01'` – hiding the column in function `DATE(date_col) = 2016` – hiding the column in function

All must do a full scan. (On the other hand, it can handy to use GROUP BY LEFT(date_col, 7) for monthly grouping, but that is not an INDEX issue.)

This is efficient, and can use an index:

```
date_col >= '2016-01-01'
    AND date_col  < '2016-01-01' + INTERVAL 3 MONTH
```

This case works because both right-hand values are converted to constants, then it is a “range”. I like the design pattern with INTERVAL because it avoids computing the last day of the month. And it avoids tacking on ‘23:59:59’, which is wrong if you have microsecond times. (And other cases.)

EXPLAIN Key_len

Perform EXPLAIN SELECT… (and EXPLAIN FORMAT=JSON SELECT… if you have 5.6.5). Look at the Key that it chose, and the Key_len. From those you can deduce how many columns of the index are being used for filtering. (JSON makes it easier to get the answer.) From that you can decide whether it is using as much of the INDEX as you thought. Caveat: Key_len only covers the WHERE part of the action; the non-JSON output won’t easily say whether GROUP BY or ORDER BY was handled by the index.

IN

`IN (1,99,3)` is sometimes optimized as efficiently as “=”, but not always. Older versions of MySQL did not optimize it as well as newer versions. (5.6 is possibly the main turning point.)

IN (SELECT …)

From version 4.1 through 5.5, IN (SELECT …) was very poorly optimized. The SELECT was effectively re-evaluated every time. Often it can be transformed into a JOIN, which works much faster. Heres is a pattern to follow:

```
SELECT  ...
    FROM  a
    WHERE  test_a
      AND  x IN (
        SELECT  x
            FROM  b
            WHERE  test_b
                );
⇒
SELECT  ...
    FROM  a
    JOIN  b USING(x)
    WHERE  test_a
      AND  test_b;
```

The SELECT expressions will need “a.” prefixing the column names.

Alas, there are cases where the pattern is hard to follow.

5.6 does some optimizing, but probably not as good as the JOIN.

If there is a JOIN or GROUP BY or ORDER BY LIMIT in the subquery, that complicates the JOIN in new format. So, it might be better to use this pattern:

```
SELECT  ...
    FROM  a
    WHERE  test_a
      AND  x IN ( SELECT  x  FROM ... );
⇒
SELECT  ...
    FROM  a
    JOIN       ( SELECT  x  FROM ... ) b
        USING(x)
    WHERE  test_a;
```

Caveat: If you end up with two subqueries JOINed together, note that neither has any indexes, hence performance can be very bad. (5.6 improves on it by dynamically creating indexes for subqueries.)

There is work going on in MariaDB and Oracle 5.7, in relation to “NOT IN”, “NOT EXISTS”, and “LEFT JOIN..IS NULL”; here is an old discussion on the topic
So, what I say here may not be the final word.

Explode/Implode

When you have a JOIN and a GROUP BY, you may have the situation where the JOIN exploded more rows than the original query (due to many:many), but you wanted only one row from the original table, so you added the GROUP BY to implode back to the desired set of rows.

This explode + implode, itself, is costly. It would be better to avoid them if possible.

Sometimes the following will work.

Using DISTINCT or GROUP BY to counteract the explosion

```
SELECT  DISTINCT
        a.*,
        b.y
    FROM a
    JOIN b
⇒
SELECT  a.*,
        ( SELECT GROUP_CONCAT(b.y) FROM b WHERE b.x = a.x ) AS ys
    FROM a
```

When using second table just to check for existence:

```
SELECT  a.*
    FROM a
    JOIN b  ON b.x = a.x
    GROUP BY a.id
⇒
SELECT  a.*,
    FROM a
    WHERE EXISTS ( SELECT *  FROM b  WHERE b.x = a.x )
```

Another variant

Many-to-many mapping table

Do it this way.

```
CREATE TABLE XtoY (
        # No surrogate id for this table
        x_id MEDIUMINT UNSIGNED NOT NULL,   -- For JOINing to one table
        y_id MEDIUMINT UNSIGNED NOT NULL,   -- For JOINing to the other table
        # Include other fields specific to the 'relation'
        PRIMARY KEY(x_id, y_id),            -- When starting with X
        INDEX      (y_id, x_id)             -- When starting with Y
    ) ENGINE=InnoDB;
```

Notes:

- Lack of an AUTO_INCREMENT id for this table – The PK given is the 'natural' PK; there is no good reason for a surrogate.
- "MEDIUMINT" – This is a reminder that all INTs should be made as small as is safe (smaller ⇒ faster). Of course the declaration here must match the definition in the table being linked to.
- "UNSIGNED" – Nearly all INTs may as well be declared non-negative
- "NOT NULL" – Well, that's true, isn't it?
- "InnoDB" – More efficient than MyISAM because of the way the PRIMARY KEY is clustered with the data in InnoDB.
- "INDEX(y_id, x_id)" – The PRIMARY KEY makes it efficient to go one direction; this index makes the other direction efficient. No need to say UNIQUE; that would be extra effort on INSERTs.
- In the secondary index, saying justINDEX(y_id) would work because it would implicit include x_id. But I would rather make it more obvious that I am hoping for a 'covering' index.

To conditionally INSERT new links, use IODKU

Note that if you had an AUTO_INCREMENT in this table, IODKU would "burn" ids quite rapidly.

Subqueries and UNIONs

Each subquery SELECT and each SELECT in a UNION can be considered separately for finding the optimal INDEX.

Exception: In a "correlated" ("dependent") subquery, the part of the WHERE that depends on the outside table is not easily factored into the INDEX generation. (Cop out!)

JOINs

The first step is to decide what order the optimizer will go through the tables. If you cannot figure it out, then you may need to be pessimistic and create two indexes for each table – one assuming the table will be used first, one assuming that it will come later in the table order.

The optimizer usually starts with one table and extracts the data needed from it. As it finds a useful (that is, matches the WHERE clause, if any) row, it reaches into the 'next' table. This is called NLJ ("Nested Loop Join"). The process of filtering and reaching to the next table continues through the rest of the tables.

The optimizer usually picks the "first" table based on these hints:

- STRAIGHT_JOIN forces the table order.
- The WHERE clause limits which rows needed (whether indexed or not).
- The table to the "left" in a LEFT JOIN usually comes before the "right" table. (By looking at the table definitions, the optimizer may decide that "LEFT" is irrelevant.)
- The current INDEXes will encourage an order.
- etc.

Running EXPLAIN tells you the table order that the Optimizer is very likely to use today. After adding a new INDEX, the optimizer may pick a different table order. You should anticipate the order changing, guess at what order makes

the most sense, and build the INDEXes accordingly. Then rerun EXPLAIN to see if the Optimizer's brain was on the same wavelength you were on.

You should build the INDEX for the "first" table based on any parts of the WHERE, GROUP BY, and ORDER BY clauses that are relevant to it. If a GROUP/ORDER BY mentions a different table, you should ignore that clause.

The second (and subsequent) table will be reached into based on the ON clause. (Instead of using commajoin, please write JOINs with the JOIN keyword and ON clause!) In addition, there could be parts of the WHERE clause that are relevant. GROUP/ORDER BY are not to be considered in writing the optimal INDEX for subsequent tables.

PARTITIONing

PARTITIONing is rarely a substitute for a good INDEX.

PARTITION BY RANGE is a technique that is sometimes useful when indexing fails to be good enough. In a two-dimensional situation such as nearness in a geographical sense, one dimension can partially be handled by partition pruning; then the other dimension can be handled by a regular index (preferably the PRIMARY KEY). This goes into more detail: Find nearest 10 pizza parlors.

FULLTEXT

FULLTEXT is now implemented in InnoDB as well as MyISAM. It provides a way to search for "words" in TEXT columns. This is much faster (when it is applicable) than col LIKE '%word%'.

```
WHERE x = 1
      AND MATCH (...) AGAINST (...)
```

always(?) uses the FULLTEXT index first. That is, the whole Algorithm is invalidated when one of the ANDs is a MATCH.

Signs of a Newbie

- No "compound" (aka "composite") indexes
- No PRIMARY KEY
- Redundant indexes (especially blatant is PRIMARY KEY(id), KEY(id))
- Most or all columns individually indexes ("But I indexed everything")
- "Commajoin" – That is `FROM a, b WHERE a.x=b.x` instead of `FROM a JOIN b ON a.x=b.x`

Speeding up wp_postmeta

The published table (see Wikipedia) is

```
CREATE TABLE wp_postmeta (
      meta_id BIGINT(20) UNSIGNED NOT NULL AUTO_INCREMENT,
      post_id BIGINT(20) UNSIGNED NOT NULL DEFAULT '0',
      meta_key VARCHAR(255) DEFAULT NULL,
      meta_value LONGTEXT,
      PRIMARY KEY (meta_id),
      KEY post_id (post_id),
      KEY meta_key (meta_key)
    ) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
```

The problems:

- The AUTO_INCREMENT provides no benefit; in fact it slows down most queries and clutters disk.
- Much better is PRIMARY KEY(post_id, meta_key) – clustered, handles both parts of usual JOIN.
- BIGINT is overkill, but that can't be fixed without changing other tables.
- VARCHAR(255) can be a problem in 5.6 with utf8mb4; see workarounds below.

- When would `meta_key` or `meta_value` ever be NULL?

The solutions:

```
CREATE TABLE wp_postmeta (
        post_id BIGINT UNSIGNED NOT NULL,
        meta_key VARCHAR(255) NOT NULL,
        meta_value LONGTEXT NOT NULL,
        PRIMARY KEY(post_id, meta_key),
        INDEX(meta_key)
        ) ENGINE=InnoDB;
```

Postlog

Initial posting: March, 2015; Refreshed Feb, 2016; Add DATE June, 2016; Add WP example May, 2017.

The tips in this document apply to MySQL, MariaDB, and Percona.

See also

- Percona 2015 Tutorial Slides
- Some info in the MySQL manual: ORDER BY Optimization
- A short, but complicated, example
- MySQL manual page on range accesses in composite indexes
- Some discussion of JOIN
- Indexing 101: Optimizing MySQL queries on a single table
  (Stephane Combaudon - Percona)
- A complex query, well explained.

This blog is the consolidation of a Percona tutorial I gave in 2013, plus many years of experience in fixing thousands of slow queries on hundreds of systems.
I apologize that this does not tell you how create INDEXes for all SELECTs. Some are just too complex.

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: random

This page is licensed: CC BY-SA / Gnu FDL

Compound (Composite) Indexes

A mini-lesson in "compound indexes" ("composite indexes")

This document starts out trivial and perhaps boring, but builds up to more interesting information, perhaps things you did not realize about how MariaDB and MySQL indexing works.

This also explains EXPLAIN (to some extent).

(Most of this applies to other databases, too.)

The query to discuss

The question is "When was Andrew Johnson president of the US?".

The available table `Presidents` looks like:

```
+-----+------------+----------------+-----------+
| seq | last_name  | first_name     | term      |
+-----+------------+----------------+-----------+
|   1 | Washington | George         | 1789-1797 |
|   2 | Adams      | John           | 1797-1801 |
...
|   7 | Jackson    | Andrew         | 1829-1837 |
...
|  17 | Johnson    | Andrew         | 1865-1869 |
...
|  36 | Johnson    | Lyndon B.      | 1963-1969 |
...
```

(“Andrew Johnson” was picked for this lesson because of the duplicates.)

What index(es) would be best for that question? More specifically, what would be best for

```
SELECT  term
        FROM  Presidents
        WHERE  last_name = 'Johnson'
          AND  first_name = 'Andrew';
```

Some INDEXes to try…

- No indexes
- INDEX(first_name), INDEX(last_name) (two separate indexes)
- “Index Merge Intersect”
- INDEX(last_name, first_name) (a “compound” index)
- INDEX(last_name, first_name, term) (a “covering” index)
- Variants

No indexes

Well, I am fudging a little here. I have a PRIMARY KEY on `seq`, but that has no advantage on the query we are studying.

```
SHOW CREATE TABLE Presidents \G
CREATE TABLE `presidents` (
  `seq` TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `last_name` VARCHAR(30) NOT NULL,
  `first_name` VARCHAR(30) NOT NULL,
  `term` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`seq`)
) ENGINE=InnoDB AUTO_INCREMENT=45 DEFAULT CHARSET=utf8

EXPLAIN  SELECT  term
   FROM  Presidents
   WHERE  last_name = 'Johnson'
   AND  first_name = 'Andrew';
+----+-------------+------------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table      | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+------------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | Presidents | ALL  | NULL          | NULL | NULL    | NULL |   44 | Using where |
+----+-------------+------------+------+---------------+------+---------+------+------+-------------+

# Or, using the other form of display:  EXPLAIN ... \G
           id: 1
  select_type: SIMPLE
        table: Presidents
         type: ALL        <-- Implies table scan
possible_keys: NULL
          key: NULL       <-- Implies that no index is useful, hence table scan
      key_len: NULL
          ref: NULL
         rows: 44         <-- That's about how many rows in the table, so table scan
        Extra: Using where
```

Implementation Details

First, let's describe how InnoDB stores and uses indexes.

- The data and the PRIMARY KEY are "clustered" together in on BTree.
- A BTree lookup is quite fast and efficient. For a million-row table there might be 3 levels of BTree, and the top two levels are probably cached.
- Each secondary index is in another BTree, with the PRIMARY KEY at the leaf.
- Fetching 'consecutive' (according to the index) items from a BTree is very efficient because they are stored consecutively.
- For the sake of simplicity, we can count each BTree lookup as 1 unit of work, and ignore scans for consecutive items. This approximates the number of disk hits for a large table in a busy system.

For MyISAM, the PRIMARY KEY is not stored with the data, so think of it as being a secondary key (over-simplified).

INDEX(first_name), INDEX(last_name)

The novice, once he learns about indexing, decides to index lots of columns, one at a time. But…

MariaDB rarely uses more than one index at a time in a query. So, it will analyze the possible indexes.

- first_name – there are 2 possible rows (one BTree lookup, then scan consecutively)
- last_name – there are 2 possible rows
  Let's say it picks last_name. Here are the steps for doing the SELECT:

1. Using INDEX(last_name), find 2 index entries with last_name = 'Johnson'.
2. Get the PRIMARY KEY (implicitly added to each secondary index in InnoDB); get (17, 36).
3. Reach into the data using seq = (17, 36) to get the rows for Andrew Johnson and Lyndon B. Johnson.
4. Use the rest of the WHERE clause filter out all but the desired row.
5. Deliver the answer (1865-1869).

```
EXPLAIN  SELECT  term
  FROM  Presidents
  WHERE  last_name = 'Johnson'
  AND  first_name = 'Andrew'  \G
  select_type: SIMPLE
        table: Presidents
         type: ref
possible_keys: last_name, first_name
          key: last_name
      key_len: 92                <-- VARCHAR(30) utf8 may need 2+3*30 bytes
          ref: const
         rows: 2                 <-- Two 'Johnson's
        Extra: Using where
```

“Index Merge Intersect”

OK, so you get really smart and decide that MariaDB should be smart enough to use both name indexes to get the answer. This is called “Intersect”.

1. Using INDEX(last_name), find 2 index entries with last_name = ‘Johnson’; get (7, 17)
2. Using INDEX(first_name), find 2 index entries with first_name = ‘Andrew’; get (17, 36)
3. “And” the two lists together (7,17) & (17,36) = (17)
4. Reach into the data using seq = (17) to get the row for Andrew Johnson.
5. Deliver the answer (1865-1869).

```
id: 1
  select_type: SIMPLE
        table: Presidents
         type: index_merge
possible_keys: first_name,last_name
          key: first_name,last_name
      key_len: 92,92
          ref: NULL
         rows: 1
        Extra: Using intersect(first_name,last_name); Using where
```

The EXPLAIN fails to give the gory details of how many rows collected from each index, etc.

INDEX(last_name, first_name)

This is called a “compound” or “composite” index since it has more than one column.

1. Drill down the BTree for the index to get to exactly the index row for Johnson+Andrew; get seq = (17).
2. Reach into the data using seq = (17) to get the row for Andrew Johnson.
3. Deliver the answer (1865-1869).
   This is much better. In fact this is usually the “best”.

```
ALTER TABLE Presidents
        (DROP old indexes AND...)
        ADD INDEX compound(last_name, first_name);

           id: 1
  select_type: SIMPLE
        table: Presidents
         type: ref
possible_keys: compound
          key: compound
      key_len: 184             <-- The length of both fields
          ref: const,const     <-- The WHERE clause gave constants for both
         rows: 1               <-- Goodie!  It homed in on the one row.
        Extra: Using where
```

“Covering”: INDEX(last_name, first_name, term)

Surprise! We can actually do a little better. A “Covering” index is one in which *all* of the fields of the SELECT are found in the index. It has the added bonus of not having to reach into the “data” to finish the task.

1. Drill down the BTree for the index to get to exactly the index row for Johnson+Andrew; get seq = (17).
2. Deliver the answer (1865-1869).
   The “data” BTree is not touched; this is an improvement over “compound”.

```
... ADD INDEX covering(last_name, first_name, term);

           id: 1
  select_type: SIMPLE
        table: Presidents
         type: ref
possible_keys: covering
          key: covering
      key_len: 184
          ref: const,const
         rows: 1
        Extra: Using where; Using index   <-- Note
```

Everything is similar to using “compound”, except for the addition of “Using index”.

Variants

- What would happen if you shuffled the fields in the WHERE clause?
  Answer: The order of ANDed things does not matter.
- What would happen if you shuffled the fields in the INDEX?
  Answer: It may make a huge difference. More in a minute.
- What if there are extra fields on the end?
  Answer: Minimal harm; possibly a lot of good (eg, ‘covering’).
- Reduncancy? That is, what if you have both of these: INDEX(a), INDEX(a,b)?
  Answer: Reduncy costs something on INSERTs; it is rarely useful for SELECTs.
- Prefix? That is, INDEX(last_name(5). first_name(5))
  Answer: Don’t bother; it rarely helps, and often hurts. (The details are another topic.)

More examples:

```
INDEX(last, first)
    ... WHERE last = '...' -- good (even though `first` is unused)
    ... WHERE first = '...' -- index is useless

    INDEX(first, last), INDEX(last, first)
    ... WHERE first = '...' -- 1st index is used
    ... WHERE last = '...' -- 2nd index is used
    ... WHERE first = '...' AND last = '...' -- either could be used equally well

    INDEX(last, first)
    Both of these are handled by that one INDEX:
    ... WHERE last = '...'
    ... WHERE last = '...' AND first = '...'

    INDEX(last), INDEX(last, first)
    In light of the above example, don't bother including INDEX(last).
```

Postlog

Refreshed – Oct, 2012; more links – Nov 2016

See also

- Cookbook on designing the best index for a SELECT
- Sheeri's discussing of Indexes
- Slides on EXPLAIN
- Mysql manual page on range accesses in composite indexes
- Overhead of Composite Indexes
- Size and other limits on Indexes

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: index1

This page is licensed: CC BY-SA / Gnu FDL

Foreign Keys

Overview

A foreign key is a constraint which can be used to enforce data integrity. It is composed of a column or a set of columns in a table, called the *child table*, which references to a column or a set of columns in a table called the *parent table*. With foreign keys, MariaDB performs checks to enforce that integrity rules are always enforced. For a more exhaustive explanation, see Relational databases: Foreign Keys.

> **Note:**
>
> Foreign keys can only be used with storage engines that support them. The default InnoDB supports foreign keys.

> **Warning:**
>
> Partitioned tables cannot contain foreign keys, and cannot be referenced by a foreign key.

Syntax

```
CREATE TABLE b(for_key INT REFERENCES a(not_key));
```

MariaDB applies the constraint if possible. See the Examples below.

Foreign keys are created with CREATE TABLE or ALTER TABLE. The foreign key definition has this syntax:

```
[CONSTRAINT [symbol]] FOREIGN KEY
    [index_name] (index_col_name, ...)
    REFERENCES tbl_name (index_col_name,...)
    [ON DELETE reference_option]
    [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```

> **Note:**
>
> The `symbol` clause is used in error messages and must be unique per database, or, as of MariaDB 12.1, unique per table.
>
> The `symbol` clause is optional. If it isn't specified, MariaDB automatically assigns one.

If MariaDB automatically creates an index for the foreign key (because it does not exist and is not explicitly created), its name is `index_name`.

Requirements and Limitations

Foreign keys have the following requirements:

- Referenced columns in the parent table must be a an index or a prefix of an index.
- Foreign key columns and referenced columns must be of the same type, or similar types. For integer types, the size and sign must also be the same.
- For string types, the foreign key column and the referenced column must use the same character set **and** the same collation. Two columns that differ only in collation, such as `utf8mb4_general_ci` and `utf8mb4_uca1400_ai_ci`, cannot be compared, so a foreign key between them is rejected. See Changing the Character Set or Collation of a Foreign Key Column.
- Both foreign key columns and referenced columns can be PERSISTENT columns. However, the `ON UPDATE CASCADE`, `ON UPDATE SET NULL`, `ON DELETE SET NULL` clauses are not allowed in this case.
- The parent and the child table must use the same storage engine, and must not be `TEMPORARY` or partitioned tables. However, they can be the same table.

Foreign keys in MariaDB have the following limitations:

> **CURRENT**
>
> Foreign key names must be **unique per table**.
>
> **< 12.1**
>
> Foreign key names must be **unique per database**.

- Supported only by InnoDB.
- Cannot be used with views.
- The `SET DEFAULT` action is not supported.
- Foreign key actions do not activate triggers.
- If `ON UPDATE CASCADE` recurses to update the same table it has previously updated during the cascade, it acts like `RESTRICT`.
- Indexed generated columns (both `VIRTUAL` and `PERSISTENT`) are not supported as InnoDB foreign key indexes.
- The columns in the child table must be a `BTREE` index (not `HASH`, `RTREE`, or `FULLTEXT` — see SHOW INDEX), or the leftmost part of a `BTREE` index.
- Index prefixes are not supported, which means that TEXT and BLOB columns cannot be used as foreign keys.

Constraints

If a foreign keys exists, each row in the child table must match a row in the parent table. Multiple child rows can match the same parent row. A child row *matches* a parent row if all its foreign key values are identical to the row values in the parent table. However, if at least one of the foreign key values is `NULL`, the row has no parent, but it is still allowed.

MariaDB performs certain checks to guarantee that data integrity is enforced:

- Trying to insert non-matching rows (or update matching rows in a way that makes them non-matching rows) in the child table produces a 1452 error (SQLSTATE `23000`).
- When a row in the parent table is deleted and at least one child row exists, MariaDB performs an action which depends on the `ON DELETE` clause of the foreign key.
- When a value in the column referenced by a foreign key changes and at least one child row exists, MariaDB performs an action which depends on the `ON UPDATE` clause of the foreign key.
- Trying to drop a table that is referenced by a foreign key produces a 1217 error (SQLSTATE '23000').

- A TRUNCATE TABLE statement against a table containing one or more foreign keys is executed as a DELETE without a `WHERE` clause, so that the foreign keys are enforced for each row.

The allowed actions for `ON DELETE` and `ON UPDATE` are:

- `RESTRICT`: The change on the parent table is prevented. The statement terminates with a 1451 error (SQLSTATE '2300'). This is the default behavior for both `ON DELETE` and `ON UPDATE`.
- `NO ACTION`: Synonym for `RESTRICT`.
- `CASCADE`: The change is allowed and propagates on the child table. For example, if a parent row is deleted, the child row is also deleted; if a parent row's ID changes, the child row's ID changes, too.
- `SET NULL`: The change is allowed, and the child row's foreign key columns are set to `NULL`.
- `SET DEFAULT`: This clause is not supported.

`DELETE` or `UPDATE` statements triggered by foreign keys do not activate triggers and are not counted in the Com_delete and Com_update status variables.

Foreign key constraints can be disabled by setting the foreign_key_checks server system variable to `0`. This speeds up the insertion of large quantities of data.

Setting `foreign_key_checks` to `0` suspends the checks on data, but it does not permit schema changes that would leave the constraint itself invalid. Regardless of the setting, a column used in a foreign key cannot be dropped, renamed, or changed in a way that alters how its values compare, including a change of type, character set, or collation. See Changing the Character Set or Collation of a Foreign Key Column.

> **Note:**
>
> For detailed information about constraints, see this page.

Changing the Character Set or Collation of a Foreign Key Column

Because a foreign key requires both string columns to share a character set and collation, converting one table without the other breaks the constraint. MariaDB refuses the conversion rather than allowing it:

- Converting the child table fails with error 1832, `Cannot change column '...': used in a foreign key constraint '...'`.
- Converting the parent table fails with error 1833, `Cannot change column '...': used in a foreign key constraint '...' of table '...'`.

> **Warning:**
>
> Setting `foreign_key_checks` to `0` does not lift this restriction, and neither does requesting `ALGORITHM=COPY` or `ALGORITHM=INPLACE`. The conversion is rejected in every combination of the two. This is deliberate: allowing it would leave a foreign key whose columns can no longer be compared, producing a schema that cannot be restored from a dump.

Instead, drop the constraint, convert both tables, and re-create the constraint. Dropping it in the **same** `ALTER TABLE` statement as the column change is supported, so the child table needs only one statement:

```
SET FOREIGN_KEY_CHECKS=0;

ALTER TABLE child
  DROP FOREIGN KEY fk_name,
  MODIFY col VARCHAR(24) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;

ALTER TABLE parent
  MODIFY col VARCHAR(24) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;

ALTER TABLE child
  ADD CONSTRAINT fk_name FOREIGN KEY (col) REFERENCES parent (col);

SET FOREIGN_KEY_CHECKS=1;
```

The order matters. The parent table cannot be converted while a child foreign key still references it, and a child's constraint cannot be dropped from within the parent's `ALTER TABLE`. Across a whole schema, drop every affected constraint first, convert all the tables, then re-create the constraints.

Re-creating a constraint while `foreign_key_checks` is `0` does not validate the existing rows against it. To have them validated, restore `foreign_key_checks` to `1` before the `ADD CONSTRAINT` statement.

To list the constraints that need dropping and re-creating, query the KEY_COLUMN_USAGE table:

```
SELECT TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME,
       REFERENCED_TABLE_NAME, REFERENCED_COLUMN_NAME
  FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
 WHERE TABLE_SCHEMA = DATABASE()
   AND REFERENCED_TABLE_NAME IS NOT NULL;
```

Note:

Converting a table to a different collation changes how its values compare and sort, which also changes what counts as a duplicate in a unique index. Where a conversion is only needed to make two tables match, converting the older tables **forward** to the newer collation is usually preferable to moving the newer ones back. See Setting Character Sets and Collations.

Metadata

The REFERENTIAL_CONSTRAINTS Information Schema table contains information about foreign keys:

```
MariaDB [information_schema]> SELECT * FROM REFERENTIAL_CONSTRAINTS \G
*************************** 1. row ***************************
       CONSTRAINT_CATALOG: def
        CONSTRAINT_SCHEMA: nation
          CONSTRAINT_NAME: countries_ibfk_1
UNIQUE_CONSTRAINT_CATALOG: def
 UNIQUE_CONSTRAINT_SCHEMA: nation
   UNIQUE_CONSTRAINT_NAME: PRIMARY
            MATCH_OPTION: NONE
             UPDATE_RULE: RESTRICT
             DELETE_RULE: RESTRICT
              TABLE_NAME: countries
    REFERENCED_TABLE_NAME: regions
*************************** 2. row ***************************
       CONSTRAINT_CATALOG: def
        CONSTRAINT_SCHEMA: nation
          CONSTRAINT_NAME: regions_ibfk_1
UNIQUE_CONSTRAINT_CATALOG: def
 UNIQUE_CONSTRAINT_SCHEMA: nation
   UNIQUE_CONSTRAINT_NAME: PRIMARY
            MATCH_OPTION: NONE
             UPDATE_RULE: RESTRICT
             DELETE_RULE: RESTRICT
              TABLE_NAME: regions
    REFERENCED_TABLE_NAME: continents
...
```

The individual columns are listed in the KEY_COLUMN_USAGE table:

```
MariaDB [information_schema]> SELECT * FROM KEY_COLUMN_USAGE LIMIT 2 \G
*************************** 1. row ***************************
           CONSTRAINT_CATALOG: def
            CONSTRAINT_SCHEMA: nation
              CONSTRAINT_NAME: PRIMARY
                TABLE_CATALOG: def
                 TABLE_SCHEMA: nation
                   TABLE_NAME: countries
                  COLUMN_NAME: country_id
             ORDINAL_POSITION: 1
POSITION_IN_UNIQUE_CONSTRAINT: NULL
      REFERENCED_TABLE_SCHEMA: NULL
        REFERENCED_TABLE_NAME: NULL
       REFERENCED_COLUMN_NAME: NULL
*************************** 2. row ***************************
           CONSTRAINT_CATALOG: def
            CONSTRAINT_SCHEMA: nation
              CONSTRAINT_NAME: country_code2
                TABLE_CATALOG: def
                 TABLE_SCHEMA: nation
                   TABLE_NAME: countries
                  COLUMN_NAME: country_code2
             ORDINAL_POSITION: 1
POSITION_IN_UNIQUE_CONSTRAINT: NULL
      REFERENCED_TABLE_SCHEMA: NULL
        REFERENCED_TABLE_NAME: NULL
       REFERENCED_COLUMN_NAME: NULL
```

The InnoDB-specific Information Schema tables also contain information about the InnoDB foreign keys. The foreign key information is stored in the INNODB_SYS_FOREIGN table. Data about the individual columns are stored in INNODB_SYS_FOREIGN_COLS.

Another way of retrieving information about a table's foreign keys is the SHOW CREATE TABLE statement:

```
MariaDB [nation]> SHOW CREATE TABLE countries \G
*************************** 1. row ***************************
       Table: countries
Create Table: CREATE TABLE `countries` (
  `country_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) DEFAULT NULL,
  `area` decimal(10,2) NOT NULL,
  `national_day` date DEFAULT NULL,
  `country_code2` char(2) NOT NULL,
  `country_code3` char(3) NOT NULL,
  `region_id` int(11) NOT NULL,
  PRIMARY KEY (`country_id`),
  UNIQUE KEY `country_code2` (`country_code2`),
  UNIQUE KEY `country_code3` (`country_code3`),
  KEY `region_id` (`region_id`),
  CONSTRAINT `countries_ibfk_1` FOREIGN KEY (`region_id`) REFERENCES `regions` (`region_id`)
) ENGINE=InnoDB AUTO_INCREMENT=240 DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_uca1400_ai_ci
```

Examples

Creating and Using Foreign Keys

In this example, we create an `author` and a `book` table, both having a primary key called `id`. The `book` table also has a foreign key composed from a field called `author_id`, which refers to the `author` table primary key. The foreign key constraint name is optional, but we specify it because we want it to appear in error messages: `fk_book_author`.

```
CREATE TABLE author (
  id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL
) ENGINE = InnoDB;

CREATE TABLE book (
  id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(200) NOT NULL,
  author_id SMALLINT UNSIGNED NOT NULL,
  CONSTRAINT `fk_book_author`
    FOREIGN KEY (author_id) REFERENCES author (id)
    ON DELETE CASCADE
    ON UPDATE RESTRICT
) ENGINE = InnoDB;
```

When trying to insert a book with a non-existing author, we get an error:

```
INSERT INTO book (title, author_id) VALUES ('Necronomicon', 1);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
 (`test`.`book`, CONSTRAINT `fk_book_author` FOREIGN KEY (`author_id`)
  REFERENCES `author` (`id`) ON DELETE CASCADE)
```

The error basically says that `author_id` is mandatory because otherwise the foreign key constraint is violated. So, let's insert two authors and their books:

```
INSERT INTO author (name) VALUES ('Abdul Alhazred');
INSERT INTO book (title, author_id) VALUES ('Necronomicon', LAST_INSERT_ID());

INSERT INTO author (name) VALUES ('H.P. Lovecraft');
INSERT INTO book (title, author_id) VALUES
  ('The call of Cthulhu', LAST_INSERT_ID()),
  ('The colour out of space', LAST_INSERT_ID());
```

Those `INSERT` statements first add an author name to the `name` column of the author table, and the `id` column is automatically updated thanks to `AUTO_INCREMENT`. For the next `INSERT` statement (adding the book or books), we

use the `LAST_INSERT_ID()` function, which uses that newly generated ID, filling it in for the `book` table.

Now, let's delete the second author from the `author` table. When we created the foreign key, we specified `ON DELETE CASCADE`. This propagates the deletion to the `book` table, making the deleted author's books disappear, too:

```
DELETE FROM author WHERE name = 'H.P. Lovecraft';

SELECT * FROM book;
+----+--------------+-----------+
| id | title        | author_id |
+----+--------------+-----------+
|  3 | Necronomicon |         1 |
+----+--------------+-----------+
```

We also specified `ON UPDATE RESTRICT`. This prevents modifying an author's `id` (the column referenced by the foreign key) if a child row exists:

```
UPDATE author SET id = 10 WHERE id = 1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
 (`test`.`book`, CONSTRAINT `fk_book_author` FOREIGN KEY (`author_id`)
  REFERENCES `author` (`id`) ON DELETE CASCADE)
```

Referencing Keys in Tables

This example demonstrates which keys to reference for foreign keys, and what happens when the wrong key (`not_key`) is referenced (error 1005):

```
CREATE TABLE a(a_key INT PRIMARY KEY, not_key INT);

CREATE TABLE b(for_key INT REFERENCES a(not_key));
ERROR 1005 (HY000): Can't create table `test`.`b`
  (errno: 150 "Foreign key constraint is incorrectly formed")

CREATE TABLE c(for_key INT REFERENCES a(a_key));

SHOW CREATE TABLE c;
+-------+-----------------------------------------------------------------------------------+
| Table | Create Table                                                                      |
+-------+-----------------------------------------------------------------------------------+
| c     | CREATE TABLE `c` (
  `for_key` INT(11) DEFAULT NULL,
  KEY `for_key` (`for_key`),
  CONSTRAINT `c_ibfk_1` FOREIGN KEY (`for_key`) REFERENCES `a` (`a_key`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-------+-----------------------------------------------------------------------------------+

INSERT INTO a VALUES (1,10);
Query OK, 1 row affected (0.004 sec)

INSERT INTO c VALUES (10);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
  (`test`.`c`, CONSTRAINT `c_ibfk_1` FOREIGN KEY (`for_key`) REFERENCES `a` (`a_key`))

INSERT INTO c VALUES (1);
Query OK, 1 row affected (0.004 sec)

SELECT * FROM c;
+---------+
| for_key |
+---------+
|       1 |
+---------+
```

This page is licensed: CC BY-SA / Gnu FDL

Ignored Indexes

> **Note:**
>
> This feature is available from MariaDB 10.6.

Ignored indexes are indexes that are visible and maintained, but which are not used by the optimizer. MySQL 8 has a similar feature which they call “invisible indexes”.

Syntax

By default, an index is not ignored. One can mark existing index as ignored (or not ignored) with an ALTER TABLE statement:

```
ALTER TABLE table_name ALTER {KEY|INDEX} [IF EXISTS] key_name [NOT] IGNORED;
```

It is also possible to specify IGNORED attribute when creating an index with a CREATE TABLE, or CREATE INDEX statement:

```
CREATE TABLE table_name (
  ...
  INDEX index_name ( ...) [NOT] IGNORED
  ...
```

```
CREATE INDEX index_name (...) [NOT] IGNORED ON tbl_name (...);
```

table’s primary key cannot be ignored. This applies to both explicitly defined primary key, as well as implicit primary key - if there is no explicit primary key defined but the table has a unique key containing only NOT NULL columns, the first of such keys becomes the implicitly defined primary key.

Handling for ignored indexes

The optimizer will treats ignored indexes as if they didn’t exist. They will not be used in the query plans, or as a source of statistical information.
Also, an attempt to use an ignored index in a `USE INDEX`, `FORCE INDEX`, or `IGNORE INDEX` hint will result in an error - the same what would have if one used a name of a non-existent index.

Information about whether or not indexes are ignored can be viewed in the IGNORED column in the Information Schema STATISTICS table or the SHOW INDEX statement.

Intended Usage

The primary use case is as follows: a DBA sees an index that seems to have little or no usage and considers whether to remove it. Dropping the index is a risk as it may still be needed in a few cases. For example, the optimizer may rely on the estimates provided by the index without using the index in query plans. If dropping an index causes an issue, it will take a while to re-create the index. On the other hand, marking the index as ignored (or not ignored) is instant, so the suggested workflow is:

1. Mark the index as ignored
2. Check if everything continues to work
3. If not, mark the index as not ignored.
4. If everything continues to work, one can safely drop the index.

Examples

```
CREATE TABLE t1 (id INT PRIMARY KEY, b INT, KEY k1(b) IGNORED);
```

```
CREATE OR REPLACE TABLE t1 (id INT PRIMARY KEY, b INT, KEY k1(b));
ALTER TABLE t1 ALTER INDEX k1 IGNORED;
```

```
CREATE OR REPLACE TABLE t1 (id INT PRIMARY KEY, b INT);
CREATE INDEX k1 ON t1(b) IGNORED;
```

```
SELECT * FROM INFORMATION_SCHEMA.STATISTICS WHERE TABLE_NAME = 't1'\G
*************************** 1. row ***************************
TABLE_CATALOG: def
 TABLE_SCHEMA: test
   TABLE_NAME: t1
   NON_UNIQUE: 0
 INDEX_SCHEMA: test
   INDEX_NAME: PRIMARY
 SEQ_IN_INDEX: 1
  COLUMN_NAME: id
    COLLATION: A
  CARDINALITY: 0
     SUB_PART: NULL
       PACKED: NULL
     NULLABLE:
   INDEX_TYPE: BTREE
      COMMENT:
INDEX_COMMENT:
      IGNORED: NO
*************************** 2. row ***************************
TABLE_CATALOG: def
 TABLE_SCHEMA: test
   TABLE_NAME: t1
   NON_UNIQUE: 1
 INDEX_SCHEMA: test
   INDEX_NAME: k1
 SEQ_IN_INDEX: 1
  COLUMN_NAME: b
    COLLATION: A
  CARDINALITY: 0
     SUB_PART: NULL
       PACKED: NULL
     NULLABLE: YES
   INDEX_TYPE: BTREE
      COMMENT:
INDEX_COMMENT:
      IGNORED: YES
```

```
SHOW INDEXES FROM t1\G
*************************** 1. row ***************************
        Table: t1
   Non_unique: 0
     Key_name: PRIMARY
 Seq_in_index: 1
  Column_name: id
    Collation: A
  Cardinality: 0
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
      Ignored: NO
*************************** 2. row ***************************
        Table: t1
   Non_unique: 1
     Key_name: k1
 Seq_in_index: 1
  Column_name: b
    Collation: A
  Cardinality: 0
     Sub_part: NULL
       Packed: NULL
         Null: YES
   Index_type: BTREE
      Comment:
Index_comment:
      Ignored: YES
```

The optimizer does not make use of an index when it is ignored, while if the index is not ignored (the default), the optimizer will consider it in the optimizer plan, as shown in the EXPLAIN output.

```
CREATE OR REPLACE TABLE t1 (id INT PRIMARY KEY, b INT, KEY k1(b) IGNORED);

EXPLAIN SELECT * FROM t1 ORDER BY b;
+------+-------------+-------+------+---------------+------+---------+------+------+----------------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra          |
+------+-------------+-------+------+---------------+------+---------+------+------+----------------+
|    1 | SIMPLE      | t1    | ALL  | NULL          | NULL | NULL    | NULL | 1    | Using filesort |
+------+-------------+-------+------+---------------+------+---------+------+------+----------------+

ALTER TABLE t1 ALTER INDEX k1 NOT IGNORED;

EXPLAIN SELECT * FROM t1 ORDER BY b;
+------+-------------+-------+-------+---------------+------+---------+------+------+-------------+
| id   | select_type | table | type  | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+-------+-------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | t1    | index | NULL          | k1   | 5       | NULL | 1    | Using index |
+------+-------------+-------+-------+---------------+------+---------+------+------+-------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Index Statistics

How Index Statistics Help the Query Optimizer

Understanding index statistics is crucial for the MariaDB query optimizer to efficiently execute queries. Accurate and current statistics guide the optimizer in choosing the best way to access data, similar to using a personal address book for quicker searches rather than a larger phone book. Up-to-date index statistics ensure optimized query performance.

Value Groups

The statistics primarily focus on groups of index elements with identical values. In a primary key, each index is unique, resulting in a group size of one. In a non-unique index, multiple keys may share the same value. The worst-case scenario involves large groups with identical values, such as an index on a boolean field.

MariaDB makes heavy use of the average group size statistic. For example, if there are 100 rows, and twenty groups with the same index values, the average group size would be five.

However, averages can be skewed by extremes, and the usual culprit is NULL values. The row of 100 may have 19 groups with an average size of 1, while the other 81 values are all NULL. MariaDB may think five is a good average size and choose to use that index, and then end up having to read through 81 rows with identical keys, taking longer than an alternative.

Dealing with NULLs

There are three main approaches to the problem of NULLs. NULL index values can be treated as a single group (`nulls_equal`). This is usually fine, but if you have large numbers of NULLs the average group size is slanted higher, and the optimizer may miss using the index for ref accesses when it would be useful. This is the default used by InnoDB. The opposite approach is `nulls_unequal`, with each NULL forming its own group of one. Conversely, the average group size is slanted lower, and the optimizer may use the index for ref accesses when not suitable. This is the default used by the Aria and MyISAM storage engines. A third option, `nulls_ignored`, sees NULLs ignored altogether from index group calculations.

The default approaches can be changed by setting the aria_stats_method, myisam_stats_method and innodb_stats_method server variables.

Null-Safe and Regular Comparisons

The comparison operator used plays an important role. If two values are compared with <=> (see the null-safe-equal comparison operator), and both are null, 1 is returned. If the same values are compared with = (see the equal comparison operator) null is returned. For example:

```
SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;
+---------+---------------+------------+
| 1 <=> 1 | NULL <=> NULL | 1 <=> NULL |
+---------+---------------+------------+
|       1 |             1 |          0 |
+---------+---------------+------------+

SELECT 1 = 1, NULL = NULL, 1 = NULL;
+-------+-------------+----------+
| 1 = 1 | NULL = NULL | 1 = NULL |
+-------+-------------+----------+
|     1 |        NULL |     NULL |
+-------+-------------+----------+
```

Engine-Independent Statistics

MariaDB 10.0.1 introduced a way to gather statistics independently of the storage engine. See Engine-independent table statistics.

Histogram-Based Statistics

Histogram-Based Statistics were introduced in MariaDB 10.0.2, and are collected by default from MariaDB 10.4.3.

See Also

- User Statistics. This plugin provides user, client, table and index usage statistics.
- InnoDB Persistent Statistics

- Engine-independent Statistics
- Histogram-based Statistics
- Ignored Indexes

This page is licensed: CC BY-SA / Gnu FDL

Latitude/Longitude Indexing

The problem

You want to find the nearest 10 pizza parlors, but you cannot figure out how to do it efficiently in your huge database. Database indexes are good at one-dimensional indexing, but poor at two-dimensions.

You might have tried

- INDEX(lat), INDEX(lon) – but the optimizer used only one
- INDEX(lat,lon) – but it still had to work too hard
- Sometimes you ended up with a full table scan – Yuck.

WHERE SQRT(…)< … – No chance of using any index.

WHERE lat BETWEEN … AND lng BETWEEN… – This has some chance of using such indexes.

The goal is to look only at records “close”, in both directions, to the target lat/lng.

A solution – first, the principles

PARTITIONs in MariaDB and MySQL sort of give you a way to have two clustered indexes. So, if we could slice up (partition) the globe in one dimension and use ordinary indexing in the other dimension, maybe we can get something approximating a 2D index. This 2D approach keeps the number of disk hits significantly lower than 1D approaches, thereby speeding up “find nearest” queries.

It works. Not perfectly, but better than the alternatives.

What to PARTITION on? It seems like latitude or longitude would be a good idea. Note that longitudes vary in width, from 69 miles (111 km) at the equator, to 0 at the poles. So, latitude seems like a better choice.

How many PARTITIONs? It does not matter a lot. Some thoughts:

- 90 partitions - 2 degrees each. (I don’t like tables with too many partitions; 90 seems like plenty.)
- 50-100 - evenly populated. (This requires code. For 2.7M placenames, 85 partitions varied from 0.5 degrees to very wide partitions at the poles.)
- Don’t have more than 100 partitions, there are inefficiencies in the partition implementation.

How to PARTITION? Well, MariaDB and MySQL are very picky. So FLOAT/DOUBLE are out. DECIMAL is out. So, we are stuck with some kludge. Essentially, we need to convert Lat/Lng to some size of INT and use PARTITION BY RANGE.

Representation choices

To get to a datatype that can be used in PARTITION, you need to “scale” the latitude and longitude. (Consider only the *INTs; the other datatypes are included for comparison)

```
Datatype           Bytes       resolution
   ------------------ -----  --------------------------------
   Deg*100 (SMALLINT)     4  1570 m    1.0 mi  Cities
   DECIMAL(4,2)/(5,2)     5  1570 m    1.0 mi  Cities
   SMALLINT scaled        4   682 m    0.4 mi  Cities
   Deg*10000 (MEDIUMINT)  6    16 m     52 ft  Houses/Businesses
   DECIMAL(6,4)/(7,4)     7    16 m     52 ft  Houses/Businesses
   MEDIUMINT scaled       6   2.7 m    8.8 ft
   FLOAT                  8   1.7 m    5.6 ft
   DECIMAL(8,6)/(9,6)     9    16cm    1/2 ft  Friends in a mall
   Deg*10000000 (INT)     8    16mm    5/8 in  Marbles
   DOUBLE                16   3.5nm     ...    Fleas on a dog
```

(Sorted by resolution)

What these mean…

Deg*100 (SMALLINT) – you take the lat/lng, multiply by 100, round, and store into a SMALLINT. That will take 2 bytes for each dimension, for a total of 4 bytes. Two items might be 1570 meters apart, but register as having the same latitude and longitude.

DECIMAL(4,2) for latitude and DECIMAL(5,2) for longitude will take 2+3 bytes and have no better resolution than Deg*100.

SMALLINT scaled – Convert latitude into a SMALLINT SIGNED by doing (degrees / 90 * 32767) and rounding; longitude by (degrees / 180 * 32767).

FLOAT has 24 significant bits; DOUBLE has 53. (They don't work with PARTITIONing but are included for completeness. Often people use DOUBLE without realizing how much an overkill it is, and how much space it takes.)

Sure, you could do DEG_1000 and other "in between" cases, but there is no advantage. DEG_1000 takes as much space as DEG*10000, but has less resolution.

So, go down the list to see how much resolution you need, then pick an encoding you are comfortable with. However, since we are about to use latitude as a "partition key", it must be limited to one of the INTs. For the sample code, I will use Deg*10000 (MEDIUMINT).

GCDist – compute "great circle distance"

GCDist is a helper FUNCTION that correctly computes the distance between two points on the globe.

The code has been benchmarked at about 20 microseconds per call on a 2011-vintage PC. If you had to check a million points, that would take 20 seconds – far too much for a web application. So, one goal of the Procedure that uses it will be to minimize the usage of this function. With the code presented here, the function need be called only a few dozen or few hundred times, except in pathological cases.

Sure, you could use the Pythagorean formula. And it would work for most applications. But it does not take extra effort to do the GC. Furthermore, GC works across a pole and across the dateline. And, a Pythagorean function is not that much faster.

For efficiency, GCDist understands the scaling you picked and has that stuff hardcoded. I am picking "Deg*10000", so the function expects 350000 for representing 35 degrees. If you choose a different scaling, you will need to change the code.

GCDist() takes 4 scaled DOUBLEs – lat1, lon1, lat2, lon2 – and returns a scaled number of "degrees" representing the distance.

The table of representation choices says 52 feet of resolution for Deg*10000 and DECIMAL(x,4). Here is how it was calculated: To measuring a diagonal between lat/lng (0,0) and (0.0001,00001) (one 'unit in the last place'):

GCDist(0,0,1,1) * 69.172 / 10000 * 5280 = 51.65, where

- 69.172 miles/degree of latitude
- 10000 units per degree for the scaling chosen
- 5280 feet / mile.

(No, this function does not compensate for the Earth being an oblate spheroid, etc.)

Required table structure

There will be one table (plus normalization tables as needed). The one table must be partitioned and indexed as indicated below.

Fields and indexes

- PARTITION BY RANGE(lat)
- lat – scaled latitude (see above)
- lon – scaled longitude
- PRIMARY KEY(lon, lat, …) – lon must be first; something must be added to make it UNIQUE
- id – (optional) you may need to identify the rows for your purposes; AUTO_INCREMENT if you like
- INDEX(id) – if `id` is AUTO_INCREMENT, then this plain INDEX (not UNIQUE, not PRIMARY KEY) is necessary
- ENGINE=InnoDB – so the PRIMARY KEY will be “clustered”
- Other indexes – keep to a minimum (this is a general performance rule for large tables)

For most of this discussion, lat is assumed to be MEDIUMINT – scaled from -90 to +90 by multiplying by 10000. Similarly for lon and -180 to +180.

The PRIMARY KEY must

- start with `lon` since the algorithm needs the “clustering” that InnoDB will provide, and
- include `lat` somewhere, since it is the PARTITION key, and
- contain something to make the key UNIQUE (lon+lat is unlikely to be sufficient).

The FindNearest PROCEDURE will do multiple SELECTs something like this:

```
WHERE lat    BETWEEN @my_lat - @dlat
                     AND @my_lat + @dlat   -- PARTITION Pruning and bounding box
        AND lon    BETWEEN @my_lon - @dlon
                     AND @my_lon + @dlon   -- first part of PK
        AND condition                      -- filter out non-pizza parlors
```

The query planner will

- Do PARTITION “pruning” based on the latitude; then
- Within a PARTITION (which is effectively a table), use lon do a ‘clustered’ range scan; then
- Use the “condition” to filter down to the rows you desire, plus recheck lat.
  This design leads to very few disk blocks needing to be read, which is the main goal of the design.

Note that this does not even call GCDist. That comes in the last pass when the ORDER BY and LIMIT are used.

The stored procedure has a loop. At least two SELECTs will be executed, but with proper tuning; usually no more than about 6 SELECTs will be performed. Because of searching by the PRIMARY KEY, each SELECT hits only one block, sometimes more, of the table. Counting the number of blocks hit is a crude, but effective way, of comparing the performance of multiple designs. By comparison, a full table scan will probably touch thousands of blocks. A simple INDEX(lat) probably leads to hitting hundreds of blocks.

Filtering… An argument to the FindNearest procedure includes a boolean expression (“condition”) for a WHERE clause. If you don’t need any filtering, pass in “1”. To avoid “SQL injection”, do not let web users put arbitrary expressions; instead, construct the “condition” from inputs they provide, thereby making sure it is safe.

The algorithm

The algorithm is embodied in a stored procedure because of its complexity.

- You feed it a starting width for a “square” and a number of items to find.
- It builds a “square” around where you are.
- A SELECT is performed to see how many items are in the square.
- Loop, doubling the width of the square, until enough items are found.
- Now, a ‘last’ SELECT is performed to get the exact distances, sort them (ORDER BY) and LIMIT to the desired number.
- If spanning a pole or the dateline, a more complex SELECT is used.

The next section (“Performance”) should make this a bit clearer as it walks through some examples.

Performance

Because of all the variations, it is hard to get a meaningful benchmark. So, here is some hand-waving instead.

Each SELECT is constrained by a “square” defined by a latitude range and a longitude range. (See the WHERE clause mentioned above, or in the sample code below.) Because of the way longitude lines warp, the longitude range of the “square” will be more degrees than the latitude range. Let’s say the latitude partitioning is 3 degrees wide in the area where you are searching. That is over 200 miles (over 300km), so you are very likely to have a latitude range smaller than the partition width. Still, if you are reaching from the edge of a latitude stripe, the square could span two partitions. After partition pruning down to one (sometimes more) partition, the query is then constrained by a longitude range. (Remember, the PRIMARY KEY starts with `lon`.) If an InnoDB data block contains 100 rows (a handy Rule of Thumb), the select will touch one (or a few) block. If the square spans two (or more) partitions, then the same logic applies to each partition.

So, scanning the square will involve as little as one block; rarely more than a few blocks. The number of blocks is mostly independent of the dataset size.

The primary use case for this algorithm is when the data is significantly larger than will fit into cache (the buffer_pool). Hence, the main goal is to minimize the number of disk hits.

Now let’s look at some edge cases, and argue that the number of blocks is still better (usually) than with traditional indexing techniques.

What if you are looking for Starbucks in a dense city? There would be dozens, maybe hundreds per square mile. If you start the guess at 100 miles, the SELECTs would be hitting lots of blocks – not efficient. In this case, the “starting distance” should be small, say, 2 miles. Let’s say your app wants the closest 10 stores. In this example, you would probably find more than 10 Starbucks within 2 miles in 1 InnoDB block in one partition. Even though there is a second SELECT to finish off the query, it would be hitting the same block. Total: One block hit == cheap.

Let’s say you start with a 5 mile square. Since there are upwards of 200 Starbucks within a 5-miles radius in some dense cities of the world, that might imply 300 in our “square”. That maps to about 4 disk blocks, and a modest amount of CPU to chew through the 300 records. Still not bad.

Now, suppose you are on an ocean liner somewhere in the Pacific. And there is one Starbucks onboard, but you are looking for the nearest 10. If you again start with 2 miles, it will take several iterations to find 10 sites. But, let’s walk through it anyway. The first probe will hit one partition (maybe 2), and find just one hit. The second probe doubles the width of the square; 4 miles will still give you one hit – the same hit in the same block, which is now cached, so we won’t count it as a second disk I/O. Eventually the square will be wide enough to span multiple partitions. Each

extra partition will be one new disk hit to discover no sites in the square. Finally, the square will hit Chile or Hawaii or Fiji and find some more sites, perhaps enough to stop the iteration. Since the main criteria in determining the number of disk hits is the number of partitions hit, we do not want to split the world into too many partitions. If there are, say, 40 partitions, then I have just described a case where there might be 20 disk hits.

2-degree partitions might be good for a global table of stores or restaurants. A 5-mile starting distance might be good when filtering for Starbucks. 20 miles might be better for a department store.

Now, let's discuss the 'last' SELECT, wherein the square is expanded by SQRT(2) and it uses the Great Circle formula to precisely order the N results. The SQRT(2) is in case that the N items were all at the corners of the 'square'. Growing the square by this much allows us to catch any other sites that were just outside the old square.

First, note that this 'last' SELECT is hitting the same block(s) that the iteration hit, plus possibly hitting some more blocks. It is hard to predict how many extra blocks might be hit. Here's a pathological case. You are in the middle of a desert; the square grows and grows. Eventually it finds N sites. There is a big city just outside the final square from the iterating. Now the 'last' SELECT kicks in, and it includes lots of sites in this big city. "Lots of sites" –> lots of blocks –> lots of disk hits.

Discussion of reference code

Here's the gist of the stored procedure FindNearest().

- Make a guess at how close to "me" to look.
- See how many items are in a 'square' around me, after filtering.
- If not enough, repeat, doubling the width of the square.
- After finding enough, or giving up because we are looking "too far", make one last pass to get all the data, ORDERed and LIMITed

Note that the loop merely uses 'squares' of lat/lng ranges. This is crude, but works well with the partitioning and indexing, and avoids calling to GCDist (until the last step). In the sample code, I picked 15 miles as starting value. Adjusting this will have some impact on the Procedure's performance, but the impact will vary with the use cases. A rough way to set the radius is to guess what will find the desired LIMIT about half the time. (This value is hardcoded in the PROCEDURE.)

Parameters passed into FindNearest():

- your Latitude – -90..90 (not scaled – see hardcoded conversion in PROCEDURE)
- your Longitude – -180..180 (not scaled)
- Start distance – (miles or km) – see discussion below
- Max distance – in miles or km – see hardcoded conversion in PROCEDURE
- Limit – maximum number of items to return
- Condition – something to put after 'AND' (more discussion above)

The function will find the nearest items, up to Limit that meet the Condition. But it will give up at Max distance. (If you are at the South Pole, why bother searching very far for the tenth pizza parlor?)

Because of the "scaling", "hardcoding", "Condition", the table name, etc, this PROCEDURE is not truly generic; the code must be modified for each application. Yes, I could have designed it to pass all that stuff in. But what a mess.

The "_start_dist" gives some control over the performance. Making this too small leads to extra iterations; too big leads to more rows being checked. If you choose to tune the Stored Procedure, do the following. "SELECT @iterations" after calling the SP for a number of typical values. If the value is usually 1, then decrease _start_dist. If it is usually 2 or more, then increase it.

Timing: Under 10ms for "typical" usage; any dataset size. Slower for pathological cases (low min distance, high max distance, crossing dateline, bad filtering, cold cache, etc)

End-cases:

- By using GC distance, not Pythagoras, distances are ‘correct’ even near poles.
- Poles – Even if the “nearest” is almost 360 degrees away (longitude), it can find it.
- Dateline – There is a small, ‘contained’, piece of code for crossing the Dateline. Example: you are at +179 deg longitude, and the nearest item is at -179.

The procedure returns one resultset, SELECT *, distance.

- Only rows that meet your Condition, within Max distance are returned
- At most Limit rows are returned
- The rows will be ordered, “closest” first.
- “dist” will be in miles or km (based on a hardcoded constant in the SP)

Reference code, assuming deg*10000 and ‘miles’

This version is based on scaling “Deg*10000 (MEDIUMINT)”.

```
DELIMITER //

DROP function IF EXISTS GCDist //
CREATE FUNCTION GCDist (
        _lat1 DOUBLE,  -- Scaled Degrees north for one point
        _lon1 DOUBLE,  -- Scaled Degrees west for one point
        _lat2 DOUBLE,  -- other point
        _lon2 DOUBLE
    ) RETURNS DOUBLE
    DETERMINISTIC
    CONTAINS SQL  -- SQL but does not read or write
    SQL SECURITY INVOKER  -- No special privileges granted
-- Input is a pair of latitudes/longitudes multiplied by 10000.
--    For example, the south pole has latitude -900000.
-- Multiply output by .0069172 to get miles between the two points
--    or by .0111325 to get kilometers
BEGIN
    -- Hardcoded constant:
    DECLARE _deg2rad DOUBLE DEFAULT PI()/1800000;  -- For scaled by 1e4 to MEDIUMINT
    DECLARE _rlat1 DOUBLE DEFAULT _deg2rad * _lat1;
    DECLARE _rlat2 DOUBLE DEFAULT _deg2rad * _lat2;
    -- compute as if earth's radius = 1.0
    DECLARE _rlond DOUBLE DEFAULT _deg2rad * (_lon1 - _lon2);
    DECLARE _m     DOUBLE DEFAULT COS(_rlat2);
    DECLARE _x     DOUBLE DEFAULT COS(_rlat1) - _m * COS(_rlond);
    DECLARE _y     DOUBLE DEFAULT               _m * SIN(_rlond);
    DECLARE _z     DOUBLE DEFAULT SIN(_rlat1) - SIN(_rlat2);
    DECLARE _n     DOUBLE DEFAULT SQRT(
                         _x * _x +
                         _y * _y +
                         _z * _z    );
    RETURN  2 * ASIN(_n / 2) / _deg2rad;   -- again--scaled degrees
END;
//
DELIMITER ;

DELIMITER //
-- FindNearest (about my 6th approach)
DROP PROCEDURE IF EXISTS FindNearest6 //
CREATE
PROCEDURE FindNearest (
        IN _my_lat DOUBLE,  -- Latitude of me [-90..90] (not scaled)
        IN _my_lon DOUBLE,  -- Longitude [-180..180]
        IN _START_dist DOUBLE,  -- Starting estimate of how far to search: miles or km
        IN _max_dist DOUBLE,  -- Limit how far to search: miles or km
        IN _limit INT,     -- How many items to try to get
        IN _condition VARCHAR(1111)   -- will be ANDed in a WHERE clause
    )
    DETERMINISTIC
BEGIN
    -- lat and lng are in degrees -90..+90 and -180..+180
    -- All computations done in Latitude degrees.
    -- Thing to tailor
    --   *Locations* -- the table
    --   Scaling of lat, lon; here using *10000 in MEDIUMINT
    --   Table name
    --   miles versus km.

    -- Hardcoded constant:
    DECLARE _deg2rad DOUBLE DEFAULT PI()/1800000;  -- For scaled by 1e4 to MEDIUMINT

    -- Cannot use params in PREPARE, so switch to @variables:
    -- Hardcoded constant:
    SET @my_lat := _my_lat * 10000,
        @my_lon := _my_lon * 10000,
        @deg2dist := 0.0069172,  -- 69.172 for miles; 111.325 for km  *** (mi vs km)
        @start_deg := _start_dist / @deg2dist,  -- Start with this radius first (eg, 15 miles)
        @max_deg := _max_dist / @deg2dist,
```

```
    @cutoff := @max_deg / SQRT(2),  -- (slightly pessimistic)
    @dlat := @start_deg,  -- note: must stay positive
    @lon2lat := COS(_deg2rad * @my_lat),
    @iterations := 0;        -- just debugging

-- Loop through, expanding search
--   Search a 'square', repeat with bigger square until find enough rows
--   If the initial probe found _limit rows, then probably the first
--   iteration here will find the desired data.
-- Hardcoded table name:
-- This is the "first SELECT":
SET @sql = CONCAT(
    "SELECT COUNT(*) INTO @near_ct
        FROM Locations
        WHERE lat    BETWEEN @my_lat - @dlat
                         AND @my_lat + @dlat   -- PARTITION Pruning and bounding box
          AND lon    BETWEEN @my_lon - @dlon
                         AND @my_lon + @dlon   -- first part of PK
          AND ", _condition);
PREPARE _sql FROM @sql;
MainLoop: LOOP
    SET @iterations := @iterations + 1;
    -- The main probe: Search a 'square'
    SET @dlon := ABS(@dlat / @lon2lat);  -- good enough for now  -- note: must stay positive
    -- Hardcoded constants:
    SET @dlon := IF(ABS(@my_lat) + @dlat >= 900000, 3600001, @dlon);  -- near a Pole
    EXECUTE _sql;
    IF ( @near_ct >= _limit OR         -- Found enough
         @dlat >= @cutoff ) THEN       -- Give up (too far)
        LEAVE MainLoop;
    END IF;
    -- Expand 'square':
    SET @dlat := LEAST(2 * @dlat, @cutoff);   -- Double the radius to search
END LOOP MainLoop;
DEALLOCATE PREPARE _sql;

-- Out of loop because found _limit items, or going too far.
-- Expand range by about 1.4 (but not past _max_dist),
-- then fetch details on nearest 10.

-- Hardcoded constant:
SET @dlat := IF( @dlat >= @max_deg OR @dlon >= 1800000,
            @max_deg,
            GCDist(ABS(@my_lat), @my_lon,
                   ABS(@my_lat) - @dlat, @my_lon - @dlon) );
        -- ABS: go toward equator to find farthest corner (also avoids poles)
        -- Dateline: not a problem (see GCDist code)

-- Reach for longitude line at right angle:
-- sin(dlon)*cos(lat) = sin(dlat)
-- Hardcoded constant:
SET @dlon := IFNULL(ASIN(SIN(_deg2rad * @dlat) /
                         COS(_deg2rad * @my_lat))
                        / _deg2rad -- precise
                    , 3600001);    -- must be too near a pole

-- This is the "last SELECT":
-- Hardcoded constants:
IF (ABS(@my_lon) + @dlon < 1800000 OR    -- Usual case - not crossing dateline
    ABS(@my_lat) + @dlat <  900000) THEN -- crossing pole, so dateline not an issue
    -- Hardcoded table name:
    SET @sql = CONCAT(
        "SELECT *,
                @deg2dist * GCDist(@my_lat, @my_lon, lat, lon) AS dist
            FROM Locations
            WHERE lat BETWEEN @my_lat - @dlat
                          AND @my_lat + @dlat   -- PARTITION Pruning and bounding box
              AND lon BETWEEN @my_lon - @dlon
                          AND @my_lon + @dlon   -- first part of PK
```

```
                  AND ", _condition, "
                HAVING dist <= ", _max_dist, "
                ORDER BY dist
                LIMIT ", _limit
                       );
    ELSE
        -- Hardcoded constants and table name:
        -- Circle crosses dateline, do two SELECTs, one for each side
        SET @west_lon := IF(@my_lon < 0, @my_lon, @my_lon - 3600000);
        SET @east_lon := @west_lon + 3600000;
        -- One of those will be beyond +/- 180; this gets points beyond the dateline
        SET @sql = CONCAT(
            "( SELECT *,
                    @deg2dist * GCDist(@my_lat, @west_lon, lat, lon) AS dist
                FROM Locations
                WHERE lat BETWEEN @my_lat - @dlat
                              AND @my_lat + @dlat   -- PARTITION Pruning and bounding box
                  AND lon BETWEEN @west_lon - @dlon
                              AND @west_lon + @dlon   -- first part of PK
                  AND ", _condition, "
                HAVING dist <= ", _max_dist, " )
            UNION ALL
            ( SELECT *,
                    @deg2dist * GCDist(@my_lat, @east_lon, lat, lon) AS dist
                FROM Locations
                WHERE lat BETWEEN @my_lat - @dlat
                              AND @my_lat + @dlat   -- PARTITION Pruning and bounding box
                  AND lon BETWEEN @east_lon - @dlon
                              AND @east_lon + @dlon   -- first part of PK
                  AND ", _condition, "
                HAVING dist <= ", _max_dist, " )
            ORDER BY dist
            LIMIT ", _limit
                       );
    END IF;

    PREPARE _sql FROM @sql;
    EXECUTE _sql;
    DEALLOCATE PREPARE _sql;
END;
//
DELIMITER ;
<<code>>

== Sample

Find the 5 cities with non-zero population (out of 3 million) nearest to (+35.15, -90.15). Start with a
10-mile bounding box and give up at 100 miles.

<<code>>
CALL FindNearestLL(35.15, -90.05, 10, 100, 5, 'population > 0');
+---------+--------+---------+---------+--------------+--------------+-------+------------+--------------
+---------------------+------------------------+
| id      | lat    | lon     | country | ascii_city   | city         | state | population | @gcd_ct := 0
| dist                | @gcd_ct := @gcd_ct + 1 |
+---------+--------+---------+---------+--------------+--------------+-------+------------+--------------
+---------------------+------------------------+
| 3023545 | 351494 | -900489 | us      | memphis      | Memphis      | TN    |     641608 |            0
| 0.07478733189367963 |                      3 |
| 2917711 | 351464 | -901844 | us      | west memphis | West Memphis | AR    |      28065 |            0
|   7.605683607627499 |                      2 |
| 2916457 | 352144 | -901964 | us      | marion       | Marion       | AR    |       9227 |            0
|     9.3994963998986 |                      1 |
| 3020923 | 352044 | -898739 | us      | bartlett     | Bartlett     | TN    |      43264 |            0
|  10.643941157860604 |                      7 |
| 2974644 | 349889 | -900125 | us      | southaven    | Southaven    | MS    |      38578 |            0
|  11.344042217329935 |                      5 |
+---------+--------+---------+---------+--------------+--------------+-------+------------+--------------
+---------------------+------------------------+
```

```
5 rows in set (0.00 sec)
Query OK, 0 rows affected (0.04 sec)

SELECT COUNT(*) FROM ll_table;
+----------+
| COUNT(*) |
+----------+
|  3173958 |
+----------+
1 row in set (5.04 sec)

FLUSH STATUS;
CALL...
SHOW SESSION STATUS LIKE 'Handler%';

SHOW session status LIKE 'Handler%';
+----------------------------+-------+
| Variable_name              | Value |
+----------------------------+-------+
| Handler_read_first         | 1     |
| Handler_read_key           | 3     |
| Handler_read_next          | 1307  |  -- some index, some tmp, but far less than 3 million.
| Handler_read_rnd           | 5     |
| Handler_read_rnd_next      | 13    |
| Handler_write              | 12    |  -- it needed a tmp
+----------------------------+-------+
```

Postlog

There is a “Haversine” algorithm that is twice as fast as the GCDist function here. But it has a fatal flaw of sometimes returning NULL for the distance between a point and itself. (This is because of computing a number slightly bigger than 1.0, then trying to take the ACOS of it.)

See also

- Cities used for testing
- A forum thread
- StackOverflow discussion
- Sample
- Z-ordering

Rick James graciously allowed us to use this article in the documentation.

Rick James’ site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: latlng

This page is licensed: CC BY-SA / Gnu FDL

Primary Keys with Nullable Columns

MariaDB deals with primary keys over nullable columns according to the SQL standards.

Take the following table structure:

```
CREATE TABLE t1(
  c1 INT NOT NULL AUTO_INCREMENT,
  c2 INT NULL DEFAULT NULL,
  PRIMARY KEY(c1,c2)
);
```

Column c2 is part of a primary key, and thus it cannot be NULL.

Before MariaDB 10.1.7, MariaDB (as well as versions of MySQL before MySQL 5.7) would silently convert it into a NOT NULL column with a default value of *0*.

Since MariaDB 10.1.7, the column is converted to NOT NULL, but without a default value. If we then attempt to insert a record without explicitly setting *c2*, a warning (or, in strict mode, an error), will be thrown, for example:

```
INSERT INTO t1() VALUES();
Query OK, 1 row affected, 1 warning (0.00 sec)
Warning (Code 1364): Field 'c2' doesn't have a default value

SELECT * FROM t1;
+----+----+
| c1 | c2 |
+----+----+
|  1 |  0 |
+----+----+
```

MySQL, since 5.7, will abort such a CREATE TABLE with an error.

The MariaDB 10.1.7 behavior adheres to the SQL 2003 standard.

SQL-2003, Part II, “Foundation” says:

11.7 ****Syntax Rules

…

5) If the specifies PRIMARY KEY, then for each in the explicit or implicit for which NOT NULL is not specified, NOT NULL is implicit in the .

Essentially this means that all PRIMARY KEY columns are automatically converted to NOT NULL. Furthermore:

11.5 General Rules

…

3) When a site S is set to its default value,

…

b) If the data descriptor for the site includes a , then S is set to the value specified by that .

…

e) Otherwise, S is set to the null value.

There is no concept of “no default value” in the standard. Instead, a column always has an implicit default value of NULL. On insertion it might however fail the NOT NULL constraint. MariaDB and MySQL instead mark such a column as “not having a default value”. The end result is the same — a value must be specified explicitly or an INSERT will fail.

MariaDB since 10.1.7 behaves in a standard compatible manner — being part of a PRIMARY KEY, the nullable column gets an automatic NOT NULL constraint, on insertion one must specify a value for such a column. MariaDB before 10.1.7 was automatically assigning a default value of 0 — this behavior was non-standard. Issuing an error at CREATE TABLE time is also non-standard.

See Also

- MDEV-12248 describes an edge-case that may result in replication problems when replicating from a master server before this change to a slave server after this change.

This page is licensed: CC BY-SA / Gnu FDL

Storage Engine Index Types

This refers to the index_type definition when creating an index, i.e. BTREE, HASH or RTREE.

For more information on general types of indexes, such as primary keys, unique indexes etc, go to Getting Started with Indexes.

Storage Engine	Permitted Indexes
Aria	BTREE, RTREE
MyISAM	BTREE, RTREE
InnoDB	BTREE
MEMORY/HEAP	HASH, BTREE

BTREE is generally the default index type. For MEMORY tables, HASH is the default.

Understanding the B-tree and hash data structures can help predict how different queries perform on different storage engines that use these data structures in their indexes, particularly for the MEMORY storage engine that lets you choose B-tree or hash indexes.
B-Tree Index Characteristics

B-tree Indexes

B-tree indexes are used for column comparisons using the >, >=, =, >=, < or BETWEEN operators, as well as for LIKE comparisons that begin with a constant.

For example, the query `SELECT * FROM Employees WHERE First_Name LIKE 'Maria%';` can make use of a B-tree index, while `SELECT * FROM Employees WHERE First_Name LIKE '%aria';` cannot.

B-tree indexes also permit leftmost prefixing for searching of rows.

If the number or rows doesn't change, hash indexes occupy a fixed amount of memory, which is lower than the memory occupied by BTREE indexes.

Hash Indexes

Hash indexes, in contrast, can only be used for equality comparisons, so those using the = or <=> operators. They cannot be used for ordering, and provide no information to the optimizer on how many rows exist between two values.

Hash indexes do not permit leftmost prefixing - only the whole index can be used.

R-tree Indexes

See SPATIAL for more information.

This page is licensed: CC BY-SA / Gnu FDL

Full-Text Indexes

See Full-Text Index Overview.

Definitive full-text index guide: MATCH() AGAINST syntax, FULLTEXT index creation, NATURAL/BOOLEAN/QUERY EXPANSION modes, stopwords, ft_min_word_length.

See Full-Text Index Stopwords.

Explains stopwords for full-text indexes, the differing default lists in MyISAM and InnoDB, and how to override them, plus the full default MyISAM stopword list.

This page is licensed: CC BY-SA / Gnu FDL

Full-Text Index Overview

MariaDB has support for full-text indexing and searching:

- A full-text index in MariaDB is an index of type FULLTEXT, and it allows more options when searching for portions of text from a field.
- Full-text indexes can be used only with MyISAM, Aria, InnoDB and Mroonga tables, and can be created only for CHAR, VARCHAR, or TEXT columns.
- Partitioned tables cannot contain fulltext indexes, even if the storage engine supports them.
- A FULLTEXT index definition can be given in the CREATE TABLE statement when a table is created, or added later using ALTER TABLE or CREATE INDEX.
- For large data sets, it is much faster to load your data into a table that has no FULLTEXT index and then create the index after that, than to load data into a table that has an existing FULLTEXT index.

Full-text searching is performed using MATCH() … AGAINST syntax. MATCH() takes a comma-separated list that names the columns to be searched. AGAINST takes a string to search for, and an optional modifier that indicates what type of search to perform. The search string must be a literal string, not a variable or a column name.

```
MATCH (col1,col2,...) AGAINST (expr [search_modifier])
```

Excluded Results

- Partial words are excluded.
- Words less than 4 (MyISAM) or 3 (InnoDB) characters in length will not be stored in the fulltext index. This value can be adjusted by changing the ft_min_word_length system variable (or, for InnoDB, innodb_ft_min_token_size).
- Words longer than 84 characters in length will also not be stored in the fulltext index. This values can be adjusted by changing the ft_max_word_length system variable (or, for InnoDB, innodb_ft_max_token_size).
- Stopwords are a list of common words such as “once” or “then” that do not reflect in the search results unless IN BOOLEAN MODE is used. The stopword list for MyISAM/Aria tables and InnoDB tables can differ. See stopwords for details and a full list, as well as for details on how to change the default list.
- For MyISAM/Aria fulltext indexes only, if a word appears in more than half the rows, it is also excluded from the results of a fulltext search.
- For InnoDB indexes, only committed rows appear - modifications from the current transaction do not apply.

Relevance

MariaDB calculates a relevance for each result, based on a number of factors, including the number of words in the index, the number of unique words in a row, the total number of words in both the index and the result, and the weight of the word. In English, ‘cool’ will be weighted less than ‘dandy’, at least at present! The relevance can be returned as part of a query simply by using the MATCH function in the field list.

Types of Full-Text search

IN NATURAL LANGUAGE MODE

IN NATURAL LANGUAGE MODE is the default type of full-text search, and the keywords can be omitted. There are no special operators, and searches consist of one or more comma-separated keywords.

Searches are returned in descending order of relevance.

IN BOOLEAN MODE

Boolean search permits the use of a number of special operators:

Operator	Description
+	The word is mandatory in all rows returned.
-	The word cannot appear in any row returned.
<	The word that follows has a lower relevance than other words, although rows containing it will still match
>	The word that follows has a higher relevance than other words.
()	Used to group words into subexpressions.
~	The word following contributes negatively to the relevance of the row (which is different to the '-' operator, which specifically excludes the word, or the '<' operator, which still causes the word to contribute positively to the relevance of the row.
*	The wildcard, indicating zero or more characters. It can only appear at the end of a word.
"	Anything enclosed in the double quotes is taken as a whole (so you can match phrases, for example).

Searches are not returned in order of relevance, and nor does the 50% limit apply. Stopwords and word minimum and maximum lengths still apply as usual.

WITH QUERY EXPANSION

A query expansion search is a modification of a natural language search. The search string is used to perform a regular natural language search. Then, words from the most relevant rows returned by the search are added to the search string and the search is done again. The query returns the rows from the second search. The IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION or WITH QUERY EXPANSION modifier specifies a query expansion search. It can be useful when relying on implied knowledge within the data, for example that MariaDB is a database.

Examples

Creating a table, and performing a basic search:

```
CREATE TABLE ft_myisam(copy TEXT,FULLTEXT(copy)) ENGINE=MyISAM;

INSERT INTO ft_myisam(copy) VALUES ('Once upon a time'),
  ('There was a wicked witch'), ('Who ate everybody up');

SELECT * FROM ft_myisam WHERE MATCH(copy) AGAINST('wicked');
+--------------------------+
| copy                     |
+--------------------------+
| There was a wicked witch |
+--------------------------+
```

Multiple words:

```
SELECT * FROM ft_myisam WHERE MATCH(copy) AGAINST('wicked,witch');
+--------------------------------+
| copy                           |
+--------------------------------+
| There was a wicked witch       |
+--------------------------------+
```

Since 'Once' is a stopword, no result is returned:

```
SELECT * FROM ft_myisam WHERE MATCH(copy) AGAINST('Once');
Empty set (0.00 sec)
```

Inserting the word 'wicked' into more than half the rows excludes it from the results:

```
INSERT INTO ft_myisam(copy) VALUES ('Once upon a wicked time'),
  ('There was a wicked wicked witch'), ('Who ate everybody wicked up');

SELECT * FROM ft_myisam WHERE MATCH(copy) AGAINST('wicked');
Empty set (0.00 sec)
```

Using IN BOOLEAN MODE to overcome the 50% limitation:

```
SELECT * FROM ft_myisam WHERE MATCH(copy) AGAINST('wicked' IN BOOLEAN MODE);
+-------------------------------+
| copy                          |
+-------------------------------+
| There was a wicked witch      |
| Once upon a wicked time       |
| There was a wicked wicked witch |
| Who ate everybody wicked up   |
+-------------------------------+
```

Returning the relevance:

```
SELECT copy,MATCH(copy) AGAINST('witch') AS relevance
  FROM ft_myisam WHERE MATCH(copy) AGAINST('witch');
+-------------------------------+--------------------+
| copy                          | relevance          |
+-------------------------------+--------------------+
| There was a wicked witch      | 0.6775632500648499 |
| There was a wicked wicked witch | 0.5031757950782776 |
+-------------------------------+--------------------+
```

WITH QUERY EXPANSION. In the following example, ‘MariaDB’ is always associated with the word ‘database’, so it is returned when query expansion is used, even though not explicitly requested.

```
CREATE TABLE ft2(copy TEXT,FULLTEXT(copy)) ENGINE=MyISAM;

INSERT INTO ft2(copy) VALUES
 ('MySQL vs MariaDB database'),
 ('Oracle vs MariaDB database'),
 ('PostgreSQL vs MariaDB database'),
 ('MariaDB overview'),
 ('Foreign keys'),
 ('Primary keys'),
 ('Indexes'),
 ('Transactions'),
 ('Triggers');

SELECT * FROM ft2 WHERE MATCH(copy) AGAINST('database');
+------------------------------+
| copy                         |
+------------------------------+
| MySQL vs MariaDB database    |
| Oracle vs MariaDB database   |
| PostgreSQL vs MariaDB database |
+------------------------------+
3 rows in set (0.00 sec)

SELECT * FROM ft2 WHERE MATCH(copy) AGAINST('database' WITH QUERY EXPANSION);
+------------------------------+
| copy                         |
+------------------------------+
| MySQL vs MariaDB database    |
| Oracle vs MariaDB database   |
| PostgreSQL vs MariaDB database |
| MariaDB overview             |
+------------------------------+
4 rows in set (0.00 sec)
```

Partial word matching with IN BOOLEAN MODE:

```
SELECT * FROM ft2 WHERE MATCH(copy) AGAINST('Maria*' IN BOOLEAN MODE);
+-------------------------------+
| copy                          |
+-------------------------------+
| MySQL vs MariaDB database     |
| Oracle vs MariaDB database    |
| PostgreSQL vs MariaDB database |
| MariaDB overview              |
+-------------------------------+
```

Using boolean operators

```
SELECT * FROM ft2 WHERE MATCH(copy) AGAINST('+MariaDB -database'
  IN BOOLEAN MODE);
+------------------+
| copy             |
+------------------+
| MariaDB overview |
+------------------+
```

See Also

- For simpler searches of a substring in text columns, see the LIKE operator.

This page is licensed: CC BY-SA / Gnu FDL

Full-Text Index Stopwords

Stopwords are used to provide a list of commonly-used words that can be ignored for the purposes of Full-text-indexes.

Full-text indexes built in MyISAM and InnoDB have different stopword lists by default.

MyISAM Stopwords

For full-text indexes on MyISAM tables, by default, the list is built from the file `storage/myisam/ft_static.c`, and searched using the server's character set and collation. The ft_stopword_file system variable allows the default list to be overridden with words from another file, or for stopwords to be ignored altogether.

If the stopword list is changed, any existing full-text indexes need to be rebuilt

The following table shows the default list of stopwords, although you should always treat `storage/myisam/ft_static.c` as the definitive list. See the Fulltext Index Overview for more details, and Full-text-indexes for related articles.

a's	able	about	above
according	accordingly	across	actually
after	afterwards	again	against
ain't	all	allow	allows
almost	alone	along	already
also	although	always	am
among	amongst	an	and
another	any	anybody	anyhow
anyone	anything	anyway	anyways
anywhere	apart	appear	appreciate

appropriate	are	aren’t	around
as	aside	ask	asking
associated	at	available	away
awfully	be	became	because
become	becomes	becoming	been
before	beforehand	behind	being
believe	below	beside	besides
best	better	between	beyond
both	brief	but	by
c’mon	c’s	came	can
can’t	cannot	cant	cause
causes	certain	certainly	changes
clearly	co	com	come
comes	concerning	consequently	consider
considering	contain	containing	contains
corresponding	could	couldn’t	course
currently	definitely	described	despite
did	didn’t	different	do
does	doesn’t	doing	don’t
done	down	downwards	during
each	edu	eg	eight
either	else	elsewhere	enough
entirely	especially	et	etc
even	ever	every	everybody
everyone	everything	everywhere	ex
exactly	example	except	far
few	fifth	first	five
followed	following	follows	for
former	formerly	forth	four
from	further	furthermore	get
gets	getting	given	gives
go	goes	going	gone
got	gotten	greetings	had
hadn’t	happens	hardly	has
hasn’t	have	haven’t	having
he	he’s	hello	help
hence	her	here	here’s
hereafter	hereby	herein	hereupon
hers	herself	hi	him
himself	his	hither	hopefully

how	howbeit	however	i’d
i’ll	i’m	i’ve	ie
if	ignored	immediate	in
inasmuch	inc	indeed	indicate
indicated	indicates	inner	insofar
instead	into	inward	is
isn’t	it	it’d	it’ll
it’s	its	itself	just
keep	keeps	kept	know
knows	known	last	lately
later	latter	latterly	least
less	lest	let	let’s
like	liked	likely	little
look	looking	looks	ltd
mainly	many	may	maybe
me	mean	meanwhile	merely
might	more	moreover	most
mostly	much	must	my
myself	name	namely	nd
near	nearly	necessary	need
needs	neither	never	nevertheless
new	next	nine	no
nobody	non	none	noone
nor	normally	not	nothing
novel	now	nowhere	obviously
of	off	often	oh
ok	okay	old	on
once	one	ones	only
onto	or	other	others
otherwise	ought	our	ours
ourselves	out	outside	over
overall	own	particular	particularly
per	perhaps	placed	please
plus	possible	presumably	probably
provides	que	quite	qv
rather	rd	re	really
reasonably	regarding	regardless	regards
relatively	respectively	right	said
same	saw	say	saying
says	second	secondly	see

seeing	seem	seemed	seeming
seems	seen	self	selves
sensible	sent	serious	seriously
seven	several	shall	she
should	shouldn’t	since	six
so	some	somebody	somehow
someone	something	sometime	sometimes
somewhat	somewhere	soon	sorry
specified	specify	specifying	still
sub	such	sup	sure
t’s	take	taken	tell
tends	th	than	thank
thanks	thanx	that	that’s
thats	the	their	theirs
them	themselves	then	thence
there	there’s	thereafter	thereby
therefore	therein	theres	thereupon
these	they	they’d	they’ll
they’re	they’ve	think	third
this	thorough	thoroughly	those
though	three	through	throughout
thru	thus	to	together
too	took	toward	towards
tried	tries	truly	try
trying	twice	two	un
under	unfortunately	unless	unlikely
until	unto	up	upon
us	use	used	useful
uses	using	usually	value
various	very	via	viz
vs	want	wants	was
wasn’t	way	we	we’d
we’ll	we’re	we’ve	welcome
well	went	were	weren’t
what	what’s	whatever	when
whence	whenever	where	where’s
whereafter	whereas	whereby	wherein
whereupon	wherever	whether	which
while	whither	who	who’s
whoever	whole	whom	whose

why	will	willing	wish
with	within	without	won't
wonder	would	wouldn't	yes
yet	you	you'd	you'll
you're	you've	your	yours
yourself	yourselves	zero	

InnoDB Stopwords

Stopwords on full-text indexes are only enabled if the innodb_ft_enable_stopword system variable is set (by default it is) at the time the index was created.

The stopword list is determined as follows:

- If the innodb_ft_user_stopword_table system variable is set, that table is used as a stopword list.
- If `innodb_ft_user_stopword_table` is not set, the table set by innodb_ft_server_stopword_table is used.
- If neither variable is set, the built-in list is used, which can be viewed by querying the INNODB_FT_DEFAULT_STOPWORD table in the Information Schema.

In the first two cases, the specified table must exist at the time the system variable is set and the full-text index created. It must be an InnoDB table with a single column, a VARCHAR named VALUE.

The default InnoDB stopword list differs from the default MyISAM list, being much shorter, and contains the following words:

a	about	an	are
as	at	be	by
com	de	en	for
from	how	i	in
is	it	la	of
on	or	that	the
this	to	was	what
when	where	who	will
with	und	the	www

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Hints

Optimizer hints are options available that affect the execution plan.

SELECT Modifiers have been in MariaDB for a long time, while Expanded (New-Style) Optimizer Hints were introduced in MariaDB 12.0 and 12.1.

See SELECT Modifier Hints.

SELECT modifier hints such as HIGH_PRIORITY, SQL_CACHE, SQL_NO_CACHE, and SQL_BUFFER_RESULT that adjust how individual SELECT statements are executed.

See Expanded New-Style Optimizer Hints.

New-style optimizer hints (MariaDB 12.0+) that give granular, per-query control over optimizer choices without changing server-wide variables.

See Index-Level Hints.

Index-level optimizer hints that tell the optimizer which indexes to use, ignore, or prefer for a given table.

See Query Block Naming.

Assign names to query blocks with QB_NAME() so optimizer hints can target specific subqueries or statement blocks.

See Table-Level Hints.

Table-level optimizer hints that influence join order and per-table optimizer behavior, optionally scoped to a named query block.

See Also

- Use optimizer_switch to enable/disable specific optimizations

This page is licensed: CC BY-SA / Gnu FDL

SELECT Modifier Hints

HIGH PRIORITY

`HIGH_PRIORITY` gives the statement a higher priority. If the table is locked, high priority `SELECT`s will be executed as soon as the lock is released, even if other statements are queued. `HIGH_PRIORITY` applies only if the storage engine only supports table-level locking (`MyISAM`, `MEMORY`, `MERGE`). See HIGH_PRIORITY and LOW_PRIORITY clauses for details.

SQL_CACHE / SQL_NO_CACHE

If the query_cache_type system variable is set to 2 or `DEMAND`, and the current statement is cacheable, `SQL_CACHE` causes the query to be cached and `SQL_NO_CACHE` causes the query not to be cached. For `UNION`s, `SQL_CACHE` or `SQL_NO_CACHE` should be specified for the first query. See also The Query Cache for more detail and a list of the types of statements that aren't cacheable.

SQL_BUFFER_RESULT

`SQL_BUFFER_RESULT` forces the optimizer to use a temporary table to process the result. This is useful to free locks as soon as possible.

SQL_SMALL_RESULT / SQL_BIG_RESULT

`SQL_SMALL_RESULT` and `SQL_BIG_RESULT` tell the optimizer whether the result is very big or not. Usually, `GROUP BY` and `DISTINCT` operations are performed using a temporary table. Only if the result is very big, using a temporary table is not convenient. The optimizer automatically knows if the result is too big, but you can force the optimizer to use a temporary table with `SQL_SMALL_RESULT`, or avoid the temporary table using `SQL_BIG_RESULT`.

STRAIGHT_JOIN

`STRAIGHT_JOIN` applies to the JOIN queries, and tells the optimizer that the tables must be read in the order they appear in the `SELECT`. For `const` and `system` table this options is sometimes ignored.

SQL_CALC_FOUND_ROWS

`SQL_CALC_FOUND_ROWS` is only applied when using the `LIMIT` clause. If this option is used, MariaDB will count how many rows would match the query, without the `LIMIT` clause. That number can be retrieved in the next query, using FOUND_ROWS().

USE/FORCE/IGNORE INDEX

`USE INDEX`, `FORCE INDEX` and `IGNORE INDEX` constrain the query planning to a specific index. For further information about some of these options, see How to force query plans.

This page is licensed: CC BY-SA / Gnu FDL

Expanded New-Style Optimizer Hints

> **Note:**
>
> New-style optimizer hints were introduced in MariaDB 12.0 and 12.1.

Description

In order to control optimizer choices of query plans, one can use optimizer_switch, join cache level and other system variables. However, these variables affect execution of all queries but not some specific ones. To get more granular control, you can:

- Specify server variables before execution of every query (or group of queries);
- Use SET STATEMENT ... FOR to temporarily change server variables for a specific query;
- Use optimizer hints.

The benefit of optimizer hints is that they can control specific parts of the query: how to handle a particular `SELECT` or subquery, which indexes to use to access certain tables, etc.

Technically, hints are specifically formatted comments embedded right into the query text. For example:

```
SELECT /*+ JOIN_ORDER(t2, t1) NO_INDEX(t2)*/ t1.* FROM t1, t2 ... ;
SELECT /*+ NO_SEMIJOIN() */ * FROM t1 WHERE t1.a IN (...);
SELECT /*+ MERGE(dt1) */ * FROM (SELECT * FROM t1) AS dt1;
DELETE /*+ NO_BNL(t2@qb1) */ * FROM t1 WHERE a IN (SELECT /*+ QB_NAME(qb1) */ .. );
UPDATE /*+ NO_RANGE_OPTIMIZATION(t1 PRIMARY) */  * FROM t1 ...;
```

Syntax

Hints sequence starts with `/*+` and ends with `*/`. There can be an arbitrary number of hints in the sequence, separated by spaces. Hints are recognized by the parser if they follow the initial keyword `SELECT`, `UPDATE`, `DELETE`. Each query block can have its own set of hints, for example:

```
SELECT /*+ ... */ ... FROM t1 WHERE a IN (SELECT /*+ ... */ ...);
UPDATE /*+ ... */ ... WHERE a IN (SELECT /*+ ... */ ... );
SELECT /*+ ... */ ... UNION ALL SELECT /*+ ... */ ...;
```

`INSERT ... SELECT` statements support hints only at the `SELECT` part, not at the `INSERT` part:

```
INSERT INTO ... SELECT /*+ ... */ ...;
```

A single query block may have multiple hints in a single hint sequence, for example:

```
SELECT /*+ NO_BKA(qb1) INDEX(t1 idx1) JOIN_PREFIX(t2, t1)*/ ... FROM t1, t2 ...;
```

but multiple sequences are not supported:

```
SELECT /*+ NO_BKA(qb1) */ /*+ INDEX(t1 idx1) */ ... FROM t1 ...;
```

Incorrect syntax, duplication of hints or other inconsistencies produce warnings:

```
SELECT /*+ QB_NAME(qb1) QB_NAME(qb1 ) */ * FROM t2 ...
...
Warnings:
Warning 4219    Hint QB_NAME(`qb1`) is ignored as conflicting/duplicated

SELECT /*+ BKA(t1) NO_BKA(t1) */ * FROM t1 ...
...
Warnings:
Warning 4219    Hint NO_BKA(`t1`) is ignored as conflicting/duplicated

SELECT /*+ INDEX(t1) JOIN_INDEX(t1) */ a FROM t1 ...
...
Warnings:
Warning 4240    Hint JOIN_INDEX(`t1` ) is ignored as conflicting/duplicated (an index hint of the same
type or opposite kind has already been specified for this table)

SELECT /*+ SEMIJOIN(loosescan @qb1) */ a FROM t1 ...
...
Warnings:
Warning 1064    Optimizer hint syntax error near '@qb1) */ a FROM t1' at line 1

SELECT /*+ SUBQUERY(@qb1 materialization) */ a FROM t1 ...
...
Warnings:
Warning 4220    Query block name `qb1` is not found for SUBQUERY hint
```

Hints that were not rejected as invalid/conflicting/duplicated are visible in the output of `EXPLAIN EXTENDED`, in section “Warnings”:

```
EXPLAIN EXTENDED SELECT /*+ INDEX(t1)*/ a FROM t1;
id      select_type     table   type    possible_keys   key     key_len ref     rows    filtered
Extra
1       SIMPLE  t1      index   NULL    i_a     5       NULL    256     100.00  Using index
Warnings:
Note    1003    select /*+ INDEX(`t1`@`select#1`) */ `test`.`t1`.`a` AS `a` from `test`.`t1`
```

General Notes

- If a table has an alias, hints must refer to the alias, not the table name.
- Table names in hints cannot be qualified with schema names.

Hints Inside Views

Hints may be specified within VIEW’s during their creation. For example:

```
CREATE VIEW v1 AS
  SELECT /*+ GROUP_INDEX(t1 idx1) */ FROM t1 ... GROUP BY ... HAVING ...
```

The hints are then visible in the output of `SHOW CREATE VIEW`.

Limitations

Hints inside views are applied locally within that VIEW’s scope and do not affect the outer query. Query blocks declared inside views cannot be referenced from hints in the outer query.

```
-- Query block name `qb_v1` is not accessible from outer queries using `v1`!
CREATE VIEW v1 AS
  SELECT /*+ QB_NAME(qb_v1) */ FROM t1 ... GROUP BY ... HAVING ...
```

To reference inner view objects from outside the view, refer to other query block addressing methods like query block names with path or implicit names based on aliases.

Hint Hierarchy

Hints can apply at different levels:

- global - the hints affect *the whole query*;
- query-block-level - the hints affect *a certain query block* of the query;
- table-level - the hints affect *certain tables*;
- index-level - the hints affect *particular indexes of tables*.

Available Optimizer Hints

This table provides an overview of optimizer hints supported in MariaDB, showing which optimizations each hint controls and at what level (global, query block, table, or index) they can be applied.

Hint Name	Affected Optimization	Applicable Scopes
`BKA`, `NO_BKA`	Batched key access join	Query block, Table
`BNL`, `NO_BNL`	Block nested-loop join	Query block, Table
`DERIVED_CONDITION_PUSHDOWN`, `NO_DERIVED_CONDITION_PUSHDOWN`	Condition pushdown for derived tables	Query block, Table
`GROUP_INDEX`, `NO_GROUP_INDEX`	Use of indexes for `GROUP BY` operations	Table, Index
`INDEX`, `NO_INDEX`	Use of indexes	Table, Index
`INDEX_MERGE`, `NO_INDEX_MERGE`	Index merge	Table, Index
`JOIN_FIXED_ORDER`	Straight join order	Query block
`JOIN_INDEX`, `NO_JOIN_INDEX`	Use of indexes for data access	Index
`JOIN_ORDER`	Join order	Table
`JOIN_PREFIX`	Join order for first tables	Table
`JOIN_SUFFIX`	Join order for last tables	Table
`MAX_EXECUTION_TIME`	Query execution time limit	Global
`MERGE`, `NO_MERGE`	Derived table/CTE merging	Query block, Table
`MRR`, `NO_MRR`	Multi-Range Read	Table, Index
`NO_ICP`	Index Condition Pushdown	Table, Index
`NO_RANGE_OPTIMIZATION`	Range optimization	Table, Index
`ORDER_INDEX`, `NO_ORDER_INDEX`	Use of indexes for sorting	Table, Index
`QB_NAME`	Assigns name to query block	Query block
`ROWID_FILTER`, `NO_ROWID_FILTER`	Rowid filtering	Table, Index
`SEMIJOIN`, `NO_SEMIJOIN`	Semi-join optimization	Query block
`SPLIT_MATERIALIZED`, `NO_SPLIT_MATERIALIZED`	Lateral derived / split materialization	Table
`SUBQUERY`	Subquery transformation	Query block

Syntax of Global Hints

Such hints affect *the whole query*.

Syntax

```
hint_name(arguments)
```

Currently, there is only one global hint:

- `MAX_EXECUTION_TIME`

Syntax of Query-Block-Level Hints

These hints affect *a certain query block* of the query.

Syntax

```
hint_name([@query_block_name])
```

Examples

```
SELECT /*+ NO_BNL() */ * FROM t1 JOIN t2 ON t1.a = t2.a;
```

This statement contains one query block (since its name is not specified explicitly, this query block is assigned an implicit name `select#1`). The `NO_BNL()` hint does not specify a query block name, so it is applied to the query block where the hint is specified. The hint disables block nested loop join for any tables of this query block.

```
SELECT /*+ BKA(@qb1)*/ a FROM
  (SELECT /*+ QB_NAME(qb1)*/ * FROM t1 JOIN t2 ON t1.a = t2.a) AS dt;
```

This statement contains two query blocks:

- the topmost one which does not have an explicit name;
- the one corresponding to derived table `dt` which is assigned an explicit name `qb1` with the use of `QB_NAME` hint.

Here the `BKA(@qb1)` hint addresses query block name `qb1`. The hint enables batched key access join for tables of this query block.

This syntax is equivalent to:

```
SELECT a FROM
  (SELECT /*+ BKA(@qb1) QB_NAME(qb1)*/ * FROM t1 JOIN t2 ON t1.a = t2.a) AS dt;

SELECT a FROM
  (SELECT /*+ BKA()*/ * FROM t1 JOIN t2 ON t1.a = t2.a) AS dt;
```

See query block naming for more information.

Syntax of Table-Level Hints

These hints affect *certain tables* of the statement.

There are two syntax variants of table-level hints: to affect tables from the same query block, and to affect tables from different query blocks.

Tables From the Same Query Block

This variant addresses some or all tables of a particular query block:

```
hint_name([@query_block_name] [tbl_name [, tbl_name] ...])
```

Both `@query_block_name` and the list of `tbl_name`'s are optional. If `@query_block_name` is **not** specified, the hint applies to tables of the query block the hint is added to. If no tables are specified in the list, the hint affects the whole query block and effectively becomes a query-block-level hint.

Examples

```
SELECT /*+ NO_MERGE(dt) */ dt.a, dt.b
  FROM (SELECT a, b FROM t1) AS dt
  JOIN (SELECT a, b FROM t2) AS dt2 ON dt.a = dt2.a;
```

The hint disables merging of derived table `dt` into the upper `SELECT`. More information about this hint can be found here.

If a user wants to disable merging of both `dt` and `dt2`, they can mention both derived table names in the hint body:

```
SELECT /*+ NO_MERGE(dt, dt2) */ dt.a, dt.b
  FROM (SELECT a, b FROM t1) AS dt
  JOIN (SELECT a, b FROM t2) AS dt2 ON dt.a = dt2.a;
```

Alternatively, since there are no more derived tables in the statement besides `dt` and `dt2`, the same effect can be achieved with the query-block level variant of the hint:

```
SELECT /*+ NO_MERGE() */ dt.a, dt.b
  FROM (SELECT a, b FROM t1) AS dt
  JOIN (SELECT a, b FROM t2) AS dt2 ON dt.a = dt2.a;
```

Here the effect of the hint is applied to the scope of the main query block.

Now let's consider a more complicated example when a user has a statement with two derived tables one of which is nested into another:

```
SELECT dt.a, dt.b
  FROM (
    SELECT t1.a, t1.b FROM t1
    JOIN (SELECT a, b FROM t2) AS dt_inner ON t1.a = dt_inner.a
  ) dt_outer;
```

If a user wants to disable merging of the inner derived table `dt_inner`, there are three ways of doing that:

- assign an explicit name to the query block that the inner derived table belongs to, and address `dt_inner` with a hint from the topmost query block:

```
SELECT /*+ NO_MERGE(@inner_qb_name dt_inner) */ dt.a, dt.b
  FROM (
    SELECT /*+ QB_NAME(inner_qb_name) */ t1.a, t1.b FROM t1
    JOIN (SELECT a, b FROM t2) AS dt_inner ON t1.a = dt_inner.a
  ) dt_outer;
```

If there were more derived tables in `inner_qb_name` query block to address, they all should have been mentioned in the hint body, for example: `NO_MERGE(@inner_qb_name dt_inner, dt_inner2, dt_inner3)`

- Place the hint right into the inner query block:

```
SELECT dt.a, dt.b
  FROM (
    SELECT /*+ NO_MERGE(dt2)*/ t1.a, t1.b FROM t1
    JOIN (SELECT a, b FROM t2) AS dt2 ON t1.a = dt2.a
  ) dt;
```

- Use an alternative variant of the syntax that is described in the paragraph below:

```
SELECT /*+ NO_MERGE(dt_inner@inner_qb_name) */ dt.a, dt.b
  FROM (
    SELECT /*+ QB_NAME(inner_qb_name) */ t1.a, t1.b FROM t1
    JOIN (SELECT a, b FROM t2) AS dt_inner ON t1.a = dt_inner.a
  ) dt_outer;
```

Tables From Different Query Blocks

This variant of the syntax addresses tables from different query blocks:

```
hint_name([tbl_name@query_block_name [, tbl_name@query_block_name] ...])
```

For example, a user wants to disable batched key access join for table `t2` from derived query `dt` and for table `t3` from the topmost query block. They can mention both table names in the hint body as follows:

```
SELECT /*+ NO_BKA(t2@inner_qb, t3)*/ SUM(t3.a) FROM
    (SELECT /*+ QB_NAME(inner_qb)*/ t1.b FROM t1 JOIN t2 ON t1.a = t2.a) AS dt
    JOIN t3 ON t3.a = dt.b;
```

If they run `EXPLAIN EXTENDED` for this query, they will see the hint applied to both tables from the hint body:

```
Warnings:
Note    1003    select /*+ NO_BKA(`t2`@`inner_qb`) NO_BKA(`t3`@`select#1`) */ sum(`test`.`t3`.`a`) AS
`SUM(t3.a)` from `test`.`t1` join `test`.`t2` join `test`.`t3` where `test`.`t3`.`a` = `test`.`t1`.`b`
and `test`.`t1`.`a` = `test`.`t2`.`a`
```

`select#1` here is the implicit system name of the topmost query block (see more about this at query block naming).

In fact, the user can add the hints to the query in the same way as it is displayed above:

```
SELECT /*+ NO_BKA(`t2`@`inner_qb`) NO_BKA(`t3`@`select#1`) */ SUM(t3.a) FROM
    (SELECT /*+ QB_NAME(inner_qb)*/ t1.b FROM t1 JOIN t2 ON t1.a = t2.a) AS dt
    JOIN t3 ON t3.a = dt.b;
```

or

```
SELECT /*+ NO_BKA(`t2`@`inner_qb`, `t3`@`select#1`) */ SUM(t3.a) FROM
    (SELECT /*+ QB_NAME(inner_qb)*/ t1.b FROM t1 JOIN t2 ON t1.a = t2.a) AS dt
    JOIN t3 ON t3.a = dt.b;
```

It is also possible to assign a name to the topmost query block and refer each table by the explicit block name:

```
SELECT /*+ QB_NAME(topmost) NO_BKA(`t2`@`inner_qb`, `t3`@`topmost`) */ SUM(t3.a) FROM
    (SELECT /*+ QB_NAME(inner_qb)*/ t1.b FROM t1 JOIN t2 ON t1.a = t2.a) AS dt
    JOIN t3 ON t3.a = dt.b;
```

or use implicit system names of query blocks:

```
SELECT /*+ NO_BKA(`t2`@`select#2`, `t3`@`select#1`) */ SUM(t3.a) FROM
    (SELECT /*+ QB_NAME(qb1)*/ t1.b FROM t1 JOIN t2 ON t1.a = t2.a) AS dt
    JOIN t3 ON t3.a = dt.b;
```

List of Available Table-Level Hints

See description and the list of available table-level hints here.

Syntax of Index-Level Hints

These hints affect the use of all or certain indexes of a table.

Possible syntax variants are:

```
hint_name(table_name [index_name [, index_name] ...])
hint_name(table_name@query_block [index_name [, index_name] ...])
```

`table_name` / `table_name@query_block` is necessary while the list of `index_name`'s can be omitted. In the latter case the hint applies at the table level. However, index-level hints cannot be elevated to the scope of a query block, i.e., syntax `hint_name(@query_block)` is not allowed.

Let us say a user has a table:

```
CREATE TABLE t1 (a INT, b INT, c INT, d INT,
                 KEY i_a(a), KEY i_b(b),
                 KEY i_ab(a,b), KEY i_c(c), KEY i_d(d));
```

and the optimizer chooses range scan of index `i_a` for the query

```
SELECT a FROM t1 WHERE a > 1 AND a < 3;
```

If the user wants to enforce the optimizer employ full scan of `t1`, they can add `NO_INDEX` hint:

```
SELECT /*+ NO_INDEX(t1)*/ a FROM t1 WHERE a > 1 AND a < 3;
```

If for some reason the optimizer chooses a suboptimal index when there is a more efficient one (say, `i_ab`), the user can force the optimizer to choose the preferred index:

```
SELECT /*+ INDEX(t1 i_ab)*/a FROM t1 WHERE a > 1 AND a < 3
```

While `INDEX` / `NO_INDEX` hints control the use of indexes for any operations, `GROUP_INDEX` / `NO_GROUP_INDEX`, `JOIN_INDEX` / `NO_JOIN_INDEX` and `ORDER_INDEX` / `NO_ORDER_INDEX` provide more fine-grained control.

List of Available Index-Level Hints

See description and the list of available index-level hints here.

Join Order Hints

> **Note:**
>
> Available from MariaDB 12.0.

Overview

These hints allow to control the order in which tables of a query are joined.

Generally, these hints follow the syntax rules of table-level hints with some important differences.

Syntax of the `JOIN_FIXED_ORDER` hint:

```
hint_name([@query_block_name])
```

In contrast to other table-level hints, `JOIN_FIXED_ORDER` does not allow specifying table names in the hint body.

Syntax variants of other join-order hints:

```
hint_name([@query_block_name] tbl_name [, tbl_name] ...)
hint_name(tbl_name[@query_block_name] [, tbl_name[@query_block_name]] ...)
```

Here the query block name may be omitted, but at least one table name must be specified.

`JOIN_FIXED_ORDER()`

Forces the optimizer to join tables using the order in which they appear in the `FROM` clause. This is the same as specifying `SELECT STRAIGHT_JOIN`.

Examples

```
-- Tables will be joined in the order `t2` -> `t1` even if order `t1` -> `t2` looks more promising:
SELECT /*+ JOIN_FIXED_ORDER() */ f1, f2
  FROM t2 JOIN t1 ON t1.id = t2.id ORDER BY f1, f2;
```

`JOIN_ORDER()`

Instructs the optimizer to join tables using the specified table order. The hint applies to the named tables. The optimizer may place tables that are not named anywhere in the join order, including between specified tables.

`JOIN_PREFIX()`

Instructs the optimizer to join tables using the specified table order for the first tables of the join execution plan. The hint applies to the named tables. The optimizer places all other tables after the named tables.

`JOIN_SUFFIX()`

Instructs the optimizer to join tables using the specified table order for the last tables of the join execution plan. The hint applies to the named tables. The optimizer places all other tables before the named tables. Example:

```
SELECT /*+ JOIN_PREFIX(t2, t5@subq2)
           JOIN_ORDER(t4@subq2, t1)
           JOIN_SUFFIX(t3)*/ count(*)
             FROM t1 JOIN t2 JOIN t3
               WHERE t1.a IN (SELECT /*+ QB_NAME(subq1) */ a FROM t4)
                 AND t2.a IN (SELECT /*+ QB_NAME(subq2) */ a FROM t5);
```

Subquery Hints

> **Note:**
>
> Available from MariaDB 12.0.

Overview

Subquery hints determine:

- If Semi-join subquery optimizations are to be used;
- Which semijoin strategies are permitted;
- When semijoins are not used, whether to use subquery materialization or IN-TO-EXISTS transformation.

`SEMIJOIN()`, `NO_SEMIJOIN()`

Syntax

```
hint_name([@query_block_name] [strategy [, strategy] ...])
```

- `hint_name`: `SEMIJOIN` or `NO_SEMIJOIN`.
- `strategy`: name of the strategy to be enabled (in case of `SEMIJOIN` hint) or disabled (in case of `NO_SEMIJOIN` hint). The following strategy names are permitted: `DUPSWEEDOUT`, `FIRSTMATCH`, `LOOSESCAN`, `MATERIALIZATION`.

For `SEMIJOIN` hints, if no strategies are named, semi-join is used based on the strategies enabled according to the `optimizer_switch` system variable, if possible. If strategies are named, but inapplicable for the statement, `DUPSWEEDOUT` is used.

For `NO_SEMIJOIN` hints, semi-join is not used if no strategies are named. If named strategies rule out all applicable strategies for the statement, `DUPSWEEDOUT` is used.

If a subquery is nested within another, and both are merged into a semi-join of an outer query, any specification of semi-join strategies for the innermost query are ignored. `SEMIJOIN` and `NO_SEMIJOIN` hints can still be used to enable or disable semi-join transformations for such nested subqueries.

Examples

```
SELECT /*+ NO_SEMIJOIN(@subq1 FIRSTMATCH, LOOSESCAN) */ * FROM t2
  WHERE t2.a IN (SELECT /*+ QB_NAME(subq1) */ a FROM t3);

SELECT /*+ SEMIJOIN(@subq1 MATERIALIZATION, DUPSWEEDOUT) */ * FROM t2
  WHERE t2.a IN (SELECT /*+ QB_NAME(subquery1) */ a FROM t3);
```

`SUBQUERY()`

Syntax

```
SUBQUERY([@query_block_name] strategy)
```

- `strategy`: allowed values are `INTOEXISTS` and `MATERIALIZATION`.

Examples

```
SELECT id, a IN (SELECT /*+ SUBQUERY(MATERIALIZATION) */ a FROM t1) FROM t2;

SELECT /*+ SUBQUERY(@qb1 INTOEXISTS) */* FROM t2 WHERE t2.a IN (SELECT /*+ QB_NAME(qb1)*/ a FROM t1);
```

Global Hints

`MAX_EXECUTION_TIME()`

> **Note:**
>
> Available from MariaDB 12.0.

Global hint to limit query execution time:

```
SELECT /*+ MAX_EXECUTION_TIME(milliseconds) */ ...  ;
```

A query that doesn't finish in the time specified will be aborted with an error.

However, if `@@max_statement_time` system variable is set, the hint will be ignored and a warning produced. Note that this contradicts the stated principle that "new-style hints are more specific than server variable settings, so they override the server variable settings".

This page is licensed: CC BY-SA / Gnu FDL

Index-Level Hints

Overview

These hints follow the syntax of index-level hints which includes two variants:

```
hint_name(table_name [index_name [, index_name] ...])
hint_name(table_name@query_block [index_name [, index_name] ...])
```

They enable or disable the specified indexes for index scans for specific operations. The following rules apply:

- for *enabling (or whitelisting)* hints (`GROUP_INDEX` , `JOIN_INDEX` , `INDEX` , `ORDER_INDEX`):
  - if the hint body includes only a table name and no index names, the optimizer is free to choose any applicable index of that table;
  - if the body includes both a table name and one or more index names, the optimizer will only consider the specified indexes for that table;
- for *disabling (or blacklisting)* hints (`NO_GROUP_INDEX` , `NO_JOIN_INDEX` , `NO_INDEX` , `NO_ORDER_INDEX`):
  - if the hint body includes only a table name and no index names, the optimizer will not use any indexes for that table;
  - if the body includes both a table name and one or more index names, the optimizer will not use the specified indexes for that table but can use any other not listed.

`GROUP_INDEX()` , `NO_GROUP_INDEX()`

> **Note:**
>
> These hints are available from MariaDB 12.1.

These hints enable or disable the specified indexes for index scans for `GROUP BY` operations but do not affect other operations: access to table data and sorting. Equivalent to old-style hints `FORCE INDEX FOR GROUP BY` and `IGNORE INDEX FOR GROUP BY` .

Examples

```
-- Do not use indexes of `t1` for the `GROUP BY` operation
SELECT /*+ NO_GROUP_INDEX(t1)*/ a, MAX(b) FROM t1 GROUP BY a;
```

```
-- Force `idx_ab` index to be used for `GROUP BY`, although other
-- indexes may be also applicable
SELECT /*+ GROUP_INDEX(t1 idx_ab) */ a, MAX(b) FROM t1 GROUP BY a;
```

`JOIN_INDEX()` , `NO_JOIN_INDEX()`

> **Note:**
>
> These hints are available from MariaDB 12.1.

Enable or disable the specified indexes for an access method (`range` , `ref` , etc.). Equivalent to old-style hints `FORCE INDEX FOR JOIN` and `IGNORE INDEX FOR JOIN` .

Example

```
-- Force the optimizer to consider only `idx1` and `idx2` indexes to access
-- table `t1` data, although there may be other indexes applicable
SELECT /*+ JOIN_INDEX(t1 idx1, idx2)*/a FROM t1 WHERE a > 1 AND a < 3;
```

`INDEX()` , `NO_INDEX()`

> **Note:**
>
> These hints are available from MariaDB 12.1.

Enable or disable the specified indexes for all scopes (access to table data, grouping and sorting). Equivalent to old-style hints `FORCE INDEX` and `IGNORE INDEX`.

Examples

```
-- Forbid using indexes `i_a` and `i_ab` but do not prevent the optimizer
-- from using any other applicable indexes
SELECT /*+ NO_INDEX(t1 i_a, i_ab)*/ a, MAX(b) FROM t1 WHERE a > 1 AND a < 3 GROUP BY a;
```

```
-- Force the optimizer to use index `i_a`
UPDATE /*+ INDEX(t1 i_a) */ t1 SET d = 1 WHERE a = 1 AND b = 2 AND c = 3;
```

```
-- Do not allow the optimizer to use index `i_ab`, however any other
-- applicable index can be used
DELETE /*+ NO_INDEX(t1 i_ab) */ FROM t1 WHERE a = 1 AND b = 2 AND c = 3;
```

`ORDER_INDEX()`, `NO_ORDER_INDEX()`

> **Note:**
>
> These hints are available from MariaDB 12.1.

These hints enable or disable the specified indexes for sorting rows but do not affect other operations: access to table data and grouping. Equivalent to old-style hints `FORCE INDEX FOR ORDER BY` and `IGNORE INDEX FOR ORDER BY`.

Example

```
-- Forbid using of index `i_ab` for sorting. However, the optimizer is free
-- to use any other available index
SELECT /*+ NO_ORDER_INDEX(t1 i_ab) */ a FROM t1 ORDER BY a;
```

Combining Multiple Index Hints

Multiple index hints can be combined in a single query unless they conflict with each other. For example:

```
-- No indexes are allowed for grouping, index `i_ab` is forbidden for ordering,
-- but all indexes are usable for data access
SELECT /*+ NO_GROUP_INDEX(t1), NO_ORDER_INDEX(t1 i_ab) */ a
FROM t1 WHERE a > 1 AND a < 3
GROUP BY a
ORDER BY a;
```

Here the hints do not conflict with each other because they target different operations.

A conflict may happen when the same class of hints is applied to a single table, for example:

```
SELECT /*+ JOIN_INDEX(t1 i_a) NO_JOIN_INDEX(t1 i_d) */ a FROM t1;
```

or when an umbrella hint `INDEX`/`NO_INDEX` is specified:

```
SELECT /*+ INDEX(t1 i_a) JOIN_INDEX(t1 i_d) */ a FROM t1 WHERE a < 3;
```

Here the warning will be generated:

```
Warning 4238    Hint JOIN_INDEX(`t1` `i_d`) is ignored as conflicting/duplicated (an index hint of the
same type has already been specified for this table)
```

because `INDEX` hint is essentially a combination of `JOIN_INDEX`|`GROUP_INDEX`|`ORDER_INDEX`.

If a single SQL query includes both old-style and new-style index hints, old-style hints are silently ignored, for example:

```
SELECT /*+ INDEX(t1 i_a) */ * FROM t1 IGNORE INDEX(i_a) WHERE a = 1 AND b = 2 AND c = 3;
```

New-style `INDEX(t1 i_a)` has effect while old-style `IGNORE INDEX(i_a)` does not.

`INDEX_MERGE()`, `NO_INDEX_MERGE()`

> **Note:**
>
> This hint is available from MariaDB 12.2.

The `INDEX_MERGE` and `NO_INDEX_MERGE` optimizer hints provide granular control over the optimizer's use of index merge strategies. They allow users to override the optimizer's cost-based calculations and global switch settings, to force or prevent the merging of indexes for specific tables.

Syntax

```
/*+ INDEX_MERGE(table_name [index_name, ...]) */
/*+ NO_INDEX_MERGE(table_name [index_name, ...]) */
```

Behavior

The hints operate by modifying the set of keys the optimizer considers for merge operations. The specific behavior depends on whether specific index keys are provided within the hint.

INDEX_MERGE Hint

This hint instructs the optimizer to employ an index merge strategy.

- Without arguments: When specified as `INDEX_MERGE(tbl)`, the optimizer considers all available keys for that table and selects the cheapest index merge combination.
- With specific keys: When specified with keys, for instance, `INDEX_MERGE(tbl key1, key2)`, the optimizer considers only the listed keys for the merge operation. All other keys are excluded from consideration for index merging.

  > The `INDEX_MERGE` hint overrides the global optimizer_switch. Even if a specific strategy (such as index_merge_intersection) is disabled globally, the hint forces the optimizer to attempt the strategy using the specified keys.

NO_INDEX_MERGE Hint

This hint instructs the optimizer to avoid index merge strategies.

- Without arguments: When specified as `NO_INDEX_MERGE(tbl)`, index merge optimizations are completely disabled for the specified table.
- With specific keys: When specified with keys, for instance, `NO_INDEX_MERGE(tbl key1)`, the listed keys are excluded from consideration. The optimizer may still perform a merge using other available keys. However, if excluding the listed keys leaves insufficient row-ordered retrieval (ROR) scans available, no merge is performed.

Algorithm Selection and Limitations

While these hints control which keys are candidates for merging, they do not directly dictate the specific merge algorithm (Intersection, Union, or Sort-Union).

- Indirect Control: You can influence the strategy indirectly by combining these hints with optimizer_switch settings, but specific algorithm selection is not guaranteed.
- Invalid Hints: If a hint directs the optimizer to use specific indexes, but those indexes do not provide sufficient ROR scans to form a valid plan, the server is unable to honor the hint. In this scenario, the server emits a warning.

Examples

In the following examples, the index_merge_intersection switch is globally disabled. However, the `INDEX_MERGE` hint forces the optimizer to consider specific keys (`f2` and `f4`), resulting in an intersection strategy.

You can see that we disable intersection with `NO_INDEX_MERGE` for the following query and the behavior reflects in the EXPLAIN output. The query after that shows with the hint enabling merge–an intersection of `f3,f4` is used. In the last example, a different intersection is used: `f3,PRIMARY` .

No intersection (no merged indexes):

```
MariaDB [test]> EXPLAIN SELECT /*+ NO_INDEX_MERGE(t1 f2, f4, f3) */ COUNT(*) FROM t1 WHERE f4 = 'h' AND
f3 = 'b' AND f5 = 'i'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
         type: ref
possible_keys: PRIMARY,f3,f4
          key: f3
      key_len: 9
          ref: const,const
         rows: 1
        Extra: Using index condition; Using where
1 row in set (0.009 sec)
```

Intersection of keys `f3, f4` :

```
MariaDB [test]> EXPLAIN SELECT /*+ INDEX_MERGE(t1 f2, f4, f3) */ COUNT(*) FROM t1 WHERE f4 = 'h' AND f3 =
'b' AND f5 = 'i'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
         type: index_merge
possible_keys: PRIMARY,f3,f4
          key: f3,f4
      key_len: 9,9
          ref: NULL
         rows: 1
        Extra: Using intersect(f3,f4); Using where; Using index
1 row in set (0.010 sec)
```

Intersection of keys `PRIMARY, f3` :

```
MariaDB [test]> EXPLAIN SELECT COUNT(*) FROM t1 WHERE f4 = 'h' AND f3 = 'b' AND f5 = 'i'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
         type: index_merge
possible_keys: PRIMARY,f3,f4
          key: f3,PRIMARY
      key_len: 9,4
          ref: NULL
         rows: 1
        Extra: Using intersect(f3,PRIMARY); Using where
1 row in set (0.006 sec)
```

`NO_ICP()`

This hint is available from MariaDB 12.0.

Disables Index Condition Pushdown for a specified table or a subset of its indexes. Effectively overrides the optimizer_switch setting `index_condition_pushdown=on`.

Examples

```
-- By default, ICP is enabled and we can see "Using index condition" in the "Extra" field of EXPLAIN
output
> EXPLAIN SELECT * FROM tbl WHERE key_col1 BETWEEN 10 AND 11 AND key_col2 LIKE '%foo%';
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
| id | select_type | table | type  | possible_keys | key      | key_len | ref  | rows | Extra
|
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
|  1 | SIMPLE      | tbl   | range | key_col1      | key_col1 | 5       | NULL |    2 | Using index
condition |
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+

-- Hint disables ICP, so there is no more "Using index condition" in the EXPLAIN output
> EXPLAIN SELECT /*+ NO_ICP(tbl)*/ * FROM tbl WHERE key_col1 BETWEEN 10 AND 11 AND key_col2 LIKE '%foo%';
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
| id | select_type | table | type  | possible_keys | key      | key_len | ref  | rows | Extra
|
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
|  1 | SIMPLE      | tbl   | range | key_col1      | key_col1 | 5       | NULL |    2 |
|
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+

-- Same effect is achieved by specifying both table name and index name
> EXPLAIN SELECT /*+ NO_ICP(tbl key_col1)*/ * FROM tbl WHERE key_col1 BETWEEN 10 AND 11 AND key_col2 LIKE
'%foo%';
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
| id | select_type | table | type  | possible_keys | key      | key_len | ref  | rows | Extra
|
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
|  1 | SIMPLE      | tbl   | range | key_col1      | key_col1 | 5       | NULL |    2 |
|
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
```

`NO_RANGE_OPTIMIZATION()`

> **Note:**
>
> This hint is available from MariaDB 12.0.

An index-level hint that disables range optimization for certain index(es) or a whole table. There is no corresponding optimizer switch setting for this optimization.

Examples

```
-- By default, the optimizer chooses `PRIMARY` key for `range` access to `t1`
> EXPLAIN SELECT f1 FROM t3 WHERE f1 > 30 AND f1 < 33;
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
id   | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
1    |  SIMPLE     |  t3   | range | PRIMARY,f2_idx|  PRIMARY |    4    | NULL |   2  |     100.00  |
Using where; Using index
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+

-- Exclude `PRIMARY` key from indexes allowed for `range` access, `f2_idx` is used instead
EXPLAIN EXTENDED SELECT /*+ NO_RANGE_OPTIMIZATION(t3 PRIMARY) */ f1 FROM t3 WHERE f1 > 30 AND f1 < 33;
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
id   | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
1    |  SIMPLE     |  t3   | range | PRIMARY,f2_idx| f2_idx   |   4     |  NULL|  2   |   100.00    |
Using where; Using index
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+

-- This hint excludes all indexes applicable for `range` access, so the optimizer switches to the `index`
access
EXPLAIN EXTENDED SELECT /*+ NO_RANGE_OPTIMIZATION(t3 PRIMARY, f2_idx) */ f1 FROM t3 WHERE f1 > 30 AND f1
< 33;
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
id   | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
1    | SIMPLE      | t3    | index | PRIMARY,f2_idx|  PRIMARY |  4      | NULL | 56   |   4.11      |
Using where; Using index

-- The hint disables range access for any indexes of `t3`, `index` access is used as in the test above
EXPLAIN EXTENDED SELECT /*+ NO_RANGE_OPTIMIZATION(t3) */ f1 FROM t3 WHERE f1 > 30 AND f1 < 33;
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
id   | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
1    | SIMPLE      | t3    | index | PRIMARY,f2_idx|  PRIMARY |  4      | NULL | 56   |   4.11      |
Using where; Using index
```

`MRR()`, `NO_MRR()`

> **Note:**
>
> These hints are available from MariaDB 12.0.

These hints allow to control the multi-range read optimization, effectively overriding the optimizer switch setting `mrr=[on|off]`.

Examples

```
-- MRR optimization is disabled by optimizer_switch setting:
set optimizer_switch='mrr=off';

--Turn on MRR for `t1` table
SELECT /*+ MRR(t1) */ * FROM t1 WHERE f2 <= 3 AND 3 <= f3;

--Turn on MRR for `IDX2` of table `t1`
SELECT /*+ MRR(t1 IDX2) */ * FROM t1 WHERE f2 <= 3 AND 3 <= f3;
```

```
-- MRR optimization is enabled by optimizer_switch setting (default setting):
set optimizer_switch='mrr=on';

-- Disable multi-range read when accessing table `t1`
SELECT /*+ NO_MRR(t1) */ * FROM t1 WHERE f2 <= 3 AND 3 <= f3;
```

`ROWID_FILTER()`, `NO_ROWID_FILTER()`

> **Note:**
>
> These hints are available from MariaDB 12.1.

Allows to control the ROWID filter optimization. Effectively they override the optimizer switch setting `rowid_filter=[on|off]`.

- Rules for `ROWID_FILTER()` hint:
  - if the hint body includes only a table name and no index names, the optimizer is free to choose any indexes applicable for rowid-filtering optimization;
  - if the body includes both a table name and one or more index names, the optimizer will only consider the specified indexes for rowid-filtering;
  - the hint forces the use of rowid-filtering (if applicable) despite being less efficient than not using the optimization (from the optimizer's point of view);
- Rules for `NO_ROWID_FILTER()` hint:
  - if the hint body includes only a table name and no index names, the optimizer will not use any indexes for rowid-filtering;
  - if the body includes both a table name and one or more index names, the optimizer will not use the specified indexes for rowid-filtering but can use any other not listed.

Examples

```
SELECT /*+ rowid_filter(t1)*/ a FROM t1 WHERE a < 'e' AND b > 't';
SELECT /*+ no_rowid_filter(t1)*/ a FROM t1 WHERE a < 'e' AND b > 't';
SELECT /*+ rowid_filter(t1 key_a, key_b)*/ a FROM t1 WHERE a < 'e' AND b > 't';
SELECT /*+ no_rowid_filter(t1 key_c)*/ a FROM t1 WHERE a < 'e' AND b > 't';
```

Interaction with Other Index Hints

If `[NO_]INDEX()`, `[NO_]JOIN_INDEX()` hints are specified for the same table, then only indexes whitelisted by `INDEX()` or `JOIN_INDEX()` can be considered for rowid filters. `ROWID_FILTER(` hint cannot force the use of indexes that are disabled by `NO_INDEX()` / `NO_JOIN_INDEX()` or omitted from `INDEX()` / `JOIN_INDEX()`. For example, if one index should be used for data access and another for a rowid filter, the filter index must be mentioned in *both* hints:

```
SELECT /*+ index(t1 key_a, key_b) rowid_filter(t1 key_b) */ a FROM t1 ....
```

This page is licensed: CC BY-SA / Gnu FDL

Query Block Naming

Optimizer hints can address certain named query blocks within the query. Query block is either a top-level `SELECT` / `UPDATE` / `DELETE` statement or a subquery within the statement.

There are three ways to address query blocks:

- by an explicit name assigned using `QB_NAME()` hint
- by implicit name based on position of the query block in the query
- by implicit name based on alias of derived table, view or CTE

Explicit Query Block Names

- `QB_NAME(query_block_name)` hint is used to assign a name to a query block in which this hint is placed.

Examples

Assigning a name to a top-level `SELECT` statement:

```
SELECT /*+ QB_NAME(foo) */ * FROM t1 ...;
```

Assigning a name to a subquery:

```
SELECT * FROM (SELECT /*+ QB_NAME(subq) */* FROM t1 ...) dt ...;
```

Assigning names to multiple query blocks:

```
SELECT /*+ QB_NAME(qb_outer) */ ...
  FROM (SELECT /*+ QB_NAME(dt1) */ ...
  FROM (SELECT /*+ QB_NAME(dt2) */ ... FROM ...) dt2) dt1 ...
```

The assigned name can then be used to refer to the query block in optimizer hints, for example:

```
SELECT /*+ QB_NAME(qb_outer) NO_ICP(@qb1 t1) BNL(@dt1) INDEX(t2@dt2 idx1) */ ...
  FROM t1, (SELECT /*+ QB_NAME(dt1) */ ...
      FROM (SELECT /*+ QB_NAME(dt2) */ ... FROM t2 ...) dt2) dt1 ...
```

The scope of a query block name is the whole statement. It is invalid to use the same name for multiple query blocks. You can refer to the query block “down into subquery”, “down into derived table”, “up to the parent” and “to a right sibling in the `UNION`”. You cannot refer “to a left sibling in a `UNION`”.

Limitations

Query block names declared inside view definitions (`CREATE VIEW ...`) are only visible inside the view. They are not accessible from outside the view.

Explicit Query Block Names With Path

> **Note:**
>
> Available from MariaDB 13.0.

`QB_NAME(query_block_name, query_block_path)` hint is used to assign names to query blocks nested within views, derived tables, and CTEs

Syntax

```
QB_NAME(name, query_block_path)
```

where

```
query_block_path ::= query_block_path_element
                     [ {. query_block_path_element }... ]
query_block_path_element ::= @ qb_path_element_select_num |
                               qb_path_element_view_sel
qb_path_element_view_sel ::= qb_path_element_view_name
                             [ @ qb_path_element_select_num ]
```

`query_block_path` is similar to a path in filesystem in some sense. Each next element descends further into the query blocks structure. Path elements are separated with dots (`.`).

Each path element can be one of:

- View/derived table/CTE name.
- Select number in the current query block, such as `@SEL_1` , `@SEL_2` , etc.
- Combination of view/derived table/CTE name and select number, such as `v1@SEL_1` , `dt1@SEL_2` , etc.

Examples

```
CREATE VIEW v1 AS SELECT * FROM t1 WHERE a < 1000;

-- The name `qb_v1` is assigned to the inner query block of the view `v1`
SELECT /*+ qb_name(qb_v1, v1) no_index(t1@qb_v1)*/* FROM v1;

-- This means the same but specifies that `v1` is present in @SEL_1
-- of the current query block. So, @SEL_1 can be omitted here
SELECT /*+ qb_name(qb_v1, v1@SEL_1) no_index(t1@qb_v1)*/* FROM v1;

-- However, here we have two occurrences of `v1` in different query blocks
-- (@SEL_1 and @SEL_2), so specifying @SEL_1 is essential
SELECT /*+ qb_name(qb_v1, v1@SEL_1) no_index(t1@qb_v1)*/* FROM v1 WHERE a < 10
UNION
SELECT * FROM v1 WHERE b > 100;
```

Path consisting of two elements:

```
-- The view has two query blocks
CREATE VIEW v1 AS SELECT * FROM t1 WHERE a < 10
                  UNION
                  SELECT * FROM t2 WHERE b > 20;

-- This hint means that SELECT#2 of view `v1`, which is present in
-- SELECT#1 of the current query block, gets the name `qb_v1`.
SELECT /*+ qb_name(qb_v1, v1@sel_1 .@sel_2) */ * FROM v1;
```

Views and derived tables may be nested on multiple levels, for example:

```
-- The path follows `dt1` -> `dt2` -> `v2` -> `@SEL_2` of `v2`
SELECT /*+ qb_name(dt2_dt1_v1_1, dt1 .dt2 .v2 .@SEL_2)
           no_index(t1@dt2_dt1_v1_1)*/ v1.*
FROM v1
     JOIN (SELECT v1.* FROM v1
           JOIN (SELECT * FROM v2) dt2
          ) dt1
```

Limitations

- Only SELECT statements support QB names with path. DML operations (UPDATE, DELETE, INSERT) do not support them
- QB names with path are not supported inside view definitions (`CREATE VIEW ...`)

Implicit Names Based on Position

> **Note:**
>
> Available from MariaDB 12.2.

Besides the given name, any query block has an implicit name `select_#n` (where `#n` stands for a number). You can see these numbers in the column `id` of the `EXPLAIN` output:

```
-- `select#1` addresses the topmost query block joining tables `e` and `s`.
-- The hint disables merging of derived table `(select * from salaries) s`
EXPLAIN
SELECT /*+ NO_MERGE(s@`select#1`) */ e.emp_name, s.salary
FROM employees e JOIN (select * from salaries) s ON e.emp_id = s.emp_id
WHERE s.salary > (SELECT AVG(salary) FROM salaries);
+----+-------------+-------+-------+---------------+----------+---------+------+------+------------------
-----+
| id | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+------------------
-----+
|  1 | PRIMARY     | e     | ALL   | PRIMARY       | NULL     | NULL    | NULL |    5 |   100.00    |
NULL
|  1 | PRIMARY     | <derived2> | ref | key0          | key0     | 5       | test.e.emp_id |    1 |
100.00    | Using where
|  3 | SUBQUERY    | salaries | ALL | NULL          | NULL     | NULL    | NULL |    5 |   100.00    |
NULL
|  2 | DERIVED     | salaries | ALL | NULL          | NULL     | NULL    | NULL |    5 |   100.00    |
NULL
+----+-------------+-------+-------+---------------+----------+---------+------+------+------------------
-----+
```

```
-- `select#2` addresses the query block corresponding to derived table `T`
EXPLAIN SELECT /*+ JOIN_ORDER(@`select#2` twenty,ten) */  *
FROM (SELECT ten.a AS a FROM (ten JOIN twenty)
      WHERE (ten.a = twenty.a) LIMIT 1000 ) T;
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
| id | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
|  1 | PRIMARY     | <derived2> | ALL | NULL          | NULL     | NULL    | NULL |  200 |   100.00    |
NULL
|  2 | DERIVED     | twenty  | ALL | NULL          | NULL     | NULL    | NULL |   20 |   100.00    |
NULL
|  2 | DERIVED     | ten     | ALL | NULL          | NULL     | NULL    | NULL |   10 |   100.00    |
Using where; Using join buffer (flat, BNL join)
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
```

Limitations

- The positional numbers of query blocks are not stable and can change if the query is modified or its execution plan changes (for example, due to statistics re-collection). If a more stable query block addressing is needed then explicit naming is probably a better choice.
- Implicit names based on position are not supported inside view definitions (`CREATE VIEW ...`).

Implicit Names Based on Aliases

> **Note:**
>
> Available from MariaDB 13.0

It is possible to address a query block corresponding to a derived table, view or CTE using its name or an alias in the query.

Examples

```
-- Addressing a table inside a derived table using implicit QB name
SELECT /*+ no_index(t1@dt) */ *
  FROM (SELECT * FROM t1 WHERE a > 10) AS DT;
-- this is an equivalent to:
SELECT /*+ no_index(t1@dt) */ * FROM
  (SELECT /*+ qb_name(dt)*/ * FROM t1 WHERE a > 10) AS DT;
-- Addressing a query block corresponding to the derived table
SELECT /*+ no_bnl(@dt) */ *
  FROM (SELECT * FROM t1, t2 WHERE t1.a > t2.a) AS DT;

-- View
CREATE VIEW v1 AS SELECT * FROM t1 WHERE a > 10 AND b > 100;

-- referencing a table inside a view by implicit QB name:
SELECT /*+ index_merge(t1@v1 idx_a, idx_b) */ *
  FROM v1, t2 WHERE v1.a = t2.a;
-- could be equivalent to, if query blocks names specified inside view definition
-- were accessible from the outer query:
CREATE VIEW v1 AS SELECT /*+ qb_name(qb_v1) */ *
  FROM t1 WHERE a > 10 AND b > 100;
SELECT /*+ index_merge(t1@qb_v1 idx_a, idx_b) */ *
  FROM v1, t2 WHERE v1.a = t2.a;

-- CTE
WITH aless100 AS (SELECT a FROM t1 WHERE b <100)
  SELECT /*+ index(t1@aless100) */ * FROM aless100;
-- equivalent to:
WITH aless100 AS (SELECT /*+ qb_name(aless100) */ a FROM t1 WHERE b <100)
  SELECT /*+ index(t1@aless100) */ * FROM aless100;
```

Limitations

- Only `SELECT` statements support implicit QB names based on aliases. DML operations (`UPDATE`, `DELETE`, `INSERT`) do not support them.
- Implicit names based on aliases are not supported inside view definitions (`CREATE VIEW ...`).

This page is licensed: CC BY-SA / Gnu FDL

Table-Level Hints

Supported variants of syntax:

```
hint_name([@query_block_name])
hint_name([@query_block_name] [tbl_name [, tbl_name] ...])
hint_name([tbl_name@query_block_name [, tbl_name@query_block_name] ...])
```

See Syntax of query-block and table level hints for more information.

`BKA()`, `NO_BKA()`

> **Note:**
>
> Available from MariaDB 12.0.

Enable or disable batched key access join algorithm.

`BNL()`, `NO_BNL()`

> **Note:**
>
> Available from MariaDB 12.0.

Enable or disable block nested-loop join algorithm.

These hints override the settings of system variables used to manage usage of block-based join algorithms.

Examples

```
SET join_cache_level = 2;
SELECT /*+ BKA(t13) */ * FROM t12, t13
  WHERE t12.a=t13.a AND (t13.b+1 <= t13.b+1);
```

`join_cache_level = 2` limits the use of block-based join algorithms to flat or incremental BNL (BKA joins are not allowed). However, with the `BKA(t13)` hint tables `t12` and `t13` will be joined using the BKA hashed algorithm.

The same works in the opposite direction too. Say we have a very permissive setting `join_cache_level = 8` that allows all block-based join algorithms. However, with the `NO_BKA(t13)` hint tables `t12` and `t13` will be joined using the BNL hashed algorithm.

```
SET join_cache_level = 8;
SET optimizer_switch='join_cache_bka=on';
SELECT /*+ NO_BKA(t13) */ * FROM t12, t13
  WHERE t12.a=t13.a AND (t13.b+1 <= t13.b+1);
```

These hints can be applied either to particular tables or to the whole query-block. When applied to particular tables, the hint body should mention tables that *receive* data from previous tables, i.e. which appear later in the join order. That means the hints in examples above will work if the optimizer joins tables `t12` and `t13` in that order. If the optimizer joins tables in the reverse order, the hints should mention `t12` in their body instead.

`DERIVED_CONDITION_PUSHDOWN()`, `NO_DERIVED_CONDITION_PUSHDOWN()`

> **Note:**
>
> Available from MariaDB 12.1.

These hints enable or disable the Condition Pushdown into Derived Table Optimization, effectively overriding the optimizer switch setting `condition_pushdown_for_derived=[on|off]`. They apply to derived tables, views and CTEs of the query.

Examples

```
SELECT /*+ NO_DERIVED_CONDITION_PUSHDOWN(v1) */ *
  FROM v1, t2 WHERE
  ((v1.max_c>300) AND (v1.avg_c>t2.d) AND (v1.b=t2.b)) OR
    ((v1.max_c<135) AND (v1.max_c<t2.c) AND (v1.a=t2.a));
```

Here the hint disables pushing conditions `(max_c > 300 or max_c < 135)` into the view `v1`.

In the next example the optimization is disabled by the optimizer switch setting, however the hint allows pushing conditions into the view `v1` inside `cte`:

```
set optimizer_switch='condition_pushdown_for_derived=off';

explain format=JSON
WITH cte AS (
  SELECT /*+ QB_NAME(qb1) */ max_c, avg_c
  FROM v1,t2
  WHERE ((v1.max_c>300) and (v1.avg_c>t2.d) and (v1.b=t2.b)) or
    ((v1.max_c<135) and (v1.max_c<t2.c) and (v1.a=t2.a))
)
SELECT /*+ DERIVED_CONDITION_PUSHDOWN(@qb1) */ * FROM cte;
```

Example with pushing down conditions into a derived table:

```
SELECT /*+ NO_DERIVED_CONDITION_PUSHDOWN(dt) */ * FROM (
  SELECT qb.b AS a
  FROM (
    SELECT 1 * t4.b AS b, 1 * t4.a AS a FROM t4
  ) qb
  GROUP BY qb.a
) dt WHERE (dt.a = 2);
```

`MERGE()`, `NO_MERGE()`

> **Note:**
>
> Available from MariaDB 12.1.

These hints enable or disable the Derived Table Merge Optimization, effectively overriding the optimizer switch setting `derived_merge=[on|off]`.

Examples

Let's say there is a statement:

```
EXPLAIN EXTENDED
SELECT a, b
FROM (SELECT a, b FROM t1) AS dt
WHERE a < 3 AND b > 8;

+----+-------------+-------+-------+---------------+----------+---------+------+------+------------------
-----+
| id | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+------------------
-----+
|  1 | SIMPLE      | t1    | ALL   | NULL          | NULL     | NULL    | NULL |   10 |     100.00  |
Using where
+----+-------------+-------+-------+---------------+----------+---------+------+------+------------------
-----+
```

By default the derived table `dt` gets merged into the main `SELECT`, so the output of `EXPLAIN` displays only one line.

However, if the user does not want `dt` to be merged, they can add `NO_MERGE()` hint:

```
EXPLAIN EXTENDED
SELECT /*+ NO_MERGE(dt)*/ a, b
FROM (SELECT a, b FROM t1) AS dt
WHERE a < 3 AND b > 8;

+----+-------------+-------+-------+---------------+----------+---------+------+------+----------
-----+
| id | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+----------
-----+
|  1 | PRIMARY     | <derived2> | ALL | NULL          | NULL     | NULL    | NULL |   10 |     100.00  |
Using where
|  2 | DERIVED     | t1    | ALL   | NULL          | NULL     | NULL    | NULL |   10 |     100.00  |
Using where
+----+-------------+-------+-------+---------------+----------+---------+------+------+----------
-----+
```

Now the output displays two lines indicating that `dt` has not been merged but materialized instead. The same effect could be achieved by applying query-block level variant of the hint: `/*+ NO_MERGE()*/` (however, the hint would affect not only `dt` but all other derived tables of the query block, if there were any).

The hints can be applied to CTE's too:

```
set session optimizer_switch='derived_merge=off';

EXPLAIN
WITH cte AS (SELECT a, b FROM t1)
SELECT /*+ MERGE(cte)*/ a, b
FROM cte WHERE a < 3 AND b > 8;
```

However, they do **not** have effect on merging of views.

`SPLIT_MATERIALIZED()`, `NO_SPLIT_MATERIALIZED()`

> **Note:**
>
> Available from MariaDB 12.0.

These hints control the Split-materialized optimization also known as Lateral Derived optimization.

Let's look at the statement which will employ the lateral derived optimization by default:

```
EXPLAIN
SELECT t1.n1 FROM t1,
   (SELECT n1, n2 FROM t1 WHERE c1 = 'a' GROUP BY n1) AS t
WHERE t.n1 = t1.n1 AND t.n2 = t1.n2 AND c1 = 'a' GROUP BY n1;

+----+-------------+-------+-------+---------------+----------+---------+------+------+----------
-----+
| id | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+----------
-----+
|  1 | PRIMARY     | t1    | ref   | c1,n1_c1_n2   | c1       |       1 | const|    2 |     100.00  |
Using index condition; Using where; Using temporary; Using filesort
|  1 | PRIMARY     | <derived2> | ref | key0          | key0     |       8 | test.t1.n1,test.t1.n2|    1
|     100.00  |
|  2 | LATERAL DERIVED | t1 | ref | c1,n1_c1_n2   | n1_c1_n2 |       4 | test.t1.n1|    1 |     100.00  |
Using where; Using index
+----+-------------+-------+-------+---------------+----------+---------+------+------+----------
-----+
```

`select_type` = `LATERAL DERIVED` at the bottom line of `EXPLAIN` output indicates the optimization has been applied.

If the user applies the hint, the optimization will be disabled and there will be no more `LATERAL DERIVED` in the output:

```
EXPLAIN
SELECT /*+ NO_SPLIT_MATERIALIZED(t) */ t1.n1 FROM t1,
   (SELECT n1, n2 FROM t1 WHERE c1 = 'a' GROUP BY n1) AS t
WHERE t.n1 = t1.n1 AND t.n2 = t1.n2 AND c1 = 'a' GROUP BY n1;

+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
| id | select_type | table |  type | possible_keys | key      | key_len |  ref | rows |  filtered   |
Extra
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
|  1 | PRIMARY     | t1    | ref   | c1,n1_c1_n2   | c1       |       1 | const|    2 |     100.00  |
Using index condition; Using where; Using temporary; Using filesort
|  1 | PRIMARY     | <derived2> | ref | key0          | key0     |       8 | test.t1.n1,test.t1.n2|    1
|     100.00  |
|  2 | DERIVED     | t1    | ref   | c1            | c1       |       1 | const|    2 |     100.00  |
Using index condition; Using where; Using temporary; Using filesort
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
```

As with other hints, users can use `SPLIT_MATERIALIZED()` to force the optimization when it is not chosen by the cost-based optimizer or disabled by default (`optimizer_switch='split_materialized=off'`).

This page is licensed: CC BY-SA / Gnu FDL

Compression

See Compression Plugins.

Compression provider plugins that supply compression algorithms to MariaDB.

See Storage-Engine Independent Column Compression.

Storage-engine-independent compression for BLOB and TEXT columns, independent of the storage engine.

This page is licensed: CC BY-SA / Gnu FDL

Compression Plugins

MariaDB starting with 10.7.0

Compressions plugins were added in a MariaDB 10.7.0 preview release.

The various MariaDB storage engines, such as InnoDB, RocksDB, Mroonga, can use different compression libraries.

Before MariaDB 10.7.0, each separate library would have to be compiled in order to be available for use, resulting in numerous runtime/rpm/deb dependencies, most of which would never be used by users.

From MariaDB 10.7.0, five additional MariaDB compression libraries (besides the default zlib) are available as plugins (note that these affect InnoDB and Mroonga only; RocksDB still uses the compression algorithms from its own library):

- bzip2
- lzma
- lz4

- lzo
- snappy

Installing

Depending on how MariaDB was installed, the libraries may already be available for installation, or may first need to be installed as .deb or .rpm packages, for example:

```
apt-get install mariadb-plugin-provider-lz4
```

Once available, install as a plugin, for example:

```
INSTALL SONAME 'provider_lz4';
```

The compression algorithm can then be used, for example, in InnoDB compression:

```
SET GLOBAL innodb_compression_algorithm = lz4;
```

Upgrading

When upgrading from a release without compression plugins, if a non-zlib compression algorithm was used, those tables will be unreadable until the appropriate compression library is installed. mariadb-upgrade should be run. The `--force` option (to run mariadb-check) or `mariadb-check` itself will indicate any problems with compression, for example:

```
Warning  : MariaDB tried to use the LZMA compression, but its provider plugin is not loaded

Error    : Table 'test.t' doesn't exist in engine

status   : Operation failed
```

or

```
Error    : Table test/t is compressed with lzma, which is not currently loaded.
  Please load the lzma provider plugin to open the table

error    : Corrupt
```

In this case, the appropriate compression plugin should be installed, and the server restarted.

See Also

- 10.7 preview feature: Compression Provider Plugins (mariadb.org blog)
- Add zstd as a compression plugin - MDEV-34290

This page is licensed: CC BY-SA / Gnu FDL

Storage-Engine Independent Column Compression

Storage-engine independent column compression enables TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT, VARCHAR and VARBINARY columns to be compressed.

This is performed by means of a new COMPRESSED column attribute: `COMPRESSED[=<compression_method>]`

Currently the only supported compression method is `zlib`.

Field Length Compatibility

When using the `COMPRESSED` attribute, note that FIELD LENGTH is reduced by 1; for example, a BLOB has a length of 65535, while BLOB COMPRESSED has 65535-1. See MDEV-15592.

New System Variables

`column_compression_threshold`

- Description: Minimum column data length eligible for compression.
- Command line: `--column-compression-threshold=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100`
- Range: `0` to `4294967295`

`column_compression_zlib_level`

- Description: zlib compression level (1 gives best speed, 9 gives best compression).
- Command line: `--column-compression-zlib-level=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `6`
- Range: `1` to `9`

`column_compression_zlib_strategy`

- Description: The strategy parameter is used to tune the compression algorithm. Use the value `DEFAULT_STRATEGY` for normal data, `FILTERED` for data produced by a filter (or predictor), `HUFFMAN_ONLY` to force Huffman encoding only (no string match), or `RLE` to limit match distances to one (run-length encoding). Filtered data consists mostly of small values with a somewhat random distribution. In this case, the compression algorithm is tuned to compress them better. The effect of `FILTERED` is to force more Huffman coding and less string matching; it is somewhat intermediate between `DEFAULT_STRATEGY` and `HUFFMAN_ONLY`. `RLE` is designed to be almost as fast as `HUFFMAN_ONLY`, but give better compression for PNG image data. The strategy parameter only affects the compression ratio but not the correctness of the compressed output even if it is not set appropriately. `FIXED` prevents the use of dynamic Huffman codes, allowing for a simpler decoder for special applications.
- Command line: `--column-compression-zlib-strategy=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `DEFAULT_STRATEGY`
- Valid Values: `DEFAULT_STRATEGY`, `FILTERED`, `HUFFMAN_ONLY`, `RLE`, `FIXED`

`column_compression_zlib_wrap`

- Description: If set to `1` (`0` is default), generate zlib header and trailer and compute adler32 check value. It can be used with storage engines that don't provide data integrity verification to detect data corruption.
- Command line: `--column-compression-zlib-wrap{=0|1}`

- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

New Status Variables

`Column_compressions`

- Description: Incremented each time field data is compressed.
- Scope: Global, Session
- Data Type: `numeric`

`Column_decompressions`

- Description: Incremented each time field data is decompressed.
- Scope: Global, Session
- Data Type: `numeric`

Limitations

- The only supported method currently is zlib.
- The CSV storage engine stores data uncompressed on-disk even if the COMPRESSED attribute is present.
- It is not possible to create indexes over compressed columns.

Comparison with InnoDB Page Compression

Storage-independent column compression is different to InnoDB Page Compression in a number of ways.

- It is storage engine independent, while InnoDB page compression applies to InnoDB only.
- By being specific to a column, one can access non-compressed fields without the decompression overhead.
- Only zlib is available, while InnoDB page compression can offer alternative compression algorithms.
- It is not recommended to use multiple forms of compression over the same data.
- It is intended for compressing large blobs, while InnoDB page compression is suitable for a more general case.
- Columns cannot be indexed, while with InnoDB page compression indexes are possible as usual.

Examples

```
CREATE TABLE cmp (i TEXT COMPRESSED);

CREATE TABLE cmp2 (i TEXT COMPRESSED=zlib);
```

See Also

- InnoDB Page Compression
- InnoDB Compressed Row Format

This page is licensed: CC BY-SA / Gnu FDL

Optimizing Data Structure

See Numeric vs String Fields.

Why choosing numeric over string columns, where possible, improves storage size and comparison speed.

See Optimizing MEMORY Tables.

Optimizing MEMORY-engine tables, including choosing between B-tree and hash indexes.

See Optimizing String and Character Fields.

Optimizing string and character columns, including matching character sets and collations for faster comparisons.

This page is licensed: CC BY-SA / Gnu FDL

Numeric vs String Fields

A large numeric value is stored in far fewer bytes than the equivalent string value. It is therefore faster to move and compare numeric data, so it's best to choose numeric columns for unique id's and other similar fields.

This page is licensed: CC BY-SA / Gnu FDL

Optimizing MEMORY Tables

MEMORY tables are a good choice for data that needs to be accessed often, and is rarely updated. Being in memory, it's not suitable for critical data or for storage, but if data can be moved to memory for reading without needing to be regenerated often, if at all, it can provide a significant performance boost.

The MEMORY Storage Engine has a key feature in that it permits its indexes to be either B-tree or Hash. Choosing the best index type can lead to better performance. See Storage Engine index types for more on the characteristics of each index type.

This page is licensed: CC BY-SA / Gnu FDL

Optimizing String and Character Fields

Comparing String Columns

When values from different columns are compared, the comparison runs more quickly when the columns are of the same character set and collation. If they are different, the strings need to be converted while the query runs. So, where possible, declare string columns using the same character set and collation when you may need to compare them.

VARCHAR vs BLOB

ORDER BY and GROUP BY clauses can generate temporary tables in memory (see MEMORY Storage Engine) if the original table doesn't contain any BLOB fields. If a column is less than 8KB, you can make use of a Binary VARCHAR rather than a BLOB.

This page is licensed: CC BY-SA / Gnu FDL

Optimizing Tables

See OPTIMIZE TABLE.

Complete MariaDB performance optimization guide. Complete reference for query tuning, indexing strategies, and configuration improvements for production use.

See Defragmenting InnoDB Tablespaces.

How to reclaim free space and defragment InnoDB tablespaces after deleting rows.

See Entity-Attribute-Value Implementation.

Implementing the entity-attribute-value (EAV) model in MariaDB, and its trade-offs.

See IP Range Table Performance.

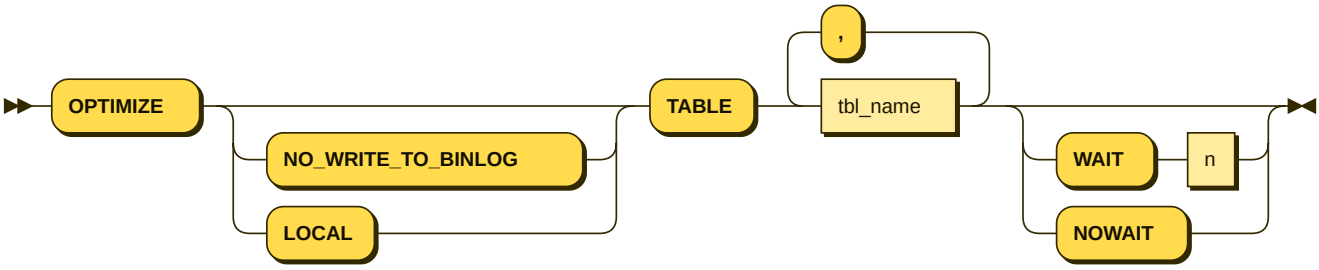
Efficiently storing and querying large sets of non-overlapping ranges, such as IP address ranges.

This page is licensed: CC BY-SA / Gnu FDL

OPTIMIZE TABLE

Syntax

```
OPTIMIZE [NO_WRITE_TO_BINLOG | LOCAL] TABLE
    tbl_name [, tbl_name] ...
    [WAIT n | NOWAIT]
```



Railroad diagram of OPTIMIZE TABLE — equivalent to the BNF above

Description

`OPTIMIZE TABLE` has two main functions. It can either be used to defragment tables, or to update the InnoDB fulltext index.

WAIT/NOWAIT

Set the lock wait timeout. See WAIT and NOWAIT.

Defragmenting

`OPTIMIZE TABLE` works for InnoDB (before MariaDB 10.1.1, only if the innodb_file_per_table server system variable is set), Aria, MyISAM and ARCHIVE tables, and should be used if you have deleted a large part of a table or if you have made many changes to a table with variable-length rows (tables that have VARCHAR, VARBINARY, BLOB, or TEXT columns). Deleted rows are maintained in a linked list and subsequent `INSERT` operations reuse old row positions.

This statement requires SELECT and INSERT privileges for the table.

By default, `OPTIMIZE TABLE` statements are written to the binary log and will be replicated. The `NO_WRITE_TO_BINLOG` keyword (`LOCAL` is an alias) will ensure the statement is not written to the binary log.

`OPTIMIZE TABLE` statements are not logged to the binary log if read_only is set. See also Read-Only Replicas.

`OPTIMIZE TABLE` is also supported for partitioned tables. You can use ALTER TABLE … OPTIMIZE PARTITION to optimize one or more partitions.

You can use `OPTIMIZE TABLE` to reclaim the unused space and to defragment the data file. With other storage engines, `OPTIMIZE TABLE` does nothing by default, and returns this message: ” The storage engine for the table doesn’t support optimize”. However, if the server has been started with the `--skip-new` option, `OPTIMIZE TABLE` is linked to ALTER TABLE, and recreates the table. This operation frees the unused space and updates index statistics.

The Aria storage engine supports progress reporting for this statement.

If a MyISAM table is fragmented, concurrent inserts will not be performed until an `OPTIMIZE TABLE` statement is executed on that table, unless the concurrent_insert server system variable is set to `ALWAYS`.

Updating an InnoDB fulltext index

When rows are added or deleted to an InnoDB fulltext index, the index is not immediately re-organized, as this can be an expensive operation. Change statistics are stored in a separate location. The fulltext index is only fully re-organized when an `OPTIMIZE TABLE` statement is run.

By default, an OPTIMIZE TABLE will defragment a table. In order to use it to update fulltext index statistics, the innodb_optimize_fulltext_only system variable must be set to `1`. This is intended to be a temporary setting and should be reset to `0` once the fulltext index has been re-organized.

Since fulltext re-organization can take a long time, the innodb_ft_num_word_optimize variable limits the re-organization to a number of words (2000 by default). You can run multiple OPTIMIZE statements to fully re-organize the index.

Defragmenting InnoDB tablespaces

MariaDB 10.1.1 merged the Facebook/Kakao defragmentation patch, allowing one to use `OPTIMIZE TABLE` to defragment InnoDB tablespaces. For this functionality to be enabled, the innodb_defragment system variable must be enabled. No new tables are created and there is no need to copy data from old tables to new tables. Instead, this feature loads `n` pages (determined by innodb-defragment-n-pages) and tries to move records so that pages would be full of records and then frees pages that are fully empty after the operation. Note that tablespace files (including ibdata1) will not shrink as the result of defragmentation, but one will get better memory utilization in the InnoDB buffer pool as there are fewer data pages in use.

See Defragmenting InnoDB Tablespaces for more details.

See Also

- Optimize Table in InnoDB with ALGORITHM set to INPLACE
- Optimize Table in InnoDB with ALGORITHM set to NOCOPY
- Optimize Table in InnoDB with ALGORITHM set to INSTANT

This page is licensed: GPLv2, originally from fill_help_tables.sql

Defragmenting InnoDB Tablespaces

Overview

When rows are deleted from an InnoDB table, the rows are simply marked as deleted and not physically deleted. The free space is not returned to the operating system for reuse.

The purge thread will physically delete index keys and rows, but the free space introduced is still not returned to operating system. This can lead to gaps in the pages. If you have variable length rows, new rows may be larger than old rows and cannot make use of the available space.

You can run OPTIMIZE TABLE or ALTER TABLE

ENGINE=InnoDB to reconstruct the table. Unfortunately running `OPTIMIZE TABLE` against an InnoDB table stored in the shared table-space file `ibdata1` does two things:

- Makes the table's data and indexes contiguous inside `ibdata1`.
- Increases the size of `ibdata1` because the contiguous data and index pages are appended to `ibdata1`.

InnoDB Defragmentation

The feature described below has been deprecated in MariaDB 11.0 and was removed in MariaDB 11.1.0. See MDEV-30544 and MDEV-30545.

MariaDB 10.1 merged Facebook's defragmentation code prepared for MariaDB by Matt, Seong Uck Lee from Kakao. The only major difference to Facebook's code and Matt's patch is that MariaDB does not introduce new literals to SQL and makes no changes to the server code. Instead, OPTIMIZE TABLE is used and all code changes are inside the InnoDB/XtraDB storage engines.

The behaviour of `OPTIMIZE TABLE` is unchanged by default, and to enable this new feature, you need to set the innodb_defragment system variable to `1`.

```
[mysqld]
...
innodb-defragment=1
```

No new tables are created and there is no need to copy data from old tables to new tables. Instead, this feature loads `n` pages (determined by innodb-defragment-n-pages) and tries to move records so that pages would be full of records and then frees pages that are fully empty after the operation.

Note that tablespace files (including ibdata1) will not shrink as the result of defragmentation, but one will get better memory utilization in the InnoDB buffer pool as there are fewer data pages in use.

A number of new system and status variables for controlling and monitoring the feature are introduced.

System Variables

- innodb_defragment: Enable InnoDB defragmentation.
- innodb_defragment_n_pages: Number of pages considered at once when merging multiple pages to defragment.
- innodb_defragment_stats_accuracy: Number of defragment stats changes there are before the stats are written to persistent storage.
- innodb_defragment_fill_factor_n_recs: Number of records of space that defragmentation should leave on the page.
- innodb_defragment_fill_factor: Indicates how full defragmentation should fill a page.
- innodb_defragment_frequency: Maximum times per second for defragmenting a single index.

Status Variables

- Innodb_defragment_compression_failures: Number of defragment re-compression failures
- Innodb_defragment_failures: Number of defragment failures.
- Innodb_defragment_count: Number of defragment operations.

Example

```
SET @@global.innodb_file_per_table = 1;
SET @@global.innodb_defragment_n_pages = 32;
SET @@global.innodb_defragment_fill_factor = 0.95;
CREATE TABLE tb_defragment (
pk1 BIGINT(20) NOT NULL,
pk2 BIGINT(20) NOT NULL,
fd4 TEXT,
fd5 VARCHAR(50) DEFAULT NULL,
PRIMARY KEY (pk1),
KEY ix1 (pk2)
) ENGINE=InnoDB;

DELIMITER //
CREATE PROCEDURE innodb_insert_proc (repeat_count INT)
BEGIN
  DECLARE current_num INT;
  SET current_num = 0;
  WHILE current_num < repeat_count DO
    INSERT INTO tb_defragment VALUES (current_num, 1, REPEAT('Abcdefg', 20), REPEAT('12345',5));
    INSERT INTO tb_defragment VALUES (current_num+1, 2, REPEAT('HIJKLM', 20), REPEAT('67890',5));
    INSERT INTO tb_defragment VALUES (current_num+2, 3, REPEAT('HIJKLM', 20), REPEAT('67890',5));
    INSERT INTO tb_defragment VALUES (current_num+3, 4, REPEAT('HIJKLM', 20), REPEAT('67890',5));
    SET current_num = current_num + 4;
  END WHILE;
END//
DELIMITER ;
COMMIT;

SET autocommit=0;
CALL innodb_insert_proc(50000);
COMMIT;
SET autocommit=1;
```

After these CREATE and INSERT operations, the following information can be seen from the INFORMATION SCHEMA:

```
SELECT COUNT(*) AS Value FROM information_schema.innodb_buffer_page
  WHERE table_name LIKE '%tb_defragment%' AND index_name = 'PRIMARY';
Value
313

SELECT COUNT(*) AS Value FROM information_schema.innodb_buffer_page
  WHERE table_name LIKE '%tb_defragment%' AND index_name = 'ix1';
Value
72

SELECT COUNT(stat_value) FROM mysql.innodb_index_stats
  WHERE table_name LIKE '%tb_defragment%' AND stat_name IN ('n_pages_freed');
COUNT(stat_value)
0

SELECT COUNT(stat_value) FROM mysql.innodb_index_stats
  WHERE table_name LIKE '%tb_defragment%' AND stat_name IN ('n_page_split');
COUNT(stat_value)
0

SELECT COUNT(stat_value) FROM mysql.innodb_index_stats
  WHERE table_name LIKE '%tb_defragment%' AND stat_name IN ('n_leaf_pages_defrag');
COUNT(stat_value)
0

SELECT table_name, data_free/1024/1024 AS data_free_MB, table_rows FROM information_schema.tables
  WHERE engine LIKE 'InnoDB' AND table_name LIKE '%tb_defragment%';
TABLE_NAME data_free_MB table_rows
tb_defragment 4.00000000 50051

SELECT table_name, index_name, SUM(number_records), SUM(data_size) FROM
information_schema.innodb_buffer_page
  WHERE table_name LIKE '%tb_defragment%' AND index_name LIKE 'PRIMARY';
TABLE_NAME index_name SUM(number_records) SUM(data_size)
`test`.`tb_defragment` PRIMARY 25873 4739939

SELECT table_name, index_name, SUM(number_records), SUM(data_size) FROM
information_schema.innodb_buffer_page
  WHERE table_name LIKE '%tb_defragment%' AND index_name LIKE 'ix1';
TABLE_NAME index_name SUM(number_records) SUM(data_size)
`test`.`tb_defragment` ix1 50071 1051775
```

Deleting three-quarters of the records, leaving gaps, and then optimizing:

```
DELETE FROM tb_defragment WHERE pk2 BETWEEN 2 AND 4;

OPTIMIZE TABLE tb_defragment;
TABLE   Op  Msg_type    Msg_text
test.tb_defragment  OPTIMIZE    status  OK
SHOW status LIKE '%innodb_def%';
Variable_name   Value
Innodb_defragment_compression_failures  0
Innodb_defragment_failures  1
Innodb_defragment_count 4
```

Now some pages have been freed, and some merged:

```
SELECT COUNT(*) AS Value FROM information_schema.innodb_buffer_page
  WHERE table_name LIKE '%tb_defragment%' AND index_name = 'PRIMARY';
Value
0

SELECT COUNT(*) AS Value FROM information_schema.innodb_buffer_page
  WHERE table_name LIKE '%tb_defragment%' AND index_name = 'ix1';
Value
0

SELECT COUNT(stat_value) FROM mysql.innodb_index_stats
  WHERE table_name LIKE '%tb_defragment%' AND stat_name IN ('n_pages_freed');
COUNT(stat_value)
2

SELECT COUNT(stat_value) FROM mysql.innodb_index_stats
  WHERE table_name LIKE '%tb_defragment%' AND stat_name IN ('n_page_split');
COUNT(stat_value)
2

SELECT COUNT(stat_value) FROM mysql.innodb_index_stats
  WHERE table_name LIKE '%tb_defragment%' AND stat_name IN ('n_leaf_pages_defrag');
COUNT(stat_value)
2

SELECT table_name, data_free/1024/1024 AS data_free_MB, table_rows FROM information_schema.tables
  WHERE engine LIKE 'InnoDB';
TABLE_NAME data_free_MB table_rows
innodb_index_stats 0.00000000 8
innodb_table_stats 0.00000000 0
tb_defragment 4.00000000 12431

SELECT table_name, index_name, SUM(number_records), SUM(data_size) FROM
information_schema.innodb_buffer_page
  WHERE table_name LIKE '%tb_defragment%' AND index_name LIKE 'PRIMARY';
TABLE_NAME index_name SUM(number_records) SUM(data_size)
`test`.`tb_defragment` PRIMARY 690 102145

SELECT table_name, index_name, SUM(number_records), SUM(data_size) FROM
information_schema.innodb_buffer_page
  WHERE table_name LIKE '%tb_defragment%' AND index_name LIKE 'ix1';
TABLE_NAME index_name SUM(number_records) SUM(data_size)
`test`.`tb_defragment` ix1 5295 111263
```

See Defragmenting unused space on InnoDB tablespace on the Mariadb.org blog for more details.

This page is licensed: CC BY-SA / Gnu FDL

Entity-Attribute-Value Implementation

The desires

- Open-ended set of “attributes” (key=value) for each “entity”. That is, the list of attributes is not known at development time, and will grow in the future. (This makes one column per attribute impractical.)
- “ad hoc” queries testing attributes.
- Attribute values come in different types (numbers, strings, dates, etc.)
- Scale to lots of entities, yet perform well.

It goes by various names

- EAV – Entity - Attribute - Value
- key-value
- RDF – This is a flavor of EAV

- MariaDB has dynamic columns that look something like the solution below, with the added advantage of being able to index the columns otherwise hidden in the blob. (There are caveats.)
- MySQL 5.7 Has JSON datatype, plus functions to access parts
- MongoDB, CouchDB – and others – Not SQL-based.

Bad solution

- Table with 3 columns: entity_id, key, value
- The “value” is a string, or maybe multiple columns depending on datatype or other kludges.
- a JOIN b ON a.entity=b.entity AND b.key=‘x’ JOIN c ON … WHERE a.value=… AND b.value=…

The problems

- The SELECTs get messy – multiple JOINs
- Datatype issues – It’s clumsy to be putting numbers into strings
- Numbers stored in VARCHAR do not compare ‘correctly’, especially for range tests.
- Bulky.
- Dedupping the values is clumsy.

A solution

Decide which columns need to be searched/sorted by SQL queries. No, you don’t need all the columns to be searchable or sortable. Certain columns are frequently used for selection; identify these. You probably won’t use all of them in all queries, but you will use some of them in every query.

The solution uses one table for all the EAV stuff. The columns include the searchable fields plus one BLOB. Searchable fields are declared appropriately (INT, TIMESTAMP, etc). The BLOB contains JSON-encoding of all the extra fields.

The table should be InnoDB, hence it should have a PRIMARY KEY. The entity_id is the ‘natural’ PK. Add a small number of other indexes (often ‘composite’) on the searchable fields. PARTITIONing is unlikely to be of any use, unless the Entities should purged after some time. (Example: News Articles)

But what about the ad hoc queries?

You have included the most important fields to search on – date, category, etc. These should filter the data down significantly. When you also need to filter on something more obscure, that will be handled differently. The application code will look at the BLOB for that; more on this later.

Why it works

- You are not really going to search on more than a few fields.
- The disk footprint is smaller; Smaller –> More cacheable –> Faster
- It needs no JOINs
- The indexes are useful
- The one table has one row per entity, and can grow as needed. (EAV needs many rows per entity.)
- Performance is as good as the indexes you have on the ‘searchable fields’.
- Optionally, you can duplicate the indexed fields in the BLOB.
- Values missing from ‘searchable fields’ would need to be NULL (or whatever), and the code would need to deal with such.

Details on the BLOB/JSON

- Build the extra (or all) key-value pairs in a hash (associative array) in your application. Encode it. COMPRESS it. Insert that string into the BLOB.
- JSON is recommended, but not mandatory; it is simpler than XML. Other serializations (eg, YAML) could be used.
- COMPRESS the JSON and put it into a BLOB (or MEDIUMBLOB) instead of a TEXT field. Compression gives about 3x shrinkage.
- When SELECTing, UNCOMPRESS the blob. Decode the string into a hash. You are now ready to interrogate/display any of the extra fields.
- If you choose to use the JSON features of MariaDB or 5.7, you will have to forgo the compression feature described.
- MySQL 5.7.8's JSON native JSON datatype uses a binary format for more efficient access.

Conclusions

- Schema is reasonably compact (compression, real datatypes, less redundancy, etc, than EAV)
- Queries are fast (since you have picked 'good' indexes)
- Expandable (JSON is happy to have new fields)
- Compatible (No 3rd party products, just supported products)
- Range tests work (unlike storing INTs in VARCHARs)
- (Drawback) Cannot use the non-indexed attributes in WHERE or ORDER BY clauses, must deal with that in the app. (MySQL 5.7 partially alleviates this.)

Postlog

Posted Jan, 2014; Refreshed Feb, 2016.

- MariaDB's Dynamic Columns
- MySQL 5.7's JSON

This looks very promising; I will need to do more research to see how much of this article is obviated by it: Using MySQL as a Document Store in 5.7,more DocStore discussion

If you insist on EAV, set optimizer_search_depth=1.

See also

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: eav

This page is licensed: CC BY-SA / Gnu FDL

IP Range Table Performance

The situation

Your data includes a large set of non-overlapping 'ranges'. These could be IP addresses, datetimes (show times for a single station), zipcodes, etc.

You have pairs of start and end values; one 'item' belongs to each such 'range'. So, instinctively, you create a table with start and end of the range, plus info about the item. Your queries involve a WHERE clause that compares for being between the start and end values.

The problem

Once you get a large set of items, performance degrades. You play with the indexes, but find nothing that works well. The indexes fail to lead to optimal functioning because the database does not understand that the ranges are non-overlapping.

The solution

I will present a solution that enforces the fact that items cannot have overlapping ranges. The solution builds a table to take advantage of that, then uses Stored Routines to get around the clumsiness imposed by it.

Performance

The instinctive solution often leads to scanning half the table to do just about anything, such as finding the item containing an 'address'. In complexity terms, this is Order(N).

The solution here can usually get the desired information by fetching a single row, or a small number of rows. It is Order(1).

In a large table, "counting the disk hits" is the important part of performance. Since InnoDB is used, and the PRIMARY KEY (clustered) is used, most operations hit only 1 block.

Finding the 'block' where a given IP address lives:

- For start of block: One single-row fetch using the PRIMARY KEY
- For end of block: Ditto. The record containing this will be 'adjacent' to the other record.

For allocating or freeing a block:

- 2-7 SQL statements, hitting the clustered PRIMARY KEY for the rows containing and immediately adjacent to the block.
- One SQL statement is a DELETE; if hits as many rows as are needed for the block.
- The other statements hit one row each.

Design decisions

This is crucial to the design and its performance:

- Having just one address in the row.
  These were alternative designs; they seemed to be no better, and possibly worse:
- That one address could have been the 'end' address.
- The routine parameters for a 'block' could have be start of this block and start of next block.
- The IPv4 parameters could have been dotted quads; I chose to keep the reference implementation simpler instead.
- The IPv6 parameters are 32-digit hex because it was the simpler that BINARY(16) or IPv5 for a reference implementation.

The interesting work is in the Ips, not the second table, so I focus on it. The inconvenience of JOINing to the second table is small compared to the performance gains.

Details

Two, not one, tables will be used. The first table (`Ips` in the reference implementations) is carefully designed to be optimal for all the basic operations needed. The second table contains other information about the 'owner' of each 'item'. In the reference implementations `owner` is an id used to JOIN the two tables. This discussion centers around `Ips` and how to efficiently map IP(s) to/from owner(s). The second table has "PRIMARY KEY(owner)".

In addition to the two-table schema, there are a set of Stored Routines to encapsulate the necessary code.

One row of Ips represents one ‘item’ by specifying the starting IP address and the ‘owner’. The next row gives the starting IP address of the next “address block”, thereby indirectly providing the ending address for the current block.

This lack of explicitly stating the “end address” leads to some clumsiness. The stored routines hide it from the user.

A special owner (indicated by ‘0’) is reserved for “free” or “not-owned” blocks. Hence, sparse allocation of address blocks is no problem. Also, the ‘free’ owner is handled no differently than real owners, so there are no extra Stored Routines for such.

Links below give “reference” implementations for IPv4 and IPv6. You will need to make changes for non-IP situations, and may need to make changes even for IP situations.

These are the main stored routines provided:

- IpIncr, IpDecr – for adding/subtracting 1
- IpStore – for allocating/freeing a range
- IpOwner, IpRangeOwners, IpFindRanges, Owner2IpStarts, Owner2IpRanges – for lookups
- IpNext, IpEnd – IP of start of next block, or end of current block

None of the provided routines JOIN to the other table; you may wish to develop custom queries based on the given reference Stored Procedures.

The Ips table’s size is proportional to the number of blocks. A million ‘owned’ blocks may be 20-50MB. This varies due to

- number of ‘free’ gaps (between zero and the number of owned blocks)
- datatypes used for `ip` and `owner`
- InnoDB overhead
  Even 100M blocks is quite manageable in today’s hardware. Once things are cached, most operations would take only a few milliseconds. A trillion blocks would work, but most operations would hit the disk a few times – only a few times.

Reference implementation of IPv4

This specific to IPv4 (32 bit, a la ‘196.168.1.255’). It can handle anywhere from ‘nothing assigned’ (1 row) to ‘everything assigned’ (4B rows) ‘equally’ well. That is, to ask the question “who owns ‘11.22.33.44’” is equally efficient regardless of how many blocks of IP addresses exist in the table. (OK, caching, disk hits, etc may make a slight difference.) The one function that can vary is the one that reassigns a range to a new owner. Its speed is a function of how many existing ranges need to be consumed, since those rows will be DELETEd. (It helps that they are, by schema design, ‘clustered’.)

Notes on the Reference implementation for IPv4:

- Externally, the user may use the dotted quad notation (11.22.33.44), but needs to convert to INT UNSIGNED for calling the Stored Procs.
- The user is responsible for converting to/from the calling datatype (INT UNSIGNED) when accessing the stored routine; suggest INET_ATON/INET_NTOA.
- The internal datatype for addresses is the same as the calling datatype (INT UNSIGNED).
- Adding and subtracting 1 (simple arithmetic).
- The datatype of an ‘owner’ (MEDIUMINT UNSIGNED: 0..16M) – adjust if needed.
- The address “Off the end” (255.255.255.255+1 - represented as NULL).
- The table is initialized to one row: (ip=0, owner=0), meaning “all addresses are free
  See the comments in the code for more details.

(The reference implementation does not handle CDRs. Such should be easy to add on, by first turning it into an IP range.)

Reference implementation of IPv6

The code for handling IP address is more complex, but the overall structure is the same as for IPv4. Launch into it only if you need IPv6.

Notes on the reference implementation for IPv6:

- Externally, IPv6 has a complex string, VARCHAR(39) CHARACTER SET ASCII. The Stored Procedure IpStr2Hex() is provided.
- The user is responsible for converting to/from the calling datatype (BINARY(16)) when accessing the stored routine; suggest INET6_ATON/INET6_NTOA.
- The internal datatype for addresses is the same as the calling datatype (BINARY(16)).
- Communication with the Stored routines is via 32-char hex strings.
- Inside the Procedures, and in the Ips table, an address is stored as BINARY(16) for efficiency. HEX() and UNHEX() are used at the boundaries.
- Adding/subtracting 1 is rather complex (see the code).
- The datatype of an 'owner' (MEDIUMINT UNSIGNED: 0..16M); 'free' is represented by 0. You may need a bigger datatype.
- The address "Off the end" (ffff.ffff.ffff.ffff.ffff.ffff.ffff.ffff+1 is represented by NULL).
- The table is initialized to one row: (UNHEX('00000000000000000000000000000000'), 0), meaning "all addresses are free.
- You may need to decide on a canonical representation of IPv4 in IPv6.
  See the comments in the code for more details.

The INET6* functions were first available in MySQL 5.6.3 and MariaDB 10.0.3

Adapting to a different non-IP 'address range' data

- The external datatype for an 'address' should be whatever is convenient for the application.
- The datatype for the 'address' in the table must be ordered, and should be as compact as possible.
- You must write the Stored functions (IpIncr, IpDecr) for incrementing/decrementing an 'address'.
- An 'owner' is an id of your choosing, but smaller is better.
- A special value (such as 0 or '') must be provided for 'free'.
- The table must be initialized to one row: (SmallestAddress, Free)

"Owner" needs a special value to represent "not owned". The reference implementations use "=" and "!=" to compare two 'owners'. Numeric values and strings work nicely with those operators; NULL does not. Hence, please do not use NULL for "not owned".

Since the datatypes are pervasive in the stored routines, adapting a reference implementation to a different concept of 'address' would require multiple minor changes.

The code enforces that consecutive blocks never have the same 'owner', so the table is of 'minimal' size. Your application can assume that such is always the case.

Postlog

Original writing – Oct, 2012; Notes on INET6 functions – May, 2015.

See also

- Related blog
- Another approach
- Free IP tables

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: ipranges

This page is licensed: CC BY-SA / Gnu FDL

Optimizing Queries

See Aborting Statements that Exceed a Certain Time to Execute.

How to abort statements that exceed a maximum execution time.

See Big DELETEs.

Techniques for deleting large numbers of rows from big tables efficiently.

See Charset Narrowing Optimization.

The Charset Narrowing optimization, which speeds up equality comparisons between columns of different character sets.

See Data Sampling: Techniques for Efficiently Finding a Random Row.

Efficient techniques for selecting random rows, avoiding a slow ORDER BY RAND().

See Data Warehousing High Speed Ingestion.

Techniques for high-speed data ingestion when INSERT performance is the bottleneck.

See Data Warehousing Summary Tables.

Creating and maintaining summary tables to speed up data-warehouse queries.

See Data Warehousing Techniques.

Techniques for improving performance of data-warehouse-style tables.

See DISTINCT removal in aggregate functions.

How the optimizer removes redundant DISTINCT from aggregate-function arguments.

See Equality propagation optimization.

The equality propagation optimization, which propagates equality conditions through a query's WHERE clause.

See Filesort with Small LIMIT Optimization.

The filesort optimization that uses a priority queue when sorting with a small LIMIT.

See FORCE INDEX.

Using FORCE INDEX to make the optimizer use an index instead of a table scan.

See Groupwise Max in MariaDB.

Techniques for finding the largest, or top, row within each group.

See GUID/UUID Performance.

Why random GUID/UUID keys hurt index performance, and how to mitigate it.

See hash_join_cardinality optimizer_switch Flag.

The hash_join_cardinality optimizer_switch flag, which affects hash-join cardinality estimates.

See How to Quickly Insert Data Into MariaDB.

Techniques for inserting data into MariaDB as quickly as possible.

See IGNORE INDEX.

Using IGNORE INDEX to tell the optimizer not to consider specific indexes.

See Index Condition Pushdown.

Index Condition Pushdown, which pushes WHERE conditions into the index scan to reduce row reads.

See Index Hints: How to Force Query Plans.

Using index hints to influence the query plan when the optimizer's choice is not ideal.

See index_merge sort_intersection.

The index_merge sort_intersection access method, which merges results from several index scans.

See LIMIT ROWS EXAMINED.

Using LIMIT ROWS EXAMINED to cap how many rows a query examines.

See MariaDB 5.3 Optimizer Debugging.

An obsolete optimizer-debugging facility from an early MariaDB release (historical).

See not_null_range_scan Optimization.

The not_null_range_scan optimization, which builds range scans from inferred NOT NULL conditions.

See optimizer_switch.

The optimizer_switch system variable, used to enable or disable individual optimizations.

See optimizer_adjust_secondary_key_costs.

The optimizer_adjust_secondary_key_costs setting, which tunes cost estimates for secondary keys.

See optimizer_join_limit_pref_ratio Optimization.

An optimization that can pick a join order which shortens ORDER BY … LIMIT queries.

See Optimizing for "Latest News"-style Queries.

Optimizing queries that fetch the latest items on a topic, such as news feeds.

See Pagination Optimization.

Efficient pagination techniques for splitting long lists across pages.

See Pivoting in MariaDB.

Techniques for pivoting row data into a spreadsheet-like columnar layout.

See Query Limits and Timeouts.

The methods MariaDB provides to limit or time out queries.

See reorder_outer_joins.

The reorder_outer_joins optimizer_switch flag, which controls reordering of independent outer joins.

See Rollup Unique User Counts.

Efficiently rolling up unique-user counts without repeatedly reprocessing large logs.

See Rowid Filtering Optimization.

The rowid filtering optimization, which pre-filters rowids to reduce expensive row lookups.

See Sargable DATE and YEAR.

The sargable DATE and YEAR optimization, letting the optimizer use indexes for certain date and year conditions.

See Sargable UPPER.

The sargable UPPER optimization, letting the optimizer use indexes for UPPER() expressions.

See USE INDEX.

Using USE INDEX to limit which indexes the optimizer considers.

See Virtual Column Support in the Optimizer.

Optimizer support for virtual (generated) columns.

See Optimization Strategies.

Discover effective optimization strategies for MariaDB Server queries. This section provides a variety of techniques and approaches to enhance query performance and overall database efficiency.

See Optimizations for Derived Tables.

Optimize derived tables in MariaDB Server queries. This section provides techniques and strategies to improve the performance of subqueries and complex joins, enhancing overall query efficiency.

See Statistics for Optimizing Queries.

Utilize statistics to optimize queries in MariaDB Server. This section explains how the database uses statistical information to generate efficient query execution plans and improve performance.

See Subquery Optimizations.

Optimize subqueries in MariaDB Server for improved performance. This section provides techniques and best practices to ensure your nested queries execute efficiently and enhance overall query speed.

See Table Elimination.

Learn about table elimination for query optimization in MariaDB Server. This section explains how the optimizer removes unnecessary tables from query plans, improving performance.

This page is licensed: CC BY-SA / Gnu FDL

Aborting Statements that Exceed a Certain Time to Execute

Overview

MariaDB 10.1.1 introduced the max_statement_time system variable. When set to a non-zero value, the server attempts to abort any queries taking longer than this time in seconds.

> **Important:**
>
> The abortion is not immediate; the server checks the timer status at specific intervals during execution. Consequently, a query may run slightly longer than the specified time before being detected and stopped.

The default is zero, and no limits are then applied. The aborted query has no effect on any larger transaction or connection contexts. The variable is of type double, thus you can use subsecond timeout. For example you can use value 0.01 for 10 milliseconds timeout.

The value can be set globally or per session, as well as per user or per query (see below).
Replicas are not affected by this variable, however from MariaDB 10.10, there is slave_max_statement_time which serves the same purpose on replicas only.

An associated status variable, max_statement_time_exceeded, stores the number of queries that have exceeded the execution time specified by max_statement_time, and a `MAX_STATEMENT_TIME_EXCEEDED` column was added to the CLIENT_STATISTICS and USER STATISTICS Information Schema tables.

The feature was based upon a patch by Davi Arnaut.

> **Warning:**
>
> **Important Note on Reliability**
>
> `MAX_STATEMENT_TIME` relies on the execution thread checking the "killed" flag, which happens intermittently.
>
> - Long Running Operations: If a query enters a long processing phase where the flag is not checked (e.g., certain storage engine operations or complex calculations), it may continue running significantly past the limit.
> - Resource Protection: Because the abort is not guaranteed to be instantaneous or strictly enforced in all code paths, `MAX_STATEMENT_TIME` should not be relied upon as the sole mechanism for preventing resource exhaustion (such as filling up temporary disk space).

User max_statement_time

max_statement_time can be stored per user with the GRANT … MAX_STATEMENT_TIME syntax.

Per-query max_statement_time

By using max_statement_time in conjunction with SET STATEMENT, it is possible to limit the execution time of individual queries. For example:

```
SET STATEMENT max_statement_time=100 FOR
  SELECT field1 FROM table_name ORDER BY field1;
```

max_statement_time per query
Individual queries can also be limited by adding a `MAX_STATEMENT_TIME` clause to the query. For example:

```
SELECT MAX_STATEMENT_TIME=2 * FROM t1;
```

Limitations

- max_statement_time does not work in embedded servers.
- max_statement_time does not work for COMMIT statements in a Galera cluster (see MDEV-18673 for discussion).
- Check Intervals: The timeout is checked only at specific points during query execution. Queries stuck in operations where the check code path is not hit will not abort until they reach a checkpoint. This can result in query times exceeding the `MAX_STATEMENT_TIME` value.

Differences Between the MariaDB and MySQL Implementations

MySQL 5.7.4 introduced similar functionality, but the MariaDB implementation differs in a number of ways.

- The MySQL version of max_statement_time (`max_execution_time`) is defined in milliseconds, not seconds
- MySQL's implementation can only kill SELECTs, while MariaDB's can kill any queries (excluding stored procedures).
- MariaDB only introduced the max_statement_time_exceeded status variable, while MySQL also introduced a number of other variables which were not seen as necessary in MariaDB.
- The `SELECT MAX_STATEMENT_TIME = N ...` syntax is not valid in MariaDB. In MariaDB one should use `SET STATEMENT MAX_STATEMENT_TIME=N FOR...`.

See Also

- Query limits and timeouts
- lock_wait_timeout variable

This page is licensed: CC BY-SA / Gnu FDL

Big DELETEs

The problem

How to DELETE lots of rows from a large table? Here is an example of purging items older than 30 days:

```
DELETE FROM tbl WHERE
  ts < CURRENT_DATE() - INTERVAL 30 DAY
```

If there are millions of rows in the table, this statement may take minutes, maybe hours.

Any suggestions on how to speed this up?

Why it is a problem

- MyISAM will lock the table during the entire operation, thereby nothing else can be done with the table.
- InnoDB won't lock the table, but it will chew up a lot of resources, leading to sluggishness.
- InnoDB has to write the undo information to its transaction logs; this significantly increases the I/O required.
- Replication, being asynchronous, will effectively be delayed (on Slaves) while the DELETE is running.

InnoDB and undo

To be ready for a crash, a transactional engine such as InnoDB will record what it is doing to a log file. To make that somewhat less costly, the log file is sequentially written. If the log files you have (there are usually 2) fill up because the delete is really big, then the undo information spills into the actual data blocks, leading to even more I/O.

Deleting in chunks avoids some of this excess overhead.

Limited benchmarking of total delete elapsed time show two observations:

- Total delete time approximately doubles above some 'chunk' size (as opposed to below that threshold). I do not have a formula relating the log file size with the threshold cutoff.
- Chunk size below several hundred rows is slower. This is probably because the overhead of starting/ending each chunk dominates the timing.

Solutions

- PARTITION – Requires some careful setup, but is excellent for purging a time-base series.
- DELETE in chunks – Carefully walk through the table N rows at a time.

PARTITION

The idea here is to have a sliding window of partitions. Let's say you need to purge news articles after 30 days. The "partition key" would be the datetime (or timestamp) that is to be used for purging, and the PARTITIONs would be "range". Every night, a cron job would come along and build a new partition for the next day, and drop the oldest partition.

Dropping a partition is essentially instantaneous, much faster than deleting that many rows. However, you must design the table so that the entire partition can be dropped. That is, you cannot have some items living longer than others.

PARTITION tables have a lot of restrictions, some are rather weird. You can either have no UNIQUE (or PRIMARY) key on the table, or every UNIQUE key must include the partition key. In this use case, the partition key is the datetime. It should not be the first part of the PRIMARY KEY (if you have a PRIMARY KEY).

You can PARTITION InnoDB or MyISAM tables.

Since two news articles could have the same timestamp, you cannot assume the partition key is sufficient for uniqueness of the PRIMARY KEY, so you need to find something else to help with that.

Reference implementation for Partition maintenance

MariaDB docs on PARTITION

Deleting in chunks

Although the discussion in this section talks about DELETE, it can be used for any other “chunking”, such as, say, UPDATE, or SELECT plus some complex processing.

(This discussion applies to both MyISAM and InnoDB.)

When deleting in chunks, be sure to avoid doing a table scan. The code below is good at that; it scans no more than 1001 rows in any one query. (The 1000 is tunable.)

Assuming you have news articles that need to be purged, and you have a schema something like

```
CREATE TABLE tbl
      id INT UNSIGNED NOT NULL AUTO_INCREMENT,
      ts TIMESTAMP,
      ...
      PRIMARY KEY(id)
```

Then, this pseudo-code is a good way to delete the rows older than 30 days:

```
@a = 0
   LOOP
      DELETE FROM tbl
         WHERE id BETWEEN @a AND @a+999
           AND ts < DATE_SUB(CURRENT_DATE(), INTERVAL 30 DAY)
      SET @a = @a + 1000
      sleep 1  -- be a nice guy
   UNTIL end of table
```

Notes (Most of these caveats will be covered later):

- It uses the PK instead of the secondary key. This gives much better locality of disk hits, especially for InnoDB.
- You could (should?) do something to avoid walking through recent days but doing nothing. Caution – the code for this could be costly.
- The 1000 should be tweaked so that the DELETE usually takes under, say, one second.
- No INDEX on ts is needed. (This helps INSERTs a little.)
- If your PRIMARY KEY is compound, the code gets messier.
- This code will not work without a numeric PRIMARY or UNIQUE key.
- Read on, we’ll develop messier code to deal with most of these caveats.

If there are big gaps in `id` values (and there will after the first purge), then

```
@a = SELECT MIN(id) FROM tbl
   LOOP
      SELECT @z := id FROM tbl WHERE id >= @a ORDER BY id LIMIT 1000,1
      IF @z IS NULL
         EXIT LOOP  -- last chunk
      DELETE FROM tbl
         WHERE id >= @a
           AND id <  @z
           AND ts < DATE_SUB(CURRENT_DATE(), INTERVAL 30 DAY)
      SET @a = @z
      sleep 1  -- be a nice guy, especially in replication
   ENDLOOP
   # Last chunk:
   DELETE FROM tbl
      WHERE id >= @a
        AND ts < DATE_SUB(CURRENT_DATE(), INTERVAL 30 DAY)
```

That code works whether id is numeric or character, and it mostly works even if id is not UNIQUE. With a non-unique key, the risk is that you could be caught in a loop whenever @z==@a. That can be detected and fixed thus:

```
...
      SELECT @z := id FROM tbl WHERE id >= @a ORDER BY id LIMIT 1000,1
      IF @z == @a
         SELECT @z := id FROM tbl WHERE id > @a ORDER BY id LIMIT 1
   ...
```

The drawback is that there could be more than 1000 items with a single id. In most practical cases, that is unlikely.

If you do not have a primary (or unique) key defined on the table, and you have an INDEX on ts, then consider

```
LOOP
      DELETE FROM tbl
         WHERE ts < DATE_SUB(CURRENT_DATE(), INTERVAL 30 DAY)
         ORDER BY ts   -- to use the index, and to make it deterministic
         LIMIT 1000
   UNTIL no rows deleted
```

This technique is NOT recommended because the LIMIT leads to a warning on replication about it being non-deterministic (discussed below).

InnoDB chunking recommendation

- Have a ‘reasonable’ size for innodb_log_file_size.
- Use AUTOCOMMIT=1 for the session doing the deletions.
- Pick about 1000 rows for the chunk size.
- Adjust the row count down if asynchronous replication (Statement Based) causes too much delay on the Slaves or hogs the table too much.

Iterating through a compound key

To perform the chunked deletes recommended above, you need a way to walk through the PRIMARY KEY. This can be difficult if the PK has more than one column in it.

To efficiently to do compound ‘greater than’:

Assume that you left off at ($g, $s) (and have handled that row):

```
INDEX(Genus, species)
   SELECT/DELETE ...
      WHERE Genus >= '$g' AND ( species  > '$s' OR Genus > '$g' )
      ORDER BY Genus, species
      LIMIT ...
```

Addenda: The above AND/OR works well in older versions of MySQL; this works better in MariaDB and newer versions of MySQL:

```
WHERE ( Genus = '$g' AND species  > '$s' ) OR Genus > '$g' )
```

A caution about using @variables for strings. If, instead of '$g', you use @g, you need to be careful to make sure that @g has the same CHARACTER SET and COLLATION as `Genus`, else there could be a charset/collation conversion on the fly that prevents the use of the INDEX. Using the INDEX is vital for performance. It may require a COLLATE clause on SET NAMES and/or the @g in the SELECT.

Reclaiming the disk space

This is costly. (Switch to the PARTITION solution if practical.)

MyISAM leaves gaps in the table (.MYD file); OPTIMIZE TABLE will reclaim the freed space after a big delete. But it may take a long time and lock the table.

InnoDB is block-structured, organized in a BTree on the PRIMARY KEY. An isolated deleted row leaves a block less full. A lot of deleted rows can lead to coalescing of adjacent blocks. (Blocks are normally 16KB - see innodb_page_size.)

In InnoDB, there is no practical way to reclaim the freed space from ibdata1, other than to reuse the freed blocks eventually.

The only option with innodb_file_per_table = 0 is to dump ALL tables, remove ibdata*, restart, and reload. That is rarely worth the effort and time.

InnoDB, even with innodb_file_per_table = 1, won't give space back to the OS, but at least it is only one table to rebuild with. In this case, something like this should work:

```
CREATE TABLE new LIKE main;
   INSERT INTO new SELECT * FROM main;  -- This could take a long time
   RENAME TABLE main TO old, new TO main;   -- Atomic swap
   DROP TABLE old;   -- Space freed up here
```

You do need enough disk space for both copies. You must not write to the table during the process.

Deleting more than half a table

The following technique can be used for any combination of

- Deleting a large portion of the table more efficiently
- Add PARTITIONing
- Converting to innodb_file_per_table = ON
- Defragmenting

This can be done by chunking, or (if practical) all at once:

```
-- Optional:  SET GLOBAL innodb_file_per_table = ON;
   CREATE TABLE New LIKE Main;
   -- Optional:  ALTER TABLE New ADD PARTITION BY RANGE ...;
   -- Do this INSERT..SELECT all at once, or with chunking:
   INSERT INTO New
      SELECT * FROM Main
         WHERE ...;  -- just the rows you want to keep
   RENAME TABLE main TO Old, New TO Main;
   DROP TABLE Old;   -- Space freed up here
```

Notes:

- You do need enough disk space for both copies.
- You must not write to the table during the process. (Changes to Main may not be reflected in New.)

Non-deterministic replication

Any UPDATE, DELETE, etc with LIMIT that is replicated to slaves (via Statement Based Replication) *may* cause inconsistencies between the Master and Slaves. This is because the actual order of the records discovered for updating/deleting may be different on the slave, thereby leading to a different subset being modified. To be safe, add ORDER BY to such statements. Moreover, be sure the ORDER BY is deterministic – that is, the fields/expressions in the ORDER BY are unique.

An example of an ORDER BY that does not quite work: Assume there are multiple rows for each 'date':

```
DELETE * FROM tbl ORDER BY date LIMIT 111
```

Given that id is the PRIMARY KEY (or UNIQUE), this will be safe:

```
DELETE * FROM tbl ORDER BY date, id LIMIT 111
```

Unfortunately, even with the ORDER BY, MySQL has a deficiency that leads to a bogus warning in mysqld.err. See Spurious "Statement is not safe to log in statement format." warnings

Some of the above code avoids this spurious warning by doing

```
SELECT @z := ... LIMIT 1000,1;  -- not replicated
   DELETE ... BETWEEN @a AND @z;   -- deterministic
```

That pair of statements guarantees no more than 1000 rows are touched, not the whole table.

Replication and KILL

If you KILL a DELETE (or any? query) on the master in the middle of its execution, what will be replicated?

If it is InnoDB, the query should be rolled back. (Exceptions??)

In MyISAM, rows are DELETEd as the statement is executed, and there is no provision for ROLLBACK. Some of the rows will be deleted, some won't. You probably have no clue of how much was deleted. In a single server, simply run the delete again. The delete is put into the binlog, but with error 1317. Since replication is supposed to keep the master and slave in sync, and since it has no clue of how to do that, replication stops and waits for manual intervention. In a HA (High Available) system using replication, this is a minor disaster. Meanwhile, you need to go to each slave(s) and verify that it is stuck for this reason, then do

```
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;
   START SLAVE;
```

Then (presumably) re-executing the DELETE will finish the aborted task.

(That is yet another reason to move all your tables from MyISAM to InnoDB.)

SBR vs RBR; Galera

TBD – “Row Based Replication” may impact this discussion.

Postlog

The tips in this document apply to MySQL, MariaDB, and Percona.

See also

Rick James graciously allowed us to use this article in the documentation.

Rick James’ site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: deletebig

This page is licensed: CC BY-SA / Gnu FDL

Charset Narrowing Optimization

The Charset Narrowing optimization handles equality comparisons like:

```
utf8mb3_key_column=utf8mb4_expression
```

It enables the optimizer to construct `ref` access to `utf8mb3_key_column` based on this equality. The optimization supports comparisons of columns that use `utf8mb3_general_ci` to expressions that use `utf8mb4_general_ci` .

The optimization was introduced in MariaDB 10.6.16, MariaDB 10.10.7, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3 and MariaDB 11.2.2, where it is **OFF** by default. From MariaDB 11.7, it is `ON` by default.

Description

MariaDB supports both the UTF8MB3 and UTF8MB4 character sets. It is possible to construct join queries that compare values in UTF8MB3 to UTF8MB4.

Suppose, we have the table *’users* that uses UTF8MB4:

```
CREATE TABLE users (
  user_name_mb4 VARCHAR(100) COLLATE utf8mb4_general_ci,
  ...
);
```

and table *orders* that uses UTF8MB3:

```
CREATE TABLE orders (
  user_name_mb3 VARCHAR(100) COLLATE utf8mb3_general_ci,
  ...,
  INDEX idx1(user_name_mb3)
);
```

One can join *users* to *orders* on user_name:

```
SELECT * FROM orders, users WHERE orders.user_name_mb3=users.user_name_mb4;
```

Internally the optimizer will handle the equality by converting the UTF8MB3 value into UTF8MB4 and then doing the comparison. One can see the call to `CONVERT` in EXPLAIN FORMAT=JSON output or Optimizer Trace:

```
CONVERT(orders.user_name_mb3 USING utf8mb4) = users.user_name_mb4
```

This produces the expected result but the query optimizer is not able to use the index over `orders.user_name_mb3` to find matches for values of `users.user_name_mb4`.

The EXPLAIN of the above query looks like this:

```
EXPLAIN SELECT * FROM orders, users WHERE orders.user_name_mb3=users.user_name_mb4;
+------+-------------+--------+------+---------------+------+---------+------+-------+------------------
-----------------------------+
| id   | select_type | table  | type | possible_keys | key  | key_len | ref  | rows  | Extra
|
+------+-------------+--------+------+---------------+------+---------+------+-------+------------------
-----------------------------+
|    1 | SIMPLE      | users  | ALL  | NULL          | NULL | NULL    | NULL | 1000  |
|
|    1 | SIMPLE      | orders | ALL  | NULL          | NULL | NULL    | NULL | 10330 | Using where; Using
join buffer (flat, BNL join) |
+------+-------------+--------+------+---------------+------+---------+------+-------+------------------
-----------------------------+
```

The Charset Narrowing optimization enables the optimizer to perform the comparison between UTF8MB3 and UTF8MB4 values by “narrowing” the value in UTF8MB4 to UTF8MB3. The `CONVERT` call is no longer needed, and the optimizer is able to use the equality to construct ref access:

```
SET optimizer_switch='cset_narrowing=ON';

EXPLAIN SELECT * FROM orders, users WHERE orders.user_name_mb3=users.user_name_mb4;
+------+-------------+--------+------+---------------+------+---------+---------------------+------+-----
------------------+
| id   | select_type | table  | type | possible_keys | key  | key_len | ref                 | rows |
Extra                 |
+------+-------------+--------+------+---------------+------+---------+---------------------+------+-----
------------------+
|    1 | SIMPLE      | users  | ALL  | NULL          | NULL | NULL    | NULL                | 1000 |
Using where           |
|    1 | SIMPLE      | orders | ref  | idx1          | idx1 | 303     | users.user_name_mb4 | 1    |
Using index condition |
+------+-------------+--------+------+---------------+------+---------+---------------------+------+-----
------------------+
```

Controlling the Optimization

The optimization is controlled by an optimizer_switch flag. Specify:

```
SET optimizer_switch='cset_narrowing=ON';
```

to enable the optimization.

References

- MDEV-32113: utf8mb3_key_col=utf8mb4_value cannot be used for ref access
- Blog post: Making “tbl.utf8mb3_key_column=utf8mb4_expr” sargable

This page is licensed: CC BY-SA / Gnu FDL

Data Sampling: Techniques for Efficiently Finding a Random Row

Fetching random rows from a table (beyond ORDER BY RAND())

The problem

One would like to do “SELECT … ORDER BY RAND() LIMIT 10” to get 10 rows at random. But this is slow. The optimizer does

- Fetch all the rows – this is costly
- Append RAND() to the rows
- Sort the rows – also costly
- Pick the first 10.

All the algorithms given below are “fast”, but most introduce flaws:

- Bias – some rows are more like to be fetched than others.
- Repetitions – If two random sets contain the same row, they are likely to contain other dups.
- Sometimes failing to fetch the desired number of rows.

“Fast” means avoiding reading all the rows. There are many techniques that require a full table scan, or at least an index scan. They are not acceptable for this list. There is even a technique that averages half a scan; it is relegated to a footnote.

Metrics

Here’s a way to measure performance without having a big table.

```
FLUSH STATUS;
    SELECT ...;
    SHOW SESSION STATUS LIKE 'Handler%';
```

If some of the “Handler” numbers look like the number of rows in the table, then there was a table scan.

None of the queries presented here need a full table (or index) scan. Each has a time proportional to the number of rows returned.

Virtually all published algorithms involve a table scan. The previously published version of this blog had, embarrassingly, several algorithms that had table scans.

Sometimes the scan can be avoided via a subquery. For example, the first of these will do a table scan; the second will not.

```
SELECT *  FROM RandTest AS a
  WHERE id = FLOOR(@min + (@max - @min + 1) * RAND());  -- BAD: table scan
SELECT *
 FROM RandTest AS a
 JOIN (
   SELECT FLOOR(@min + (@max - @min + 1) * RAND()) AS id -- Good; single eval.
      ) b  USING (id);
```

Case: Consecutive AUTO_INCREMENT without gaps, 1 row returned

- Requirement: AUTO_INCREMENT id
- Requirement: No gaps in id

```
SELECT r.*
      FROM (
          SELECT FLOOR(mm.min_id + (mm.max_id - mm.min_id + 1) * RAND()) AS id
              FROM (
                  SELECT MIN(id) AS min_id,
                         MAX(id) AS max_id
                      FROM RandTest
                   ) AS mm
           ) AS init
      JOIN  RandTest AS r  ON r.id = init.id;
```

(Of course, you might be able to simplify this. For example, min_id is likely to be 1. Or precalculate limits into @min and @max.)

Case: Consecutive AUTO_INCREMENT without gaps, 10 rows

- Requirement: AUTO_INCREMENT id
- Requirement: No gaps in id
- Flaw: Sometimes delivers fewer than 10 rows

```
-- First select is one-time:
  SELECT @min := MIN(id),
         @max := MAX(id)
      FROM RandTest;
  SELECT DISTINCT *
      FROM RandTest AS a
      JOIN (
          SELECT FLOOR(@min + (@max - @min + 1) * RAND()) AS id
              FROM RandTest
              LIMIT 11    -- more than 10 (to compensate for dups)
           ) b  USING (id)
      LIMIT 10;           -- the desired number of rows
```

The FLOOR expression could lead to duplicates, hence the inflated inner LIMIT. There could (rarely) be so many duplicates that the inflated LIMIT leads to fewer than the desired 10 different rows. One approach to that Flaw is to rerun the query if it delivers too few rows.

A variant:

```
SELECT r.*
      FROM (
          SELECT FLOOR(mm.min_id + (mm.max_id - mm.min_id + 1) * RAND()) AS id
              FROM (
                  SELECT MIN(id) AS min_id,
                         MAX(id) AS max_id
                      FROM RandTest
                   ) AS mm
              JOIN ( SELECT id dummy FROM RandTest LIMIT 11 ) z
           ) AS init
      JOIN  RandTest AS r  ON r.id = init.id
      LIMIT 10;
```

Again, ugly but fast, regardless of table size.

Case: AUTO_INCREMENT with gaps, 1 or more rows returned

- Requirement: AUTO_INCREMENT, possibly with gaps due to DELETEs, etc
- Flaw: Only semi-random (rows do not have an equal chance of being picked), but it does partially compensate for the gaps
- Flaw: The first and last few rows of the table are less likely to be delivered.

This gets 50 “consecutive” ids (possibly with gaps), then delivers a random 10 of them.

```
-- First select is one-time:
SELECT @min := MIN(id),
       @max := MAX(id)
    FROM RandTest;
SELECT a.*
    FROM RandTest a
    JOIN ( SELECT id FROM
            ( SELECT id
                FROM ( SELECT @min + (@max - @min + 1 - 50) * RAND()
                  AS start FROM DUAL ) AS init
                JOIN RandTest y
                WHERE    y.id > init.start
                ORDER BY y.id
                LIMIT 50         -- Inflated to deal with gaps
            ) z ORDER BY RAND()
           LIMIT 10              -- number of rows desired (change to 1 if looking for a single row)
         ) r ON a.id = r.id;
```

Yes, it is complex, but yes, it is fast, regardless of the table size.

Case: Extra FLOAT column for randomizing

(Unfinished: need to check these.)

Assuming `rnd` is a FLOAT (or DOUBLE) populated with RAND() and INDEXed:

- Requirement: extra, indexed, FLOAT column
- Flaw: Fetches 10 adjacent rows (according to `rnd`), hence not good randomness
- Flaw: Near ‘end’ of table, can’t find 10 rows.

```
SELECT r.*
      FROM ( SELECT RAND() AS start FROM DUAL ) init
      JOIN RandTest r
      WHERE r.rnd >= init.start
      ORDER BY r.rnd
      LIMIT 10;
```

- These two variants attempt to resolve the end-of-table flaw:

```
SELECT r.*
      FROM ( SELECT RAND() * ( SELECT rnd
                        FROM RandTest
                        ORDER BY rnd DESC
                        LIMIT 10,1 ) AS start
           ) AS init
      JOIN RandTest r
      WHERE r.rnd > init.start
      ORDER BY r.rnd
      LIMIT 10;

  SELECT @start := RAND(),
         @cutoff := CAST(1.1 * 10 + 5 AS DECIMAL(20,8)) / TABLE_ROWS
      FROM information_schema.TABLES
      WHERE TABLE_SCHEMA = 'dbname'
        AND TABLE_NAME = 'RandTest'; -- 0.0030
  SELECT d.*
      FROM (
          SELECT a.id
              FROM RandTest a
              WHERE rnd BETWEEN @start AND @start + @cutoff
           ) sample
      JOIN RandTest d USING (id)
      ORDER BY rand()
      LIMIT 10;
```

Case: UUID or MD5 column

- Requirement: UUID/GUID/MD5/SHA1 column exists and is indexed.
- Similar code/benefits/flaws to AUTO_INCREMENT with gaps.
- Needs 7 random HEX digits:

```
RIGHT( HEX( (1<<24) * (1+RAND()) ), 6)
```

can be used as a `start` for adapting a gapped AUTO_INCREMENT case. If the field is BINARY instead of hex, then

```
UNHEX(RIGHT( HEX( (1<<24) * (1+RAND()) ), 6))
```

See also

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: random

This page is licensed: CC BY-SA / Gnu FDL

Data Warehousing High Speed Ingestion

The problem

You are ingesting lots of data. Performance is bottlenecked in the INSERT area.

This will be couched in terms of Data Warehousing, with a huge `Fact` table and Summary (aggregation) tables.

Overview of solution

- Have a separate staging table.
- Inserts go into `Staging` .
- Normalization and Summarization reads Staging, not Fact.
- After normalizing, the data is copied from Staging to Fact.

`Staging` is one (or more) tables in which the data lives only long enough to be handed off to Normalization, Summary, and the Fact tables.

Since we are probably talking about a billion-row table, shrinking the width of the Fact table by normalizing (as mentioned here). Changing an INT to a MEDIUMINT will save a GB. Replacing a string by an id (normalizing) saves many GB. This helps disk space and cacheability, hence speed.

Injection speed

Some variations:

- Big dump of data once an hour, versus continual stream of records.
- The input stream could be single-threaded or multi-threaded.
- You might have 3rd party software tying your hands.

Generally the fastest injection rate can be achieved by "staging" the INSERTs in some way, then batch processing the staged records. This blog discusses various techniques for staging and batch processing.

Normalization

Let's say your Input has a VARCHAR `host_name` column, but you need to turn that into a smaller MEDIUMINT `host_id` in the Fact table. The "Normalization" table, as I call it, looks something like

```
CREATE TABLE Hosts (
    host_id  MEDIUMINT UNSIGNED  NOT NULL AUTO_INCREMENT,
    host_name VARCHAR(99) NOT NULL,
    PRIMARY KEY (host_id),      -- for mapping one direction
    INDEX(host_name, host_id)   -- for mapping the other direction
) ENGINE=InnoDB;                -- InnoDB works best for Many:Many mapping table
```

Here's how you can use `Staging` as an efficient way achieve the swap from name to id.

Staging has two fields (for this normalization example):

```
host_name VARCHAR(99) NOT NULL,     -- Comes from the insertion process
    host_id  MEDIUMINT UNSIGNED  NULL,  -- NULL to start with; see code below
```

Meawhile, the Fact table has:

```
host_id  MEDIUMINT UNSIGNED NOT NULL,
```

SQL #1 (of 2):

```
# This should not be in the main transaction, and it should be done with autocommit = ON
    # In fact, it could lead to strange errors if this were part
    #    of the main transaction and it ROLLBACKed.
    INSERT IGNORE INTO Hosts (host_name)
        SELECT DISTINCT s.host_name
            FROM Staging AS s
            LEFT JOIN Hosts AS n  ON n.host_name = s.host_name
            WHERE n.host_id IS NULL;
```

By isolating this as its own transaction, we get it finished in a hurry, thereby minimizing blocking. By saying IGNORE, we don't care if other threads are 'simultaneously' inserting the same host_names.

There is a subtle reason for the LEFT JOIN. If, instead, it were INSERT IGNORE..SELECT DISTINCT, then the INSERT would preallocate auto_increment ids for as many rows as the SELECT provides. This is very likely to "burn" a lot of ids, thereby leading to overflowing MEDIUMINT unnecessarily. The LEFT JOIN leads to finding just the new ids that are needed (except for the rare possibility of a 'simultaneous' insert by another thread). More rationale: Mapping table

SQL #2:

```
# Also not in the main transaction, and it should be with autocommit = ON
    # This multi-table UPDATE sets the ids in Staging:
    UPDATE   Hosts AS n
        JOIN Staging AS s  ON s.host_name = n host_name
        SET s.host_id = n.host_id
```

This gets the IDs, whether already existing, set by another thread, or set by SQL #1.

If the size of `Staging` changes depending on the busy versus idle times of the day, this pair of SQL statements has another comforting feature. The more rows in `Staging`, the more efficient the SQL runs, thereby helping compensate for the "busy" times.

The companion Data Warehouse article folds SQL #2 into the INSERT INTO Fact. But you may need host_id for further normalization steps and/or Summarization steps, so this explicit UPDATE shown here is often better.

Flip-flop staging

The simple way to stage is to ingest for a while, then batch-process what is in `Staging`. But that leads to new records piling up waiting to be staged. To avoid that issue, have 2 processes:

- one process (or set of processes) for INSERTing into `Staging`;
- one process (or set of processes) to do the batch processing (normalization, summarization).

To keep the processes from stepping on each other, we have a pair of staging tables:

- `Staging` is being INSERTed into;
- `StageProcess` is one being processed for normalization, summarization, and moving to the Fact table.
  A separate process does the processing, then swaps the tables:

```
DROP   TABLE StageProcess;
    CREATE TABLE StageProcess LIKE Staging;
    RENAME TABLE Staging TO tmp, StageProcess TO Staging, tmp TO StageProcess;
```

This may not seem like the shortest way to do it, but has these features:

- The DROP + CREATE might be faster than TRUNCATE, which is the desired effect.
- The RENAME is atomic, so the INSERT process(es) never find that `Staging` is missing.

A variant on the 2-table flip-flop is to have a separate `Staging` table for each Insertion process. The Processing process would run around to each Staging in turn.

A variant on that would be to have a separate processing process for each Insertion process.

The choice depends on which is faster (insertion or processing). There are tradeoffs; a single processing thread avoids some locks, but lacks some parallelism.

Engine choice

`Fact` table – InnoDB, if for no other reason than that a system crash would not need a REPAIR TABLE. (REPAIRing a billion-row MyISAM table can take hours or days.)

Normalization tables – InnoDB, primarily because it can be done efficiently with 2 indexes, whereas, MyISAM would need 4 to achieve the same efficiency.

`Staging` – Lots of options here.

- If you have multiple Inserters and a single Staging table, InnoDB is desirable due to row-level, not table-level, locking.
- MEMORY may be the fastest and it avoids I/O. This is good for a single staging table.
- For multiple Inserters, a separate Staging table for each Inserter is desired.
- For multiple Inserters into a single Staging table, InnoDB may be faster. (MEMORY does table-level locking.)
- With one non-InnoDB Staging table per Inserter, using an explicit LOCK TABLE avoids repeated implicit locks on each INSERT.
- But, if you are doing LOCK TABLE and the Processing thread is separate, an UNLOCK is necessary periodically to let the RENAME grab the table.
- “Batch INSERTs” (100-1000 rows per SQL) eliminates much of the issues of the above bullet items.

Confused? Lost? There are enough variations in applications that make it impractical to predict what is best. Or, simply good enough. Your ingestion rate may be low enough that you don’t hit the brick walls that I am helping you avoid.

Should you do “CREATE TEMPORARY TABLE”? Probably not. Consider `Staging` as part of the data flow, not to be DROPped.

Summarization

This is mostly covered here: Summary Tables
Summarize from the Staging table instead of the Fact table.

Replication Issues

Row Based Replication (RBR) is probably the best option.

The following allows you to keep more of the Ingestion process in the Master, thereby not bogging down the Slave(s) with writes to the Staging table.

- RBR
- `Staging` is in a separate database
- That database is not replicated (binlog-ignore-db on Master)
- In the Processing steps, USE that database, reach into the main db via syntax like “MainDb.Hosts”. (Otherwise, the binlog-ignore-db does the wrong thing.)

That way

- Writes to `Staging` are not replicated.
- Normalization sends only the few updates to the normalization tables.
- Summarization sends only the updates to the summary tables.
- Flip-flop does not replicate the DROP, CREATE or RENAME.

Sharding

You could possibly spread the data you are trying ingest across multiple machines in a predictable way (sharding on hash, range, etc). Running “reports” on a sharded Fact table is a challenge unto itself. On the other hand, Summary Tables rarely get too big to manage on a single machine.

For now, Sharding is beyond the scope of this blog.

Push me vs pull me

I have implicitly assumed the data is being pushed into the database. If, instead, you are “pulling” data from some source(s), then there are some different considerations.

Case 1: An hourly upload; run via cron

1. Grab the upload, parse it
2. Put it into the Staging table
3. Normalize – each SQL in its own transaction (autocommit)
4. BEGIN
5. Summarize
6. Copy from Staging to Fact.
7. COMMIT

If you need parallelism in Summarization, you will have to sacrifice the transactional integrity of steps 4-7.

Caution: If these steps add up to more than an hour, you are in deep dodo.

Case 2: You are polling for the data

It is probably reasonable to have multiple processes doing this, so it will be detailed about locking.

1. Create a Staging table for this polling processor.
   Loop:

2. With some locked mechanism, decide which ‘thing’ to poll.
3. Poll for the data, pull it in, parse it. (Potentially polling and parsing are significantly costly)
4. Put it into the process-specific Staging table
5. Normalize – each SQL in its own transaction (autocommit)
6. BEGIN
7. Summarize
8. Copy from Staging to Fact.
9. COMMIT
10. Declare that you are finished with this ‘thing’ (see step 1)
    EndLoop.

iblog_file_size should be larger than the change in the STATUS “Innodb_os_log_written” across the BEGIN… COMMIT transaction (for either Case).

See also

- companion Data Warehouse blog
- companion Summary Table blog
- a forum thread that prodded me into writing this blog
- StackExchange discussion

Rick James graciously allowed us to use this article in the documentation.

Rick James’ site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: staging_table

This page is licensed: CC BY-SA / Gnu FDL

Data Warehousing Summary Tables

Preface

This document discusses the creation and maintenance of “Summary Tables”. It is a companion to the document on Data Warehousing Techniques.

The basic terminology (“Fact Table”, “Normalization”, etc) is covered in that document.

Summary tables for data warehouse “reports”

Summary tables are a performance necessity for large tables. MariaDB and MySQL do not provide any automated way to create such, so I am providing techniques here.

(Other vendors provide something similar with “materialized views”.)

When you have millions or billions of rows, it takes a long time to summarize the data to present counts, totals, averages, etc, in a size that is readily digestible by humans. By computing and saving subtotals as the data comes in, one can make “reports” run much faster. (I have seen 10x to 1000x speedups.) The subtotals go into a “summary table”. This document guides you on efficiency in both creating and using such tables.

General structure of a summary table

A summary table includes two sets of columns:

- Main KEY: date + some dimension(s)
- Subtotals: COUNT(*), SUM(…), …; but not AVG()

The “date” might be a DATE (a 3-byte native datatype), or an hour, or some other time interval. A 3-byte MEDIUMINT UNSIGNED ‘hour’ can be derived from a DATETIME or TIMESTAMP via

```
FLOOR(UNIX_TIMESTAMP(dt) / 3600)
   FROM_UNIXTIME(hour * 3600)
```

The “dimensions” (a DW term) are some of the columns of the “Fact” table. Examples: Country, Make, Product, Category, Host Non-dimension examples: Sales, Quantity, TimeSpent

There would be one or more indexes, usually starting with some dimensions and ending with the date field. By ending with the date, one can efficiently get a range of days/weeks/etc. even when each row summarizes only one day.

There will typically be a “few” summary tables. Often one summary table can serve multiple purposes sufficiently efficiently.

As a rule of thumb, a summary table will have one-tenth the number of rows as the Fact table. (This number is very loose.)

Example

Let’s talk about a large chain of car dealerships.
The Fact table has all the sales with columns such as datetime, salesman_id, city, price, customer_id, make, model, model_year.
One Summary table might focus on sales:

```
PRIMARY KEY(city, datetime),
   Aggregations: ct, sum_price

   # Core of INSERT..SELECT:
   DATE(datetime) AS DATE, city, COUNT(*) AS ct, SUM(price) AS sum_price

   # Reporting average price FOR last month, broken down BY city:
   SELECT city,
          SUM(sum_price) / SUM(ct) AS 'AveragePrice'
      FROM SalesSummary
      WHERE datetime BETWEEN ...
      GROUP BY city;

   # Monthly sales, nationwide, FROM same summary TABLE:
   SELECT MONTH(datetime) AS 'Month',
          SUM(ct)         AS 'TotalSalesCount'
          SUM(sum_price)  AS 'TotalDollars'
      FROM SalesSummary
      WHERE datetime BETWEEN ...
      GROUP BY MONTH(datetime);
   # This might benefit FROM a secondary INDEX(datetime)
```

When to augment the summary table(s)?

“Augment” in this section means to add new rows into the summary table or increment the counts in existing rows.

Plan A: “While inserting” rows into the Fact table, augment the summary table(s). This is simple, and workable for a smaller DW database (under 10 Fact table rows per second). For larger DW databases, Plan A likely to be too costly to be practical.

Plan B: “Periodically”, via cron or an EVENT.

Plan C: “As needed”. That is, when someone asks for a report, the code first updates the summary tables that will be needed.

Plan D: “Hybrid” of B and C. C, by itself, can led to long delays for the report. By also doing B, those delays can be kept low.

Plan E: (This is not advised.) “Rebuild” the entire summary table from the entire Fact table. The cost of this is prohibitive for large tables. However, Plan E may be needed when you decide to change the columns of a Summary Table, or discover a flaw in the computations.
To lessen the impact of an entire build, adapt the chunking techniques in Deleting in chunks .

Plan F: “Staging table”. This is primarily for very high speed ingestion. It is mentioned briefly in this blog, and discussed more thoroughly in the companion blog: High Speed Ingestion

Summarizing while Inserting (one row at a time)

```
INSERT INTO Fact ...;
    INSERT INTO Summary (..., ct, foo, ...) VALUES (..., 1, foo, ...)
        ON DUPLICATE KEY UPDATE ct = ct+1, sum_foo = sum_foo + VALUES(foo), ...;
```

IODKU (Insert On Duplicate Key Update) will update an existing row or create a new row. It knows which to do based on the Summary table’s PRIMARY KEY.

Caution: This approach is costly, and will not scale to an ingestion rate of over, say, 10 rows per second (Or maybe 50/second on SSDs). More discussion later.

Summarizing periodically vs as-needed

If your reports need to be up-to-the-second, you need “as needed” or “hybrid”. If your reports have less urgency (eg, weekly reports that don’t include ‘today’), then “periodically” might be best.

For a daily summaries, augmenting the summary tables could be done right after midnight. But, beware of data coming “late”.

For both “periodic” and “as needed”, you need a definitive way of keeping track of where you “left off”.

Case 1: You insert into the Fact table first and it has an AUTO_INCREMENT id: Grab MAX(id) as the upper bound for summarizing and put it either into some other secure place (an extra table), or put it into the row(s) in the Summary table as you insert them. (Caveat: AUTO_INCREMENT ids do not work well in multi-master, including Galera, setups.)

Case 2: If you are using a ‘staging’ table, there is no issue. (More on staging tables later.)

Summarizing while batch inserting

This applies to multi-row (batch) INSERT and LOAD DATA.

The Fact table needs an AUTO_INCREMENT id, and you need to be able to find the exact range of ids inserted. (This may be impractical in any multi-master setup.)

Then perform bulk summarization using

```
FROM Fact
   WHERE id BETWEEN min_id AND max_id
```

Summarizing when using a staging table

Load the data (via INSERTs or [LOAD DATA) en masse into a “staging table”. Then perform batch summarization from the Staging table. And batch copy from the Staging table to the Fact table. Note that the Staging table is handy for batching “normalization” during ingestion.
See also [data-warehousing-high-speed-ingestion|High Speed Ingestion

Summary table: PK or not?

Let's say your summary table has a DATE, `dy`, and a dimension, `foo`. The question is: Should (foo, dy) be the PRIMARY KEY? Or a non-UNIQUE index?

Case 1: PRIMARY KEY (foo, dy) and summarization is in lock step with, say, changes in `dy`.

This case is clean and simple – until you get to endcases. How will you handle the case of data arriving 'late'? Maybe you will need to recalculate some chunks of data? If so, how?

Case 2: (foo, dy) is a non-UNIQUE INDEX.

This case is clean and simple, but it can clutter the summary table because multiple rows can occur for a given (foo, dy) pair. The report will always have to SUM() up values because it cannot assume there is only one row, even when it is reporting on a single `foo` for a single `dy`. This forced-SUM is not really bad – you should do it anyway; that way all your reports are written with one pattern.

Case 3: PRIMARY KEY (foo, dy) and summarization can happen anytime.

Since you should be using InnoDB, there needs to be an explicit PRIMARY KEY.
One approach when you do not have a 'natural' PK is this:

```
id INT UNSIGNED AUTO_INCREMENT NOT NULL,
   ...
   PRIMARY KEY(foo, dy, id),  -- `id` added to make unique
   INDEX(id)                  -- sufficient to keep AUTO_INCREMENT happy
```

This case pushes the complexity onto the summarization by doing a IODKU.

Advice? Avoid Case 1; too messy. Case 2 is ok if the extra rows are not too common. Case 3 may be the closest to "once size fits all".

Averages, etc.

When summarizing, include `COUNT(*) AS ct and SUM(foo) AS sum_foo`. When reporting, the "average" is computed as SUM(sum_foo) / SUM(ct). That is mathematically correct.

Exception… Let's say you are looking at weather temperatures. And you monitoring station gets the temp periodically, but unreliably. That is, the number of readings for a day varies. Further, you decide that the easiest way to compensate for the inconsistency is to do something like: Compute the avg temp for each day, then average those across the month (or other timeframe).

Formula for Standard Deviation:

```
SQRT( SUM(sum_foo2)/SUM(ct) - POWER(SUM(sum_foo)/SUM(ct), 2) )
```

Where sum_foo2 is SUM(foo * foo) from the summary table. sum_foo and sum_foo2 should be FLOAT. FLOAT gives you about 7 significant digits, which is more than enough for things like average and standard deviation. FLOAT occupies 4 bytes. DOUBLE would give you more precision, but occupies 8 bytes. INT and BIGINT are not practical because they may lead to complaints about overflow.

Staging table

The idea here is to first load a set of Fact records into a "staging table", with the following characteristics (at least):

- The table is repeatedly populated and truncated
- Inserts could be individual or batched, and from one or many clients
- SELECTs will be table scans, so no indexes needed
- Inserting will be fast (InnoDB may be the fastest)

- Normalization can be done in bulk, hence efficiently
- Copying to the Fact table will be fast
- Summarization can be done in bulk, hence efficiently
- “Bursty” ingestion is smoothed by this process
- Flip-flop a pair of Staging tables

If you have bulk inserts (Batch INSERT or LOAD DATA) then consider doing the normalization and summarization immediately after each bulk insert.

More details: High Speed Ingestion

Extreme design

Here is a more complex way to design the system, with the goal of even more scaling.

- Use master-slave setup: ingest into master; report from slave(s).
- Feed ingestion through a staging table (as described above)
- Single-source of data: ENGINE=MEMORY; multiple sources: InnoDB
- binlog_format = ROW
- Use binlog_ignore_db to avoid replicating staging – necessitating putting it in a separate database.
- Do the summarization from Staging
- Load Fact via INSERT INTO Fact … SELECT FROM Staging …

Explanation and comments:

- ROW + ignore_db avoids replicating Staging, yet replicates the INSERTs based on it. Hence, it lightens the write load on the Slaves
- If using MEMORY, remember that it is volatile – recover from a crash by starting the ingestion over.
- To aid with debugging, TRUNCATE or re-CREATE Staging at the start of the next cycle.
- Staging needs no indexes – all operations read all rows from it.

Stats on the system that this ‘extreme design’ came from: Fact Table: 450GB, 100M rows/day (batch of 4M/hour), 60 day retention (60+24 partitions), 75B/row, 7 summary tables, under 10 minutes to ingest and summarize the hourly batch. The INSERT..SELECT handled over 20K rows/sec going into the Fact table.
Spinning drives (not SSD) with RAID-10.

“Left Off”

One technique involves summarizing some of the data, then recording where you “left off”, so that next time, you can start there. There are some subtle issues with “left off” that you should be cautious of.

If you use a DATETIME or TIMESTAMP as “left off”, beware of multiple rows with the same value.

- Plan A: Use a compound “left off” (eg, TIMESTAMP + ID). This is messy, error prone, etc.
- Plan B: WHERE ts >= $left_off AND ts < $max_ts – avoids dups, but has other problems (below)
- Separate threads could COMMIT TIMESTAMPs out of order.

If you use an AUTO_INCREMENT as “left off” beware of:

- In InnoDB, separate threads could COMMIT ids in the ‘wrong’ order.
- Multi-master (including Galera and InnoDB Cluster), could lead to ordering issues.

So, nothing works, at least not in a multi-threaded environment?

If you can live with an occasional hiccup (skipped record), then maybe this is ‘not a problem’ for you.

The “Flip-Flop Staging” is a safe alternative, optionally combined with the “Extreme Design”.

Flip-flop staging

If you have many threads simultaneously INSERTing into one staging table, then here is an efficient way to handle a large load: Have a process that flips that staging table with another, identical, staging table, and performs bulk normalization, Fact insertion, and bulk summarization.

The flipping step uses a fast, atomic, RENAME.

Here is a sketch of the code:

```
# Prep FOR flip:
    CREATE TABLE new LIKE Staging;

    # Swap (flip) Staging tables:
    RENAME TABLE Staging TO old, new TO Staging;

    # Normalize new `foo`s:
    # (autocommit = 1)
    INSERT IGNORE INTO Foos SELECT fpp FROM old LEFT JOIN Foos ...

    # Prep FOR possible deadlocks, etc
    WHILE...
    START TRANSACTION;

    # ADD TO Fact:
    INSERT INTO Fact ... FROM old JOIN Foos ...

    # Summarize:
    INSERT INTO Summary ... FROM old ... GROUP BY ...

    COMMIT;
    end-WHILE

    # Cleanup:
    DROP TABLE old;
```

Meanwhile, ingestion can continue writing to `Staging`. The ingestion INSERTs will conflict with the RENAME, but will be resolved gracefully and silently and quickly.

How fast should you flip-flop? Probably the best scheme is to

- Have a job that flip-flops in a tight loop (no delay, or a small delay, between iterations), and
- Have a CRON that serves only as a “keep-alive” to restart the job if it dies.

If Staging is ‘big’, an iteration will take longer, but run more efficiently. Hence, it is self-regulating.

In a Galera (or InnoDB Cluster?) environment, each node could be receiving input. If can afford to loose a few rows, have `Staging` be a non-replicated MEMORY table. Otherwise, have one `Staging` per node and be InnoDB; it will be more secure, but slower and not without problems. In particular, if a node dies completely, you somehow need to process its `Staging` table.

Multiple summary tables

- Look at the reports you will need.
- Design a summary table for each.
- Then look at the summary tables – you are likely to find some similarities.
- Merge similar ones.

To look at what a report needs, look at the WHERE clause that would provide the data. Some examples, assuming data about service records for automobiles: The GROUP BY to gives a clue of what the report might be about.

1. WHERE make = ? AND model_year = ? GROUP BY service_date, service_type

2. WHERE make = ? AND model = ? GROUP BY service_date, service_type
3. WHERE service_type = ? GROUP BY make, model, service_date
4. WHERE service_date between ? and ? GROUP BY make, model, model_year

You need to allow for ‘ad hoc’ queries? Well, look at all the ad hoc queries – they all have a date range, plus nail down one or two other things. (I rarely see something as ugly as ‘%CL%’ for nailing down another dimension.) So, start by thinking of date plus one or two other dimensions as the ‘key’ into a new summary table. Then comes the question of what data might be desired – counts, sums, etc. Eventually you have a small set of summary tables. Then build a front end to allow them to pick only from those possibilities. It should encourage use of the existing summary tables, not be truly ‘open ended’.

Later, another ‘requirement’ may surface. So, build another summary table. Of course, it may take a day to initially populate it.

Games on summary tables

Does one ever need to summarize a summary table? Yes, but only in extreme situations. Usually a ‘weekly’ report can be derived from a ‘daily’ summary table; building a separate weekly summary table not being worth the effort.

Would one ever PARTITION a Summary Table? Yes, in extreme situations, such as the table being large, and

- Need to purge old data (unlikely), or
- Recent’ data is usually requested, and the index(es) fail to prevent table scans (rare). (“Partition pruning” to the rescue.)

See also

Rick James graciously allowed us to use this article in the documentation.

Rick James’ site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: summarytables

Examples

- 1876
- 1766831
- 1766831

This page is licensed: CC BY-SA / Gnu FDL

Data Warehousing Techniques

Preface

This document discusses techniques for improving performance for data-warehouse-like tables in MariaDB and MySQL.

- How to load large tables.
- Normalization.
- Developing ‘summary tables’ to make ‘reports’ efficient.
- Purging old data.

Details on summary tables is covered in the companion document: Summary Tables.

Terminology

This list mirrors “Data Warehouse” terminology.

- Fact table – The one huge table with the ‘raw’ data.
- Summary table – a redundant table of summarized data that could – use for efficiency
- Dimension – columns that identify aspects of the dataset (region, country, user, SKU, zipcode, …)
- Normalization table (dimension table) – mapping between strings an ids; used for space and speed.
- Normalization – The process of building the mapping (‘New York City’ <-> 123)

Fact table

Techniques that should be applied to the huge Fact table.

- id INT/BIGINT UNSIGNED NOT NULL AUTO_INCREMENT
- PRIMARY KEY (id)
- Probably no other INDEXes
- Accessed only via id
- All VARCHARs are “normalized”; ids are stored instead
- ENGINE = InnoDB
- All “reports” use summary tables, not the Fact table
- Summary tables may be populated from ranges of id (other techniques described below)

There are exceptions where the Fact table must be accessed to retrieve multiple rows. However, you should minimize the number of INDEXes on the table because they are likely to be costly on INSERT.

Why keep the Fact table?

Once you have built the Summary table(s), there is not much need for the Fact table. One option that you should seriously consider is to not have a Fact table. Or, at least, you could purge old data from it sooner than you purge the Summary tables. Maybe even keep the Summary tables forever.

Case 1: You need to find the raw data involved in some event. But how will you find those row(s)? This is where a secondary index may be required.

If a secondary index is bigger than can be cached in RAM, and if the column(s) being indexed is random, then each row inserted may cause a disk hit to update the index. This limits insert speed to something like 100 rows per second (on ordinary disks). Multiple random indexes slow down insertion further. RAID striping and/or SSDs speed up insertion. Write caching helps, but only for bursts.

Case 2: You need some event, but you did not plan ahead with the optimal INDEX. Well, if the data is PARTITIONed on date, so even if you have a clue of when the event occurred, “partition pruning” will keep the query from being too terribly slow.

Case 3: Over time, the application is likely to need new ‘reports’, which may lead to a new Summary table. At this point, it would be handy to scan through the old data to fill up the new table.

Case 4: You find a flaw in the summarization, and need to rebuild an existing Summary table.

Cases 3 and 4 both need the “raw” data. But they don’t necessarily need the data sitting in a database table. It could be in the pre-database format (such as log files). So, consider not building the Fact table, but simply keep the raw data, compressed, on some file system.

Batching the load of the Fact table

When talking about billions of rows in the Fact table, it is essentially mandatory that you “batch” the inserts. There are two main ways:

- INSERT INTO Fact (.,.,.) VALUES (.,.,.), (.,.,.), …; – “Batch insert”
- LOAD DATA …;

A third way is to INSERT or LOAD into a Staging table, then

- INSERT INTO Fact SELECT * FROM Staging;
  This INSERT..SELECT allows you to do other things, such as normalization. More later.

Batched INSERT Statement

Chunk size should usually be 100-1000 rows.

- 100-1000 an insert will run 10 times as fast as single-row inserts.
- Beyond 100, you may be interfering replication and SELECTs.
- Beyond 1000, you are into diminishing returns – virtually no further performance gains.
- Don't go past, say, 1MB for the constructed INSERT statement. This deals with packet sizes, etc. (1MB is unlikely to be hit for a Fact table.)
  Decide whether your application should lean toward the 100 or the 1000.

If your data is coming in continually, and you are adding a batching layer, let's do some math. Compute your ingestion rate – R rows per second.

- If R < 10 (= 1M/day = 300M/year) – single-row INSERTs would probably work fine (that is, batching is optional)
- If R < 100 (3B records per year) – secondary indexes on Fact table may be ok
- If R < 1000 (100M records/day) – avoid secondary indexes on Fact table.
- If R > 1000 – Batching may not work.
  Decide how long (S seconds) you can stall loading the data in order to collect a batch of rows.
- If S < 0.1s – May not be able to keep up

If batching seems viable, then design the batching layer to gather for S seconds or 100-1000 rows, whichever comes first.

(Note: Similar math applies to rapid UPDATEs of a table.)

Normalization (Dimension) table

Normalization is important in Data Warehouse applications because it significantly cuts down on the disk footprint and improves performance. There are other reasons for normalizing, but space is the important one for DW.

Here is a typical pattern for a Dimension table:

```
CREATE TABLE Emails (
        email_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,  -- don't make bigger than needed
        email VARCHAR(...) NOT NULL,
        PRIMARY KEY (email),  -- for looking up one way
        INDEX(email_id)  -- for looking up the other way (UNIQUE is not needed)
    ) ENGINE = InnoDB;  -- to get clustering
```

Notes:

- MEDIUMINT is 3 bytes with UNSIGNED range of 0..16M; pick SMALLINT, INT, etc, based on a conservative estimate of how many 'foo's you will eventually have.
- datatype sizes
- There may be more than one VARCHAR in the table. Example: For cities, you might have City and Country.
- InnoDB is better than MyISAM because of way the two keys are structured.
- The secondary key is effectively (email_id, email), hence 'covering' for certain queries.
- It is OK to not specify an AUTO_INCREMENT to be UNIQUE.

Batched normalization

I bring this up as a separate topic because of some of the subtle issues that can happen.

You may be tempted to do

```
INSERT IGNORE INTO Foos
        SELECT DISTINCT foo FROM Staging;  -- not wise
```

It has the problem of “burning” AUTO_INCREMENT ids. This is because MariaDB pre-allocates ids before getting to “IGNORE”. That could rapidly increase the AUTO_INCREMENT values beyond what you expected.

Better is this…

```
INSERT IGNORE INTO Foos
        SELECT DISTINCT foo
            FROM Staging
            LEFT JOIN Foos ON Foos.foo = Staging.foo
            WHERE Foos.foo_id IS NULL;
```

Notes:

- The LEFT JOIN .. IS NULL finds the `foo`s that are not yet in Foos.
- This INSERT..SELECT must not be done inside the transaction with the rest of the processing. Otherwise, you add to deadlock risks, leading to burned ids.
- IGNORE is used in case you are doing the INSERT from multiple processes simultaneously.

Once that INSERT is done, this will find all the foo_ids it needs:

```
INSERT INTO Fact (..., foo_id, ...)
        SELECT ..., Foos.foo_id, ...
            FROM Staging
            JOIN Foos ON Foos.foo = Staging.foo;
```

An advantage of “Batched Normalization” is that you can summarize directly from the Staging table. Two approaches:

Case 1: PRIMARY KEY (dy, foo) and summarization is in lock step with, say, changes in `dy`.

- This approach can have troubles if new data arrives after you have summarized the day’s data.

```
INSERT INTO Summary (dy, foo, ct, blah_total)
        SELECT  DATE(dt) AS dy, foo,
                COUNT(*) AS ct, SUM(blah) AS blah_total)
            FROM Staging
            GROUP BY 1, 2;
```

Case 2: (dy, foo) is a non-UNIQUE INDEX.

- Same code as Case 1.
- By having the index be non-UNIQUE, delayed data simply shows up as extra rows.
- You need to take care to avoid summarizing the data twice. (The id on the Fact table may be a good tool for that.)

Case 3: PRIMARY KEY (dy, foo) and summarization can happen anytime.

```
INSERT INTO Summary (dy, foo, ct, blah_total)
        ON DUPLICATE KEY UPDATE
            ct = ct + VALUE(ct),
            blah_total = blah_total + VALUE(bt)
        SELECT  DATE(dt) AS dy, foo,
                COUNT(*) AS ct, SUM(blah) AS bt)
            FROM Staging
            GROUP BY 1, 2;
```

Too many choices?

This document lists a number of ways to do things. Your situation may lead to one approach being more/less acceptable. But, if you are thinking “Just tell me what to do!”, then here:

- Batch load the raw data into a temporary table (`Staging`).
- Normalize from `Staging` – use code in Case 3.
- INSERT .. SELECT to move the data from `Staging` into the Fact table
- Summarize from `Staging` to Summary table(s) via IODKU (Insert … On Duplicate Key Update).
- Drop the Staging

Those techniques should perform well and scale well in most cases. As you develop your situation, you may discover why I described alternative solutions.

Purging old data

Typically the Fact table is PARTITION BY RANGE (10-60 ranges of days/weeks/etc) and needs purging (DROP PARTITION) periodically. This discusses a safe/clean way to design the partitioning and do the DROPs: Purging PARTITIONs

Master / slave

For “read scaling”, backup, and failover, use master-slave replication or something fancier. Do ingestion only on a single active master; it replicate to the slave(s). Generate reports on the slave(s).

Sharding

“Sharding” is the splitting of data across multiple servers. (In contrast, replication and Galera have the same data on all servers, requiring all data to be written to all servers.)

With the non-sharding techniques described here, terabyte(s) of data can be handled by a single machine. Tens of terabytes probably requires sharding.

Sharding is beyond the scope of this document.

How fast? How big?

With the techniques described here, you may be able to achieve the following performance numbers. I say “may” because every data warehouse situation is different, and you may require performance-hurting deviations from what I describe here. I give multiple options for some aspects; these may cover some of your deviations.

One big performance killer is UUID/GUID keys. Since they are very ‘random’, updates of them (at scale) are limited to 1 row = 1 disk hit. Plain disks can handle only 100 hits/second. RAID and/or SSD can increase that to something like 1000 hits/sec. Huge amounts of RAM (for caching the random index) are a costly solution. It is possible to turn type-1 UUIDs into roughly-chronological keys, thereby mittigating the performance problems if the UUIDs are written/read with some chronological clustering. UUID discussion

Hardware, etc:

- Single SATA drive: 100 IOPs (Input/Output operations per second)

- RAID with N physical drives – 100*N IOPs (roughly)
- SSD – 5 times as fast as rotating media (in this context)
- Batch INSERT – 100-1000 rows is 10 times as fast as INSERTing 1 row at a time (see above)
- Purge “old” data – Do not use DELETE or TRUNCATE, design so you can use DROP PARTITION (see above)
- Think of each INDEX (except the PRIMARY KEY on InnoDB) as a separate table
- Consider access patterns of each table/index: random vs at-the-end vs something in between

“Count the disk hits” – back-of-envelope performance analysis

- Random accesses to a table/index – count each as a disk hit.
- At-the-end accesses (INSERT chronologically or with AUTO_INCREMENT; range SELECT) – count as zero hits.
- In between (hot/popular ids, etc) – count as something in between
- For INSERTs, do the analysis on each index; add them up.
- For SELECTs, do the analysis on the one index used, plus the table. (Use of 2 indexes is rare.)
  Insert cost, based on datatype of first column in an index:
- AUTO_INCREMENT – essentially 0 IOPs
- DATETIME, TIMESTAMP – essentially 0 for ‘current’ times
- UUID/GUID – 1 per insert (terrible)
- Others – depends on their patterns
  SELECT cost gets a little tricky:
- Range on PRIMARY KEY – think of it as getting 100 rows per disk hit.
- IN on PRIMARY KEY – 1 disk hit per item in IN
- “=” – 1 hit (for 1 row)
- Secondary key – First compute the hits for the index, then…
- Think of each row as needing 1 disk hit.
- However, if the rows are likely to be ‘near’ each other (based on the PRIMARY KEY), then it could be < 1 disk hit/row.

More on Count the Disk Hits

How fast?

Look at your data; compute raw rows per second (or hour or day or year). There are about 30M seconds in a year; 86,400 seconds per day. Inserting 30 rows per second becomes a billion rows per year.

10 rows per second is about all you can expect from an ordinary machine (after allowing for various overheads). If you have less than that, you don’t have many worries, but still you should probably create Summary tables. If more than 10/sec, then batching, etc, becomes vital. Even on spiffy hardware, 100/sec is about all you can expect without utilizing the techniques here.

Not so fast?

Let’s say your insert rate is only one-tenth of your disk IOPs (eg, 10 rows/sec vs 100 IOPs). Also, let’s say your data is not “bursty”; that is, the data comes in somewhat soothly throughout the day.

Note that 10 rows/sec (300M/year) implies maybe 30GB for data + indexes + normalization tables + summary tables for 1 year. I would call this “not so big”.

Still, the normalization and summarization are important. Normalization keeps the data from being, say, twice as big. Summarization speeds up the reports by orders of magnitude.

Let’s design and analyse a “simple ingestion scheme” for 10 rows/second, without ‘batching’.

```
# Normalize:
    $foo_id = SELECT foo_id FROM Foos WHERE foo = $foo;
    IF NO $foo_id, THEN
        INSERT IGNORE INTO Foos ...

    # Inserts:
    BEGIN;
        INSERT INTO Fact ...;
        INSERT INTO Summary ... ON DUPLICATE KEY UPDATE ...;
    COMMIT;
    # (plus code TO deal WITH errors ON INSERTs OR COMMIT)
```

Depending on the number and randomness of your indexes, etc, 10 Fact rows may (or may not) take less than 100 IOPs.

Also, note that as the data grows over time, random indexes will become less and less likely to be cached. That is, even if runs fine with 1 year's worth of data, it may be in trouble with 2 year's worth.

For those reasons, I started this discussion with a wide margin (10 rows versus 100 IOPs).

References

- sec. 3.3.2: Dimensional Model and "Star schema"
- Summary Tables

See also

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: datawarehouse

This page is licensed: CC BY-SA / Gnu FDL

DISTINCT removal in aggregate functions

Basics

One can use `DISTINCT` keyword to de-duplicate the arguments of an aggregate function. For example:

```
SELECT COUNT(DISTINCT col1) FROM tbl1;
```

In order to compute this, MariaDB has to collect the values of `col1` and remove the duplicates. This may be computationally expensive.

After the fix for MDEV-30660 (available from MariaDB 10.5.25, MariaDB 10.6.18, MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5, MariaDB 11.2.4, MariaDB 11.4.2), the optimizer can detect certain cases when argument of aggregate function will not have duplicates and so de-duplication can be skipped.

When one can skip de-duplication

A basic example: if we're doing a select from one table, then the values of `primary_key` are already distinct:

```
SELECT aggregate_func(DISTINCT tbl.primary_key, ...) FROM tbl;
```

If the SELECT has other constant tables, that's also ok, as they will not create duplicates.

The next step: a part of the primary key can be "bound" by the GROUP BY clause. Consider a query:

```
SELECT aggregate_func(DISTINCT t1.pk1, ...) FROM t1 GROUP BY t1.pk2;
```

Suppose the table has `PRIMARY KEY(pk1, pk2)`. Grouping by `pk2` fixes the value of `pk2` within each group. Then, the values of `pk1` must be unique within each group, and de-duplication is not necessary.

Observability

`EXPLAIN` or `EXPLAIN FORMAT=JSON` do not show any details about how aggregate functions are computed. One has to look at the Optimizer Trace. Search for `aggregator_type`:

When de-duplication is necessary, it will show:

```
{
            "prepare_sum_aggregators": {
              "function": "count(distinct t1.col1)",
              "aggregator_type": "distinct"
            }
          }
```

When de-duplication is not necessary, it will show:

```
{
            "prepare_sum_aggregators": {
              "function": "count(distinct t1.pk1)",
              "aggregator_type": "simple"
            }
          }
```

This page is licensed: CC BY-SA / Gnu FDL

Equality propagation optimization

Basic idea

Consider a query with a WHERE clause:

```
WHERE col1=col2 AND ...
```

the WHERE clause will compute to true only if `col1=col2`. This means that in the rest of the WHERE clause occurrences of `col1` can be substituted with `col2` (with some limitations which are discussed in the next section). This allows the optimizer to infer additional restrictions.

For example:

```
WHERE col1=col2 AND col1=123
```

allows to infer a new equality: `col2=123`

```
WHERE col1=col2 AND col1 < 10
```

allows to infer that `col2<10`.

Identity and comparison substitution

There are some limitations to where one can do the substitution, though.

The first and obvious example is the string datatype and collations. Most commonly-used collations in SQL are “case-insensitive”, that is `'A'='a'`. Also, most collations have a “PAD SPACE” attribute, which means that comparison ignores the spaces at the end of the value, `'a'='a '`.

Now, consider a query:

```
INSERT INTO t1 (col1, col2) VALUES ('ab', 'ab   ');
SELECT * FROM t1 WHERE col1=col2 AND LENGTH(col1)=2
```

Here, `col1=col2`, the values are "equal". At the same time `LENGTH(col1)=2`, while `LENGTH(col2)=4`, which means one can't perform the substitution for the argument of LENGTH(…).

It's not only collations. There are similar phenomena when equality compares columns of different datatypes. The exact criteria of when thy happen are rather convoluted.

The take-away is: **sometimes, X=Y does not mean that one can replace any reference to X with Y**.
What one CAN do is still replace the occurrence in the comparisons `<`, `>`, `>=`, `<=`, etc.

This is how we get two kinds of substitution:

- Identity substitution: X=Y, and any occurrence of X can be replaced with Y.
- Comparison substitution: X=Y, and an occurrence of X in a comparison (X<Z) can be replaced with Y (Y<Z).

Place in query optimization

(A draft description):
Let's look at how Equality Propagation is integrated with the rest of the query optimization process.

- First, multiple-equalities are built (TODO example from optimizer trace)
  - If multiple-equality includes a constant, fields are substituted with a constant if possible.
- From this point, all optimizations like range optimization, ref access, etc make use of multiple equalities: when they see a reference to `tableX.columnY` somewhere, they also look at all the columns that tableX.columnY is equal to.
- After the join order is picked, the optimizer walks through the WHERE clause and substitutes each field reference with the "best" one - the one that can be checked as soon as possible.
  - Then, the parts of the WHERE condition are attached to the tables where they can be checked.

Interplay with ORDER BY optimization

Consider a query:

```
SELECT ... FROM ... WHERE col1=col2 ORDER BY col2
```

Suppose, there is an `INDEX(col1)`. MariaDB optimizer is able to figure out that it can use an index on `col1` (or sort by the value of `col1`) in order to resolve `ORDER BY col2`.

Optimizer trace

Look at these elements:

- `condition_processing`
- `attaching_conditions_to_tables`

More details

Equality propagation doesn't just happen at the top of the WHERE clause. It is done "at all levels" where a level is:

- A top level of the WHERE clause.
- If the WHERE clause has an OR clause, each branch of the OR clause.
- The top level of any ON expression
- (the same as above about OR-levels)

This page is licensed: CC BY-SA / Gnu FDL

Filesort with Small LIMIT Optimization

Optimization Description

When `n` is sufficiently small, the optimizer will use a priority queue for sorting. Before the optimization's porting to MariaDB 10.0, the alternative was, roughly speaking, to sort the entire output and then pick only first `n` rows.

NOTE: The problem of choosing which index to use for query with ORDER BY … LIMIT is a different problem, see optimizer_join_limit_pref_ratio-optimization.

Optimization Visibility in MariaDB

There are two ways to check whether filesort has used a priority queue.

Status Variable

The first way is to check the Sort_priority_queue_sorts status variable. It shows the number of times that sorting was done through a priority queue. (The total number of times sorting was done is a sum Sort_range and Sort_scan).

Slow Query Log

The second way is to check the slow query log. When one uses Extended statistics in the slow query log and specifies log_slow_verbosity=query_plan, slow query log entries look like this

```
# Time: 140714 18:30:39
# User@Host: root[root] @ localhost []
# Thread_id: 3  Schema: test  QC_hit: No
# Query_time: 0.053857  Lock_time: 0.000188  Rows_sent: 11  Rows_examined: 100011
# Full_scan: Yes  Full_join: No  Tmp_table: No  Tmp_table_on_disk: No
# Filesort: Yes  Filesort_on_disk: No  Merge_passes: 0  Priority_queue: Yes
SET TIMESTAMP=1405348239;SET TIMESTAMP=1405348239;
SELECT * FROM t1 WHERE col1 BETWEEN 10 AND 20 ORDER BY col2 LIMIT 100;
```

Note the "Priority_queue: Yes" on the last comment line. (`pt-query-digest` is able to parse slow query logs with the Priority_queue field)

As for `EXPLAIN`, it will give no indication whether filesort uses priority queue or the generic quicksort and merge algorithm. `Using filesort` will be shown in both cases, by both MariaDB and MySQL.

See Also

- LIMIT Optimization page in the MySQL 5.6 manual (search for "priority queue").
- MySQL WorkLog entry, WL#1393
- MDEV-415, MDEV-6430

This page is licensed: CC BY-SA / Gnu FDL

FORCE INDEX

Description

Forcing an index to be used is mostly useful when the optimizer decides to do a table scan even if you know that using an index would be better. (The optimizer could decide to do a table scan even if there is an available index when it believes that most or all rows will match and it can avoid the overhead of using the index).

FORCE INDEX works by only considering the given indexes (like with USE_INDEX) but in addition it tells the optimizer to regard a table scan as something very expensive. However if none of the 'forced' indexes can be used,

then a table scan will be used anyway.

FORCE INDEX cannot force an ignored index to be used - it will be treated as if it doesn't exist.

Example

```
CREATE INDEX Name ON City (Name);
EXPLAIN SELECT Name,CountryCode FROM City FORCE INDEX (Name)
WHERE name>="A" AND CountryCode >="A";
```

This produces:

```
id  select_type table   type    possible_keys   key key_len ref rows    Extra
1   SIMPLE  City    range   Name    Name    35  NULL    4079    Using where
```

Index Prefixes

When using index hints (USE, FORCE or IGNORE INDEX), the index name value can also be an unambiguous prefix of an index name.

See Also

- Index Hints: How to Force Query Plans for more details
- USE INDEX
- IGNORE INDEX
- Ignored Indexes

This page is licensed: CC BY-SA / Gnu FDL

Groupwise Max in MariaDB

The problem

You want to find the largest row in each group of rows. An example is looking for the largest city in each state. While it is easy to find the MAX(population) … GROUP BY state, it is hard to find the name of the `city` associated with that `population`. Alas, MySQL and MariaDB do not have any syntax to provide the solution directly.

This article is under construction, mostly for cleanup. The content is reasonably accurate during construction.

The article presents two "good" solutions. They differ in ways that make neither of them 'perfect'; you should try both and weigh the pros and cons.

Also, a few "bad" solutions will be presented, together with why they were rejected.

MySQL manual gives 3 solutions; only the "Uncorrelated" one is "good", the other two are "bad".

Sample data

To show how the various coding attempts work, I have devised this simple task: Find the largest city in each Canadian province. Here's a sample of the source data (5493 rows):

```
+------------------+----------------+------------+
| province         | city           | population |
+------------------+----------------+------------+
| Saskatchewan     | Rosetown       |       2309 |
| British Columbia | Chilliwack     |      51942 |
| Nova Scotia      | Yarmouth       |       7500 |
| Alberta          | Grande Prairie |      41463 |
| Quebec           | Sorel          |      33591 |
| Ontario          | Moose Factory  |       2060 |
| Ontario          | Bracebridge    |       8238 |
| British Columbia | Nanaimo        |      84906 |
| Manitoba         | Neepawa        |       3151 |
| Alberta          | Grimshaw       |       2560 |
| Saskatchewan     | Carnduff       |        950 |
...
```

Here's the desired output (13 rows):

```
+--------------------------+---------------+------------+
| province                 | city          | population |
+--------------------------+---------------+------------+
| Alberta                  | Calgary       |     968475 |
| British Columbia         | Vancouver     |    1837970 |
| Manitoba                 | Winnipeg      |     632069 |
| New Brunswick            | Saint John    |      87857 |
| Newfoundland and Labrador | Corner Brook  |      18693 |
| Northwest Territories    | Yellowknife   |      15866 |
| Nova Scotia              | Halifax       |     266012 |
| Nunavut                  | Iqaluit       |       6124 |
| Ontario                  | Toronto       |    4612187 |
| Prince Edward Island     | Charlottetown |      42403 |
| Quebec                   | Montreal      |    3268513 |
| Saskatchewan             | Saskatoon     |     198957 |
| Yukon                    | Whitehorse    |      19616 |
+--------------------------+---------------+------------+
```

Duplicate max

One thing to consider is whether you want – or do not want – to see multiple rows for tied winners. For the dataset being used here, that would imply that the two largest cities in a province had identical populations. For this case, a duplicate would be unlikely. But there are many groupwise-max use cases where duplicates are likely.

The two best algorithms differ in whether they show duplicates.

Using an uncorrelated subquery

Characteristics:

- Superior performance or medium performance
- It will show duplicates
- Needs an extra index
- Probably requires 5.6
- If all goes well, it will run in O(M) where M is the number of output rows.

An 'uncorrelated subquery':

```
SELECT  c1.province, c1.city, c1.population
    FROM  Canada AS c1
    JOIN
      ( SELECT  province, MAX(population) AS population
            FROM  Canada
            GROUP BY  province
      ) AS c2 USING (province, population)
    ORDER BY c1.province;
```

But this also 'requires' an extra index: INDEX(province, population). In addition, MySQL has not always been able to use that index effectively, hence the "requires 5.6". (I am not sure of the actual version.)

Without that extra index, you would need 5.6, which has the ability to create indexes for subqueries. This is indicated by <auto_key0> in the EXPLAIN. Even so, the performance is worse with the auto-generated index than with the manually generated one.

With neither the extra index, nor 5.6, this 'solution' would belong in 'The Duds' because it would run in O(N*N) time.

Using @variables

Characteristics:

- Good performance
- Does not show duplicates (picks one to show)
- Consistent O(N) run time (N = number of input rows)
- Only one scan of the data

```
SELECT
        province, city, population   -- The desired columns
    FROM
      ( SELECT  @prev := '' ) init
    JOIN
      ( SELECT  province != @prev AS first,  -- `province` is the 'GROUP BY'
                @prev := province,           -- The 'GROUP BY'
                province, city, population   -- Also the desired columns
            FROM  Canada           -- The table
            ORDER BY
                province,          -- The 'GROUP BY'
                population DESC    -- ASC for MIN(population), DESC for MAX
      ) x
    WHERE  first
    ORDER BY  province;     -- Whatever you like
```

For your application, change the lines with comments.

The duds

*** 'correlated subquery' (from MySQL doc):**

```
SELECT  province, city, population
    FROM  Canada AS c1
    WHERE  population =
      ( SELECT  MAX(c2.population)
            FROM  Canada AS c2
            WHERE  c2.province= c1.province
      )
    ORDER BY  province;
```

O(N*N) (that is, terrible) performance

*** LEFT JOIN (from MySQL doc):**

```
SELECT  c1.province, c1.city, c1.population
    FROM  Canada AS c1
    LEFT JOIN  Canada AS c2 ON c2.province = c1.province
      AND  c2.population > c1.population
    WHERE  c2.province IS NULL
    ORDER BY province;
```

Medium performance (2N-3N, depending on join_buffer_size).

For O(N*N) time,... It will take one second to do a groupwise-max on a few thousand rows; a million rows could take hours.

Top-N in each group

This is a variant on “groupwise-max” wherein you desire the largest (or smallest) N items in each group. Do these substitutions for your use case:

- province –> your ‘GROUP BY’
- Canada –> your table
- 3 –> how many of each group to show
- population –> your numeric field for determining “Top-N”
- city –> more field(s) you want to show
- Change the SELECT and ORDER BY if you desire
- DESC to get the ‘largest’; ASC for the ‘smallest’

```
SELECT
        province, n, city, population
    FROM
      ( SELECT  @prev := '', @n := 0 ) init
    JOIN
      ( SELECT  @n := if(province != @prev, 1, @n + 1) AS n,
                @prev := province,
                province, city, population
            FROM  Canada
            ORDER BY
                province   ASC,
                population DESC
      ) x
    WHERE  n <= 3
    ORDER BY  province, n;
```

Output:

```
+--------------------------+------+-----------------+------------+
| province                 | n    | city            | population |
+--------------------------+------+-----------------+------------+
| Alberta                  |    1 | Calgary         |     968475 |
| Alberta                  |    2 | Edmonton        |     822319 |
| Alberta                  |    3 | Red Deer        |      73595 |
| British Columbia         |    1 | Vancouver       |    1837970 |
| British Columbia         |    2 | Victoria        |     289625 |
| British Columbia         |    3 | Abbotsford      |     151685 |
| Manitoba                 |    1 | ...
```

The performance of this is O(N), actually about 3N, where N is the number of source rows.

EXPLAIN EXTENDED gives

```
+----+-------------+------------+--------+---------------+------+---------+------+------+----------+-----
-----------+
| id | select_type | table      | type   | possible_keys | key  | key_len | ref  | rows | filtered |
Extra          |
+----+-------------+------------+--------+---------------+------+---------+------+------+----------+-----
-----------+
|  1 | PRIMARY     | <derived2> | system | NULL          | NULL | NULL    | NULL |    1 |   100.00 |
Using filesort |
|  1 | PRIMARY     | <derived3> | ALL    | NULL          | NULL | NULL    | NULL | 5484 |   100.00 |
Using where    |
|  3 | DERIVED     | Canada     | ALL    | NULL          | NULL | NULL    | NULL | 5484 |   100.00 |
Using filesort |
|  2 | DERIVED     | NULL       | NULL   | NULL          | NULL | NULL    | NULL | NULL |     NULL | No
tables used |
+----+-------------+------------+--------+---------------+------+---------+------+------+----------+-----
-----------+
```

Explanation, shown in the same order as the EXPLAIN, but numbered chronologically:
3. Get the subquery id=2 (init)
4. Scan the output from subquery id=3 (x)
2. Subquery id=3 – the table scan of Canada

1. Subquery id=2 – `init` for simply initializing the two @variables
   Yes, it took two sorts, though probably in RAM.

Main Handler values:

```
| Handler_read_rnd           | 39    |
| Handler_read_rnd_next      | 10971 |
| Handler_write              | 5485  |  -- #rows in Canada (+1)
```

Top-n in each group, take II

This variant is faster than the previous, but depends on `city` being unique across the dataset. (from openark.org)

```
SELECT  province, city, population
        FROM  Canada
        JOIN
          ( SELECT  GROUP_CONCAT(top_in_province) AS top_cities
                FROM
                  ( SELECT  SUBSTRING_INDEX(
                                  GROUP_CONCAT(city ORDER BY  population DESC),
                            ',', 3) AS top_in_province
                        FROM  Canada
                        GROUP BY  province
                  ) AS x
          ) AS y
        WHERE  FIND_IN_SET(city, top_cities)
        ORDER BY  province, population DESC;
```

Output. Note how there can be more than 3 cities per province:

```
| Alberta                  | Calgary          |     968475 |
| Alberta                  | Edmonton         |     822319 |
| Alberta                  | Red Deer         |      73595 |
| British Columbia         | Vancouver        |    1837970 |
| British Columbia         | Victoria         |     289625 |
| British Columbia         | Abbotsford       |     151685 |
| British Columbia         | Sydney           |          0 | -- Nova Scotia's second largest is Sydney
| Manitoba                 | Winnipeg         |     632069 |
```

Main Handler values:

```
| Handler_read_next          | 5484  | -- table size
| Handler_read_rnd_next      | 5500  | -- table size + number of provinces
| Handler_write              | 14    | -- number of provinces (+1)
```

Top-n using MyISAM

(This does not need your table to be MyISAM, but it does need MyISAM tmp table for its 2-column PRIMARY KEY feature.) See previous section for what changes to make for your use case.

```
-- build tmp table to get numbering
    -- (Assumes auto_increment_increment = 1)
    CREATE TEMPORARY TABLE t (
        nth MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
        PRIMARY KEY(province, nth)
    ) ENGINE=MyISAM
        SELECT province, NULL AS nth, city, population
            FROM Canada
            ORDER BY population DESC;
    -- Output the biggest 3 cities in each province:
    SELECT province, nth, city, population
        FROM t
        WHERE nth <= 3
        ORDER BY province, nth;

+--------------------------+-----+------------------+------------+
| province                 | nth | city             | population |
+--------------------------+-----+------------------+------------+
| Alberta                  |   1 | Calgary          |     968475 |
| Alberta                  |   2 | Edmonton         |     822319 |
| Alberta                  |   3 | Red Deer         |      73595 |
| British Columbia         |   1 | Vancouver        |    1837970 |
| British Columbia         |   2 | Victoria         |     289625 |
| British Columbia         |   3 | Abbotsford       |     151685 |
| Manitoba                 |  ...

SELECT FOR CREATE:
+----+-------------+--------+------+---------------+------+---------+------+------+----------------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows | Extra          |
+----+-------------+--------+------+---------------+------+---------+------+------+----------------+
|  1 | SIMPLE      | Canada | ALL  | NULL          | NULL | NULL    | NULL | 5484 | Using filesort |
+----+-------------+--------+------+---------------+------+---------+------+------+----------------+
Other SELECT:
+----+-------------+-------+-------+---------------+---------+---------+------+------+-------------+
| id | select_type | table | type  | possible_keys | key     | key_len | ref  | rows | Extra       |
+----+-------------+-------+-------+---------------+---------+---------+------+------+-------------+
|  1 | SIMPLE      | t     | index | NULL          | PRIMARY | 104     | NULL |   22 | Using where |
+----+-------------+-------+-------+---------------+---------+---------+------+------+-------------+
```

The main handler values (total of all operations):

```
| Handler_read_rnd_next      | 10970 |
| Handler_write              | 5484  |  -- number of rows in Canada (write tmp table)
```

Both “Top-n” formulations probably take about the same amount of time.

Windowing functions

Hot off the press from Percona Live… MariaDB 10.2 has “windowing functions”, which make “groupwise max” much more straightforward.

The code: TBD

Postlog

Developed a first posted, Feb, 2015; Add MyISAM approach: July, 2015; Openark's method added: Apr, 2016; Windowing: Apr 2016

I did not include the technique(s) using GROUP_CONCAT. They are useful in some situations with small datasets. They can be found in the references below.

See also

- Jan Kneschke's blog from 2007
- StackOverflow discussion of 'Uncorrelated'
- Other references: Inner ORDER BY thrown away
- Adding a large LIMIT to a subquery may make things work. Why ORDER BY in subquery is ignored
- StackOverflow thread
- row_number(), rank(), dense_rank()
- Perentile blog

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: groupwise_max

This page is licensed: CC BY-SA / Gnu FDL

GUID/UUID Performance

The problem

GUIDs/UUIDs (Globally/Universally Unique Identifiers) are very random. Therefore, INSERTing into an index means jumping around a lot. Once the index is too big to be cached, most INSERTs involve a disk hit. Even on a beefy system, this limits you to a few hundred INSERTs per second.

MariaDB's UUID function.

This blog is mostly eliminated in MySQL 8.0 with the advent of the following function:UUID_TO_BIN(str, swap_flag).

Why it is a problem

A 'standard' GUID/UUID is composed of the time, machine identification and some other stuff. The combination should be unique, even without coordination between different computers that could be generating UUIDs simultaneously.

The top part of the GUID/UUID is the bottom part of the current time. The top part is the primary part of what would be used for placing the value in an ordered list (INDEX). This cycles in about 7.16 minutes.

Some math… If the index is small enough to be cached in RAM, each insert into the index is CPU only, with the writes being delayed and batched. If the index is 20 times as big as can be cached, then 19 out of 20 inserts will be a cache miss. (This math applies to any "random" index.)

Second problem

36 characters is bulky. If you are using that as a PRIMARY KEY in InnoDB and you have secondary keys, remember that each secondary key has an implicit copy of the PK, thereby making it bulky.

It is tempting to declare the UUID VARCHAR(36). And, since you probably are thinking globally, so you have CHARACTER SET utf8 (or utf8mb4). For utf8:

- 2 - Overhead for VAR
- 36 - chars
- 3 (or 4) bytes per character for utf8 (or utf8mb4)
  So, max length = 2+3*36 = 110 (or 146) bytes. For temp tables 108 (or 144) is actually used if a MEMORY table is used.

To compress

- utf8 is unnecessary (ascii would do); but this is obviated by the next two steps
- Toss dashes
- UNHEX
  Now it will fit in 16 bytes: BINARY(16)

Combining the problems and crafting a solution

But first, a caveat. This solution only works for "Time based" / "Version 1" UUIDs
They are recognizable by the "1" at the beginning of the third clump.

The manual's sample: 6ccd780c-baba-1026-9564-0040f4311e29 . A more current value (after a few years): 49ea2de3-17a2-11e2-8346-001eecac3efa . Notice how the 3rd part has slowly changed over time? Let's data is rearranged, thus:

```
1026-baba-6ccd780c-9564-0040f4311e29
      11e2-17a2-49ea2de3-8346-001eecac3efa
      11e2-17ac-106762a5-8346-001eecac3efa -- after a few more minutes
```

Now we have a number that increases nicely over time. Multiple sources won't be quite in time order, but they will be close. The "hot" spot for inserting into an INDEX(uuid) will be rather narrow, thereby making it quite cacheable and efficient.

If your SELECTs tend to be for "recent" uuids, then they, too, will be easily cached. If, on the other hand, your SELECTs often reach for old uuids, they will be random and not well cached. Still, improving the INSERTs will help the system overall.

Code to do it

Let's make Stored Functions to do the messy work of the two actions:

- Rearrange fields
- Convert to/from BINARY(16)

```
DELIMITER //

    CREATE FUNCTION UuidToBin(_uuid BINARY(36))
        RETURNS BINARY(16)
        LANGUAGE SQL  DETERMINISTIC  CONTAINS SQL  SQL SECURITY INVOKER
    RETURN
        UNHEX(CONCAT(
            SUBSTR(_uuid, 15, 4),
            SUBSTR(_uuid, 10, 4),
            SUBSTR(_uuid,  1, 8),
            SUBSTR(_uuid, 20, 4),
            SUBSTR(_uuid, 25) ));
    //
    CREATE FUNCTION UuidFromBin(_bin BINARY(16))
        RETURNS BINARY(36)
        LANGUAGE SQL  DETERMINISTIC  CONTAINS SQL  SQL SECURITY INVOKER
    RETURN
        LCASE(CONCAT_WS('-',
            HEX(SUBSTR(_bin,  5, 4)),
            HEX(SUBSTR(_bin,  3, 2)),
            HEX(SUBSTR(_bin,  1, 2)),
            HEX(SUBSTR(_bin,  9, 2)),
            HEX(SUBSTR(_bin, 11))
                 ));

    //
    DELIMITER ;
```

Then you would do things like

```
-- Letting MySQL create the UUID:
    INSERT INTO t (uuid, ...) VALUES (UuidToBin(UUID()), ...);

    -- Creating the UUID elsewhere:
    INSERT INTO t (uuid, ...) VALUES (UuidToBin(?), ...);

    -- Retrieving (point query using uuid):
    SELECT ... FROM t WHERE uuid = UuidToBin(?);

    -- Retrieving (other):
    SELECT UuidFromBin(uuid), ... FROM t ...;
```

Do not flip the WHERE; this will be inefficient because it won't use INDEX(uuid):

```
WHERE UuidFromBin(uuid) = '1026-baba-6ccd780c-9564-0040f4311e29' -- NO
```

Wrapup

This shows three thing for speeding up usage of GUIDs/UUIDs:

- Shrink footprint (Smaller -> more cacheable -> faster).
- Rearrange uuid to make a "hot spot" to improve cachability.
- Consider an engine whose indexing suits high-volume random inserts (MyRocks shares some architectural traits which may be beneficial in handling UUIDs, but this is hypothetical and hasn't been tested)

Note that the benefit of the "hot spot" is only partial:

- Chronologically ordered (or approximately ordered) INSERTs benefit; random ones don't.
- SELECTs/UPDATEs by "recent" uuids benefit; old ones don't benefit.

Postlog

Thanks to Trey for some of the ideas here.

The tips in this document apply to MySQL, MariaDB, and Percona.

Written Oct, 2012.

See Also

- UUID data type
- Detailed discussion of UUID indexing
- Graphical display of the random nature of UUID on PRIMARY KEY
- Benchmarks, etc, by Karthik Appigatla
- More details on the clock
- Percona benchmarks
- NHibernate can generate sequential GUIDs , but it seems to be backwards.

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: uuid

This page is licensed: CC BY-SA / Gnu FDL

hash_join_cardinality optimizer_switch Flag

MariaDB starting with 10.6.13

The hash_join_cardinality optimizer_switch flag was added in MariaDB 11.0.2, MariaDB 10.11.3, MariaDB 10.10.4, MariaDB 10.9.6, MariaDB 10.8.8 and MariaDB 10.6.13.

In MySQL and MariaDB, the output cardinality of a part of query has historically been tied to the used access method(s). This is different from the approach used in database textbooks. There, the cardinality "x JOIN y" is the same regardless of which access methods are used to compute it.

Example

Consider a query joining customers with their orders:

```
SELECT *
FROM
  customer, orders, ...
WHERE
  customer.id = orders.customer_id AND ...
```

Suppose, table orders has an index `IDX` on `orders.customer_id`.

If the query plan is using this index to fetch orders for each customer, the optimizer will use index statistics from `IDX` to estimate the number of rows in the customer-joined-with-orders.

On the other hand, if the optimizer considers a query plan that joins customer with orders without use of indexes, it will ignore the `customer.id = orders.customer_id` equality completely and will compute the output cardinality as if customer was cross-joined with orders.

Hash Join

MariaDB supports Block Hash Join. It is not enabled by default, one needs to set it join_cache_level to 3 or a bigger value to enable it.

Before MDEV-30812, Query optimization for Block Hash Join would work as described in the above example: It would assume that the join operation is a cross join.

MDEV-30812 introduces a new optimizer_switch flag, `hash_join_cardinality`. In MariaDB versions before 11.0, it is off by default.

If one sets it to ON, the optimizer will make use of column histograms when computing the cardinality of hash join operation output.

One can see the computation in the Optimizer Trace, search for `hash_join_cardinality`.

This page is licensed: CC BY-SA / Gnu FDL

How to Quickly Insert Data Into MariaDB

This article describes different techniques for inserting data quickly into MariaDB.

Background

When inserting new data into MariaDB, the things that take time are (in order of importance):

- Syncing data to disk (as part of the end of transactions)
- Adding new keys. The larger the index, the more time it takes to keep keys updated.
- Checking against foreign keys (if they exist).
- Adding rows to the storage engine.
- Sending data to the server.

The following describes the different techniques (again, in order of importance) you can use to quickly insert data into a table.

Disabling Keys

You can temporarily disable updating of non-unique indexes. This is mostly useful when there are zero (or very few) rows in the table into which you are inserting data.

```
ALTER TABLE table_name DISABLE KEYS;
BEGIN;
... inserting data WITH INSERT OR LOAD DATA ....
COMMIT;
ALTER TABLE table_name ENABLE KEYS;
```

In many storage engines (at least MyISAM and Aria), `ENABLE KEYS` works by scanning through the row data and collecting keys, sorting them and then creating the index blocks. This is an order of magnitude faster than creating the index one row at a time and it also uses less key buffer memory.

Note: When you insert into an **empty table** with INSERT or LOAD DATA, MariaDB **automatically** does a DISABLE KEYS before and an ENABLE KEYS afterwards.

When inserting big amounts of data, integrity checks are sensibly time-consuming. It is possible to disable the `UNIQUE` indexes and the foreign keys checks using the unique_checks and the foreign_key_checks system variables:

```
SET @@session.unique_checks = 0;
SET @@session.foreign_key_checks = 0;
```

For InnoDB tables, the AUTO_INCREMENT lock mode can be temporarily set to 2, which is the fastest setting:

```
SET @@global.innodb_autoinc_lock_mode = 2;
```

Also, if the table has INSERT triggers or PERSISTENT columns, you may want to drop them, insert all data, and recreate them.

Loading Text Files

The **fastest way** to insert data into MariaDB is through the LOAD DATA INFILE command.

The simplest form of the command is:

```
LOAD DATA INFILE 'file_name' INTO TABLE table_name;
```

You can also read a file locally on the machine where the client is running by using:

```
LOAD DATA LOCAL INFILE 'file_name' INTO TABLE table_name;
```

This is not as fast as reading the file on the server side, but the difference is not that big.

`LOAD DATA INFILE` is very fast because:

1. there is no parsing of SQL.
2. data is read in big blocks.
3. if the table is empty at the beginning of the operation, all non-unique indexes are disabled during the operation.
4. the engine is told to cache rows first and then insert them in big blocks (At least MyISAM and Aria support this).
5. for empty tables, some transactional engines (like Aria) do not log the inserted data in the transaction log because one can roll back the operation by just doing a TRUNCATE on the table.

Because of the above speed advantages there are many cases, when you need to insert **many** rows at a time, where it may be faster to create a file locally, add the rows there, and then use `LOAD DATA INFILE` to load them; compared to using `INSERT` to insert the rows.

You will also get progress reporting for `LOAD DATA INFILE`.

mariadb-import

You can import many files in parallel with mariadb-import (`mysqlimport` before MariaDB 10.5). For example:

```
mariadb-import --use-threads=10 database text-file-name [text-file-name...]
```

Internally mariadb-import uses LOAD DATA INFILE to read in the data.

Inserting Data with INSERT Statements

Using Big Transactions

When doing many inserts in a row, you should wrap them with `BEGIN / END` to avoid doing a full transaction (which includes a disk sync) for every row. For example, doing a begin/end every 1000 inserts will speed up your inserts by almost 1000 times.

```
BEGIN;
INSERT ...
INSERT ...
END;
BEGIN;
INSERT ...
INSERT ...
END;
...
```

The reason why you may want to have many `BEGIN/END` statements instead of just one is that the former will use up less transaction log space.

Multi-Value Inserts

You can insert many rows at once with multi-value row inserts:

```
INSERT INTO table_name VALUES(1,"row 1"),(2, "row 2"),...;
```

The limit for how much data you can have in one statement is controlled by the max_allowed_packet server variable.

Inserting Data Into Several Tables at Once

If you need to insert data into several tables at once, the best way to do so is to enable multi-row statements and send many inserts to the server at once:

```
INSERT INTO table_name_1 (auto_increment_key, data) VALUES (NULL,"row 1");
INSERT INTO table_name_2 (auto_increment, reference, data) VALUES (NULL, LAST_INSERT_ID(), "row 2");
```

LAST_INSERT_ID() is a function that returns the last `auto_increment` value inserted.

By default, the command line `mariadb` client will send the above as multiple statements.

To test this in the `mariadb` client you have to do:

```
delimiter ;;
SELECT 1; SELECT 2;;
delimiter ;
```

Note: for multi-query statements to work, your client must specify the `CLIENT_MULTI_STATEMENTS` flag to `mysql_real_connect()`.

Server Variables That Can be Used to Tune Insert Speed

Option	Description
innodb_buffer_pool_size	Increase this if you have many indexes in InnoDB/XtraDB tables
key_buffer_size	Increase this if you have many indexes in MyISAM tables
max_allowed_packet	Increase this to allow bigger multi-insert statements
read_buffer_size	Read block size when reading a file with LOAD DATA

See Server System Variables for the full list of server variables.

This page is licensed: CC BY-SA / Gnu FDL

IGNORE INDEX

Syntax

```
IGNORE INDEX [{FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
```

Description

You can tell the optimizer to not consider a particular index with the IGNORE INDEX option.

The benefit of using IGNORE_INDEX instead of USE_INDEX is that it will not disable a new index which you may add later.

Also see Ignored Indexes for an option to specify in the index definition that indexes should be ignored.

Index Prefixes

When using index hints (USE, FORCE or IGNORE INDEX), the index name value can also be an unambiguous prefix of an index name.

Example

This is used after the table name in the FROM clause:

```
CREATE INDEX Name ON City (Name);
CREATE INDEX CountryCode ON City (Countrycode);
EXPLAIN SELECT Name FROM City IGNORE INDEX (Name)
WHERE name="Helsingborg" AND countrycode="SWE";
```

This produces:

```
id  select_type table   type    possible_keys   key key_len ref rows    Extra
1   SIMPLE  City    ref CountryCode CountryCode 3   const   14  Using where
```

See Also

- See Index Hints: How to Force Query Plans for more details
- USE INDEX
- FORCE INDEX
- Ignored Indexes

This page is licensed: CC BY-SA / Gnu FDL

Index Condition Pushdown

Index Condition Pushdown is an optimization that is applied for access methods that access table data through indexes: `range`, `ref`, `eq_ref`, `ref_or_null`, and Batched Key Access.

The idea is to check part of the WHERE condition that refers to index fields (we call it *Pushed Index Condition*) as soon as we've accessed the index. If the *Pushed Index Condition* is not satisfied, we won't need to read the whole table record.

Index Condition Pushdown is **on** by default. To disable it, set its optimizer_switch flag like so:

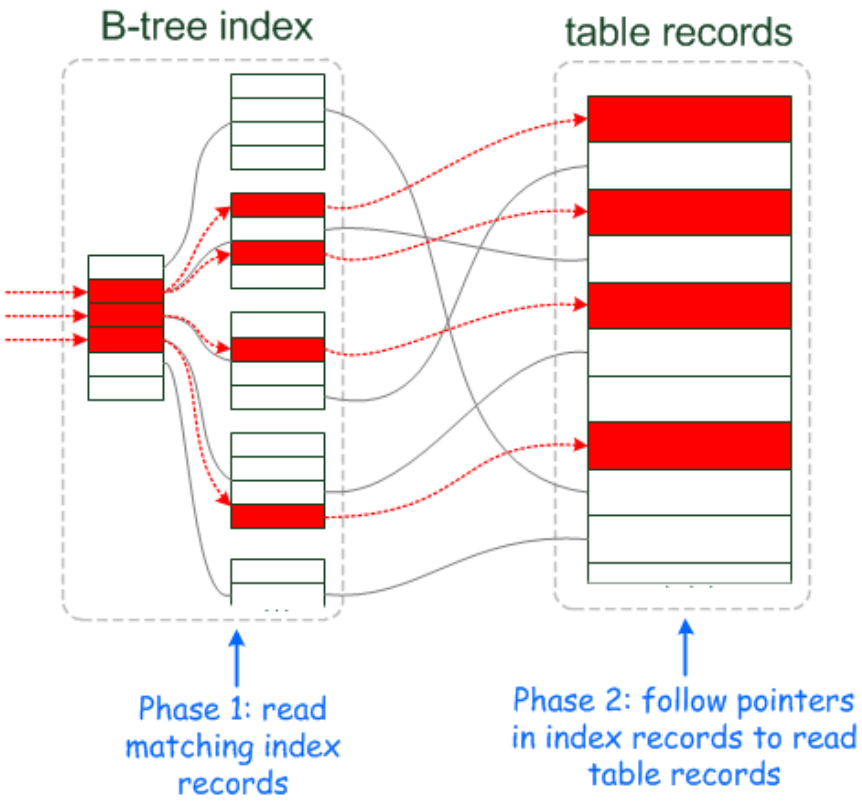
```
SET optimizer_switch='index_condition_pushdown=off'
```

When Index Condition Pushdown is used, EXPLAIN will show "Using index condition":

```
MariaDB [test]> EXPLAIN SELECT * FROM tbl WHERE key_col1 BETWEEN 10 AND 11 AND key_col2 LIKE '%foo%';
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
| id | select_type | table | type  | possible_keys | key      | key_len | ref  | rows | Extra
|
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
|  1 | SIMPLE      | tbl   | range | key_col1      | key_col1 | 5       | NULL |    2 | Using index
condition |
+----+-------------+-------+-------+---------------+----------+---------+------+------+-----------------
-----+
```

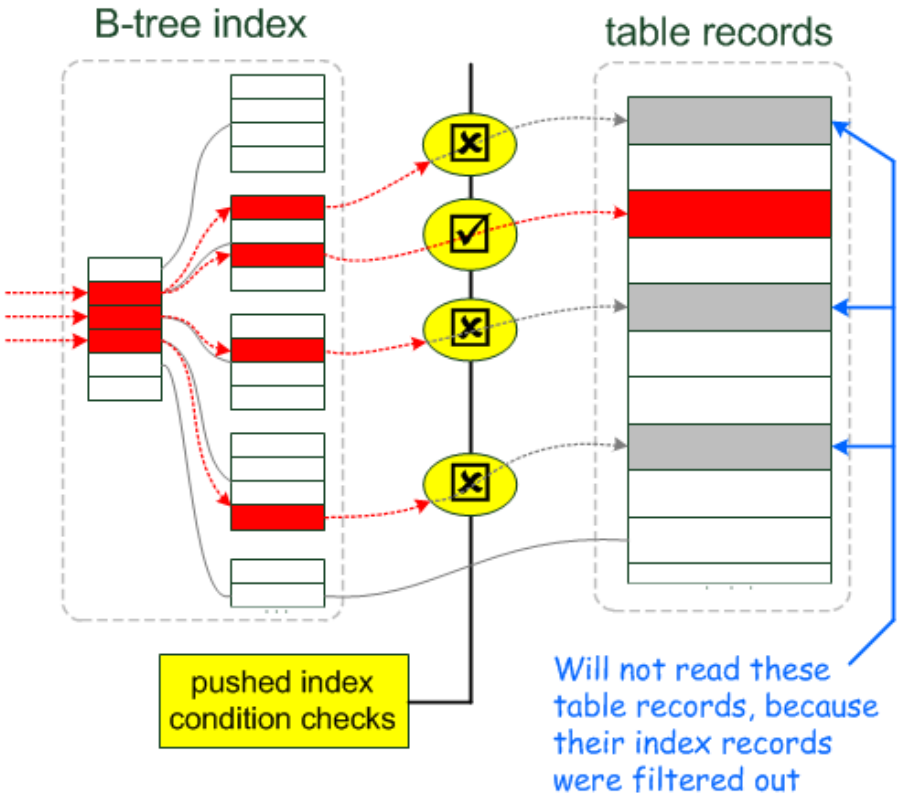
The Idea Behind Index Condition Pushdown

In disk-based storage engines, making an index lookup is done in two steps, like shown on the picture:



index-access-2phases

Index Condition Pushdown optimization tries to cut down the number of full record reads by checking whether index records satisfy part of the WHERE condition that can be checked for them:



index-access-with-icp

How much speed will be gained depends on

- How many records will be filtered out
- How expensive it was to read them

The former depends on the query and the dataset. The latter is generally bigger when table records are on disk and/or are big, especially when they have blobs.

Example Speedup

I used DBT-3 benchmark data, with scale factor=1. Since the benchmark defines very few indexes, we've added a multi-column index (index condition pushdown is usually useful with multi-column indexes: the first component(s) is what index access is done for, the subsequent have columns that we read and check conditions on).

```
ALTER TABLE lineitem ADD INDEX s_r (l_shipdate, l_receiptdate);
```

The query was to find big (l_quantity > 40) orders that were made in January 1993 that took more than 25 days to ship:

```
SELECT COUNT(*) FROM lineitem
WHERE
  l_shipdate BETWEEN '1993-01-01' AND '1993-02-01' AND
  datediff(l_receiptdate,l_shipdate) > 25 AND
  l_quantity > 40;
```

EXPLAIN without Index Condition Pushdown:

```
-+----------+-------+----------------------+-----+---------+------+--------+-------------+
 | table    | type | possible_keys         | key | key_len | ref | rows    | Extra       |
-+----------+-------+----------------------+-----+---------+------+--------+-------------+
 | lineitem | range | s_r                   | s_r | 4       | NULL | 152064 | Using where |
-+----------+-------+----------------------+-----+---------+------+--------+-------------+
```

with Index Condition Pushdown:

```
-+-----------+-------+---------------+-----+---------+------+--------+-----------------------------------
-+
 | table     | type | possible_keys | key | key_len | ref | rows     | Extra
|
-+-----------+-------+---------------+-----+---------+------+--------+-----------------------------------
-+
 | lineitem | range | s_r            | s_r | 4       | NULL | 152064 | Using index condition; Using where
|
-+-----------+-------+---------------+-----+---------+------+--------+-----------------------------------
-+
```

The speedup was:

- Cold buffer pool: from 5 min down to 1 min
- Hot buffer pool: from 0.19 sec down to 0.07 sec

Status Variables

There are two server status variables:

Variable name	Meaning
Handler_icp_attempts	Number of times pushed index condition was checked.
Handler_icp_match	Number of times the condition was matched.

That way, the value `Handler_icp_attempts - Handler_icp_match` shows the number records that the server did not have to read because of Index Condition Pushdown.

Limitations

- Currently, virtual column indexes can't be used for index condition pushdown. Instead, a generated column can be made declared STORED. Then, index condition pushdown will be possible.
- Index Condition Pushdown can't be used with backward-ordered index scan. When the optimizer needs to execute an `ORDER BY ... DESC` query which can be handled by using a backward-ordered index scan, it will disable Index Condition Pushdown.

Partitioned Tables

Index condition pushdown support for partitioned tables was added in MariaDB 11.5.

This page is licensed: CC BY-SA / Gnu FDL

Index Hints: How to Force Query Plans

The optimizer is largely cost-based and will try to choose the optimal plan for any query. However in some cases it does not have enough information to choose a perfect plan and in these cases you may have to provide hints to force the optimizer to use another plan.

You can examine the query plan for a SELECT by writing EXPLAIN before the statement. SHOW EXPLAIN shows the output of a running query. In some cases, its output can be closer to reality than `EXPLAIN`.

For the following queries, we will use the world database for the examples.

Setting up the World Example Database

Download it from world.sql.gz

Install it with:

```
mariadb-admin create world
zcat world.sql.gz | ../client/mysql world
```

or

```
mariadb-admin create world
gunzip world.sql.gz
../client/mysql world < world.sql
```

Forcing Join Order

You can force the join order by using STRAIGHT_JOIN either in the SELECT or JOIN part.

The simplest way to force the join order is to put the tables in the correct order in the `FROM` clause and use `SELECT STRAIGHT_JOIN` like so:

```
SELECT STRAIGHT_JOIN SUM(City.Population) FROM Country,City WHERE
City.CountryCode=Country.Code AND Country.HeadOfState="Volodymyr Zelenskyy";
```

If you only want to force the join order for a few tables, use `STRAIGHT_JOIN` in the `FROM` clause. When this is done, only tables connected with `STRAIGHT_JOIN` will have their order forced. For example:

```
SELECT SUM(City.Population) FROM Country STRAIGHT_JOIN City WHERE
City.CountryCode=Country.Code AND Country.HeadOfState="Volodymyr Zelenskyy";
```

In both of the above cases `Country` will be scanned first and for each matching country (one in this case) all rows in `City` will be checked for a match. As there is only one matching country this will be faster than the original query.

The output of EXPLAIN for the above cases is:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	Country	ALL	PRIMARY	NULL	NULL	NULL	239	Using where
1	SIMPLE	City	ALL	NULL	NULL	NULL	NULL	4079	Using where; Using join buffer (flat, BNL join)

This is one of the few cases where `ALL` is ok, as the scan of the `Country` table will only find one matching row.

Forcing Usage of a Specific Index for the WHERE Clause

In some cases the optimizer may choose a non-optimal index or it may choose to not use an index at all, even if some index could theoretically be used.

In these cases you have the option to either tell the optimizer to only use a limited set of indexes, ignore one or more indexes, or force the usage of some particular index.

USE INDEX: Use a Limited Set of Indexes

You can limit which indexes are considered with the USE INDEX option.

```
USE INDEX [{FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
```

The default is '`FOR JOIN`', which means that the hint only affects how the `WHERE` clause is optimized.

`USE INDEX` is used after the table name in the `FROM` clause.

Example:

```
CREATE INDEX Name ON City (Name);
CREATE INDEX CountryCode ON City (Countrycode);
EXPLAIN SELECT Name FROM City USE INDEX (CountryCode)
WHERE name="Helsingborg" AND countrycode="SWE";
```

This produces:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	City	ref	CountryCode	CountryCode	3	const	14	Using where

If we had not used USE INDEX, the `Name` index would have been in `possible keys`.

IGNORE INDEX: Don't Use a Particular Index

You can tell the optimizer to not consider some particular index with the IGNORE INDEX option.

```
IGNORE INDEX [{FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
```

This is used after the table name in the `FROM` clause:

```
CREATE INDEX Name ON City (Name);
CREATE INDEX CountryCode ON City (Countrycode);
EXPLAIN SELECT Name FROM City IGNORE INDEX (Name)
WHERE name="Helsingborg" AND countrycode="SWE";
```

This produces:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	City	ref	CountryCode	CountryCode	3	const	14	Using where

The benefit of using `IGNORE_INDEX` instead of `USE_INDEX` is that it will not disable a new index which you may add later.

Also see Ignored Indexes for an option to specify in the index definition that indexes should be ignored.

FORCE INDEX: Forcing an Index

Forcing an index to be used is mostly useful when the optimizer decides to do a table scan even if you know that using an index would be better. (The optimizer could decide to do a table scan even if there is an available index when it believes that most or all rows will match and it can avoid the overhead of using the index).

```
CREATE INDEX Name ON City (Name);
EXPLAIN SELECT Name,CountryCode FROM City FORCE INDEX (Name)
WHERE name>="A" AND CountryCode >="A";
```

This produces:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	City	range	Name	Name	35	NULL	4079	Using where

`FORCE_INDEX` works by only considering the given indexes (like with `USE_INDEX`) but in addition it tells the optimizer to regard a table scan as something very expensive. However if none of the 'forced' indexes can be used, then a table scan will be used anyway.

Index Prefixes

When using index hints (USE, FORCE or IGNORE INDEX), the index name value can also be an unambiguous prefix of an index name.

Forcing an Index to be Used for ORDER BY or GROUP BY

The optimizer will try to use indexes to resolve ORDER BY and GROUP BY.

You can use USE INDEX, IGNORE INDEX and FORCE INDEX as in the `WHERE` clause above to ensure that some specific index used:

```
USE INDEX [{FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
```

This is used after the table name in the `FROM` clause.

Example:

```
CREATE INDEX Name ON City (Name);
EXPLAIN SELECT Name,COUNT(*) FROM City
FORCE INDEX FOR GROUP BY (Name)
WHERE population >= 10000000 GROUP BY Name;
```

This produces:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	City	index	NULL	Name	35	NULL	4079	Using where

Without the FORCE INDEX option we would have '`Using where; Using temporary; Using filesort`' in the 'Extra' column, which means that the optimizer would created a temporary table and sort it.

Help the Optimizer Optimize GROUP BY and ORDER BY

The optimizer uses several strategies to optimize GROUP BY and ORDER BY:

- Resolve with an index:
  - Scan the table in index order and output data as we go. (This only works if the ORDER BY / GROUP BY can be resolved by an index after constant propagation is done).

- Filesort:
  - Scan the table to be sorted and collect the sort keys in a temporary file.
  - Sort the keys + reference to row (with filesort)
  - Scan the table in sorted order
- Use a temporary table for ORDER BY:
  - Create a temporary (in memory) table for the 'to-be-sorted' data. (If this gets bigger than `max_heap_table_size` or contains blobs then an Aria or MyISAM disk based table will be used)
  - Sort the keys + reference to row (with filesort)
  - Scan the table in sorted order

A temporary table will always be used if the fields which will be sorted are not from the first table in the JOIN order.

- Use a temporary table for GROUP BY:
  - Create a temporary table to hold the GROUP BY result with an index that matches the GROUP BY fields.
  - Produce a result row
  - If a row with the GROUP BY key exists in the temporary table, add the new result row to it. If not, create a new row.
  - Before sending the results to the user, sort the rows with filesort to get the results in the GROUP BY order.

Forcing/Disallowing TemporaryTables to be Used for GROUP BY:

Using an in-memory table (as described above) is usually the fastest option for GROUP BY if the result set is small. It is not optimal if the result set is very big. You can tell the optimizer this by using `SELECT SQL_SMALL_RESULT` or `SELECT SQL_BIG_RESULT`.

For example:

```
EXPLAIN SELECT SQL_SMALL_RESULT Name,Count(*) AS Cities
FROM City GROUP BY Name HAVING Cities > 2;
```

produces:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	City	ALL	NULL	NULL	NULL	NULL	4079	Using temporary; Using filesort

while:

```
EXPLAIN SELECT SQL_BIG_RESULT Name,Count(*) AS Cities
FROM City GROUP BY Name HAVING Cities > 2;
```

produces:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	City	ALL	NULL	NULL	NULL	NULL	4079	Using filesort

The difference is that with `SQL_SMALL_RESULT` a temporary table is used.

Forcing Usage of Temporary Tables

In some cases you may want to force the use of a temporary table for the result to free up the table/row locks for the used tables as quickly as possible.

You can do this with the `SQL_BUFFER_RESULT` option:

```
CREATE INDEX Name ON City (Name);
EXPLAIN SELECT SQL_BUFFER_RESULT Name,COUNT(*) AS Cities FROM City
GROUP BY Name HAVING Cities > 2;
```

This produces:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	City	index	NULL	Name	35	NULL	4079	Using index; Using temporary

Without `SQL_BUFFER_RESULT`, the above query would not use a temporary table for the result set.

Optimizer Switch

In MariaDB 5.3 we added an optimizer switch which allows you to specify which algorithms will be considered when optimizing a query.

See the optimizer section for more information about the different algorithms which are used.

See Also

- FORCE INDEX
- USE INDEX
- IGNORE INDEX
- GROUP BY
- Ignored Indexes

This page is licensed: CC BY-SA / Gnu FDL

index_merge sort_intersection

Prior to MariaDB 5.3, the `index_merge` access method supported `union`, `sort-union`, and `intersection` operations. Starting from MariaDB 5.3, the `sort-intersection` operation is also supported. This allows the use of `index_merge` in a broader number of cases.

This feature is disabled by default. To enable it, turn on the optimizer switch `index_merge_sort_intersection` like so:

```
SET optimizer_switch='index_merge_sort_intersection=on'
```

Limitations of index_merge/intersection

Prior to MariaDB 5.3, the `index_merge` access method had one intersection strategy called `intersection`. That strategy can only be used when merged index scans produced rowid-ordered streams. In practice this means that an `intersection` could only be constructed from equality (=) conditions.

For example, the following query will use `intersection`:

```
MySQL [ontime]> EXPLAIN SELECT AVG(arrdelay) FROM ontime WHERE depdel15=1 AND OriginState ='CA';
+--+-----------+------+-----------+-------------------+-------------------+-------+----+-----+---------
----------------------------------------+
|id|select_type|table |type       |possible_keys      |key                |key_len|ref |rows |Extra
|
+--+-----------+------+-----------+-------------------+-------------------+-------+----+-----+---------
----------------------------------------+
| 1|SIMPLE     |ontime|index_merge|OriginState,DepDel15|OriginState,DepDel15|3,5    |NULL|76952|Using
intersect(OriginState,DepDel15);Using where|
+--+-----------+------+-----------+-------------------+-------------------+-------+----+-----+---------
----------------------------------------+
```

but if you replace `OriginState ='CA'` with `OriginState IN ('CA', 'GB')` (which matches the same number of records), then `intersection` is not usable anymore:

```
MySQL [ontime]> EXPLAIN SELECT avg(arrdelay) FROM ontime where depdel15=1 and OriginState IN ('CA',
'GB');
+--+-----------+------+----+--------------------+--------+-------+-----+-----+-----------+
|id|select_type|table |type|possible_keys       |key     |key_len|ref  |rows |Extra      |
+--+-----------+------+----+--------------------+--------+-------+-----+-----+-----------+
| 1|SIMPLE     |ontime|ref |OriginState,DepDel15|DepDel15|5      |const|36926|Using where|
+--+-----------+------+----+--------------------+--------+-------+-----+-----+-----------+
```

The latter query would also run 5.x times slower (from 2.2 to 10.8 seconds) in our experiments.

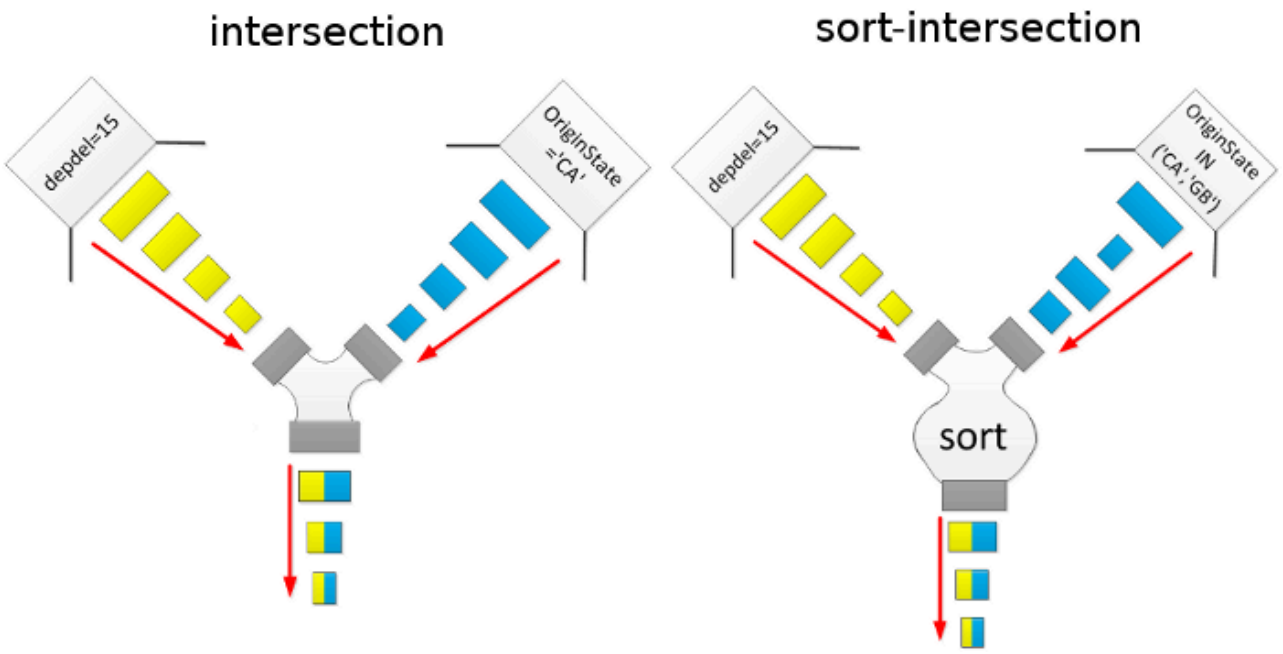
How index_merge/sort_intersection improves the situation

In MariaDB 5.3, when `index_merge_sort_intersection` is enabled, `index_merge` intersection plans can be constructed from non-equality conditions:

```
MySQL [ontime]> EXPLAIN SELECT avg(arrdelay) FROM ontime where depdel15=1 and OriginState IN ('CA',
'GB');
+--+-----------+------+-----------+--------------------+--------------------+-------+----+-----+---------
-----------------------------------------------+
|id|select_type|table |type       |possible_keys       |key                 |key_len|ref |rows |Extra
|
+--+-----------+------+-----------+--------------------+--------------------+-------+----+-----+---------
-----------------------------------------------+
| 1|SIMPLE     |ontime|index_merge|OriginState,DepDel15|DepDel15,OriginState|5,3    |NULL|60754|Using
sort_intersect(DepDel15,OriginState); Using where |
+--+-----------+------+-----------+--------------------+--------------------+-------+----+-----+---------
-----------------------------------------------+
```

In our tests, this query ran in 3.2 seconds, which is not as good as the case with two equalities, but still much better than 10.8 seconds we were getting without `sort_intersect`.

The `sort_intersect` strategy has higher overhead than `intersect` but is able to handle a broader set of `WHERE` conditions.



intersect-vs-sort-intersect

When to Use

`index_merge/sort_intersection` works best on tables with lots of records and where intersections are sufficiently large (but still small enough to make a full table scan overkill).

The benefit is expected to be bigger for io-bound loads.

This page is licensed: CC BY-SA / Gnu FDL

LIMIT ROWS EXAMINED

Syntax

```
SELECT ... FROM ... WHERE ...
[group_clause] [order_clause]
LIMIT [[OFFSET,] row_count] ROWS EXAMINED rows_limit;
```

Similar to the parameters of `LIMIT`, *`rows_limit`* can be both a prepared statement parameter, or a stored program parameter.

Description

The purpose of this optimization is to provide the means to terminate the execution of SELECT statements which examine too many rows, and thus use too many resources. This is achieved through an extension of the LIMIT clause — `LIMIT ROWS EXAMINED number_of_rows`. Whenever possible the semantics of `LIMIT ROWS EXAMINED` is the same as that of normal `LIMIT` (for instance for aggregate functions).

The `LIMIT ROWS EXAMINED` clause is taken into account by the query engine only during query execution. Thus the clause is ignored in the following cases:

- If a query is EXPLAIN-ed.
- During query optimization.
- During auxiliary operations such as writing to system tables (e.g. logs).

The clause is not applicable to DELETE or UPDATE statements, and if used in those statements produces a syntax error.

The effects of this clause are as follows:

- The server counts the number of read, inserted, modified, and deleted rows during query execution. This takes into account the use of temporary tables, and sorting for intermediate query operations.
- Once the counter exceeds the value specified in the LIMIT ROWS EXAMINED clause, query execution is terminated as soon as possible.
- The effects of terminating the query because of LIMIT ROWS EXAMINED are as follows:
  - The result of the query is a subset of the complete query, depending on when the query engine detected that the limit was reached. The result may be empty if no result rows could be computed before reaching the limit.
  - A warning is generated of the form: “Query execution was interrupted. The query examined at least 100 rows, which exceeds LIMIT ROWS EXAMINED (20). The query result may be incomplete.”
  - If query processing was interrupted during filesort, an error is returned in addition to the warning.
  - If a UNION was interrupted during execution of one of its queries, the last step of the UNION is still executed in order to produce a partial result.
  - Depending on the join and other execution strategies used for a query, the same query may produce no result at all, or a different subset of the complete result when terminated due to LIMIT ROWS EXAMINED.
  - If the query contains a GROUP BY clause, the last group where the limit was reached will be discarded.

The `LIMIT ROWS EXAMINED` clause cannot be specified on a per-subquery basis. There can be only one `LIMIT ROWS EXAMINED` clause for the whole `SELECT` statement. If a `SELECT` statement contains several subqueries with `LIMIT ROWS EXAMINED`, the one that is parsed last is taken into account.

Examples

A simple example of the clause is:

```
SELECT * FROM t1, t2 LIMIT 10 ROWS EXAMINED 10000;
```

The `LIMIT ROWS EXAMINED` clause is global for the whole statement.

If a composite query (such as `UNION`, or query with derived tables or with subqueries) contains more than one `LIMIT ROWS EXAMINED`, the last one parsed is taken into account. In this manner either the last or the outermost one is taken into account. For instance, in the query:

```
SELECT * FROM t1
WHERE c1 IN (SELECT * FROM t2 WHERE c2 > ' ' LIMIT ROWS EXAMINED 0)
LIMIT ROWS EXAMINED 11;
```

The limit that is taken into account is 11, not 0.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB 5.3 Optimizer Debugging

MariaDB 5.3 has an optimizer debugging patch. The patch is pushed into:

lp:maria-captains/maria/5.3-optimizer-debugging

The patch is wrapped in #ifdef, but there is a #define straight in mysql_priv.h so simply compiling that tree should produce a binary with optimizer debugging enabled.

The patch adds two system variables:

- `@@debug_optimizer_prefer_join_prefix`
- `@@debug_optimizer_dupsweedout_penalized`

the variables are present as session/global variables, and are also settable via the server command line.

debug_optimizer_prefer_join_prefix

If this variable is non-NULL, it is assumed to specify a join prefix as a comma-separated list of table aliases:

```
SET debug_optimizer_prefer_join_prefix='tbl1,tbl2,tbl3';
```

The optimizer will try its best to build a join plan which matches the specified join prefix. It does this by comparing join prefixes it is considering with `@@debug_optimizer_prefer_join_prefix`, and multiplying cost by a million if the plan doesn't match the prefix.

As a result, you can more-or-less control the join order. For example, let's take this query:

```
MariaDB [test]> EXPLAIN SELECT * FROM ten A, ten B, ten C;
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra
|
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
-------------+
|  1 | SIMPLE      | A     | ALL  | NULL          | NULL | NULL    | NULL |   10 |
|
|  1 | SIMPLE      | B     | ALL  | NULL          | NULL | NULL    | NULL |   10 | Using join buffer
(flat, BNL join) |
|  1 | SIMPLE      | C     | ALL  | NULL          | NULL | NULL    | NULL |   10 | Using join buffer
(flat, BNL join) |
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
-------------+
3 rows in set (0.00 sec)
```

and request a join order of C,A,B:

```
MariaDB [test]> SET debug_optimizer_prefer_join_prefix='C,A,B';
Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> EXPLAIN SELECT * FROM ten A, ten B, ten C;
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra
|
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
-------------+
|  1 | SIMPLE      | C     | ALL  | NULL          | NULL | NULL    | NULL |   10 |
|
|  1 | SIMPLE      | A     | ALL  | NULL          | NULL | NULL    | NULL |   10 | Using join buffer
(flat, BNL join) |
|  1 | SIMPLE      | B     | ALL  | NULL          | NULL | NULL    | NULL |   10 | Using join buffer
(flat, BNL join) |
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
-------------+
3 rows in set (0.00 sec)
```

We got it.

Note that this is still a *best-effort* approach:

- you won't be successful in forcing join orders which the optimizer considers invalid (e.g. for "t1 LEFT JOIN t2" you won't be able to get a join order of t2,t1).
- The optimizer does various plan pruning and may discard the requested join order before it has a chance to find out that it is a million-times cheaper than any other.

Semi-joins

It is possible to force the join order of joins plus semi-joins. This may cause a different strategy to be used:

```
MariaDB [test]> SET debug_optimizer_prefer_join_prefix=NULL;
Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> EXPLAIN SELECT * FROM ten A WHERE a IN (SELECT B.a FROM ten B, ten C WHERE C.a + A.a <
4);
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
-----+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra
|
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
-----+
|  1 | PRIMARY     | A     | ALL  | NULL          | NULL | NULL    | NULL |   10 |
|
|  1 | PRIMARY     | B     | ALL  | NULL          | NULL | NULL    | NULL |   10 | Using where
|
|  1 | PRIMARY     | C     | ALL  | NULL          | NULL | NULL    | NULL |   10 | Using where;
FirstMatch(A) |
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
-----+
3 rows in set (0.00 sec)

MariaDB [test]> SET debug_optimizer_prefer_join_prefix='C,A,B';
Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> EXPLAIN SELECT * FROM ten A WHERE a IN (SELECT B.a FROM ten B, ten C WHERE C.a + A.a <
4);
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
--------------------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra
|
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
--------------------------+
|  1 | PRIMARY     | C     | ALL  | NULL          | NULL | NULL    | NULL |   10 | Start temporary
|
|  1 | PRIMARY     | A     | ALL  | NULL          | NULL | NULL    | NULL |   10 | Using where; Using
join buffer (flat, BNL join) |
|  1 | PRIMARY     | B     | ALL  | NULL          | NULL | NULL    | NULL |   10 | Using where; End
temporary                      |
+----+-------------+-------+------+---------------+------+---------+------+------+----------------------
--------------------------+
3 rows in set (0.00 sec)
```

Semi-join materialization is a somewhat special case, because “join prefix” is not exactly what you see in the EXPLAIN output. For semi-join materialization:

- don’t put “`<subqueryN>`” into `@@debug_optimizer_prefer_join_prefix`
- instead, put all of the materialization tables into the place where you want the `<subqueryN>` line.
- Attempts to control the join order inside the materialization nest will be unsuccessful. Example: we want A-C-B-AA:

```
MariaDB [test]> SET debug_optimizer_prefer_join_prefix='A,C,B,AA';
Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> EXPLAIN SELECT * FROM ten A, ten AA WHERE A.a IN (SELECT B.a FROM ten B, ten C);
+----+-------------+-------------+--------+---------------+--------------+---------+------+------+-------
------------------------------+
| id | select_type | table       | type   | possible_keys | key          | key_len | ref  | rows | Extra
|
+----+-------------+-------------+--------+---------------+--------------+---------+------+------+-------
------------------------------+
|  1 | PRIMARY     | A           | ALL    | NULL          | NULL         | NULL    | NULL |   10 |
|
|  1 | PRIMARY     | <subquery2> | eq_ref | distinct_key  | distinct_key | 5       | func |    1 |
|
|  1 | PRIMARY     | AA          | ALL    | NULL          | NULL         | NULL    | NULL |   10 | Using
join buffer (flat, BNL join) |
|  2 | SUBQUERY    | B           | ALL    | NULL          | NULL         | NULL    | NULL |   10 |
|
|  2 | SUBQUERY    | C           | ALL    | NULL          | NULL         | NULL    | NULL |   10 |
|
+----+-------------+-------------+--------+---------------+--------------+---------+------+------+-------
------------------------------+
5 rows in set (0.00 sec)
```

but we get A-B-C-AA.

debug_optimizer_dupsweedout_penalized

There are four semi-join execution strategies:

1. `FirstMatch`
2. `Materialization`
3. `LooseScan`
4. `DuplicateWeedout`

The first three strategies have flags in @@optimizer_switch that can be used to disable them. The `DuplicateWeedout` strategy does not have a flag. This was done for a reason, as that strategy is the catch-all strategy and it can handle all kinds of subqueries, in all kinds of join orders. (We're slowly moving to the point where it will be possible to run with `FirstMatch` enabled and everything else disabled but we are not there yet.)

Since `DuplicateWeedout` cannot be disabled, there are cases where it "gets in the way" by being chosen over the strategy you need. This is what `debug_optimizer_dupsweedout_penalized` is for. if you set:

```
MariaDB [test]> SET debug_optimizer_dupsweedout_penalized=TRUE;
```

…the costs of query plans that use `DuplicateWeedout` will be multiplied by a million. This doesn't mean that you will get rid of `DuplicateWeedout` — due to Bug #898747 it is still possible to have `DuplicateWeedout` used even if a cheaper plan exits. A partial remedy to this is to run with

```
MariaDB [test]> SET optimizer_prune_level=0;
```

It is possible to use both `debug_optimizer_dupsweedout_penalized` and `debug_optimizer_prefer_join_prefix` at the same time. This should give you the desired strategy and join order.

Further reading

- See mysql-test/t/debug_optimizer.test (in the MariaDB source code) for examples

This page is licensed: CC BY-SA / Gnu FDL

not_null_range_scan Optimization

The NOT NULL range scan optimization enables the optimizer to construct range scans from NOT NULL conditions that it was able to infer from the WHERE clause.

The optimization appeared in MariaDB 10.5.0. It is not enabled by default; one needs to set an `optimizer_switch` flag to enable it.

Description

A basic (but slightly artificial) example:

```
CREATE TABLE items (
  price  DECIMAL(8,2),
  weight DECIMAL(8,2),
  ...
  INDEX(weight)
);
```

```
-- Find items that cost more than 1000 $currency_units per kg:
SET optimizer_switch='not_null_range_scan=ON';
EXPLAIN
SELECT * FROM items WHERE items.price > items.weight / 1000;
```

The WHERE condition in this form cannot be used for range scans. However, one can infer that it will reject rows that NULL for `weight`. That is, infer an additional condition that

```
weight IS NOT NULL
```

and pass it to the range optimizer. The range optimizer can, in turn, evaluate whether it makes sense to construct range access from the condition:

```
+------+-------------+-------+-------+---------------+--------+---------+------+------+-------------+
| id   | select_type | table | type  | possible_keys | key    | key_len | ref  | rows | Extra       |
+------+-------------+-------+-------+---------------+--------+---------+------+------+-------------+
|    1 | SIMPLE      | items | range | NULL          | weight | 5       | NULL | 1    | Using where |
+------+-------------+-------+-------+---------------+--------+---------+------+------+-------------+
```

Here's another example that's more complex but is based on a real-world query. Consider a join query

```
-- Find orders that were returned
SELECT * FROM current_orders AS O, order_returns AS RET
WHERE
  O.return_id= RET.id;
```

Here, the optimizer can infer the condition "return_id IS NOT NULL". If most of the orders are not returned (and so have NULL for return_id), one can use range access to scan only those orders that had a return.

Controlling the Optimization

The optimization is not enabled by default. One can enable it like so

```
SET optimizer_switch='not_null_range_scan=ON';
```

Optimizer Trace

TODO.

See Also

- MDEV-15777 - JIRA bug report which resulted in the optimization
- NULL Filtering Optimization is a related optimization in MySQL and MariaDB. It uses inferred NOT NULL conditions to perform filtering (but not index access)

This page is licensed: CC BY-SA / Gnu FDL

optimizer_switch

optimizer_switch is a server variable that one can use to enable/disable specific optimizations.

Syntax

To set or unset the various optimizations, use the following syntax:

```
SET [GLOBAL|SESSION] optimizer_switch='cmd[,cmd]...';
```

The *cmd* takes the following format:

Syntax	Description
default	Reset all optimizations to their default values.
optimization_name=default	Set the specified optimization to its default value.
optimization_name=on	Enable the specified optimization.
optimization_name=off	Disable the specified optimization.

There is no need to list all flags - only those that are specified in the command will be affected.

Available Flags

Below is a list of all *optimizer_switch* flags available in MariaDB:

Flag and MariaDB default	Supported in MariaDB since
condition_pushdown_for_derived=on	MariaDB 10.2.2
condition_pushdown_for_subquery=on	MariaDB 10.4
condition_pushdown_from_having=on	MariaDB 10.4.3
cset_narrowing=on/off	MariaDB 10.6.16, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3 and MariaDB 11.2.2
derived_merge=on	MariaDB 5.3
derived_with_keys=on	MariaDB 5.3
default	MariaDB 5.1
duplicateweedout=on	MariaDB 12.0
engine_condition_pushdown=off	MariaDB 5.5 (deprecated in MariaDB 10.1, removed in MariaDB 11.3)
exists_to_in=on	MariaDB 10.0
extended_keys=on	MariaDB 5.5.21
firstmatch=on	MariaDB 5.3
index_condition_pushdown=on	MariaDB 5.3
hash_join_cardinality=off	MariaDB 10.6.13 (MDEV-30812)
index_merge=on	MariaDB 5.1
index_merge_intersection=on	MariaDB 5.1
index_merge_sort_intersection=off	MariaDB 5.3

Flag and MariaDB default	Supported in MariaDB since
index_merge_sort_union=on	MariaDB 5.1
index_merge_union=on#	MariaDB 5.1
in_to_exists=on	MariaDB 5.3
join_cache_bka=on	MariaDB 5.3
join_cache_hashed=on	MariaDB 5.3
join_cache_incremental=on	MariaDB 5.3
loosescan=on	MariaDB 5.3
materialization=on (semi-join, non-semi-join)	MariaDB 5.3
mrr=off	MariaDB 5.3
mrr_cost_based=off	MariaDB 5.3
mrr_sort_keys=off	MariaDB 5.3
not_null_range_scan=off	MariaDB 10.5
optimize_join_buffer_size=on	MariaDB 5.3
orderby_uses_equalities=on	MariaDB 10.1.15
outer_join_with_cache=on	MariaDB 5.3
partial_match_rowid_merge=on	MariaDB 5.3
partial_match_table_scan=on	MariaDB 5.3
reorder_outer_joins	MariaDB 12.3
rowid_filter=on	MariaDB 10.4.3
sargable_casefold=on	MariaDB 11.3.0
semijoin=on	MariaDB 5.3
semijoin_with_cache=on	MariaDB 5.3
split_materialized=on	MariaDB 10.3.4
subquery_cache=on	MariaDB 5.3
table_elimination=on	MariaDB 5.1

Defaults

From version	Default optimizer_switch setting
MariaDB 12.0.1	index_merge=on, index_merge_union=on, index_merge_sort_union=on, index_merge_intersection=on, index_merge_sort_intersection=off, index_condition_pushdown=on, derived_merge=on, derived_with_keys=on, firstmatch=on, loosescan=on, **duplicateweedout=on**, materialization=on, in_to_exists=on, semijoin=on, partial_match_rowid_merge=on, partial_match_table_scan=on, subquery_cache=on, mrr=off, mrr_cost_based=off, mrr_sort_keys=off, outer_join_with_cache=on, semijoin_with_cache=on, join_cache_incremental=on, join_cache_hashed=on, join_cache_bka=on, optimize_join_buffer_size=on, table_elimination=on, extended_keys=on, exists_to_in=on, orderby_uses_equalities=on, condition_pushdown_for_derived=on, split_materialized=on, condition_pushdown_for_subquery=on, rowid_filter=on, condition_pushdown_from_having=on, not_null_range_scan=off, hash_join_cardinality=on, cset_narrowing=on, sargable_casefold=on

From version	Default optimizer_switch setting
MariaDB 11.7.0	index_merge=on, index_merge_union=on, index_merge_sort_union=on, index_merge_intersection=on, index_merge_sort_intersection=off, index_condition_pushdown=on, derived_merge=on, derived_with_keys=on, firstmatch=on, loosescan=on, materialization=on, in_to_exists=on, semijoin=on, partial_match_rowid_merge=on, partial_match_table_scan=on, subquery_cache=on, mrr=off, mrr_cost_based=off, mrr_sort_keys=off, outer_join_with_cache=on, semijoin_with_cache=on, join_cache_incremental=on, join_cache_hashed=on, join_cache_bka=on, optimize_join_buffer_size=on, table_elimination=on, extended_keys=on, exists_to_in=on, orderby_uses_equalities=on, condition_pushdown_for_derived=on, split_materialized=on, condition_pushdown_for_subquery=on, rowid_filter=on, condition_pushdown_from_having=on, not_null_range_scan=off, hash_join_cardinality=on, **cset_narrowing=on**, sargable_casefold=on
MariaDB 11.3.1	index_merge=on, index_merge_union=on, index_merge_sort_union=on, index_merge_intersection=on, index_merge_sort_intersection=off, index_condition_pushdown=on, derived_merge=on, derived_with_keys=on, firstmatch=on, loosescan=on, materialization=on, in_to_exists=on, semijoin=on, partial_match_rowid_merge=on, partial_match_table_scan=on, subquery_cache=on, mrr=off, mrr_cost_based=off, mrr_sort_keys=off, outer_join_with_cache=on, semijoin_with_cache=on, join_cache_incremental=on, join_cache_hashed=on, join_cache_bka=on, optimize_join_buffer_size=on, table_elimination=on, extended_keys=on, exists_to_in=on, orderby_uses_equalities=on, condition_pushdown_for_derived=on, split_materialized=on, condition_pushdown_for_subquery=on, rowid_filter=on, condition_pushdown_from_having=on, not_null_range_scan=off, hash_join_cardinality=on, cset_narrowing=off, **sargable_casefold=on**
MariaDB 10.6.16, MariaDB 10.11.6, MariaDB 11.0.4, MariaDB 11.1.3 and MariaDB 11.2.2	index_merge=on, index_merge_union=on, index_merge_sort_union=on, index_merge_intersection=on, index_merge_sort_intersection=off, index_condition_pushdown=on, derived_merge=on, derived_with_keys=on, firstmatch=on, loosescan=on, materialization=on, in_to_exists=on, semijoin=on, partial_match_rowid_merge=on, partial_match_table_scan=on, subquery_cache=on, mrr=off, mrr_cost_based=off, mrr_sort_keys=off, outer_join_with_cache=on, semijoin_with_cache=on, join_cache_incremental=on, join_cache_hashed=on, join_cache_bka=on, optimize_join_buffer_size=on, table_elimination=on, extended_keys=on, exists_to_in=on, orderby_uses_equalities=on, condition_pushdown_for_derived=on, split_materialized=on, condition_pushdown_for_subquery=on, rowid_filter=on, condition_pushdown_from_having=on, not_null_range_scan=off, hash_join_cardinality=on, **cset_narrowing=off**
MariaDB 11.0.2	index_merge=on, index_merge_union=on, index_merge_sort_union=on, index_merge_intersection=on, index_merge_sort_intersection=off, engine_condition_pushdown=off, index_condition_pushdown=on, derived_merge=on, derived_with_keys=on, firstmatch=on, loosescan=on, materialization=on, in_to_exists=on, semijoin=on, partial_match_rowid_merge=on, partial_match_table_scan=on, subquery_cache=on, mrr=off, mrr_cost_based=off, mrr_sort_keys=off, outer_join_with_cache=on, semijoin_with_cache=on, join_cache_incremental=on, join_cache_hashed=on, join_cache_bka=on, optimize_join_buffer_size=on, table_elimination=on, extended_keys=on, exists_to_in=on, orderby_uses_equalities=on, condition_pushdown_for_derived=on, split_materialized=on, condition_pushdown_for_subquery=on, rowid_filter=on,condition_pushdown_from_having=on, not_null_range_scan=off, **hash_join_cardinality=on**
MariaDB 10.6.13, MariaDB 10.11.3	index_merge=on, index_merge_union=on, index_merge_sort_union=on, index_merge_intersection=on, index_merge_sort_intersection=off, engine_condition_pushdown=off, index_condition_pushdown=on, derived_merge=on, derived_with_keys=on, firstmatch=on, loosescan=on, materialization=on, in_to_exists=on, semijoin=on, partial_match_rowid_merge=on, partial_match_table_scan=on, subquery_cache=on, mrr=off, mrr_cost_based=off, mrr_sort_keys=off, outer_join_with_cache=on, semijoin_with_cache=on, join_cache_incremental=on, join_cache_hashed=on, join_cache_bka=on, optimize_join_buffer_size=on, table_elimination=on, extended_keys=on, exists_to_in=on, orderby_uses_equalities=on, condition_pushdown_for_derived=on, split_materialized=on, condition_pushdown_for_subquery=on, rowid_filter=on,condition_pushdown_from_having=on, not_null_range_scan=off, **hash_join_cardinality=off**

See Also

- Quickly finding optimizer_switch values that are on or off
- The optimizer converts certain big IN predicates into IN subqueries
- optimizer_adjust_secondary_key_cost
- Optimizer hints in SELECT

This page is licensed: CC BY-SA / Gnu FDL

optimizer_adjust_secondary_key_costs

`optimizer_adjust_secondary_key_costs`

- Description: Gives the user the ability to affect how the costs for secondary keys using `ref` are calculated in the few cases when MariaDB 10.6 up to MariaDB 10.11 makes a sub-optimal choice when optimizing `ref` access, either for key lookups or `GROUP BY`. `ref`, as used by EXPLAIN, means that the optimizer is using key-lookup on one value to find the matching rows from a table. Unused from MariaDB 11.0. In MariaDB 10.6.18 the variable was changed from a number to a set of strings and `disable_forced_index_in_group_by` (value 4) was added.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `set`
- Default Value: `fix_reuse_range_for_ref`, `fix_card_multiplier`
- Range: `0` to `63` or any combination of `adjust_secondary_key_cost`, `disable_max_seek` or `disable_forced_index_in_group_by`, `fix_innodb_cardinality`, `fix_reuse_range_for_ref`, `fix_card_multiplier`
- Introduced: MariaDB 10.6.17, MariaDB 10.11.7

```
MariaDB [securedb]> select @@version;
+----------------------------------+
| @@version             |
+----------------------------------+
| 10.6.20-16-MariaDB-enterprise-log |
+----------------------------------+
1 row in set (0.001 sec)

MariaDB [securedb]> select @@optimizer_adjust_secondary_key_costs;
+-------------------------------------------+
| @@optimizer_adjust_secondary_key_costs   |
+-------------------------------------------+
| fix_reuse_range_for_ref,fix_card_multiplier |
+-------------------------------------------+
```

MariaDB starting with 11.0

`optimizer_adjust_secondary_key_costs` will be obsolete starting from MariaDB 11.0 as the new optimizer in 11.0 does not have max_seek optimization and is already using cost based choices for index usage with GROUP BY.

The value for `optimizer_adjust_secondary_key_costs` is one of more of the following:

Value	Version added	Old behavior	Change when variable is used
adjust_secondary_key_cost	10.6.17	Limit ref costs by max_seeks	The secondary key costs for ref are updated to be at least five times the clustered primary key costs if a clustered primary key exists
disable_max_seek	10.6.17	ref cost on secondary keys is limited to max_seek = min('number of expected rows'/ 10, scan_time*3)	Disable 'max_seek' optimization and do a slight adjustment of filter cost
disable_forced_index_in_group_by	10.6.18	Use a rule-based choice when deciding to use an index to resolve GROUP BY	The choice is now cost based

Value	Version added	Old behavior	Change when variable is used
fix_innodb_cardinality	10.6.19	By default InnoDB doubles the cardinality for indexes in an effort to force index usage over table scans. This can cause the optimizer to create sub-optimal plans for ranges or index entries that cover a big part of the table.	Using this option removes the doubling of cardinality in InnoDB. fix_innodb_cardinality is recommended to be used only as a server startup option, as it is enabled for a table at first usage. See MDEV-34664 for details.
fix_reuse_range_for_ref	10.6.20	Number of estimated rows for 'ref' did not always match costs from range optimizer	Use cost from range optimizer for 'ref' if all used key parts are constants. The old code did not always do this
fix_card_multiplier	10.6.20	Index selectivity can be bigger than 1.0 if index statistics is not up to date. Not on by default.	Ensure that the calculated index selectivity is never bigger than 1.0. Having index selectivity bigger than 1.0 causes MariaDB to believe that there is more rows in the table than in reality, which can cause wrong plans. This option is on by default.

One can set all options with:

```
SET @@optimizer_adjust_secondary_key_costs='all';
```

Explanations of the old behavior in MariaDB 10.x

The reason for the max_seek optimization was originally to ensure that MariaDB would use a key instead of a table scan. This works well for a lot of queries, but can cause problems when a table scan is a better choice, such as when one would have to scan more than 1/4 of the rows in the table (in which case a table scan is better).

See Also

- The optimizer_switch system variable.

This page is licensed: CC BY-SA / Gnu FDL

optimizer_join_limit_pref_ratio Optimization

Basics

Off (0) by default, when enabling this optimization, MariaDB will consider a join order that may shorten query execution time based on the `ORDER BY ... LIMIT n` clause. For small values of `n`, this may improve performance.

Set the value of `optimizer_join_limit_pref_ratio` to a non-zero value to enable this option (higher values are more conservative, recommended value is 100), or set to 0 (the default value) to disable it.

Detailed description

Problem setting

By default, the MariaDB optimizer picks a join order without considering the ORDER BY … LIMIT clause, when present.

For example, consider a query looking at latest 10 orders together with customers who made them:

```
SELECT *
FROM
  customer,ORDER
WHERE
  customer.name=ORDER.customer_name
ORDER BY
  ORDER.DATE DESC
LIMIT 10
```

The two possible plans are: `customer->orders`:

```
+------+-------------+----------+------+---------------+---------------+---------+---------------+------
+---------------------------------------------+
| id   | select_type | table    | type | possible_keys | key           | key_len | ref           | rows |
Extra                                       |
+------+-------------+----------+------+---------------+---------------+---------+---------------+------
+---------------------------------------------+
|    1 | SIMPLE      | customer | ALL  | name          | NULL          | NULL    | NULL          | 9623 |
Using where; Using temporary; Using filesort |
|    1 | SIMPLE      | orders   | ref  | customer_name | customer_name | 103     | customer.name | 1    |
|
+------+-------------+----------+------+---------------+---------------+---------+---------------+------
+---------------------------------------------+
```

and `orders->customer`:

```
+------+-------------+----------+-------+---------------+------------+---------+----------------------+--
----+-------------+
| id   | select_type | table    | type  | possible_keys | key        | key_len | ref                  |
rows | Extra       |
+------+-------------+----------+-------+---------------+------------+---------+----------------------+--
----+-------------+
|    1 | SIMPLE      | orders   | index | customer_name | order_date | 4       | NULL                 |
10   | Using where |
|    1 | SIMPLE      | customer | ref   | name          | name       | 103     | orders.customer_name | 1
|             |
+------+-------------+----------+-------+---------------+------------+---------+----------------------+--
----+-------------+
```

The `customer->orders` plan computes a join between all customers and orders, saves that result into a temporary table, and then uses filesort to get the 10 most recent orders. This query plan doesn't benefit from the fact that just 10 orders are needed.

However, in contrast, the `orders->customers` plan uses an index to read rows in the ORDER BY order. The query can stop execution once it finds 10 `order-and-customer` combinations. This is much faster than computing the entire join. Under this plan, and when this new optimization, we can leverage ORDER BY … LIMIT to stop early, when we have the 10 combinations.

Plans with LIMIT shortcuts are difficult to estimate

It is fundamentally difficult to produce a reliable estimate for ORDER BY … LIMIT shortcuts. Let's take an example from the previous section to see why. This query searches for last 10 orders that were shipped by air:

```
SELECT *
FROM
  customer,ORDER
WHERE
  customer.name=ORDER.customer_name
  AND ORDER.shipping_method='Airplane'
ORDER BY
  ORDER.DATE DESC
LIMIT 10
```

Suppose we know beforehand that 50% of orders are shipped by air.
Assuming there's no correlation between date and shipping method, `orders->customer` plan will need to scan 20 orders before we find 10 that are shipped by air.
But if there is correlation, then we may need to scan up to `(total_orders*0.5 + 10)` before we find first 10 orders that are shipped by air. Scanning about 50% of all orders can be expensive.

This situation worsens when the query has constructs whose selectivity is not known. For example, suppose the WHERE condition was

```
order.shipping_method='%Airplane%'
```

in this case, we can't reliably say whether we will be able to stop after scanning #LIMIT rows or we will need to enumerate all rows before we find #LIMIT matches.

Providing guidelines to the optimizer

Due to these challenges, the optimization is not enabled by default.

When running a mostly OLTP workload such that query WHERE conditions have suitable indexes or are not very selective, then any ORDER BY … LIMIT queries will typically find matching rows quickly. In this case, it makes sense to give the following guidance to the optimizer:

```
Do consider the query plan using LIMIT short-cutting
and prefer it if it promises at least X times speedup.
```

The value of `X` is given to the optimizer via `optimizer_join_limit_pref_ratio` setting.
Higher values carry less risk. The recommended value is 100: prefer the LIMIT join order if it promises at least 100x speedup.

References

- MDEV-34720 introduces optimizer_join_limit_pref_ratio optimization
- MDEV-18079 is about future development that would make the optimizer handle such cases without user guidance.

This page is licensed: CC BY-SA / Gnu FDL

Optimizing for "Latest News"-style Queries

The problem space

Let's say you have "news articles" (rows in a table) and want a web page showing the latest ten articles about a particular topic.

Variants on "topic":

- Category
- Tag
- Provider (of news article)
- Manufacturer (of item for sale)
- Ticker (financial stock)

Variants on "news article"

- Item for sale
- Blog comment
- Blog thread

Variants on “latest”

- Publication date (unix_timestamp)
- Most popular (keep the count)
- Most emailed (keep the count)
- Manual ranking (1..10 – ‘top ten’)

Variants on “10” - there is nothing sacred about “10” in this discussion.

The performance issues

Currently you have a table (or a column) that relates the topic to the article. The SELECT statement to find the latest 10 articles has grown in complexity, and performance is poor. You have focused on what index to add, but nothing seems to work.

- If there are multiple topics for each article, you need a many-to-many table.
- You have a flag “is_deleted” that needs filtering on.
- You want to “paginate” the list (ten articles per page, for as many pages as necessary).

The solution

First, let me give you the solution, then I will elaborate on why it works well.

- One new table called, say, Lists.
- Lists has *exactly* 3 columns: topic, article_id, sequence
- Lists has *exactly* 2 indexes: PRIMARY KEY(topic, sequence, article_id), INDEX(article_id)
- Only viewable articles are in Lists. (This avoids the filtering on “is_deleted”, etc)
- Lists is InnoDB. (This gets “clustering”.)
- “sequence” is typically the date of the article, but could be some other ordering.
- “topic” should probably be normalized, but that is not critical to this discussion.
- “article_id” is a link to the bulky row in another table(s) that provide all the details about the article.

The queries

Find the latest 10 articles for a topic:

```
SELECT  a.*
    FROM  Articles a
    JOIN  Lists s ON s.article_id = a.article_id
    WHERE  s.topic = ?
    ORDER BY  s.sequence DESC
    LIMIT  10;
```

You must *not* have any WHERE condition touching columns in Articles.

When you mark an article for deletion; you *must* remove it from Lists:

```
DELETE  FROM  Lists
    WHERE  article_id = ?;
```

I emphasize “must” because flags and other filtering is often the root of performance issues.

Why it works

By now, you may have discovered why it works.

The big goal is to minimize the disk hits. Let’s itemize how few disk hits are needed. When finding the latest articles with ‘normal’ code, you will probably find that it is doing significant scans of the Articles table, failing to quickly home

in on the 10 rows you want. With this design, there is only one extra disk hit:

- 1 disk hit: 10 adjacent, narrow, rows in Lists – probably in a single “block”.
- 10 disk hits: The 10 articles. (These hits are unavoidable, but may be cached.)
  The PRIMARY KEY, and using InnoDB, makes these quite efficient.

OK, you pay for this by removing things that you should avoid.

- 1 disk hit: INDEX(article_id) - finding a few ids
- A few more disk hits to DELETE rows from Lists.
  This is a small price to pay – and you are not paying it while the user is waiting for the page to render.

See also

Rick James graciously allowed us to use this article in the documentation.

Rick James’ site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: lists

This page is licensed: CC BY-SA / Gnu FDL

Pagination Optimization

The Desire

You have a website with news articles, or a blog, or some other thing with a list of things that might be too long for a single page. So, you decide to break it into chunks of, say, 10 items and provide a [Next] button to go the next “page”.

You spot OFFSET and LIMIT in MariaDB and decide that is the obvious way to do it.

```
SELECT  *
        FROM  items
        WHERE  messy_filtering
        ORDER BY  date DESC
        OFFSET  $M  LIMIT $N
```

Note that the problem requirement needs a [Next] link on each page so that the user can ‘page’ through the data. He does not really need “GoTo Page #”. Jump to the [First] or [Last] page may be useful.

The Problem

All is well – until you have 50,000 items in a list. And someone tries to walk through all 5000 pages. That ‘someone’ could be a search engine crawler.

Where’s the problem? Performance. Your web page is doing “SELECT … OFFSET 49990 LIMIT 10” (or the equivalent “LIMIT 49990,10”). MariaDB has to find all 50,000 rows, step over the first 49,990, then deliver the 10 for that distant page.

If it is a crawler (‘spider’) that read all the pages, then it actually touched about 125,000,000 items to read all 5,000 pages.

Reading the entire table, just to get a distant page, can be so much I/O that it can cause timeouts on the web page. Or it can interfere with other activity, causing other things to be slow.

Other Bugs

In addition to a performance problem, …

- If an item is inserted or deleted between the time you look at one page and the next, you could miss an item, or see an item duplicated.
- The pages are not easily bookmarked or sent to someone else because the contents shift over time.
- The WHERE clause and the ORDER BY may even make it so that all 50,000 items have to be read, just to find the 10 items for page 1!

What to Do?

Hardware? No, that's just a bandaid. The data will continue to grow and even the new hardware won't handle it.

Better INDEX? No. You must get away from reading the entire table to get the 5000th page.

Build another table saying where the pages start? Get real! That would be a maintenance nightmare, and expensive.

Bottom line: Don't use OFFSET; instead remember where you "left off".

```
# First page (latest 10 items):
    SELECT ... WHERE ... ORDER BY id DESC LIMIT 10
# Next page (second 10):
    SELECT ... WHERE ... AND id < $left_off ORDER BY id DESC LIMIT 10
```

With INDEX(id), this suddenly becomes very efficient.

Implementation – Getting Rid of OFFSET

You are probably doing this now: ORDER BY datetime DESC LIMIT 49990,10 You probably have some unique id on the table. This can probably be used for "left off".

Currently, the [Next] button probably has a url something like ?topic=xyz&page=4999&limit=10 The 'topic' (or 'tag' or 'provider' or 'user' or etc) says which set of items are being displayed. The product of page*limit gives the OFFSET. (The "limit=10" might be in the url, or might be hard-coded; this choice is not relevant to this discussion.)

The new variant would be ?topic=xyz&id=12345&limit=10. (Note: the 12345 is not computable from 4999.) By using INDEX(topic, id) you can efficiently say

```
WHERE topic = 'xyz'
      AND id >= 1234
    ORDER BY id
    LIMIT 10
```

That will hit only 10 rows. This is a huge improvement for later pages. Now for more details.

Implementation – "Left Off"

What if there are exactly 10 rows left when you display the current page? It would make the UI nice if you grayed out the [Next] button, wouldn't it. (Or you could suppress the button all together.)

How to do that? Instead of LIMIT 10, use LIMIT 11. That will give you the 10 items needed for the current page, plus an indication of whether there is another page. And the id for that page.

So, take the 11th id for the [Next] button: <a href=?topic=xyz&id=$id11&limit=10>Next

Implementation – Links Beyond [Next]

Let's extend the 11 trick to also find the next 5 pages and build links for them.

Plan A is to say LIMIT 51. If you are on page 12, that would give you links for pages 13 (using 11th id) through pages 17 (51st).

Plan B is to do two queries, one to get the 10 items for the current page, the other to get the next 41 ids (LIMIT 10, 41) for the next 5 pages.

Which plan to pick? It depends on many things, so benchmark.

A Reasonable Set of Links

Reaching forward and backward by 5 pages is not too much work. It would take two separate queries to find the ids in both directions. Also, having links that take you to the First and Last pages would be easy to do. No id is needed; they can be something like

```
<a href=?topic=xyz&id=FIRST&limit=10>First</a>
    <a href=?topic=xyz&id=LAST&limit=10>Last</a>
```

The UI would recognize those, then generate a SELECT with something like

```
WHERE topic = 'xyz'
    ORDER BY id ASC -- ASC for First; DESC for Last
    LIMIT 10
```

The last items would be delivered in reverse order. Either deal with that in the UI, or make the SELECT more complex:

```
( SELECT ...
        WHERE topic = 'xyz'
        ORDER BY id DESC
        LIMIT 10
    ) ORDER BY id ASC
```

Let's say you are on page 12 of lots of pages. It could show these links:

```
[First] ... [7] [8] [9] [10] [11] 12 [13] [14] [15] [16] [17] ... [Last]
```

where the ellipsis is really used. Some end cases:

```
# Page one of three:
    First [2] [3]
# Page one of many:
    First [2] [3] [4] [5] ... [Last]
#### Page two of many:
    [First] 2 [3] [4] [5] ... [Last]
#### If you jump to the Last page, you don't know what page number it is.
#### So, the best you can do is perhaps:
####    [First] ... [Prev] Last
```

Why it Works

The goal is to touch only the relevant rows, not all the rows leading up to the desired rows. This is nicely achieved, except for building links to the "next 5 pages". That may (or may not) be efficiently resolved by the simple SELECT id, discussed above. The reason that may not be efficient deals with the WHERE clause.

Let's discuss the optimal and suboptimal indexes.

For this discussion, I am assuming

- The datetime field might have duplicates – this can cause troubles
- The id field is unique
- The id field is close enough to datetime-ordered to be used instead of datetime.

Very efficient – it does all the work in the index:

```
INDEX(topic, id)
    WHERE topic = 'xyz'
      AND id >= 876
    ORDER BY id ASC
    LIMIT 10,41
<</code??
That will hit 51 consecutive index entries, 0 data rows.

Inefficient -- it must reach into the data:
<<code>>
    INDEX(topic, id)
    WHERE topic = 'xyz'
      AND id >= 876
      AND is_deleted = 0
    ORDER BY id ASC
    LIMIT 10,41
```

That will hit at least 51 consecutive index entries, plus at least 51 *randomly* located data rows.

Efficient – back to the previous degree of efficiency:

```
INDEX(topic, is_deleted, id)
    WHERE topic = 'xyz'
      AND id >= 876
      AND is_deleted = 0
    ORDER BY id ASC
    LIMIT 10,41
```

Note how all the '=' parts of the WHERE come first; then comes both the '>=' and 'ORDER BY', both on id. This means that the INDEX can be used for all the WHERE, plus the ORDER BY.

"Items 11-20 Out of 12345"

You lose the "out of" except when the count is small. Instead, say something like

```
Items 11-20 out of Many
```

Alternatively… Only a few searches will have too many items to count. Keep another table with the search criteria and a count. This count can be computed daily (or hourly) by some background script. When discovering that the topic is a busy one, look it up in the table to get

```
Items 11-20 out of about 49,000
```

The background script would round the count off.

The quick way to get an *estimated* number of rows for an InnoDB table is

```
SELECT  table_rows
        FROM  information_schema.TABLES
        WHERE  TABLE_SCHEMA = 'database_name'
          AND  TABLE_NAME = 'table_name'
```

However, it does not allow for the WHERE clause that you probably have.

Complex WHERE, or JOIN

If the search criteria cannot be confined to an INDEX in a single table, this technique is doomed. I have another paper that discusses "Lists", which solves that (which extra development work), and even improves on what is discussed here.

How Much Faster?

This depends on

- How many rows (total)
- Whether the WHERE clause prevented the efficient use of the ORDER BY
- Whether the data is bigger than the cache.
  This last one kicks in when building one page requires reading more data from disk can be cached. At that point, the problem goes from being CPU-bound to being I/O-bound. This is likely to suddenly slow down the loading of a pages by a factor of 10.

What is Lost

- Cannot “jump to Page N”, for an arbitrary N. Why do you want to do that?
- Walking backward from the end does not know the page numbers.
- The code is more complex.

Postlog

Designed about 2007; posted 2012.

See Also

- A forum discussion
- Luke calls it “seek method” or “keyset pagination”
- More by Luke

Rick James graciously allowed us to use this article in the documentation.

Rick James’ site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: pagination

This page is licensed: CC BY-SA / Gnu FDL

Pivoting in MariaDB

The problem

You want to “pivot” the data so that a linear list of values with two keys becomes a spreadsheet-like array. See examples, below.

A solution

The best solution is probably to do it in some form of client code (PHP, etc). MySQL and MariaDB do not have a syntax for SELECT that will do the work for you. The code provided here uses a stored procedure to generate code to pivot the data, and then runs the code.

You can edit the SQL generated by the stored procedure to tweak the output in a variety of ways. Or you can tweak the stored procedure to generate what you would prefer.

Reference code for solution

‘Source’ this into the mysql commandline tool:

```
DELIMITER //
DROP   PROCEDURE IF EXISTS Pivot //
CREATE PROCEDURE Pivot(
    IN tbl_name VARCHAR(99),       -- table name (or db.tbl)
    IN base_cols VARCHAR(99),      -- column(s) on the left, separated by commas
    IN pivot_col VARCHAR(64),      -- name of column to put across the top
    IN tally_col VARCHAR(64),      -- name of column to SUM up
    IN where_clause VARCHAR(99),   -- empty string or "WHERE ..."
    IN order_by VARCHAR(99)        -- empty string or "ORDER BY ..."; usually the base_cols
    )
    DETERMINISTIC
    SQL SECURITY INVOKER
BEGIN
    -- Find the distinct values
    -- Build the SUM()s
    SET @subq = CONCAT('SELECT DISTINCT ', pivot_col, ' AS val ',
                    ' FROM ', tbl_name, ' ', where_clause, ' ORDER BY 1');
    -- select @subq;

    SET @cc1 = "CONCAT('SUM(IF(&p = ', &v, ', &t, 0)) AS ', &v)";
    SET @cc2 = REPLACE(@cc1, '&p', pivot_col);
    SET @cc3 = REPLACE(@cc2, '&t', tally_col);
    -- select @cc2, @cc3;
    SET @qval = CONCAT("'\"', val, '\"'");
    -- select @qval;
    SET @cc4 = REPLACE(@cc3, '&v', @qval);
    -- select @cc4;

    SET SESSION group_concat_max_len = 10000;   -- just in case
    SET @stmt = CONCAT(
            'SELECT  GROUP_CONCAT(', @cc4, ' SEPARATOR ",\n")  INTO @sums',
            ' FROM ( ', @subq, ' ) AS top');
     select @stmt;
    PREPARE _sql FROM @stmt;
    EXECUTE _sql;                       -- Intermediate step: build SQL for columns
    DEALLOCATE PREPARE _sql;
    -- Construct the query and perform it
    SET @stmt2 = CONCAT(
            'SELECT ',
                base_cols, ',\n',
                @sums,
                ',\n SUM(', tally_col, ') AS Total'
            '\n FROM ', tbl_name, ' ',
            where_clause,
            ' GROUP BY ', base_cols,
            '\n WITH ROLLUP',
            '\n', order_by
        );
    select @stmt2;                     -- The statement that generates the result
    PREPARE _sql FROM @stmt2;
    EXECUTE _sql;                      -- The resulting pivot table output
    DEALLOCATE PREPARE _sql;
    -- For debugging / tweaking, SELECT the various @variables after CALLing.
END;
//
DELIMITER ;
```

Then do a CALL, like in the examples, below.

Variants

I thought about having several extra options for variations, but decided that would be too messy. Instead, here are instructions for implementing the variations, either by capturing the SELECT that was output by the Stored Procedure, or by modifying the SP, itself.

- The data is strings (not numeric) – Remove “SUM” (but keep the expression); remove the SUM…AS TOTAL line.

- If you want blank output instead of 0 – Currently the code says “SUM(IF(… 0))”; change the 0 to NULL, then wrap the SUM: IFNULL(SUM(…), ‘‘). Note that this will distinguish between a zero total (showing ’0’) and no data (blank).
- Fancier output – Use PHP/VB/Java/etc.
- No Totals at the bottom – Remove the WITH ROLLUP line from the SELECT.
- No Total for each row – Remove the SUM…AS Total line from the SELECT.
- Change the order of the columns – Modify the ORDER BY 1 (‘1’ meaning first column) in the SELECT DISTINCT in the SP.
- Example: ORDER BY FIND_IN_SET(DAYOFWEEK(…), ‘Sun,Mon,Tue,Wed,Thu,Fri,Sat’)

Notes about “base_cols”:

- Multiple columns on the left, such as an ID and its meaning – This is already handled by allowing base_cols to be a commalist like ‘id, meaning’
- You cannot call the SP with “foo AS ‘blah’” in hopes of changing the labels, but you could edit the SELECT to achieve that goal.

Notes about the “Totals”:

- If “base_cols” is more than one column, WITH ROLLUP will be subtotals as well as a grand total.
- NULL shows up in the Totals row in the “base_cols” column; this can be changed via something like IFNULL(…, ‘Totals’).

Example 1 - Population vs Latitude in US

```
-- Sample input:
+-------+----------------------+---------+------------+
| state | city                 | lat     | population |
+-------+----------------------+---------+------------+
| AK    | Anchorage            | 61.2181 |     276263 |
| AK    | Juneau               | 58.3019 |      31796 |
| WA    | Monroe               | 47.8556 |      15554 |
| WA    | Spanaway             | 47.1042 |      25045 |
| PR    | Arecibo              | 18.4744 |      49189 |
| MT    | Kalispell            | 48.1958 |      18018 |
| AL    | Anniston             | 33.6597 |      23423 |
| AL    | Scottsboro           | 34.6722 |      14737 |
| HI    | Kaneohe              | 21.4181 |      35424 |
| PR    | Candelaria           | 18.4061 |      17632 |
...

-- Call the Stored Procedure:
CALL Pivot('World.US', 'state', '5*FLOOR(lat/5)', 'population', '', '');

-- SQL generated by the SP:
SELECT state,
SUM(IF(5*FLOOR(lat/5) = "15", population, 0)) AS "15",
SUM(IF(5*FLOOR(lat/5) = "20", population, 0)) AS "20",
SUM(IF(5*FLOOR(lat/5) = "25", population, 0)) AS "25",
SUM(IF(5*FLOOR(lat/5) = "30", population, 0)) AS "30",
SUM(IF(5*FLOOR(lat/5) = "35", population, 0)) AS "35",
SUM(IF(5*FLOOR(lat/5) = "40", population, 0)) AS "40",
SUM(IF(5*FLOOR(lat/5) = "45", population, 0)) AS "45",
SUM(IF(5*FLOOR(lat/5) = "55", population, 0)) AS "55",
SUM(IF(5*FLOOR(lat/5) = "60", population, 0)) AS "60",
SUM(IF(5*FLOOR(lat/5) = "70", population, 0)) AS "70",
 SUM(population) AS Total
 FROM World.US  GROUP BY state
 WITH ROLLUP

-- Output from that SQL (also comes out of the SP):
+-------+---------+--------+----------+----------+----------+----------+---------+-------+--------+------
+-----------+
| state | 15      | 20     | 25       | 30       | 35       | 40       | 45      | 55    | 60     | 70
| Total     |
+-------+---------+--------+----------+----------+----------+----------+---------+-------+--------+------
+-----------+
| AK    |       0 |      0 |        0 |        0 |        0 |        0 |       0 | 60607 | 360765 | 4336
|    425708 |
| AL    |       0 |      0 |        0 |  1995225 |        0 |        0 |       0 |     0 |      0 |    0
|   1995225 |
| AR    |       0 |      0 |        0 |   595537 |   617361 |        0 |       0 |     0 |      0 |    0
|   1212898 |
| AZ    |       0 |      0 |        0 |  4708346 |   129989 |        0 |       0 |     0 |      0 |    0
|   4838335 |
...
| FL    |       0 |  34706 |  9096223 |  1440916 |        0 |        0 |       0 |     0 |      0 |    0
|  10571845 |
| GA    |       0 |      0 |        0 |  2823939 |        0 |        0 |       0 |     0 |      0 |    0
|   2823939 |
| HI    |   43050 | 752983 |        0 |        0 |        0 |        0 |       0 |     0 |      0 |    0
|    796033 |
...
| WY    |       0 |      0 |        0 |        0 |        0 |   277480 |       0 |     0 |      0 |    0
|    277480 |
| NULL  | 1792991 | 787689 | 16227033 | 44213344 | 47460670 | 61110822 | 7105143 | 60607 | 360765 | 4336
| 179123400 |
+-------+---------+--------+----------+----------+----------+----------+---------+-------+--------+------
+-----------+
```

Notice how Alaska (AK) has populations in high latitudes and Hawaii (HI) in low latitudes.

Example 2 - Home Solar Power Generation

This give the power (KWh) generated by hour and month for 2012.

```
-- Sample input:
+---------------------+------+
| ts                  | enwh |
+---------------------+------+
| 2012-06-06 11:00:00 |  523 |
| 2012-06-06 11:05:00 |  526 |
| 2012-06-06 11:10:00 |  529 |
| 2012-06-06 11:15:00 |  533 |
| 2012-06-06 11:20:00 |  537 |
| 2012-06-06 11:25:00 |  540 |
| 2012-06-06 11:30:00 |  542 |
| 2012-06-06 11:35:00 |  543 |
Note that it is a reading in watts for each 5 minutes.
So, summing is needed to get the breakdown by month and hour.

-- Invoke the SP:
CALL Pivot('details',    -- Table
           'MONTH(ts)',  -- `base_cols`, to put on left; SUM up over the month
           'HOUR(ts)',   -- `pivot_col` to put across the top; SUM up entries across the hour
           'enwh/1000',  -- The data -- watts converted to KWh
           "WHERE ts >= '2012-01-01' AND ts < '2012-01-01' + INTERVAL 1 year",  -- Limit to one year
           '');          -- assumes that the months stay in order

-- The SQL generated:
SELECT MONTH(ts),
SUM(IF(HOUR(ts) = "5", enwh/1000, 0)) AS "5",
SUM(IF(HOUR(ts) = "6", enwh/1000, 0)) AS "6",
SUM(IF(HOUR(ts) = "7", enwh/1000, 0)) AS "7",
SUM(IF(HOUR(ts) = "8", enwh/1000, 0)) AS "8",
SUM(IF(HOUR(ts) = "9", enwh/1000, 0)) AS "9",
SUM(IF(HOUR(ts) = "10", enwh/1000, 0)) AS "10",
SUM(IF(HOUR(ts) = "11", enwh/1000, 0)) AS "11",
SUM(IF(HOUR(ts) = "12", enwh/1000, 0)) AS "12",
SUM(IF(HOUR(ts) = "13", enwh/1000, 0)) AS "13",
SUM(IF(HOUR(ts) = "14", enwh/1000, 0)) AS "14",
SUM(IF(HOUR(ts) = "15", enwh/1000, 0)) AS "15",
SUM(IF(HOUR(ts) = "16", enwh/1000, 0)) AS "16",
SUM(IF(HOUR(ts) = "17", enwh/1000, 0)) AS "17",
SUM(IF(HOUR(ts) = "18", enwh/1000, 0)) AS "18",
SUM(IF(HOUR(ts) = "19", enwh/1000, 0)) AS "19",
SUM(IF(HOUR(ts) = "20", enwh/1000, 0)) AS "20",
 SUM(enwh/1000) AS Total
 FROM details WHERE ts >= '2012-01-01' AND ts < '2012-01-01' + INTERVAL 1 year GROUP BY MONTH(ts)
 WITH ROLLUP

-- That generated decimal places that I did like:
| MONTH(ts) | 5      | 6       | 7        | 8        | 9         | 10        | 11        | 12        | 13
| 14
     | 15        | 16       | 17       | 18       | 19      | 20     | Total      |
+-----------+--------+---------+----------+----------+-----------+-----------+-----------+-----------+---
--------+------
-----+-----------+----------+----------+----------+---------+--------+------------+
|         1 | 0.0000 |  0.0000 |   1.8510 |  21.1620 |   52.3190 |   73.0420 |   89.3220 |   97.0190 |
88.9720 |   75.
4970 |   50.9270 |  12.5130 |   0.5990 |   0.0000 |  0.0000 | 0.0000 |   563.2230 |
|         2 | 0.0000 |  0.0460 |   5.9560 |  35.6330 |   72.4710 |   96.5130 |  112.7770 |  126.0850 |
117.1540 |   96.
7160 |   72.5900 |  33.6230 |   4.7650 |   0.0040 |  0.0000 | 0.0000 |   774.3330 |
```

Other variations made the math go wrong. (Note that there is no CAST to FLOAT.)

While I was at it, I gave an alias to change “MONTH(ts)” to just “Month”.

So, I edited the SQL to this and ran it:

```
SELECT MONTH(ts) AS 'Month',
ROUND(SUM(IF(HOUR(ts) = "5", enwh, 0))/1000) AS "5",
...
ROUND(SUM(IF(HOUR(ts) = "20", enwh, 0))/1000) AS "20",
 ROUND(SUM(enwh)/1000) AS Total
 FROM details WHERE ts >= '2012-01-01' AND ts < '2012-01-01' + INTERVAL 1 YEAR
 GROUP BY MONTH(ts)
 WITH ROLLUP;
```

-- Which gave cleaner output:

```
+-------+------+------+------+------+------+------+------+------+------+------+------+------+-----
-+------+------+-------+
| Month | 5    | 6    | 7    | 8    | 9    | 10   | 11   | 12   | 13   | 14   | 15   | 16   | 17   | 18
| 19   | 20   | Total |
+-------+------+------+------+------+------+------+------+------+------+------+------+------+------+-----
-+------+------+-------+
|     1 |    0 |    0 |    2 |   21 |   52 |   73 |   89 |   97 |   89 |   75 |   51 |   13 |    1 |    0
|    0 |    0 |   563 |
|     2 |    0 |    0 |    6 |   36 |   72 |   97 |  113 |  126 |  117 |   97 |   73 |   34 |    5 |    0
|    0 |    0 |   774 |
|     3 |    0 |    0 |    9 |   46 |   75 |  105 |  121 |  122 |  128 |  126 |  105 |   71 |   33 |   10
|    0 |    0 |   952 |
|     4 |    0 |    1 |   14 |   63 |  111 |  146 |  171 |  179 |  177 |  158 |  141 |  105 |   65 |   26
|    3 |    0 |  1360 |
|     5 |    0 |    4 |   21 |   78 |  128 |  162 |  185 |  199 |  196 |  187 |  166 |  130 |   81 |   36
|    8 |    0 |  1581 |
|     6 |    0 |    4 |   17 |   71 |  132 |  163 |  182 |  191 |  193 |  182 |  161 |  132 |   89 |   43
|   10 |    1 |  1572 |
|     7 |    0 |    3 |   17 |   57 |  121 |  160 |  185 |  197 |  199 |  189 |  168 |  137 |   92 |   44
|   11 |    1 |  1581 |
|     8 |    0 |    1 |   11 |   48 |  104 |  149 |  171 |  183 |  187 |  179 |  156 |  121 |   76 |   32
|    5 |    0 |  1421 |
|     9 |    0 |    0 |    6 |   32 |   77 |  127 |  151 |  160 |  159 |  148 |  124 |   93 |   47 |   12
|    1 |    0 |  1137 |
|    10 |    0 |    0 |    1 |   16 |   54 |   85 |  107 |  115 |  119 |  106 |   85 |   56 |   17 |    2
|    0 |    0 |   763 |
|    11 |    0 |    0 |    5 |   30 |   57 |   70 |   84 |   83 |   76 |   64 |   35 |    8 |    1 |    0
|    0 |    0 |   512 |
|    12 |    0 |    0 |    2 |   17 |   39 |   54 |   67 |   75 |   64 |   58 |   31 |    4 |    0 |    0
|    0 |    0 |   411 |
|  NULL |    0 |   13 |  112 |  516 | 1023 | 1392 | 1628 | 1728 | 1703 | 1570 | 1294 |  902 |  506 |  203
|   38 |    2 | 12629 |
+-------+------+------+------+------+------+------+------+------+------+------+------+------+------+-----
-+------+------+-------+
```

Midday in the summer is the best time for solar panels, as you would expect. 1-2pm in July was the best.

Postlog

Posted, Feb. 2015

See Also

- Rick James graciously allowed us to use this article in the documentation.Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: pivot

This page is licensed: CC BY-SA / Gnu FDL

Query Limits and Timeouts

This article describes the different methods MariaDB provides to limit/timeout a query:

LIMIT

```
SELECT ... LIMIT row_count
OR
SELECT ... LIMIT OFFSET, row_count
OR
SELECT ... LIMIT row_count OFFSET OFFSET
```

The LIMIT clause restricts the number of returned rows.

LIMIT ROWS EXAMINED

```
SELECT ... LIMIT ROWS EXAMINED rows_limit;
```

Stops the query after 'rows_limit' number of rows have been examined.

sql_safe_updates

If the sql_safe_updates variable is set, one can't execute an UPDATE or DELETE statement unless one specifies a key constraint in the `WHERE` clause or provide a `LIMIT` clause (or both).

```
SET @@SQL_SAFE_UPDATES=1
UPDATE tbl_name SET not_key_column=val;
-> ERROR 1175 (HY000): You are using safe update mode
  and you tried to update a table without a WHERE that uses a KEY column
```

sql_select_limit

sql_select_limit acts as an automatic `LIMIT row_count` to any SELECT query.

```
SET @@SQL_SELECT_LIMIT=1000
SELECT * FROM big_table;
```

The above is the same as:

```
SELECT * FROM big_table LIMIT 1000;
```

max_join_size

If the max_join_size variable (also called `sql_max_join_size`) is set, then it will limit any SELECT statements that probably need to examine more than `MAX_JOIN_SIZE` rows.

```
SET @@MAX_JOIN_SIZE=1000;
->ERROR 1104 (42000): The SELECT would examine more than MAX_JOIN_SIZE ROWS;
SELECT COUNT(null_column) FROM big_table;
  CHECK your WHERE AND USE SET SQL_BIG_SELECTS=1 OR SET MAX_JOIN_SIZE=# IF the SELECT IS okay
```

max_statement_time

If the max_statement_time variable is set, any query (excluding stored procedures) taking longer than the value of `max_statement_time` (specified in seconds) to execute will be aborted. This can be set globally, by session, as well as per user and per query. See Aborting statements that take longer than a certain time to execute.

See Also

- WAIT and NOWAIT
- Aborting statements that take longer than a certain time to execute
- lock_wait_timeout variable

This page is licensed: CC BY-SA / Gnu FDL

reorder_outer_joins

- **Introduced in**: MariaDB 12.3
- **Type**: `optimizer_switch` flag
- **Default Value**: `OFF`
- **Dynamic**: Yes
- **Scope**: Global

Description

The `reorder_outer_joins` flag in the `optimizer_switch` system variable manages whether the query optimizer reorders independent LEFT JOIN (and equivalent RIGHT JOIN, which are internally transformed to LEFT JOINs) operations to find the cost-effective execution plans.

With independent LEFT JOINs, there are no unnecessary dependencies on other outer joined tables; instead, each joined table depends only on the base table or on the previous tables in the join order. For example, in the query:

```
SELECT * FROM t1
LEFT JOIN t3 ON t3.key1 = t1.a
LEFT JOIN t2 ON t2.key1 = t1.a;
```

Before MariaDB 12.3, the optimizer was forced to follow the written order:

```
t1 → t3 → t2
```

Even if an alternative, such as is faster, depending on table sizes and indexing.

```
 t1 → t2 → t3
```

Recommended Configuration

It is recommended to always set `optimizer_prune_level=0` when enabling this optimization feature, unless the join involves many tables (e.g., more than 10). This prevents heuristic pruning from inadvertently discarding optimal join orders that become available in the expanded search space.

Configuration Example

```
SET optimizer_switch='reorder_outer_joins=ON';
SET optimizer_prune_level=0;
```

Enabling the Optimization

When you enable this optimization, the optimizer evaluates potential join orders based on estimated costs and heuristics.

```
SET optimizer_switch='reorder_outer_joins=on';
```

When it is disabled, it restores the default behavior:

```
SET optimizer_switch='reorder_outer_joins=off';
```

With this feature is enabled, the optimizer can choose better execution order.

Example

By default, outer joins impose ordering constraints during query optimization. For example:

```
SELECT * FROM t1
LEFT JOIN t3 ON t3.key1 = t1.a
LEFT JOIN t2 ON t2.key1 = t1.a;
```

- Without `reorder_outer_joins`, the optimizer may be restricted in how it orders `t2` and `t3` relative to `t1`, even when alternative join order are logically valid.

```
SET optimizer_switch='reorder_outer_joins=OFF';

EXPLAIN SELECT * FROM t1
     LEFT JOIN t3 ON t3.key1 = t1.a
     LEFT JOIN t2 ON t2.key1 = t1.a;

-- Possible Join order: t1 → t3 → t2
```

- When `reorder_outer_joins=ON`, the optimizer can evaluate additional join orders (similar to `INNER JOINs`), which may result in better execution plans.

```
SET optimizer_prune_level=0;
SET optimizer_switch='reorder_outer_joins=ON';

EXPLAIN SELECT * FROM t1
     LEFT JOIN t3 ON t3.key1 = t1.a
     LEFT JOIN t2 ON t2.key1 = t1.a;

-- With reordering enabled, the optimizer may choose:
-- t1 → t2 → t3 (if cost-based evaluation prefers it)
```

When to Use this Feature

Enabling `reorder_outer_joins=ON` in the following scenarios would be beneficial:

- Queries with Multiple Independent LEFT JOINs: When a query performs multiple `LEFT JOIN` operations against the same base table(s) without cross-dependencies between the outer tables.
- Small to Medium Join Sizes: Most effective when the total table count is typically ≤ 10. For larger joins (> 10 tables), the expanded search space might increase optimization time.
- Complex Outer Join Logic: Beneficial when using complex join structures (including those written in Oracle-style `(+)` syntax), where greater flexibility in reordering can unlock more efficient access paths.

Performance Considerations

This optimization `reorder_outer_joins` feature expands the search space by allowing the optimizer to evaluate join orders that were previously forbidden.

- Enabling this feature, together with the following setting, can help with queries that involve a smaller number of tables by allowing more join order possibilities (up to 10 tables, for instance):

```
SET optimizer_prune_level=0;
```

It may allow you to explore a more thorough search space, which could enhance the quality of the execution plan.

- Using `optimizer_prune_level=0` may result in a considerable increase in optimization (planning) time for queries that combine more tables.
- Heuristic pruning may remove potentially optimal join orders when `optimizer_prune_level` is larger than 0, which could lead to a less effective execution plan.

Use Cases

This optimization feature can be useful when:

- A query containing multiple `LEFT JOIN` operations may provide a suboptimal execution plan.
- Using Oracle compatibility outer join syntax, where table sequence in the `FROM` clause may affect join order. For example,

```
SELECT *
FROM t1,t2,t3
WHERE t1.col1 = t2.col1(+)
  AND t1.col2 = t3.col2(+);
```

Without `reorder_outer_joins`, the optimizer may follow the table order specified in the `FROM` clause.

With `reorder_outer_joins=ON`, the optimizer considers both:

```
t1 → t2 → t3
t1 → t3 → t2
```

and select the one with the lowest estimated cost.

Limitations

- Large joins may increase execution planning time because the number of possible join orders grows factorially.
- Pruning may eliminate optimal orders early, since an execution plan that looks inefficient first, could be the best one.

See Also

- optimizer_prune_level
- optimizer_trace
- optimizer_switch
- Optimizer Switch
- EXPLAIN

This page is licensed: CC BY-SA / Gnu FDL

Rollup Unique User Counts

The Problem

The normal way to count “Unique Users” is to take large log files, sort by userid, dedup, and count. This requires a rather large amount of processing. Furthermore, the count derived cannot be rolled up. That is, daily counts cannot be added to get weekly counts – some users will be counted multiple times.

So, the problem is to store the counts is such a way as to allow rolling up.

The solution

Let’s think about what we can do with a hash of the userid. The hash could map to a bit in a bit string. A BIT_COUNT of the bit string would give the 1-bits, representing the number of users. But that bit string would have to be huge. What if we could use shorter bit strings? Then different userids would be folded into the same bit. Let’s assume we can solve that.

Meanwhile, what about the rollup? The daily bit strings can be OR’d together to get a similar bit string for the week.

We have now figured out how to do the rollup, but have created another problem – the counts are too low.

Inflating the BIT_COUNT

A sufficiently random hash (eg MD5) will fold userids into the same bits with a predictable frequency. We need to figure this out, and work backwards. That is, given that X percent of the bits are set, we need a formula that says approximately how many userids were used to get those bits.

I simulated the problem by generating random hashes and calculated the number of bits that would be set. Then, with the help of Eureqa software, I derived the formula:

Y = 0.5456_X + 0.6543_tan(1.39_X_X*X)

How good is it?

The formula is reasonably precise. It is usually within 1% of the correct value; rarely off by 2%.

Of course, if virtually all the bits are set, the formula can't be very precise. Hence, you need to plan to have the bit strings big enough to handle the expected number of Uniques. In practice, you can use less than 1 bit per Unique. This would be a huge space savings over trying to save all the userids.

Another suggestion… If you are rolling up over a big span of time (eg hourly -> monthly), the bit strings must all be the same length, and the monthly string must be big enough to handle the expected count. This is likely to lead to very sparse hourly bit strings. Hence, it may be prudent to compress the hourly strings.

Postlog

Invented Nov, 2013; published Apr, 2014

Future: Rick is working on actual code (Sep, 2016)
It is complicated by bit-wise operations being limited to BIGINT.
However, with MySQL 8.0 (freshly released), the desired bit-wise operations can be applied to BLOB, greatly simplifying my code.
I hope to publish the pre-8.0 code soon; 8.0 code later.

See also

Rick James graciously allowed us to use this article in the documentation.

Rick James' site has other useful tips, how-tos, optimizations, and debugging tips.

Original source: uniques

This page is licensed: CC BY-SA / Gnu FDL

Rowid Filtering Optimization

The target use case for rowid filtering is as follows:

- a table uses ref access on index IDX1
- but it also has a fairly restrictive range predicate on another index IDX2.

In this case, it is advantageous to:

- Do an index-only scan on index IDX2 and collect rowids of index records into a data structure that allows filtering (let's call it $FILTER).
- When doing ref access on IDX1, check $FILTER before reading the full record.

Example

Consider a query

```
SELECT ...
FROM orders JOIN lineitem ON o_orderkey=l_orderkey
WHERE
  l_shipdate BETWEEN '1997-01-01' AND '1997-01-31' AND
  o_totalprice between 200000 and 230000;
```

Suppose the condition on `l_shipdate` is very restrictive, which means lineitem table should go first in the join order. Then, the optimizer can use `o_orderkey=l_orderkey` equality to do an index lookup to get the order the line item is from. On the other hand `o_totalprice between ...` can also be rather selective.

With filtering, the query plan would be:

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: lineitem
         type: range
possible_keys: PRIMARY,i_l_shipdate,i_l_orderkey,i_l_orderkey_quantity
          key: i_l_shipdate
      key_len: 4
          ref: NULL
         rows: 98
        Extra: Using index condition
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: orders
         type: eq_ref|filter
possible_keys: PRIMARY,i_o_totalprice
          key: PRIMARY|i_o_totalprice
      key_len: 4|9
          ref: dbt3_s001.lineitem.l_orderkey
         rows: 1 (5%)
        Extra: Using where; Using rowid filter
```

Note that table `orders` has “Using rowid filter”. The `type` column has `"|filter"`, the `key` column shows the index that is used to construct the filter. `rows` column shows the expected filter selectivity, it is 5%.

ANALYZE FORMAT=JSON output for table orders will show

```
"table": {
      "table_name": "orders",
      "access_type": "eq_ref",
      "possible_keys": ["PRIMARY", "i_o_totalprice"],
      "key": "PRIMARY",
      "key_length": "4",
      "used_key_parts": ["o_orderkey"],
      "ref": ["dbt3_s001.lineitem.l_orderkey"],
      "rowid_filter": {
        "range": {
          "key": "i_o_totalprice",
          "used_key_parts": ["o_totalprice"]
        },
        "rows": 69,
        "selectivity_pct": 4.6,
        "r_rows": 71,
        "r_selectivity_pct": 10.417,
        "r_buffer_size": 53,
        "r_filling_time_ms": 0.0716
      }
```

Note the `rowid_filter` element. It has a `range` element inside it. `selectivity_pct` is the expected selectivity, accompanied by the `r_selectivity_pct` showing the actual observed selectivity.

Details

- The optimizer makes a cost-based decision about when the filter should be used.
- The filter data structure is currently an ordered array of rowids. (a Bloom filter would be better here and will probably be introduced in the future versions).
- The optimization needs to be supported by the storage engine. At the moment, it is supported by InnoDB and MyISAM. It is not supported in partitioned tables.

Limitations

- Rowid Filtering can't be used with a backward-ordered index scan. When the optimizer needs to execute an ORDER BY … DESC query and decides to handle it with a backward-ordered index scan, it will disable Rowid Filtering.

Control

Rowid filtering can be switched on/off using `rowid_filter` flag in the optimizer_switch variable. By default, the optimization is enabled.

This page is licensed: CC BY-SA / Gnu FDL

Sargable DATE and YEAR

Starting from MariaDB 11.1, conditions in the form

```
YEAR(indexed_date_col) CMP const_value
DATE(indexed_date_col) CMP const_value
```

are sargable, provided that

- CMP is any of `=`, `<=>`, `<`, `<=`, `>`, `>=`.
- `indexed_date_col` has a type of `DATE`, `DATETIME` or `TIMESTAMP` and is a part of some index.

One can swap the left and right hand sides of the equality: `const_value CMP {DATE|YEAR}(indexed_date_col)` is also handled.

Sargable here means that the optimizer is able to use such conditions to construct access methods, estimate their selectivity, or use them to perform partition pruning.

Implementation

Internally, the optimizer rewrites the condition to an equivalent condition which doesn't use `YEAR` or `DATE` functions.

For example, `YEAR(date_col)=2023` is rewritten into `date_col between '2023-01-01' and '2023-12-31'`.

Similarly, `DATE(datetime_col) <= '2023-06-01'` is rewritten into `datetime_col <= '2023-06-01 23:59:59'`.

Controlling the Optimization

The optimization is always ON, there is no Optimizer Switch flag to control it.

Optimizer Trace

The rewrite is logged as `date_conds_into_sargable` transformation. Example:

```
{
            "transformation": "date_conds_into_sargable",
            "before": "cast(t1.datetime_col as date) <= '2023-06-01'",
            "after": "t1.datetime_col <= '2023-06-01 23:59:59'"
          },
```

References

- MDEV-8320: Allow index usage for DATE(datetime_column) = const

This page is licensed: CC BY-SA / Gnu FDL

Sargable UPPER

Starting from MariaDB 11.3, expressions in the form

```
UPPER(key_col) = expr
UPPER(key_col) IN (constant-list)
```

are sargable if `key_col` uses either the `utf8mb3_general_ci` or `utf8mb4_general_ci` collation.

`UCASE` is a synonym for UPPER so is covered as well.

Sargable means that the optimizer is able to use such conditions to construct access methods, estimate their selectivity, or perform partition pruning.

Example

```
CREATE TABLE t1 (
  key1 VARCHAR(32) COLLATE utf8mb4_general_ci,
  ...
  KEY(key1)
);
```

```
EXPLAIN SELECT * FROM t1 WHERE UPPER(key1)='ABC'
+------+-------------+-------+------+---------------+------+---------+-------+------+--------------------
------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref   | rows | Extra
|
+------+-------------+-------+------+---------------+------+---------+-------+------+--------------------
------+
|    1 | SIMPLE      | t1    | ref  | key1          | key1 | 131     | const | 1    | Using where; Using
index |
+------+-------------+-------+------+---------------+------+---------+-------+------+--------------------
------+
```

Note that `ref` access is used.

An example with join:

```
EXPLAIN SELECT * FROM t0,t1 WHERE upper(t1.key1)=t0.col;
+------+-------------+-------+------+---------------+------+---------+-------------+------+-------------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref         | rows | Extra       |
+------+-------------+-------+------+---------------+------+---------+-------------+------+-------------+
|    1 | SIMPLE      | t0    | ALL  | NULL          | NULL | NULL    | NULL        | 10   | Using where |
|    1 | SIMPLE      | t1    | ref  | key1          | key1 | 131     | test.t0.col | 1    | Using index |
+------+-------------+-------+------+---------------+------+---------+-------------+------+-------------+
```

Here, the optimizer was able to construct `ref` access.

Controlling the Optimization

The optimizer_switch variable has the flag `sargable_casefold` to turn the optimization on and off. The default is ON.

Optimizer Trace

The optimization is implemented as a rewrite for a query's WHERE/ON conditions. It uses the `sargable_casefold_removal` object name in the trace:

```
"join_optimization": {
        "select_id": 1,
        "steps": [
          {
            "sargable_casefold_removal": {
              "before": "ucase(t1.key1) = t0.col",
              "after": "t1.key1 = t0.col"
            }
          },
```

References

- MDEV-31496: Make optimizer handle UCASE(varchar_col)=…
- An analog for LCASE is not possible. See MDEV-31955: Make optimizer handle LCASE(varchar_col)=… for details.

This page is licensed: CC BY-SA / Gnu FDL

USE INDEX

You can limit which indexes are considered with the `USE INDEX` option.

Syntax

```
USE INDEX [{FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
```

Description

The default is 'FOR JOIN', which means that the hint only affects how the WHERE clause is optimized.

USE INDEX is used after the table name in the FROM clause.

USE INDEX cannot use an ignored index - it will be treated as if it doesn't exist.

Index Prefixes

When using index hints (USE, FORCE or IGNORE INDEX), the index name value can also be an unambiguous prefix of an index name.

Example

```
CREATE INDEX Name ON City (Name);
CREATE INDEX CountryCode ON City (Countrycode);
EXPLAIN SELECT Name FROM City USE INDEX (CountryCode)
WHERE name="Helsingborg" AND countrycode="SWE";
```

This produces:

```
id  select_type table   type    possible_keys   key key_len ref rows    Extra
1   SIMPLE  City    ref CountryCode CountryCode 3   const   14  Using where
```

If we had not used USE INDEX, the Name index would have been in possible keys.

See Also

- Index Hints: How to Force Query Plans for more details

- IGNORE INDEX
- FORCE INDEX
- Ignored Indexes

This page is licensed: CC BY-SA / Gnu FDL

Virtual Column Support in the Optimizer

> **Note:**
>
> This feature is available from MariaDB 11.8.

The optimizer can recognize use of indexed virtual column expressions in the `WHERE` clause and use them to construct range and `ref(const)` accesses.

Example

Consider this table with data in JSON format:

```
CREATE TABLE t1 (json_data JSON);
INSERT INTO t1 VALUES('{"column1": 1234}');
INSERT INTO t1 ...
```

In order to do efficient queries over data in JSON, you can add a virtual column, and an index on that column:

```
ALTER TABLE t1
  ADD COLUMN vcol1 INT AS (cast(json_value(json_data, '$.column1') AS INTEGER)),
  ADD INDEX(vcol1);
```

Before MariaDB 11.8, you had to use `vcol1` in the `WHERE` clause. Now, you can use the virtual column expression, too:

```
-- This uses the index before 11.8:
EXPLAIN SELECT * FROM t1 WHERE vcol1=100;
-- Starting from 11.8, this uses the index, too:
EXPLAIN SELECT * FROM t1
WHERE cast(json_value(json_data, '$.column1') AS INTEGER)=100;
```

```
+------+-------------+-------+------+---------------+-------+---------+-------+------+-------+
| id   | select_type | table | type | possible_keys | key   | key_len | ref   | rows | Extra |
+------+-------------+-------+------+---------------+-------+---------+-------+------+-------+
|    1 | SIMPLE      | t1    | ref  | vcol1         | vcol1 | 5       | const | 1    |       |
+------+-------------+-------+------+---------------+-------+---------+-------+------+-------+
```

General Considerations

- In MariaDB, one has to create a virtual column and then create an index over it. Other databases allow to create an index directly over expression: `create index on t1((col1+col2))`. This is not yet supported in MariaDB (MDEV-35853).
- The `WHERE` clause must use the exact same expression as in the virtual column definition.
- The optimization is implemented in a way similar to MySQL – the optimizer finds potentially useful occurrences of `vcol_expr` in the `WHERE` clause and replaces them with `vcol_name`.
- In the optimizer trace, the rewrites are shown like this:

```
"virtual_column_substitution": {
              "condition": "WHERE",
              "resulting_condition": "t1.vcol1 = 100"
            }
```

Note:

The following improvements are available from MariaDB 12.1.

1. Improved Optimizer plans for `SELECT` statements with `ORDER BY` or `GROUP BY` virtual columns when the virtual column expressions are covered by indexes that can be used.
2. Improved Optimizer plans for `SELECT` statements with `ORDER BY` or `GROUP BY` virtual columns expressions, by substitution of the virtual column expressions with virtual columns when the virtual columns are usable indexes themselves.
3. The same improvements apply for **single-table** `UPDATE` or `DELETE` statements.

Accessing JSON fields

Cast the Value to the Desired Type

SQL is strongly-typed language while JSON is weakly-typed. This means one must specify the desired datatype when accessing JSON data from SQL. In the above example, we declared `vcol1` as `INT` and then used `(CAST ... AS INTEGER)` (both in the ALTER TABLE and in the `WHERE` clause in SELECT query:):

```
ALTER TABLE t1
  ADD COLUMN vcol1 INT AS (CAST(json_value(json_data, '$.column1') AS INTEGER)) ...
```

```
SELECT ...  WHERE ... CAST(json_value(json_data, '$.column1') AS INTEGER) ...;
```

Specify the Collation for Strings

When extracting string values, `CAST` is not necessary, as `JSON_VALUE` returns strings. However, you must take into account collations. Consider this column declared as `JSON`:

```
CREATE TABLE t1 (
  json_data JSON
  ...
```

The collation of `json_data` is `utf8mb4_bin`. The collation of `JSON_VALUE(json_data, ...)` is `utf8mb4_bin`, too.

Most use cases require a more commonly-used collation. It is possible to achieve that using the `COLLATE` clause:

```
ALTER TABLE t1
  ADD col1 VARCHAR(100) COLLATE utf8mb4_uca1400_ai_ci AS
  (json_value(js1, '$.string_column') COLLATE utf8mb4_uca1400_ai_ci),
  ADD INDEX(col1);
...
SELECT  ...
WHERE
  json_value(js1, '$.string_column') COLLATE utf8mb4_uca1400_ai_ci='string-value';
```

References

- MDEV-35616: Add basic optimizer support for virtual columns

This page is licensed: CC BY-SA / Gnu FDL

Optimization Strategies

See Duplicate Weedout Strategy.

DuplicateWeedout is a semi-join execution strategy that runs the subquery as a regular inner join and removes duplicate record combinations using a temporary table.

See FirstMatch Strategy.

FirstMatch is a semi-join execution strategy that avoids duplicate results by short-cutting subquery execution as soon as the first matching record is found, improving performance.

See Improvements to ORDER BY Optimization.

Describes MariaDB improvements to the ORDER BY optimizer, including cost- based index switching for ORDER BY with small LIMIT and multiple-equality handling.

See LooseScan Strategy.

LooseScan is a semi-join execution strategy that avoids duplicate record combinations by using an index on the subquery table to pick one row per group.

See Semi-join Materialization Strategy.

Semi-join Materialization runs an uncorrelated IN-subquery into a temporary table, then joins it via the Materialization-scan or Materialization-lookup strategy.

This page is licensed: CC BY-SA / Gnu FDL

Duplicate Weedout Strategy

`DuplicateWeedout` is an execution strategy for Semi-join subqueries.

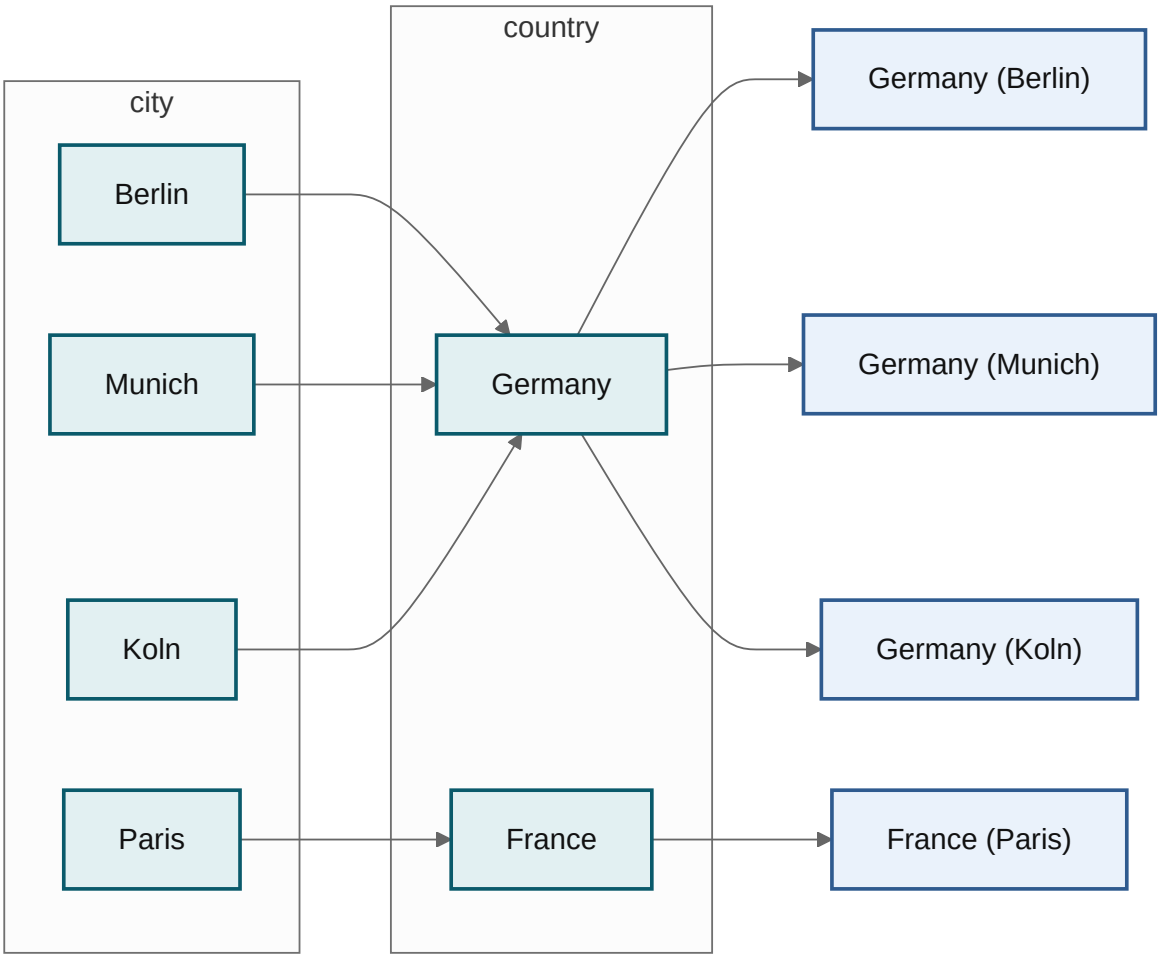
The idea

The idea is to run the semi-join (a query with uses `WHERE X IN (SELECT Y FROM ...)`) as if it were a regular inner join, and then eliminate the duplicate record combinations using a temporary table.

Suppose, you have a query where you're looking for countries which have more than 33% percent of their population in one big city:

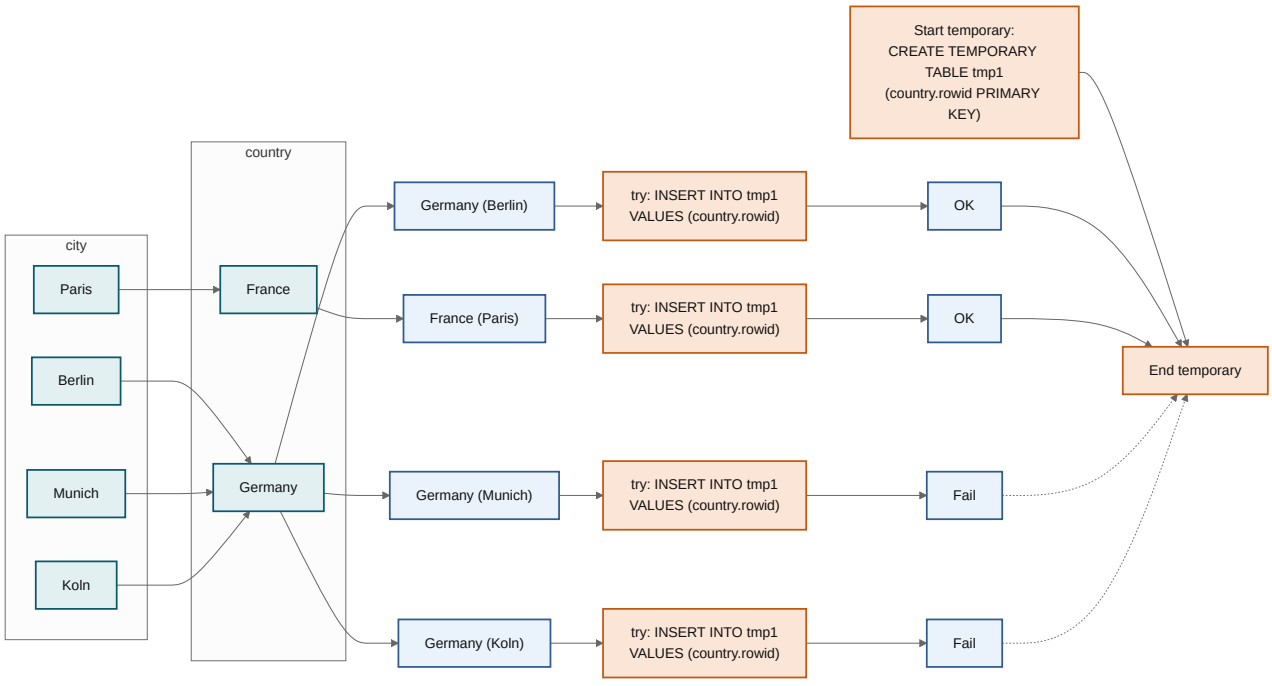
```
SELECT *
FROM Country
WHERE
   Country.code IN (SELECT City.Country
                    FROM City
                    WHERE
                      City.Population > 0.33 * Country.Population AND
                      City.Population > 1*1000*1000);
```

First, we run a regular inner join between the `City` and `Country` tables:



The inner join between `city` and `country` produces a duplicate `Germany` row for each of its three big cities.

Now, lets put `DuplicateWeedout` into the picture:



`DuplicateWeedout` inserts each candidate row's `country.rowid` into a temporary table with a primary key; the insert fails for rows that would duplicate `Germany`, so only one `Germany` row survives.

DuplicateWeedout in action

The `Start temporary` and `End temporary` from the last diagram are shown in the `EXPLAIN` output:

```
EXPLAIN SELECT * FROM Country WHERE Country.code IN
  (select City.Country from City where City.Population > 0.33 * Country.Population
   AND City.Population > 1*1000*1000)\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        TABLE: City
         type: RANGE
possible_keys: Population,Country
          KEY: Population
      key_len: 4
          ref: NULL
         ROWS: 238
        Extra: USING INDEX CONDITION; Start temporary
*************************** 2. row ***************************
           id: 1
  select_type: PRIMARY
        TABLE: Country
         type: eq_ref
possible_keys: PRIMARY
          KEY: PRIMARY
      key_len: 3
          ref: world.City.Country
         ROWS: 1
        Extra: USING WHERE; End temporary
2 rows in set (0.00 sec)
```

This query will read 238 rows from the `City` table, and for each of them will make a primary key lookup in the `Country` table, which gives another 238 rows. This gives a total of 476 rows, and you need to add 238 lookups in the temporary table (which are typically *much* cheaper since the temporary table is in-memory).

If we run the same query with semi-join optimizations disabled, we'll get:

```
EXPLAIN SELECT * FROM Country WHERE Country.code IN
  (select City.Country from City where City.Population > 0.33 * Country.Population
    AND City.Population > 1*1000*1000)\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        TABLE: Country
         type: ALL
possible_keys: NULL
          KEY: NULL
      key_len: NULL
          ref: NULL
         ROWS: 239
        Extra: USING WHERE
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        TABLE: City
         type: index_subquery
possible_keys: Population,Country
          KEY: Country
      key_len: 3
          ref: func
         ROWS: 18
        Extra: USING WHERE
2 rows in set (0.00 sec)
```

This plan will read `(239 + 239*18) = 4541` rows, which is much slower.

Factsheet

- `DuplicateWeedout` is shown as “Start temporary/End temporary” in `EXPLAIN`.
- The strategy can handle correlated subqueries.
- But it cannot be applied if the subquery has meaningful `GROUP BY` and/or aggregate functions.
- `DuplicateWeedout` allows the optimizer to freely mix a subquery’s tables and the parent select’s tables.
- There is no separate @@optimizer_switch flag for `DuplicateWeedout`. The strategy can be disabled by switching off all semi-join optimizations with `SET @@optimizer_switch='optimizer_semijoin=off'` command.

See Also

- Subquery Optimizations Map

This page is licensed: CC BY-SA / Gnu FDL

FirstMatch Strategy

`FirstMatch` is an execution strategy for Semi-join subqueries.

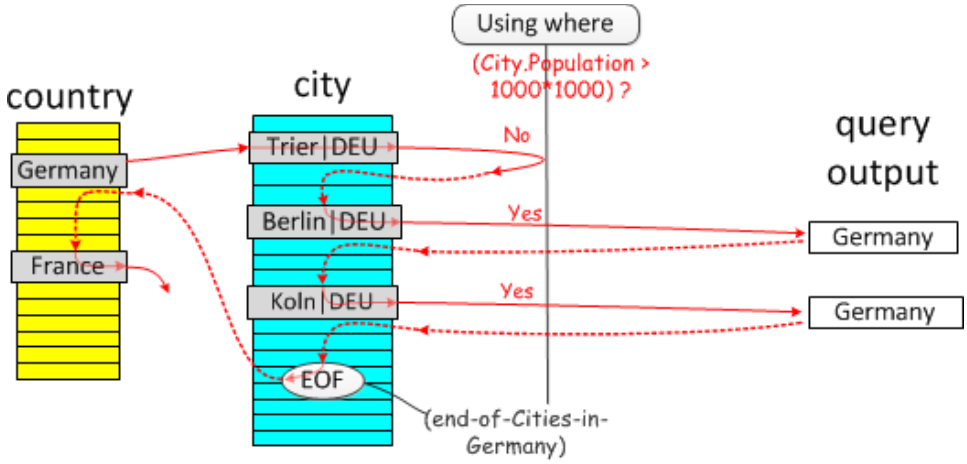
The Idea

It is similar to how `IN/EXISTS` subqueries were executed in MySQL 5.x.

Let’s take the usual example of a search for countries with big cities:

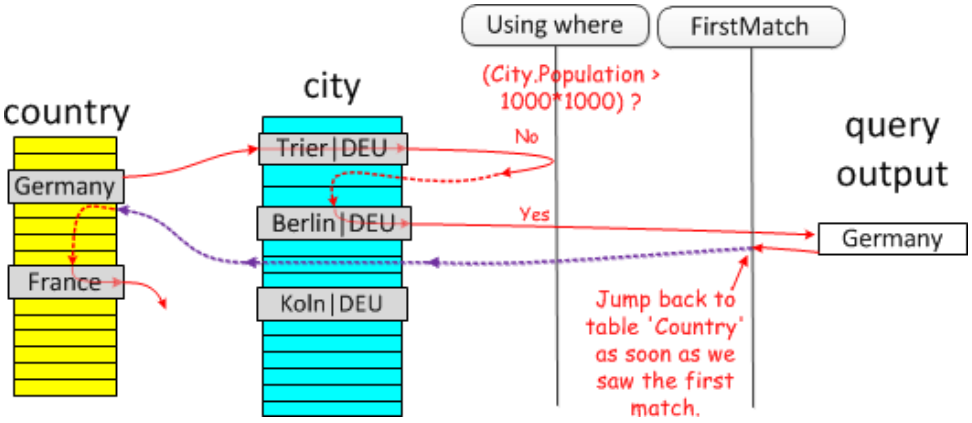
```
SELECT * FROM Country
WHERE Country.code IN (SELECT City.Country
                       FROM City
                       WHERE City.Population > 1*1000*1000)
      AND Country.continent='Europe'
```

Suppose our execution plan is to find countries in Europe, and then, for each found country, check if it has any big cities. Regular inner join execution will look as follows:



firstmatch-inner-join

Since Germany has two big cities (in this diagram), it will be put into the query output twice. This is not correct, `SELECT ... FROM Country` should not produce the same country record twice. The `FirstMatch` strategy avoids the production of duplicates by short-cutting execution as soon as the first genuine match is found:



firstmatch-firstmatch

Note that the short-cutting has to take place after `Using where` has been applied. It would have been wrong to short-cut after we found *Trier*.

FirstMatch in Action

The `EXPLAIN` for the above query looks as follows:

```
MariaDB [world]> EXPLAIN SELECT * FROM Country WHERE Country.code IN
  (select City.Country from City where City.Population > 1*1000*1000)
    AND Country.continent='Europe';
+----+-------------+---------+------+--------------------+-----------+---------+--------------------+----
--+---------------------------------+
| id | select_type | table   | type | possible_keys      | key       | key_len | ref                |
rows | Extra                           |
+----+-------------+---------+------+--------------------+-----------+---------+--------------------+----
--+---------------------------------+
|  1 | PRIMARY     | Country | ref  | PRIMARY,continent  | continent | 17      | const              |
60 | Using index condition           |
|  1 | PRIMARY     | City    | ref  | Population,Country | Country   | 3       | world.Country.Code |
18 | Using where; FirstMatch(Country) |
+----+-------------+---------+------+--------------------+-----------+---------+--------------------+----
--+---------------------------------+
2 rows in set (0.00 sec)
```

`FirstMatch(Country)` in the Extra column means that *as soon as we have produced one matching record combination, short-cut the execution and jump back to the Country* table.

The `FirstMatch` query plan is similar to one you would get in MySQL:

```
MySQL [world]> EXPLAIN SELECT * FROM Country  WHERE Country.code IN
  (select City.Country from City where City.Population > 1*1000*1000)
   AND Country.continent='Europe';
+----+--------------------+---------+---------------+--------------------+-----------+---------+-------
+------+-------------------------------------+
| id | select_type        | table   | type          | possible_keys      | key       | key_len | ref   |
rows | Extra                               |
+----+--------------------+---------+---------------+--------------------+-----------+---------+-------
+------+-------------------------------------+
|  1 | PRIMARY            | Country | ref           | continent          | continent | 17      | const |
60 | Using index condition; Using where |
|  2 | DEPENDENT SUBQUERY | City    | index_subquery | Population,Country | Country   | 3       | func  |
18 | Using where                         |
+----+--------------------+---------+---------------+--------------------+-----------+---------+-------
+------+-------------------------------------+
2 rows in set (0.01 sec)
```

These two particular query plans will execute in the same time.

Difference between FirstMatch and IN->EXISTS

The general idea behind the `FirstMatch` strategy is the same as the one behind the `IN->EXISTS` transformation, however, `FirstMatch` has several advantages:

- Equality propagation works across semi-join bounds, but not subquery bounds. Therefore, converting a subquery to semi-join and using `FirstMatch` can still give a better execution plan.
- There is only one way to apply the `IN->EXISTS` strategy and MySQL will do it unconditionally. With `FirstMatch`, the optimizer can make a choice between whether it should run the `FirstMatch` strategy as soon as all tables used in the subquery are in the join prefix, or at some later point in time.

FirstMatch Factsheet

- The `FirstMatch` strategy works by executing the subquery and short-cutting its execution as soon as the first match is found.
- This means, subquery tables must be after all of the parent select's tables that are referred from the subquery predicate.
- `EXPLAIN` shows `FirstMatch` as "`FirstMatch(tableN)`".
- The strategy can handle correlated subqueries.
- But it cannot be applied if the subquery has meaningful `GROUP BY` and/or aggregate functions.
- Use of the `FirstMatch` strategy is controlled with the `firstmatch=on|off` flag in the optimizer_switch variable.

See Also

- Semi-join subquery optimizations

This page is licensed: CC BY-SA / Gnu FDL

Improvements to ORDER BY Optimization

Available tuning for ORDER BY with small LIMIT

- In 2024, fix for MDEV-34720 has added optimizer_join_limit_pref_ratio optimization into MariaDB starting from 10.6. It allows one to enable extra optimization for ORDER BY with small LIMIT.

Older optimizations

MariaDB brought several improvements to the ORDER BY optimizer.

The fixes were made as a response to complaints by MariaDB customers, so they fix real-world optimization problems. The fixes are a bit hard to describe (as the `ORDER BY` optimizer is complicated), but here's a short description:

The ORDER BY optimizer:

- Doesn't make stupid choices when several multi-part keys and potential range accesses are present (MDEV-6402).
  - This also fixes MySQL Bug#12113.
- Always uses "range" and (not full "index" scan) when it switches to an index to satisfy `ORDER BY … LIMIT` (MDEV-6657).
- Tries hard to be smart and use cost/number of records estimates from other parts of the optimizer (MDEV-6384, MDEV-465).
  - This change also fixes MySQL Bug#36817.
- Takes full advantage of InnoDB's Extended Keys feature when checking if filesort() can be skipped (MDEV-6796).

Extra optimizations

- The ORDER BY optimizer takes multiple-equalities into account (MDEV-8989). This optimization is enabled by default.

Comparison with MySQL 5.7

In MySQL 5.7 changelog, one can find this passage:

Make switching of index due to small limit cost-based (WL#6986) : We have made the decision in make_join_select() of whether to switch to a new index in order to support "ORDER BY … LIMIT N" cost-based. This work fixes Bug#73837.

MariaDB is not using Oracle's fix (we believe `make_join_select` is not the right place to do ORDER BY optimization), but the effect is the same.

See Also

- Blog post MariaDB 10.1: Better query optimization for ORDER BY … LIMIT

This page is licensed: CC BY-SA / Gnu FDL

LooseScan Strategy

LooseScan is an execution strategy for Semi-join subqueries.

The idea

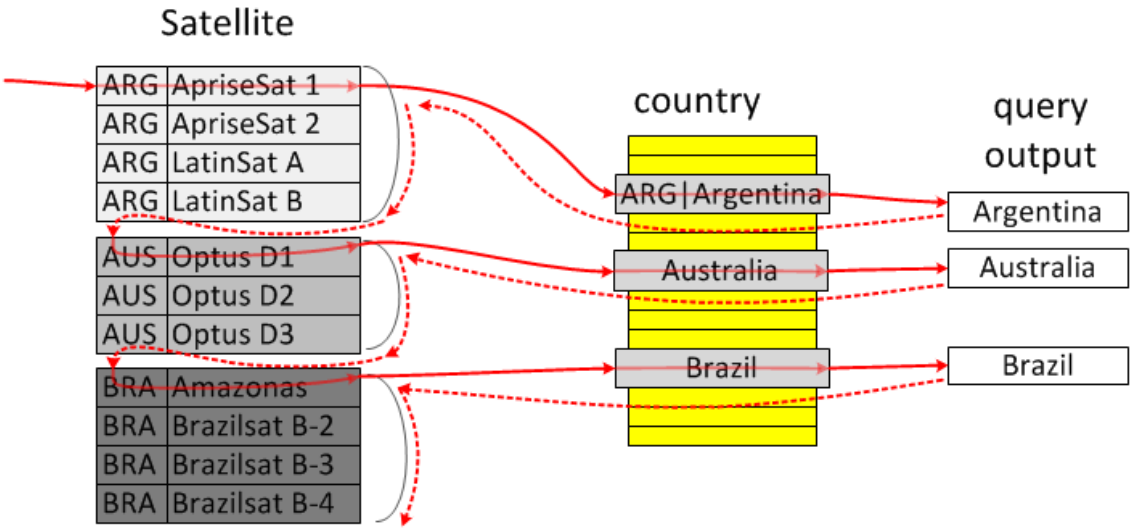
We will demonstrate the `LooseScan` strategy by example. Suppose, we're looking for countries that have satellites. We can get them using the following query (for the sake of simplicity we ignore satellites that are owned by consortiums of multiple countries):

```
SELECT * FROM Country
WHERE
  Country.code IN (SELECT country_code FROM Satellite)
```

Suppose, there is an index on `Satellite.country_code`. If we use that index, we will get satellites in the order of their owner country:

loosescan-satellites-ordered-r2

The `LooseScan` strategy doesn't really need ordering, what it needs is grouping. In the above figure, satellites are grouped by country. For instance, all satellites owned by Australia come together, without being mixed with satellites of other countries. This makes it easy to select just one satellite from each group, which you can join with its country and get a list of countries without duplicates:



loosescan-diagram-no-where

LooseScan in action

The `EXPLAIN` output for the above query looks as follows:

```
MariaDB [world]> EXPLAIN SELECT * FROM Country WHERE Country.code IN
  (select country_code from Satellite);
+----+-------------+-----------+--------+---------------+-------------+---------+----------------------
-------+------+-------------------------------------+
| id | select_type | table     | type   | possible_keys | key         | key_len | ref
| rows | Extra                               |
+----+-------------+-----------+--------+---------------+-------------+---------+----------------------
-------+------+-------------------------------------+
|  1 | PRIMARY     | Satellite | index  | country_code  | country_code | 9       | NULL
|  932 | Using where; Using index; LooseScan |
|  1 | PRIMARY     | Country   | eq_ref | PRIMARY       | PRIMARY      | 3       |
world.Satellite.country_code |    1 | Using index condition               |
+----+-------------+-----------+--------+---------------+-------------+---------+----------------------
-------+------+-------------------------------------+
```

Factsheet

- LooseScan avoids the production of duplicate record combinations by putting the subquery table first and using its index to select one record from multiple duplicates
- Hence, in order for LooseScan to be applicable, the subquery should look like:

```
expr IN (SELECT tbl.keypart1 FROM tbl ...)
```

or

```
expr IN (SELECT tbl.keypart2 FROM tbl WHERE tbl.keypart1=const AND ...)
```

- LooseScan can handle correlated subqueries
- LooseScan can be switched off by setting the `loosescan=off` flag in the optimizer_switch variable.

This page is licensed: CC BY-SA / Gnu FDL

Semi-join Materialization Strategy

Semi-join Materialization is a special kind of subquery materialization used for Semi-join subqueries. It actually includes two strategies:

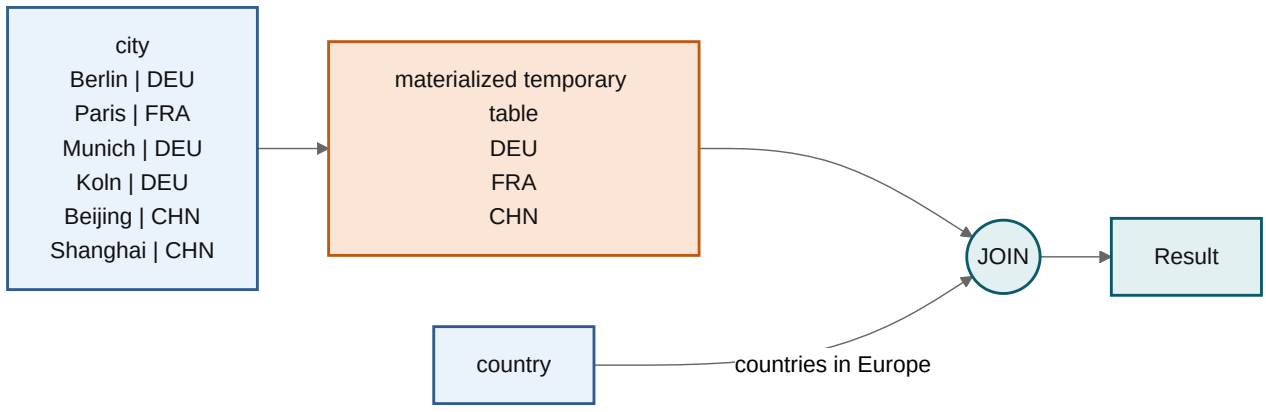
- Materialization/lookup
- Materialization/scan

The idea

Consider a query that finds countries in Europe which have big cities:

```
SELECT * FROM Country
WHERE Country.code IN (SELECT City.Country
                       FROM City
                       WHERE City.Population > 7*1000*1000)
      AND Country.continent='Europe'
```

The subquery is uncorrelated, that is, we can run it independently of the upper query. The idea of semi-join materialization is to do just that, and fill a temporary table with possible values of the City.country field of big cities, and then do a join with countries in Europe:



City rows feed a materialized temporary table of distinct country codes, which is joined with countries in Europe to produce the result.

The join can be done in two directions:

1. From the materialized table to countries in Europe

2. From countries in Europe to the materialized table

The first way involves doing a full scan on the materialized table, so we call it “Materialization-scan”.

If you run a join from Countries to the materialized table, the cheapest way to find a match in the materialized table is to make a lookup on its primary key (it has one: we used it to remove duplicates). Because of that, we call the strategy “Materialization-lookup”.

Semi-join materialization in action

Materialization-Scan

If we chose to look for cities with a population greater than 7 million, the optimizer will use Materialization-Scan and `EXPLAIN` will show this:

```
MariaDB [world]> EXPLAIN SELECT * FROM Country WHERE Country.code IN
  (select City.Country from City where  City.Population > 7*1000*1000);
+----+--------------+-------------+--------+--------------------+------------+---------+-----------------
---+------+-----------------------+
| id | select_type  | table       | type   | possible_keys      | key        | key_len | ref
| rows | Extra                 |
+----+--------------+-------------+--------+--------------------+------------+---------+-----------------
---+------+-----------------------+
|  1 | PRIMARY      | <subquery2> | ALL    | distinct_key       | NULL       | NULL    | NULL
|   15 |                       |
|  1 | PRIMARY      | Country     | eq_ref | PRIMARY            | PRIMARY    | 3       |
world.City.Country |    1 |                       |
|  2 | MATERIALIZED | City        | range  | Population,Country | Population | 4       | NULL
|   15 | Using index condition |
+----+--------------+-------------+--------+--------------------+------------+---------+-----------------
---+------+-----------------------+
3 rows in set (0.01 sec)
```

Here, you can see:

- There are still two `SELECT`s (look for columns with `id=1` and `id=2`)
- The second select (with `id=2`) has `select_type=MATERIALIZED`. This means it will be executed and its results will be stored in a temporary table with a unique key over all columns. The unique key is there to prevent the table from containing any duplicate records.
- The first select received the table name `subquery2`. This is the table that we got as a result of the materialization of the select with `id=2`.

The optimizer chose to do a full scan over the materialized table, so this is an example of a use of the Materialization-Scan strategy.

As for execution costs, we’re going to read 15 rows from table City, write 15 rows to materialized table, read them back (the optimizer assumes there won’t be any duplicates), and then do 15 eq_ref accesses to table Country. In total, we’ll do 45 reads and 15 writes.

By comparison, if you run the `EXPLAIN` with semi-join optimizations disabled, you’ll get this:

```
MariaDB [world]> EXPLAIN SELECT * FROM Country WHERE Country.code IN
  (select City.Country from City where  City.Population > 7*1000*1000);
+----+--------------------+---------+-------+--------------------+------------+---------+------+------+--
----------------------------------+
| id | select_type        | table   | type  | possible_keys      | key        | key_len | ref  | rows |
Extra                            |
+----+--------------------+---------+-------+--------------------+------------+---------+------+------+--
----------------------------------+
|  1 | PRIMARY            | Country | ALL   | NULL               | NULL       | NULL    | NULL |  239 |
Using where                      |
|  2 | DEPENDENT SUBQUERY | City    | range | Population,Country | Population | 4       | NULL |   15 |
Using index condition; Using where |
+----+--------------------+---------+-------+--------------------+------------+---------+------+------+--
----------------------------------+
```

…which is a plan to do `(239 + 239*15) = 3824` table reads.

Materialization-Lookup

Let's modify the query slightly and look for countries which have cities with a population over one million (instead of seven):

```
MariaDB [world]> EXPLAIN SELECT * FROM Country WHERE Country.code IN
  (select City.Country from City where  City.Population > 1*1000*1000) ;
+----+-------------+-------------+--------+--------------------+-------------+---------+------+------+-
----------------------+
| id | select_type  | table       | type   | possible_keys      | key          | key_len | ref  | rows |
Extra                 |
+----+-------------+-------------+--------+--------------------+-------------+---------+------+------+-
----------------------+
|  1 | PRIMARY      | Country     | ALL    | PRIMARY            | NULL         | NULL    | NULL |  239 |
|
|  1 | PRIMARY      | <subquery2> | eq_ref | distinct_key       | distinct_key | 3       | func |    1 |
|
|  2 | MATERIALIZED | City        | range  | Population,Country | Population   | 4       | NULL |  238 |
Using index condition |
+----+-------------+-------------+--------+--------------------+-------------+---------+------+------+-
----------------------+
3 rows in set (0.00 sec)
```

The `EXPLAIN` output is similar to the one which used Materialization-scan, except that:

- the `<subquery2>` table is accessed with the `eq_ref` access method
- the access uses an index named `distinct_key`

This means that the optimizer is planning to do index lookups into the materialized table. In other words, we're going to use the Materialization-lookup strategy.

With `optimizer_switch='semijoin=off,materialization=off'`), one will get this `EXPLAIN`:

```
MariaDB [world]> EXPLAIN SELECT * FROM Country WHERE Country.code IN
  (select City.Country from City where  City.Population > 1*1000*1000) ;
+----+--------------------+---------+----------------+--------------------+---------+---------+------+---
---+-------------+
| id | select_type        | table   | type           | possible_keys      | key     | key_len | ref  |
rows | Extra       |
+----+--------------------+---------+----------------+--------------------+---------+---------+------+---
---+-------------+
|  1 | PRIMARY            | Country | ALL            | NULL               | NULL    | NULL    | NULL |
239 | Using where |
|  2 | DEPENDENT SUBQUERY | City    | index_subquery | Population,Country | Country | 3       | func |
18 | Using where |
+----+--------------------+---------+----------------+--------------------+---------+---------+------+---
---+-------------+
```

One can see that both plans will do a full scan on the `Country` table. For the second step, MariaDB will fill the materialized table (238 rows read from table City and written to the temporary table) and then do a unique key lookup for each record in table `Country`, which works out to 238 unique key lookups. In total, the second step will cost `(239+238) = 477` reads and `238` temp.table writes.

Execution of the latter (`DEPENDENT SUBQUERY`) plan reads 18 rows using an index on `City.Country` for each record it receives for table `Country`. This works out to a cost of `(18*239) = 4302` reads. Had there been fewer subquery invocations, this plan would have been better than the one with Materialization. By the way, MariaDB has an option to use such a query plan, too (see FirstMatch Strategy), but it did not choose it.

Subqueries with grouping

MariaDB is able to use Semi-join materialization strategy when the subquery has grouping (other semi-join strategies are not applicable in this case).

This allows for efficient execution of queries that search for the best/last element in a certain group.

For example, let's find cities that have the biggest population on their continent:

```
EXPLAIN
SELECT * FROM City
WHERE City.Population IN (SELECT max(City.Population) FROM City, Country
                          WHERE City.Country=Country.Code
                          GROUP BY Continent)
+------+-------------+-------------+------+---------------+------------+---------+----------------------
------------+------+-----------------+
| id   | select_type  | table       | type | possible_keys | key        | key_len | ref
| rows | Extra           |
+------+-------------+-------------+------+---------------+------------+---------+----------------------
------------+------+-----------------+
|    1 | PRIMARY      | <subquery2> | ALL  | distinct_key  | NULL       | NULL    | NULL
|  239 |                 |
|    1 | PRIMARY      | City        | ref  | Population    | Population | 4       |
<subquery2>.max(City.Population) |    1 |                |
|    2 | MATERIALIZED | Country     | ALL  | PRIMARY       | NULL       | NULL    | NULL
|  239 | Using temporary |
|    2 | MATERIALIZED | City        | ref  | Country       | Country    | 3       | world.Country.Code
|   18 |                 |
+------+-------------+-------------+------+---------------+------------+---------+----------------------
------------+------+-----------------+
4 rows in set (0.00 sec)
```

the cities are:

```
+------+------------------+---------+------------+
| ID   | Name             | Country | Population |
+------+------------------+---------+------------+
| 1024 | Mumbai (Bombay)  | IND     |   10500000 |
| 3580 | Moscow           | RUS     |    8389200 |
| 2454 | Macao            | MAC     |     437500 |
|  608 | Cairo            | EGY     |    6789479 |
| 2515 | Ciudad de México | MEX     |    8591309 |
|  206 | São Paulo        | BRA     |    9968485 |
|  130 | Sydney           | AUS     |    3276207 |
+------+------------------+---------+------------+
```

Factsheet

Semi-join materialization

- Can be used for uncorrelated IN-subqueries. The subselect may use grouping and/or aggregate functions.
- Is shown in `EXPLAIN` as `type=MATERIALIZED` for the subquery, and a line with `table=<subqueryN>` in the parent subquery.

- Is enabled when one has both `materialization=on` and `semijoin=on` in the optimizer_switch variable.
- The `materialization=on|off` flag is shared with Non-semijoin materialization.

This page is licensed: CC BY-SA / Gnu FDL

Optimizations for Derived Tables

See Condition Pushdown into Derived Table Optimization.

Condition Pushdown pushes a parent query's conditions into a derived table or view that cannot be merged, letting the optimizer build efficient access paths.

See Derived Table Merge Optimization.

Derived Table Merge folds a FROM-clause subquery or view into its parent SELECT when it has no grouping, aggregates, or ORDER BY … LIMIT, avoiding a temporary table.

See Derived Table with Key Optimization.

Derived Table with Key lets a materialized derived table have one index that the optimizer uses for joins with other tables instead of a full table scan.

See Lateral Derived Optimization.

Documents Lateral Derived Optimization, also referred to as Split Grouping Optimization or Split Materialized Optimization.

See Split Materialized Optimization.

Split Materialized Optimization is another name for the Lateral Derived optimization.

This page is licensed: CC BY-SA / Gnu FDL

Condition Pushdown into Derived Table Optimization

If a query uses a derived table (or a view), the first action that the query optimizer will attempt is to apply the derived-table-merge-optimization and merge the derived table into its parent select. However, that optimization is only applicable when the select inside the derived table has a join as the top-level operation. If it has a GROUP-BY, DISTINCT, or uses window functions, then derived-table-merge-optimization is not applicable.

In that case, the Condition Pushdown optimization is applicable.

Introduction to Condition Pushdown

Consider an example

```
CREATE VIEW OCT_TOTALS AS
SELECT
  customer_id,
  SUM(amount) AS TOTAL_AMT
FROM orders
WHERE  order_date BETWEEN '2017-10-01' AND '2017-10-31'
GROUP BY customer_id;

SELECT * FROM OCT_TOTALS WHERE customer_id=1
```

The naive way to execute the above is to

1. Compute the OCT_TOTALS contents (for all customers).
2. The, select the line with customer_id=1

This is obviously inefficient, if there are 1000 customers, then one will be doing up to 1000 times more work than necessary.

However, the optimizer can take the condition `customer_id=1` and push it down into the OCT_TOTALS view.

Inside the OCT_\TOTALS, the added condition is put into its HAVING clause, so we end up with:

```
SELECT
  customer_id,
  SUM(amount) AS TOTAL_AMT
FROM orders
WHERE  order_date BETWEEN '2017-10-01' AND '2017-10-31'
GROUP BY customer_id
HAVING
  customer_id=1
```

Then, parts of HAVING clause that refer to GROUP BY columns are moved into the WHERE clause:

```
SELECT
  customer_id,
  SUM(amount) AS TOTAL_AMT
FROM orders
WHERE
  order_date BETWEEN '2017-10-01' AND '2017-10-31' AND
  customer_id=1
GROUP BY customer_id
```

Once a restriction like `customer_id=1` is in the WHERE, the query optimizer can use it to construct efficient table access paths.

Controlling the Optimization

The optimization is enabled by default. One can disable it by setting the `optimizer_switch` flag `condition_pushdown_for_derived` to OFF.

The pushdown from HAVING to WHERE part is controlled by `condition_pushdown_from_having` flag in `optimizer_switch`.

> **CURRENT**
>
> From MariaDB 12.1, it is possible to enable or disable the optimization with DERIVED_CONDITION_PUSHDOWN() and NO_DERIVED_CONDITION_PUSHDOWN() optimizer hints.
>
> **<12.1**
>
> No optimizer hint is available.

See Also

- Condition Pushdown through Window Functions (since MariaDB 10.3)
- Condition Pushdown into IN Subqueries (since MariaDB 10.4)
- The Jira task for the feature is MDEV-9197.

This page is licensed: CC BY-SA / Gnu FDL

Derived Table Merge Optimization

Background

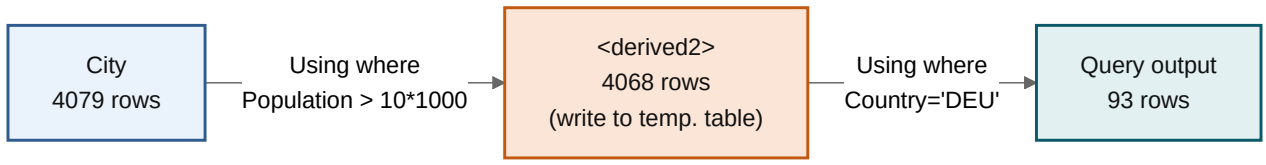
Users of “big” database systems are used to using `FROM` subqueries as a way to structure their queries. For example, if one’s first thought was to select cities with population greater than 10,000 people, and then that from these cities to select those that are located in Germany, one could write this SQL:

```
SELECT *
FROM
  (SELECT * FROM City WHERE Population > 10*1000) AS big_city
WHERE
  big_city.Country='DEU'
```

For MySQL, using such syntax was taboo. If you run EXPLAIN for this query, you can see why:

```
mysql> EXPLAIN SELECT * FROM (SELECT * FROM City WHERE Population > 1*1000)
  AS big_city WHERE big_city.Country='DEU' ;
+----+-------------+------------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table      | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+------------+------+---------------+------+---------+------+------+-------------+
|  1 | PRIMARY     | <derived2> | ALL  | NULL          | NULL | NULL    | NULL | 4068 | Using where |
|  2 | DERIVED     | City       | ALL  | Population    | NULL | NULL    | NULL | 4079 | Using where |
+----+-------------+------------+------+---------------+------+---------+------+------+-------------+
2 rows in set (0.60 sec)
```

It plans to do the following actions:



City rows are filtered on `Population`, written to the `derived2` temporary table, then filtered again on `Country` to produce the query output.

From left to right:

1. Execute the subquery: `(SELECT * FROM City WHERE Population > 1*1000)`, exactly as it was written in the query.
2. Put result of the subquery into a temporary table.
3. Read back, and apply a `WHERE` condition from the upper select, `big_city.Country='DEU'`

Executing a subquery like this is very inefficient, because the highly-selective condition from the parent select, (`Country='DEU'`) is not used when scanning the base table `City`. We read too many records from the `City` table, and then we have to write them into a temporary table and read them back again, before finally filtering them out.

Derived table merge in action

If one runs this query in MariaDB/MySQL 5.6, they get this:

```
MariaDB [world]> EXPLAIN SELECT * FROM (SELECT * FROM City WHERE Population > 1*1000)
  AS big_city WHERE big_city.Country='DEU';
+----+-------------+-------+------+--------------------+---------+---------+-------+------+-------------
----------------------+
| id | select_type | table | type | possible_keys      | key     | key_len | ref   | rows | Extra
|
+----+-------------+-------+------+--------------------+---------+---------+-------+------+-------------
----------------------+
|  1 | SIMPLE      | City  | ref  | Population,Country | Country | 3       | const |   90 | Using index
condition; Using where |
+----+-------------+-------+------+--------------------+---------+---------+-------+------+-------------
----------------------+
1 row in set (0.00 sec)
```

From the above, one can see that:

1. The output has only one line. This means that the subquery has been merged into the top-level `SELECT`.
2. Table `City` is accessed through an index on the `Country` column. Apparently, the `Country='DEU'` condition was used to construct `ref` access on the table.
3. The query will read about 90 rows, which is a big improvement over the 4079 row reads plus 4068 temporary table reads/writes we had before.

Factsheet

- Derived tables (subqueries in the `FROM` clause) can be merged into their parent select when they have no grouping, aggregates, or `ORDER BY ... LIMIT` clauses. These requirements are the same as requirements for `VIEW`s to allow `algorithm=merge`.
- The optimization is enabled by default. It can be disabled with:

```
SET @@optimizer_switch='derived_merge=OFF'
```

- From MariaDB 12.1, it is possible to enable or disable the optimization with MERGE() and NO_MERGE() optimizer hints.
- Versions of MySQL and MariaDB which do not have support for this optimization will execute subqueries even when running `EXPLAIN`. This can result in a well-known problem (see e.g. MySQL Bug #44802) of `EXPLAIN` statements taking a very long time. Starting from MariaDB 5.3+ and MySQL 5.6+ `EXPLAIN` commands execute instantly, regardless of the `derived_merge` setting.

See Also

- FAQ entry: Why is ORDER BY in a FROM subquery ignored?

This page is licensed: CC BY-SA / Gnu FDL

Derived Table with Key Optimization

The idea

If a derived table cannot be merged into its parent SELECT, it will be materialized in a temporary table, and then parent select will treat it as a regular base table.

Before MariaDB 5.3/MySQL 5.6, the temporary table would never have any indexes, and the only way to read records from it would be a full table scan. Starting from the mentioned versions of the server, the optimizer has an option to create an index and use it for joins with other tables.

Example

Consider a query: we want to find countries in Europe, that have more than one million people living in cities. This is accomplished with this query:

```
SELECT *
FROM
   Country,
   (SELECT
       SUM(City.Population) AS urban_population,
       City.Country
    FROM City
    GROUP BY City.Country
    HAVING
    urban_population > 1*1000*1000
   ) AS cities_in_country
WHERE
  Country.Code=cities_in_country.Country AND Country.Continent='Europe';
```

The EXPLAIN output for it will show:

```
+----+-------------+------------+------+------------------+-----------+---------+--------------------+--
----+---------------------------------+
| id | select_type | table      | type | possible_keys    | key       | key_len | ref                |
rows | Extra                           |
+----+-------------+------------+------+------------------+-----------+---------+--------------------+--
----+---------------------------------+
|  1 | PRIMARY     | Country    | ref  | PRIMARY,continent | continent | 17      | const              |
60 | Using index condition           |
|  1 | PRIMARY     | <derived2> | ref  | key0             | key0      | 3       | world.Country.Code |
17 |                                 |
|  2 | DERIVED     | City       | ALL  | NULL             | NULL      | NULL    | NULL               |
4079 | Using temporary; Using filesort |
+----+-------------+------------+------+------------------+-----------+---------+--------------------+--
----+---------------------------------+
```

One can see here that

- table `<derived2>` is accessed through `key0`.
- `ref` column shows `world.Country.Code`
- if we look that up in the original query, we find the equality that was used to construct `ref` access: `Country.Code=cities_in_country.Country`.

Factsheet

- The idea of “derived table with key” optimization is to let the materialized derived table have one key which is used for joins with other tables.
- The optimization is applied then the derived table could not be merged into its parent SELECT
  - which happens when the derived table doesn’t meet criteria for mergeable VIEW
- The optimization is ON by default; it can be switched off like so:

```
SET optimizer_switch='derived_with_keys=off'
```

See Also

- Optimizing Subqueries in the FROM Clause in MySQL 5.6 manual
- What is MariaDB 5.3
- Subquery Optimizations Map

This page is licensed: CC BY-SA / Gnu FDL

Lateral Derived Optimization

MariaDB supports the *Lateral Derived* optimization, also referred to as *Split Grouping Optimization* or *Split Materialized Optimization*.

Description

The optimization use cases are:

- The query uses a derived table (or a `VIEW`, or a non-recursive CTE).
- The derived table, view, or CTE has a `GROUP BY` operation as its top-level operation.
- The query only needs data from a few `GROUP BY` groups.

Example: A `VIEW` that computes totals for each customer in October:

```
CREATE VIEW OCT_TOTALS AS
SELECT
  customer_id,
  SUM(amount) AS TOTAL_AMT
FROM orders
WHERE
  order_date BETWEEN '2017-10-01' AND '2017-10-31'
GROUP BY
  customer_id;
```

Example: A query that does a join with the `customer` table to get October totals for `Customer#1` and `Customer#2`:

```
SELECT *
FROM
  customer, OCT_TOTALS
WHERE
  customer.customer_id=OCT_TOTALS.customer_id AND
  customer.customer_name IN ('Customer#1', 'Customer#2')
```

Before Lateral Derived optimization, MariaDB executed the query as follows:

1. Materialize the view `OCT_TOTALS`. This essentially computes `OCT_TOTALS` for all customers.
2. Join it with table customer.

The `EXPLAIN` looked like this:

```
+------+-------------+------------+-------+---------------+-----------+---------+-------------------------
---+-------+--------------------------+
| id   | select_type | table      | type  | possible_keys | key       | key_len | ref
| rows  | Extra                     |
+------+-------------+------------+-------+---------------+-----------+---------+-------------------------
---+-------+--------------------------+
|    1 | PRIMARY     | customer   | range | PRIMARY,name  | name      | 103     | NULL
| 2     | Using where; Using index |
|    1 | PRIMARY     | <derived2> | ref   | key0          | key0      | 4       |
test.customer.customer_id | 36    |                           |
|    2 | DERIVED     | orders     | index | NULL          | o_cust_id | 4       | NULL
| 36738 | Using where              |
+------+-------------+------------+-------+---------------+-----------+---------+-------------------------
---+-------+--------------------------+
```

It is obvious that step #1 is inefficient: We compute totals for all customers in the database, while we will only need them for two customers. (If there are 1000 customers, we are doing 500x more work than needed here.)

Lateral Derived Optimization addresses this case. It turns the computation of `OCT_TOTALS` into what SQL Standard refers to as "`LATERAL` subquery": a subquery that may have dependencies on the outside tables. This allows

pushing the equality `customer.customer_id=OCT_TOTALS.customer_id` down into the derived table/view, where it can be used to limit the computation to compute totals only for the customer of interest.

The query plan looks as follows:

1. Scan table `customer` and find `customer_id` for `Customer#1` and `Customer#2`.
2. For each `customer_id`, compute the October totals, for this specific customer.

The `EXPLAIN` output looks like this:

```
+------+-----------------+------------+-------+---------------+-----------+---------+--------------------
-------+------+-------------------------+
| id   | select_type     | table      | type  | possible_keys | key       | key_len | ref
| rows | Extra                    |
+------+-----------------+------------+-------+---------------+-----------+---------+--------------------
-------+------+-------------------------+
|    1 | PRIMARY         | customer   | range | PRIMARY,name  | name      | 103     | NULL
| 2    | Using where; Using index |
|    1 | PRIMARY         | <derived2> | ref   | key0          | key0      | 4       |
test.customer.customer_id | 2    |                           |
|    2 | LATERAL DERIVED | orders     | ref   | o_cust_id     | o_cust_id | 4       |
test.customer.customer_id | 1    | Using where               |
+------+-----------------+------------+-------+---------------+-----------+---------+--------------------
-------+------+-------------------------+
```

Note the line with `id=2`: select_type is `LATERAL DERIVED`. And table `customer` uses ref access referring to `customer.customer_id`, which is normally not allowed for derived tables.

In `EXPLAIN FORMAT=JSON` output, the optimization is shown like this:

```
...
        "table": {
          "table_name": "<derived2>",
          "access_type": "ref",
...
          "materialized": {
            "lateral": 1,
```

Note the `"lateral": 1` member.

Controlling the Optimization

Lateral Derived is enabled by default. The optimizer makes a cost-based decision whether the optimization should be used.

If you need to disable the optimization, it has an optimizer_switch flag. It can be disabled like this:

```
SET optimizer_switch='split_materialized=off'
```

> **CURRENT**
>
> It is possible to enable or disable the optimization with optimizer hints SPLIT_MATERIALIZED or NO_SPLIT_MATERIALIZED.
>
> For example, by default, this table and query makes use of the optimization:

```
CREATE TABLE t1 ( n1 INT(10) NOT NULL, n2 INT(10) NOT NULL, c1 CHAR(1) NOT NULL, KEY c1 (c1), KEY
n1_c1_n2 (n1,c1,n2) ) ENGINE=innodb CHARSET=latin1;
INSERT INTO t1 VALUES (0, 2, 'a'), (1, 3, 'a');
INSERT INTO t1 SELECT seq+1,seq+2,'c' FROM seq_1_to_1000;
ANALYZE TABLE t1;
EXPLAIN SELECT t1.n1 FROM t1, (SELECT n1, n2 FROM t1 WHERE c1 = 'a' GROUP BY n1) AS t WHERE t.n1 = t1.n1
AND t.n2 = t1.n2 AND c1 = 'a' GROUP BY n1\G
```

```
*************************** 1\. row ***************************
           id: 1
  select_type: PRIMARY
        table: t1
         type: ref
possible_keys: c1,n1_c1_n2
          key: c1
      key_len: 1
          ref: const
         rows: 2
        Extra: Using index condition; Using where; Using temporary; Using filesort
*************************** 2\. row ***************************
           id: 1
  select_type: PRIMARY
        table: <derived2>
         type: ref
possible_keys: key0
          key: key0
      key_len: 8
          ref: test.t1.n1,test.t1.n2
         rows: 1
        Extra:
*************************** 3\. row ***************************
           id: 2
  select_type: LATERAL DERIVED
        table: t1
         type: ref
possible_keys: c1,n1_c1_n2
          key: n1_c1_n2
      key_len: 4
          ref: test.t1.n1
         rows: 1
        Extra: Using where; Using index
```

The optimization can be disabled like this:

```
EXPLAIN SELECT /*+ NO_SPLIT_MATERIALIZED(t) */ t1.n1 FROM t1, (SELECT n1, n2 FROM t1 WHERE c1 = 'a'
GROUP BY n1) AS t WHERE t.n1 = t1.n1 AND t.n2 = t1.n2 AND c1 = 'a' GROUP BY n1 \G

*************_*_************ 1\. row ************_*_**************          id: 1
  select_type: PRIMARY
        table: t1
         type: ref
possible_keys: c1,n1_c1_n2
          key: c1
      key_len: 1
          ref: const
         rows: 2
        Extra: Using index condition; Using where; Using temporary; Using filesort
************_*_************ 2\. row ************_*_************
           id: 1
  select_type: PRIMARY
        table: <derived2>
         type: ref
possible_keys: key0
          key: key0
      key_len: 8
          ref: test.t1.n1,test.t1.n2
         rows: 1
        Extra:
************_*_************ 3\. row ************_*_************
           id: 2
  select_type: DERIVED
        table: t1
         type: ref
possible_keys: c1
          key: c1
      key_len: 1
          ref: const
         rows: 2
        Extra: Using index condition; Using where; Using temporary; Using filesort
```

< 12.1

No such optimizer hint is available.

References

- Jira task: MDEV-13369
- Commit: b14e2b044b

This page is licensed: CC BY-SA / Gnu FDL

Split Materialized Optimization

This is another name for Lateral Derived Optimization.

This page is licensed: CC BY-SA / Gnu FDL

Statistics for Optimizing Queries

See Engine-Independent Table Statistics.

Engine-independent table statistics store optimizer statistics in mysql database tables, collected via ANALYZE TABLE and controlled by the use_stat_tables variable.

See Histogram-Based Statistics.

Histogram-based statistics give the optimizer value-distribution data for indexed and non-indexed columns, improving query plans through better selectivity estimates.

See InnoDB Persistent Statistics.

InnoDB persistent statistics store index and table statistics on disk so they survive server restarts, controlled by innodb_stats_persistent and related variables.

See Slow Query Log Extended Statistics.

Extends the slow query log with query-plan, EXPLAIN, and storage-engine statistics, controlled by log_slow_verbosity, log_slow_filter, and log_slow_rate_limit.

See User Statistics.

The userstat plugin adds USER, CLIENT, INDEX, and TABLE statistics as INFORMATION_SCHEMA tables plus SHOW and FLUSH statements to track server activity and load.

This page is licensed: CC BY-SA / Gnu FDL

Engine-Independent Table Statistics

Introduction

The MySQL/MariaDB optimizer prior to MariaDB 10 used to rely on storage engines (e.g. InnoDB) to provide statistics for the query optimizer. This approach worked; however it had some deficiencies:

- Storage engines provided poor statistics (this was fixed to some degree with the introduction of Persistent Statistics).
- The statistics were supplied through the MySQL Storage Engine Interface, which puts a lot of restrictions on what kind of data is supplied (for example, there is no way to get any data about value distribution in a non-indexed column)
- There was little control of the statistics. There was no way to "pin" current statistic values, or provide values on your own, etc.

Engine-independent table statistics lift these limitations:

- Statistics are stored in regular tables in the `mysql` database. You can read and update the values.
- More data is collected/used.

Histogram-based statistics are a subset of engine-independent table statistics (EITS) that can improve the query plan chosen by the optimizer in certain situations.

Statistics are stored in three tables, mysql.table_stats, mysql.column_stats and mysql.index_stats.

Usage or updating of data from these tables is controlled by use_stat_tables variable. Possible values are:

Value	Meaning
'never'	The optimizer doesn't use data from statistics tables.
'complementary'	The optimizer uses data from statistics tables if the same kind of data is not provided by the storage engine.
'preferably'	Prefer the data from statistics tables, if it's not available there, use the data from the storage engine.
'complementary_for_queries'	Same as complementary, but for queries only (to avoid needlessly collecting for ANALYZE TABLE).
'preferably_for_queries'	Same as preferably, but for queries only (to avoid needlessly collecting for ANALYZE TABLE). Default.

Collecting Statistics with the ANALYZE TABLE Statement

Engine-independent statistics are collected by doing full table and full index scans, and this process can be quite expensive.

The ANALYZE TABLE statement can be used to collect table statistics. However, simply running `ANALYZE TABLE table_name` does not collect engine-independent (or histogram) statistics by default.

When the ANALYZE TABLE statement is executed, MariaDB makes a call to the table's storage engine, and the storage engine collects its own statistics for the table. The specific behavior depends on the storage engine. For the default InnoDB storage engine, see InnoDB Persistent Statistics for more information.

ANALYZE TABLE may also collect engine-independent statistics for the table. The specific behavior depends on the value of the use_stat_tables system variable. Engine-independent statistics will only be collected if one of the following is true:

- The use_stat_tables system variable is set to `complementary` or `preferably`.
- The ANALYZE TABLE statement includes the `PERSISTENT FOR` clause.

The use_stat_tables system variable is set to `preferably_for_queries` by default. With this value, engine-independent statistics are used by default if available, but they are not collected by default. If you want to use engine-independent statistics with the default configuration, then you will have to collect them by executing the ANALYZE TABLE statement and by specifying the `PERSISTENT FOR` clause. It is recommended to collect engine-independent statistics on as-needed basis, so typically one will not have engine-independent statistics for all indexes/all columns.

When to collect statistics is very dependent on the dataset. If data changes frequently it may be necessary to collect statistics more frequently, and the benefits may be very noticeable (see This one trick can make MariaDB 30x faster!). If the data distribution is relatively static, the costs of collecting may outweigh any benefits.

Collecting Statistics for Specific Columns or Indexes

The syntax for the ANALYZE TABLE statement has been extended with the `PERSISTENT FOR` clause. This clause allows one to collect engine-independent statistics only for particular columns or indexes. This clause also allows one to collect engine-independent statistics, regardless of the value of the use_stat_tables system variable. For example:

```
ANALYZE TABLE table_name PERSISTENT FOR ALL;
```

Statistics for columns using the BLOB and TEXT data types are not collected. If a column using one of these types is explicitly specified, then a warning is returned.

Examples of Statistics Collection

```
-- update all engine-independent statistics for all columns and indexes
ANALYZE TABLE tbl PERSISTENT FOR ALL;

-- update specific columns and indexes:
ANALYZE TABLE tbl PERSISTENT FOR COLUMNS (col1,col2,...) INDEXES (idx1,idx2,...);

-- empty lists are allowed:
ANALYZE TABLE tbl PERSISTENT FOR COLUMNS (col1,col2,...) INDEXES ();
ANALYZE TABLE tbl PERSISTENT FOR COLUMNS () INDEXES (idx1,idx2,...);

-- the following will only update mysql.table_stats fields:
ANALYZE TABLE tbl PERSISTENT FOR COLUMNS () INDEXES ();

-- when use_stat_tables is set to 'COMPLEMENTARY' or 'PREFERABLY',
-- a simple ANALYZE TABLE  collects engine-independent statistics for all columns and indexes.
SET SESSION use_stat_tables='COMPLEMENTARY';
ANALYZE TABLE tbl;
```

Manual Updates to Statistics Tables

Statistics are stored in three tables, mysql.table_stats, mysql.column_stats and mysql.index_stats.

It is possible to update statistics tables manually. One should modify the table(s) with regular INSERT/UPDATE/DELETE statements. Statistics data will be re-read when the tables are re-opened. One way to force all tables to be re-opened is to issue FLUSH TABLES command.

A few scenarios where one might need to update statistics tables manually:

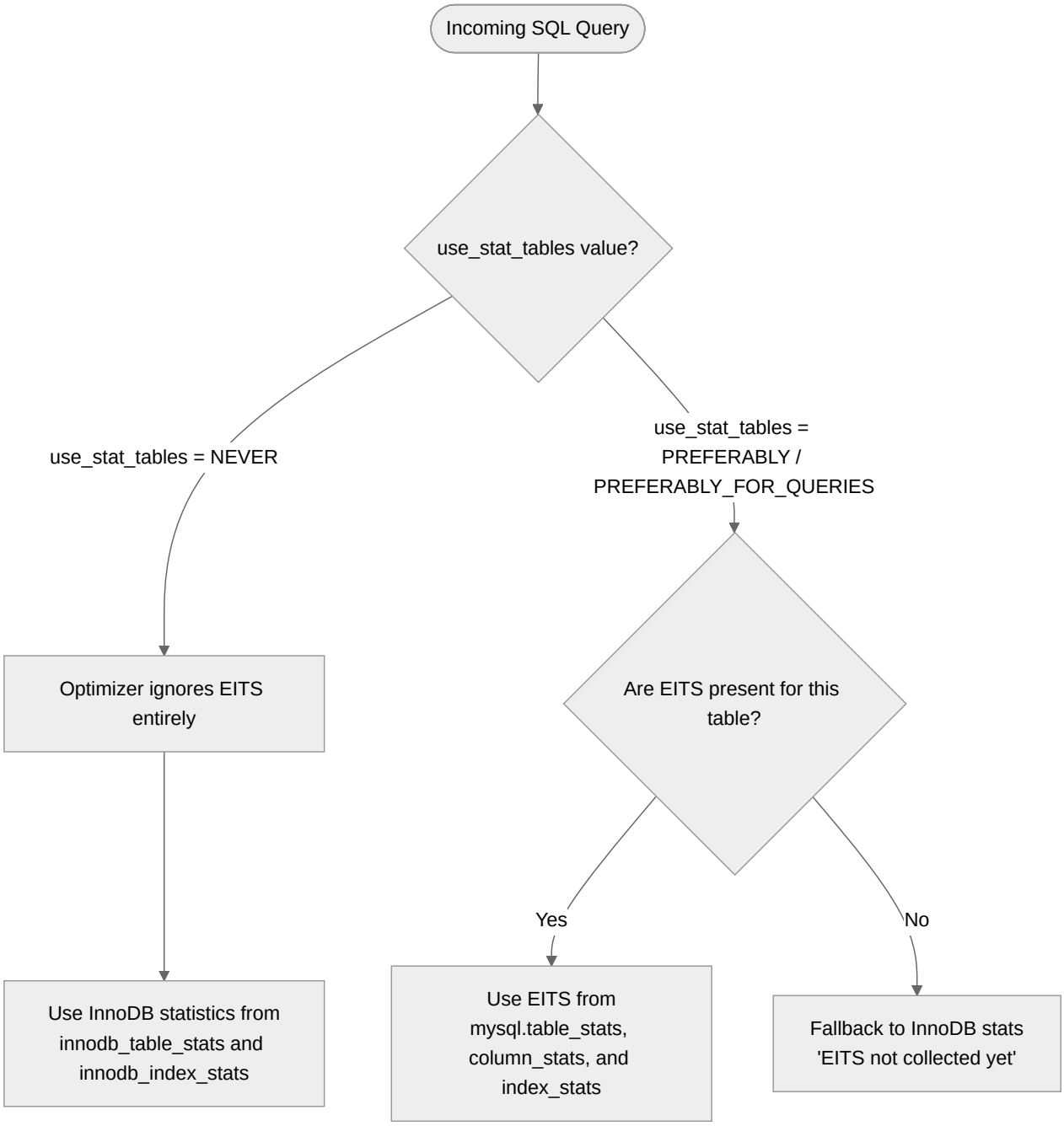
- Deleting the statistics. Currently, the ANALYZE TABLE command will collect the statistics, but there is no special command to delete statistics.
- Running `ANALYZE` on a different server. To collect engine-independent statistics ANALYZE TABLE does a full table scan, which can put too much load on the server. It is possible to run ANALYZE on the slave, and then take the data from statistics tables on the slave and apply it on the master.
- In some cases, knowledge of the database allows one to compute statistics manually in a more efficient way than `ANALYZE` does. One can compute the statistics manually and put it into the database.

EITS[30] vs. InnoDB Statistics

This section visually explains how MariaDB decides which statistics to use, and what happens during the `ANALYZE` stage. These flows are critical for understanding why toggling `use_stat_tables` affects both performance and optimizer behavior.

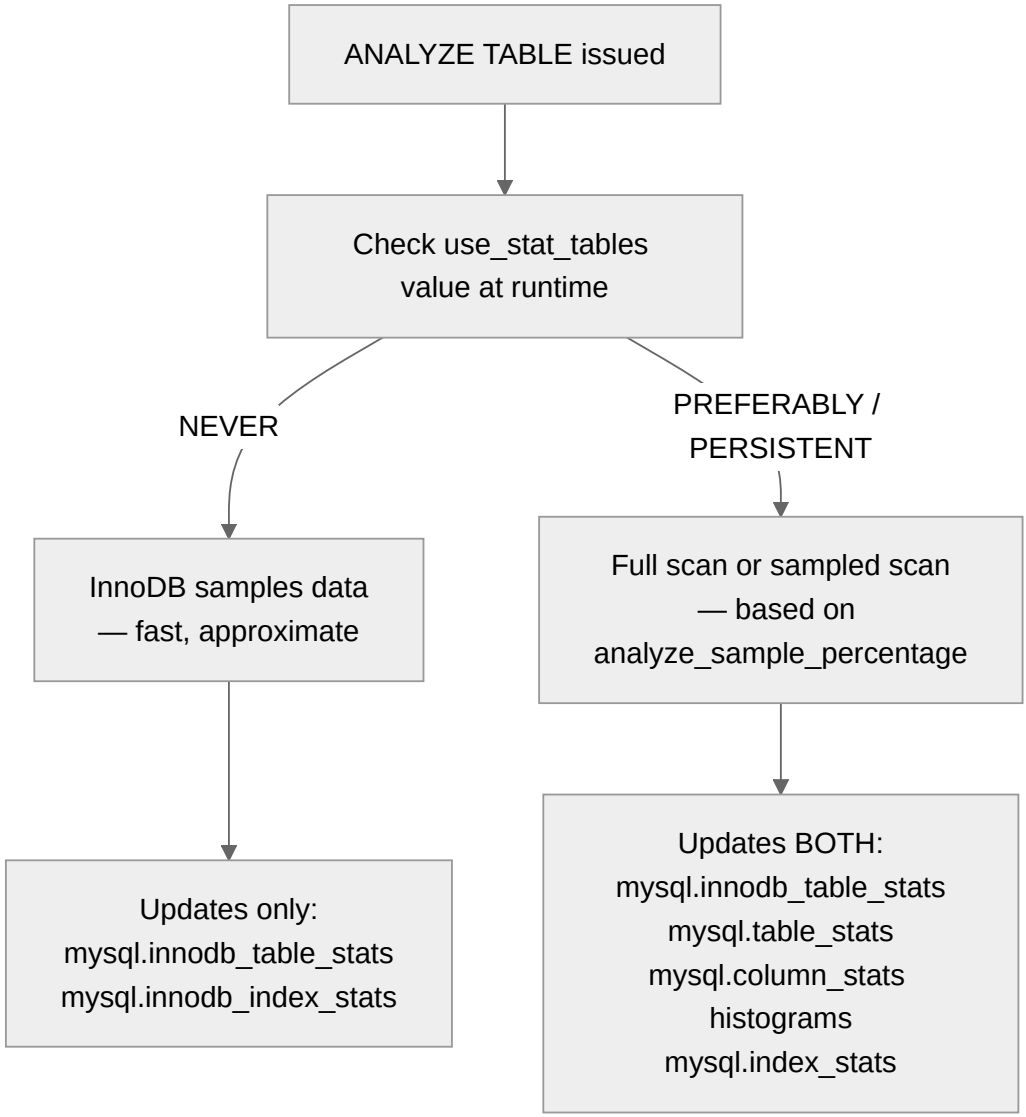
> **Note:**
>
> **MariaDB 10.6 and later** include several optimizer enhancements that affect how these statistics are utilized. These improvements include:
>
> - More accurate cost-based optimization decisions
> - Improved index and condition selection
> - Enhanced selectivity estimation during `ANALYZE` operations

Optimizer Statistics Selection Flow (Query Execution Time)



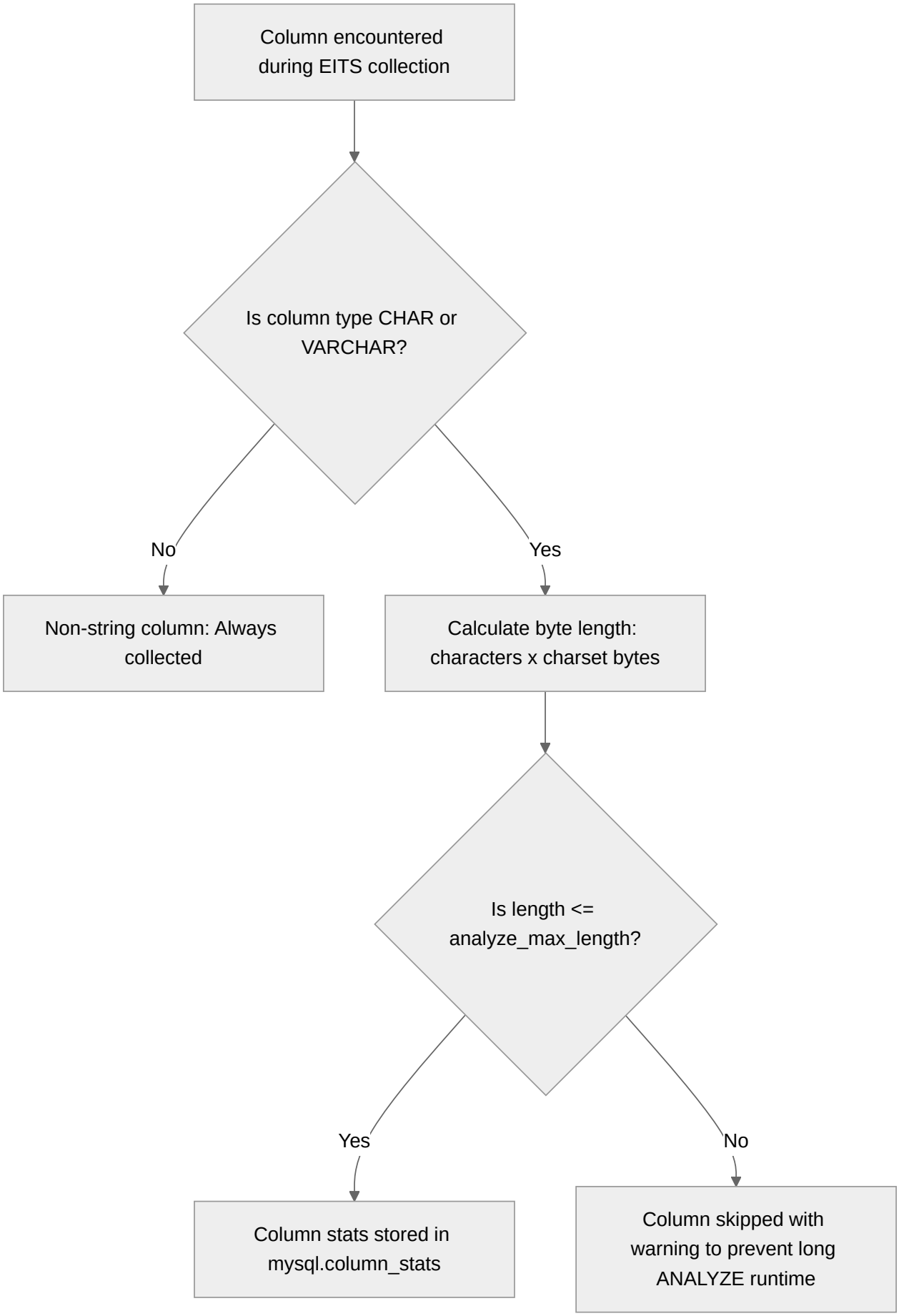
- NEVER: Optimizer always uses InnoDB stats, even if EITS exists.
- PREFERABLY: Optimizer prefers EITS and transparently falls back if missing.

ANALYZE TABLE – Statistics Collection Flow



- NEVER: Fast, safe, but low precision.
- PERSISTENT: Explicit, predictable, and recommended.

Column Statistics Collection Flow (analyze_max_length)



- `utf8mb4` multiplies size by 4 (compared to `latin1`)
- `VARCHAR (255)` ≈ 1020 bytes
- `analyze_max_length` = 2048 is a safe global default.

Scenario / Environment	Recommended Mode	Why This Works
Small DB (<100 GB), simple queries	NEVER	InnoDB stats are sufficient and fastest to maintain
Mixed workloads, general production	PREFERABLY_FOR_QUERIES	Safe default: optimizer uses EITS when present, ANALYZE is explicit
Analytics / reporting heavy	PREFERABLY	Ensures rich column & index stats for complex joins
Large DB (1–10 TB), 24/7 traffic	PREFERABLY_FOR_QUERIES + scheduled ANALYZE TABLE PERSISTENT	Avoids accidental heavy ANALYZE during peak hours
SaaS / multi-tenant systems	PREFERABLY_FOR_QUERIES	Predictable behavior across thousands of tables
Highly volatile tables (frequent writes)	NEVER or selective PERSISTENT	EITS can go stale quickly on hot tables
Troubleshooting bad query plans	PREFERABLY (temporarily)	Forces optimizer to use detailed stats
Upgrade from MariaDB 10.6	Start with PREFERABLY_FOR_QUERIES	Minimizes plan regressions

See Also

- Index Statistics
- InnoDB Persistent Statistics
- Histogram-based Statistics
- JSON histograms (mariadb.org blog)
- Improving MariaDB's optimizer with better selectivity estimates - Sergey Petrunia - Server Fest 2021 (video)

This page is licensed: CC BY-SA / Gnu FDL

Histogram-Based Statistics

Histogram-based statistics are a mechanism to improve the query plan chosen by the optimizer in certain situations. Before their introduction, all conditions on non-indexed columns were ignored when searching for the best execution plan. Histograms can be collected for both indexed and non-indexed columns, and are made available to the optimizer.

Histogram statistics are stored in the mysql.column_stats table, which stores data for engine-independent table statistics, and so are essentially a subset of engine-independent table statistics.

Histograms are used by default from MariaDB 10.4.3 if they are available. However, histogram statistics are not automatically collected, as collection is expensive, requiring a full table scan. See Collecting Statistics with the ANALYZE TABLE Statement for details.

Consider this example, using the following query:

```
SELECT * FROM t1,t2 WHERE t1.a=t2.a AND t2.b BETWEEN 1 AND 3;
```

Let's assume that

- table t1 contains 100 records
- table t2 contains 1000 records

- there is a primary index on t1(a)
- there is a secondary index on t2(a)
- there is no index defined on column t2.b
- the selectivity of the condition t2.b BETWEEN (1,3) is high (~ 1%)

Before histograms were introduced, the optimizer would choose the plan that:

- accesses t1 using a table scan
- accesses t2 using index t2(a)
- checks the condition t2.b BETWEEN 1 AND 3

This plan examines all rows of both tables and performs 100 index look-ups.

With histograms available, the optimizer can choose the following, more efficient plan:

- accesses table t2 in a table scan
- checks the condition t2.b BETWEEN 1 AND 3
- accesses t1 using index t1(a)

This plan also examine all rows from t2, but it performs only 10 look-ups to access 10 rows of table t1.

System Variables

There are a number of system variables that affect histograms.

histogram_size

The histogram_size variable determines the size, in bytes, from 0 to 255, used for a histogram. This is effectively the number of bins for `histogram_type=SINGLE_PREC_HB` or number of bins/2 for `histogram_type=DOUBLE_PREC_HB`. If it is set to 0 (the default for MariaDB 10.4.2 and below), no histograms are created when running an ANALYZE TABLE.

histogram_type

The histogram_type variable determines whether single precision (`SINGLE_PREC_HB`) or double precision (`DOUBLE_PREC_HB`) height-balanced histograms are created. From MariaDB 10.4.3, double precision is the default. For MariaDB 10.4.2 and below, single precision is the default.

From MariaDB 10.8, `JSON_HB`, JSON-format histograms, are accepted.

optimizer_use_condition_selectivity

The optimizer_use_condition_selectivity controls which statistics can be used by the optimizer when looking for the best query execution plan.

- `1` Use selectivity of predicates as in MariaDB 5.5.
- `2` Use selectivity of all range predicates supported by indexes.
- `3` Use selectivity of all range predicates estimated without histogram.
- `4` Use selectivity of all range predicates estimated with histogram.
- `5` Additionally use selectivity of certain non-range predicates calculated on record sample.

From MariaDB 10.4.1, the default is `4`. Until MariaDB 10.4.0, the default is `1`.

Example

Here is an example of the dramatic impact histogram-based statistics can make. The query is based on DBT3 Benchmark Q20 with 60 million records in the `lineitem` table.

```
SELECT SQL_CALC_FOUND_ROWS s_name, s_address FROM
supplier, nation WHERE
  s_suppkey IN
    (SELECT ps_suppkey FROM partsupp WHERE
      ps_partkey IN (SELECT p_partkey FROM part WHERE
         p_name LIKE 'forest%') AND
    ps_availqty >
      (SELECT 0.5 * SUM(l_quantity) FROM lineitem WHERE
        l_partkey = ps_partkey AND l_suppkey = ps_suppkey AND
        l_shipdate >= DATE('1994-01-01') AND
        l_shipdate < DATE('1994-01-01') + INTERVAL '1' YEAR ))
  AND s_nationkey = n_nationkey
  AND n_name = 'CANADA'
  ORDER BY s_name
  LIMIT 10;
```

First,

```
SET optimizer_switch='materialization=off,semijoin=off';
```

```
+---+-------- +----------+-------+...+------+----------+------------
| id| sel_type| table    | type  |...| rows | filt | Extra
+---+-------- +----------+-------+...+------+----------+------------
| 1 | PRIMARY | nation   | ALL   |...| 25   |100.00 | Using where;...
| 1 | PRIMARY | supplier | ref   |...| 1447 |100.00 | Using where; Subq
| 2 | DEP SUBQ| partsupp | idxsq |...| 38   |100.00 | Using where
| 4 | DEP SUBQ| lineitem | ref   |...| 3    |100.00 | Using where
| 3 | DEP SUBQ| part     | unqsb |...| 1    |100.00 | Using where
+---+-------- +----------+-------+...+------+----------+------------

10 ROWS IN SET
(51.78 sec)
```

Next, a really bad plan, yet one sometimes chosen:

```
+---+-------- +----------+-------+...+------+----------+------------
| id| sel_type| table    | type  |...| rows | filt | Extra
+---+-------- +----------+-------+...+------+----------+------------
| 1 | PRIMARY | supplier | ALL   |...|100381|100.00 | Using where; Subq
| 1 | PRIMARY | nation   | ref   |...| 1    |100.00 | Using where
| 2 | DEP SUBQ| partsupp | idxsq |...| 38   |100.00 | Using where
| 4 | DEP SUBQ| lineitem | ref   |...| 3    |100.00 | Using where
| 3 | DEP SUBQ| part     | unqsb |...| 1    |100.00 | Using where
+---+-------- +----------+-------+...+------+----------+------------

10 ROWS IN SET
(7 min 33.42 sec)
```

Persistent statistics don't improve matters:

```
SET use_stat_tables='preferably';
+---+-------- +----------+-------+...+------+----------+------------
| id| sel_type| table    | type  |...| rows | filt | Extra
+---+-------- +----------+-------+...+------+----------+------------
| 1 | PRIMARY | supplier | ALL   |...|10000 |100.00 | Using where;
| 1 | PRIMARY | nation   | ref   |...| 1    |100.00 | Using where
| 2 | DEP SUBQ| partsupp | idxsq |...| 80   |100.00 | Using where
| 4 | DEP SUBQ| lineitem | ref   |...| 7    |100.00 | Using where
| 3 | DEP SUBQ| part     | unqsb |...| 1    |100.00 | Using where
+---+-------- +----------+-------+...+------+----------+------------

10 ROWS IN SET
(7 min 40.44 sec)
```

The default flags for optimizer_switch do not help much:

```
SET optimizer_switch='materialization=DEFAULT,semijoin=DEFAULT';
+---+-------- +----------+-------+...+------+----------+------------
| id| sel_type| table    | type  |...| rows  | filt  | Extra
+---+-------- +----------+-------+...+------+----------+------------
| 1 | PRIMARY | supplier | ALL   |...|10000  |100.00 | Using where;
| 1 | PRIMARY | nation   | ref   |...| 1     |100.00 | Using where
| 1 | PRIMARY | <subq2>  | eq_ref|...| 1     |100.00 |
| 2 | MATER   | part     | ALL   |.. |2000000|100.00 | Using where
| 2 | MATER   | partsupp | ref   |...| 4     |100.00 | Using where; Subq
| 4 | DEP SUBQ| lineitem | ref   |...| 7     |100.00 | Using where
+---+-------- +----------+-------+...+------+----------+------------

10 ROWS IN SET
(5 min 21.44 sec)
```

Using statistics doesn't help either:

```
SET optimizer_switch='materialization=DEFAULT,semijoin=DEFAULT';
SET optimizer_use_condition_selectivity=4;

+---+-------- +----------+-------+...+------+----------+------------
| id| sel_type| table    | type  |...| rows  | filt  | Extra
+---+-------- +----------+-------+...+------+----------+------------
| 1 | PRIMARY | nation   | ALL   |...| 25    |4.00   | Using where
| 1 | PRIMARY | supplier | ref   |...| 4000  |100.00 | Using where;
| 1 | PRIMARY | <subq2>  | eq_ref|...| 1     |100.00 |
| 2 | MATER   | part     | ALL   |.. |2000000|1.56   | Using where
| 2 | MATER   | partsupp | ref   |...| 4     |100.00 | Using where; Subq
| 4 | DEP SUBQ| lineitem | ref   |...| 7     | 30.72 | Using where
+---+-------- +----------+-------+...+------+----------+------------

10 ROWS IN SET
(5 min 22.41 sec)
```

Now, taking into account the cost of the dependent subquery:

```
SET optimizer_switch='materialization=DEFAULT,semijoin=DEFAULT';
SET optimizer_use_condition_selectivity=4;
SET optimizer_switch='expensive_pred_static_pushdown=ON';
+---+-------- +----------+-------+...+------+----------+------------
| id| sel_type| table    | type  |...| rows | filt  | Extra
+---+-------- +----------+-------+...+------+----------+------------
| 1 | PRIMARY | nation   | ALL   |...| 25   | 4.00  | Using where
| 1 | PRIMARY | supplier | ref   |...| 4000 |100.00 | Using where;
| 2 | PRIMARY | partsupp | ref   |...| 80   |100.00 |
| 2 | PRIMARY | part     | eq_ref|...| 1    | 1.56  | where; Subq; FM
| 4 | DEP SUBQ| lineitem | ref   |...| 7    | 30.72 | Using where
+---+-------- +----------+-------+...+------+----------+------------

10 ROWS IN SET
(49.89 sec)
```

Finally, using join_buffer as well:

```
SET optimizer_switch= 'materialization=DEFAULT,semijoin=DEFAULT';
SET optimizer_use_condition_selectivity=4;
SET optimizer_switch='expensive_pred_static_pushdown=ON';
SET join_cache_level=6;
SET optimizer_switch='mrr=ON';
SET optimizer_switch='mrr_sort_keys=ON';
SET join_buffer_size=1024*1024*16;
SET join_buffer_space_limit=1024*1024*32;
+---+-------- +----------+-------+...+------+----------+------------
| id| sel_type| table    | type  |...| rows | filt |  Extra
+---+-------- +----------+-------+...+------+----------+------------
| 1 | PRIMARY | nation   | AL  L |...| 25   | 4.00  | Using where
| 1 | PRIMARY | supplier | ref   |...| 4000 |100.00 | where; BKA
| 2 | PRIMARY | partsupp | ref   |...| 80   |100.00 | BKA
| 2 | PRIMARY | part     | eq_ref|...| 1    | 1.56  | where Sq; FM; BKA
| 4 | DEP SUBQ| lineitem | ref   |...| 7    | 30.72 | Using where
+---+-------- +----------+-------+...+------+----------+------------

10 ROWS IN SET
(35.71 sec)
```

See Also

- DECODE_HISTOGRAM()
- Index Statistics
- InnoDB Persistent Statistics
- Engine-independent Statistics
- JSON Histograms (mariadb.org blog)
- Improved histograms in MariaDB 10.8 - Sergei Petrunia - FOSDEM 2022 (video)
- Improving MariaDB's optimizer with better selectivity estimates - Sergey Petrunia - Server Fest 2021 (video)

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Persistent Statistics

InnoDB statistics are stored on disk and are therefore persistent. Prior to MariaDB 10.0, InnoDB statistics were not stored on disk, meaning that on server restarts the statistics would need to be recalculated, which is both needless computation, as well as leading to inconsistent query plans.

There are a number of variables that control persistent statistics:

- innodb_stats_persistent - when set (the default) enables InnoDB persistent statistics.
- innodb_stats_auto_recalc - when set (the default), persistent statistics are automatically recalculated when the table changes significantly (more than 10% of the rows)
- innodb_stats_persistent_sample_pages - Number of index pages sampled (default 20) when estimating cardinality and statistics for indexed columns. Increasing this value will increases index statistics accuracy, but use more I/O resources when running ANALYZE TABLE.

These settings can be overwritten on a per-table basis by use of the STATS_PERSISTENT, STATS_AUTO_RECALC and STATS_SAMPLE_PAGES clauses in a CREATE TABLE or ALTER TABLE statement.

Details of the statistics are stored in two system tables in the mysql database:

- innodb_table_stats
- innodb_index_stats

The ANALYZE TABLE statement can be used to recalculate InnoDB statistics.

The RENAME TABLE statement triggers a reload of the statistics.

MariaDB starting with 10.11.12

Prior to MariaDB 10.11.12, MariaDB 11.4.6 and MariaDB 11.8.2, FLUSH TABLES also caused InnoDB statistics to be reloaded. From MariaDB 10.11.12, MariaDB 11.4.6 and MariaDB 11.8.2, this is no longer the case.

See Also

- Index Statistics
- Engine-independent Statistics
- Histogram-based Statistics

This page is licensed: CC BY-SA / Gnu FDL

Slow Query Log Extended Statistics

Overview

- Added extra logging to slow log of 'Thread_id, Schema, Query Cache hit, Rows sent and Rows examined'
- Added optional logging to slow log, through log_slow_verbosity, of query plan statistics
- Added new session variables log_slow_rate_limit, log_slow_verbosity, log_slow_filter
- Added log-slow-file as synonym for 'slow-log-file', as most slow-log variables start with 'log-slow'
- Added log-slow-time as synonym for long-query-time.

Session Variables

log_slow_verbosity

You can set the verbosity of what's logged to the slow query log by setting the log_slow_verbosity variable to a combination of the following values:

- `All` (From MariaDB 10.6.16)
  - Enable all verbosity options.
- `Query_plan`
  - For select queries, log information about the query plan. This includes "Full_scan", "Full_join", "Tmp_table", "Tmp_table_on_disk", "Filesort", "Filesort_on_disk" and number of "Merge_passes during sorting"
- `explain`
  - EXPLAIN output is logged in the slow query log. See explain-in-the-slow-query-log for details.
- `Innodb` (From MariaDB 10.6.15. Before that this option did nothing)
  - Kept for compatibility. Same as `engine`.
- `engine` (From MariaDB 10.6.15)
  - Writes statistics from the storage engine. This includes:

Option	Description	Engine
Pages_accessed	Number of pages accessed from page buffer (innodb-buffer-pool / key cache)	InnoDB
Pages_updated	Number of pages updated in memory	InnoDB
Pages_read_time	Milliseconds spend reading pages from storage	InnoDB
Old_rows_read	Number of retrievals of old versions of rows in the engine (versioning)	InnoDB
Engine_time	Milliseconds spent inside engine calls (read_row / read_next_row etc)	All

- `Warnings` (From MariaDB 10.6.16)
  - Print all errors, warnings and notes related to statement, up to `log_slow_max_warnings` lines.

- `full`.
  - Old shortcut to enable all the verbosity options

The default value for `log_slow_verbosity` is ' ', to be compatible with MySQL 5.1.

The possible values for `log_slow_verbosity are` innodb,query_plan,explain,engine,warnings `. Multiple options are separated by ','.` log_slow_verbosity is not supported when log_output='TABLE'.

In the future we will add more `engine` statistics and also support for other engines.

log_slow_filter

You can define which queries to log to the slow query log by setting the variable log_slow_filter to a combination of the following values:

- `All` (From MariaDB 10.6.16)
  - Enable all filter options. `log_slow_filter` will be shown as having all options set.
- `admin`
  - Log administrative statements (create, optimize, drop etc…)
  - log_slow_admin_statements maps to this option.
- `filesort`
  - Log statement if it uses filesort
- `filesort_on_disk`
  - Log statement if it uses filesort that needs temporary tables on disk
- `filesort_priority_queue` (from MariaDB 10.3.2)
  - Log statement if it uses filesort with priority_queue (filesort can either use disk or priority queue).
- `full_join`
  - Log statements that don't uses indexes to join tables
- `full_scan`
  - Log statements that use full table scans
- `not_using_index` (From MariaDB 10.3.1)
  - Logs queries that don't use an index, or that perform a full index scan where the index doesn't limit the number of rows
  - Disregards long_query_time, unlike other options!
  - log_queries_not_using_indexes maps to this option
- `query_cache`
  - Log statements that are resolved by the query cache
- `query_cache_miss`
  - Log statements that are not resolved by the query cache
- `tmp_table`
  - Log statements that use in memory temporary tables
- `tmp_table_on_disk`
  - Log statements that use temporary tables on disk

Multiple options are separated by ','. If you don't specify any options everything will be logged (same as setting the value to `All`.

log_slow_rate_limit

The log_slow_rate_limit variable limits logging to the slow query log by not logging every query (only one query / log_slow_rate_limit is logged). This is mostly useful when debugging and you get too much information to the slow query log.

Note that in any case, only queries that takes longer than **log_slow_time** or**long_query_time**' are logged (as before).

log_slow_max_warnings

MariaDB starting with 10.6.16

If one enables the warning option for `log_slow_verbosity`, all notes and warnings for a slow query will also be added to the slow query log. This is very usable when one has enabled warnings for Notes when an index cannot be used. `log_slow_max_warnings` limits the number of warnings printed to the slow query log per query. The default value is 10.

Credits

Part of this addition is based on the microslow patch from Percona.

See also

- Notes when an index cannot be used because of type conversions

This page is licensed: CC BY-SA / Gnu FDL

User Statistics

The User Statistics (userstat) plugin creates the USER_STATISTICS, CLIENT_STATISTICS, the INDEX_STATISTICS, and the TABLE_STATISTICS tables in the INFORMATION_SCHEMA database. As an alternative to these tables, the plugin also adds the SHOW USER_STATISTICS, the SHOW CLIENT_STATISTICS, the SHOW INDEX_STATISTICS, and the SHOW TABLE_STATISTICS statements.

These tables and commands can be used to understand the server activity better and to identify the sources of your database's load.

The plugin also adds the FLUSH USER_STATISTICS, FLUSH CLIENT_STATISTICS, FLUSH INDEX_STATISTICS, and FLUSH TABLE_STATISTICS statements.

The MariaDB implementation of this plugin is based on the userstatv2 patch from Percona and Ourdelta. The original code comes from Google (Mark Callaghan's team) with additional work from Percona, Ourdelta, and Weldon Whipple. The MariaDB implementation provides the same functionality as the userstatv2 patch but a lot of changes have been made to make it faster and to better fit the MariaDB infrastructure.

How it Works

The `userstat` plugin works by keeping several hash tables in memory. All variables are incremented while the query is running. At the end of each statement the global values are updated.

Enabling the Plugin

By default statistics are not collected. This is to ensure that statistics collection does not cause any extra load on the server unless desired.

Set the userstat=ON system variable in a relevant server option group in an option file to enable the plugin. For example:

```
[mariadb]
...
userstat = 1
```

The value can also be changed dynamically. For example:

```
SET GLOBAL userstat=1;
```

Using the Plugin

Using the Information Schema Table

The `userstat` plugin creates the USER_STATISTICS, CLIENT_STATISTICS, the INDEX_STATISTICS, and the TABLE_STATISTICS tables in the INFORMATION_SCHEMA database.

```
SELECT * FROM INFORMATION_SCHEMA.USER_STATISTICS\G
*************************** 1. row ***************************
                 USER: root
     TOTAL_CONNECTIONS: 1
CONCURRENT_CONNECTIONS: 0
        CONNECTED_TIME: 297
             BUSY_TIME: 0.001725
              CPU_TIME: 0.001982
        BYTES_RECEIVED: 388
            BYTES_SENT: 2327
  BINLOG_BYTES_WRITTEN: 0
             ROWS_READ: 0
             ROWS_SENT: 12
          ROWS_DELETED: 0
         ROWS_INSERTED: 13
          ROWS_UPDATED: 0
       SELECT_COMMANDS: 4
       UPDATE_COMMANDS: 0
        OTHER_COMMANDS: 3
   COMMIT_TRANSACTIONS: 0
 ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
      LOST_CONNECTIONS: 0
         ACCESS_DENIED: 0
         EMPTY_QUERIES: 1
```

```
SELECT * FROM INFORMATION_SCHEMA.CLIENT_STATISTICS\G
*************************** 1. row ***************************
               CLIENT: localhost
     TOTAL_CONNECTIONS: 3
CONCURRENT_CONNECTIONS: 0
        CONNECTED_TIME: 4883
             BUSY_TIME: 0.009722
              CPU_TIME: 0.0102131
        BYTES_RECEIVED: 841
            BYTES_SENT: 13897
  BINLOG_BYTES_WRITTEN: 0
             ROWS_READ: 0
             ROWS_SENT: 214
          ROWS_DELETED: 0
         ROWS_INSERTED: 207
          ROWS_UPDATED: 0
       SELECT_COMMANDS: 10
       UPDATE_COMMANDS: 0
        OTHER_COMMANDS: 13
   COMMIT_TRANSACTIONS: 0
 ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
      LOST_CONNECTIONS: 0
         ACCESS_DENIED: 0
         EMPTY_QUERIES: 1
1 row in set (0.00 sec)
```

```
SELECT * FROM INFORMATION_SCHEMA.INDEX_STATISTICS WHERE TABLE_NAME = "author";
+--------------+------------+------------+-----------+
| TABLE_SCHEMA | TABLE_NAME | INDEX_NAME | ROWS_READ |
+--------------+------------+------------+-----------+
| books        | author     | by_name    |        15 |
+--------------+------------+------------+-----------+
```

```
SELECT * FROM INFORMATION_SCHEMA.TABLE_STATISTICS WHERE TABLE_NAME='user';
+--------------+------------+-----------+--------------+-----------------------+
| TABLE_SCHEMA | TABLE_NAME | ROWS_READ | ROWS_CHANGED | ROWS_CHANGED_X_INDEXES |
+--------------+------------+-----------+--------------+-----------------------+
| mysql        | user       |         5 |            2 |                     2 |
+--------------+------------+-----------+--------------+-----------------------+
```

Using the SHOW Statements

As an alternative to the INFORMATION_SCHEMA tables, the `userstat` plugin also adds the SHOW USER_STATISTICS, the SHOW CLIENT_STATISTICS, the SHOW INDEX_STATISTICS, and the SHOW TABLE_STATISTICS statements.

These commands are another way to display the information stored in the information schema tables. WHERE clauses are accepted. LIKE clauses are accepted but ignored.

```
SHOW USER_STATISTICS
SHOW CLIENT_STATISTICS
SHOW INDEX_STATISTICS
SHOW TABLE_STATISTICS
```

Flushing Plugin Data

The `userstat` plugin also adds the FLUSH USER_STATISTICS, FLUSH CLIENT_STATISTICS, FLUSH INDEX_STATISTICS, and FLUSH TABLE_STATISTICS statements, which discard the information stored in the specified information schema table.

```
FLUSH USER_STATISTICS
FLUSH CLIENT_STATISTICS
FLUSH INDEX_STATISTICS
FLUSH TABLE_STATISTICS
```

Versions

USER_STATISTICS

Version	Status	Introduced
2.0	Stable	MariaDB 10.1.18
2.0	Gamma	MariaDB 10.1.1

CLIENT_STATISTICS

Version	Status	Introduced
2.0	Stable	MariaDB 10.1.13
2.0	Gamma	MariaDB 10.1.1

INDEX_STATISTICS

Version	Status	Introduced
2.0	Stable	MariaDB 10.1.13
2.0	Gamma	MariaDB 10.1.1

TABLE_STATISTICS

Version	Status	Introduced
2.0	Stable	MariaDB 10.1.18
2.0	Gamma	MariaDB 10.1.1

System Variables

`userstat`

- Description: If set to `1`, user statistics will be activated.
- Command line: `--userstat=1`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

Status Variables

User Statistics introduced a number of new status variables:

- access_denied_errors
- binlog_bytes_written
- busy_time (requires userstat to be set to be recorded)
- cpu_time (requires userstat to be set to be recorded)
- empty_queries

- rows_read
- rows_sent

This page is licensed: CC BY-SA / Gnu FDL

Subquery Optimizations

See Condition Pushdown Into IN subqueries.

Describes Condition Pushdown into IN subqueries, enabled through the condition_pushdown_for_subquery optimizer_switch flag.

See Conversion of Big IN Predicates Into Subqueries.

Converts large IN predicates into IN subqueries once the list exceeds in_predicate_conversion_threshold elements, avoiding costly range analysis.

See EXISTS-to-IN Optimization.

EXISTS-to-IN rewrites trivially correlated and semi-join EXISTS subqueries into IN subqueries so MariaDB's richer IN-subquery strategies can apply.

See Non-semi-join Subquery Optimizations.

Covers strategies for IN-subqueries that cannot become semi-joins, chiefly materialization (with NULL-aware partial matching) and the IN-to-EXISTS transformation.

See Optimizing GROUP BY and DISTINCT Clauses in Subqueries.

Automatically removes redundant DISTINCT and GROUP BY-without-HAVING clauses in IN/ALL/ANY/SOME/EXISTS subqueries, allowing more efficient query plans.

See Semi-join Subquery Optimizations.

Explains semi-join IN-subqueries and MariaDB's five semi-join execution strategies: table pullout, FirstMatch, Materialization, LooseScan, and DuplicateWeedout.

See Subquery Cache.

The subquery cache speeds up correlated subqueries by caching results with their correlation parameters, avoiding re-execution when a result is already cached.

See Subquery Optimizations Map.

Presents a map of the subquery types allowed in SQL and the optimizer strategies MariaDB provides to handle each, with links to individual optimization pages.

See Table Pullout Optimization.

Table pullout rewrites a semi-join subquery as a join by pulling tables out into the parent SELECT based on UNIQUE or PRIMARY key definitions.

This page is licensed: CC BY-SA / Gnu FDL

Condition Pushdown Into IN subqueries

This article describes Condition Pushdown into IN subqueries as implemented in MDEV-12387.

optimizer_switch flag name: `condition_pushdown_for_subquery`.

This page is licensed: CC BY-SA / Gnu FDL

Conversion of Big IN Predicates Into Subqueries

Starting from MariaDB 10.3, the optimizer converts certain big IN predicates into IN subqueries.

That is, an IN predicate in the form

```
COLUMN [NOT] IN (const1, const2, .... )
```

is converted into an equivalent IN-subquery:

```
column [NOT] IN (select ... from temporary_table)
```

which opens new opportunities for the query optimizer.

The conversion happens if the following conditions are met:

- the IN list has more than 1000 elements (One can control it through the in_predicate_conversion_threshold parameter).
- the [NOT] IN condition is at the top level of the WHERE/ON clause.

Controlling the Optimization

- The optimization is on by default. MariaDB 10.3.18 (and debug builds prior to that) introduced the in_predicate_conversion_threshold variable. Set to `0` to disable the optimization.

Benefits of the Optimization

If `column` is a key-prefix, MariaDB optimizer will process the condition

```
COLUMN [NOT] IN (const1, const2, .... )
```

by trying to construct a range access. If the list is large, the analysis may take a lot of memory and CPU time. The problem gets worse when `column` is a part of a multi-column index and the query has conditions on other parts of the index.

Conversion of IN predicates into subqueries bypass the range analysis, which means the query optimization phase will use less CPU and memory.

Possible disadvantages of the conversion are:

- The optimization may convert 'IN LIST elements' key accesses to a table scan (if there is no other usable index for the table)
- The estimates for the number of rows matching the `IN (...)` are less precise.

See Also

- IN operator

Links

MDEV-12176

This page is licensed: CC BY-SA / Gnu FDL

EXISTS-to-IN Optimization

MySQL (including MySQL 5.6) has only one execution strategy for EXISTS subqueries. The strategy is essentially the straightforward, "naive" execution, without any rewrites.

MariaDB 5.3 introduced a rich set of optimizations for IN subqueries. Since then, it makes sense to convert an EXISTS subquery into an IN so that the new optimizations can be used.

`EXISTS` will be converted into `IN` in two cases:

1. Trivially correlated EXISTS subqueries
2. Semi-join EXISTS

We will now describe these two cases in detail

Trivially-correlated EXISTS subqueries

Often, EXISTS subquery is correlated, but the correlation is trivial. The subquery has form

```
EXISTS (SELECT ...  FROM ... WHERE outer_col= inner_col AND inner_where)
```

and “outer_col” is the only place where the subquery refers to outside fields.
In this case, the subquery can be re-written into uncorrelated IN:

```
outer_col IN (SELECT inner_col FROM ... WHERE inner_where)
```

(`NULL` values require some special handling, see below). For uncorrelated IN subqueries, MariaDB is able a cost-based choice between two execution strategies:

- IN-to-EXISTS (basically, convert back into EXISTS)
- Materialization

That is, converting trivially-correlated `EXISTS` into uncorrelated `IN` gives query optimizer an option to use Materialization strategy for the subquery.

Currently, EXISTS->IN conversion works only for subqueries that are at top level of the WHERE clause, or are under NOT operation which is directly at top level of the WHERE clause.

Semi-join EXISTS subqueries

If `EXISTS` subquery is an AND-part of the `WHERE` clause:

```
SELECT ... FROM outer_tables WHERE EXISTS (SELECT ...) AND ...
```

then it satisfies the main property of semi-join subqueries:

with semi-join subquery, we’re only interested in records of outer_tables that have matches in the subquery

Semi-join optimizer offers a rich set of execution strategies for both correlated and uncorrelated subqueries. The set includes FirstMatch strategy which is an equivalent of how EXISTS suqueries are executed, so we do not lose any opportunities when converting an EXISTS subquery into a semi-join.

In theory, it makes sense to convert all kinds of EXISTS subqueries: convert both correlated and uncorrelated ones, convert irrespectively of whether the subquery has inner=outer equality.

In practice, the subquery will be converted only if it has inner=outer equality. Both correlated and uncorrelated subqueries are converted.

Handling of NULL values

TODO: rephrase this:

- IN has complicated NULL-semantics. NOT EXISTS doesn’t.
- EXISTS-to-IN adds IS NOT NULL before the subquery predicate, when required

Control

The optimization is controlled by the `exists_to_in` flag in optimizer_switch. Before MariaDB 10.0.12, the optimization was OFF by default. Since MariaDB 10.0.12, it has been ON by default.

Limitations

EXISTS-to-IN doesn't handle

- subqueries that have GROUP BY, aggregate functions, or HAVING clause
- subqueries are UNIONs
- a number of degenerate edge cases

This page is licensed: CC BY-SA / Gnu FDL

Non-semi-join Subquery Optimizations

Certain kinds of IN-subqueries cannot be flattened into semi-joins. These subqueries can be both correlated or non-correlated. In order to provide consistent performance in all cases, MariaDB provides several alternative strategies for these types of subqueries. Whenever several strategies are possible, the optimizer chooses the optimal one based on cost estimates.

The two primary non-semi-join strategies are materialization (also called outside-in materialization), and in-to-exists transformation. Materialization is applicable only for non-correlated subqueries, while in-to-exist can be used both for correlated and non-correlated subqueries.

Applicability

An IN subquery cannot be flattened into a semi-join in the following cases. The examples below use the *World* database from the MariaDB regression test suite.

Subquery in a disjunction (OR)

The subquery is located directly or indirectly under an OR operation in the WHERE clause of the outer query.

Query pattern:

```
SELECT ... FROM ... WHERE (expr1, ..., exprN) [NOT] IN (SELECT ... ) OR expr;
```

Example:

```
SELECT Name FROM Country
WHERE (Code IN (SELECT Country FROM City WHERE City.Population > 100000) OR
       Name LIKE 'L%') AND
      surfacearea > 1000000;
```

Negated subquery predicate (NOT IN)

The subquery predicate itself is negated.

Query pattern:

```
SELECT ... FROM ... WHERE ... (expr1, ..., exprN) NOT IN (SELECT ... ) ...;
```

Example:

```
SELECT Country.Name
FROM Country, CountryLanguage
WHERE Code NOT IN (SELECT Country FROM CountryLanguage WHERE LANGUAGE = 'English')
  AND CountryLanguage.Language = 'French'
  AND Code = Country;
```

Subquery in the SELECT or HAVING clause

The subquery is located in the SELECT or HAVING clauses of the outer query.

Query pattern:

```
SELECT field1, ..., (SELECT ...)  WHERE ...;
SELECT ...  WHERE ... HAVING (SELECT ...);
```

Example:

```
SELECT Name, City.id IN (SELECT capital FROM Country WHERE capital IS NOT NULL) AS is_capital
FROM City
WHERE City.population > 10000000;
```

Subquery with a UNION

The subquery itself is a UNION, while the IN predicate may be anywhere in the query where IN is allowed.

Query pattern:

```
... [NOT] IN (SELECT ... UNION SELECT ...)
```

Example:

```
SELECT * FROM City WHERE (Name, 91) IN
(SELECT Name, round(Population/1000) FROM City WHERE Country = "IND" AND Population > 2500000
UNION
 SELECT Name, round(Population/1000) FROM City WHERE Country = "IND" AND Population < 100000);
```

Materialization for non-correlated IN-subqueries

Materialization basics

The basic idea of subquery materialization is to execute the subquery and store its result in an internal temporary table indexed on all its columns. Naturally, this is possible only when the subquery is non-correlated. The IN predicate tests whether its left operand is present in the subquery result. Therefore it is not necessary to store duplicate subquery result rows in the temporary table. Storing only unique subquery rows provides two benefits - the size of the temporary table is smaller, and the index on all its columns can be unique.

If the size of the temporary table is less than the tmp_table_size system variable, the table is a hash-indexed in-memory HEAP table. In the rare cases when the subquery result exceeds this limit, the temporary table is stored on disk in an ARIA or MyISAM B-tree indexed table (ARIA is the default).

Subquery materialization happens on demand during the first execution of the IN predicate. Once the subquery is materialized, the IN predicate is evaluated very efficiently by index lookups of the outer expression into the unique index of the materialized temporary table. If there is a match, IN is TRUE, otherwise IN is FALSE.

NULL-aware efficient execution

An IN predicate may produce a NULL result if there is a NULL value in either of its arguments. Depending on its location in a query, a NULL predicate value is equivalent to FALSE. These are the cases when substituting NULL

with FALSE would reject exactly the same result rows. A NULL result of IN is indistinguishable from a FALSE if the IN predicate is:

- not negated,
- not a function argument,
- inside a WHERE or ON clause.

In all these cases the evaluation of IN is performed as described in the previous paragraph via index lookups into the materialized subquery. In all remaining cases when NULL cannot be substituted with FALSE, it is not possible to use index lookups. This is not a limitation in the server, but a consequence of the NULL semantics in the ANSI SQL standard.

Suppose an IN predicate is evaluated as

```
NULL IN (SELECT
not_null_col FROM t1)
```

, that is, the left operand of IN is a NULL value, and there are no NULLs in the subquery. In this case the value of IN is neither FALSE, nor TRUE. Instead it is NULL. If we were to perform an index lookup with the NULL as a key, such a value would not be found in not_null_col, and the IN predicate would incorrectly produce a FALSE.

In general, an NULL value on either side of an IN acts as a “wildcard” that matches any value, and if a match exists, the result of IN is NULL. Consider the following example:

If the left argument of IN is the row: `(7, NULL, 9)`, and the result of the right subquery operand of IN is the table:

```
(7, 8, 10)
(6, NULL, NULL)
(7, 11, 9)
```

The the IN predicate matches the row `(7, 11, 9)`, and the result of IN is NULL. Matches where the differing values on either side of the IN arguments are matched by a NULL in the other IN argument, are called *partial matches*.

In order to efficiently compute the result of an IN predicate in the presence of NULLs, MariaDB implements two special algorithms for partial matching, described here in detail.

- Rowid-merge partial matching
  This technique is used when the number of rows in the subquery result is above a certain limit. The technique creates special indexes on some of the columns of the temporary table, and merges them by alternative scanning of each index thus performing an operation similar to set-intersection.
- Table scan partial matching
  This algorithm is used for very small tables when the overhead of the rowid-merge algorithm is not justifiable. Then the server simply scans the materialized subquery, and checks for partial matches. Since this strategy doesn’t need any in-memory buffers, it is also used when there is not enough memory to hold the indexes of the rowid-merge strategy.

Limitations

In principle the subquery materialization strategy is universal, however, due to some technical limitations in the MariaDB server, there are few cases when the server cannot apply this optimization.

- BLOB fields
  Either the left operand of an IN predicate refers to a BLOB field, or the subquery selects one or more BLOBs.
- Incomparable fields
  TODO

In the above cases, the server reverts to the IN-TO-EXISTS transformation.

The IN-TO-EXISTS transformation

This optimization is the only subquery execution strategy that existed in older versions of MariaDB and MySQL prior to MariaDB 5.3. We have made various changes and fixed a number of bugs in this code as well, but in essence it remains the same.

Performance discussion

Example speedup over MySQL 5.x and MariaDB 5.1/5.2

Depending on the query and data, either of the two strategies described here may result in orders of magnitude better/worse plan than the other strategy. Older versions of MariaDB and any current MySQL version (including MySQL 5.5, and MySQL 5.6 DMR as of July 2011) implement only the IN-TO-EXISTS transformation. As illustrated below, this strategy is inferior in many common cases to subquery materialization.

Consider the following query over the data of the DBT3 benchmark scale 10. Find customers with top balance in their nations:

```
SELECT * FROM part
WHERE p_partkey IN
      (SELECT l_partkey FROM lineitem
       WHERE l_shipdate BETWEEN '1997-01-01' AND '1997-02-01')
ORDER BY p_retailprice DESC LIMIT 10;
```

The times to run this query is as follows:

- Execution time in MariaDB 5.2/MySQL 5.x (any MySQL): > 1 h
  The query takes more than one hour (we didn't wait longer), which makes it impractical to use subqueries in such cases. The EXPLAIN below shows that the subquery was transformed into a correlated one, which indicates an IN-TO-EXISTS transformation.

```
+--+------------------+--------+--------------+------------------+----+------+--------------------------
-+
|id|select_type       |table   |type          |key               |ref |rows  |Extra
|
+--+------------------+--------+--------------+------------------+----+------+--------------------------
-+
| 1|PRIMARY           |part    |ALL           |NULL              |NULL|199755|Using where; Using
filesort|
| 2|DEPENDENT SUBQUERY|lineitem|index_subquery|i_l_suppkey_partkey|func|    14|Using where
|
+--+------------------+--------+--------------+------------------+----+------+--------------------------
-+
```

- Execution time in MariaDB 5.3: 43 sec
  In MariaDB 5.3 it takes less than a minute to run the same query. The EXPLAIN shows that the subquery remains uncorrelated, which is an indication that it is being executed via subquery materialization.

```
+--+------------+-----------+------+-----------------+----+------+------------------------------+
|id|select_type |table      |type  |key              |ref |rows  |Extra                         |
+--+------------+-----------+------+-----------------+----+------+------------------------------+
| 1|PRIMARY     |part       |ALL   |NULL             |NULL|199755|Using temporary; Using filesort|
| 1|PRIMARY     |<subquery2>|eq_ref|distinct_key     |func|     1|                              |
| 2|MATERIALIZED|lineitem   |range |l_shipdate_partkey|NULL|160060|Using where; Using index      |
+--+------------+-----------+------+-----------------+----+------+------------------------------+
```

The speedup here is practically infinite, because both MySQL and older MariaDB versions cannot complete the query in any reasonable time.

In order to show the benefits of partial matching we extended the *customer* table from the DBT3 benchmark with two extra columns:

- c_pref_nationkey - preferred nation to buy from,
- c_pref_brand - preferred brand.

Both columns are prefixed with the percent NULL values in the column, that is, c_pref_nationkey_05 contains 5% NULL values.

Consider the query “Find all customers that didn’t buy from a preferred country, and from a preferred brand within some date ranges”:

```
SELECT COUNT(*)
FROM customer
WHERE (c_custkey, c_pref_nationkey_05, c_pref_brand_05) NOT IN
  (SELECT o_custkey, s_nationkey, p_brand
   FROM orders, supplier, part, lineitem
   WHERE l_orderkey = o_orderkey AND
         l_suppkey = s_suppkey AND
         l_partkey = p_partkey AND
         p_retailprice < 1200 AND
         l_shipdate >= '1996-04-01' AND l_shipdate < '1996-04-05' AND
         o_orderdate >= '1996-04-01' AND o_orderdate < '1996-04-05');
```

- Execution time in MariaDB 5.2/MySQL 5.x (any MySQL): 40 sec
- Execution time in MariaDB 5.3: 2 sec

The speedup for this query is 20 times.

Performance guidelines

TODO

Optimizer control

In certain cases it may be necessary to override the choice of the optimizer. Typically this is needed for benchmarking or testing purposes, or to mimic the behavior of an older version of the server, or if the optimizer made a poor choice.

All the above strategies can be controlled via the following switches in optimizer_switch system variable.

- materialization=on/off
  In some very special cases, even if materialization was forced, the optimizer may still revert to the IN-TO-EXISTS strategy if materialization is not applicable. In the cases when materialization requires partial matching (because of the presence of NULL values), there are two subordinate switches that control the two partial matching strategies:
  - partial_match_rowid_merge=on/off
    This switch controls the Rowid-merge strategy. In addition to this switch, the system variable rowid_merge_buff_size controls the maximum memory available to the Rowid-merge strategy.
  - partial_match_table_scan=on/off
    Controls the alternative partial match strategy that performs matching via a table scan.
- in_to_exists=on/off
  This switch controls the IN-TO-EXISTS transformation.
- tmp_table_size and max_heap_table_size system variables
  The tmp_table_size system variable sets the upper limit for internal MEMORY temporary tables. If an internal temporary table exceeds this size, it is converted automatically into a Aria or MyISAM table on disk with a B-tree index. Notice however, that a MEMORY table cannot be larger than max_heap_table_size.

The two main optimizer switches - *materialization* and *in_to_exists* cannot be simultaneously off. If both are set to off, the server will issue an error.

This page is licensed: CC BY-SA / Gnu FDL

Optimizing GROUP BY and DISTINCT Clauses in Subqueries

A DISTINCT clause and a GROUP BY without a corresponding HAVING clause have no meaning in IN/ALL/ANY/SOME/EXISTS subqueries. The reason is that IN/ALL/ANY/SOME/EXISTS only check if an outer row satisfies some condition with respect to all or any row in the subquery result. Therefore is doesn't matter if the subquery has duplicate result rows or not - if some condition is true for some row of the subquery, this condition will be true for all duplicates of this row. Notice that GROUP BY without a corresponding HAVING clause is equivalent to a DISTINCT.

MariaDB 5.3 and later versions automatically remove DISTINCT and GROUP BY without HAVING if these clauses appear in an IN/ALL/ANY/SOME/EXISTS subquery. For instance:

```
SELECT * FROM t1
WHERE t1.a > ALL(SELECT DISTINCT b FROM t2 WHERE t2.c > 100)
```

is transformed to:

```
SELECT * FROM t1
WHERE t1.a > ALL(SELECT b FROM t2 WHERE t2.c > 100)
```

Removing these unnecessary clauses allows the optimizer to find more efficient query plans because it doesn't need to take care of post-processing the subquery result to satisfy DISTINCT / GROUP BY.

This page is licensed: CC BY-SA / Gnu FDL

Semi-join Subquery Optimizations

MariaDB has a set of optimizations specifically targeted at *semi-join subqueries*.

What is a Semi-Join Subquery

A semi-join subquery has a form of

```
SELECT ... FROM outer_tables WHERE expr IN (SELECT ... FROM inner_tables ...) AND ...
```

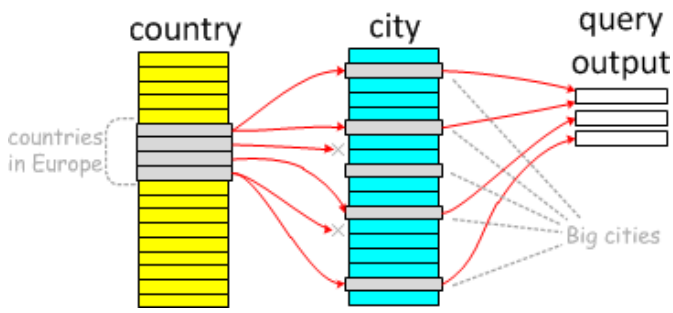
that is, the subquery is an IN-subquery and it is located in the WHERE clause. The most important part here is

with semi-join subquery, we're only interested in records of outer_tables that have matches in the subquery

Let's see why this is important. Consider a semi-join subquery:

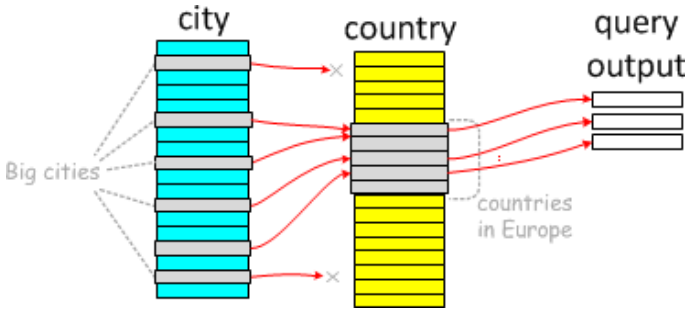
```
SELECT * FROM Country
WHERE
  Continent='Europe' AND
  Country.Code IN (SELECT City.country
                   FROM City
                   WHERE City.Population>1*1000*1000);
```

One can execute it "naturally", by starting from countries in Europe and checking if they have populous Cities:



semi-join-outer-to-inner

The semi-join property also allows “backwards” execution: we can start from big cities, and check which countries they are in:



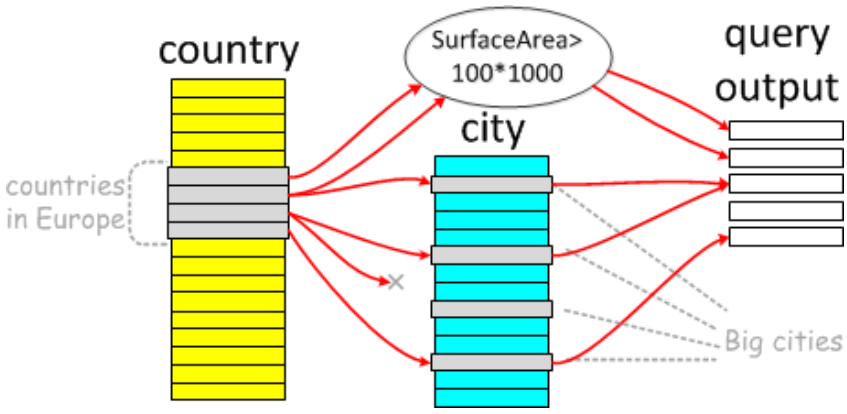
semi-join-inner-to-outer

To contrast, let’s change the subquery to be non-semi-join:

```
SELECT * FROM Country
WHERE
   Country.Continent='Europe' AND
   (Country.Code IN (SELECT City.country
                     FROM City WHERE City.Population>1*1000*1000)
    OR Country.SurfaceArea > 100*1000  -- Added this part
   );
```

It is still possible to start from countries, and then check

- if a country has any big cities
- if it has a large surface area:



non-semi-join-subquery

The opposite, city-to-country way is not possible. This is not a semi-join.

Difference from Inner Joins

Semi-join operations are similar to regular relational joins. There is a difference though: with semi-joins, you don't care how many matches an inner table has for an outer row. In the above countries-with-big-cities example, Germany will be returned once, even if it has three cities with populations of more than one million each.

Semi-Join Optimizations in MariaDB

MariaDB uses semi-join optimizations to run IN subqueries.The optimizations are enabled by default. You can disable them by turning off their optimizer_switch like so:

```
SET optimizer_switch='semijoin=off'
```

MariaDB has five different semi-join execution strategies:

- Table pullout optimization
- FirstMatch execution strategy
- Semi-join Materialization execution strategy
- LooseScan execution strategy
- DuplicateWeedout execution strategy

See Also

- Subquery Optimizations Map
- "Observations about subquery use cases" blog post
- http:en.wikipedia.org/wiki/Semijoin

This page is licensed: CC BY-SA / Gnu FDL

Subquery Cache

The goal of the subquery cache is to optimize the evaluation of correlated subqueries by storing results together with correlation parameters in a cache and avoiding re-execution of the subquery in cases where the result is already in the cache.

Administration

The cache is on by default. One can switch it off using the optimizer_switch `subquery_cache` setting, like so:

```
SET optimizer_switch='subquery_cache=off';
```

The efficiency of the subquery cache is visible in 2 statistical variables:

- Subquery_cache_hit - Global counter for all subquery cache hits.
- Subquery_cache_miss - Global counter for all subquery cache misses.

The session variables tmp_table_size and max_heap_table_size influence the size of in-memory temporary tables in the table used for caching. It cannot grow more than the minimum of the above variables values (see the Implementation section for details).

Visibility

Your usage of the cache is visible in `EXTENDED EXPLAIN` output (warnings) as `"<expr_cache><//list of parameters//>(//cached expression//)"`.
For example:

```
EXPLAIN EXTENDED SELECT * FROM t1 WHERE a IN (SELECT b FROM t2);
+----+--------------------+-------+------+---------------+------+---------+------+------+----------+-----
--------+
| id | select_type        | table | type | possible_keys | key  | key_len | ref  | rows | filtered |
Extra       |
+----+--------------------+-------+------+---------------+------+---------+------+------+----------+-----
--------+
|  1 | PRIMARY            | t1    | ALL  | NULL          | NULL | NULL    | NULL |    2 |   100.00 |
Using where |
|  2 | DEPENDENT SUBQUERY | t2    | ALL  | NULL          | NULL | NULL    | NULL |    2 |   100.00 |
Using where |
+----+--------------------+-------+------+---------------+------+---------+------+------+----------+-----
--------+
2 rows in set, 1 warning (0.00 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
----------+
| Level | Code | Message
|
+-------+------+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
----------+
| Note  | 1003 | SELECT `test`.`t1`.`a` AS `a` from `test`.`t1` WHERE <expr_cache><`test`.`t1`.`a`>
(<in_optimizer>(`test`.`t1`.`a`,<exists>(SELECT 1 FROM `test`.`t2` WHERE (<cache>(`test`.`t1`.`a`) =
`test`.`t2`.`b`)))) |
+-------+------+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
----------+
1 row in set (0.00 sec)
```

In the example above the presence of `"<expr_cache><`test`.`t1`.`a`>(...)"` is how you know you are using the subquery cache.

Implementation

Every subquery cache creates a temporary table where the results and all parameters are stored. It has a unique index over all parameters. First the cache is created in a MEMORY table (if doing this is impossible the cache becomes disabled for that expression). When the table grows up to the minimum of `tmp_table_size` and `max_heap_table_size`, the hit rate will be checked:

- if the hit rate is really small (<0.2) the cache will be disabled.
- if the hit rate is moderate (<0.7) the table will be cleaned (all records deleted) to keep the table in memory
- if the hit rate is high the table will be converted to a disk table (for 5.3.0 it can only be converted to a disk table).

```
hit rate = hit / (hit + miss)
```

Performance Impact

Here are some examples that show the performance impact of the subquery cache (these tests were made on a 2.53 GHz Intel Core 2 Duo MacBook Pro with dbt-3 scale 1 data set).

example	cache on	cache off	gain	hit	miss	hit rate	1	2	3	4
example	cache on	cache off	gain	hit	miss	hit rate				
1	1.01sec	1 hour 31 min 43.33sec	5445x	149975	25	99.98%				
2	0.21sec	1.41sec	6.71x	6285	220	96.6%				
3	2.54sec	2.55sec	1.00044x	151	461	24.67%				
4	1.87sec	1.95sec	0.96x	0	23026	0%				

Example 1

Dataset from DBT-3 benchmark, a query to find customers with balance near top in their nation:

```
SELECT COUNT(*) FROM customer
WHERE
   c_acctbal > 0.8 * (SELECT MAX(c_acctbal)
                      FROM customer C
                      WHERE C.c_nationkey=customer.c_nationkey
                      GROUP BY c_nationkey);
```

Example 2

DBT-3 benchmark, Query #17

```
SELECT SUM(l_extendedprice) / 7.0 AS avg_yearly
FROM lineitem, part
WHERE
  p_partkey = l_partkey AND
  p_brand = 'Brand#42' AND p_container = 'JUMBO BAG' AND
  l_quantity < (SELECT 0.2 * AVG(l_quantity) FROM lineitem
                WHERE l_partkey = p_partkey);
```

Example 3

DBT-3 benchmark, Query #2

```
SELECT
        s_acctbal, s_name, n_name, p_partkey, p_mfgr, s_address, s_phone, s_comment
FROM
        part, supplier, partsupp, nation, region
WHERE
        p_partkey = ps_partkey AND s_suppkey = ps_suppkey AND p_size = 33
        AND p_type LIKE '%STEEL' AND s_nationkey = n_nationkey
        AND n_regionkey = r_regionkey AND r_name = 'MIDDLE EAST'
        AND ps_supplycost = (
                SELECT
                        MIN(ps_supplycost)
                FROM
                        partsupp, supplier, nation, region
                WHERE
                        p_partkey = ps_partkey AND s_suppkey = ps_suppkey
                        AND s_nationkey = n_nationkey AND n_regionkey = r_regionkey
                        AND r_name = 'MIDDLE EAST'
        )
ORDER BY
        s_acctbal DESC, n_name, s_name, p_partkey;
```

Example 4

DBT-3 benchmark, Query #20

```
SELECT
        s_name, s_address
FROM
        supplier, nation
WHERE
        s_suppkey IN (
                SELECT
                        DISTINCT (ps_suppkey)
                FROM
                        partsupp, part
                WHERE
                        ps_partkey=p_partkey
                        AND p_name LIKE 'indian%'
                        AND ps_availqty > (
                                SELECT
                                        0.5 * SUM(l_quantity)
                                FROM
                                        lineitem
                                WHERE
                                        l_partkey = ps_partkey
                                        AND l_suppkey = ps_suppkey
                                        AND l_shipdate >= '1995-01-01'
                                        AND l_shipdate < date_ADD('1995-01-01',INTERVAL 1 YEAR)
                                )
        )
        AND s_nationkey = n_nationkey AND n_name = 'JAPAN'
ORDER BY
        s_name;
```

See Also

- Query cache
- blog post describing impact of subquery cache optimization on queries used by DynamicPageList MediaWiki extension
- mariadb-subquery-cache-in-real-use-case.html Another use case from the real world

This page is licensed: CC BY-SA / Gnu FDL

Subquery Optimizations Map

Below is a map showing all types of subqueries allowed in the SQL language, and the optimizer strategies available to handle them.

- Uncolored areas represent different kinds of subqueries, for example:
  - Subqueries that have form `x IN (SELECT ...)`
  - Subqueries that are in the `FROM` clause
  - .. and so forth
- The size of each uncolored area roughly corresponds to how important (i.e. frequently used) that kind of subquery is. For example, `x IN (SELECT ...)` queries are the most important, and `EXISTS (SELECT ...)` are relatively unimportant.
- Colored areas represent optimizations/execution strategies that are applied to handle various kinds of subqueries.
- The color of optimization indicates which version of MySQL/MariaDB it was available in (see legend below)

Map of the kinds of subqueries the optimizer recognizes and the strategies available to handle each one.

Some things are not on the map:

- MariaDB doesn't evaluate expensive subqueries when doing optimization (this means, EXPLAIN is always fast). MySQL 5.6 has made a progress in this regard, but its optimizer will still evaluate certain kinds of subqueries (for example, scalar-context subqueries used in range predicates)

Links to pages about individual optimizations:

- IN->EXISTS
- Subquery Caching
- Semi-join optimizations
  - Table pullout
  - FirstMatch
  - Materialization, +scan, +lookup
  - LooseScan
  - DuplicateWeedout execution strategy
- Non-semi-join Materialization (including NULL-aware and partial matching)
- Derived table optimizations
  - Derived table merge
  - Derived table with keys

See also

- Subquery optimizations in MariaDB 5.3

This page is licensed: CC BY-SA / Gnu FDL

Table Pullout Optimization

Table pullout is an optimization for Semi-join subqueries.

The idea of Table Pullout

Sometimes, a subquery can be re-written as a join. For example:

```
SELECT *
FROM City
WHERE City.Country IN (SELECT Country.Code
                       FROM Country
                       WHERE Country.Population < 100*1000);
```

If we know that there can be, at most, one country with a given value of `Country.Code` (we can tell that if we see that table Country has a primary key or unique index over that column), we can re-write this query as:

```
SELECT City.*
FROM
  City, Country
WHERE
 City.Country=Country.Code AND Country.Population < 100*1000;
```

Table pullout in action

If one runs EXPLAIN for the above query in MySQL 5.1-5.6 or MariaDB 5.1-5.2, they'll get this plan:

```
MySQL [world]> EXPLAIN SELECT * FROM City WHERE City.Country IN (SELECT Country.Code FROM Country WHERE
Country.Population < 100*1000);
+----+--------------------+---------+----------------+--------------------+---------+---------+------+--
----+-------------+
| id | select_type        | table   | type           | possible_keys      | key     | key_len | ref  |
rows | Extra       |
+----+--------------------+---------+----------------+--------------------+---------+---------+------+--
----+-------------+
|  1 | PRIMARY            | City    | ALL            | NULL               | NULL    | NULL    | NULL |
4079 | Using where |
|  2 | DEPENDENT SUBQUERY | Country | unique_subquery | PRIMARY,Population | PRIMARY | 3       | func |
1 | Using where |
+----+--------------------+---------+----------------+--------------------+---------+---------+------+--
----+-------------+
2 rows in set (0.00 sec)
```

It shows that the optimizer is going to do a full scan on table `City`, and for each city it will do a lookup in table `Country`.

If one runs the same query in MariaDB 5.3, they will get this plan:

```
MariaDB [world]> EXPLAIN SELECT * FROM City WHERE City.Country IN (SELECT Country.Code FROM Country WHERE
Country.Population < 100*1000);
+----+-------------+---------+-------+--------------------+------------+---------+--------------------+--
----+-----------------------+
| id | select_type | table   | type  | possible_keys      | key        | key_len | ref                |
rows | Extra                 |
+----+-------------+---------+-------+--------------------+------------+---------+--------------------+--
----+-----------------------+
|  1 | PRIMARY     | Country | range | PRIMARY,Population | Population | 4       | NULL               |
37 | Using index condition |
|  1 | PRIMARY     | City    | ref   | Country            | Country    | 3       | world.Country.Code |
18 |                       |
+----+-------------+---------+-------+--------------------+------------+---------+--------------------+--
----+-----------------------+
2 rows in set (0.00 sec)
```

The interesting parts are:

- Both tables have `select_type=PRIMARY`, and `id=1` as if they were in one join.
- The `Country` table is first, followed by the `City` table.

Indeed, if one runs EXPLAIN EXTENDED; SHOW WARNINGS, they will see that the subquery is gone and it was replaced with a join:

```
MariaDB [world]> SHOW warnings\G
*************************** 1. row ***************************
  Level: Note
   Code: 1003
Message: SELECT `world`.`City`.`ID` AS `ID`,`world`.`City`.`Name` AS
`Name`,`world`.`City`.`Country` AS `Country`,`world`.`City`.`Population` AS
`Population`


   FROM `world`.`City` JOIN `world`.`Country` WHERE

((`world`.`City`.`Country` = `world`.`Country`.`Code`) and (`world`.`Country`.
`Population` < (100 * 1000)))
1 row in set (0.00 sec)
```

Changing the subquery into a join allows feeding the join to the join optimizer, which can make a choice between two possible join orders:

1. City -> Country

2. Country -> City

as opposed to the single choice of

1. City->Country

which we had before the optimization.

In the above example, the choice produces a better query plan. Without pullout, the query plan with a subquery would read `(4079 + 1*4079)=8158` table records. With table pullout, the join plan would read `(37 + 37 * 18) = 703` rows. Not all row reads are equal, but generally, reading `10` times fewer table records is faster.

Table pullout fact sheet

- Table pullout is possible only in semi-join subqueries.
- Table pullout is based on `UNIQUE` / `PRIMARY` key definitions.
- Doing table pullout does not cut off any possible query plans, so MariaDB will always try to pull out as much as possible.
- Table pullout is able to pull individual tables out of subqueries to their parent selects. If all tables in a subquery have been pulled out, the subquery (i.e. its semi-join) is removed completely.
- One common bit of advice for optimizing MySQL has been “If possible, rewrite your subqueries as joins”. Table pullout does exactly that, so manual rewrites are no longer necessary.

Controlling table pullout

There is no separate @@optimizer_switch flag for table pullout. Table pullout can be disabled by switching off all semi-join optimizations with `SET @@optimizer_switch='semijoin=off'` command.

This page is licensed: CC BY-SA / Gnu FDL

Table Elimination

See Table Elimination External Resources.

Links to external resources demonstrating table elimination in MariaDB.

See Table Elimination in MariaDB.

Shows how the MariaDB table elimination module merges a view and removes outer-joined tables that are guaranteed a single match and unused elsewhere in the query.

See Table Elimination in Other Databases.

Compares table elimination support in Microsoft SQL Server 2005/2008 and Oracle 11g, noting SQL Server’s more advanced handling of subselect join conditions.

See Table Elimination User Interface.

Shows how to confirm table elimination by inspecting EXPLAIN output for absent tables, and notes the debug-build table_elimination switch.

See What is Table Elimination?.

Introduces table elimination, which resolves a query without accessing some referenced tables, using an Anchor Modeling actors example to show when it applies.

This page is licensed: CC BY-SA / Gnu FDL

Table Elimination External Resources

- an example of how to do this in MariaDB

This page is licensed: CC BY-SA / Gnu FDL

Table Elimination in MariaDB

The first thing the MariaDB optimizer does is to merge the `VIEW` definition into the query to obtain:

```
SELECT ACRAT_rating
FROM
  ac_anchor
  LEFT JOIN ac_name ON ac_anchor.AC_ID=ac_name.AC_ID
  LEFT JOIN ac_dob ON ac_anchor.AC_ID=ac_dob.AC_ID
  LEFT JOIN ac_rating ON (ac_anchor.AC_ID=ac_rating.AC_ID AND
                          ac_rating.ACRAT_fromdate =
                            (SELECT MAX(sub.ACRAT_fromdate)
                             FROM ac_rating sub WHERE sub.AC_ID = ac_rating.AC_ID))
WHERE
 ACNAM_name='Gary Oldman'
```

It's important to realize that the obtained query has a useless part:

- `left join ac_dob on ac_dob.AC_ID=...` will produce exactly one matching record:
  - `primary key(ac_dob.AC_ID)` guarantees that there will be at most one match for any value of `ac_anchor.AC_ID`,
  - and if there won't be a match, `LEFT JOIN` will generate a NULL-complemented "row"
- and we don't care what the matching record is, as table `ac_dob` is not used anywhere else in the query.

This means that the `left join ac_dob on ...` part can be removed from the query and this is what Table Elimination module does. The detection logic is rather smart, for example it would be able to remove the `left join ac_rating on ...` part as well, together with the subquery (in the above example it won't be removed because ac_rating used in the selection list of the query). The Table Elimination module can also handle nested outer joins and multi-table outer joins.

See Also

- This page is based on the following blog post about table elimination:?p=58

This page is licensed: CC BY-SA / Gnu FDL

Table Elimination in Other Databases

In addition to MariaDB, Table Elimination is found in both Microsoft SQL Server 2005/2008 and Oracle 11g. Of the two, Microsoft SQL Server 2005/2008 seems to have the most advanced implementation.
Oracle 11g has been confirmed to use table elimination but not to the same extent.

To compare the two, we will look at the following query:

```
SELECT
 A.colA
FROM
 tableA A
LEFT OUTER JOIN
 tableB B
ON
 B.id = A.id;
```

When using A as the left table we ensure that the query will return at least as many rows as there are in that table. For rows where the join condition (B.id = A.id) is not met the selected column (A.colA) will still contain its original value. The not seen B.* row would contain all NULL:s.

However, the result set could actually contain more rows than what is found in tableA if there are duplicates of the column B.id in tableB. If A contains a row [1, “val1”] and B the rows [1, “other1a”],[1, “other1b”] then two rows will match in the join condition. The only way to know what the result will look like is to actually touch both tables during execution.

Instead, let’s say tableB contains rows that make it possible to place a unique constraint on the column B.id, for example, which is often the case with a primary key. In this situation we know that we will get exactly as many rows as there are in tableA, since joining with tableB cannot introduce any duplicates. Furthermore, as in the example query, if we do not select any columns from tableB, touching that table during execution is unnecessary. We can remove the whole join operation from the execution plan.

Both SQL Server 2005/2008 and Oracle 11g deploy table elimination in the case described above. Let us look at a more advanced query, where Oracle fails.

```
SELECT
 A.colA
FROM
 tableA A
LEFT OUTER JOIN
 tableB B
ON
 B.id = A.id
AND
 B.fromDate = (
   SELECT
     MAX(sub.fromDate)
   FROM
     tableB sub
   WHERE
     sub.id = A.id
 );
```

In this example we have added another join condition, which ensures that we only pick the matching row from tableB having the latest fromDate. In this case tableB will contain duplicates of the column B.id, so in order to ensure uniqueness the primary key has to contain the fromDate column as well. In other words the primary key of tableB is (B.id, B.fromDate).

Furthermore, since the subselect ensures that we only pick the latest B.fromDate for a given B.id we know that at most one row will match the join condition. We will again have the situation where joining with tableB cannot affect the number of rows in the result set. Since we do not select any columns from tableB, the whole join operation can be eliminated from the execution plan.

SQL Server 2005/2008 will deploy table elimination in this situation as well. We have not found a way to make Oracle 11g use it for this type of query. Queries like these arise in two situations. Either when you have a denormalized model consisting of a fact table with several related dimension tables, or when you have a highly normalized model where each attribute is stored in its own table. The example with the subselect is common whenever you store historized/versioned data.

See Also

- This page is based on the following blog post about table elimination:?p=58

This page is licensed: CC BY-SA / Gnu FDL

Table Elimination User Interface

One can check that table elimination is working by looking at the output of `EXPLAIN [EXTENDED]` and not finding there the tables that were eliminated:

```
EXPLAIN SELECT ACRAT_rating FROM actors WHERE ACNAM_name='Gary Oldman';
+----+--------------------+-----------+--------+---------------+---------+---------+--------------------
-+------+-------------+
| id | select_type        | table     | type   | possible_keys | key     | key_len | ref
| rows | Extra       |
+----+--------------------+-----------+--------+---------------+---------+---------+--------------------
-+------+-------------+
|  1 | PRIMARY            | ac_anchor | index  | PRIMARY       | PRIMARY | 4       | NULL
|    2 | Using index |
|  1 | PRIMARY            | ac_name   | eq_ref | PRIMARY       | PRIMARY | 4       | test.ac_anchor.AC_ID
|    1 | Using where |
|  1 | PRIMARY            | ac_rating | ref    | PRIMARY       | PRIMARY | 4       | test.ac_anchor.AC_ID
|    1 |             |
|  3 | DEPENDENT SUBQUERY | sub       | ref    | PRIMARY       | PRIMARY | 4       | test.ac_rating.AC_ID
|    1 | Using index |
+----+--------------------+-----------+--------+---------------+---------+---------+--------------------
-+------+-------------+
```

Note that `ac_dob` table is not in the output. Now let's try getting birthdate instead:

```
EXPLAIN SELECT ACDOB_birthdate FROM actors WHERE ACNAM_name='Gary Oldman';
+----+-------------+-----------+--------+---------------+---------+---------+----------------------+-----
-+-------------+
| id | select_type | table     | type   | possible_keys | key     | key_len | ref                  | rows
| Extra       |
+----+-------------+-----------+--------+---------------+---------+---------+----------------------+-----
-+-------------+
|  1 | PRIMARY     | ac_anchor | index  | PRIMARY       | PRIMARY | 4       | NULL                 |    2
| Using index |
|  1 | PRIMARY     | ac_name   | eq_ref | PRIMARY       | PRIMARY | 4       | test.ac_anchor.AC_ID |    1
| Using where |
|  1 | PRIMARY     | ac_dob    | eq_ref | PRIMARY       | PRIMARY | 4       | test.ac_anchor.AC_ID |    1
|             |
+----+-------------+-----------+--------+---------------+---------+---------+----------------------+-----
-+-------------+
3 rows in set (0.01 sec)
```

The `ac_dob` table is there while `ac_rating` and the subquery are gone. Now, if we just want to check the name of the actor:

```
EXPLAIN SELECT count(*) FROM actors WHERE ACNAM_name='Gary Oldman';
+----+-------------+-----------+--------+---------------+---------+---------+----------------------+-----
-+-------------+
| id | select_type | table     | type   | possible_keys | key     | key_len | ref                  | rows
| Extra       |
+----+-------------+-----------+--------+---------------+---------+---------+----------------------+-----
-+-------------+
|  1 | PRIMARY     | ac_anchor | index  | PRIMARY       | PRIMARY | 4       | NULL                 |    2
| Using index |
|  1 | PRIMARY     | ac_name   | eq_ref | PRIMARY       | PRIMARY | 4       | test.ac_anchor.AC_ID |    1
| Using where |
+----+-------------+-----------+--------+---------------+---------+---------+----------------------+-----
-+-------------+
2 rows in set (0.01 sec)
```

In this case it will eliminate both the `ac_dob` and `ac_rating` tables.

Removing tables from a query does not make the query slower, and it does not cut off any optimization opportunities, so table elimination is unconditional and there are no plans on having any kind of query hints for it.

For debugging purposes there is a `table_elimination=on|off` switch in debug builds of the server.

See Also

- This page is based on the following blog post about table elimination:?p=58

This page is licensed: CC BY-SA / Gnu FDL

What is Table Elimination?

The basic idea behind table elimination is that sometimes it is possible to resolve a query without even accessing some of the tables that the query refers to. One can invent many kinds of such cases, but in Table Elimination we targeted only a certain class of SQL constructs that one ends up writing when they are querying highly-normalized data.

The sample queries were drawn from “Anchor Modeling”, a database modeling technique which takes normalization to the extreme. The slides at the anchor modeling website have an in-depth explanation of Anchor modeling and its merits, but the part that’s important for table elimination can be shown with an example.

Suppose the database stores information about actors, together with their names, birthdays, and ratings, where ratings can change over time:

ACTOR		
string	Name	
date	DateOfBirth	
int	Rating	historized: multiple values over time

The Actor entity with its Name, Date of Birth, and Rating attributes; Rating is historized, so it can hold multiple values over time.

According to anchor modeling, each attribute should go into its own table:

- the ‘anchor’ table which only has a synthetic primary key:

```
CREATE TABLE  ac_anchor(AC_ID INT PRIMARY KEY);
```

- a table for the ‘name’ attribute:

```
CREATE TABLE ac_name(AC_ID INT, ACNAM_name CHAR(N),
                     PRIMARY KEY(AC_ID));
```

- a table for the ‘birthdate’ attribute:

```
CREATE TABLE ac_dob(AC_ID INT,
                    ACDOB_birthdate DATE,
                    PRIMARY KEY(AC_ID));
```

- a table for the ‘rating’ attribute, which is historized:

```
CREATE TABLE ac_rating(AC_ID INT,
                       ACRAT_rating INT,
                       ACRAT_fromdate DATE,
                       PRIMARY KEY(AC_ID, ACRAT_fromdate));
```

With this approach it becomes easy to add/change/remove attributes, but this comes at a cost of added complexity in querying the data: in order to answer the simplest, select-star question of displaying actors and their current ratings one has to write outer joins:

Display actors, with their names and current ratings:

```
SELECT
  ac_anchor.AC_ID, ACNAM_Name, ACDOB_birthdate, ACRAT_rating
FROM
  ac_anchor
  LEFT JOIN ac_name ON ac_anchor.AC_ID=ac_name.AC_ID
  LEFT JOIN ac_dob ON ac_anchor.AC_ID=ac_dob.AC_ID
  LEFT JOIN ac_rating ON (ac_anchor.AC_ID=ac_rating.AC_ID AND
                          ac_rating.ACRAT_fromdate =
                            (SELECT MAX(sub.ACRAT_fromdate)
                             FROM ac_rating sub WHERE sub.AC_ID = ac_rating.AC_ID))
```

We don't want to write the joins every time we need to access an actor's properties, so we'll create a view:

```
CREATE VIEW actors AS
  SELECT  ac_anchor.AC_ID, ACNAM_Name,  ACDOB_birthdate, ACRAT_rating
  FROM <see the SELECT above>
```

This will allow us to access the data as if it was stored in a regular way:

```
SELECT ACRAT_rating FROM actors WHERE ACNAM_name='Gary Oldman'
```

And this is where table elimination will be needed.

See Also

- This page is based on the following blog post about table elimination:?p=58

This page is licensed: CC BY-SA / Gnu FDL

Query Optimizer

See Block-Based Join Algorithms.

Block-based join algorithms in MariaDB: Block Nested Loop, Block Hash Join, and Batch Key Access.

See Condition Selectivity Computation Internals.

How the optimizer computes condition selectivities (internals).

See Extended Keys.

The Extended Keys optimization, which uses primary-key parts appended to secondary indexes.

See MIN/MAX optimization.

The MIN/MAX optimization, which resolves MIN() and MAX() from an index without scanning rows.

See Notes When an Index Cannot Be Used.

Situations in which the optimizer cannot use an index.

See Optimizer Debugging With GDB.

Tips for debugging MariaDB optimizer code with GDB.

See Optimizer Development.

Notes on MariaDB optimizer development.

See optimizer_max_sel_arg_weight.

The optimizer_max_sel_arg_weight setting, which caps the complexity of range analysis.

See Range Optimizer.

The range optimizer, which builds range scans from WHERE conditions.

See The Optimizer Cost Model from MariaDB 11.0.

The optimizer cost model introduced in MariaDB 11.0.

See Optimizer Trace.

Optimizer trace, which records how the optimizer builds a query's execution plan.

This page is licensed: CC BY-SA / Gnu FDL

Block-Based Join Algorithms

In the versions of MariaDB/MySQL before 5.3, only one block-based join algorithm was implemented: the Block Nested Loops (BNL) join algorithm, which could only be used for inner joins.

MariaDB 5.3 enhanced the implementation of BNL joins and provides a variety of block-based join algorithms that can be used for inner joins, outer joins, and semi-joins. Block-based join algorithms in MariaDB employ a join buffer to accumulate records of the first join operand before they start looking for matches in the second join operand.

This page documents the various block-based join algorithms.

- Block Nested Loop (BNL) join
- Block Nested Loop Hash (BNLH) join
- Block Index join, also known as Batch Key Access (BKA) join
- Block Index Hash join, known as Batch Key Access Hash (BKAH) join

Block Nested Loop Join

The major difference between the implementation of BNL join in MariaDB 5.3 compared to earlier versions of MariaDB/MySQL is that the former uses a new format for records written into join buffers. This new format allows:

- More efficient use of buffer space for null field values and field values of flexible length types (like the varchar type)
- Support for so-called incremental join buffers, saving buffer space for multi-way joins
- Use of the algorithm for outer joins and semi-joins

How Block Nested Loop Join Works

The algorithm performs a join operation of tables t1 and t2 according to the following schema. The records of the first operand are written into the join buffer one by one until the buffer is full.

The records of the second operand are read from the base/temporary table one by one. For every read record r2 of table t2, the join buffer is scanned, and, for any record r1 from the buffer such that r2 matches r1, the concatenation of the interesting fields of r1 and r2 is sent to the result stream of the corresponding partial join. To read the records of t2, a full table scan, a full index scan, or a range index scan is performed. Only the records that meet the condition pushed to table t2 are checked for a match of the records from the join buffer.

When the scan of the table t2 is finished, a new portion of the records of the first operand fills the buffer, and matches for these records are looked for in t2.
The buffer refills and scans of the second operand that look for matches in the join buffer are performed again and again until the records of the first operand are exhausted. In total, the algorithm scans the second operand as many times as many refills of the join buffer occur.

More Efficient Usage of Join Buffer Space

No join buffer space is used for null field values.
Any field value of a flexible length type is not padded by 0 up to the maximal field size anymore.

Incremental Join Buffers

If we have a query with a join of three tables, t1, t2, and t3, such that table t1 is joined with table t2 and the result of this join operation is joined with table t3, then two join buffers can be used to execute the query. The first join buffer B1 is used to store the records comprising interesting fields of table t1, while the second join buffer B2 contains the records with fields from the partial join of t1 and t2. The interesting fields of any record r1 from B1 are copied into B2 for any record r1, r2 from the partial join of t1 and t2.

One could suggest storing in B2 just a pointer to the position of the r1 fields in B1, together with the interesting fields from t2. So, for any record r2 matching the record r1, the buffer B2 would contain a reference to the fields of r1 in B1 and the fields of r2. In this case, the buffer B2 is called incremental.

Incremental buffers allow to avoid copying field values from one buffer into another. They also allow to save a significant amount of buffer space if, for a record from t1, several matches from t2 are expected.

Using Join Buffers for Simple Outer Joins and Semi-joins

If a join buffer is used for a simple left outer join of tables t1 and t1 t1 LEFT JOIN t2 ON P(t1,t2) then each record r1 stored in the buffer is provided with a match flag. Initially, this flag is set off. As soon as the first match for r1 is found, this flag is set on. When all matching candidates from t2 have been checked, the records in the join buffer are scanned, and for those of them that still have their match flags off, null-complemented rows are generated.
The same match flag is used for any record in the join buffer is a semi-join operation t1 SEMI JOIN t2 ON P(t1,t2) is performed with a block-based join algorithm. When this match flag is set to on for a record r1 in the buffer, no matches from table t2 for record r1 are looked for anymore.

Block Hash Join

Block-based hash join algorithm is a new option to be used for join operations in MariaDB 5.3. It can be employed in cases when there are equi-join sub-conditions for the joined tables, in other words, when equalities of the form t2.f1= e1(t1),…,t2.fn=en(t1) can be extracted from the full join condition. Like any block-based join algorithm, this one used a join buffer filled with the records of the first operand and looked through the records of the second operand to find matches for the records in the buffer.

How Block Hash Join Works

For each refill of the join buffer and each record r1 from it the algorithm builds a hash table with the keys constructed over the values e1(r1),…en(r1). Then the records of t2 are looked through. For each record r2 from t2 that the condition pushed to the table t2, a hash key over the fields r2.f1,…, r2.fn is calculated to probe into the hash table. The probing returns those records from the buffer to which r2 matches.

As for the BNL join algorithm, this algorithm scans the second operand as many times as refills of the buffer occur. Yet it has to look only through the records of one bucket in the hash table when looking for the records to which a record from t2 matches, not through all records in the join buffer as the BNL join algorithm does.

The implementation of this algorithm in MariaDB builds the hash table with hash keys at the very end of the join buffer. That's why the number of records written into the buffer at one refill is less than for the BNL join algorithms. However, a much shorter list of possible matching candidates makes this block hash join algorithm usually much faster than the BNL join.

Batch Key Access Join

The Batch Keys Access join algorithm performs index look-ups when looking for possible matching candidates provided by the second join operand. In this respect, the algorithm behaves as the regular join algorithm. Yet BKA performs index look-ups for a batch of the records from the join buffer. For conventional database engines like InnoDB/MyISAM, it allows to fetch matching candidates in an optimal way. For the engines with remote data store, such as FederateX/Spider, the algorithm allows to save on transfers between the MySQL node and the data store nodes.

How Batch Keys Access Join Works

The implementation of the algorithm in 5.3 heavily exploits the multi-range-read interface and its properties. The interface hides the actual mechanism of fetching possible candidates for matching records from the table to be joined.
Like any block-based join algorithm, the BKA join repeatedly fills the join buffer with records of the first operand, and for each refill, it finds records from the join table that could match the records in the buffer.

To find such records, it asks the MRR interface to perform index look-ups with the keys constructed over all records from the buffer. Together with each key, the interface receives a return address - a reference to the record over which this key has been constructed. The actual implementation functions of the MRR interface organize and optimize the process of fetching the records of the joined table by the received keys. Each fetched record r2 is appended with the return address associated with the key by which the record has been found, and the result is passed to the BKA join procedure.

The procedure takes the record r1 from the join buffer by the return address, joins it with r2, and checks the join condition. If the condition is evaluated to true, the joined records are sent to the result stream of the join operation. So, for each record returned by the MRR interface, only one record from the join buffer is accessed.

The number of records from table t2 fetched by the BKA join is exactly the same as for the regular nested loops join algorithm. Yet BKA join allows to optimize the order in which the records are fetched.

Interaction of BKA Join With the MRR Functions

BKA join interacts with the MRR functions, respecting the following contract.
The join procedure calls the MRR function `multi_range_read_init`, passing it the callback functions that allow to initialize reading keys for the records in the join buffer and to iterate over these keys. It also passes the parameters of the buffer for MRR needs allocated within the join buffer space.

Then BKA join repeatedly calls the MRR function `multi_range_read_next`. The function works as an iterator function over the records fetched by index look-ups with the keys produced by a callback function set in the call of `multi_range_read_init`. A call to the function `multi_range_read_next` returns the next fetched record through the dedicated record buffer, and the associated reference to the matched record from the join buffer as the output parameter of the function.

Managing Usage of Block-Based Join Algorithms

Currently, 4 different types of block-based join algorithms are supported. For a particular join operation, each of them can be employed with a regular (flat) join buffer or with an incremental join buffer.

Three optimizer switches - `join_cache_incremental`, `join_cache_hashed`, `join_cache_bka` – and the system variable join_cache_level control which of the 8 variants of the block-based algorithms will be used for join operations.

- If `join_cache_bka` is off, then the BKA and BKAH join algorithms are not allowed.

- If `join_cache_hashed` is off, then BNLH and BKAH join algorithms are not allowed.
- If `join_cache_incremental` is off, then no incremental variants of the block-based join algorithms are allowed.

By default, the switches `join_cache_incremental`, `join_cache_hashed`, and `join_cache_bka` are set to 'on'. However, it does not mean that by default any of the block-based join algorithms is allowed to be used. All of them are allowed only if the system variable join_cache_level is set to 8. This variable can take an integer value in the interval from 0 to 8.

If the value is set to 0, no block-based algorithm can be used for a join operation. The values from 1 to 8 correspond to the following variants of block-based join algorithms:

- 1 – Flat BNL
- 2 – Incremental BNL
- 3 – Flat BNLH
- 4 – Incremental BNLH
- 5 – Flat BKA
- 6 – Incremental BKA
- 7 – Flat BKAH
- 8 – Incremental BKAH

If the value of join_cache_level is set to N, any of the block-based algorithms with a level greater than N is disallowed.

So if join_cache_level is set to 5, no usage of BKAH is allowed, and usage of incremental BKA is not allowed either, while usage of all remaining variants is controlled by the settings of the optimizer switches `join_cache_incremental`, `join_cache_hashed`, and `join_cache_bka`.

By default, join_cache_level is set to 2. In other words, only the usage of flat or incremental BNL is allowed.

By default, block-based algorithms can be used only for regular (inner) join operations. To allow them for outer join operations (left outer joins and right outer joins), the optimizer switch `outer_join_with_cache` has to be set to 'on'. Setting the optimizer switch `semijoin_with_cache` to 'on' allows using these algorithms for semi-join operations.

Currently, only incremental variants of the block-based join algorithms can be used for nested outer joins and nested semi-joins.

Size of Join Buffers

The maximum size of join buffers used by block-based algorithms is controlled by setting the join_buffer_size system variable.
This value must be large enough in order for the join buffer employed for a join operation to contain all relevant fields for at least one joined record.

MariaDB 5.3 introduced the system variable join_buffer_space_limit that limits the total memory used for join buffers in a query.

To optimize the usage of the join buffers within the limit set by `join_buffer_space_limit`, one should use the optimizer switch `optimize_join_buffer_size=on`.

When this flag is set to 'off' (default until MariaDB 10.4.2), the size of the used join buffer is taken directly from the join_buffer_size system variable.

When this flag is set to 'on' (default from MariaDB 10.4.3) then the size of the buffer depends on the estimated number of rows in the partial join whose records are to be stored in the buffer.

Related MRR Settings

To use BKA/BKAH join algorithms for InnoDB/MyISAM, one must set the optimizer switch `mrr` to 'on'. When using these algorithms for InnoDB/MyISAM, the overall performance of the join operations can be dramatically improved if the optimizer switch `mrr_sort_keys` is set to 'on'.

This page is licensed: CC BY-SA / Gnu FDL

Condition Selectivity Computation Internals

This page describes how the MariaDB optimizer computes condition selectivities.

calculate_cond_selectivity_for_table(T)

This function computes selectivity of the restrictions on a certain table T.
(TODO: name in the optimizer trace)

Selectivity is computed from

- selectivities of restrictions on different columns (histogram data)
- selectivities of potential range accesses.

Restrictions on different columns, as well as disjoint sets of columns, are considered independent, so their selectivities are multiplied.

Data From Potential Range Accesses

First, we take into account the selectivities of potential range accesses.

If range accesses on indexes IDX1 and IDX2 do not use the same table column (either the indexes do not have common columns, or they do but range accesses do not use them), then they are considered independent, and their selectivities can be multiplied.

However, in general, range accesses on different indexes may use restrictions on the same column and so cannot be considered independent.

In this case, the following approach is used:

We start with selectivity=1, an empty set of range accesses, and an empty set of columns for which we have taken the selectivity into account.

Then, we add range accesses one by one, updating the selectivity value and noting which columns we have taken into account.

Range accesses that use more key parts are added first.

If we are adding a range access $R whose columns do not overlap with the ones already added, we can just multiply the total selectivity by $R's selectivity.

If $R's columns overlap with columns we've got selectivity data for, the process is as follows:

Find the prefix of columns whose selectivity hasn't been taken into account yet.
Then, take the selectivity of the whole range access and multiply it by

```
rec_per_key[i-1]/rec_per_key[i]
```

(TODO: and this logic is not clear. More, one can produce table->cond_selectivity>1 this way. See MDEV-20740)

Data From Histograms

Then, we want to take into account selectivity data from histograms. Each histogram covers one single column.

If the selectivity of a column hasn't been taken into account on the previous step, we take it into account now by multiplying the selectivity by it. Otherwise, we assume that range access has fully taken the column selectivity into account and do nothing.

The third step is sampling-based selectivity data which is out of the scope of this document.

table_cond_selectivity()

This function computes selectivity of restrictions that can be applied after table T has been joined with the join prefix `{T1, ..., Tk}`.

There are two cases:

- Table T uses ref access. In this case, the returned rows match the equalities ref_access is constructed from. Restrictions on just table T are not checked, yet.
- Table T uses ALL/index/quick select. In this case, restrictions on table T have been applied but cross-table restrictions were not.

This page is licensed: CC BY-SA / Gnu FDL

Extended Keys

Syntax

Enable:

```
SET optimizer_switch='extended_keys=on';
```

Disable:

```
SET optimizer_switch='extended_keys=off';
```

Description

Extended Keys is an optimization set with the optimizer_switch system variable, which makes use of existing components of InnoDB keys to generate more efficient execution plans. Using these components in many cases allows the server to generate execution plans which employ index-only look-ups. It is set by default.

Extended keys can be used with:

- ref and eq-ref accesses
- range scans
- index-merge scans
- loose scans
- min/max optimizations

Examples

An example of how extended keys could be employed for a query built over a DBT-3/TPC-H database with one added index defined on `p_retailprice`:

```
SELECT o_orderkey
FROM  part, lineitem, orders
WHERE p_retailprice > 2095 AND o_orderdate='1992-07-01'
      AND o_orderkey=l_orderkey AND p_partkey=l_partkey;
```

The above query asks for the `orderkeys` of the orders placed on 1992-07-01 which contain parts with a retail price greater than $2095.

Using Extended Keys, the query could be executed by the following execution plan:

1. Scan the entries of the index `i_p_retailprice` where `p_retailprice>2095` and read `p_partkey` values from the extended keys.
2. For each value `p_partkey` make an index look-up into the table lineitem employing index `i_l_partkey` and fetch the values of `l_orderkey` from the extended index.
3. For each fetched value of `l_orderkey`, append it to the date `'1992-07-01'` and use the resulting key for an index look-up by index `i_o_orderdate` to fetch the values of `o_orderkey` from the found index entries.

All access methods of this plan do not touch table rows, which results in much better performance.

Here is the explain output for the above query:

```
MariaDB [dbt3sf10]> EXPLAIN
   -> SELECT o_orderkey
   ->   FROM part, lineitem, orders
   ->   WHERE p_retailprice > 2095 AND o_orderdate='1992-07-01'
   ->         AND o_orderkey=l_orderkey AND p_partkey=l_partkey\G
*************************** 1. row ***************************
          id: 1
 select_type: SIMPLE
       table: part
        type: range
possible_keys: PRIMARY,i_p_retailprice
         key: i_p_retailprice
     key_len: 9
         ref: NULL
        rows: 100
       Extra: Using where; Using index
*************************** 2. row ***************************
          id: 1
 select_type: SIMPLE
       table: lineitem
        type: ref
possible_keys: PRIMARY,i_l_suppkey_partkey,i_l_partkey,i_l_orderkey,i_l_orderkey_quantity
         key: i_l_partkey
     key_len: 5
         ref: dbt3sf10.part.p_partkey
        rows: 15
       Extra: Using index
*************************** 3. row ***************************
          id: 1
 select_type: SIMPLE
       table: orders
        type: ref
possible_keys: PRIMARY,i_o_orderdate
         key: i_o_orderdate
     key_len: 8
         ref: const,dbt3sf10.lineitem.l_orderkey
        rows: 1
       Extra: Using index
3 rows in set (0.00 sec)
```

See Also

- MWL#247
- Blog post about the development of this feature

This page is licensed: CC BY-SA / Gnu FDL

MIN/MAX optimization

Min/Max optimization without GROUP BY

MariaDB and MySQL can optimize the MIN() and MAX() functions to be a single row lookup in the following cases:

- There is only one table used in the `SELECT`.
- You only have constants, `MIN()` and `MAX()` in the `SELECT` part.
- The argument to `MIN()` and `MAX()` is a simple column reference that is part of a key.
- There is no `WHERE` clause or the `WHERE` is used with a constant for all prefix parts of the key before the argument to `MIN()` / `MAX()`.
- If the argument is used in the `WHERE` clause, it can be compared to a constant with `<` or `<=` in case of `MAX()` and with `>` or `>=` in case of `MIN()`.

Here are some examples to clarify this.
In this case we assume there is an index on columns `(a,b,c)`

```
SELECT MIN(a),MAX(a) FROM t1
SELECT MIN(b) FROM t1 WHERE a=const
SELECT MIN(b),MAX(b) FROM t1 WHERE a=const
SELECT MAX(c) FROM t1 WHERE a=const AND b=const
SELECT MAX(b) FROM t1 WHERE a=const AND b<const
SELECT MIN(b) FROM t1 WHERE a=const AND b>const
SELECT MIN(b) FROM t1 WHERE a=const AND b BETWEEN const AND const
SELECT MAX(b) FROM t1 WHERE a=const AND b BETWEEN const AND const
```

- Instead of `a=const` the condition `a IS NULL` can be used.

The above optimization also works for subqueries:

```
SELECT x FROM t2 WHERE y= (SELECT MIN(b) FROM t1 WHERE a=const)
```

Cross joins, where there is no join condition for a table, can also be optimized to a few key lookups:

```
SELECT MIN(t1.key_part_1), MAX(t2.key_part_1) FROM t1, t2
```

Min/Max optimization with GROUP BY

MariaDB and MySQL support loose index scan, which can speed up certain `GROUP BY` queries. The basic idea is that when scanning a `BTREE` index (the most common index type for the MariaDB storage engines) we can jump over identical values for any prefix of a key and thus speed up the scan significantly.

Loose scan is possible in the following cases:

- The query uses only one table.
- The `GROUP BY` part only uses indexed columns in the same order as in the index.
- The only aggregated functions in the `SELECT` part are `MIN()` and `MAX()` functions and all of them using the same column which is the next index part after the used `GROUP BY` columns.
- Partial indexed columns cannot be used (like only indexing 10 characters of a `VARCHAR(20)` column).

Loose scan will apply for your query if EXPLAIN shows `Using index for group-by` in the `Extra` column.
In this case the optimizer will do only one extra row fetch to calculate the value for `MIN()` or `MAX()` for every unique key prefix.

The following examples assume that the table `t1` has an index on `(a,b,c)`.

```
SELECT a, b, MIN(c),MAX(c) FROM t1 GROUP BY a,b
```

See also

- MIN()
- MAX()
- MySQL manual on loose index scans

This page is licensed: CC BY-SA / Gnu FDL

Notes When an Index Cannot Be Used

MariaDB starting with 10.6.16

This is a new note added in 10.6.16.

Warning About Incompatible Index Comparison

A frequent mistake database developers make is to compare an indexed column with another column that is not compatible with the indexed column. For example, comparing string columns with number columns, or using incompatible character sets or collations.

Because of this we have introduced notes (low severity warnings) when an indexed column cannot use the index to lookup rows.

The warnings are of different types:

If one compares an indexed column with a value of a different type one, will get a warning like the following:

```
Note   1105    Cannot use key `PRIMARY` part[0] for lookup: `test`.`t1`.`id` of
  type `char` = "1" of type `bigint`
```

If one compares indexed character columns with a value of an incompatible collation, one will get a warning like the following:

```
Note   1105    Cannot use key `s2` part[0] for lookup: `test`.`t1`.`s2` of
  collation `latin1_swedish_ci` = "'a' collate latin1_german1_ci" of collation `latin1_german1_ci`
```

Note that in MariaDB 10.6 to MariaDB 11.3 we will use the error 1105 (Unknown error), as we cannot add an new error code in a GA version. In MariaDB 11.4 we will change this to be a unique error code.

Enabling the Note

By default, the warning is only shown when executing EXPLAIN on a query.
To enable for all queries, use the option/server variable:

```
In config file:
--note-verbosity=all

As a server variable:
@@note_verbosity="all";
```

note_verbosity describes with note categories one want to get notes for. Be aware that if the old sql_notes variable is 0, one will not get any notes.

It can have one or many of the following options:

Option	Description
basic	All old notes.
unusable_keys	Give warnings for unusable keys for SELECT, DELETE and UPDATE.
explain	Give warnings about unusable keys for EXPLAIN.

One can also set `note_verbosity` to the value of `all` to set all options.

Enabling Warnings and Notes for the Slow Query Log

One can get the note about incompatible keys also in the slow query log by adding the option `warnings` in the `log_slow_verbosity` option/variable. It will automatically be enabled if one uses `log_slow_verbosity=all`.

```
In config file:
--log-slow-verbosity=warnings

As a server variable:
@@log_slow_verbosity="all";
```

See Also

- Type conversions

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Debugging With GDB

Some useful things for debugging optimizer code.

Useful Print Functions

- `dbug_print_item()` prints the contents of an `Item` object into a buffer and returns pointer to it.
- `dbug_print_sel_arg()` prints an individual `SEL_ARG` object (NOT the whole graph or tree) and returns pointer to the buffer holding the printout.
- `dbug_print_table_row` prints the current row buffer of the given table.
- There are more `dbug_print_XX` functions for various data structures

Printing the Optimizer Trace

The optimizer trace is collected as plain text.
One can print the contents of the trace collected so far as follows:

```
printf "%s\n", thd->opt_trace->current_trace->current_json->output.str.Ptr
```

Starting from 11.0, there is `dbug_print_opt_trace()` function which one call from gdb.

Printing Current Partial Join Prefix

`best_access_path()` is a function that adds another table to the join prefix.

When in or around that function, the following can be useful:

A macro to print the join prefix already constructed:

```
define bap_print_prefix
  set $i=0
  while ($i < idx)
    p join->positions[$i++].table->table->alias.Ptr
  end
end
```

Other Settings

- May need to set innodb_fatal_semaphore_wait_threshold to be high enough?

Other development topics

mtr tests

In order to get innodb to have stable statistics, use this in the test:

```
--source include/innodb_stable_estimates.inc
```

Also consider `set global innodb_max_purge_lag_wait=0;`
(TODO: how these two compare?)

Coding guidelines

Run this to get your patch auto-formatted according to our coding style in `.clang-format` :

```
git diff -U0 --no-color --relative HEAD^ | clang-format-diff -p1 -i
```

See Also

- How to collect large optimizer traces

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Development

Notes about Optimizer Development

mysql-test

InnoDB Estimates are unstable

This is caused by background statistics update. It may cause the numbers in EXPLAIN output to be off-by-one. It may also cause different query plans to be picked on different runs (See e.g. MDEV-32901 for details)

On a per-table basis, one can use `STATS_AUTO_RECALC=0` as table parameter.

On a per-file basis, one can use this include:

```
--source mysql-test/include/innodb_stable_estimates.inc
```

Run mtr with Optimizer Trace enabled

TODO

This page is licensed: CC BY-SA / Gnu FDL

optimizer_max_sel_arg_weight

Basics

As mentioned in the Range Optimizer, ranges on multiple key parts can create a combinatorial amount of ranges.

`optimizer_max_sel_arg_weight` setting is a limit to reduce the number of ranges generated by dropping restrictions on higher key parts if the number of ranges becomes too high.

(Note that there is also `optimizer_max_sel_args` which limits the number of intermediary SEL_ARG objects that can be created. This is a different limitation)

Combinatorial number of ranges

Let's reuse the example from the Range Optimizer page.

```
CREATE TABLE t2 (
  keypart1 INT,
  keypart2 VARCHAR(100),
  keypart3 INT,
  INDEX idx(keypart1, keypart2, keypart3)
);
```

```
SELECT * FROM t2
WHERE
  keypart1 IN (1,2,3,4,5,6,7,8,9,10) AND keypart2 IN ('a','b', 'c') AND keypart3 IN (1,2,3,4);
```

Range optimizer will produce 10 * 3 * 4 = 120 ranges.

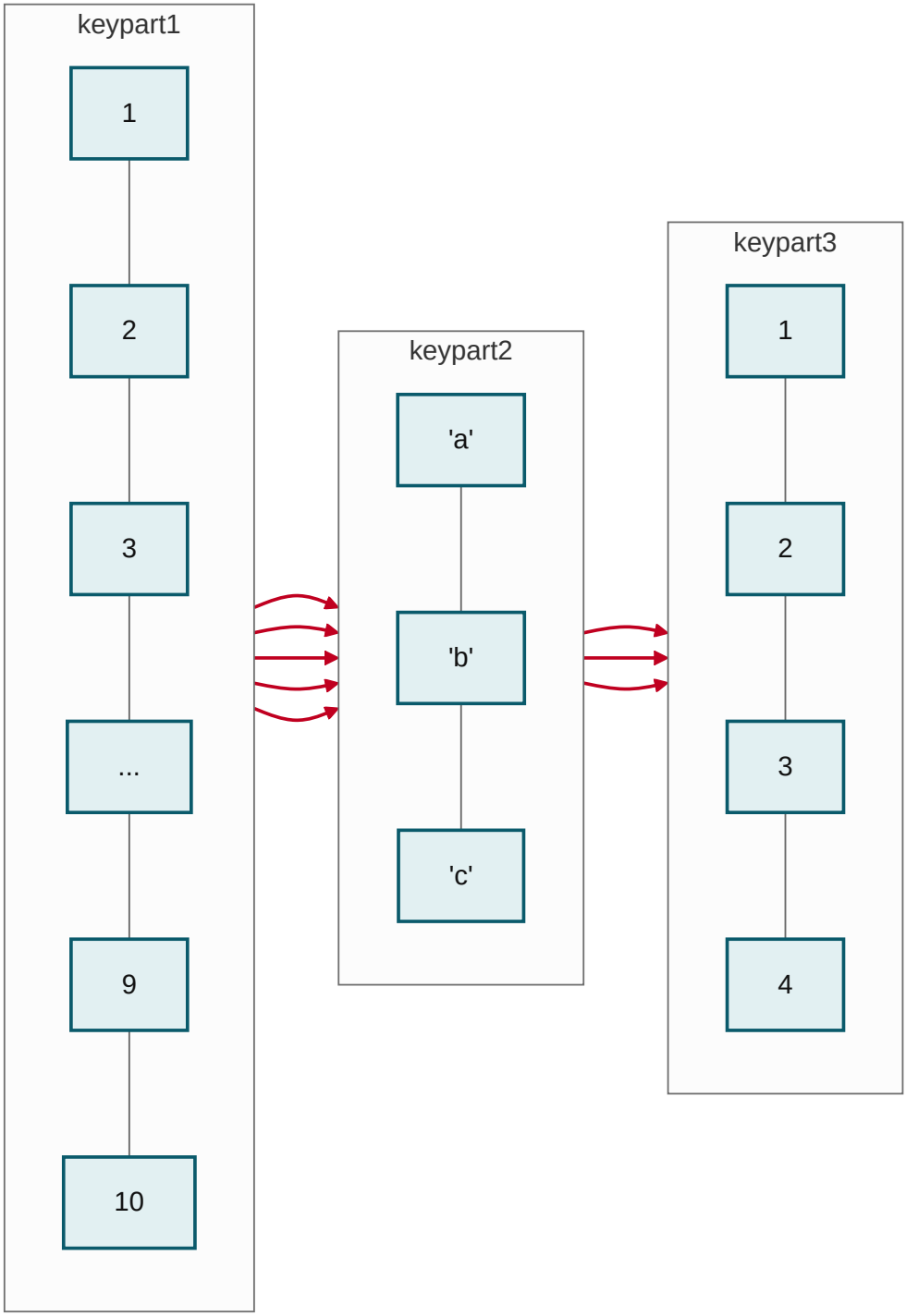
```
SELECT * FROM information_schema.optimizer_trace\G
```

```
//...
                    "range_scan_alternatives": [
                      {
                        "index": "idx",
                        "ranges": [
                          "(1,a,1) <= (keypart1,keypart2,keypart3) <= (1,a,1)",
                          "(1,a,2) <= (keypart1,keypart2,keypart3) <= (1,a,2)",
                          "(1,a,3) <= (keypart1,keypart2,keypart3) <= (1,a,3)",
                          "(1,a,4) <= (keypart1,keypart2,keypart3) <= (1,a,4)",
                          "(1,b,1) <= (keypart1,keypart2,keypart3) <= (1,b,1)",
                          //... # 114 lines omitted ...
                           "(3,b,3) <= (keypart1,keypart2,keypart3) <= (3,b,3)",
                          "(3,b,4) <= (keypart1,keypart2,keypart3) <= (3,b,4)",
                         ],
```

This number is fine but if your IN-list are thousands then the number of ranges can in the millions which may cause excessive CPU or memory usage (Note: this however is avoided in some cases when IN-predicate is converted into subquery. But there are cases when that is not done)

SEL_ARG graph

Internally, the Range Optimizer builds this kind of graph:



The SEL_ARG graph built for `keypart1 IN (1,2,...,10) AND keypart2 IN ('a','b','c') AND keypart3 IN (1,2,3,4)`*. Black links chain adjacent intervals on the same key part; red links connect a key part to the next key part.*

Vertical black lines connect adjacent “intervals” on the same key part.
Red lines connect a key part to a subsequent key part.

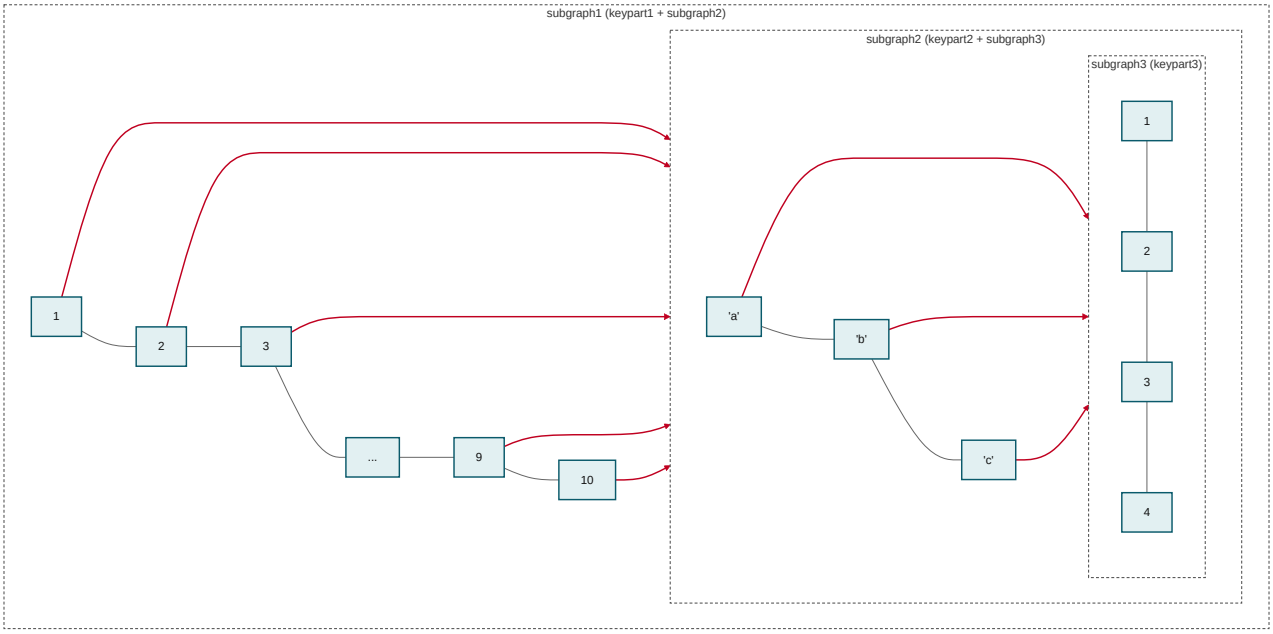
To produce ranges, one walks this graph by starting from left most corner.
Walking right “attaches” the ranges on one key part to another to form multi-part ranges. One must mind that Not all combinations produce multi-part ranges, though.

Walking top-to-bottom produces adjacent ranges.

Weight of SEL_ARG graph

How do we limit the number of ranges?
We should remove the parts of SEL_ARG graph that describe ranges on big key parts.
That way, we can still build ranges, although we will build fewer ranges that may contain more rows.

Due to the way the graph is constructed, we cannot tell how many ranges it would produce, so we introduce a parameter “weight” which is easy to compute and is roughly proportional to the number of ranges we estimate to produce.



The same graph annotated with the nested subgraphs used to compute its weight: subgraph3 (keypart3) sits inside subgraph2 (keypart2 + subgraph3), which sits inside subgraph1 (keypart1 + subgraph2).

Here is how the weight is computed:

- The weight of subgraph3 is just the number of nodes, 4.
- The weight of subbraph2 the number of nodes for keypart2 (3), and the weight of subgraph1 multiplied by 3 since there are 3 references to it.
- The weight of subgraph1 is the number of nodes for keypart1 (10) plus the weight of subgraph2 multiplied by 10 since there are 10 references to it.

Here the total weight is 160 which has the same order of magnitude as the number of ranges.

SEL_ARG graphs are constructed for all parts of WHERE clause and are AND/ORed according to the AND/OR structure of the WHERE clause (after normalization). If the optimizer notices that it has produced a SEL_ARG graph that exceeds the maximum weight, the parts of the graph describing higher key parts are removed until the weight is within the limit.

Example of effect of limiting weight

Continuing with our example:

```
-- This is very low, don't use in production:
SET @@optimizer_max_sel_arg_weight=50;
SELECT * FROM t2 WHERE keypart1 IN (1,2,3,4,5,6,7,8,9,10) AND keypart2 IN ('a','b', 'c') AND keypart3 IN
(1,2,3,4);
SELECT * FROM information_schema.optimizer_trace\G
```

shows

```
"range_scan_alternatives": [
                      {
                        "index": "idx",
                        "ranges": [
                          "(1,a) <= (keypart1,keypart2) <= (1,a)",
                          "(1,b) <= (keypart1,keypart2) <= (1,b)",
                         // (30 lines in total)
                          "(10,b) <= (keypart1,keypart2) <= (10,b)",
                          "(10,c) <= (keypart1,keypart2) <= (10,c)"
                        ],
```

One can see that now the range list is much smaller, 30 lines instead of 120. This was achieved by discarding the restrictions on `keypart3`.

This page is licensed: CC BY-SA / Gnu FDL

Range Optimizer

Range optimizer is a part of MariaDB (and MySQL) optimizer that takes as input

- the table and index definition(s)
- the WHERE condition (or ON expression if the table is inner in an outer join)

and constructs list of ranges one can scan in an index to read the rows that match the WHERE condition, or a superset of it. It can also construct an “index_merge” plan, where one needs ranges from two or more indexes to compute a union (formed from multiple condition disjunctions) and/or intersection (formed from multiple condition conjunctions).

Basic example

Consider a table

```
CREATE TABLE t1 (
  key1 INT,
  key2 VARCHAR(100),
  ...
  INDEX(key1),
  INDEX(key2)
);
```

and query

```
-- Turn on optimizer trace so we can see the ranges:
SET optimizer_trace=1;
EXPLAIN SELECT * FROM t1 WHERE key1<10 AND key2='foo';
SELECT * FROM information_schema.optimizer_trace\G
```

This shows the ranges that the optimizer was able to infer:

```
"range_scan_alternatives": [
                      {
                        "index": "key1",
                        "ranges": ["(NULL) < (key1) < (10)"],
                        ...
                      },
                      {
                        "index": "key2",
                        "ranges": ["(foo) <= (key2) <= (foo)"],
                        ...
                      }
                    ],
```

Ranges are non-overlapping

Range optimizer produces a list of ranges without overlaps. Consider this WHERE clause where conditions do have overlaps:

```
SELECT * FROM t1 WHERE (key1 BETWEEN 10 AND 20  AND key1 > 14)  OR key1 IN (17, 22, 33);
SELECT * FROM information_schema.optimizer_trace\G
```

We get

```
...
                  "analyzing_range_alternatives": {
                    "range_scan_alternatives": [
                      {
                        "index": "key1",
                        "ranges": [
                          "(14) < (key1) <= (20)",
                          "(22) <= (key1) <= (22)",
                          "(33) <= (key1) <= (33)"
                        ],
```

Ranges for multi-part indexes

Let's consider an index with multiple key parts. (note: due to Extended Keys optimization, an index may have more key parts than you've explicitly defined)

```
CREATE TABLE t2 (
  keypart1 INT,
  keypart2 VARCHAR(100),
  keypart3 INT,
  INDEX idx(keypart1, keypart2, keypart3)
);
```

Range optimizer will generate a *finite* set of ranges over lexicographical ordering over `(keypart1, keypart2, ...)`.

Example:

```
SELECT * FROM t2 WHERE keypart1 IN (1,2,3) AND keypart2 BETWEEN 'bar' AND 'foo';
SELECT * FROM information_schema.optimizer_trace\G
```

gives

```
"range_scan_alternatives": [
                      {
                        "index": "idx",
                        "ranges": [
                          "(1,bar) <= (keypart1,keypart2) <= (1,foo)",
                          "(2,bar) <= (keypart1,keypart2) <= (2,foo)",
                          "(3,bar) <= (keypart1,keypart2) <= (3,foo)"
                        ],
```

Compare with a similar query:

```
SELECT * FROM t2 WHERE keypart1 BETWEEN 1 AND 3 AND keypart2 BETWEEN 'bar' AND 'foo';
SELECT * FROM information_schema.optimizer_trace\G
```

this will generate just one bigger range:

```
"range_scan_alternatives": [
                      {
                        "index": "idx",
                        "ranges": ["(1,bar) <= (keypart1,keypart2) <= (3,foo)"],
                        ...
```

which includes for example rows like `(keypart1,keypart2)=(1,zzz)`. One could argue that the optimizer could be able to figure out that for condition `keypart1 between 1 and 3` the only possible values are 1, 2, 3 but this is not implemented.

Not all comparisons produce ranges

Note that some keypart comparisons produce multi-part ranges while some do not.
The governing rule is the same: the conditions together must produce an interval (or a finite number of intervals) in lexicographic order in the index.

Some examples:

```
WHERE keypart1<= 10 AND keypart2<'foo'
```

can use the second keypart:

```
"ranges": ["(NULL) < (keypart1,keypart2) < (10,foo)"],
```

but the interval will still include rows like `(keypart1, keypart2) = (8, 'zzzz')`

Non-inclusive bound on keypart1 prevents any use of keypart2. For

```
WHERE keypart1< 10 keypart2<'foo';
```

we get

```
"ranges": ["(NULL) < (keypart1) < (10)"]
```

Non-agreeing comparison (less than and greater than) do not produce a multi-part range:

```
WHERE keypart1<= 10 AND keypart2>'foo';
```

gives

```
"ranges": ["(NULL) < (keypart1) <= (10)"],
```

A “Hole” in keyparts means higher keypart cannot be used.

```
WHERE keypart1= 10 AND keypart3<='foo';
```

gives

```
"ranges": ["(10) <= (keypart1) <= (10)"]
```

Combinatorial blow-ups

For multi-part keys, range analyzer can produce ranges that are “tight”, that is, they only include rows that will match the WHERE condition.
On the other hand, some SQL constructs can produce very large (combinatorial) amounts of ranges.
Consider a query

```
SELECT * FROM t2 WHERE keypart1 IN (1,2,3,4) AND keypart2 IN ('a','b', 'c')
```

two IN-lists produce 3*4 =12 ranges:

```
"range_scan_alternatives": [
                      {
                        "index": "idx",
                        "ranges": [
                          "(1,a) <= (keypart1,keypart2) <= (1,a)",
                          "(1,b) <= (keypart1,keypart2) <= (1,b)",
                          "(1,c) <= (keypart1,keypart2) <= (1,c)",
                          "(2,a) <= (keypart1,keypart2) <= (2,a)",
                          "(2,b) <= (keypart1,keypart2) <= (2,b)",
                          "(2,c) <= (keypart1,keypart2) <= (2,c)",
                          "(3,a) <= (keypart1,keypart2) <= (3,a)",
                          "(3,b) <= (keypart1,keypart2) <= (3,b)",
                          "(3,c) <= (keypart1,keypart2) <= (3,c)",
                          "(4,a) <= (keypart1,keypart2) <= (4,a)",
                          "(4,b) <= (keypart1,keypart2) <= (4,b)",
                          "(4,c) <= (keypart1,keypart2) <= (4,c)"
                        ],
```

if one adds `and keypart3 IN (1,2,3,4,5)`, the amount of ranges will be 3*4*5=60 and so forth. See optimizer_max_sel_arg_weight on how to combat this.

This page is licensed: CC BY-SA / Gnu FDL

The Optimizer Cost Model from MariaDB 11.0

Background

Before MariaDB 11.0, the MariaDB Query optimizer used a ‘basic cost’ of 1 for:

- One disk access
- Fetching a key
- Fetching a row based on the rowid (= unique row identifier) from the key

There were some smaller costs:

- filter lookup: 0.01
- Examining a where clause: 0.20
- Comparing two keys: 0.05
- Fetching a row through an index from a temporary memory table: 0.05

The above costs are reasonable for finding out the best index to use.
However, they where not good for finding out if we should use a table scan, index scan or range lookup. The cost for

the different engines were not properly calibrated.

New Cost Model

In MariaDB 11.0 we have fixed the above shortcomings by changing the basic cost for 'storage engine operations' to be 1 millisecond. This means that for most queries the query cost (`LAST_QUERY_COST`) should be close (or at least proportional) to the time the server is spending in the storage engine + join_cache + sorting.

Note that the user level costs are in **microseconds** (as milliseconds would have so many zero's that it makes it hard to compare values).

The engine costs have also been separated into smaller parts to make things more accurate.

The "disk"-read cost now assumes a mid level SSD disk with 400MB/second. This can be changed by the end user by modifying `OPTIMIZER_DISK_READ_COST`.

All engine specific costs are visible in `information_schema.optimizer_costs`.

For example:

The "default" cost for an engine can be found with:

```
SELECT * FROM information_schema.optimizer_costs WHERE engine="DEFAULT"\G
*************************** 1. row ***************************
                        ENGINE: DEFAULT
       OPTIMIZER_DISK_READ_COST: 10.240000
OPTIMIZER_INDEX_BLOCK_COPY_COST: 0.035600
     OPTIMIZER_KEY_COMPARE_COST: 0.011361
        OPTIMIZER_KEY_COPY_COST: 0.015685
      OPTIMIZER_KEY_LOOKUP_COST: 0.435777
   OPTIMIZER_KEY_NEXT_FIND_COST: 0.082347
      OPTIMIZER_DISK_READ_RATIO: 0.020000
        OPTIMIZER_ROW_COPY_COST: 0.060866
      OPTIMIZER_ROW_LOOKUP_COST: 0.130839
   OPTIMIZER_ROW_NEXT_FIND_COST: 0.045916
   OPTIMIZER_ROWID_COMPARE_COST: 0.002653
      OPTIMIZER_ROWID_COPY_COST: 0.002653
```

The above costs are the default (base) for all engines and should be reasonable for engines that does not have a clustered index (like MyISAM, Aria etc). The default costs can be changed by specifying just the cost as an argument, like `mariadbd --optimizer-disk-read-cost=20` or from SQL: `set global optimizer_disk_read_cost=20`. An engine specific cost can be tuned by prefixing the cost with the engine name, like `set global innodb.optimizer_disk_read_cost=20`.

An engine can tune some or all of the above cost in the storage engine interface.
Here follows the cost for the InnoDB storage engine.

```
SELECT * FROM information_schema.optimizer_costs WHERE engine="innodb"\G
*************************** 1. row ***************************
                        ENGINE: InnoDB
       OPTIMIZER_DISK_READ_COST: 10.240000
OPTIMIZER_INDEX_BLOCK_COPY_COST: 0.035600
     OPTIMIZER_KEY_COMPARE_COST: 0.011361
        OPTIMIZER_KEY_COPY_COST: 0.015685
      OPTIMIZER_KEY_LOOKUP_COST: 0.791120
   OPTIMIZER_KEY_NEXT_FIND_COST: 0.099000
      OPTIMIZER_DISK_READ_RATIO: 0.020000
        OPTIMIZER_ROW_COPY_COST: 0.060870
      OPTIMIZER_ROW_LOOKUP_COST: 0.765970
   OPTIMIZER_ROW_NEXT_FIND_COST: 0.070130
   OPTIMIZER_ROWID_COMPARE_COST: 0.002653
      OPTIMIZER_ROWID_COPY_COST: 0.002653
```

As can be seen, the `ROW_LOOKUP_COST` is close to the `KEY_LOOKUP_COST`, which is because InnoDB has clustered primary key indexes and is using it to find the row from a secondary index.

Some engines, like `HEAP` / `MEMORY` implement their own cost functions as different indexes in the same engine can have different costs. This is why some of the cost numbers for these engines are 0.

There are also some SQL level costs that are independent of the storage engine:

```
SELECT * FROM information_schema.global_variables WHERE variable_name LIKE "%WHERE%cost%" OR
variable_name LIKE "%scan%cost%";
+--------------------------+----------------+
| VARIABLE_NAME            | VARIABLE_VALUE |
+--------------------------+----------------+
| OPTIMIZER_SCAN_SETUP_COST | 10.000000      |
| OPTIMIZER_WHERE_COST      | 0.032000       |
+--------------------------+----------------+
```

Description of the Different Cost Variables

Time and cost are quite interchangeable in the new cost model. Below we will use cost for most things, except for `OPTIMIZER_DISK_READ_COST` as one should use published/tested timings for the SSD/harddisk if one wants to change the value..

Variable	Type	Description
OPTIMIZER_DISK_READ_COST	Engine	Time in microseconds to read a 4K block from a disk/SSD. The default is set for a 400MB/second SSD
OPTIMIZER_INDEX_BLOCK_COPY_COST	Engine	Cost to lock and a copy a block from the global cache to a local cache. This cost is added for every block accessed, independent of whether they are cached or not
OPTIMIZER_KEY_COMPARE_COST	Engine	Cost to compare two keys
OPTIMIZER_KEY_COPY_COST	Engine	Cost to copy a key from the index to a local buffer as part of searching for a key
OPTIMIZER_KEY_LOOKUP_COST	Engine	Cost to find a key entry in the index (index read)
OPTIMIZER_KEY_NEXT_FIND_COST	Engine	Cost to find the next key in the index (index next)
OPTIMIZER_DISK_READ_RATIO	Engine	The ratio of BLOCK_NOT_IN_CACHE/CACHE_READS. The cost of disk usage is calculated as estimated_blocks * OPTIMIZER_DISK_READ_RATIO * OPTIMIZER_DISK_READ_COST. A value of 0 means that all blocks are always in the cache. A value of 1 means that a block is never in the cache
OPTIMIZER_ROW_COPY_COST	Engine	Cost of copying a row to a local buffer. Should be slightly more than OPTIMIZER_KEY_COPY_COST
OPTIMIZER_ROW_LOOKUP_COST	Engine	Cost to find a row based on the rowid (Rowid is stored in the index together with the key)
OPTIMIZER_ROW_NEXT_FIND_COST	Engine	Cost of finding the next row

Variable	Type	Description
OPTIMIZER_ROWID_COMPARE_COST	Engine	Cost of comparing two rowids
OPTIMIZER_ROWID_COPY_COST	Engine	Cost of copying a rowid from the index
OPTIMIZER_SCAN_SETUP_COST	Session	Cost of starting a table or index scan. This has a low value to encourage the optimizer to use index lookup also tables with very few rows
OPTIMIZER_WHERE_COST	Session	Cost to execute the WHERE clause for every found row. Increasing this variable will encourage the optimizer to find plans which read fewer rows

More information of the costs and how they were calculated can be found in the `Docs/optimizer_costs.txt` file in the MariaDB Source distributions.

Other Optimizer Cost Changes

- When counting disk accesses, we assume that all rows and index data are cached for the duration of the query. This is to avoid the following problem:
  - table t1 with 1 million_rows is scanned
    - For each row we do a lookup in table t2, which has only 10 rows

If we would count all lookups in t2, there would be 1 million lookups. If this would be the case, the optimizer would choose to use a join cache on the rows in t1 and do a table scan over t2.

- The cost of sorting (filesort) is now more accurate, which allows the optimizer to better choose between index scan and filesort for `ORDER BY/GROUP BY` queries.

A lot of rule-based cost has been changed to be cost-based:

- The decision to use an index (and which index) for resolving `ORDER BY/GROUP BY` were only partly cost-based before.
- The old optimizer would limit the number of 'expected key lookups' to 10% of the number of rows. This would cause the optimizer to use an index to scan a big part of a table when a full table scan would be much faster. This code is now removed.
- InnoDB would limit the number of rows in a range to 50% of the total rows, which would confuse the optimizer for big ranges. The cap is now removed.
- If there was a usable filter for an index, it was sometimes used without checking the complete cost of the filter.
- 'Aggregate distinct optimization with indexes' is now cost-based. This will change many queries from "Using index for group-by (scanning)" to "Using index for group-by".

Other Notable Plan Changes

- Indexes can now be used for `ORDER BY/GROUP BY` in sub queries (instead of filesort)
- Derived tables and queries with `UNION` can now create a distinct key (instead of a key with duplicates) to speed up key accesses.
- Indexes with more used key parts are preferred if the number of resulting rows is the same:
  - `WHERE key_part_1 = 1 and key_part_2 < 10`
  - This will now use a `RANGE` over both key parts instead of using lookups on key_part_1.
- For very small tables, index lookup is preferred over table scan.

- `EXPLAIN` does not report “Using index” for scans using a clustered primary key as technically this a table scan.

When the Optimizer Changes Matter

The new, improved optimizer should be able to find a better plan

- If you are using queries with more than two tables.
- If you have indexes with a lot of identical values.
- If you are using ranges that cover more than 10% of a table.
  - `WHERE key between 1 and 1000 -- Table has values 1-2000`
- If you have complex queries when not all used columns are or can be indexed.
  - In which case you may need to depend on selectivity to get the right plan.
- If you are using queries mixing different storage engines.
  - Like using tables from both InnoDB and Memory in the same query.
- If you have had to use `FORCE INDEX` to get a good plan.
- If using ANALYZE TABLE made your plans worse (or not good enough).
- If your queries have lots of derived tables (subselects).
- You are using `ORDER BY / GROUP BY` that could be resolved via indexes.

Changing Costs

All engine and “SQL level” cost variables can be changed via
MariaDB startup options, in configuration files or dynamically using SQL.

In Configuration Files (and Command Line)

```
[mariadbd]
# Archive is using a hard disk (typical seek is 8-10 ms)
archive.OPTIMIZER_DISK_READ_COST=8000
# All other engines are using an SSD.
OPTIMIZER_DISK_READ_COST=10.240000
```

From SQL

```
# Tell optimizer TO find a plan WITH AS few accepted ROWS AS possible
SET SESSION OPTIMIZER_WHERE_COST=1.0;
# Inform the optimizer that InnoDB buffer pool has a 80% hit rate
SET GLOBAL innodb.OPTIMIZER_DISK_READ_RATIO=0.20;
```

- Note engine costs are `GLOBAL` while other costs can also be `SESSION`.
- To keep things fast, engine-specific costs are stored in the table definition (TABLE_SHARE). One effect of this is that if one changes the cost for an engine, it will only take effect when new, not previously cached tables are accessed. You can use FLUSH TABLES to force the table to use the new costs at next access.

Examples of Changing Costs

- `OPTIMIZER_WHERE_COST` is added as a cost for all ‘accepted rows’. Increasing this variable will cause the optimizer to choose plans with less estimated rows.
- One can specify the kind of disk used by the system by changing `OPTIMIZER_DISK_READ_COST`. This should be the time to do a random read of a 4096 byte block.
- The cost of a potential disk read is calculated as `OPTIMIZER_DISK_READ_COST * OPTIMIZER_DISK_READ_RATIO`. Increasing `OPTIMIZER_DISK_READ_RATIO` will inform the optimizer that not all data is cached.

- `OPTIMIZER_SCAN_SETUP_COST` will increase the cost of a table scan. One can increase this to avoid using table scans.

For Storage Engine Developers

The costs for an engine are set the following way when the engine plugin is loaded/initialized:

- Copy the "default" storage engine costs to the plugin engine costs.
  - #handlerton->costs `points to the engine specific cost data.`
- Call `handlerton->update_optimizer_costs()` to let the storage engine update the costs.
- Apply all user specific engine costs (from configuration files/startup) to the engine costs structure.
- When a TABLE_SHARE is created, the costs are copied from `handlerton->costs` to `TABLE_SHARE.optimizer_costs` . `handler::update_optimizer_costs()` is called to allow the engine to tune the cost for this specific table instance. This is done to avoid having to take any "cost" mutex while running queries.
- User changes to engine costs are stored in the data pointed to by `handlerton->costs` . This is why FLUSH TABLES is needed to activate new engine costs.
- To speed up cost access for the optimizer, `handler::set_optimizer_costs()` is called for each query to copy `OPTIMIZER_WHERE_COST` and `OPTIMIZER_SCAN_SETUP_COST` to the engine cost structure.

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Trace

See Basic Optimizer Trace Example.

Shows a complete optimizer trace example, from enabling optimizer_trace to reading the resulting JSON from INFORMATION_SCHEMA.OPTIMIZER_TRACE.

See How to Collect Large Optimizer Traces.

Explains how to collect large optimizer traces by raising max_allowed_packet and optimizer_trace_max_mem_size.

See Optimizer Trace for Developers.

Provides guidelines for server developers on what and how to write to the optimizer trace, including keeping the JSON valid.

See Optimizer Trace Guide.

Guides you through reading the JSON optimizer trace, using a simple query to illustrate the join_preparation and join_optimization steps.

See Optimizer Trace Overview.

Introduces optimizer trace, which produces a JSON document of the optimizer's decisions for SELECT/UPDATE/DELETE queries, its system variables, and the INFORMATION_SCHEMA.OPTIMIZER_TRACE table.

See Optimizer Trace Resources.

Links to external optimizer trace resources, including a MariaDB Fest 2020 walkthrough talk and the opttrace processing tool.

This page is licensed: CC BY-SA / Gnu FDL

Basic Optimizer Trace Example

```
MariaDB> set optimizer_trace='enabled=on';

MariaDB> select * from t1 where a<10;

MariaDB> select * from information_schema.optimizer_trace limit 1\G
*************************** 1. row ***************************
                            QUERY: select * from t1 where a<10
                            TRACE: {
  "steps": [
    {
      "join_preparation": {
        "select_id": 1,
        "steps": [
          {
            "expanded_query": "select t1.a AS a,t1.b AS b,t1.c AS c from t1 where t1.a < 10"
          }
        ]
      }
    },
    {
      "join_optimization": {
        "select_id": 1,
        "steps": [
          {
            "condition_processing": {
              "condition": "WHERE",
              "original_condition": "t1.a < 10",
              "steps": [
                {
                  "transformation": "equality_propagation",
                  "resulting_condition": "t1.a < 10"
                },
                {
                  "transformation": "constant_propagation",
                  "resulting_condition": "t1.a < 10"
                },
                {
                  "transformation": "trivial_condition_removal",
                  "resulting_condition": "t1.a < 10"
                }
              ]
            }
          },
          {
            "table_dependencies": [
              {
                "table": "t1",
                "row_may_be_null": false,
                "map_bit": 0,
                "depends_on_map_bits": []
              }
            ]
          },
          {
            "ref_optimizer_key_uses": []
          },
          {
            "rows_estimation": [
              {
                "table": "t1",
                "range_analysis": {
                  "table_scan": {
                    "rows": 1000,
                    "cost": 206.1
                  },
                  "potential_range_indexes": [
                    {
```

```
            "index": "a",
            "usable": true,
            "key_parts": ["a"]
          },
          {
            "index": "b",
            "usable": false,
            "cause": "not applicable"
          }
        ],
        "setup_range_conditions": [],
        "group_index_range": {
          "chosen": false,
          "cause": "no group by or distinct"
        },
        "analyzing_range_alternatives": {
          "range_scan_alternatives": [
            {
              "index": "a",
              "ranges": ["(NULL) < (a) < (10)"],
              "rowid_ordered": false,
              "using_mrr": false,
              "index_only": false,
              "rows": 10,
              "cost": 13.751,
              "chosen": true
            }
          ],
          "analyzing_roworder_intersect": {
            "cause": "too few roworder scans"
          },
          "analyzing_index_merge_union": []
        },
        "chosen_range_access_summary": {
          "range_access_plan": {
            "type": "range_scan",
            "index": "a",
            "rows": 10,
            "ranges": ["(NULL) < (a) < (10)"]
          },
          "rows_for_plan": 10,
          "cost_for_plan": 13.751,
          "chosen": true
        }
      }
    },
    {
      "selectivity_for_indexes": [
        {
          "index_name": "a",
          "selectivity_from_index": 0.01
        }
      ],
      "selectivity_for_columns": [],
      "cond_selectivity": 0.01
    }
  ]
},
{
  "considered_execution_plans": [
    {
      "plan_prefix": [],
      "table": "t1",
      "best_access_path": {
        "considered_access_paths": [
          {
            "access_type": "range",
            "resulting_rows": 10,
            "cost": 13.751,
```

```
                      "chosen": true
                    }
                  ]
                }
              }
            ]
          },
          {
            "attaching_conditions_to_tables": {
              "original_condition": "t1.a < 10",
              "attached_conditions_computation": [],
              "attached_conditions_summary": [
                {
                  "table": "t1",
                  "attached": "t1.a < 10"
                }
              ]
            }
          }
        ]
      }
    },
    {
      "join_execution": {
        "select_id": 1,
        "steps": []
      }
    }
  ]
}
MISSING_BYTES_BEYOND_MAX_MEM_SIZE: 0
          INSUFFICIENT_PRIVILEGES: 0
```

This page is licensed: CC BY-SA / Gnu FDL

How to Collect Large Optimizer Traces

Optimizer traces can be large for some queries.

In order to collect a large trace, you need to perform the following steps (using 128 MB as an example):

```
set global max_allowed_packet=128*1024*1024;
```

Reconnect specifying `--max-allowed-packet=128000000` for the client as well.

```
set optimizer_trace=1;
set optimizer_trace_max_mem_size=127*1024*1024;
```

Now, one can run the query and save the large trace.

See Also

- optimizer_trace system variable
- optimizer_trace_max_mem_size system variable
- max_allowed_packet system variable

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Trace for Developers

This article describes guidelines for what/how to write to Optimizer Trace when doing server development.

Basic considerations

The trace is a “structured log” of what was done by the optimizer. Prefer to do tracing as soon as a rewrite/decision is made (instead of having a separate trace_something() function).

Generally, a function should expect to find the trace in a state where we’re writing an array. The rationale is that array elements are ordered, while object members are not (even if they come in a certain order in the JSON text). We’re writing a log, so it’s natural for different entries to form an array.

Typically you’ll want to start an unnamed object, then use member names to show what kind of entry you’re about to write:

```
[
  ...,  # Something before us
  {
    "my_new_rewrite": {
       "from": "foo",
       "to": "bar",
       ...
    }
  }
  ...
```

(TODO other considerations)

Making sure the trace is valid

`Json_writer_object` and `Json_writer_array` classes use RAII idiom and ensure that JSON objects and arrays are “closed” in the reverse order they were started.

However, they do not ensure these constraints:

- JSON objects must have named members.
- JSON arrays must have unnamed members.

Tracing code has runtime checks for these. Attempt to write invalid JSON will cause assertion failure.

Test coverage

It is possible to run `mysql-test-run` with this argument

```
--mysqld=--optimizer_trace=enabled=on
```

This will run all tests with tracing on. As mentioned earlier, debug build will perform checks that we are not producing invalid trace.

The BuildBot instance atalso runs tests with this argument, see `mtr_opttrace` pass in kvm-fulltest and kvm-fulltest2.

Debugging

See optimizer-debugging-with-gdb/#printing-the-optimizer-trace for commands to print the trace for the current statement.

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Trace Guide

Optimizer Trace Guide

The optimizer trace uses the JSON format. It is basically a structured log file showing what actions were taken by the query optimizer.

A Basic Example

Let's take a simple query:

```
MariaDB> explain select * from t1 where a<10;
+------+-------------+-------+-------+---------------+------+---------+------+------+--------------------
---+
| id   | select_type | table | type  | possible_keys | key  | key_len | ref  | rows | Extra
|
+------+-------------+-------+-------+---------------+------+---------+------+------+--------------------
---+
|    1 | SIMPLE      | t1    | range | a             | a    | 5       | NULL | 10   | Using index
condition |
+------+-------------+-------+-------+---------------+------+---------+------+------+--------------------
---+
```

One can see the full trace here. Taking only the component names, one gets:

```
MariaDB> select * from information_schema.optimizer_trace limit 1\G
*************************** 1. row ***************************
                            QUERY: select * from t1 where a<10
                            TRACE:
{
  "steps": [
    {
      "join_preparation": { ... }
    },
    {
      "join_optimization": {
        "select_id": 1,
        "steps": [
          { "condition_processing": { ... } },
          { "table_dependencies": [ ... ] },
          { "ref_optimizer_key_uses": [ ... ] },
          { "rows_estimation": [
              {
                "range_analysis": {
                   "analyzing_range_alternatives" : { ... },
                  "chosen_range_access_summary": { ... },
                },
                "selectivity_for_indexes" : { ... },
                "selectivity_for_columns" : { ... }
              }
            ]
          },
          { "considered_execution_plans": [ ... ] },
          { "attaching_conditions_to_tables": { ... } }
        ]
      }
    },
    {
      "join_execution": { ... }
    }
  ]
}
```

Trace Structure

For each SELECT, there are two "Steps":

- `join_preparation`
- `join_optimization`

Join preparation shows early query rewrites. `join_optmization` is where most of the query optimizations are done. They are:

- `condition_processing` - basic rewrites in WHERE/ON conditions.
- `ref_optimizer_key_uses` - Construction of possible ways to do ref and eq_ref accesses.
- `rows_estimation` - Consideration of range and index_merge accesses.
- `considered_execution_plans` - Join optimization itself, that is, choice of the join order.
- `attaching_conditions_to_tables` - Once the join order is fixed, parts of the WHERE clause are “attached” to tables to filter out rows as early as possible.

The above steps are for just one SELECT. If the query has subqueries, each SELECT will have these steps, and there will be extra steps/rewrites to handle the subquery construct itself.

Extracting Trace Components

If you are interested in some particular part of the trace, MariaDB has two functions that come in handy:

- JSON_EXTRACT extracts a part of JSON document
- JSON_DETAILED presents it in a user-readable way.

For example, the contents of the `analyzing_range_alternatives` node can be extracted like so:

```
MariaDB> select JSON_DETAILED(JSON_EXTRACT(trace, '$**.analyzing_range_alternatives'))
   ->   from INFORMATION_SCHEMA.OPTIMIZER_TRACE\G
*************************** 1. row ***************************
JSON_DETAILED(JSON_EXTRACT(trace, '$**.analyzing_range_alternatives')): [
    {
        "range_scan_alternatives":
        [
            {
                "index": "a_b_c",
                "ranges":
                [
                    "(1) <= (a,b) < (4,50)"
                ],
                "rowid_ordered": false,
                "using_mrr": false,
                "index_only": false,
                "rows": 4,
                "cost": 6.2509,
                "chosen": true
            }
        ],
        "analyzing_roworder_intersect":
        {
            "cause": "too few roworder scans"
        },
        "analyzing_index_merge_union": []
    }
]
```

Examples of Various Information in the Trace

Basic Rewrites

A lot of applications construct database query text on the fly, which sometimes means that the query has constructs that are repetitive or redundant. In most cases, the optimizer will be able to remove them. One can check the trace to be sure:

```
explain select * from t1 where not (col1 >= 3);
```

Optimizer trace will show:

```
"steps": [
  {
    "join_preparation": {
      "select_id": 1,
      "steps": [
        {
          "expanded_query": "select t1.a AS a,t1.b AS b,t1.col1 AS col1 from t1 where t1.col1 < 3"
        }
```

Here, one can see that `NOT` was removed.

Similarly, one can also see that `IN(...)` with one element is the same as equality:

```
explain select * from t1 where col1  in (1);
```

will show

```
"join_preparation": {
    "select_id": 1,
    "steps": [
      {
        "expanded_query": "select t1.a AS a,t1.b AS b,t1.col1 AS col1 from t1 where t1.col1 = 1"
```

On the other hand, converting an UTF-8 column to UTF-8 is not removed:

```
explain select * from t1 where convert(utf8_col using utf8) = 'hello';
```

will show

```
"join_preparation": {
    "select_id": 1,
    "steps": [
      {
        "expanded_query": "select t1.a AS a,t1.b AS b,t1.col1 AS col1,t1.utf8_col AS utf8_col from t1
where convert(t1.utf8_col using utf8) = 'hello'"
          }
```

so redundant `CONVERT` calls should be used with caution.

VIEW Processing

MariaDB has two algorithms to handle VIEWs: merging and materialization. If you run a query that uses a VIEW, the trace will have either

```
"view": {
              "table": "view1",
              "select_id": 2,
              "algorithm": "merged"
            }
```

or

```
{
            "view": {
              "table": "view2",
              "select_id": 2,
              "algorithm": "materialized"
            }
          },
```

depending on which algorithm was used.

Range Optimizer - What Ranges Will Be Scanned

The MariaDB optimizer has a complex part called the Range Optimizer. This is a module that examines WHERE (and ON) clauses and constructs index ranges that need to be scanned to answer the query. The rules for constructing the ranges are quite complex.

An example: Consider a table

```
CREATE TABLE some_events (
  start_date DATE,
  end_date DATE,
  ...
  key (start_date, end_date)
);
```

and a query:

```
explain select * from some_events where start_date >= '2019-09-10' and end_date <= '2019-09-14';
+------+-------------+-------------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table       | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+-------------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | some_events | ALL  | start_date    | NULL | NULL    | NULL | 1000 | Using where |
+------+-------------+-------------+------+---------------+------+---------+------+------+-------------+
```

One might think that the optimizer would be able to use the restrictions on both *start_date* and *end_date* to construct a narrow range to be scanned. But this is not so, one of the restrictions creates a left-endpoint range and the other one creates a right-endpoint range, hence they cannot be combined.

```
select
   JSON_DETAILED(JSON_EXTRACT(trace, '$**.analyzing_range_alternatives')) as trace
from information_schema.optimizer_trace\G
*************************** 1. row ***************************
trace: [
    {
        "range_scan_alternatives":
        [
            {
                "index": "start_date",
                "ranges":
                [
                    "(2019-09-10,NULL) < (start_date,end_date)"
                ],
...
```

the potential range only uses one of the bounds.

Ref Access Options

Index-based Nested-loops joins are called “ref access” in the MariaDB optimizer.

The optimizer analyzes the WHERE/ON conditions and collects all equality conditions that can be used by ref access using some index.

The list of conditions can be found in the `ref_optimizer_key_uses` node.
(TODO example)

Join Optimization

The join optimizer's node is named `considered_execution_plans`.

The optimizer constructs the join orders in a left-to-right fashion. That is, if the query is a join of three tables:

```
SELECT * FROM t1, t2, t3 WHERE ...
```

then the optimizer will

- Pick the first table (say, it is t1),
- consider adding another table (say, t2), and construct a prefix "t1, t2"
- consider adding the third table (t3), and constructing a prefix "t1, t2, t3", which is a complete join plan
  Other join orders will be considered as well.

The basic operation here is: "given a join prefix of tables A,B,C …, try adding table X to it".
In JSON, it looks like this:

```
{
        "plan_prefix": ["t1", "t2"],
        "table": "t3",
        "best_access_path": {
          "considered_access_paths": [
            {
              ...
            }
          ]
        }
      }
```

(search for `plan_prefix` followed by `table`).

If you are interested in how the join order of #t1,t2,t3

was constructed (or not constructed), you need to search for these patterns:

- `"plan_prefix":[], "table":"t1"`
- `"plan_prefix":["t1"], "table":"t2"`
- `"plan_prefix":["t1", "t2"], "table":"t3"`

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Trace Overview

Usage

This feature produces a trace as a JSON document for any SELECT/UPDATE/DELETE containing information about decisions taken by the optimizer during the optimization phase (choice of table access method, various costs, transformations, etc). This feature helps to explain why some decisions were taken by the optimizer and why some were rejected.

Associated System Variables

- optimizer_trace='enabled=on/off'
  - Default value is off
- optimizer_trace_max_mem_size= value

    - Default value: 1048576
- optimizer_record_context{=1|0}
    - Default value: OFF. From MariaDB 12.1.

INFORMATION_SCHEMA.OPTIMIZER_TRACE

Each connection stores a trace from the last executed statement. One can view the trace by reading the Information Schema OPTIMIZER_TRACE table.

Structure of the optimizer trace table:

```
SHOW CREATE TABLE INFORMATION_SCHEMA.OPTIMIZER_TRACE \G
*************************** 1. row ***************************
       Table: OPTIMIZER_TRACE
Create Table: CREATE TEMPORARY TABLE `OPTIMIZER_TRACE` (
  `QUERY` longtext NOT NULL DEFAULT '',
  `TRACE` longtext NOT NULL DEFAULT '',
  `MISSING_BYTES_BEYOND_MAX_MEM_SIZE` int(20) NOT NULL DEFAULT 0,
  `INSUFFICIENT_PRIVILEGES` tinyint(1) NOT NULL DEFAULT 0
) ENGINE=Aria DEFAULT CHARSET=utf8 PAGE_CHECKSUM=0
```

Optimizer Trace Contents

See Optimizer Trace Guide for an overview of what one can find in the trace.

Traceable Queries

These include SELECT, UPDATE, DELETE as well as their multi-table variants and all of the preceding prefixed by EXPLAIN and ANALYZE.

Enabling Optimizer Trace

To enable optimizer trace run:

```
SET optimizer_trace='enabled=on';
```

Memory Usage

Each trace is stored as a string. It is extended (with realloc()) as the optimization progresses and appends data to it. The optimizer_trace_max_mem_size variable sets a limit on the total amount of memory used by the current trace.

If this limit is reached, the current trace isn't extended (so it will be incomplete), and the MISSING_BYTES_BEYOND_MAX_MEM_SIZE column will show the number of bytes missing from this trace.

Privilege Checking

In complex scenarios where the query uses SQL SECURITY DEFINER views or stored routines, it may be that a user is denied from seeing the trace of its query because it lacks some extra privileges on those objects. In that case, the trace will be shown as empty and the INSUFFICIENT_PRIVILEGES column will show "1".

Limitations

Currently, only one trace is stored. It is not possible to trace the sub-statements of a stored routine; only the statement at the top level is traced.

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Trace Resources

- Optimizer Trace Walkthrough talk at MariaDB Fest 2020:
- A tool for processing Optimizer Trace: opttrace . Doesn't work with MariaDB at the moment but everyone is welcome to make it work.

This page is licensed: CC BY-SA / Gnu FDL

System and Status Variables

System Variables

See Big Query Settings.

Settings and features aimed at big queries, several of which are disabled by default.

See Handling Too Many Connections.

How to handle and avoid the 'too many connections' error on busy servers.

See MariaDB Optimization for MySQL Users.

MariaDB optimization options and defaults that differ from MySQL, for users migrating from MySQL.

See Optimizing key_buffer_size.

Tuning key_buffer_size for servers with mostly MyISAM tables.

See Optimizing table_open_cache.

Tuning table_open_cache to balance performance against open-file usage.

See OQGRAPH System and Status Variables.

System and status variables for the OQGRAPH storage engine.

See Sample my.cnf Files.

Sample my.cnf configuration files for different memory sizes and storage engines.

See Segmented Key Cache.

Segmented key caches, which split the MyISAM key cache into structures to reduce contention.

See Semisynchronous Replication Plugin Status Variables.

Status variables for the semisynchronous replication plugin.

See InnoDB Server Status Variables.

InnoDB server status variables, viewable with SHOW STATUS.

See Setting Innodb Buffer Pool Size Dynamically.

How to resize the InnoDB buffer pool dynamically, in chunks.

See Sphinx Status Variables.

Status variables for the Sphinx storage engine.

See Spider Status Variables.

Status variables for the Spider storage engine.

See SQL Error Log System Variables and Options.

System variables and options for the SQL Error Log plugin.

See SSL/TLS Status Variables.

Status variables related to TLS/SSL-encrypted connections.

See System and Status Variables Added By Major Release.

Discover system and status variables added by major MariaDB Server releases. This section helps you track new configuration options and monitoring metrics introduced in different versions.

This page is licensed: CC BY-SA / Gnu FDL

Big Query Settings

MariaDB 5.3 and beyond have a number of features that are targeted at big queries and so are disabled by default.

This page describes recommended settings for IO-bound queries that shovel through lots of records.

First, turn on Batched Key Access:

```
# Turn on disk-ordered reads
optimizer_switch='mrr=on'
optimizer_switch='mrr_cost_based=off'

# Turn on Batched Key Access (BKA)
join_cache_level = 6
```

Give BKA buffer space to operate on.
Ideally, it should have enough space to fit all the data examined by the query.

```
# Size limit for the whole join
join_buffer_space_limit = 300M

# Limit for each individual table
join_buffer_size = 100M
```

Turn on index_merge/sort-intersection:

```
optimizer_switch='index_merge_sort_intersection=on'
```

If your queries examine big fraction of the tables (somewhere more than ~ 30%), turn on hash join:

```
# Turn on both Hash Join and Batched Key Access
join_cache_level = 8
```

This page is licensed: CC BY-SA / Gnu FDL

Handling Too Many Connections

Systems that get too busy can return the `too_many_connections` error.

When the number of threads_connected exceeds the max_connections server variable, it's time to make a change. Viewing the threads_connected status variable shows only the current number of connections, but it's more useful to see what the value has peaked at, and this is shown by the max_used_connections status variable.

This error may be a symptom of slow queries and other bottlenecks, but if the system is running smoothly this can be addressed by increasing the value of max_connections.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Optimization for MySQL Users

MariaDB contains many new options and optimizations which, for compatibility or other reasons, are not enabled in the default install.
Enabling them helps you gain extra performance from the same hardware when upgrading from MySQL to MariaDB. This article contains information on options and settings which you should enable, or at least look in to, when making the switch.

```
aria-pagecache-buffer-size=##
```

If you are using a log of on-disk temporary tables, increase the above to as much as you can afford. See Aria Storage Engine for more details.

```
key-cache-segments=8
```

If you use/have a lot of MyISAM files, increase the above to 4 or 8. See Segmented Key Cache and Segmented Key Cache Performance for more information.

```
thread-handling=pool-of-threads
```

Threadpool is a great way to increase performance in situations where queries are relatively short and the load is CPU bound (e.g. OLTP workloads). To enable it, add the above to your my.cnf file. See Threadpool in 5.5 for more information.

This page is licensed: CC BY-SA / Gnu FDL

Optimizing key_buffer_size

key_buffer_size is a MyISAM variable which determines the size of the index buffers held in memory, which affects the speed of index reads. Note that Aria tables by default make use of an alternative setting, aria-pagecache-buffer-size.

A good rule of thumb for servers consisting particularly of MyISAM tables is for about 25% or more of the available server memory to be dedicated to the key buffer.

A good way to determine whether to adjust the value is to compare the key_read_requests value, which is the total value of requests to read an index, and the key_reads values, the total number of requests that had to be read from disk.

The ratio of key_reads to key_read_requests should be as low as possible, 1:100 is the highest acceptable, 1:1000 is better, and 1:10 is terrible.

The effective maximum size might be lower than what is set, depending on the server's available physical RAM and the per-process limit determined by the operating system.

If you don't make use of MyISAM tables at all, you can set this to a very low value, such as 64K.

This page is licensed: CC BY-SA / Gnu FDL

Optimizing table_open_cache

table_open_cache can be a useful variable to adjust to improve performance.

Each concurrent session accessing the same table does so independently. This improves performance, although it comes at a cost of extra memory usage.

table_open_cache indicates the maximum number of tables the server can keep open in any one table cache instance. Ideally, you'd like this set so as to re-open a table as infrequently as possible.

However, note that this is not a hard limit. When the server needs to open a table, it evicts the least recently used closed table from the cache, and adds the new table. If all tables are used, the server adds the new table and does not evict any table. As soon as a table is not used anymore, it will be evicted from the list even if no table needs to be open, until the number of open tables will be equal to *table_open_cache*

table_open_cache has defaulted to 2000 since MariaDB 10.1.7. Before that, the default was 400.

You can view the current setting in the my.cnf file, or by running:

```
SELECT @@table_open_cache;
+--------------------+
| @@table_open_cache |
+--------------------+
|                400 |
+--------------------+
```

To evaluate whether you could do with a higher table_open_cache, look at the number of opened tables, in conjunction with the server uptime (Opened_tables and Uptime status variables):

```
SHOW global status LIKE 'opened_tables';
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| Opened_tables | 354858 |
+---------------+--------+
```

If the number of opened tables is increasing rapidly, you should look at increasing the table_open_cache value. Try to find a value that sees a slow, or possibly even no, increase in the number of opened tables.

Make sure that your operating system can cope with the number of open file descriptors required by the table_open_cache setting. If table_open_cache is set too high, MariaDB may start to refuse connections as the operating system runs out of file descriptors. Also note that the MyISAM (and Aria?) storage engines need two file descriptors per open table.

It's possible that the open_table_cache can even be reduced.

If your number of open_tables has not yet reached the table_open_cache_size, and the server has been up a while, you can look at decreasing the value.

```
SHOW global status LIKE 'open_tables';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Open_tables   | 354   |
+---------------+-------+
```

The open table cache can be emptied with FLUSH TABLES or with the `flush-tables` or `refresh` mariadb-admin commands.

Automatic Creation of New Table Open Cache Instances

MariaDB Server can create multiple instances of the table open cache. It initially creates just a single instance. However, whenever it detects contention on the existing instances, it will automatically create a new instance. When the number of instances has been increased due to contention, it does not decrease again.

When MariaDB Server creates a new instance, it prints a message like the following to the error log:

```
[Note] Detected table cache mutex contention at instance 1: 25% waits. Additional
  table cache instance activated. Number of instances after activation: 2.
```

The maximum number of instances is defined by the table_open_cache_instances system variable. The default value of the table_open_cache_instances system variable is `8`, which is expected to handle up to 100 CPU cores. If your system is larger than this, then you may benefit from increasing the value of this system variable.

Depending on the ratio of actual available file handles, and table_open_cache size, the max. instance count may be auto adjusted to a lower value on server startup.

The implementation and behavior of this feature is different than the same feature in MySQL 5.6.

This page is licensed: CC BY-SA / Gnu FDL

OQGRAPH System and Status Variables

This page documents system and status variables related to the OQGRAPH storage engine. See Server Status Variables and Server System Variables for complete list of all system and status variables.

System Variables

`oqgraph_allow_create_integer_latch`

- Description: Created when the OQGRAPH storage engine is installed, if set to `1` (`0` is default), permits the `latch` field to be an integer (see OQGRAPH Overview). This deprecated feature was removed in MariaDB 11.5.
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`
- Removed: MariaDB 11.5.

Status Variables

`Oqgraph_boost_version`

- Description: OQGRAPH boost version.
- Scope: Global, Session
- Data Type: `string`

`Oqgraph_compat_mode`

- Description: Whether or not legacy tables with integer latches are supported.
- Scope: Global, Session
- Data Type: `string`

`Oqgraph_verbose_debug`

- Description: Whether or not verbose debugging is enabled. If it is, performance may be adversely impacted
- Scope: Global, Session
- Data Type: `string`

This page is licensed: CC BY-SA / Gnu FDL

Sample my.cnf Files

Place holder for sample my.cnf files, customized for different memory size and storage engines. In addition, we'd like to hear from you what works for you, so the knowledge can be crowd-sourced and shared.

This page is licensed: CC BY-SA / Gnu FDL

Segmented Key Cache

About Segmented Key Cache

A segmented key cache is a collection of structures for regular MyISAM key caches called key cache segments. Segmented key caches mitigate one of the major problems of the simple key cache: thread contention for key cache lock (mutex). With regular key caches, every call of a key cache interface function must acquire this lock. So threads compete for this lock even in the case when they have acquired shared locks for the file and the pages they want to read from are in the key cache buffers.

When working with a segmented key cache any key cache interface function that needs only one page has to acquire the key cache lock only for the segment the page is assigned to. This makes the chances for threads not having to compete for the same key cache lock better.

Any page from a file can be placed into a buffer of only one segment. The number of the segment is calculated from the file number and the position of the page in the file, and it's always the same for the page. Pages are evenly distributed among segments.

The idea and the original code of the segmented key cache was provided by Fredrik Nylander from Stardoll.com. The code was extensively reworked, improved, and eventually merged into MariaDB by Igor Babaev from Monty Program (now MariaDB Corporation).

You can find some benchmark results comparing various settings on the Segmented Key Cache Performance page.

Segmented Key Cache Syntax

New global variable: key_cache_segments. This variable sets the number of segments in a key cache. Valid values for this variable are whole numbers between `0` and `64`. If the number of segments is set to a number greater than `64` the number of segments will be truncated to 64 and a warning will be issued. A value of `0` means the key cache is a regular (i.e. non-segmented) key cache. This is the default. If `key_cache_segments` is `1` (or higher) then the new key cache segmentation code is used. In practice there is no practical use of a single-segment segmented key cache except for testing purposes, and setting `key_cache_segments = 1` should be slower than any other option and should not be used in production.

Other global variables used when working with regular key caches also apply to segmented key caches: key_buffer_size,key_cache_age_threshold, key_cache_block_size, and key_cache_division_limit.

Segmented Key Cache Statistics

Statistics about the key cache can be found by looking at the KEY_CACHES table in the INFORMATION_SCHEMA database. Columns in this table are:

Column Name	Description
KEY_CACHE_NAME	The name of the key cache
SEGMENTS	total number of segments (set to NULL for regular key caches)
SEGMENT_NUMBER	segment number (set to NULL for any regular key caches and for rows containing aggregation statistics for segmented key caches)
FULL_SIZE	memory for cache buffers/auxiliary structures
BLOCK_SIZE	size of the blocks
USED_BLOCKS	number of currently used blocks
UNUSED_BLOCKS	number of currently unused blocks
DIRTY_BLOCKS	number of currently dirty blocks
READ_REQUESTS	number of read requests

Column Name	Description
READS	number of actual reads from files into buffers
WRITE_REQUESTS	number of write requests
WRITES	number of actual writes from buffers into files

See Also

- Segmented Key Cache Performance

This page is licensed: CC BY-SA / Gnu FDL

Semisynchronous Replication Plugin Status Variables

This page documents status variables related to the Semisynchronous Replication Plugin (which has been merged into the main server from MariaDB 10.3.3). See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

See also the Full list of MariaDB options, system and status variables.

`Rpl_semi_sync_master_clients`

- Description: Number of semisynchronous slaves.
- Data Type: `numeric`

`Rpl_semi_sync_master_get_ack`

- Description: Number of acknowledgements received by the master from slaves.
- Data Type: `numeric`

`Rpl_semi_sync_master_net_avg_wait_time`

- Description: Average time the master waited for a slave to reply, in microseconds.
- Data Type: `numeric`

`Rpl_semi_sync_master_net_wait_time`

- Description: Total time the master waited for slave replies, in microseconds.
- Data Type: `numeric`

`Rpl_semi_sync_master_net_waits`

- Description: Total number of times the master waited for slave replies.
- Data Type: `numeric`

`Rpl_semi_sync_master_no_times`

- Description: Number of times the master turned off semisynchronous replication. The global value can be flushed by FLUSH STATUS.
- Data Type: `numeric`

`Rpl_semi_sync_master_no_tx`

- Description: Number of commits that have not been successfully acknowledged by a slave. The global value can be flushed by FLUSH STATUS.
- Data Type: `numeric`

`Rpl_semi_sync_master_request_ack`

- Description: Number of acknowledgement requests sent to slaves.
- Data Type: `numeric`

`Rpl_semi_sync_master_status`

- Description: Whether or not semisynchronous replication is currently operating on the master. The value will be `ON` if both the plugin has been enabled and a commit acknowledgment has occurred. It will be `OFF` if either the plugin has not been enabled, or the master is replicating asynchronously due to a commit acknowledgment timeout.
- Data Type: `boolean`

`Rpl_semi_sync_master_timefunc_failures`

- Description: Number of times the master failed when calling a time function, such as gettimeofday(). The global value can be flushed by FLUSH STATUS.
- Data Type: `numeric`

`Rpl_semi_sync_master_tx_avg_wait_time`

- Description: Average time the master waited for each transaction, in microseconds.
- Data Type: `numeric`

`Rpl_semi_sync_master_tx_wait_time`

- Description: Total time the master waited for transactions, in microseconds.
- Data Type: `numeric`

`Rpl_semi_sync_master_tx_waits`

- Description: Total number of times the master waited for transactions.
- Data Type: `numeric`

`Rpl_semi_sync_master_wait_pos_backtraverse`

- Description: Total number of times the master waited for an event that had binary coordinates lower than previous events waited for. Occurs if the order in which transactions start waiting for a reply is different from the order in which their binary log events were written. The global value can be flushed by FLUSH STATUS.
- Data Type: `numeric`

`Rpl_semi_sync_master_wait_sessions`

- Description: Number of sessions that are currently waiting for slave replies.
- Data Type: `numeric`

`Rpl_semi_sync_master_yes_tx`

- Description: Number of commits that have been successfully acknowledged by a slave. The global value can be flushed by FLUSH STATUS.
- Data Type: `numeric`

`Rpl_semi_sync_slave_status`

- Description: Whether or not semisynchronous replication is currently operating on the slave. `ON` if both semi-sync has been enabled for the replica (i.e. by setting the variable rpl_semi_sync_slave_enabled to `TRUE`) and

the slave I/O thread is running.

- Data Type: `boolean`

`Rpl_semi_sync_slave_send_ack`

- Description: Number of acknowledgements the slave sent to the master.
- Data Type: `numeric`

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Server Status Variables

See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

Much of the InnoDB information here can also be seen with a SHOW ENGINE INNODB STATUS statement.

See also the Full list of MariaDB options, system and status variables.

`Innodb_adaptive_hash_cells`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5, this status variable is present in XtraDB.
  - In MariaDB 10.1 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.0.0

`Innodb_adaptive_hash_hash_searches`

- Description: Hash searches as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - Before the variable was introduced in MariaDB 10.5.0, use the `adaptive_hash_searches` counter in the information_schema.INNODB_METRICS table instead.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.5.0

`Innodb_adaptive_hash_heap_buffers`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5, this status variable is present in XtraDB.
  - In MariaDB 10.1 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.0.0

`Innodb_adaptive_hash_non_hash_searches`

- Description: Non-hash searches as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output. From MariaDB 10.6.2, not updated if innodb_adaptive_hash_index is not enabled (the default).
  - In MariaDB 10.1, MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present. Use the `adaptive_hash_searches_btree` counter in the information_schema.INNODB_METRICS table instead.
  - From MariaDB 10.5, this status variable is present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5, MariaDB 10.5.0

`Innodb_async_reads_pending`

- Description: Number of async read I/O operations currently in progress (from SUBMITTED to COMPLETED). See InnoDB Asynchronous I/O.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_reads_queue_size`

- Description: Current size of the async I/O read queue. See InnoDB Asynchronous I/O Queuing Mechanism.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_reads_tasks_running`

- Description: Number of async read I/O operations currently in the EXECUTING_COMPLETION_TASK state. See InnoDB Asynchronous I/O.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_reads_total_count`

- Description: Total number of async read completion tasks that have finished execution. See InnoDB Asynchronous I/O.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_reads_total_enqueues`

- Description: Total number of async read operations that were queued. Includes those still waiting and making up innodb_async_reads_queue_size. See InnoDB Asynchronous I/O.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_reads_wait_slot_sec`

- Description: Total wait time for a free I/O slot (see Waiting for IO Slots). See InnoDB Asynchronous I/O Waiting for IO Slots.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_writes_pending`

- Description: Number of async write I/O operations currently in progress (from SUBMITTED to COMPLETED). See InnoDB Asynchronous I/O.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_writes_queue_size`

- Description: Current size of the async I/O write queue. See InnoDB Asynchronous I/O Queuing Mechanism.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_writes_tasks_running`

- Description: Number of async write I/O operations currently in the EXECUTING_COMPLETION_TASK state. See InnoDB Asynchronous I/O.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_writes_total_count`

- Description: Total number of async write completion tasks that have finished execution. See InnoDB Asynchronous I/O.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_writes_total_enqueues`

- Description: Total number of async write operations that were queued. Includes those still waiting and making up innodb_async_writes_queue_size. See InnoDB Asynchronous I/O.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_async_writes_wait_slot_sec`

- Description: See InnoDB Asynchronous I/O.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.5

`Innodb_available_undo_logs`

- Description: Total number available InnoDB undo logs. Differs from the innodb_undo_logs system variable, which specifies the number of active undo logs.
- Scope: Global
- Data Type: `numeric`

`Innodb_background_log_sync`

- Description: As shown in the BACKGROUND THREAD section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5, this status variable is present in XtraDB.
  - In MariaDB 10.1, MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB only), MariaDB 10.5.0

`Innodb_buffer_pool_bytes_data`

- Description: Number of bytes contained in the InnoDB buffer pool, both dirty (modified) and clean (unmodified). See also Innodb_buffer_pool_pages_data, which can contain pages of different sizes in the case of compression.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_bytes_dirty`

- Description: Number of dirty (modified) bytes contained in the InnoDB buffer pool. See also Innodb_buffer_pool_pages_dirty, which can contain pages of different sizes in the case of compression.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_dump_status`

- Description: A text description of the progress or final status of the last Innodb buffer pool dump.
- Scope: Global
- Data Type: `string`
- Introduced: MariaDB 10.0.0

`Innodb_buffer_pool_load_incomplete`

- Description: Whether or not the loaded buffer pool is incomplete, for example after a shutdown or abort during innodb buffer pool load from file caused an incomplete save.
- Scope: Global
- Data Type: `boolean`
- Introduced: MariaDB 10.3.5

`Innodb_buffer_pool_load_status`

- Description: A text description of the progress or final status of the last Innodb buffer pool load.
- Scope: Global
- Data Type: `string`

- Introduced: MariaDB 10.0.0

`Innodb_buffer_pool_pages_data`

- Description: Number of InnoDB buffer pool pages which contain data, both dirty (modified) and clean (unmodified). See also Innodb_buffer_pool_bytes_data.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_pages_dirty`

- Description: Number of InnoDB buffer pool pages which contain dirty (modified) data. See also innodb_buffer_pool_bytes_dirty.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_pages_flushed`

- Description: Number of InnoDB buffer pool pages which have been flushed.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_pages_LRU_flushed`

- Description: Flush list as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_buffer_pool_pages_LRU_freed`

- Description: Monitor the number of pages that were freed by a buffer pool LRU eviction scan, without flushing.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.6.0

`Innodb_buffer_pool_pages_free`

- Description: Number of free InnoDB buffer pool pages.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_pages_made_not_young`

- Description: Pages not young as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.

- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_buffer_pool_pages_made_young`

- Description: Pages made young as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_buffer_pool_pages_misc`

- Description: Number of InnoDB buffer pool pages set aside for internal use.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_pages_old`

- Description: Old database page, as shown in the BUFFER POOL AND MEMORY section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present for XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_buffer_pool_pages_total`

- Description: Total number of InnoDB buffer pool pages.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_read_ahead`

- Description: Number of pages read into the InnoDB buffer pool by the read-ahead background thread.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_read_ahead_evicted`

- Description: Number of pages read into the InnoDB buffer pool by the read-ahead background thread that were evicted without having been accessed by queries.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_read_ahead_rnd`

- Description: Number of random read-aheads.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_read_requests`

- Description: Number of requests to read from the InnoDB buffer pool .
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_reads`

- Description: Number of reads that could not be satisfied by the InnoDB buffer pool and had to be read from disk.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_resize_status`

- Description: Progress of the dynamic InnoDB buffer pool resizing operation. See Setting Innodb Buffer Pool Size Dynamically.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.2.2

`Innodb_buffer_pool_wait_free`

- Description: Number of times InnoDB waited for a free page before reading or creating a page. Normally, writes to the InnoDB buffer pool happen in the background. When no clean pages are available, dirty pages are flushed first in order to free some up. This counts the numbers of wait for this operation to finish. If this value is not small, look at increasing innodb_buffer_pool_size.
- Scope: Global
- Data Type: `numeric`

`Innodb_buffer_pool_write_requests`

- Description: Number of requests to write to the InnoDB buffer pool .
- Scope: Global
- Data Type: `numeric`

`Innodb_buffered_aio_submitted`

- Description:
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.5.0

`Innodb_bulk_operations`

- Description: Number of bulk insert operations to InnoDB tables.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.6.19, MariaDB 10.11.9, MariaDB 11.1.6, MariaDB 11.2.5, MariaDB 11.4.3

`Innodb_checkpoint_age`

- Description: The amount of data written to the InnoDB redo log since the last checkpoint, as shown in the LOG section of the SHOW ENGINE INNODB STATUS output. (This is equivalent to subtracting "Last checkpoint at" from "Log sequence number", cf the RedoLog details). Despite the name ("age"), the value is not a duration, but really an amount of data (in bytes) !
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_checkpoint_max_age`

- Description: Max checkpoint age, as shown in the LOG section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_checkpoint_target_age`

- Description: Checkpoint age target, as shown in the LOG section of the SHOW ENGINE INNODB STATUS output. XtraDB only. Removed in MariaDB 10.0 and replaced with MySQL 5.6's flushing implementation.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.0

`Innodb_current_row_locks`

- Description: Number of current row locks on InnoDB tables as shown in the TRANSACTIONS section of the SHOW ENGINE INNODB STATUS output. Renamed from InnoDB_row_lock_numbers in XtraDB 5.5.8-20.1.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5

`Innodb_data_fsyncs`

- Description: Number of InnoDB fsync (sync-to-disk) calls. fsync call frequency can be influenced by the innodb_flush_method configuration option.
- Scope: Global
- Data Type: `numeric`

`Innodb_data_pending_fsyncs`

- Description: Number of pending InnoDB fsync (sync-to-disk) calls. fsync call frequency can be influenced by the innodb_flush_method configuration option.

- Scope: Global
- Data Type: `numeric`

`Innodb_data_pending_reads`

- Description: Number of pending InnoDB reads.
- Scope: Global
- Data Type: `numeric`

`Innodb_data_pending_writes`

- Description: Number of pending InnoDB writes.
- Scope: Global
- Data Type: `numeric`

`Innodb_data_read`

- Description: Number of InnoDB bytes read since server startup (not to be confused with Innodb_data_reads).
- Scope: Global
- Data Type: `numeric`

`Innodb_data_reads`

- Description: Number of InnoDB read operations (not to be confused with Innodb_data_read).
- Scope: Global
- Data Type: `numeric`

`Innodb_data_writes`

- Description: Number of InnoDB write operations.
- Scope: Global
- Data Type: `numeric`

`Innodb_data_written`

- Description: Number of InnoDB bytes written since server startup. From MariaDB 10.5.1, no longer includes writes to the redo log file `ib_logfile0`, which continue to be counted by Innodb_os_log_written. An error in counting was introduced in MariaDB 10.5.7 until MariaDB 10.5.20, MariaDB 10.6.13, MariaDB 10.8.8, MariaDB 10.9.6, MariaDB 10.10.4 and MariaDB 10.11.3 (MDEV-31124) in which writes via the doublewrite buffer started to be counted incorrectly, without multiplying them by innodb_page_size. A workaround for the error could be the following formulae: real_data_written = Innodb_data_written + (innodb_page_size - 1) * Innodb_dblwr_pages_writteninnodb_written = real_data_written + Innodb_os_log_written
- Scope: Global
- Data Type: `numeric`

`Innodb_dblwr_pages_written`

- Description: Number of pages written to the InnoDB doublewrite buffer.
- Scope: Global
- Data Type: `numeric`

`Innodb_dblwr_writes`

- Description: Number of writes to the InnoDB doublewrite buffer.

- Scope: Global
- Data Type: `numeric`

`Innodb_deadlocks`

- Description: Total number of InnoDB deadlocks. Deadlocks occur when at least two transactions are waiting for the other to finish, creating a circular dependency. InnoDB usually detects these quickly, returning an error.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_defragment_compression_failures`

- Description: Number of defragment re-compression failures. See Defragmenting InnoDB Tablespaces.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.1

`Innodb_defragment_count`

- Description: Number of defragment operations. See Defragmenting InnoDB Tablespaces.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.1

`Innodb_defragment_failures`

- Description: Number of defragment failures. See Defragmenting InnoDB Tablespaces.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.1

`Innodb_dict_tables`

- Description: Number of entries in the XtraDB data dictionary cache. This Percona XtraDB variable was removed in MariaDB 10/XtraDB 5.6 as it was replaced with MySQL 5.6's table_definition_cache implementation.
- Scope: Global
- Data Type: `numeric`
- Introduced: XtraDB 5.0.77-b13
- Removed: MariaDB 10.0

`Innodb_encryption_n_merge_blocks_decrypted`

- Description:
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.28, MariaDB 10.2.9, MariaDB 10.3.2

`Innodb_encryption_n_merge_blocks_encrypted`

- Description:
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.28, MariaDB 10.2.9, MariaDB 10.3.2

`Innodb_encryption_n_rowlog_blocks_decrypted`

- Description:
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.28, MariaDB 10.2.9, MariaDB 10.3.2

`Innodb_encryption_n_rowlog_blocks_encrypted`

- Description:
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.28, MariaDB 10.2.9, MariaDB 10.3.2

`Innodb_encryption_n_temp_blocks_decrypted`

- Description:
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.2.26, MariaDB 10.3.17, MariaDB 10.4.7

`Innodb_encryption_n_temp_blocks_encrypted`

- Description:
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.2.26, MariaDB 10.3.17, MariaDB 10.4.7

`Innodb_encryption_num_key_requests`

- Description: Was not present in MariaDB 10.5.2.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.2.4

`Innodb_encryption_rotation_estimated_iops`

- Description: See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.1.3

`Innodb_encryption_rotation_pages_flushed`

- Description: See Table and Tablespace Encryption.
- Scope: Global

- Data Type: `numeric`
- Removed: MariaDB 10.1.3

`Innodb_encryption_rotation_pages_modified`

- Description: See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.1.3

`Innodb_encryption_rotation_pages_read_from_cache`

- Description: See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.1.3

`Innodb_encryption_rotation_pages_read_from_disk`

- Description: See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.1.3

`Innodb_have_atomic_builtins`

- Description: Whether the server has been built with atomic instructions, provided by the CPU ensuring that critical low-level operations can't be interrupted. XtraDB only.
- Scope: Global
- Data Type: `boolean`

`Innodb_have_bzip2`

- Description: Whether the server has the bzip2 compression method available. See InnoDB/XtraDB Page Compression.
- Scope: Global
- Data Type: `boolean`
- Introduced: MariaDB 10.1.0

`Innodb_have_lz4`

- Description: Whether the server has the lz4 compression method available. See InnoDB/XtraDB Page Compression.
- Scope: Global
- Data Type: `boolean`
- Introduced: MariaDB 10.1.0

`Innodb_have_lzma`

- Description: Whether the server has the lzma compression method available. See InnoDB/XtraDB Page Compression.
- Scope: Global
- Data Type: `boolean`

- Introduced: MariaDB 10.1.0

`Innodb_have_lzo`

- Description: Whether the server has the lzo compression method available. See InnoDB/XtraDB Page Compression.
- Scope: Global
- Data Type: `boolean`
- Introduced: MariaDB 10.1.0

`Innodb_have_punch_hole`

- Description:
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.2.4

`Innodb_have_snappy`

- Description: Whether the server has the snappy compression method available. See InnoDB/XtraDB Page Compression.
- Scope: Global
- Data Type: `boolean`
- Introduced: MariaDB 10.1.3

`Innodb_history_list_length`

- Description: History list length as shown in the TRANSACTIONS section of the SHOW ENGINE INNODB STATUS output. XtraDB only until introduced in MariaDB 10.5.0.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_discarded_delete_marks`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_discarded_deletes`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.

- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_discarded_inserts`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_free_list`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_merged_delete_marks`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_merged_deletes`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_merged_inserts`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_merges`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_segment_size`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_ibuf_size`

- Description: As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_instant_alter_column`

- Description: See Instant ADD COLUMN for InnoDB.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.3.2

`Innodb_log_waits`

- Description: Number of times InnoDB was forced to wait for log writes to be flushed due to the log buffer being too small.
- Scope: Global
- Data Type: `numeric`

`Innodb_log_write_requests`

- Description: Number of requests to write to the InnoDB redo log.
- Scope: Global
- Data Type: `numeric`

`Innodb_log_writes`

- Description: Number of writes to the InnoDB redo log.
- Scope: Global
- Data Type: `numeric`

`Innodb_lsn_current`

- Description: Log sequence number as shown in the LOG section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_lsn_flushed`

- Description: Flushed up to log sequence number as shown in the LOG section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_lsn_last_checkpoint`

- Description: Log sequence number last checkpoint as shown in the LOG section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_lsn_archived`

- Description: The Log Sequence Number (LSN) since which a complete InnoDB log archive is available. When `innodb_log_archive` is `OFF`, the value advances to the latest checkpoint each time a log checkpoint is executed. When `innodb_log_archive` is `ON`, the value does not change — the archive is, by construction, complete from this LSN onward. The initial value at startup is taken from `innodb_log_archive_start`, or auto-detected from the available log files if that parameter is `0`.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 13.0

`Innodb_master_thread_1_second_loops`

- Description: As shown in the BACKGROUND THREAD section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5, this status variable is present in XtraDB.
  - In MariaDB 10.1 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.0

`Innodb_master_thread_10_second_loops`

- Description: As shown in the BACKGROUND THREAD section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5, this status variable is present in XtraDB.
  - In MariaDB 10.1 and later, this status variable is not present
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.0

`Innodb_master_thread_active_loops`

- Description: As shown in the BACKGROUND THREAD section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.0.9 (XtraDB-only), MariaDB 10.5.0:

`Innodb_master_thread_background_loops`

- Description: As shown in the BACKGROUND THREAD section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5, this status variable is present in XtraDB.
  - In MariaDB 10.1 and later, this status variable is not present
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.0

`Innodb_master_thread_idle_loops`

- Description: As shown in the BACKGROUND THREAD section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.0.9 (XtraDB-only), MariaDB 10.5.0:

`Innodb_master_thread_main_flush_loops`

- Description: As shown in the BACKGROUND THREAD section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5, this status variable is present in XtraDB.
  - In MariaDB 10.1 and later, this status variable is not present
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.0

`Innodb_master_thread_sleeps`

- Description: As shown in the BACKGROUND THREAD section of the SHOW ENGINE INNODB STATUS output. XtraDB only.
  - In MariaDB 5.5, this status variable is present in XtraDB.
  - In MariaDB 10.1, MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present. Use the `innodb_master_thread_sleeps` counter in the information_schema.INNODB_METRICS table instead.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.0

`Innodb_max_trx_id`

- Description: As shown in the TRANSACTIONS section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_mem_adaptive_hash`

- Description: As shown in the BUFFER POOL AND MEMORY section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.

- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_mem_dictionary`

- Description: As shown in the BUFFER POOL AND MEMORY section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2, MariaDB 10.3, and MariaDB 10.4, this status variable is not present.
  - In MariaDB 10.5, this status variable was reintroduced.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0

`Innodb_mem_total`

- Description: As shown in the BUFFER POOL AND MEMORY section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5

`Innodb_mutex_os_waits`

- Description: Mutex OS waits as shown in the SEMAPHORES section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5

`Innodb_mutex_spin_rounds`

- Description: Mutex spin rounds as shown in the SEMAPHORES section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5

`Innodb_mutex_spin_waits`

- Description: Mutex spin waits as shown in the SEMAPHORES section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.

    - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5

`Innodb_num_index_pages_written`

- Description:
- Scope:
- Data Type: `numeric`
- Introduced: MariaDB 10.1.0

`Innodb_num_non_index_pages_written`

- Description:
- Scope:
- Data Type: `numeric`
- Introduced: MariaDB 10.1.0

`Innodb_num_open_files`

- Description: Number of open files held by InnoDB. InnoDB only.
- Scope: Global
- Data Type: `numeric`

`Innodb_num_page_compressed_trim_op`

- Description: Number of trim operations performed.
- Scope: Global
- Data Type: `numeric`

`Innodb_num_page_compressed_trim_op_saved`

- Description: Number of trim operations not done because of an earlier trim.
- Scope: Global
- Data Type: `numeric`

`Innodb_num_pages_decrypted`

- Description: Number of pages page decrypted. See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`

`Innodb_num_pages_encrypted`

- Description: Number of pages page encrypted. See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`

`Innodb_num_pages_page_compressed`

- Description: Number of pages that are page compressed.
- Scope: Global

- Data Type: `numeric`

`Innodb_num_pages_page_compression_error`

- Description: Number of compression errors.
- Scope: Global
- Data Type: `numeric`

`Innodb_num_pages_page_decompressed`

- Description: Number of pages compressed with page compression that are decompressed.
- Scope: Global
- Data Type: `numeric`

`Innodb_num_pages_page_encryption_error`

- Description: Number of page encryption errors. See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.3
- Removed: MariaDB 10.1.4

`Innodb_oldest_view_low_limit_trx_id`

- Description: As shown in the TRANSACTIONS section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.2

`Innodb_onlineddl_pct_progress`

- Description: Shows the progress of in-place alter table. It might be not so accurate because in-place alter is highly dependent on disk and buffer pool status. See Monitoring progress and temporal memory usage of Online DDL in InnoDB.
- Scope: Global
- Data Type: `numeric`

`Innodb_onlineddl_rowlog_pct_used`

- Description: Shows row log buffer usage in 5-digit integer (10000 means 100.00%). See Monitoring progress and temporal memory usage of Online DDL in InnoDB.
- Scope: Global
- Data Type: `numeric`

`Innodb_onlineddl_rowlog_rows`

- Description: Number of rows stored in the row log buffer. See Monitoring progress and temporal memory usage of Online DDL in InnoDB.
- Scope: Global

- Data Type: `numeric`

`Innodb_os_log_fsyncs`

- Description: Number of InnoDB log fsync (sync-to-disk) requests.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.8

`Innodb_os_log_pending_fsyncs`

- Description: Number of pending InnoDB log fsync (sync-to-disk) requests.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.8

`Innodb_os_log_pending_writes`

- Description: Number of pending InnoDB log writes.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.8

`Innodb_os_log_written`

- Description: Number of bytes written to the InnoDB log.
- Scope: Global
- Data Type: `numeric`

`Innodb_page_compression_saved`

- Description: Number of bytes saved by page compression.
- Scope:
- Data Type:

`Innodb_page_compression_trim_sect512`

- Description: Number of TRIM operations performed for the page-compression/NVM Compression workload for the 512 byte block-size.
- Scope:
- Data Type: `numeric`
- Introduced: MariaDB 10.1.0, MariaDB 10.0.15 Fusion-io
- Removed: MariaDB 10.2.4

`Innodb_page_compression_trim_sect1024`

- Description: Number of TRIM operations performed for the page-compression/NVM Compression workload for the 1K block-size.
- Scope:
- Data Type: `numeric`
- Introduced: MariaDB 10.1.2, MariaDB 10.0.15 Fusion-io
- Removed: MariaDB 10.2.4

`Innodb_page_compression_trim_sect2048`

- Description: Number of TRIM operations performed for the page-compression/NVM Compression workload for the 2K block-size.
- Scope:
- Data Type: `numeric`
- Introduced: MariaDB 10.1.2, MariaDB 10.0.15 Fusion-io
- Removed: MariaDB 10.2.4

`Innodb_page_compression_trim_sect4096`

- Description: Number of TRIM operations performed for the page-compression/NVM Compression workload for the 4K block-size.
- Scope:
- Data Type: `numeric`
- Introduced: MariaDB 10.1.0, MariaDB 10.0.15 Fusion-io
- Removed: MariaDB 10.2.4

`Innodb_page_compression_trim_sect8192`

- Description: Number of TRIM operations performed for the page-compression/NVM Compression workload for the 8K block-size.
- Scope:
- Data Type: `numeric`
- Introduced:MariaDB 10.1.2, MariaDB 10.0.15 Fusion-io
- Removed: MariaDB 10.2.4

`Innodb_page_compression_trim_sect16384`

- Description: Number of TRIM operations performed for the page-compression/NVM Compression workload for the 16K block-size.
- Scope:
- Data Type: `numeric`
- Introduced: MariaDB 10.1.2, MariaDB 10.0.15 Fusion-io
- Removed: MariaDB 10.2.4

`Innodb_page_compression_trim_sect32768`

- Description: Number of TRIM operations performed for the page-compression/NVM Compression workload for the 32K block-size.
- Scope:
- Data Type: `numeric`
- Introduced: MariaDB 10.1.2, MariaDB 10.0.15 Fusion-io
- Removed: MariaDB 10.2.4

`Innodb_page_size`

- Description: Page size used by InnoDB. Defaults to 16KB, can be compiled with a different value.
- Scope: Global
- Data Type: `numeric`

`Innodb_pages_created`

- Description: Number of InnoDB pages created.
- Scope: Global
- Data Type: `numeric`

`Innodb_pages_read`

- Description: Number of InnoDB pages read.
- Scope: Global
- Data Type: `numeric`

`Innodb_pages0_read`

- Description: Counter for keeping track of reads of the first page of InnoDB data files, because the original implementation of data-at-rest-encryption for InnoDB introduced new code paths for reading the pages. Removed in MariaDB 10.4 as the extra reads of the first page were removed, and the encryption subsystem will be initialized whenever we first read the first page of each data file, in fil_node_open_file().
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.2.4, MariaDB 10.1.21
- Removed: MariaDB 10.4

`Innodb_pages_written`

- Description: Number of InnoDB pages written.
- Scope: Global
- Data Type: `numeric`

`Innodb_purge_trx_id`

- Description: Purge transaction id as shown in the TRANSACTIONS section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.2

`Innodb_purge_undo_no`

- Description: As shown in the TRANSACTIONS section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.2

`Innodb_read_views_memory`

- Description: As shown in the BUFFER POOL AND MEMORY section of the SHOW ENGINE INNODB STATUS output. Shows the total of memory in bytes allocated for the InnoDB read view.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5.32
- Removed: MariaDB 10.2

`Innodb_row_lock_current_waits`

- Description: Number of pending row lock waits on InnoDB tables.
- Scope: Global
- Data Type: `numeric`

`Innodb_row_lock_numbers`

- Description: Number of current row locks on InnoDB tables as shown in the TRANSACTIONS section of the SHOW ENGINE INNODB STATUS output. Renamed to InnoDB_current_row_locks in XtraDB 5.5.10-20.1.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5 / XtraDB 5.5.8-20
- Removed: MariaDB 5.5 / XtraDB 5.5.10-20.1

`Innodb_row_lock_time`

- Description: Total time in milliseconds spent getting InnoDB row locks.
- Scope: Global
- Data Type: `numeric`

`Innodb_row_lock_time_avg`

- Description: Average time in milliseconds spent getting an InnoDB row lock.
- Scope: Global
- Data Type: `numeric`

`Innodb_row_lock_time_max`

- Description: Maximum time in milliseconds spent getting an InnoDB row lock.
- Scope: Global
- Data Type: `numeric`

`Innodb_row_lock_waits`

- Description: Number of times InnoDB had to wait before getting a row lock.
- Scope: Global
- Data Type: `numeric`

`Innodb_rows_deleted`

- Description: Number of rows deleted from InnoDB tables that where not system tables. Almost equivalent to Handler_delete which does include system tables.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.10

`Innodb_rows_inserted`

- Description: Number of rows inserted into InnoDB tables that where not system tables. No direct equivalent in Handler status variables.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.10

`Innodb_rows_read`

- Description: Number of rows read from InnoDB tables that where not system tables. Almost equivalent to the sum of Handler_read* status variables which do include system tables.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.10

`Innodb_rows_updated`

- Description: Number of rows updated in InnoDB tables that where not system tables. Almost equivalent to Handler_update which does include system tables.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.10

`Innodb_s_lock_os_waits`

- Description: As shown in the SEMAPHORES section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.2

`Innodb_s_lock_spin_rounds`

- Description: As shown in the SEMAPHORES section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.2

`Innodb_s_lock_spin_waits`

- Description: As shown in the SEMAPHORES section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.2

`Innodb_scrub_background_page_reorganizations`

- Description: See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.3
- Removed: MariaDB 10.5.2

`Innodb_scrub_background_page_split_failures_missing_index`

- Description: See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.3
- Removed: MariaDB 10.5.2

`Innodb_scrub_background_page_split_failures_out_of_filespace`

- Description: See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.1.3
- Removed: MariaDB 10.5.2

`Innodb_scrub_background_page_split_failures_underflow`

- Description: See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.5.2

`Innodb_scrub_background_page_split_failures_unknown`

- Description: See Table and Tablespace Encryption.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.5.2

`Innodb_scrub_background_page_splits`

- Description: See Table and Tablespace Encryption.
- Scope: Global

- Data Type: `numeric`
- Removed: MariaDB 10.5.2

`Innodb_scrub_log`

- Description:
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.2.4
- Removed: MariaDB 10.5.2

`Innodb_secondary_index_triggered_cluster_reads`

- Description: Used to track the effectiveness of the Prefix Index Queries Optimization (MDEV-6929). Removed in MariaDB 10.10 as the optimization is now always enabled.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.10

`Innodb_secondary_index_triggered_cluster_reads_avoided`

- Description: Used to track the effectiveness of the Prefix Index Queries Optimization (MDEV-6929). Removed in MariaDB 10.10 as the optimization is now always enabled.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.10

`Innodb_system_rows_deleted`

- Description: Number of rows deleted on system tables.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.10

`Innodb_system_rows_inserted`

- Description: Number of rows inserted on system tables.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.10

`Innodb_system_rows_read`

- Description: Number of rows read on system tables.
- Scope: Global
- Data Type: `numeric`
- Removed: MariaDB 10.10.0

`Innodb_system_rows_updated`

- Description: Number of rows updated on system tables.
- Scope: Global

- Data Type: `numeric`
- Removed: MariaDB 10.10.0

`Innodb_truncated_status_writes`

- Description: Number of times output from SHOW ENGINE INNODB STATUS has been truncated.
- Scope: Global
- Data Type: `numeric`

`Innodb_undo_truncations`

- Description: Number of undo tablespace truncation operations.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 10.3.10

`Innodb_x_lock_os_waits`

- Description: As shown in the SEMAPHORES section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.2

`Innodb_x_lock_spin_rounds`

- Description: As shown in the SEMAPHORES section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.2

`Innodb_x_lock_spin_waits`

- Description: As shown in the SEMAPHORES section of the SHOW ENGINE INNODB STATUS output.
  - In MariaDB 5.5 and MariaDB 10.1, this status variable is present in XtraDB.
  - In MariaDB 10.2 and later, this status variable is not present.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 5.5
- Removed: MariaDB 10.2

This page is licensed: CC BY-SA / Gnu FDL

Setting Innodb Buffer Pool Size Dynamically

Resizing the buffer pool is performed in chunks determined by the size of the innodb_buffer_pool_chunk_size variable.

The resize operation waits until all active transactions and operations are completed, and new transactions and operations that need to access the buffer pool must wait until the resize is complete (although when decreasing the size, access is permitted when defragmenting and withdrawing pages).

Nested transactions may fail if started after the buffer pool resize has begun.

The new buffer pool size must be a multiple of innodb_buffer_pool_chunk_size * innodb_buffer_pool_instances (note that `innodb_buffer_pool_instances` is ignored from MariaDB 10.5, and removed in MariaDB 10.6, as the buffer pool is no longer split into multiple instances). If you attempt to set a different figure, the value is automatically adjusted to a multiple of at least the attempted size. Note that adjusting the innodb_buffer_pool_chunk_size setting can result in a change in the buffer pool size.

The number of chunks as calculated by innodb_buffer_pool_size / innodb_buffer_pool_chunk_size should not exceed 1000 in order to avoid performance issues.

A background thread performs the resizing operation. The Innodb_buffer_pool_resize_status status variable shows the progress of the resizing operation, for example:

```
SHOW STATUS LIKE 'Innodb_buffer_pool_resize_status';
+---------------------------------+---------------------------------+
| Variable_name                   | Value                           |
+---------------------------------+---------------------------------+
| Innodb_buffer_pool_resize_status | Resizing also other hash tables. |
+---------------------------------+---------------------------------+
```

or

```
SHOW STATUS LIKE 'Innodb_buffer_pool_resize_status';
+---------------------------------+-----------------------------------------------------+
| Variable_name                   | Value                                               |
+---------------------------------+-----------------------------------------------------+
| Innodb_buffer_pool_resize_status | Completed resizing buffer pool at 161103 16:26:54. |
+---------------------------------+-----------------------------------------------------+
```

Progress is also logged in the error log.

This page is licensed: CC BY-SA / Gnu FDL

Sphinx Status Variables

This page documents status variables related to the Sphinx storage engine. See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

See also the Full list of MariaDB options, system and status variables.

`Sphinx_error`

- Description: See SHOW ENGINE SPHINX STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Sphinx_time`

- Description: See SHOW ENGINE SPHINX STATUS.
- Scope: Global, Session

- Data Type: `numeric`

`Sphinx_total`

- Description: See SHOW ENGINE SPHINX STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Sphinx_total_found`

- Description: See SHOW ENGINE SPHINX STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Sphinx_word_count`

- Description: See SHOW ENGINE SPHINX STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Sphinx_words`

- Description: See SHOW ENGINE SPHINX STATUS.
- Scope: Global, Session
- Data Type: `numeric`

This page is licensed: CC BY-SA / Gnu FDL

Spider Status Variables

The following status variables are associated with the Spider storage engine. See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

See also the Full list of MariaDB options, system and status variables.

`Spider_direct_aggregate`

- Description: Number of times direct aggregate has happened in spider. That is, in a partitioned spider table, instead of scanning individual rows in remote tables corresponding to partitions for aggregation functions like `SUM` and `COUNT`, spider forwards aggregate queries to these tables directly and aggregate the results.
- Scope: Global, Session
- Data Type: `numeric`

`Spider_direct_delete`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.3.3

`Spider_direct_order_limit`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Spider_direct_update`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.3.3

`Spider_mon_table_cache_version`

- Description: The version of the spider monitoring table cache, always less than or equal to `Spider_mon_table_cache_version_req`. If the inequality is strict, then the cache is refreshed when the spider udf `spider_ping_table` is called. When the cache is refreshed, the value of `Spider_mon_table_cache_version` is brought up to be the same value as `Spider_mon_table_cache_version_req`.
- Scope: Global, Session
- Initial value: 0
- Data Type: `numeric`

`Spider_mon_table_cache_version_req`

- Description: The required version of the spider monitoring table cache. A call to the spider udf spider_flush_table_mon_cache will cause its value to be incremented by one, thus ensuring a refresh of the cache will happen when needed.
- Scope: Global, Session
- Initial value: 1
- Data Type: `numeric`

`Spider_parallel_search`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.3.3

This page is licensed: CC BY-SA / Gnu FDL

SQL Error Log System Variables and Options

This page documents system variables and options related to the SQL_Error_Log Plugin. See Server System Variables for a complete list of system variables and instructions on setting them.

See also the Full list of MariaDB options, system and status variables.

Options

`sql_error_log`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

        - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
    - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--sql-error-log=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

System Variables

`sql_error_log_filename`

- Description: The name (and optionally path) of the logfile containing the errors. Rotation will use a naming convention such as `sql_error_log_filename.001`. If no path is specified, the log file will be written to the data directory.
- Command line: `--sql-error-log-filename=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `sql_errors.log`

`sql_error_log_rate`

- Description: The logging sampling rate. Setting to `10`, for example, means that one in ten errors will be logged. If set to zero, logging is disabled. The default, `1`, logs every error.
- Command line: `--sql-error-log-rate=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`

`sql_error_log_rotate`

- Description: Setting to #1 `forces log rotation.`
- Command line: `--sql-error-log-rate[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`sql_error_log_rotations`

- Description: Number of rotations before the log is removed. When rotated, the current log file is stored and a new, empty, log is created. Any rotations older than this setting are removed.
- Command line: `--sql-error-log-rotations=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`

- Default Value: `9`
- Range: `1` to `999`

`sql_error_log_size_limit`

- Description: The log file size limit in bytes. After reaching this size, the log file is rotated.
- Command line: `--sql-error-log-size-limit=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1000000`
- Range: `100` to `9223372036854775807`

`sql_error_log_warnings`

- Description: If set, log warnings in addition to errors.
- Command line: `--sql-error-log-warnings={0,1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 10.11.5

`sql_error_log_with_db_and_thread_info`

- Description: If enabled, it prints the database name and the thread ID in the log in addition to already existing columns.
- Command line: `--sql-error-log-with-db-and-thread-info=value`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.6.17, MariaDB 10.11.7, MariaDB 11.0.5, MariaDB 11.1.4, MariaDB 11.2.3

This page is licensed: CC BY-SA / Gnu FDL

SSL/TLS Status Variables

The status variables listed on this page relate to encrypting data during transfer with the Transport Layer Security (TLS) protocol. Often, the term Secure Socket Layer (SSL) is used interchangeably with TLS, although strictly speaking, the SSL protocol is a predecessor to TLS and is no longer considered secure.

For compatibility reasons, the TLS status variables in MariaDB still use the `Ssl_` prefix, but MariaDB only supports its more secure successors. For more information on SSL/TLS in MariaDB, see Secure Connections Overview.

Some of these status values are not under the control of the server, but are reporting back settings of the underlying SSL library used by the server.

Variables

`Ssl_accept_renegotiates`

- Description: Number of negotiations needed to establish the TLS connection. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_accepts`

- Description: Number of accepted TLS handshakes. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_callback_cache_hits`

- Description: Number of sessions retrieved from the session cache. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_cipher`

- Description: The TLS cipher currently in use.
- Scope: Global, Session
- Data Type: `string`

`Ssl_cipher_list`

- Description: List of the available TLS ciphers.
  This is not necessarily the list of ciphers the MariaDB server can actually accept, but rather the list of ciphers supported by the underlying SSL library in general. E.g. some of these may not be compatible with the servers SSL certificate and so can't be used to connect to that server.
- Scope: Global, Session
- Data Type: `string`

`Ssl_client_connects`

- Description: Number of TLS handshakes started in client mode. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_connect_renegotiates`

- Description: Number of negotiations needed to establish the connection to a TLS-enabled master. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_ctx_verify_depth`

- Description: Number of tested TLS certificates in the chain. The global value can be flushed by FLUSH STATUS.

- Scope: Global
- Data Type: `numeric`

`Ssl_ctx_verify_mode`

- Description: Mode used for TLS context verification.The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_default_timeout`

- Description: Default timeout for TLS, in seconds.
- Scope: Global, Session
- Data Type: `numeric`

`Ssl_finished_accepts`

- Description: Number of successful TLS sessions in server mode. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_finished_connects`

- Description: Number of successful TLS sessions in client mode. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_server_not_after`

- Description: Last valid date for the server TLS certificate.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.0

`Ssl_server_not_before`

- Description: First valid date for the server TLS certificate.
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.0

`Ssl_session_cache_hits`

- Description: Number of TLS sessions found in the session cache. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_session_cache_misses`

- Description: Number of TLS sessions not found in the session cache. The global value can be flushed by FLUSH STATUS.

- Scope: Global
- Data Type: `numeric`

`Ssl_session_cache_mode`

- Description: Mode used for TLS caching by the server.
- Scope: Global
- Data Type: `string`

`Ssl_session_cache_overflows`

- Description: Number of sessions removed from the session cache because it was full. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_session_cache_size`

- Description: Size of the session cache. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_session_cache_timeouts`

- Description: Number of sessions which have timed out. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_sessions_reused`

- Description: Number of sessions reused. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Ssl_used_session_cache_entries`

- Description: Current number of sessions in the session cache. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Ssl_verify_depth`

- Description: TLS verification depth.
  How many levels within the certificate signing chain the verification should cover. This is not set by the server itself but by the OpenSSL configuration, and will typically show a very large number like 18446744073709551615, basically meaning “always verify the full signing chain up to the root CA”
- Scope: Global, Session
- Data Type: `numeric`

`Ssl_verify_mode`

- Description: TLS verification mode.

The OpenSSL verification mode flags in effect on the server side.

The value so far is always 5 when OpenSSL encryption is in effect, and 0 when not using encryption, or when the server is compiled against WolfSSL.

The value 5 is a combination of the `SSL_VERIFY_PEER` and `SSL_VERIFY_CLIENT_ONCE` flags, meaning that the client may send a certificate of its own connect, but will not be asked to send one again later for the duration of the encrypted connection.

The `SSL_VERIFY_FAIL_IF_NO_PEER_CERT` is not used at this point to enforce that the client sends a certificate if the database user was created with `REQUIRE X509`, `REQUIRE ISSUER` or `REQUIRE SUBJECT`, whether a client certificate was indeed sent, and whether it fits additional `REQUIRE` conditions, is checked by the server itself at a later state.

See also #notes for a description of the OpenSSL verify mode flags.

- Scope: Global, Session
- Data Type: `numeric`

`Ssl_version`

- Description: TLS version in use by the current session, possible values are `TLSv1.0`, `TLSv1.1`, `TLSv1.2` and `TLSv1.3`
- Scope: Global, Session
- Data Type: `string`

See Also

- Server Status Variables - complete list of status variables.
- Full list of MariaDB options, system and status variables

This page is licensed: CC BY-SA / Gnu FDL

System and Status Variables Added By Major Release

System and status variables added by major MariaDB Server releases, split by edition. Older, unmaintained releases are listed separately.

See Community Server.

System and status variables added in maintained MariaDB Community Server releases.

See Enterprise Server.

System and status variables added in MariaDB Enterprise Server releases.

See System and Status Variables Added By Major Unmaintained Release.

System and status variables added in older, unmaintained MariaDB Server releases.

This page is licensed: CC BY-SA / Gnu FDL

Community Server

System and Status Variables Added in Community Server

See System Variables Added in MariaDB 12.3.

System variables added in the MariaDB 12.3 series, with links to each and the release in which it was added.

See Status Variables Added in MariaDB 12.3.

No new status variables were added in the MariaDB 12.3 series.

See System Variables Added in MariaDB 12.2.

No new system variables were added in the MariaDB 12.2 series.

See Status Variables Added in MariaDB 12.2.

No new status variables were added in the MariaDB 12.2 series.

See System Variables Added in MariaDB 12.1.

System variables added in the MariaDB 12.1 series, with links to each and the release in which it was added.

See Status Variables Added in MariaDB 12.1.

Status variables added in the MariaDB 12.1 series, with links to each and the release in which it was added.

See System Variables Added in MariaDB 12.0.

System variables added in the MariaDB 12.0 series, with links to each and the release in which it was added.

See System Variables Added in MariaDB 11.8.

System variables added in the MariaDB 11.8 series, with links to each and the release in which it was added.

See Status Variables Added in MariaDB 11.8.

Status variables added in the MariaDB 11.8 series, with links to each and the release in which it was added.

See System Variables Added in MariaDB 11.4.

System variables added in the MariaDB 11.4 series, with links to each and the release in which it was added.

See Status Variables Added in MariaDB 11.4.

Status variables added in the MariaDB 11.4 series, with links to each and the release in which it was added.

See System Variables Added in MariaDB 10.11.

System variables added in the MariaDB 10.11 series, with links to each and the release in which it was added.

See System Variables Added in MariaDB 10.6.

System variables added in the MariaDB 10.6 series, with links to each and the release in which it was added.

See Status Variables Added in MariaDB 10.6.

Status variables added in the MariaDB 10.6 series, with links to each and the release in which it was added.

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 12.3

This is a list of system variables that have been added in the MariaDB 12.3 series.

Variable	Added
binlog_directory	MariaDB 12.3.1
binlog_row_event_fragment_threshold	MariaDB 12.3.1
binlog_storage_engine	MariaDB 12.3.1
innodb_binlog_state_interval	MariaDB 12.3.1
path	MariaDB 12.3.1

See Also

- System Variables Added in MariaDB 12.2

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 12.3

No new status variables were added in the MariaDB 12.3 series.

See Also

- System Variables Added in MariaDB 12.3
- Status Variables Added in Enterprise Server 12.3
- Status Variables Added in MariaDB 12.2

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 12.2

This is a list of system variables that have been added in the MariaDB 12.2 series.

No new system variables were added in the MariaDB 12.2 series. The 12.2 release focused on optimizer hints and Oracle-compatibility functions rather than new configuration options.

See Also

- System Variables Added in MariaDB 12.1
- System Variables Added in MariaDB 12.3

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 12.2

No new status variables were added in the MariaDB 12.2 series.

See Also

- System Variables Added in MariaDB 12.2
- Status Variables Added in MariaDB 12.1
- Status Variables Added in MariaDB 12.3

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 12.1

This is a list of system variables that have been added in the MariaDB 12.1series.

Variable	Added
analyze_max_length	MariaDB 12.1.1
aria_pagecache_segments	MariaDB 12.1.0
caching_sha2_password_auto_generate_rsa_keys	MariaDB 12.1.0
caching_sha2_password_digest_rounds	MariaDB 12.1.0
caching_sha2_password_private_key_path	MariaDB 12.1.0
caching_sha2_password_public_key_path	MariaDB 12.1.0
metadata_locks_instances	MariaDB 12.1.0
optimizer_record_context	MariaDB 12.1.0
wsrep_applier_retry_count	MariaDB 12.1.0

See Also

- System Variables Added in MariaDB 12.0

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 12.1

This is a list of status variables that have been added in the MariaDB 12.1 series.

`Feature_vector_index` first appeared in MariaDB 11.8; the MariaDB 12.0 series predates the vector index feature's merge into the 12.x line, so it did not enter 12.x until MariaDB 12.1.2 and is listed here for the 12.1 series.

Variable	Added
Feature_vector_index	MariaDB 12.1.2

See Also

- System Variables Added in MariaDB 12.1
- Status Variables Added in MariaDB 11.8
- Status Variables Added in MariaDB 12.2

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 12.0

This is a list of system variables that have been added in the MariaDB 12.0 series.

Variable	Added
analyze_max_length	MariaDB 12.0.2
create_tmp_table_binlog_formats	MariaDB 12.0.1
master_info_file	MariaDB 12.0.1
max_open_cursors	MariaDB 12.0.1
replicate_same_server_id	MariaDB 12.0.1
show_slave_auth_info	MariaDB 12.0.1
ssl_passphrase	MariaDB 12.0.1

See Also

- System Variables Added in MariaDB 11.8

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 11.8

Variable	Added
analyze_max_length	MariaDB 11.8.3
innodb_buffer_pool_size_auto_min	MariaDB 11.8.2
innodb_buffer_pool_size_max	MariaDB 11.8.2
innodb_log_checkpoint_now	MariaDB 11.8.2

For system variables added since the previous long-term release, MariaDB 11.4, see:

- System Variables Added in MariaDB 11.7
- System Variables Added in MariaDB 11.6

- System Variables Added in MariaDB 11.5

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 11.8

This is a list of status variables that have been added in the MariaDB 11.8 series.

Variable	Added
Feature_vector_index	MariaDB 11.8.4
Max_tmp_space_used	MariaDB 11.8.0
Tmp_space_used	MariaDB 11.8.0
innodb_async_reads_pending	MariaDB 11.8.0
innodb_async_reads_queue_size	MariaDB 11.8.0
innodb_async_reads_tasks_running	MariaDB 11.8.0
innodb_async_reads_total_count	MariaDB 11.8.0
innodb_async_reads_total_enqueues	MariaDB 11.8.0
innodb_async_reads_wait_slot_sec	MariaDB 11.8.0
innodb_async_writes_pending	MariaDB 11.8.0
innodb_async_writes_queue_size	MariaDB 11.8.0
innodb_async_writes_tasks_running	MariaDB 11.8.0
innodb_async_writes_total_count	MariaDB 11.8.0
innodb_async_writes_total_enqueues	MariaDB 11.8.0
innodb_async_writes_wait_slot_sec	MariaDB 11.8.0

See Also

- System Variables Added in MariaDB 11.8
- Status Variables Added in Enterprise Server 11.8
- Status Variables Added in MariaDB 11.4

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 11.4

This is a list of system variables that have been added in the MariaDB 11.4 series.

Variable	Added
analyze_max_length	MariaDB 11.4.8
binlog_gtid_index	MariaDB 11.4.0
binlog_gtid_index_page_size	MariaDB 11.4.0
binlog_gtid_index_span_min	MariaDB 11.4.0
binlog_legacy_event_pos	MariaDB 11.4.0
binlog_space_limit	MariaDB 11.4.0
innodb_alter_copy_bulk	MariaDB 11.4.3
innodb_buffer_pool_size_auto_min	MariaDB 11.4.6
innodb_buffer_pool_size_max	MariaDB 11.4.6
innodb_log_checkpoint_now	MariaDB 11.4.6

Variable	Added
innodb_log_file_mmap	MariaDB 11.4.4
innodb_log_spin_wait_delay	MariaDB 11.4.2
innodb_log_write_ahead_size	MariaDB 11.4.3
innodb_snapshot_isolation	MariaDB 11.4.2
max_binlog_total_size	MariaDB 11.4.0
optimizer_join_limit_pref_ratio	MariaDB 11.4.4
server_uid	MariaDB 11.4.3
slave_connections_needed_for_purge	MariaDB 11.4.0

For system variables added since the previous long-term release, MariaDB 10.11, see:

- System Variables Added in MariaDB 11.3
- System Variables Added in MariaDB 11.2
- System Variables Added in MariaDB 11.1
- System Variables Added in MariaDB 11.0

See Also

- Status Variables Added in MariaDB 11.4

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 11.4

This is a list of status variables that were added in the MariaDB 11.4 series.

Variable	Added
binlog_gtid_index_hit	MariaDB 11.4.0
binlog_gtid_index_miss	MariaDB 11.4.0

For status variables added since the previous long-term release, MariaDB 10.11, see:

- Status Variables Added in MariaDB 11.0

See Also

- System variables added in MariaDB 11.4

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.11

This is a list of system variables that have been added in the MariaDB 10.11 series.

Variable	Added
analyze_max_length	MariaDB 10.11.14
innodb_alter_copy_bulk	MariaDB 10.11.9
innodb_buffer_pool_size_auto_min	MariaDB 10.11.12
innodb_buffer_pool_size_max	MariaDB 10.11.12
innodb_log_checkpoint_now	MariaDB 10.11.12
innodb_log_file_mmap	MariaDB 10.11.10

Variable	Added
innodb_log_spin_wait_delay	MariaDB 10.11.8
innodb_log_write_ahead_size	MariaDB 10.11.9
innodb_snapshot_isolation	MariaDB 10.11.8
log_slow_max_warnings	MariaDB 10.11.6
log_slow_min_examined_row_limit	MariaDB 10.11.0
log_slow_query	MariaDB 10.11.0
log_slow_query_file	MariaDB 10.11.1
log_slow_query_time	MariaDB 10.11.0
optimizer_join_limit_pref_ratio	MariaDB 10.11.10
optimizer_max_sel_args	MariaDB 10.11.6
replicate_rewrite_db	MariaDB 10.11.0
server_uid	MariaDB 10.11.9
system_versioning_insert_history	MariaDB 10.11.0

For system variables added since the previous long-term release, MariaDB 10.6, see:

- System Variables Added in MariaDB 10.10
- System Variables Added in MariaDB 10.9
- System Variables Added in MariaDB 10.8

For system variables that have been removed or deprecated, see Upgrading from MariaDB 10.6 to MariaDB 10.11.

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.6

This is a list of system variables that have been added in the MariaDB 10.6 series. The list does not include variables that are not part of the default release.

Variable	Added
analyze_max_length	MariaDB 10.6.23
binlog_expire_logs_seconds	MariaDB 10.6.0
innodb_deadlock_report	MariaDB 10.6.0
innodb_read_only_compressed	MariaDB 10.6.0
innodb_snapshot_isolation	MariaDB 10.6.18
log_slow_max_warnings	MariaDB 10.6.16
note_verbosity	MariaDB 10.6.16
optimizer_adjust_secondary_key_costs	MariaDB 10.6.17
optimizer_join_limit_pref_ratio	MariaDB 10.6.20
optimizer_max_sel_args	MariaDB 10.6.16
server_uid	MariaDB 10.6.19
wsrep_mode	MariaDB 10.6.0

For system variables that have been removed or deprecated, see Upgrading from MariaDB 10.5 to MariaDB 10.6.

See Also

- Status Variables Added in MariaDB 10.6
- System Variables Added in MariaDB 10.5
- System Variables Added in MariaDB 10.4

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 10.6

This is a list of status variables that were added in the MariaDB 10.6 series.

Variable	Added
Innodb_buffer_pool_pages_lru_freed	MariaDB 10.6.0
resultset_metadata_skipped	MariaDB 10.6.0

See Also

- System variables added in MariaDB 10.6
- Status variables added in MariaDB 10.5
- Status variables added in MariaDB 10.4

This page is licensed: CC BY-SA / Gnu FDL

Enterprise Server

System and Status Variables Added in Enterprise Server

See System Variables Added in Enterprise Server 12.3.

System variables new in MariaDB Enterprise Server 12.3, relative to the previous Enterprise Server release series.

See Status Variables Added in Enterprise Server 12.3.

No new status variables were added in MariaDB Enterprise Server 12.3.

See System Variables Added in Enterprise Server 11.8.

System variables new in MariaDB Enterprise Server 11.8, relative to the previous Enterprise Server release series.

See Status Variables Added in Enterprise Server 11.8.

Status variables new in MariaDB Enterprise Server 11.8, relative to the previous Enterprise Server release series.

See System Variables Added in Enterprise Server 11.4.

System variables new in MariaDB Enterprise Server 11.4, relative to the previous Enterprise Server release series.

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in Enterprise Server 12.3

This is a list of system variables that are new in MariaDB Enterprise Server 12.3, relative to the previous Enterprise Server release series, Enterprise Server 11.8.

Because Enterprise Server backports selected features across release lines, some system variables that were introduced in Community Server 12.0, 12.1, and 12.3 are **not** listed here: they had already been backported into Enterprise Server 11.8 and so were not new in Enterprise Server 12.3. Examples are `analyze_max_length`, `aria_pagecache_segments`, `max_open_cursors`, and `metadata_locks_instances`.

Variable	Added
binlog_directory	Enterprise Server 12.3
binlog_row_event_fragment_threshold	Enterprise Server 12.3
binlog_storage_engine	Enterprise Server 12.3
create_tmp_table_binlog_formats	Enterprise Server 12.3
innodb_binlog_state_interval	Enterprise Server 12.3
master_info_file	Enterprise Server 12.3
optimizer_record_context	Enterprise Server 12.3
path	Enterprise Server 12.3
replicate_same_server_id	Enterprise Server 12.3
show_slave_auth_info	Enterprise Server 12.3
ssl_passphrase	Enterprise Server 12.3
wsrep_applier_retry_count	Enterprise Server 12.3

See Also

- System Variables Added in MariaDB 12.3 (Community Server)

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in Enterprise Server 12.3

No new status variables were added in MariaDB Enterprise Server 12.3, relative to the previous Enterprise Server release series, Enterprise Server 11.8.

See Also

- Status Variables Added in MariaDB 12.3 (Community Server)
- System Variables Added in Enterprise Server 12.3

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in Enterprise Server 11.8

This is a list of system variables that are new in MariaDB Enterprise Server 11.8, relative to the previous Enterprise Server release series, Enterprise Server 11.4.

Because Enterprise Server backports selected features across release lines, some system variables that were introduced in Community Server 11.5 through 11.8 are **not** listed here: they had already been backported into Enterprise Server 11.4 and so were not new in Enterprise Server 11.8. Examples are `max_tmp_total_space_usage` and the `mhnsw_*` vector-index variables.

Variable	Added
binlog_large_commit_threshold	Enterprise Server 11.8
innodb_adaptive_hash_index_cells	Enterprise Server 11.8
log_slow_always_query_time	Enterprise Server 11.8
max_open_cursors	Enterprise Server 11.8
slave_abort_blocking_timeout	Enterprise Server 11.8
wsrep_sync_wait_timeout	Enterprise Server 11.8

See Also

- System Variables Added in MariaDB 11.8 (Community Server)

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in Enterprise Server 11.8

This is a list of status variables that are new in MariaDB Enterprise Server 11.8, relative to the previous Enterprise Server release series, Enterprise Server 11.4.

Because Enterprise Server backports selected features across release lines, some status variables that were introduced in Community Server 11.5 through 11.8 are **not** listed here: they had already been backported into Enterprise Server 11.4 and so were not new in Enterprise Server 11.8. Examples are `Max_tmp_space_used` and `Tmp_space_used`.

Variable	Added
Feature_vector_index	Enterprise Server 11.8
innodb_async_reads_pending	Enterprise Server 11.8
innodb_async_reads_queue_size	Enterprise Server 11.8
innodb_async_reads_tasks_running	Enterprise Server 11.8
innodb_async_reads_total_count	Enterprise Server 11.8
innodb_async_reads_total_enqueues	Enterprise Server 11.8
innodb_async_reads_wait_slot_sec	Enterprise Server 11.8
innodb_async_writes_pending	Enterprise Server 11.8
innodb_async_writes_queue_size	Enterprise Server 11.8
innodb_async_writes_tasks_running	Enterprise Server 11.8
innodb_async_writes_total_count	Enterprise Server 11.8
innodb_async_writes_total_enqueues	Enterprise Server 11.8
innodb_async_writes_wait_slot_sec	Enterprise Server 11.8

See Also

- Status Variables Added in MariaDB 11.8 (Community Server)
- System Variables Added in Enterprise Server 11.8

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in Enterprise Server 11.4

This is a list of system variables that are new in MariaDB Enterprise Server 11.4, relative to the previous Enterprise Server release series, Enterprise Server 10.6.

Because Enterprise Server backports selected features across release lines, some system variables that were introduced in Community Server 10.7 through 11.4 are **not** listed here: they had already been backported into Enterprise Server 10.6 and so were not new in Enterprise Server 11.4. Examples are `replicate_rewrite_db` and `slave_max_statement_time`.

Variable	Added
allow_suspicious_udfs	Enterprise Server 11.4
aria_pagecache_segments	Enterprise Server 11.4
binlog_alter_two_phase	Enterprise Server 11.4

Variable	Added
binlog_do_db	Enterprise Server 11.4
binlog_gtid_index	Enterprise Server 11.4
binlog_gtid_index_page_size	Enterprise Server 11.4
binlog_gtid_index_span_min	Enterprise Server 11.4
binlog_ignore_db	Enterprise Server 11.4
binlog_legacy_event_pos	Enterprise Server 11.4
binlog_row_event_max_size	Enterprise Server 11.4
binlog_space_limit	Enterprise Server 11.4
block_encryption_mode	Enterprise Server 11.4
character_set_collations	Enterprise Server 11.4
innodb_alter_copy_bulk	Enterprise Server 11.4
innodb_buffer_pool_size_auto_min	Enterprise Server 11.4
innodb_data_file_buffering	Enterprise Server 11.4
innodb_data_file_write_through	Enterprise Server 11.4
innodb_log_file_mmap	Enterprise Server 11.4
innodb_log_file_write_through	Enterprise Server 11.4
innodb_log_spin_wait_delay	Enterprise Server 11.4
innodb_truncate_temporary_tablespace_now	Enterprise Server 11.4
log_slow_min_examined_row_limit	Enterprise Server 11.4
log_slow_query	Enterprise Server 11.4
log_slow_query_file	Enterprise Server 11.4
log_slow_query_time	Enterprise Server 11.4
max_binlog_total_size	Enterprise Server 11.4
max_tmp_session_space_usage	Enterprise Server 11.4
max_tmp_total_space_usage	Enterprise Server 11.4
metadata_locks_instances	Enterprise Server 11.4
mhnsw_default_distance	Enterprise Server 11.4
mhnsw_default_m	Enterprise Server 11.4
mhnsw_ef_search	Enterprise Server 11.4
mhnsw_max_cache_size	Enterprise Server 11.4
new_mode	Enterprise Server 11.4
optimizer_disk_read_cost	Enterprise Server 11.4
optimizer_disk_read_ratio	Enterprise Server 11.4
optimizer_extra_pruning_depth	Enterprise Server 11.4
optimizer_index_block_copy_cost	Enterprise Server 11.4
optimizer_key_compare_cost	Enterprise Server 11.4
optimizer_key_copy_cost	Enterprise Server 11.4
optimizer_key_lookup_cost	Enterprise Server 11.4
optimizer_key_next_find_cost	Enterprise Server 11.4

Variable	Added
optimizer_row_copy_cost	Enterprise Server 11.4
optimizer_row_lookup_cost	Enterprise Server 11.4
optimizer_row_next_find_cost	Enterprise Server 11.4
optimizer_rowid_compare_cost	Enterprise Server 11.4
optimizer_rowid_copy_cost	Enterprise Server 11.4
optimizer_scan_setup_cost	Enterprise Server 11.4
optimizer_where_cost	Enterprise Server 11.4
redirect_url	Enterprise Server 11.4
skip_grant_tables	Enterprise Server 11.4
slave_connections_needed_for_purge	Enterprise Server 11.4
system_versioning_insert_history	Enterprise Server 11.4
transaction_isolation	Enterprise Server 11.4
transaction_read_only	Enterprise Server 11.4
wsrep_allowlist	Enterprise Server 11.4
wsrep_status_file	Enterprise Server 11.4

See Also

- System Variables Added in MariaDB 11.4 (Community Server)

This page is licensed: CC BY-SA / Gnu FDL

System and Status Variables Added By Major Unmaintained Release

- System Variables Added in MariaDB 11.7
- System Variables Added in MariaDB 11.6
- System Variables Added in MariaDB 11.5
- Status Variables Added in MariaDB 11.5
- System Variables Added in MariaDB 11.3
- System Variables Added in MariaDB 11.2
- System Variables Added in MariaDB 11.1
- System Variables Added in MariaDB 11.0
- Status Variables Added in MariaDB 11.0
- System Variables Added in MariaDB 10.10
- System Variables Added in MariaDB 10.9
- System Variables Added in MariaDB 10.8
- System Variables Added in MariaDB 10.5
- Status Variables Added in MariaDB 10.5
- System Variables Added in MariaDB 10.4
- Status Variables Added in MariaDB 10.4
- System Variables Added in MariaDB 10.3
- Status Variables Added in MariaDB 10.3
- System Variables Added in MariaDB 10.2
- Status Variables Added in MariaDB 10.2
- System Variables Added in MariaDB 10.1

- Status Variables Added in MariaDB 10.1
- System Variables Added in MariaDB 10.0
- Status Variables Added in MariaDB 10.0
- System Variables Added in MariaDB 5.5

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 11.7

This is a list of system variables that have been added in the MariaDB 11.7 series.

Variable	Added
binlog_large_commit_threshold	MariaDB 11.7.0
innodb_log_file_mmap	MariaDB 11.7.1
log_slow_always_query_time	MariaDB 11.7.0
mhnsw_default_distance	MariaDB 11.7.1
mhnsw_default_m	MariaDB 11.7.1
mhnsw_ef_search	MariaDB 11.7.1
mhnsw_max_cache_size	MariaDB 11.7.1
slave_abort_blocking_timeout	MariaDB 11.7.0

See Also

- System Variables Added in MariaDB 11.6

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 11.6

This is a list of system variables that have been added in the MariaDB 11.6 series.

Variable	Added
innodb_log_file_mmap	MariaDB 11.6.2
optimizer_join_limit_pref_ratio	MariaDB 11.6.2
server_uid	MariaDB 11.6.1

See Also

- System Variables Added in MariaDB 11.5

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 11.5

This is a list of system variables that have been added in the MariaDB 11.5 series.

Variable	Added
innodb_alter_copy_bulk	MariaDB 11.5.2
innodb_log_spin_wait_delay	MariaDB 11.5.1
innodb_log_write_ahead_size	MariaDB 11.5.2
max_tmp_session_space_usage	MariaDB 11.5.0
max_tmp_total_space_usage	MariaDB 11.5.0

Variable	Added
server_uid	MariaDB 11.5.2

See Also

- Status Variables Added in MariaDB 11.5
- System Variables Added in MariaDB 11.4

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 11.5

This is a list of status variables that were added in the MariaDB 11.5 series.

Variable	Added
max_tmp_space_used	MariaDB 11.5.0
Innodb_async_reads_pending	MariaDB 11.5.0
Innodb_async_reads_queue_size	MariaDB 11.5.0
Innodb_async_reads_tasks_running	MariaDB 11.5.0
Innodb_async_reads_total_enqueues	MariaDB 11.5.0
Innodb_async_reads_total_count	MariaDB 11.5.0
Innodb_async_reads_wait_slot_sec	MariaDB 11.5.0
Innodb_async_writes_pending	MariaDB 11.5.0
Innodb_async_writes_queue_size	MariaDB 11.5.0
Innodb_async_writes_tasks_running	MariaDB 11.5.0
Innodb_async_writes_total_enqueues	MariaDB 11.5.0
Innodb_async_writes_total_count	MariaDB 11.5.0
Innodb_async_writes_wait_slot_sec	MariaDB 11.5.0
tmp_space_used	MariaDB 11.5.0

See Also

- System variables added in MariaDB 11.5

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 11.3

This is a list of system variables that have been added in the MariaDB 11.3 series.

Variable	Added
innodb_truncate_temporary_tablespace_now	MariaDB 11.3.0
redirect_url	MariaDB 11.3.0

See Also

- System Variables Added in MariaDB 11.2

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 11.2

This is a list of system variables that have been added in the MariaDB 11.2 series.

Variable	Added
binlog_do_db	MariaDB 11.2.0
binlog_ignore_db	MariaDB 11.2.0
binlog_row_event_max_size	MariaDB 11.2.0
block_encryption_mode	MariaDB 11.2.0
character_set_collations	MariaDB 11.2.1
innodb_alter_copy_bulk	MariaDB 11.2.5
innodb_log_file_mmap	MariaDB 11.2.6
innodb_log_spin_wait_delay	MariaDB 11.2.4
innodb_log_write_ahead_size	MariaDB 11.2.5
innodb_snapshot_isolation	MariaDB 11.2.4
optimizer_join_limit_pref_ratio	MariaDB 11.2.6
server_uid	MariaDB 11.2.5

See Also

- System Variables Added in MariaDB 11.1

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 11.1

This is a list of system variables that have been added in the MariaDB 11.1 series.

Variable	Added
innodb_alter_copy_bulk	MariaDB 11.1.6
innodb_log_spin_wait_delay	MariaDB 11.1.5
innodb_log_write_ahead_size	MariaDB 10.11.9
innodb_snapshot_isolation	MariaDB 11.1.6
log_slow_max_warnings	MariaDB 11.1.3
optimizer_max_sel_args	MariaDB 11.1.3
server_uid	MariaDB 11.1.6
transaction_isolation	MariaDB 11.1.0
transaction_read_only	MariaDB 11.1.0

See Also

- System Variables Added in MariaDB 11.0

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 11.0

This is a list of system variables that have been added in the MariaDB 11.0 series.

Variable	Added
innodb_data_file_buffering	MariaDB 11.0.1

Variable	Added
innodb_data_file_write_through	MariaDB 11.0.1
innodb_log_file_write_through	MariaDB 11.0.1
log_slow_max_warnings	MariaDB 11.0.4
innodb_log_spin_wait_delay	MariaDB 11.0.6
innodb_snapshot_isolation	MariaDB 11.0.6
optimizer_max_sel_args	MariaDB 11.0.4

For system variables that have been removed or deprecated, see Upgrading from MariaDB 10.11 to MariaDB 11.0.

See Also

- Status Variables Added in MariaDB 11.0
- System Variables Added in MariaDB 10.11

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 11.0

This is a list of status variables that were added in the MariaDB 11.0 series.

Variable	Added
max_used_connections_time	MariaDB 11.0.2

See Also

- System variables added in MariaDB 11.0

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.10

This is a list of system variables that have been added in the MariaDB 10.10 series.

Variable	Added
allow_suspicious_udfs	MariaDB 10.10.1
optimizer_extra_pruning_depth	MariaDB 10.10.1
log_slow_max_warnings	MariaDB 10.10.7
optimizer_max_sel_args	MariaDB 10.10.7
skip_grant_tables	MariaDB 10.10.1
slave_max_statement_time	MariaDB 10.10.1
wsrep_allowlist	MariaDB 10.10.1

See Also

- System Variables Added in MariaDB 10.9

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.9

This is a list of system variables that have been added in the MariaDB 10.9 series.

Variable	Added
rocksdb_log_dir	MariaDB 10.9.1
wsrep_status_file	MariaDB 10.9.1

See Also

- System Variables Added in MariaDB 10.8
- System Variables Added in MariaDB 10.6

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.8

This is a list of system variables that have been added in the MariaDB 10.8 series.

Variable	Added
binlog_alter_two_phase	MariaDB 10.8.1
innodb_log_file_buffering	MariaDB 10.8.4

For system variables that have been removed or deprecated, see Upgrading from MariaDB 10.7 to MariaDB 10.8.

See Also

- System Variables Added in MariaDB 10.6

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.5

This is a list of system variables that have been added in the MariaDB 10.5 series. The list does not include variables that are not part of the default release.

Variable	Added
binlog_row_metadata	MariaDB 10.5.0
innodb_instant_alter_column_allowed	MariaDB 10.5.3
innodb_lru_flush_size	MariaDB 10.5.7
innodb_max_purge_lag_wait	MariaDB 10.5.7
optimizer_max_sel_arg_weight	MariaDB 10.5.9
performance_schema_events_transactions_history_long_size	MariaDB 10.5.2
performance_schema_events_transactions_history_size	MariaDB 10.5.2
performance_schema_max_index_stat	MariaDB 10.5.2
performance_schema_max_memory_classes	MariaDB 10.5.2
performance_schema_max_metadata_locks	MariaDB 10.5.2
performance_schema_max_prepared_statement_instances	MariaDB 10.5.2
performance_schema_max_program_instances	MariaDB 10.5.2
performance_schema_max_sql_text_length	MariaDB 10.5.2
performance_schema_max_statement_stack	MariaDB 10.5.2
performance_schema_max_table_lock_stat	MariaDB 10.5.2
require_secure_transport	MariaDB 10.5.2
s3_access_key	MariaDB 10.5

Variable	Added
s3_block_size	MariaDB 10.5
s3_bucket	MariaDB 10.5
s3_debug	MariaDB 10.5
s3_host_name	MariaDB 10.5
s3_pagecache_age_threshold	MariaDB 10.5
s3_pagecache_buffer_size	MariaDB 10.5
s3_pagecache_division_limit	MariaDB 10.5
s3_pagecache_file_hash_size	MariaDB 10.5
s3_protocol_version	MariaDB 10.5
s3_region	MariaDB 10.5
s3_secret_key	MariaDB 10.5
server_uid	MariaDB 10.5.26
sql_if_exists	MariaDB 10.5.2
thread_pool_dedicated_listener	MariaDB 10.5.0
thread_pool_exact_stats	MariaDB 10.5.0

For system variables that have been removed or deprecated, see Upgrading from MariaDB 10.4 to MariaDB 10.5.

See Also

- Status Variables Added in MariaDB 10.5
- System Variables Added in MariaDB 10.4
- System Variables Added in MariaDB 10.3

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 10.5

This is a list of status variables that were added in the MariaDB 10.5 series.

Variable	Added
Innodb_adaptive_hash_hash_searches	MariaDB 10.5.0
Innodb_adaptive_hash_non_hash_searches	MariaDB 10.5.0
Innodb_background_log_sync	MariaDB 10.5.0
Innodb_buffer_pool_pages_made_not_young	MariaDB 10.5.0
Innodb_buffer_pool_pages_made_young	MariaDB 10.5.0
Innodb_buffer_pool_pages_old	MariaDB 10.5.0
Innodb_buffer_pool_pages_LRU_flushed	MariaDB 10.5.0
Innodb_buffered_aio_submitted	MariaDB 10.5.0
Innodb_checkpoint_age	MariaDB 10.5.0
Innodb_checkpoint_max_age	MariaDB 10.5.0
Innodb_deadlocks	MariaDB 10.5.0
Innodb_ibuf_discarded_delete_marks	MariaDB 10.5.0
Innodb_ibuf_discarded_deletes	MariaDB 10.5.0

Variable	Added
Innodb_ibuf_discarded_inserts	MariaDB 10.5.0
Innodb_ibuf_free_list	MariaDB 10.5.0
Innodb_ibuf_merged_delete_marks	MariaDB 10.5.0
Innodb_ibuf_merged_deletes	MariaDB 10.5.0
Innodb_ibuf_merged_inserts	MariaDB 10.5.0
Innodb_ibuf_merges	MariaDB 10.5.0
Innodb_ibuf_segment_size	MariaDB 10.5.0
Innodb_ibuf_size	MariaDB 10.5.0
Innodb_lsn_current	MariaDB 10.5.0
Innodb_lsn_flushed	MariaDB 10.5.0
Innodb_lsn_last_checkpoint	MariaDB 10.5.0
Innodb_master_thread_active_loops	MariaDB 10.5.0
Innodb_master_thread_idle_loops	MariaDB 10.5.0
Innodb_max_trx_id	MariaDB 10.5.0
Innodb_mem_adaptive_hash	MariaDB 10.5.0
Innodb_mem_dictionary	MariaDB 10.5.0
performance_schema_index_stat_lost	MariaDB 10.5.2
performance_schema_memory_classes_lost	MariaDB 10.5.2
performance_schema_metadata_lock_lost	MariaDB 10.5.2
performance_schema_nested_statement_lost	MariaDB 10.5.2
performance_schema_prepared_statements_lost	MariaDB 10.5.2
performance_schema_program_lost	MariaDB 10.5.2
performance_schema_table_lock_stat_lost	MariaDB 10.5.2
S3_pagecache_blocks_not_flushed	MariaDB 10.5
S3_pagecache_blocks_unused	MariaDB 10.5
S3_pagecache_blocks_used	MariaDB 10.5
S3_pagecache_reads	MariaDB 10.5

See Also

- System variables added in MariaDB 10.5
- Status variables added in MariaDB 10.4
- Status variables added in MariaDB 10.3

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.4

This is a list of system variables that have been added in the MariaDB 10.4 series. The list does not include variables that are not part of the default release.

Variable	Added
analyze_sample_percentage	MariaDB 10.4.3
default_password_lifetime	MariaDB 10.4.3

Variable	Added
disconnect_on_expired_password	MariaDB 10.4.3
gtid_cleanup_batch_size	MariaDB 10.4.1
innodb_encrypt_temporary_ables	MariaDB 10.4.7
innodb_instant_alter_column_allowed	MariaDB 10.4.13
max_password_errors	MariaDB 10.4.2
optimizer_trace	MariaDB 10.4.3
optimizer_trace_max_mem_size	MariaDB 10.4.3
tcp_nodelay	MariaDB 10.4.0
tls_version	MariaDB 10.4.6
wsrep_certification_rules	MariaDB 10.4.3
wsrep_trx_fragment_size	MariaDB 10.4.2
wsrep_trx_fragment_unit	MariaDB 10.4.2

For system variables that have been removed or deprecated, see Upgrading from MariaDB 10.3 to MariaDB 10.4.

See Also

- Status Variables Added in MariaDB 10.4
- System Variables Added in MariaDB 10.5
- System Variables Added in MariaDB 10.3
- System Variables Added in MariaDB 10.2

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 10.4

This is a list of status variables that were added in the MariaDB 10.4 series.

Variable	Added
Aborted_connects_preauth	MariaDB 10.4.5
Com_backup	MariaDB 10.4.1
Com_backup_lock	MariaDB 10.4.2
Feature_application_time_periods	MariaDB 10.4.5
wsrep_applier_thread_count	MariaDB 10.4.7
wsrep_rollbacker_thread_count	MariaDB 10.4.7

See Also

- System Variables Added in MariaDB 10.4
- Status Variables Added in MariaDB 10.5
- Status Variables Added in MariaDB 10.3
- Status Variables Added in MariaDB 10.2

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.3

This is a list of system variables that have been added in the MariaDB 10.3 series. The list does not include variables that are not part of the default release.

Variable	Added
alter_algorithm	MariaDB 10.3.7
bind_address	MariaDB 10.3.3
binlog_file_cache_size	MariaDB 10.3.3
gtid_pos_auto_engines	MariaDB 10.3.1
column_compression_threshold	MariaDB 10.3.2
column_compression_zlib_level	MariaDB 10.3.2
column_compression_zlib_strategy	MariaDB 10.3.2
column_compression_zlib_wrap	MariaDB 10.3.2
idle_readonly_transaction_timeout	MariaDB 10.3.0
idle_transaction_timeout	MariaDB 10.3.0
idle_write_transaction_timeout	MariaDB 10.3.0
in_predicate_conversion_threshold	MariaDB 10.3.18
innodb_encrypt_temporary_ables	MariaDB 10.3.17
innodb_instant_alter_column_allowed	MariaDB 10.3.23
log_disabled_statements	MariaDB 10.3.1
log_slow_disabled_statements	MariaDB 10.3.1
rpl_semi_sync_*	MariaDB 10.3.3
secure_timestamp	MariaDB 10.3.7
slave_transaction_retry_errors	MariaDB 10.3.3
slave_transaction_retry_interval	MariaDB 10.3.3
system_versioning_alter_history	MariaDB 10.3.4
system_versioning_asof	MariaDB 10.3.4
system_versioning_innodb_algorithm_simple	MariaDB 10.3.4
tcp_keepalive_interval	MariaDB 10.3.3
tcp_keepalive_probes	MariaDB 10.3.3
tcp_keepalive_time	MariaDB 10.3.3
version_source_revision	MariaDB 10.3.2
wsrep_certification_rules	MariaDB 10.3.13
wsrep_reject_queries	MariaDB 10.3.6

For system variables that have been removed or deprecated, see Upgrading from MariaDB 10.2 to MariaDB 10.3.

See Also

- Status Variables Added in MariaDB 10.3
- System Variables Added in MariaDB 10.4
- System Variables Added in MariaDB 10.2
- System Variables Added in MariaDB 10.1

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 10.3

This is a list of status variables that were added in the MariaDB 10.3 series.

Variable	Added
Acl_package_body_grants	MariaDB 10.3.5
Acl_package_spec_grants	MariaDB 10.3.5
Com_alter_sequence	MariaDB 10.3.1
Com_create_package	MariaDB 10.3.5
Com_create_package_body	MariaDB 10.3.5
Com_create_sequence	MariaDB 10.3.0
Com_drop_package	MariaDB 10.3.5
Com_drop_package_body	MariaDB 10.3.5
Com_drop_sequence	MariaDB 10.3.0
Com_show_create_package	MariaDB 10.3.5
Com_show_create_package_body	MariaDB 10.3.5
Com_show_package_status	MariaDB 10.3.5
Com_show_package_body_status	MariaDB 10.3.5
Feature_json	MariaDB 10.3.7
Feature_system_versioning	MariaDB 10.3.7
Handler_tmp_delete	MariaDB 10.3.0
Innodb_buffer_pool_load_incomplete	MariaDB 10.3.5
Memory_used_initial	MariaDB 10.3.3
Rpl_semi_sync*	MariaDB 10.3.3
Rpl_transactions_multi_engine	MariaDB 10.3.1
Table_open_cache_active_instances	MariaDB 10.3.3
Table_open_cache_hits	MariaDB 10.3.3
Table_open_cache_misses	MariaDB 10.3.3
Table_open_cache_overflows	MariaDB 10.3.3
Transactions_gtid_foreign_engine	MariaDB 10.3.1
Transactions_multi_engine	MariaDB 10.3.1

See Also

- System variables added in MariaDB 10.3
- Status variables added in MariaDB 10.4
- Status variables added in MariaDB 10.2
- Status variables added in MariaDB 10.1

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.2

This is a list of system variables that have been added in the MariaDB 10.2 series. The list does not include variables that are not part of the default release.

Variable	Added
aria_recover_options	MariaDB 10.2.0
eq_range_index_dive_limit	MariaDB 10.2.18

Variable	Added
innodb_adaptive_hash_index_parts	MariaDB 10.2.2
innodb_deadlock_detect	MariaDB 10.2.6
innodb_default_row_format	MariaDB 10.2.2
innodb_encrypt_temporary_ables	MariaDB 10.2.26
innodb_fill_factor	MariaDB 10.2.2
innodb_flush_sync	MariaDB 10.2.2
innodb_lock_schedule_algorithm	MariaDB 10.2.3
innodb_log_checksums	MariaDB 10.2.2
innodb_log_optimize_ddl	MariaDB 10.2.17
innodb_log_write_ahead_size	MariaDB 10.2.2
innodb_max_undo_log_size	MariaDB 10.2.2
innodb_page_cleaners	MariaDB 10.2.2
innodb_purge_rseg_truncate_frequency	MariaDB 10.2.2
innodb_safe_truncate	MariaDB 10.2.19
innodb_stats_include_delete_marked	MariaDB 10.2.6
innodb_temp_data_file_path	MariaDB 10.2.2
max_recursive_iterations	MariaDB 10.2.2
read_binlog_speed_limit	MariaDB 10.2.3
session_track_schema	MariaDB 10.2.2
session_track_state_change	MariaDB 10.2.2
session_track_system_variables	MariaDB 10.2.2
session_track_transaction_info	MariaDB 10.2.2
slave_parallel_workers	MariaDB 10.2.0
table_open_cache_instances	MariaDB 10.2.2
thread_pool_prio_kickup_timer	MariaDB 10.2.2
thread_pool_priority	MariaDB 10.2.2
tmp_disk_table_size	MariaDB 10.2.7
tmp_memory_table_size	MariaDB 10.2.7
wsrep_certification_rules	MariaDB 10.2.22

See Also

- Status Variables Added in MariaDB 10.2
- System Variables Added in MariaDB 10.3
- System Variables Added in MariaDB 10.1
- System Variables Added in MariaDB 10.0

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 10.2

This is a list of status variables that were added in the MariaDB 10.2 series.

Variable	Added
Com_alter_user	MariaDB 10.2.0
Com_execute_immediate	MariaDB 10.2.3
Com_multi	MariaDB 10.2.0
Com_show_create_user	MariaDB 10.2.0
Innodb_buffer_pool_resize_status	MariaDB 10.2.2
Innodb_encryption_num_key_requests	MariaDB 10.2.4
Innodb_have_punch_hole	MariaDB 10.2.4
Innodb_pages0read	MariaDB 10.2.4
Innodb_scrub_log	MariaDB 10.2.4

See Also

- System variables added in MariaDB 10.2
- Status variables added in MariaDB 10.3
- Status variables added in MariaDB 10.1
- Status variables added in MariaDB 10.0

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.1

This is a list of system variables that were added in the MariaDB 10.1 series.

The list excludes the following variables, related to storage engines and plugins included in MariaDB 10.1:

- Spider System Variables

Variable	Added
aria_encrypt_tables	MariaDB 10.1.3
default_tmp_storage_engine	MariaDB 10.1.0
encrypt_binlog	MariaDB 10.1.7
encrypt_tmp_disk_tables	MariaDB 10.1.3
encrypt_tmp_files	MariaDB 10.1.5
enforce_storage_engine	MariaDB 10.1.4
explicit_defaults_for_timestamp	MariaDB 10.1.8
innodb_background_scrub_data_check_interval	MariaDB 10.1.3
innodb_background_scrub_data_compressed	MariaDB 10.1.3
innodb_background_scrub_data_interval	MariaDB 10.1.3
innodb_background_scrub_data_uncompressed	MariaDB 10.1.3
innodb_buf_dump_status_frequency	MariaDB 10.1.6
innodb_compression_algorithm	MariaDB 10.1.0
innodb_default_encryption_key_id	MariaDB 10.1.4
innodb_default_row_format	MariaDB 10.1.32
innodb_defragment	MariaDB 10.1.1
innodb_defragment_fill_factor	MariaDB 10.1.1

Variable	Added
innodb_defragment_fill_factor_n_recs	MariaDB 10.1.1
innodb_defragment_frequency	MariaDB 10.1.1
innodb_defragment_n_pages	MariaDB 10.1.1
innodb_defragment_stats_accuracy	MariaDB 10.1.1
innodb_disallow_writes	MariaDB 10.1.3
innodb_encrypt_log	MariaDB 10.1.3
innodb_encrypt_tables	MariaDB 10.1.3
innodb_encryption_rotate_key_age	MariaDB 10.1.3
innodb_encryption_rotation_iops	MariaDB 10.1.3
innodb_encryption_threads	MariaDB 10.1.3
innodb_fatal_semaphore_wait_threshold	MariaDB 10.1.2
innodb_force_primary_key	MariaDB 10.1.0
innodb_idle_flush_pct	MariaDB 10.1.2
innodb_immediate_scrub_data_uncompressed	MariaDB 10.1.3
innodb_instrument_semaphores	MariaDB 10.1.3
innodb_mtflush_threads	MariaDB 10.1.0
innodb_prefix_index_cluster_optimization	MariaDB 10.1.2
innodb_scrub_log	MariaDB 10.1.3
innodb_scrub_log_speed	MariaDB 10.1.4
innodb_use_mtflush	MariaDB 10.1.0
innodb_use_trim	MariaDB 10.1.0
log_bin_basename	MariaDB 10.1.6
log_bin_index	MariaDB 10.1.6
log_tc_size	MariaDB 10.1.3
max_digest_length	MariaDB 10.1.5
max_session_mem_used	MariaDB 10.1.21
max_statement_time	MariaDB 10.1.1
mysql56_temporal_format	MariaDB 10.1.2
relay_log_basename	MariaDB 10.1.6
strict_password_validation	MariaDB 10.1.2
skip_parallel_replication	MariaDB 10.1.3
slave_parallel_mode	MariaDB 10.1.3
slave_run_triggers_for_rbr	MariaDB 10.1.1
version_ssl_library	MariaDB 10.1.3
wsrep_certification_rules	MariaDB 10.1.38
wsrep_dirty_reads	MariaDB 10.1.3

See Also

- Status Variables Added in MariaDB 10.1
- System Variables Added in MariaDB 10.2

- System Variables Added in MariaDB 10.0

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 10.1

This is a list of status variables that were added in the MariaDB 10.1 series.

The list excludes status related to the following storage engines included in MariaDB 10.1:

- Galera Status Variables

Variable	Added
Acl_column_grants	MariaDB 10.1.4
Acl_database_grants	MariaDB 10.1.4
Acl_function_grants	MariaDB 10.1.4
Acl_procedure_grants	MariaDB 10.1.4
Acl_proxy_users	MariaDB 10.1.4
Acl_role_grants	MariaDB 10.1.4
Acl_roles	MariaDB 10.1.4
Acl_table_grants	MariaDB 10.1.4
Acl_users	MariaDB 10.1.4
Master_gtid_wait_count	MariaDB 10.1.4
Com_compound_sql	MariaDB 10.1.1
Com_create_temporary_table	MariaDB 10.1.6
Com_create_drop_table	MariaDB 10.1.6
Com_show_generic	MariaDB 10.1.1
Innodb_defragment_compression_failures	MariaDB 10.1.1
Innodb_defragment_count	MariaDB 10.1.1
Innodb_defragment_failures	MariaDB 10.1.1
Innodb_encryption_rotation_estimated_iops	MariaDB 10.1.3
Innodb_encryption_rotation_pages_flushed	MariaDB 10.1.3
Innodb_encryption_rotation_pages_modified	MariaDB 10.1.3
Innodb_encryption_rotation_pages_read_from_cache	MariaDB 10.1.3
Innodb_encryption_rotation_pages_read_from_disk	MariaDB 10.1.3
Innodb_have_bzip2	MariaDB 10.1.0
Innodb_have_lz4	MariaDB 10.1.0
Innodb_have_lzma	MariaDB 10.1.0
Innodb_have_lzo	MariaDB 10.1.0
Innodb_have_snappy	MariaDB 10.1.3
Innodb_num_index_pages_written	MariaDB 10.1.0
Innodb_num_non_index_pages_written	MariaDB 10.1.0
Innodb_onlineddl_pct_progress	MariaDB 10.1.1
Innodb_onlineddl_rowlog_pct_used	MariaDB 10.1.1
Innodb_onlineddl_rowlog_rows	MariaDB 10.1.1

Variable	Added
Innodb_num_page_compressed_trim_op	MariaDB 10.1.0
Innodb_num_page_compressed_trim_op_saved	MariaDB 10.1.0
Innodb_num_pages_page_compressed	MariaDB 10.1.0
Innodb_num_pages_page_compression_error	MariaDB 10.1
Innodb_num_pages_page_decompressed	MariaDB 10.1.0
Innodb_num_pages_page_decrypted	MariaDB 10.1.3
Innodb_num_pages_page_encrypted	MariaDB 10.1.3
Innodb_page_compression_saved	MariaDB 10.1.0
Innodb_page_compression_trim_sect512	MariaDB 10.1.0
Innodb_page_compression_trim_sect1024	MariaDB 10.1.2
Innodb_page_compression_trim_sect2048	MariaDB 10.1.2
Innodb_page_compression_trim_sect4096	MariaDB 10.1.0
Innodb_page_compression_trim_sect8192	MariaDB 10.1.2
Innodb_page_compression_trim_sect16384	MariaDB 10.1.2
Innodb_page_compression_trim_sect32768	MariaDB 10.1.2
Innodb_secondary_index_triggered_cluster_reads	MariaDB 10.1.2
Innodb_secondary_index_triggered_cluster_reads_avoided	MariaDB 10.1.2
Master_gtid_wait_count	MariaDB 10.1.4
Master_gtid_wait_time	MariaDB 10.1.4
Master_gtid_wait_timeouts	MariaDB 10.1.4
Max_statement_time_exceeded	MariaDB 10.1.1

See Also

- System variables added in MariaDB 10.1
- Status variables added in MariaDB 10.2
- Status variables added in MariaDB 10.0

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 10.0

This is a list of system variables that were added in the MariaDB 10.0 series.

The list excludes the following variables, related to storage engines and plugins included in MariaDB 10.0:

- Connect System Variables
- Galera System Variables
- Mroonga System Variables
- Query Response Time Plugin Variables
- Spider System Variables

Variable	Added
aria_pagecache_file_hash_size	MariaDB 10.0.13
binlog_commit_wait_count	MariaDB 10.0.5
binlog_commit_wait_usec	MariaDB 10.0.5

Variable	Added
default_master_connection	MariaDB 10.0.1
default_regex_flags	MariaDB 10.0.11
gtid_binlog_pos	MariaDB 10.0.3
gtid_binlog_state	MariaDB 10.0.5
gtid_current_pos	MariaDB 10.0.3
gtid_domain_id	MariaDB 10.0.2
gtid_ignore_duplicates	MariaDB 10.0.10
gtid_seq_no	MariaDB 10.0.2
gtid_slave_pos	MariaDB 10.0.3
gtid_strict_mode	MariaDB 10.0.3
histogram_size	MariaDB 10.0.2
histogram_type	MariaDB 10.0.2
innodb_adaptive_flushing_lwm	MariaDB 10.0
innodb_adaptive_max_sleep_delay	MariaDB 10.0
innodb_api_bk_commit_interval	MariaDB 10.0
innodb_api_disable_rowlock	MariaDB 10.0
innodb_api_enable_binlog	MariaDB 10.0
innodb_api_enable_mdl	MariaDB 10.0
innodb_api_trx_level	MariaDB 10.0
innodb_buffer_pool_dump_at_shutdown	MariaDB 10.0
innodb_buffer_pool_dump_now	MariaDB 10.0
innodb_buffer_pool_filename	MariaDB 10.0
innodb_buffer_pool_load_abort	MariaDB 10.0
innodb_buffer_pool_load_at_startup	MariaDB 10.0
innodb_buffer_pool_load_now	MariaDB 10.0
innodb_change_buffer_max_size	MariaDB 10.0
innodb_checksum_algorithm	MariaDB 10.0
innodb_cleaner_lsn_age_factor	MariaDB 10.0.9
innodb_cmp_per_index_enabled	MariaDB 10.0
innodb_compression_failure_threshold_pct	MariaDB 10.0
innodb_compression_level	MariaDB 10.0
innodb_compression_pad_pct_max	MariaDB 10.0
innodb_disable_sort_file_cache	MariaDB 10.0
innodb_empty_free_list_algorithm	MariaDB 10.0.9
innodb_flush_neighbors	MariaDB 10.0
innodb_foreground_preflush	MariaDB 10.0.9
innodb_ft_aux_table	MariaDB 10.0
innodb_ft_cache_size	MariaDB 10.0
innodb_ft_enable_diag_print	MariaDB 10.0

Variable	Added
innodb_ft_enable_stopword	MariaDB 10.0
innodb_ft_max_token_size	MariaDB 10.0
innodb_ft_min_token_size	MariaDB 10.0
innodb_ft_num_word_optimize	MariaDB 10.0
innodb_ft_result_cache_limit	MariaDB 10.0.9
innodb_ft_server_stopword_table	MariaDB 10.0
innodb_ft_sort_pll_degree	MariaDB 10.0
innodb_ft_total_cache_size	MariaDB 10.0.9
innodb_ft_user_stopword_table	MariaDB 10.0
innodb_io_capacity_max	MariaDB 10.0
innodb_log_arch_dir	MariaDB 10.0.9
innodb_log_arch_expire_sec	MariaDB 10.0.9
innodb_log_archive	MariaDB 10.0.9
innodb_log_compressed_pages	MariaDB 10.0.9
innodb_max_dirty_pages_pct_lwm	MariaDB 10.0
innodb_max_purge_lag_delay	MariaDB 10.0
innodb_monitor_disable	MariaDB 10.0
innodb_monitor_enable	MariaDB 10.0
innodb_monitor_reset	MariaDB 10.0
innodb_monitor_reset_all	MariaDB 10.0
innodb_online_alter_log_max_size	MariaDB 10.0
innodb_optimize_fulltext_only	MariaDB 10.0
innodb_random_read_ahead	MariaDB 10.0
innodb_read_only	MariaDB 10.0
innodb_recovery_stats	MariaDB 10.0
innodb_sched_priority_cleaner	MariaDB 10.0.9
innodb_simulate_comp_failures	MariaDB 10.0.14
innodb_sort_buffer_size	MariaDB 10.0
innodb_stats_auto_recalc	MariaDB 10.0
innodb_stats_modified_counter	MariaDB 10.0.15
innodb_stats_persistent	MariaDB 10.0
innodb_stats_persistent_sample_pages	MariaDB 10.0
innodb_stats_traditional	MariaDB 10.0.16
innodb_stats_transient_sample_pages	MariaDB 10.0
innodb_status_output	MariaDB 10.0.14
innodb_status_output_locks	MariaDB 10.0.14
innodb_sync_array_size	MariaDB 10.0
innodb_undo_directory	MariaDB 10.0
innodb_undo_logs	MariaDB 10.0

Variable	Added
innodb_undo_tablespaces	MariaDB 10.0
key_cache_file_hash_size	MariaDB 10.0.13
last_gtid	MariaDB 10.0.9
oqgraph_allow_create_integer_latch	MariaDB 10.0.7
performance_schema_accounts_size	MariaDB 10.0.0
performance_schema_digests_size	MariaDB 10.0.0
performance_schema_events_stages_history_long_size	MariaDB 10.0.0
performance_schema_events_stages_history_size	MariaDB 10.0.0
performance_schema_events_statements_history_long_size	MariaDB 10.0.0
performance_schema_events_statements_history_size	MariaDB 10.0.0
performance_schema_hosts_size	MariaDB 10.0.0
performance_schema_max_digest_length	MariaDB 10.0.21
performance_schema_max_socket_classes	MariaDB 10.0.0
performance_schema_max_socket_instances	MariaDB 10.0.0
performance_schema_max_stage_classes	MariaDB 10.0.0
performance_schema_max_statement_classes	MariaDB 10.0.0
performance_schema_session_connect_attrs_size	MariaDB 10.0.4
performance_schema_setup_actors_size	MariaDB 10.0.0
performance_schema_setup_objects_size	MariaDB 10.0.0
performance_schema_users_size	MariaDB 10.0.0
slave_ddl_exec_mode	MariaDB 10.0.8
slave_domain_parallel_threads	MariaDB 10.0.9
slave_parallel_max_queued	MariaDB 10.0.5
slave_parallel_threads	MariaDB 10.0.5
ssl_crl	MariaDB 10.0.0
ssl_crlpath	MariaDB 10.0.0
use_stat_tables	MariaDB 10.0.1
version_malloc_library	MariaDB 10.0.1

See Also

- Status Variables Added in MariaDB 10.0
- System Variables Added in MariaDB 10.1
- Upgrading from MariaDB 5.5 to MariaDB 10.0

This page is licensed: CC BY-SA / Gnu FDL

Status Variables Added in MariaDB 10.0

This is a list of status variables that were added in the MariaDB 10.0 series.

The list excludes status related to the following storage engines included in MariaDB 10.0:

- Galera Status Variables
- Mroonga Status Variables

- Spider Status Variables

Variable	Added
Binlog_group_commit_trigger_count	MariaDB 10.0.18
Binlog_group_commit_trigger_timeout	MariaDB 10.0.18
Binlog_group_commit_trigger_lock_wait	MariaDB 10.0.18
Com_create_role	MariaDB 10.0.5
Com_drop_role	MariaDB 10.0.5
Com_get_diagnostics	MariaDB 10.0.4
Com_grant_role	MariaDB 10.0.5
Com_revoke_grant	MariaDB 10.0.5
Com_show_explain	MariaDB 10.0.0
Com_start_all_slaves	MariaDB 10.0.0
Com_start_slave	MariaDB 10.0.0
Com_stop_all_slaves	MariaDB 10.0.0
Com_stop_slave	MariaDB 10.0.0
Connection_errors_accept	MariaDB 10.0.4
Connection_errors_internal	MariaDB 10.0.4
Connection_errors_max_connections	MariaDB 10.0.4
Connection_errors_peer_address	MariaDB 10.0.4
Connection_errors_select	MariaDB 10.0.4
Connection_errors_tcpwrap	MariaDB 10.0.4
Delete_scan	MariaDB 10.0.27
Feature_delay_key_write	MariaDB 10.0.13
Handler_external_lock	MariaDB 10.0.0
Handler_read_retry	MariaDB 10.0.27
Innodb_buffer_pool_dump_status	MariaDB 10.0.0
Innodb_buffer_pool_load_status	MariaDB 10.0.0
Innodb_master_thread_active_loops	MariaDB 10.0.9
Innodb_master_thread_idle_loops	MariaDB 10.0.9
Innodb_num_open_files	MariaDB 10.0.0
Innodb_system_rows_deleted	MariaDB 10.0.15
Innodb_system_rows_inserted	MariaDB 10.0.15
Innodb_system_rows_read	MariaDB 10.0.15
Innodb_system_rows_updated	MariaDB 10.0.15
Memory_used	MariaDB 10.0.1
Oqgraph_boost_version	MariaDB 10.0.7
Performance_schema_accounts_lost	MariaDB 10.0.0
Performance_schema_hosts_lost	MariaDB 10.0.0
Performance_schema_socket_classes_lost	MariaDB 10.0.0
Performance_schema_socket_instances_lost	MariaDB 10.0.0

Variable	Added
Performance_schema_stage_classes_lost	MariaDB 10.0.0
Performance_schema_statement_classes_lost	MariaDB 10.0.0
Slave_skipped_errors	MariaDB 10.0.18
Sort_priority_queue_sorts	MariaDB 10.0.13
Update_scan	MariaDB 10.0.27

See also

- System variables added in MariaDB 10.0
- Status variables added in MariaDB 10.1

This page is licensed: CC BY-SA / Gnu FDL

System Variables Added in MariaDB 5.5

This is a list of system variables that were added in the MariaDB 5.5 series.

The list excludes variables related to non-default storage engines and plugins that can be added to MariaDB 5.5:

Variable	Added
innodb_adaptive_flushing_method	MariaDB 5.5.20
innodb_adaptive_hash_index_partitions	MariaDB 5.5.20
innodb_blocking_buffer_pool_restore	MariaDB 5.5.20
innodb_buffer_pool_instances	MariaDB 5.5.20
nnodb_buffer_pool_restore_at_startup	MariaDB 5.5.20
innodb_change_buffering_debug	MariaDB 5.5.20
innodb_corrupt_table_action	MariaDB 5.5.20
innodb_flush_checkpoint_debug	
innodb_force_load_corrupted	MariaDB 5.5.20
innodb_import_table_from_xtrabackup	MariaDB 5.5.20
innodb_large_prefix	MariaDB 5.5.20
innodb_purge_batch_size	MariaDB 5.5.20
innodb_purge_threads	MariaDB 5.5.20
innodb_recovery_update_relay_log	MariaDB 5.5.20
innodb_rollback_segments	MariaDB 5.5.20
innodb_use_global_flush_log_at_trx_commit	MariaDB 5.5.20
innodb_use_native_aio	MariaDB 5.5.20

See also

- System Variables Added in MariaDB 10.0
- Upgrading from MariaDB 5.3 to MariaDB 5.5

This page is licensed: CC BY-SA / Gnu FDL

Replication

See Replication Overview.

Explore the core concepts of MariaDB standard replication. Learn about the primary-replica architecture, data redundancy strategies, and how to scale read operations effectively.

See Choosing a Replication Strategy.

Choose a MariaDB replication strategy by matching the replication method and format to your goal — reducing downtime, scaling reads, or aggregating data — and weigh the trade-offs.

See Binary Log Formats.

Compare the three binary logging formats — statement-based, row-based, and mixed — including their trade-offs and how to select one with `binlog_format`.

See InnoDB-Based Binary Log.

From MariaDB 12.3, the binary log can be stored in InnoDB-managed, page-structured files integrated with InnoDB crash recovery, instead of traditional flat binary log files.

See Replication Statements.

Access the complete reference of SQL statements used to manage replication. This guide covers commands for controlling primaries, configuring replicas, and monitoring status.

See Setting Up Replication.

Complete guide to MariaDB replication setup. Complete walkthrough for primary-replica topology with binary logging and GTID configuration.

See Global Transaction ID.

Complete GTID replication reference: CHANGE MASTER master_use_gtid=current_pos|slave_pos, gtid_slave_pos table (InnoDB), START REPLICA UNTIL master_gtid_pos.

See Read-Only Replicas.

Learn how to configure replicas as read-only instances. This ensures data integrity by preventing accidental writes on the replica while allowing it to process replication events.

See Multi-Source Replication.

Discover how to replicate data from multiple primaries to a single replica. This guide covers the configuration for aggregating data from different sources into one MariaDB instance.

See Multi-Master Ring Replication.

Explore the ring topology where each server acts as both primary and replica. Learn the configuration steps and caveats for setting up a circular replication environment.

See Delayed Replication.

Learn to configure a time lag for your replica. Delayed replication helps recover from human errors on the primary, such as accidental drop commands, by preserving older states.

See Parallel Replication.

Boost MariaDB Server replication performance with parallel replication. This section explains how to configure replicas to apply events concurrently, reducing lag and improving throughput.

See Semisynchronous Replication.

Enhance data consistency with semisynchronous replication. Ensure that the primary waits for at least one replica to acknowledge receipt of a transaction before committing.

See Row-based Replication With No Primary Key.

Understand the performance implications and best practices for replicating tables without primary keys when using row-based logging, including how to avoid full table scans.

See Unsafe Statements for Statement-Based Replication.

Identify SQL statements that are non-deterministic and unsafe for statement-based replication. Learn why these queries cause divergence and how to switch to row-based logging.

See Replication Filters.

Learn how to filter specific databases or tables from being replicated. This guide covers configuration options to replicate only the data you need on specific replicas.

See Replication Threads.

Gain insight into the background threads that drive replication. Understand the roles of the I/O thread, SQL thread, and binlog dump thread in moving data between servers.

See Replication and Foreign Keys.

Understand how foreign key constraints interact with replication. Learn best practices for managing cascading deletes and updates across primary and replica servers.

See Replication and Binary Log System Variables.

Complete Replication and Binary Log System Variables reference for MariaDB. Complete guide for configuration values, scope settings, and performance impact.

See Replication and Binary Log Status Variables.

View the status variables used to monitor replication health. Learn how to interpret metrics regarding log positions, connection status, and event counts.

See Binlog Event Checksum Interoperability.

Learn about checksum compatibility between different MariaDB versions. Ensure older replicas can correctly verify binary log events generated by newer primaries.

See Binlog Event Checksums.

Configure checksums to detect data corruption in binary logs. Learn how to enable verification to ensure the integrity of replication events during transmission and storage.

See Changing a Replica to Become the Primary.

Follow the procedure to promote a replica to a primary role. This guide details the steps for planned failovers or topology reorganization with minimal downtime.

See Replication When the Primary and Replica Have Different Table Definitions.

Understand the rules and limitations when replicating between tables with differing structures. Learn how attribute promotion and column handling work in row-based replication.

See START TRANSACTION … WITH CONSISTENT SNAPSHOT.

Learn how to start a consistent transaction for backups or replication setup. This command ensures a consistent view of the database without locking tables unnecessarily.

See Restricting Speed of Reading Binlog from Primary by a Replica.

Configure throughput limits for replication traffic. Learn to throttle the binlog download speed to prevent replication from consuming all available network bandwidth.

See Running Triggers on the Replica for Row-based Events.

Understand how triggers behave under row-based replication. Learn when and why triggers are not executed on the replica and how to manage complex logic in this mode.

See Selectively Skipping Replication of Binlog Events.

Learn techniques to bypass specific replication events. This guide explains how to ignore individual transactions or errors to restore replication flow after a stoppage.

This page is licensed: CC BY-SA / Gnu FDL

Replication Overview

> **Note:**
>
> In MariaDB 11.8, the default character set and collation have changed. This has implications on replicating to older replicas, particularly replicas running MariaDB 10.6 or older.
>
> See this section for details, and how to configure MariaDB 11.8+ primaries to replicate to older replicas.

Overview

Replication is a feature allowing the contents of one or more servers (called primaries) to be mirrored on one or more servers (called replicas).

You can exert control over which data to replicate. All databases, one or more databases, or tables within a database can each be selectively replicated.

The main mechanism used in replication is the binary log. If binary logging is enabled, all updates to the database (data manipulation and data definition) are written into the binary log as binlog events. Replicas read the binary log from each primary in order to access the data to replicate. A relay log is created on the replica, using the same format as the binary log, and this is used to perform the replication. Old relay log files are removed when no longer needed.

A replica server keeps track of the position in the primary's binlog of the last event applied on the replica. This allows the replica server to re-connect and resume from where it left off after replication has been temporarily stopped. It also allows a replica to disconnect, be cloned and then have the new replica resume replication from the same primary.

Primaries and replicas do not need to be in constant communication with each other. It's quite possible to take servers offline or disconnect from the network, and when they come back, replication will continue where it left off.

Replication Formats

There are three kinds of replication format – essentially, they're binary log formats, and therefore documented on this page: Binary Log Formats:

- Statement-based replication (SBR)
- Row-based replication (RBR)
- Mixed replication

Replication Uses

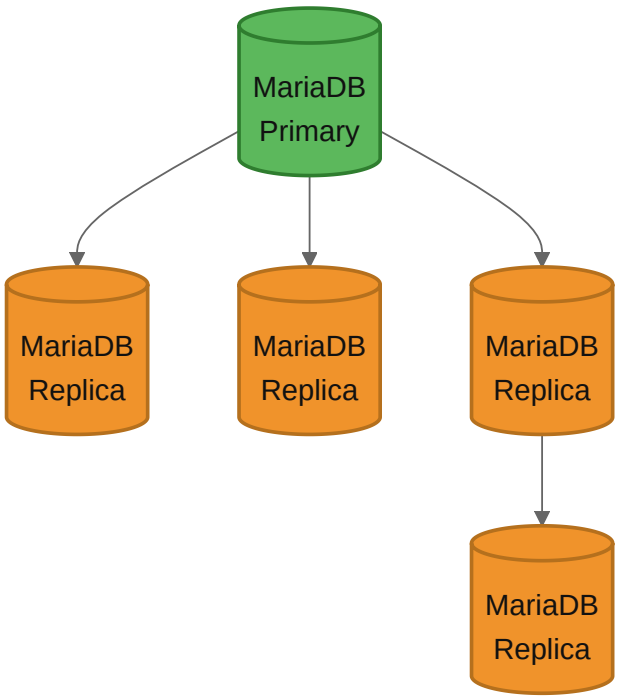
Replication is used in a number of common scenarios. Uses include:

- Scalability. By having one or more replicas, reads can be spread over multiple servers, reducing the load on the primary. The most common scenario for a high-read, low-write environment is to have one primary, where all the writes occur, replicating to multiple replicas, which handle most of the reads.

- Data analysis. Analyzing data may have too much of an impact on a primary server, and this can similarly be handled on a replica, while the primary continues unaffected by the extra load.
- Backup assistance. Backups can more easily be run if a server is not actively changing the data. A common scenario is to replicate the data to a replica, which is then disconnected from the primary with the data in a stable state. Backup is then performed from this server. See Replication as a Backup Solution.
- Distribution of data. Instead of being connected to a remote primary, it's possible to replicate the data locally and work from this data instead.

Common Replication Setups

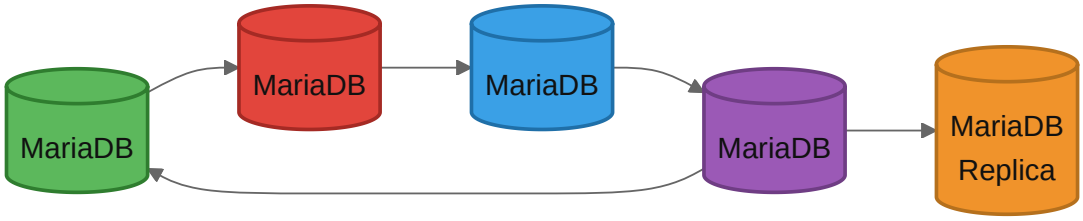
Standard Replication



Standard replication: one primary fans out to multiple replicas; a replica can chain to a further downstream replica.

- Provides infinite read scale out.
- Provides high-availability by upgrading replica to primary.
- Setting up standard replication

Ring Replication

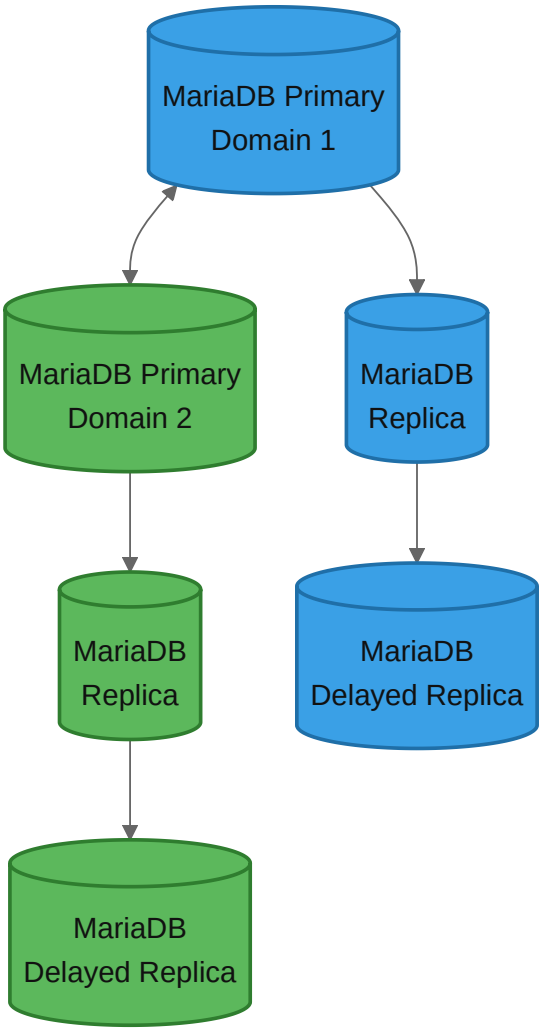


Ring replication: each primary replicates to the next in a closed loop; here one node also feeds a replica.

- Provides read and write scaling.
- Doesn't handle conflicts.

- If one primary fails, replication stops.
- More about Multi-master ring replication

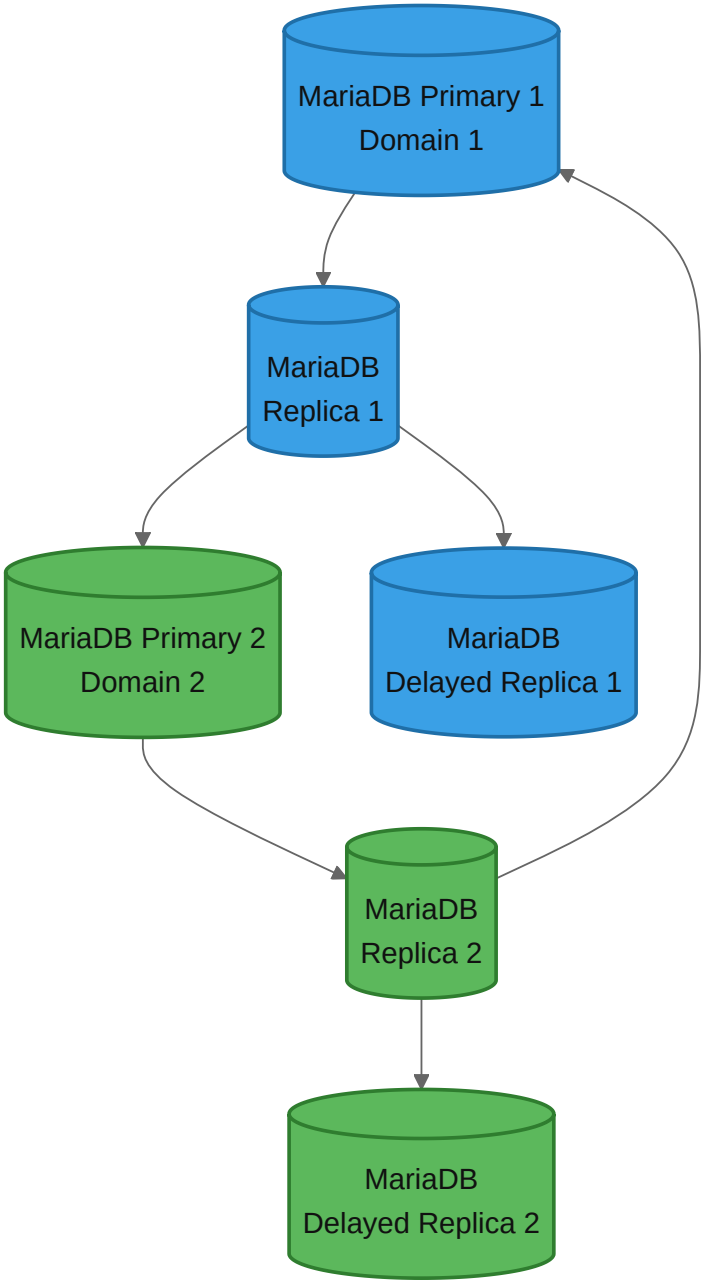
Ring Replication with slaves



Multi-master ring with replicas: two primaries replicate to each other; each also has a replica and a delayed replica.

- Provides read and write scaling.
- Doesn't handle conflicts.
- If one primary fails, replication stops.
- More about Multi-master ring replication

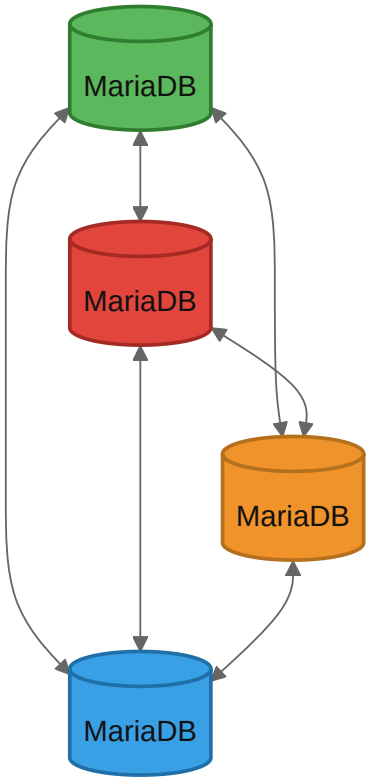
Ring Replication with replication through slaves



Multi-master ring relayed through replicas: each replica forwards to the other domain's primary, closing the ring.

- Provides read and write scaling.
- Doesn't handle conflicts.
- If one primary fails, replication stops.
- More about Multi-master ring replication

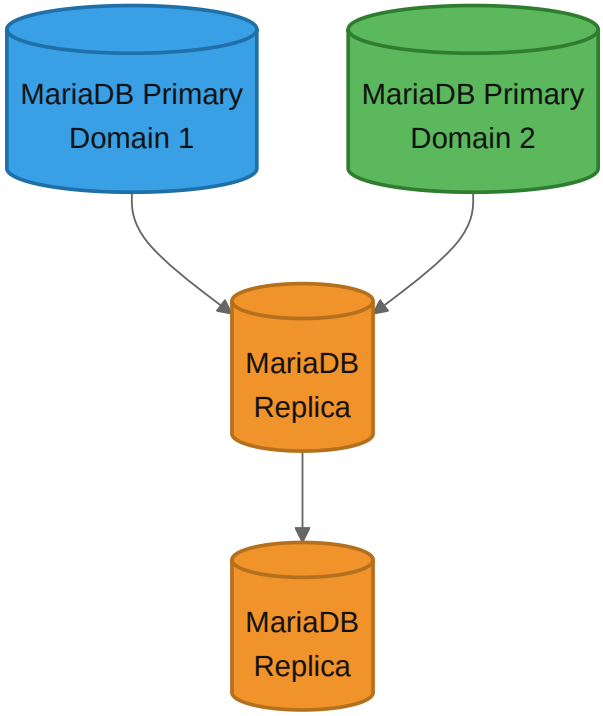
Star Replication



Star replication: every primary replicates with every other, so all nodes converge to the same data.

- Provides read and write scaling.
- Doesn't handle conflicts.
- Have to use replication filters to avoid duplication of data.
- MariaDB Galera Cluster, which is a virtually synchronous multi-primary (multi-master) cluster for MariaDB, has a similar configuration and can handle conflicts.

Multi-Source Replication



Multi-source replication: one replica pulls from two primaries on separate domains and applies them in parallel.

- Allows you to combine data from different sources.
- Different domains executed independently in parallel on all replicas.
- More about Multi-Source replication

Cross-Version Replication Compatibility

The following table describes replication compatibility between different MariaDB Community Server versions. In general, the replica should be of the same or a later version than the primary. The constraint also applies to minor/patch releases:

Replica ↓ / Primary →	**MariaDB 10.4**	**MariaDB 10.5**	**MariaDB 10.6**	**MariaDB 10.11**	**MariaDB 11.4**	**MariaDB 11.8**	**MariaDB 12.3**	**MariaDB 13.0**
MariaDB 10.4	✅	⛔	⛔	⛔	⛔	⛔	⛔	⛔
MariaDB 10.5	✅	✅	⛔	⛔	⛔	⛔	⛔	⛔
MariaDB 10.6	✅	✅	✅	⛔	⛔	⛔	⛔	⛔
MariaDB 10.11	✅	✅	✅	✅	⛔	⛔	⛔	⛔
MariaDB 11.4	✅	✅	✅	✅	✅	⛔	⛔	⛔
MariaDB 11.8	✅	✅	✅	✅	✅	✅	⛔	⛔
MariaDB 12.3	✅	✅	✅	✅	✅	✅	✅	⛔
MariaDB 13.0	✅	✅	✅	✅	✅	✅	✅	✅

- ✅: This combination is supported.
- ⛔: This combination is not supported.

Note: where it is not officially supported to replicate to a server with a lesser minor version, replication can still be safe for:

- DMLs logged in ROW binlog_format, and
- DMLS logged in STATEMENT format and DDLs where neither use features that do not yet exist on the replica

provided the configurations for each server allow for consistent behavior in the execution of the events (i.e. the execution of the event should not be reliant on newer configuration variables, character sets/collations, etc, that don't exist on the replica). Additionally note, if binlog_format=MIXED, it may be possible that the higher-versioned server (primary) may consider it safe to log a transaction using STATEMENT binlog format, while the older-versioned replica categorizes it as unsafe, which will result in an error while the replica tries to execute the transaction. See this page for more details on unsafe statements.

The table shows the general version constraint only; it does not mean that every listed combination is separately tested. Individual releases can introduce changes that affect replication — for example, new data types, changed defaults, or changes to how statements are written to the binary log. Before setting up replication between different versions, review the release notes (the "Changes & Improvements" page) for the versions involved. Any change that affects cross-version replication compatibility is called out there.

> **Note:**
>
> **MariaDB Enterprise Server:** The table above applies to MariaDB Community Server. MariaDB Enterprise Server releases are based on a Community Server version with additional backported features, so cross-version replication combinations are not separately tested for Enterprise Server. The same general rule applies — a replica should run the same or a later version than its primary — but before replicating between different Enterprise Server versions, or between Enterprise Server and Community Server, review the MariaDB Enterprise Server release notes for the versions involved for any changes that affect replication.

For replication compatibility details between MariaDB and MySQL, see MariaDB versus MySQL - Compatibility: Replication Compatibility.

See Also

- Setting Up Replication
- Replication Compatibility Between MariaDB and MySQL
- MariaDB Galera Cluster and M/S replication (video)

This page is licensed: CC BY-SA / Gnu FDL

Choosing a Replication Strategy

Choosing the right replication strategy in MariaDB is less about the underlying technology itself and more about which specific "pain point" your architecture needs to solve: preventing downtime, accelerating slow reads, securing data across distances, or aggregating data for analytics.

This guide helps you determine which replication method and format to choose for your cluster, and what trade-offs you will make.

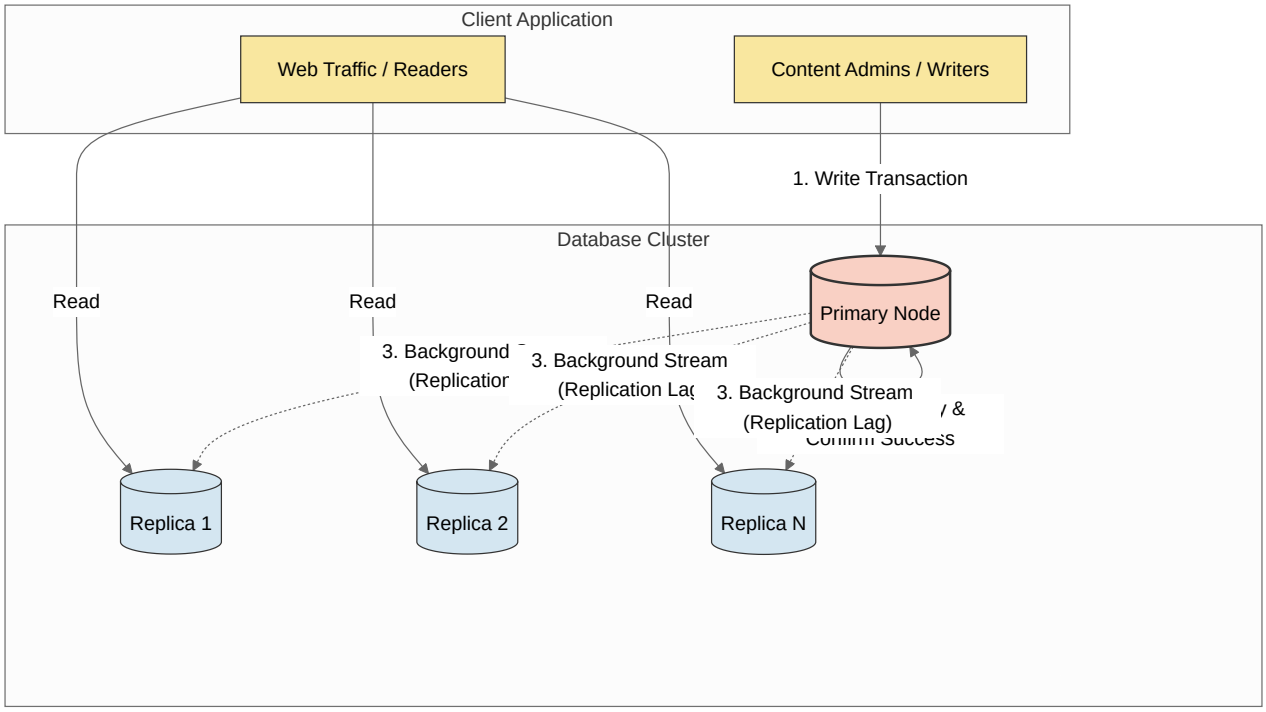
Overview of Replication Methods

Before mapping out specific use cases, it is important to understand the four primary replication protocols available in the MariaDB ecosystem:

Asynchronous Replication (Primary-Replica)

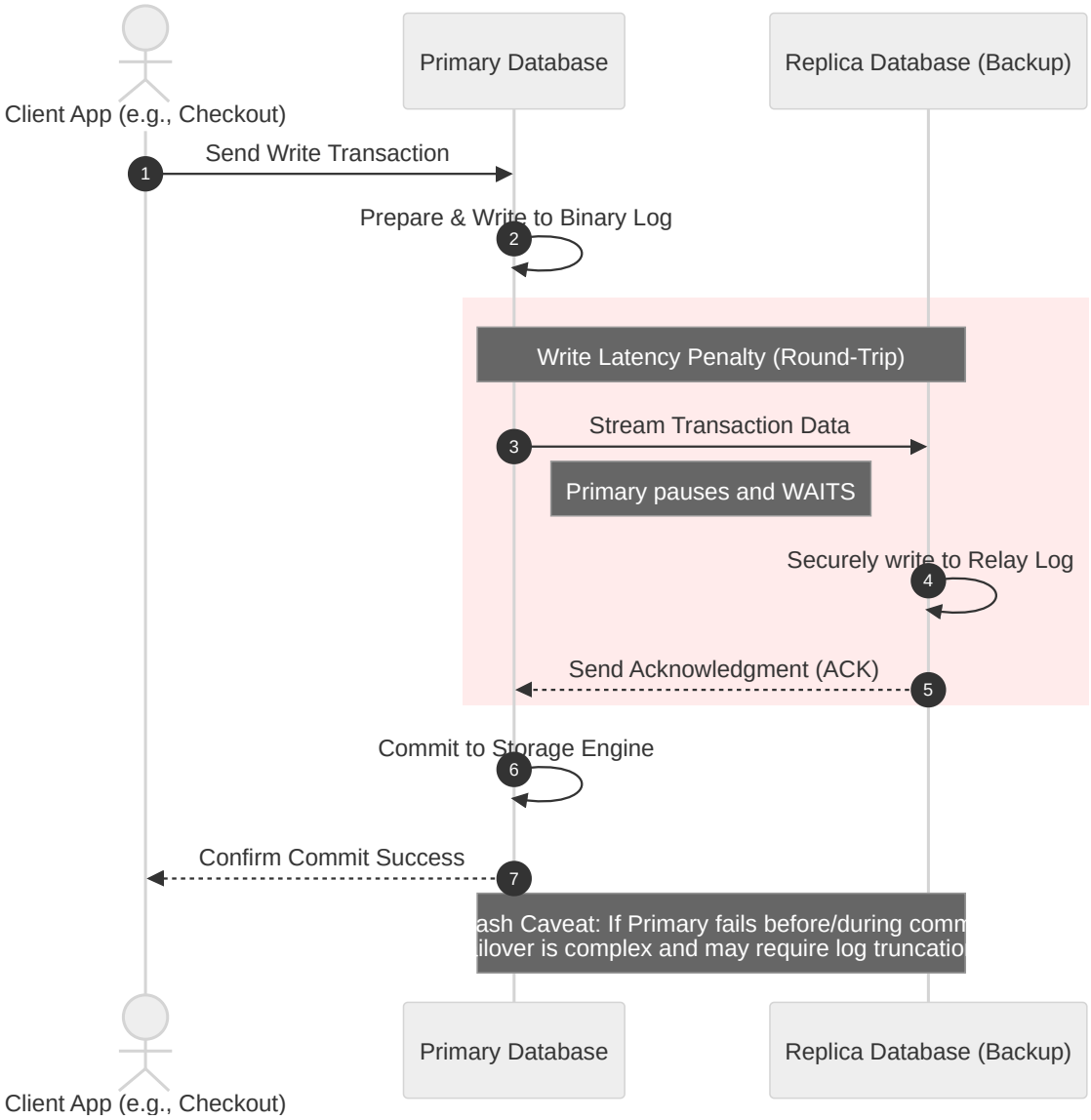
The standard method where the primary node commits a transaction locally and streams it to replicas in the background. This method is ideal for read-heavy applications, such as a high-traffic news website where a single primary database handles content updates while multiple replicas serve read-only web traffic to millions of visitors. While a momentary delay before a new article appears on all servers is an acceptable trade-off for massive read

scalability, this inherent replication lag means users might occasionally read stale data. Furthermore, if the primary server crashes before the background stream catches up, the most recent updates will be permanently lost.



Semisynchronous Replication

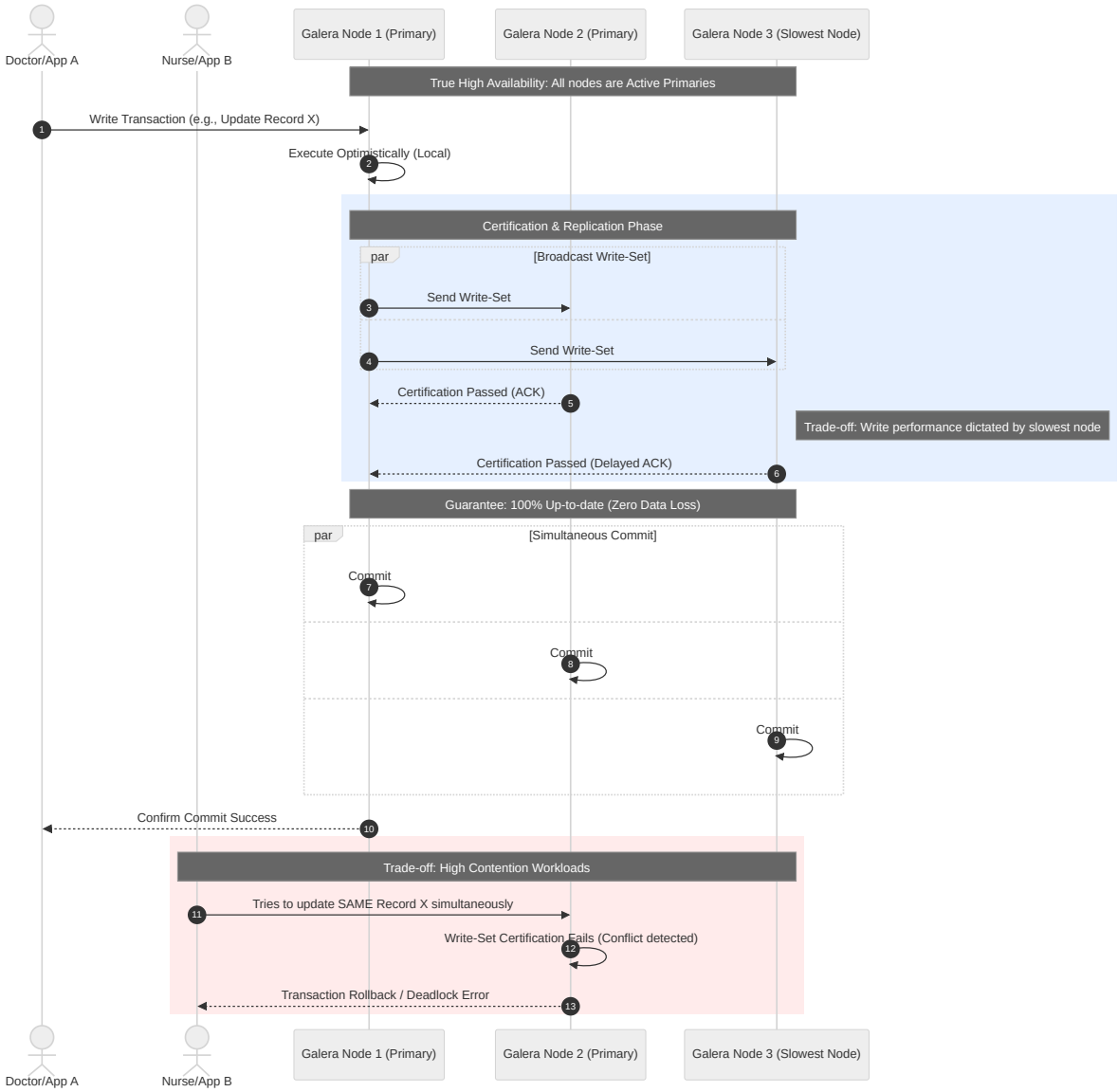
A middle ground where the primary waits for at least one replica to acknowledge receipt of the data before confirming a commit. This setup is well-suited for systems like an e-commerce checkout spanning two nearby data centers, where ensuring at least one backup server has securely logged a transaction prevents a “lost order” if the primary loses power.



However, this increased data integrity comes with a strict write latency penalty while the primary waits for the round-trip acknowledgment from the replica. Additionally, if the primary crashes, failover can be complex and might still involve data loss or require truncating the binary log, depending on the server's specific wait configuration.

Virtually Synchronous Replication (Galera Cluster)

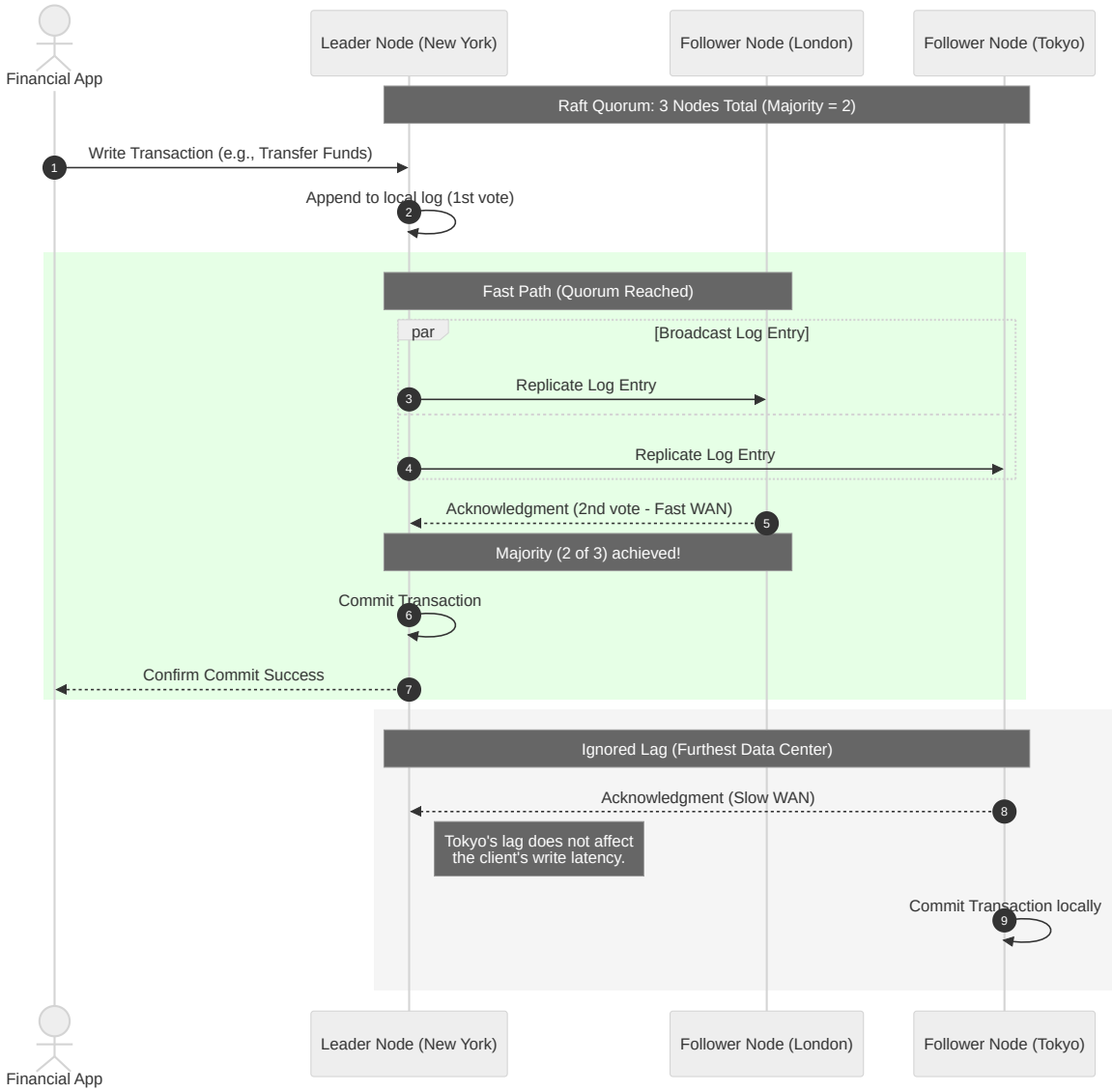
A multi-primary solution where all active nodes must receive and accept a transaction before it commits, guaranteeing zero data loss and true high availability. This is the "gold standard" for environments like a critical healthcare records system hosted within a single enterprise data center, where every updated record is applied simultaneously across all nodes so users always see 100% up-to-date information.



The trade-off for this consistency is that write performance is entirely dictated by the slowest node in the cluster. Additionally, high-contention workloads with multiple clients writing to the same rows on different nodes can result in frequent transaction rollbacks or cluster deadlocks.

Quorum Replication (MariaDB Enterprise Cluster)

A Raft-based solution that requires acknowledgment from a *majority* of nodes (a quorum) rather than all of them, providing robust fault tolerance with faster write performance across wide-area networks.



For example, a globally distributed financial application spanning data centers in New York, London, and Tokyo only needs agreement from two of the three regions to commit data, keeping the application fast by effectively ignoring the network lag of the furthest data center.

Replication Methods Comparison Matrix

Replication Method	Write Performance	Data Safety (Node Crash)	Automatic Failover	Ideal Architectural Fit
Asynchronous	Fastest (No waiting for replicas)	High Risk (Potential data loss if primary crashes)	No (Requires external tools like MaxScale)	Read-heavy applications and distant Disaster Recovery.
Semisynchronous	Fast (Waits for only 1 replica)	Low Risk (Data is backed up before commit)	No (Requires external tools like MaxScale)	Environments needing strong data safety without full clustering.
Virtually Sync (Galera)	Slowest (Dictated by the slowest node)	Zero Data Loss (All nodes are identical)	Yes (Built-in cluster membership)	Local High Availability (Single Data Center).
Quorum (Raft)	Fast / Moderate(Dictated by the majority)	Zero Data Loss(Majority quorum agreed)	Yes (Automated leader election)	Geo-Distributed High Availability (Multi-Data Center).

Choosing a Strategy by Use Case

Identify your primary architectural challenge below to discover the best-fit replication topology.

High Availability (HA) & Fault Tolerance

> **Note:**
>
> Zero data loss and automatic failover. If one server dies, another takes over instantly without impacting the application.

- Galera Cluster (Virtually Synchronous): The “gold standard” for local high availability. It is a synchronous, multi-primary solution where every node has the exact same data at the same time.
  - *Trade-off:* Write latency is dictated by the slowest node, as all active nodes must acknowledge the transaction.
- MariaDB Enterprise Cluster (Quorum/Raft): *(Enterprise Technical Preview)* Designed for environments that need HA but cannot afford Galera’s write latency. It requires acknowledgment from only a *majority* (quorum) of nodes to commit a write, effectively ignoring network lag from the slowest data centers.
- Semisynchronous Replication: A middle ground where the primary server waits for at least one replica to acknowledge receipt of the data before confirming a “success” to the client. It prevents data loss during a crash without the full performance overhead of Galera.

Read Scaling & Performance

> **Note:**
>
> The application has thousands of users browsing (reading) but only a few writing, and you need to offload the read workload to maintain performance.

- Primary-Replica (Asynchronous): One “Primary” handles all data changes, while changes are streamed in the background to multiple “Replicas” that handle the read queries. This offers the highest performance for scaling high-traffic apps.
  - Trade-off: You risk momentary replication lag and potential data loss if the primary crashes before streaming its latest updates.

Geographic Distribution & Disaster Recovery

> **Note:**
>
> You need a “Plan B” in a different geographic region to survive a total data center outage, or you need a safeguard against catastrophic human error.

- Hybrid Replication (Geo-DR): Combines methods for the best of both worlds. A synchronous Galera Cluster is used locally for High Availability, while standard asynchronous replication streams data to a distant Disaster Recovery (DR) replica over a WAN.
- Delayed Replication (Human Error DR): A replica that intentionally stays a set amount of time (e.g., one hour) behind the primary. If a user accidentally runs a destructive command like `DROP TABLE`, you can recover the lost data from the delayed replica before the mistake replicates across your infrastructure.

Data Aggregation & Analytics

> **Note:**
>
> To pull data from many different live production databases into one centralized location for reporting, without slowing down your production apps.

- Multi-Source Replication: One replica is configured to receive data streams from several different primary servers concurrently. This is perfect for populating a centralized Data Warehouse where you can run heavy analytical queries entirely isolated from your live production environments.

The Role of MariaDB MaxScale

Managing replication topologies manually can be highly complex. MariaDB MaxScale is an advanced database proxy that sits between your application and your database cluster to automate traffic routing and failover across any replication method:

- In Asynchronous / Semisynchronous Setups: MaxScale acts as a traffic cop, automatically routing `SELECT` queries to your replicas and `INSERT` / `UPDATE` queries to your primary. If the primary fails, MaxScale can automatically promote a replica and reroute traffic without application downtime.
- In Galera Cluster Setups: MaxScale transparently masks node failures or evictions by routing traffic only to healthy nodes. It can also perform intelligent read/write splitting across the cluster to optimize overall throughput.
- In MariaDB Enterprise Cluster Setups: MaxScale automatically detects the current Raft leader to ensure writes are routed appropriately, while load balancing read queries across the follower nodes.

> **Note:**
>
> MaxScale is a commercial MariaDB Enterprise product, which should be factored into your architectural decision-making. For technical implementation details, refer to the MaxScale documentation.

Quick Comparison Decision Matrix

Architectural Goal	Recommended Solution	Consistency Type	Key Trade-off / Benefit
No Downtime (Local HA)	Galera Cluster	Synchronous	Guarantees zero data loss, but slower writes.
No Downtime (Geo HA)	MariaDB Enterprise Cluster	Quorum (Raft)	Faster writes across WAN, but potential bottleneck at the leader.
Data Safety (Middle Ground)	Semisynchronous Replication	Semisynchronous	Prevents data loss during crash, but introduces a slight latency penalty.
Read Scaling	Primary-Replica + MaxScale	Asynchronous	Maximum performance, but risks replication lag.
Disaster Recovery (WAN)	Hybrid Replication	Sync (Local) / Async (Remote)	Safely bridges data centers, but setup is complex.
Reporting / BI	Multi-Source Replication	Asynchronous	Safely aggregates data without impacting production.
Human Error Recovery	Delayed Replication	Asynchronous (Delayed)	Saves against accidental `DROP TABLE` executions.

Choosing a Binary Log Format

Once you select your replication strategy, you must configure how data changes are recorded in the MariaDB binary log. For most high-availability architectures, Row-Based Replication (RBR) or Mixed Format is required to ensure absolute data consistency across replicas.

Statement-Based Replication (SBR) should generally be avoided in HA setups due to reliability concerns, acting only as a fallback for specific bulk-update operations.

This page is licensed: CC BY-SA / Gnu FDL

Replication Statements

Replication SQL statements can be found in the Reference section.

This page is licensed: CC BY-SA / Gnu FDL

Setting Up Replication

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Getting replication working involves steps on both the master server/s and steps on the replica server/s.

Setting up a Replica with mariadb-backup

If you want to use mariadb-backup to set up a replication replica, review the information under Setting up a Replication Replica with MariaDB-Backup.

Setting up replication the “traditional” way is covered below.

Versions

In general, when replicating across different versions of MariaDB, it is best that the master is an older version than the slave. MariaDB versions are usually backward compatible, while of course older versions cannot always be forward compatible. See also Replicating from MySQL Master to MariaDB Replica.

Follow these steps to set up MariaDB replication:

Step 1

Configure the Master

- Enable binary logging if it's not already enabled. See Activating the Binary Log and Binary log formats for details.
- Give the master a unique server_id. All slaves must also be given a server_id. This can be a number from 1 to (2^{32})-1, and must be unique for each server in the replicating group.
- Specify a unique name for your replication logs with –log-basename. If this is not specified your host name will be used and there will be problems if the hostname ever changes.
- Slaves will need permission to connect and start replicating from a server. Usually this is done by creating a dedicated slave user, and granting that user permission only to replicate (REPLICATION SLAVE permission).

Example Enabling Replication for MariaDB

Add the following into your my.cnf file and restart the database.

```
[mariadb]
log-bin
server_id=1
log-basename=master1
binlog-format=mixed
```

The server id is a unique number for each MariaDB/MySQL server in your network.binlog-format specifies how your statements are logged. This mainly affects the size of the binary log that is sent between the Master and the Replicas.

Then execute the following SQL with the mysql command line client:

```
CREATE USER 'replication_user'@'%' IDENTIFIED BY 'bigs3cret';
GRANT REPLICATION SLAVE ON *.* TO 'replication_user'@'%';
```

Example Enabling Replication for MySQL

If you want to enable replication from MySQL 5.7 or earlier to MariaDB, you can do it in almost the same way as between MariaDB servers. The main difference is that MySQL doesn't support `log-basename`.

```
[mysqld]
log-bin
server_id=1
```

> **Warning:**
>
> Replication from MySQL 8.0 to MariaDB requires more configuration.

Step 2

Check Settings

There are a number of options that may impact or break replication. Check the following settings to avoid problems.

- skip-networking. If `skip-networking=1`, the server will limit connections to localhost only, and prevent all remote replicas from connecting.
- bind-address. Similarly, if the address the server listens for TCP/IP connections is `127.0.0.1` (localhost), remote replica connections will fail.

Step 3

Configure the Replica

Give the replica a unique server_id. All servers, whether masters or replicas, are given a server_id. This can be a number from `1` to `(2<sup>32</sup>)-1`, and must be unique for each server in the replicating group. The server will need to be restarted in order for a change in this option to take effect.

Step 4

Get the Master's Binary Log Coordinates

Now you need prevent any changes to the data while you view the binary log position. You'll use this to tell the replica at exactly which point it should start replicating from.

- On the master, flush and lock all tables by running `FLUSH TABLES WITH READ LOCK`. Keep this session running - exiting it will release the lock.
- Get the current position in the binary log by running SHOW MASTER STATUS:

```
SHOW MASTER STATUS;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| master1-bin.000096 |      568 |              |                  |
+--------------------+----------+--------------+------------------+
```

- Record the File and Position details. If binary logging has just been enabled, these will be blank.
- Now, with the lock still in place, copy the data from the master to the replica. See Backup, Restore and Import for details on how to do this.
- Note for live databases: You just need to make a local copy of the data, you don't need to keep the master locked until the replica has imported the data.
- Once the data has been copied, you can release the lock on the master by running UNLOCK TABLES.

```
UNLOCK TABLES;
```

Step 5

Start the Replica

- Once the data has been imported, you are ready to start replicating. Begin by running a CHANGE MASTER TO, making sure that `MASTER_LOG_FILE` matches the file and `MASTER_LOG_POS` the position returned by the earlier `SHOW MASTER STATUS`:

```
CHANGE MASTER TO
  MASTER_HOST='master.domain.com',
  MASTER_USER='replication_user',
  MASTER_PASSWORD='bigs3cret',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='master1-bin.000096',
  MASTER_LOG_POS=568,
  MASTER_CONNECT_RETRY=10;
```

If you are starting a replica against a fresh master that was configured for replication from the start, then you don't have to specify `MASTER_LOG_FILE` and `MASTER_LOG_POS`.

Use Global Transaction ID (GTID)

It is generally recommended to use (GTIDs), as it has a number of benefits. All that is needed is to add the `MASTER_USE_GTID` option to the `CHANGE MASTER` statement:

```
CHANGE MASTER TO MASTER_USE_GTID = slave_pos
```

See Global Transaction ID for a full description.

- Now start the replica with the START REPLICA statement:

```
START REPLICA;
```

- Check that the replication is working by executing the SHOW SLAVE STATUS command:

```
SHOW REPLICA STATUS \G
```

- If replication is working correctly, both the values of `Slave_IO_Running` and `Slave_SQL_Running` should be `Yes`:

```
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

Replicating from MySQL Master to MariaDB Replica

- Replicating from MySQL 5.5 to MariaDB should just work. When using a MariaDB as a replica, it may be necessary to set binlog_checksum to `NONE`.
- Replicating from MySQL 5.6 without GTID to MariaDB 10+ should work.
- Replication from MySQL 5.6 with GTID, binlog_rows_query_log_events and ignorable events works. In this case MariaDB will remove the MySQL GTIDs and other unneeded events and instead adds its own GTIDs.

> **Warning:**
>
> Replication from MySQL 8 to MariaDB requires MariaDB 11.4.5 or newer.

See Also

- Differences between Statement-based, mixed and row logging
- Replication and Foreign Keys

- Replication as a Backup Solution
- Multi-source Replication
- Global Transaction ID
- Parallel Replication
- Replication and Binary Log System Variables
- Replication and Binary Log Status Variables
- Semisynchronous Replication
- Delayed Replication

This page is licensed: CC BY-SA / Gnu FDL

Global Transaction ID

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

> **Warning:**
>
> MariaDB and MySQL have different GTID implementations, and they are **not compatible** with each other. MariaDB can be a replica for a MySQL primary, but MySQL cannot be a replica for a MariaDB primary.

Overview

MariaDB replication in general works as follows (see Replication overview for more information):

On a primary server, all updates to the database (DML and DDL) are written into the binary log as binlog events. A replica server connects to the primary and reads the binlog events, then applies the events locally to replicate the same changes as done on the primary. A server can be both a primary and a replica at the same time, and it is thus possible for binlog events to be replicated through multiple levels of servers.

A replica server keeps track of the position in the primary's binlog of the last event applied on the replica. This allows the replica server to re-connect and resume from where it left off after replication has been temporarily stopped. It also allows a replica to disconnect, be cloned and then have the new replica resume replication from the same primary.

Global transaction ID (GTID) introduces a new event attached to each event group in the binlog. An event group is a collection of events that are always applied as a unit. They are best thought of as a "transaction", though they also include non-transactional DML statements, as well as DDL. When an event group is replicated from primary server to replica server, the global transaction ID is preserved. The GTID is globally unique across an entire group of servers, making it easy to uniquely identify the same binlog events on different servers that replicate each other. GTIDs are generated for all event groups, independent of binlog_format (i.e. `ROW`, `STATEMENT`, and `MIXED` formats are all supported).

Benefits

Using global transaction ID provides two main benefits:

1. Easy to change a replica server to connect to and replicate from a different primary server.

The replica remembers the global transaction ID of the last event group applied from the old primary. This makes it easy to know where to resume replication on the new primary, since the global transaction IDs are known throughout the entire replication hierarchy. This is not the case when using old-style replication; in this case the

replica knows only the specific file name and offset of the old primary server of the last event applied. There is no simple way to guess from this the correct file name and offset on a new primary.

2. The state of the replica is recorded in a crash-safe way.

The replica keeps track of its current position (the global transaction ID of the last transaction applied) in the mysql.gtid_slave_pos system table. If this table is using a transactional storage engine (such as InnoDB, which is the default), then updates to the state are done in the same transaction as the updates to the data. This makes the state crash-safe; if the replica server crashes, crash recovery on restart will make sure that the recorded replication position matches the changes that were actually replicated. This is not the case for old-style replication, where the state is recorded in a file relay-log.info, which is updated independently of the actual data changes and can easily get out of sync if the replica server crashes. (This works for DML to transactional tables; non-transactional tables in general are not crash-safe in MariaDB, and DDL is not yet fully crash-safe in replication.)

Because of these two benefits, it is generally recommended to use global transaction ID for any replication setups. However, old-style replication continues to work as always, so there is no pressing need to change existing setups. Global transaction ID integrates smoothly with old-style replication, and the two can be used freely together in the same replication hierarchy. There is no special configuration needed of the server to start using global transaction ID. Before MariaDB 10.10, it must be explicitly enabled for a replica server with the appropriate CHANGE MASTER option. Since MariaDB 10.10, global transaction ID is used by default.

Implementation

A global transaction ID, or GTID for short, consists of three numbers separated with dashes '-'. For example:

```
0-1-10
```

- The first number `0` is the domain ID, which is specific for global transaction ID (more on this below). It is a 32-bit unsigned integer.
- The second number is the server ID, the same as is also used in old-style replication. It is a 32-bit unsigned integer.
- The third number is the sequence number. This is a 64-bit unsigned integer that is monotonically increasing for each new event group logged into the binlog.

The server ID is set to the server ID of the server where the event group is first logged into the binlog. The sequence number is increased on a server for every event group logged. Since server IDs must be unique for every server, this makes the (`server_id`, `sequence_number`) pair, and hence the whole GTID, globally unique.

Using a 64-bit number provides ample range that there should be no risk of it overflowing in the foreseeable future. However, one should not artificially (by setting `gtid_seq_no`) inject a GTID with a very high sequence number close to the limit of 64-bit.

The Domain ID

When events are replicated from a primary server to a replica server, the events are always logged into the replica's binlog in the same order that they were read from the primary's binlog. Thus, if there is only ever a single primary server receiving (non-replication) updates at a time, then the binlog order will be identical on every server in the replication hierarchy.

This consistent binlog order is used by the replica to keep track of its current position in the replication. Basically, the replica remembers the GTID of the last event group replicated from the primary. When reconnecting to a primary, whether the same one or a new one, it sends this GTID position to the primary, and the primary starts sending events from the first event after the corresponding event group.

However, if user updates are done independently on multiple servers at the same time, then in general it is not possible for binlog order to be identical across all servers. This can happen when using multi-source replication, with

multi-master ring topologies, or just if manual updates are done on a replica that is replicating from active primary. If the binlog order is different on the new primary from the order on the old primary, then it is not sufficient for the replica to keep track of a single GTID to completely record the current state.

The domain ID, the first component of the GTID, is used to handle this.

In general, the binlog is not a single ordered stream. Rather, it consists of a number of different streams, each one identified by its own domain ID. Within each stream, GTIDs always have the same order in every server binlog. However, different streams can be interleaved in different ways on different servers.

A replica server then keeps track of its replication position by recording the last GTID applied within each replication stream. When connecting to a new primary, the replica can start replication from a different point in the binlog for each domain ID.

For more details on using multi-master setups and multiple domain IDs, see Use with multi-source replication and other multi-primary setups.

Simple replication setups only have a single primary being updated by the application at any one time. In such setups, there is only a single replication stream needed. Then domain ID can be ignored, and left as the default of `0` on all servers.

Using Global Transaction IDs

Global transaction ID is enabled automatically. Each event group logged to the binlog receives a GTID event, as can be seen with mariadb-binlog or SHOW BINLOG EVENTS.

The replica automatically keeps track of the GTID of the last applied event group, as can be seen from the gtid_slave_pos variable:

```
SELECT @@GLOBAL.gtid_slave_pos
0-1-1
```

When a replica connects to a primary, it can use either global transaction ID or old-style filename/offset to decide where in the primary binlogs to start replicating from. To use global transaction ID, use the CHANGE MASTER *master_use_gtid* option (default since MariaDB 10.10):

```
CHANGE MASTER TO master_use_gtid = { slave_pos | current_pos | no }
```

A replica is configured to use GTID by `CHANGE MASTER TO master_use_gtid=slave_pos`. When the replica connects to the primary, it will start replication at the position of the last GTID replicated to the replica, which can be seen in the variable gtid_slave_pos. Since GTIDs are the same across all replication servers, the replica can then be pointed to a different primary, and the correct position is determined automatically.

But suppose that we set up two servers A and B and let A be the primary and B the replica. It runs for a while. Then at some point we take down A, and B becomes the new primary. Then later we want to add A back, this time as a replica.

Since A was never a replica before, it does not have any prior replicated GTIDs, and gtid_slave_pos is empty. To allow A to be added as a replica automatically, use `CHANGE MASTER TO MASTER_DEMOTE_TO_SLAVE = 1`. This initializes the GTID position from the value of the variable gtid_current_pos, which also takes into account GTIDs written into the binlog when the server was a primary.

As an alternative, the option `master_use_gtid=current_pos` can be used to always connect using the value of the variable gtid_current_pos instead of gtid_slave_pos. When using `master_use_gtid=current_pos` there is no need to consider whether a server was a primary or a replica prior to using CHANGE MASTER. But care must be taken not to inject extra transactions into the binlog on the replica server that are not intended to be replicated to other servers. If such an extra transaction is the most recent when the replica starts, it will be used as the starting point of

replication. This will probably fail because that transaction is not present on the primary. To avoid local changes on a replica server to go into the binlog, set sql_log_bin to `0`.

Because of the potential problem with extra transactions on the replica, the use of `master_use_gtid=current_pos` is not recommended. Using `master_use_gtid=slave_pos` avoids that changes to the binlog on the replica affects the GTID replication position. Then the replica always connects to the primary at the position of the last replicated GTID. This may avoid some surprises for users that expect behavior consistent with traditional replication, where the replication position is never changed by local changes done on a server. And a server that was previously primary can still easily be turned into a replica using the `MASTER_DEMOTE_TO_SLAVE` option of `CHANGE MASTER`.

If a replica is configured with the binlog disabled, `current_pos` and `slave_pos` are equivalent.

Even when a replica is configured to connect with the old-style binlog filename and offset (`CHANGE MASTER TO master_log_file=..., master_log_pos=...`), it still keeps track of the current GTID position in `@@GLOBAL.gtid_slave_pos`. This means that an existing replica previously configured and running can be changed to connect with GTID (to the same or a new primary) simply with:

```
CHANGE MASTER TO master_use_gtid = slave_pos
```

The replica remembers that `master_use_gtid=slave_pos|master_pos` was specified, and uses it also for subsequent connects, until it is explicitly changed by specifying `master_log_file/pos=...` or `master_use_gtid=no`. The current value can be seen as the field `Using_Gtid` of `SHOW SLAVE STATUS`:

```
SHOW SLAVE STATUS\G
...
Using_Gtid: Slave_pos
```

The replica server internally uses the mysql.gtid_slave_pos table to store the GTID position (and so preserve the value of `@@GLOBAL.gtid_slave_pos` across server restarts).

In order to be crash-safe, this table must use a transactional storage engine such as InnoDB. When MariaDB is first installed, the table is created using the default storage engine - which itself defaults to InnoDB. If there is a need to change the storage engine for this table (to make it transactional on a system configured with MyISAM as the default storage engine, for example), use ALTER TABLE:

```
ALTER TABLE mysql.gtid_slave_pos ENGINE = InnoDB
```

The mysql.gtid_slave_pos table should not be modified in any other way. In particular, do not try to update the rows in the table to change the replica's idea of the current GTID position; instead use this:

```
SET GLOBAL gtid_slave_pos = '0-1-1'
```

The server variable gtid_pos_auto_engines can preferably be set to make the server handle the correct use of storage engine automatically. See the description of the mysql.gtid_slave_pos table for details.

Using GTIDs with Parallel Replication

If parallel replication is in use, then events that were logged with GTIDs with different gtid_domain_id values can be applied in parallel in an out-of-order manner.

Using GTIDs with MariaDB Galera Cluster

Starting with MariaDB 10.1.4, MariaDB Galera Cluster has limited support for GTIDs. See Using MariaDB GTIDs with MariaDB Galera Cluster for more information.

Setting up a New Replica Server with Global Transaction ID

Setting up a new replica server with global transaction ID is not much different from setting up an old-style replica. The basic steps are:

1. Setup the new server and load it with the initial data.
2. Start the replica replicating from the appropriate point in the primary's binlog.

Setting up a New Replica with an Empty Server

The simplest way for testing purposes is probably to setup a new, empty replica server and replicate all of the primary's binlogs from the start (this is usually not feasible in a realistic production setup, as the initial binlog files will probably have been purged or take too long to apply).

The replica server is installed in the normal way. By default, the GTID position for a newly installed server is empty, which makes the replica replicate from the start of the primary's binlogs. But if the replica was used for other purposes before, the initial position can be explicitly set to empty first:

```
SET GLOBAL gtid_slave_pos = "";
```

Next, point the replica to the primary with CHANGE MASTER. Specify `master_host` etc. as usual. But instead of specifying `master_log_file` and `master_log_pos` manually, use `master_use_gtid=slave_pos` to have GTID do it automatically:

```
CHANGE MASTER TO
 master_host="127.0.0.1",
 master_port=3310,
 master_user="root",
 master_use_gtid=slave_pos;
START SLAVE;
```

Setting up a New Replica From a Backup

The normal way to set up a new replication replica is to take a backup from an existing server (either a primary or replica in the replication topology), and then restore that backup on the server acting as the new replica, and the configure it to start replicating from the appropriate position in the primary's binary log.

It is important that the position at which replication is started corresponds exactly to the state of the data at the point in time that the backup was taken. Otherwise, the replica can end up with different data than the primary because of missing or duplicated transactions. Of course, if there are no writes to the server being backed up during the backup process, then a simple SHOW MASTER STATUS will give the correct position.

See the description of the specific backup tool to determine how to get the binary log position that corresponds to the backup.

Once the current binary log position for the backup has been obtained, in the form of a binary log file name and position, the corresponding GTID position can be obtained from BINLOG_GTID_POS() on the server that was backed up:

```
SELECT BINLOG_GTID_POS("master-bin.000001", 600);
```

The new replica can then start replicating from the primary by setting the correct value for gtid_slave_pos, and then executing CHANGE MASTER with the relevant values for the primary, and then starting the replica threads by executing START REPLICA. For example:

```
SET GLOBAL gtid_slave_pos = "0-1-2";
CHANGE MASTER TO
 master_host="127.0.0.1",
 master_port=3310,
 master_user="root",
 master_use_gtid=slave_pos;
START SLAVE;
```

This method is particularly useful when setting up a new replica from a backup of the primary. Remember to ensure that the value of server_id configured on the new replica is different from that of any other server in the replication topology.

If the backup was taken of an existing replica server, then the new replica should already have the correct GTID position stored in the mysql.gtid_slave_pos table. This is assuming that this table was backed up and that it was backed up in a consistent manner with changes to other tables. In this case, there is no need to explicitly look up the GTID position on the old server and set it on the new replica - it will be already correctly loaded from the mysql.gtid_slave_pos table. This however does not work if the backup was taken from the primary - because then the current GTID position is contained in the binary log, not in the mysql.gtid_slave_pos table or any other table.

Setting up a New Replica with mariadb-backup

A new replica can easily be set up with mariadb-backup, which is a fork of Percona XtraBackup. See Setting up a Replica with mariadb-backup for more information.

Setting up a New Replica with mariadb-dump

A new replica can also be set up with mariadb-dump.

mariadb-dump automatically includes the GTID position as a comment in the backup file if either the –master-data or –dump-slave option is used. It also automatically includes the commands to set gtid_slave_pos and execute CHANGE MASTER in the backup file if the –gtid option is used with either the –master-data or –dump-slave option.

Switching An Existing Old-Style Replica To Use GTID.

If there is already an existing replica running using old-style binlog filename/offset position, then this can be changed to use GTID directly. This can be useful for upgrades for example, or where there are already tools to setup new replica using old-style binlog positions.

When a replica connects to a primary using old-style binlog positions, and the primary supports GTID, the replica automatically downloads the GTID position at connect and updates it during replication. Thus, once a replica has connected to the GTID-aware primary at least once, it can be switched to using GTID without any other actions needed:

```
STOP SLAVE;
CHANGE MASTER TO
 master_host="127.0.0.1",
 master_port=3310,
 master_user="root",
 master_use_gtid=slave_pos;
START SLAVE;
```

Changing a Replica to Replicate From a Different Primary

Once replication is running with GTID (`master_use_gtid=current_pos|slave_pos`), the replica can be pointed to a new primary simply by specifying in `CHANGE MASTER` the new `master_host` (and if required `master_port` , `master_user` , and `master_password`):

```
STOP SLAVE;
CHANGE MASTER TO
 master_host='127.0.0.1',
 master_port=3312;
START SLAVE;
```

The replica has a record of the GTID of the last applied transaction from the old primary, and since GTIDs are identical across all servers in a replication hierarchy, the replica will just continue from the appropriate point in the new primary's binlog.

It is important to understand how this change of primary work. The binlog is an ordered stream of events (or multiple streams, one per replication domain, (see Use with multi-source replication and other multi-primary setups). Events within the stream are always applied in the same order on every replica that replicates it. The MariaDB GTID relies on this ordering, so that it is sufficient to remember just a single point within the stream. Since event order is the same on every server, switching to the point of the same GTID in the binlog of another server will give the same result.

This translates into some responsibility for the user. The MariaDB GTID replication is fully asynchronous, and fully flexible in how it can be configured. This makes it possible to use it in ways where the assumption that binlog sequence is the same on all servers is violated. In such cases, when changing primary, GTID will still attempt to continue at the point of current GTID in the new binlog.

The most common way that binlog sequence gets different between servers is when the user/DBA does updates directly on a replica server (and these updates are written into the replica's binlog). This results in events in the replica's binlog that are not present on the primary or any other replicas. This can be avoided by setting the session variable sql_log_bin false while doing such updates, so they do not go into the binlog.

It is normally best to avoid any differences in binlogs between servers. That being said, MariaDB replication is designed for maximum flexibility, and there can be valid reasons for introducing such differences from time to time. It this case, it just needs to be understood that the GTID position is a single point in each binlog stream (one per replication domain), and how this affects the user's particular setup.

Differences can also occur when two primary are active at the same time in a replication hierarchy. This happens when using a multi-master ring. But it can also occur in a simple master-slave setup, during switch to a new primary, if changes on the old primary is not allowed to fully replicate to all replica servers before switching primary. Normally, to switch primary, first writes to the old primary should be stopped, then one should wait for all changes to be replicated to the new primary, and only then should writes begin on the new primary. Deliberately using multiple active primary is also supported, this is described in the next section.

The GTID strict mode can be used to enforce identical binlogs across servers. When it is enabled, most actions that would cause differences are rejected with an error.

Use With Multi-Source Replication and Other Multi-Master Setups

MariaDB global transaction ID supports having multiple primaries active at the same time. Typically this happens with either multi-source replication or multi-master ring setups.

In such setups, each active primary must be configured with its own distinct replication domain ID, gtid_domain_id. The binlog will then in effect consists of multiple independent streams, one per active primary. Within one replication domain, binlog order is always the same on every server. But two different streams can be interleaved differently in different server binlogs.

The GTID position of a given replica is then not a single GTID. Rather, it becomes the GTID of the last event group applied for each value of domain ID, in effect the position reached in each binlog stream. When the replica connects to a primary, it can continue from one stream in a different binlog position than another stream. Since order within

one stream is consistent across all servers, this is sufficient to always be able to continue replication at the correct point in any new primary server(s).

Domain IDs are assigned by the DBA, according to the need of the application. The default value of `@@GLOBAL.gtid_domain_id` is `0`. This is appropriate for most replication setups, where only a single primary is active at a time. The MariaDB server never by itself introduces new `domain_id` values to the binlog.

When using multi-source replication, where a single replica connects to multiple primaries at the same time, each such primary should be configured with its own distinct domain ID.

Similarly, in a multi-master ring topology, where all primary in the ring are updated by the application concurrently (with some mechanism to avoid conflicts), a distinct domain ID should be configured for each server (In a multi-master ring where the application is careful to only do updates on one primary at a time, a single domain ID is sufficient).

Normally, a replica server should not receive direct updates (as this creates binlog differences compared to the primary). Thus it does not matter what value of `gtid_domain_id` is set on a replica, though it may make sense to make it the same as the primary (if not using multi-master) to make it easy to promote the replica as a new primary. Of course, if a replica is itself an active primary, as in a multi-master ring topology, the domain ID should be set according to the server's role as active primary.

Note that domain ID and server ID are distinct concepts. It is possible to use a different domain ID on each server, but this is normally not desirable. It makes the current GTID position (`@@global.gtid_slave_pos`) more complicated to understand and work with, and loses the concept of a single ordered binlog stream across all servers. It is recommended only to configure as many domain IDs as there are primary servers actively being updated by the application at the same time.

It is not an error in itself to configure domain IDs incorrectly (for example, not configuring them at all). The ring continues to work as before even though everything is configured to use the default domain ID `0`. It is even possible to use GTID for replication between the servers. However, care must be taken when switching a replica to a different primary. If the binlog order between the old and the new primary differs, then a single GTID position to start replication from in the new primary's binlog may not be sufficient.

Multiple Redundant Replication Paths

Using GTID with multi-source replication, it is possible to set up multiple redundant replication paths. For example:

```
M1 <-> M2
  M1 -> S1
  M1 -> S2
  M2 -> S1
  M2 -> S2
```

Here, M1 and M2 are setup in a master-master ring. S1 and S2 both replicate from each of M1 and M2. Each event generated on M1 will now arrive twice at S1, through the paths M1->S1 and M1->M2->S1. This way, if the network connection between M1 and S1 is broken, the replication can continue uninterrupted through the alternate path through M2. Note that this is an advanced setup, and good familiarity with MariaDB replication is recommended to successfully operate it.

The option –gtid-ignore-duplicates must be enabled to use multiple redundant replication paths. This is necessary to avoid each event being applied twice on the replica as it arrives through each path. The GTID of every event will be compared against the sequence number of the current GTID replica position (within each domain), and will be skipped if less than or equal. Thus it is required that sequence numbers are strictly increasing within each domain for –gtid-ignore-duplicates to function correctly, and setting –gtid-strict-mode=1 to help enforce this is recommended.

The –gtid-ignore-duplicates options also relaxes the requirement for connection to the primary. In the above example, when S1 connects to M2, it may connect at a GTID position from M1 that has not yet been applied on M2.

When –gtid-ignore-duplicates is enabled, the connection will be allowed, and S1 will start receiving events from M2 once the GTID has been replicated from M1 to M2. This can also be used to use replication filters in parts of a replication topology, to allow a replica to connect to a GTID position which was filtered on a primary. When –gtid-ignore-duplicates is enabled, the connecting replica will start receiving events from the primary at the first GTID sequence number that is larger than the connect-position.

Deleting Unused Domains

FLUSH BINARY LOGS DELETE_DOMAIN_ID=(list-of-domains) can be used to discard obsolete GTID domains from the server's binary log state. In order for this to be successful, no event group from the listed GTID domains can be present in existing binary log files. If some still exist, then they must be purged prior to executing this command.

If the command completes successfully, then it also rotates the binary log.

The old domains will still appear in gtid_slave_pos. To get rid of these, you can stop the replica and execute `SET GLOBAL` on the replica, specifying only the list of active domains to be retained:

```
SET GLOBAL gtid_slave_pos="<position with the old or unused domains removed>
```

Additional Syntax For Global Transaction ID

CHANGE MASTER

CHANGE MASTER has an option, `master_use_gtid=[current_pos|slave_pos|no]`. When enabled (set to *`current_pos`* or *`slave_pos`*), the replica will connect to the primary using the GTID position. When disabled (set to "no"), the old-style binlog filename/offset position is used to decide where to start replicating when connecting. The value *`replica_pos`* can be used as an alias for *`slave_pos`*. Unlike in the old-style, when GTID is enabled, the values of the MASTER_LOG_FILE and MASTER_LOG_POS options are not updated per received event in master_info_file file.

The value of `master_use_gtid` is saved across server restarts (in `master.info`). The current value can be seen as the field `Using_Gtid` in the output of `SHOW SLAVE STATUS`.

For a detailed look at the difference between the *`current_pos`* and *`slave_pos`* options, see Using global transaction IDs

START REPLICA UNTIL master_gtid_pos=xxx

When starting replication with START REPLICA, it is possible to request the replica to run only until a specific GTID position is reached. Once that position is reached, the replica stops.

The syntax for this is:

```
START SLAVE UNTIL master_gtid_pos = <GTID position>
```

The replica starts replication from the current GTID position, running up to and including the event with the GTID specified, and then stops. Note that this stops both the IO thread and the SQL thread (unlike `START SLAVE UNTIL MASTER_LOG_FILE/MASTER_LOG_POS`, which stops only the SQL thread).

If multiple GTIDs are specified, then they must be with distinct replication domain ID, for example:

```
START SLAVE UNTIL master_gtid_pos = "1-11-100,2-21-50"
```

With multiple domains in the `UNTIL` condition, each domain runs only up to and including the specified position, so it is possible for different domains to stop at different places in the binlog (each domain will resume from the stopped position when the replica is started the next time).

Not specifying a replication domain at all in the `UNTIL` condition means that the domain is stopped immediately, nothing is replicated from that domain. In particular, specifying the empty string will stop the replica immediately.

When using `START SLAVE UNTIL master_gtid_pos = XXX`, if the UNTIL position is present in the primary's binlog then it is permissible for the start position to be missing on the primary. In this case, replication for the associated domains stop immediately.

Both replica threads must be already stopped when using `UNTIL master_gtid_pos`, otherwise an error occurs. It is also an error if the replica is not configured to use GTID (`CHANGE MASTER TO master_use_gtid=current_pos|slave_pos`). And both threads must be started at the same time, the `IO_THREAD` or `SQL_THREAD` options can not be used to start only one of them.

`START SLAVE UNTIL master_gtid_pos=XXX` is particularly useful for promoting a new primary among a set of replicas when the old primary goes away and replicas may have reached different positions in the old primary's binlog. The new primary needs to be ahead of all the other replicas to avoid losing events. This can be achieved by picking one server, say S1, and replicating any missing events from each other server S2, S3, …, Sn:

```
CHANGE MASTER TO master_host="S2";
    START SLAVE UNTIL master_gtid_pos = "<S2 GTID position>";
    ...
    CHANGE MASTER TO master_host="Sn";
    START SLAVE UNTIL master_gtid_pos = "<Sn GTID position>";
```

Once this is completed, S1 has all events present on any of the servers. It can now be selected as the new primary, and all the other servers set to replicate from it.

> **Note:**
>
> The following functionality is available from MariaDB 11.3.

SQL_BEFORE_GTIDS|SQL_AFTER_GTIDS

The `START SLAVE UNTIL` statement contains the options `SQL_BEFORE_GTIDS` and `SQL_AFTER_GTIDS`, to allow control of whether the replica stops before or after a provided GTID state. Its syntax is:

```
START SLAVE UNTIL (SQL_BEFORE_GTIDS|SQL_AFTER_GTIDS)="<gtid_list>"
```

When providing `SQL_BEFORE_GTIDS="<gtid_list>"`, the replica executes all transactions up to the first GTID found in the provided list, and stop immediately. In contrast to the default behavior of `UNTIL`, this executes transactions from all domains on the primary until the replica stops due to seeing a GTID on the list.

`START SLAVE UNTIL SQL_AFTER_GTIDS="<gtid_list>"` is an alias to the default behavior of `START SLAVE UNTIL master_gtid_pos="<gtid_list>"`. That is, the replica only executes transactions originating from domain ids provided in the list, and will stop once all transactions provided in the `UNTIL` list have all been executed.

Example

If a primary server has a binary log consisting of the following GTIDs:

- 0-1-1
- 1-1-1
- 0-1-2
- 1-1-2
- 0-1-3
- 1-1-3

If a fresh replica (for instance, one with an empty GTID position, `@@gtid_slave_pos=''`) is started with `SQL_BEFORE_GTIDS`, for example, `START SLAVE UNTIL SQL_BEFORE_GTIDS="1-1-2"`, the resulting `gtid_slave_pos` of the replica is "0-1-2,1-1-1". This is because the replica will execute all events until it sees the transaction with GTID 1-1-2 and immediately stop without executing it. However, if a replica is started with `SQL_AFTER_GTIDS`, i.e. `START SLAVE UNTIL SQL_AFTER_GTIDS="1-1-2"` then the resulting `gtid_slave_pos` of the replica will be "1-1-2". This is because it only executes events from domain 1 until it has executed the provided GTID.

BINLOG_GTID_POS().

The BINLOG_GTID_POS() function takes as input an old-style binary log position in the form of a file name and a file offset. It looks up the position in the current binlog, and returns a string representation of the corresponding GTID position. If the position is not found in the current binlog, `NULL` is returned.

MASTER_GTID_WAIT

The MASTER_GTID_WAIT function is useful in replication for controlling primary/replica synchronization, and blocks until the replica has read and applied all updates up to and including the specified GTID position. See MASTER_GTID_WAIT for details.

Binlog Indexing

> **Note:**
>
> The following functionality is available from MariaDB 11.4.

Previously, when a replica connected, MariaDB needed to scan binlog files from the beginning in order to find the place to start replicating. If replica reconnects are frequent, this can be slow. Now, indexing is done on the binlog files, allowing GTIDs to be quickly found. This also detects if old-style replication tries to connect at an incorrect file offset (eg. in the middle of an event), avoiding sending potentially corrupted events.

The feature is enabled by default. The size of the binlog index file (`.idx`) is generally less than 1% the size of the binlog, so should not have any negative impacts and should not normally need tuning. However, the feature can be disabled or managed with the following system variables:

- binlog_gtid_index - enable/disable the feature
- binlog_gtid_index_page_size - adjust the size of the pages
- binlog_gtid_index_span_min - adjust the sparseness of the index

There are two status variables that can be used to monitor the effectiveness of the index:

- binlog_gtid_index_hit - incremented for each successful lookup in a GTID index.
- binlog_gtid_index_miss - incremented when a GTID index lookup is not possible, which indicates that the index file is missing (eg. binlog written by old server version without GTID index support), or corrupt.

System Variables

binlog_gtid_index

Enables/disables binlog indexing.

binlog_gtid_index_page_size

Adjusts the size of the pages.

binlog_gtid_index_span_min

Adjusts the sparseness of the index.

`gtid_slave_pos`

This system variable contains the GTID of the last transaction applied to the database by the server's replica threads for each replication domain. This system variable's value is automatically updated whenever a replica thread applies an event group. This system variable's value can also be manually changed by users, so that the user can change the GTID position of the replica threads.

When using multi-source replication, the same GTID position is shared by all replica connections. In this case, different primaries should use different replication domains by configuring different gtid_domain_id values. If one primary was using a gtid_domain_id value of `1`, and if another primary was using a gtid_domain_id value of `2`, then any replicas replicating from both primaries would have GTIDs with both gtid_domain_id values in `gtid_slave_pos`.

This system variable's value can be manually changed by executing SET GLOBAL, but all replica threads to be stopped with STOP REPLICA first. For example:

```
STOP ALL SLAVES;
SET GLOBAL gtid_slave_pos = "1-10-100,2-20-500";
START ALL SLAVES;
```

This system variable's value can be reset by manually changing its value to the empty string. For example:

```
SET GLOBAL gtid_slave_pos = '';
```

The GTID position defined by `gtid_slave_pos` can be used as a replica's starting replication position by setting MASTER_USE_GTID=slave_pos when the replica is configured with the CHANGE MASTER TO statement. As an alternative, the gtid_current_pos system variable can also be used as a replica's starting replication position.

If a user sets the value of the `gtid_slave_pos` system variable, and gtid_binlog_pos contains later GTIDs for certain replication domains, then gtid_current_pos will contain the GTIDs from gtid_binlog_pos for those replication domains. To protect users in this scenario, if a user sets the `gtid_slave_pos` system variable to a GTID position that is behind the GTID position in gtid_binlog_pos, then the server will give the user a warning.

This can help protect the user when the replica is configured to use gtid_current_pos as its replication position. This can also help protect the user when a server has been rolled back to restart replication from an earlier point in time, but the user has forgotten to reset gtid_binlog_pos with RESET MASTER.

The mysql.gtid_slave_pos system table is used to store the contents of global.gtid_slave_pos and preserve it over restarts.

- Command line: None
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default: Null

`gtid_binlog_pos`

This variable is the GTID of the last event group written to the binary log, for each replication domain.

Note that when the binlog is empty (such as on a fresh install with –skip-test-db, or after RESET MASTER), there are no event groups written in any replication domain, so in this case the value of `gtid_binlog_pos` will be the empty string.

The value is read-only, but it is updated whenever a DML or DDL statement is written to the binary log. The value can be reset by executing RESET MASTER, which will also delete all binary logs. However, note that RESET MASTER does not also reset gtid_slave_pos. Since gtid_current_pos is the union of gtid_slave_pos and `gtid_binlog_pos`, that means that new GTIDs added to `gtid_binlog_pos` can lag behind those in gtid_current_pos if gtid_slave_pos contains GTIDs in the same domain with higher sequence numbers. If you want to reset gtid_current_pos for a specific GTID domain in cases like this, then you will also have to change gtid_slave_pos in addition to executing RESET MASTER. See gtid_slave_pos for notes on how to change its value.

- Command line: None
- Scope: Global
- Dynamic: Read-only
- Data Type: `string`
- Default: Null

`gtid_binlog_state`

This variable holds the internal state of the binlog. The state consists of the last GTID ever logged to the binary log for every combination of `domain_id` and `server_id`. This information is used by the primary to determine whether a given GTID has been logged to the binlog in the past, even if it has later been deleted due to binlog purge. For each domain_id, the last entry in `@@gtid_binlog_state` is the last GTID logged into binlog, for instance, this is the value that appears in `@@gtid_binlog_pos`.

Normally this internal state is not needed by users, as `@@gtid_binlog_pos` is more useful in most cases. The main usage of `@@gtid_binlog_state` is to restore the state of the binlog after `RESET MASTER` (or equivalently if the binlog files are lost). If the value of `@@gtid_binlog_state` is saved before `RESET MASTER` and restored afterwards, the primary will retain information about past history, same as if `PURGE BINARY LOGS` had been used (of course the actual events in the binary logs are still deleted).

> **Note:**
>
> To set the value of `@@gtid_binlog_state`, the binary log must be empty, that is, it must not contain any GTID events and the previous value of `@@gtid_binlog_state` must be the empty string. If not, then `RESET MASTER` must be used first to erase the binary log first.

The value of `@@gtid_binlog_state` is preserved by the server across restarts by writing a file `MASTER-BIN.state`, where `MASTER-BIN` is the base name of the binlog set with the `--log-bin` option. This file is written at server shutdown, and re-read at next server start. (In case of a server crash, the data in the `MASTER-BIN.state` is not correct, and the server instead recovers the correct value during binlog crash recovery by scanning the binlog files and recording each GTID found). When using the InnoDB-based binary log (`--binlog-storage-engine=innodb`), the value of `@@gtid_binlog_state` is preserved internally by the InnoDB binlog implementation, and the file `MASTER-BIN.state` is not used.

For completeness, note that setting @@gtid_binlog_state internally executes a `RESET MASTER`. This is normally not noticeable as it can only be changed when the binlog is empty of GTID events. However, if executed e.g. immediately after upgrading to MariaDB 10.0, it is possible that the binlog is non-empty but without any GTID events, in which case all such events will be deleted, just as if `RESET MASTER` had been run.

- Command line: None
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default: Null

`gtid_current_pos`

This system variable contains the GTID of the last transaction applied to the database for each replication domain.

The value of this system variable is constructed from the values of the gtid_binlog_pos and gtid_slave_pos system variables. It gets GTIDs of transactions executed locally from the value of the gtid_binlog_pos system variable. It gets GTIDs of replicated transactions from the value of the gtid_slave_pos system variable.

For each replication domain, if the server_id of the corresponding GTID in gtid_binlog_pos is equal to the servers own server_id,*and* the sequence number is higher than the corresponding GTID in gtid_slave_pos, then the GTID from gtid_binlog_pos will be used. Otherwise the GTID from gtid_slave_pos will be used for that domain.

GTIDs from gtid_binlog_pos in which the server_id of the GTID is **not** equal to the server's own server_id are effectively ignored. If gtid_binlog_pos contains a GTID for a given replication domain, but the server_id of the GTID is **not** equal to the server's own server_id, and gtid_slave_pos does **not** contain a GTID for that given replication domain, then `gtid_current_pos` will **not** contain any GTID for that replication domain.

Thus, `gtid_current_pos` contains the most recent GTID executed on the server, whether this was done as a primary or as a replica.

The GTID position defined by `gtid_current_pos` can be used as a replica's starting replication position by setting MASTER_USE_GTID=current_pos when the replica is configured with the CHANGE MASTER TO statement. As an alternative, the gtid_slave_pos system variable can also be used as a replica's starting replication position.

The value of `gtid_current_pos` is read-only, but it is updated whenever a transaction is written to the binary log and/or replicated by a replica thread, and that transaction's GTID is considered *newer* than the current GTID for that domain. See above for the rules on how to determine if a GTID would be considered *newer*.

If you need to reset the value, see the notes on resetting gtid_slave_pos and gtid_binlog_pos, since `gtid_current_pos` is formed from the values of those variables.

- Command line: None
- Scope: Global
- Dynamic: Read-only
- Data Type: `string`
- Default: Null

`gtid_strict_mode`

The GTID strict mode is an optional setting that can be used to help DBAs enforce a strict discipline about keeping binlogs identical across multiple servers replicating using global transaction ID.

When GTID strict mode is enabled, some additional errors are enabled for situations that could otherwise cause differences between binlogs on different servers in a replication hierarchy:

1. If a replica server tries to replicate a GTID with a sequence number lower than what is already in the binlog for that replication domain, the SQL thread stops with an error (this indicates an extra transaction in the replica binlog not present on the primary).
2. Similarly, an attempt to manually binlog a GTID with a lower sequence number (by setting `@@SESSION.gtid_seq_no`) is rejected with an error.
3. If the replica tries to connect starting at a GTID that is missing in the primary's binlog, this is an error in GTID strict mode even if a GTID exists with a higher sequence number (this indicates a GTID on the replica missing on the primary). Note that this error is controlled by the setting of GTID strict mode on the connecting replica server.

GTID strict mode is off by default; this is needed to preserve backwards compatibility with existing replication setups (older versions of the server did not enforce any strict mode for binlog order). Global transaction ID is designed to

work correctly even when strict mode is not enabled. However, with strict mode enforced, the semantics is simpler and thus easier to understand, because binlog order is always identical across servers and sequence numbers are always strictly increasing within each replication domain. This can also make automated scripting of large replication setups easier to implement correctly.

When GTID strict mode is enabled, the replica will stop with an error when a problem is encountered. This allows the DBA to become aware of the problem and take corrective actions to avoid similar issues in the future. One way to recover from such an error is to temporarily disable GTID strict mode on the offending replica, to be able to replicate past the problem point (perhaps using `START SLAVE UNTIL master_gtid_pos=XXX`).

- Command line: `--gtid-strict-mode[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default: `Off`

`gtid_domain_id`

- Description: This variable is used to decide which replication domain new GTIDs are logged in for a primary server. See Use with multi-source replication and other multi-primary setups for details. This variable can also be set on the session level by a user with the SUPER privilege. This is used by mariadb-binlog to preserve the domain ID of GTID events.
- Command line: `--gtid-domain-id=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric (32-bit unsigned integer)`
- Default Value: `0`
- Range: `0` to `4294967295`

`last_gtid`

- Description: Holds the GTID that was assigned to the last transaction, or statement that was logged to the binary log. If the binary log is disabled, or if no transaction or statement was executed in the session yet, then the value is an empty string.
- Scope: Session
- Dynamic: Read-only
- Data Type: `string`

`server_id`

- Description: Server_id can be set on the session level to change which server_id value is logged in binlog events (both GTID and other events). This is used by mariadb-binlog to preserve the server ID of GTID events.
- Scope: Global, Session
- Dynamic: Yes
- Data Type: numeric (32-bit unsigned integer)

`gtid_seq_no`

- Description: The `gtid_seq_no` session variable can be set to manually override the sequence number for the following transaction's Global Transaction ID (GTID).
  By default, the server internally maintains an increasing counter where each committing transaction uses the next value from this counter as its sequence number when writing its GTID event to the binary log (along with

`@@gtid_domain_id` and `@@server_id`).
When `gtid_seq_no` is specified, it overrides this counter for the next committing transaction in the respective session; this value is then used as the sequence number when writing the GTID event to the binary log. Crucially, the server's internal counter also resets to this specified value, ensuring that subsequent transactions increment from this new value when generating their sequence numbers.
This variable is typically used internally and by tools like `mariadb-binlog` to maintain GTID consistency when applying events across different servers (e.g., from a primary to a replica).

- Command line: None
- Scope: Session
- Dynamic: Yes
- Data Type: `numeric (64-bit unsigned integer)`
- Default: Null

`gtid_ignore_duplicates`

- Description: When set, different primary connections in multi-source replication are allowed to receive and process event groups with the same GTID (when using GTID mode). Only one will be applied, any others will be ignored. Within a given replication domain, just the sequence number will be used to decide whether a given GTID has been already applied; this means it is the responsibility of the user to ensure that GTID sequence numbers are strictly increasing. With `gtid_ignore_duplicates=OFF`, a duplicate event based on domain id and sequence number, will be executed. When `--gtid-ignore-duplicate` is set, a replica is allowed to connect at a GTID position that does not exist on the primary. The replica starts receiving events once a GTID with a higher sequence number is available on the primary (within that domain). This can be used to allow a replica to connect at a GTID position that was filtered on the primary, eg. using `--replicate-ignore-table`. See also Multiple Redundant Replication Paths.
- Command line: `--gtid-ignore-duplicates=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default: `OFF`

`gtid_pos_auto_engines`

This variable is used to enable multiple versions of the mysql.gtid_slave_pos table, one for each transactional storage engine in use. This can improve replication performance if a server is using multiple different storage engines in different transactions.

The value is a list of engine names, separated by commas (`,`). Replication of transactions using these engines will automatically create new versions of the `mysql.gtid_slave_pos` table in the same engine and use that for future transactions (table creation takes place in a background thread). This avoids introducing a cross-engine transaction to update the GTID position. Only transactional storage engines are supported for `gtid_pos_auto_engines` (this currently means InnoDB or MyRocks).

The variable can be changed dynamically, but replica SQL threads should be stopped when changing it, and it will take effect when the replicas are running again.

When setting the variable on the command line or in a configuration file, it is possible to specify engines that are not enabled in the server. The server will then still start if, for example, that engine is no longer used. Attempting to set a non-enabled engine dynamically in a running server (with `SET GLOBAL gtid_pos_auto_engines`) still results in an error.

Removing a storage engine from the variable will have no effect once the new tables have been created – as long as these tables are detected, they are used.

- Command line: `--gtid-pos-auto-engines=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `string` (comma-separated list of engine names)
- Default: empty

`gtid_cleanup_batch_size`

- Description: Normally does not need tuning. How many old rows must accumulate in the mysql.gtid_slave_pos table before a background job is run to delete them. Can be increased to reduce number of commits if using many different engines with gtid_pos_auto_engines, or to reduce CPU overhead if using a huge number of different gtid_domain_ids. Can be decreased to reduce number of old rows in the table.
- Command line: `--gtid-cleanup-batch-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default: `64`
- Range: `0` to `2147483647`
- Introduced: MariaDB 10.4.1

See Also

- FLUSH binary logs

This page is licensed: CC BY-SA / Gnu FDL

Read-Only Replicas

A common replication setup is to have the replicas read-only to ensure that no one accidentally updates them. If the replica has binary logging enabled and gtid_strict_mode is used, then any update that causes changes to the binary log will stop replication.

When the variable `read_only` is set to 1, no updates are permitted except from users with the SUPER privilege (<= MariaDB 10.5.1) or READ ONLY ADMIN privilege (>= MariaDB 10.5.2) or replica servers updating from a primary. Inserting rows to log tables, updates to temporary tables and OPTIMIZE TABLE or ANALYZE TABLE statements on temporary tables are excluded from this limitation.

If read_only is set to 1, then the SET PASSWORD statement is limited only to users with the SUPER privilege (<= MariaDB 10.5.1) or READ ONLY ADMIN privilege (>= MariaDB 10.5.2).

Attempting to set the `read_only` variable to 1 will fail if the current session has table locks or transactions pending.

The statement will wait for other sessions that hold table locks. While the attempt to set read_only is waiting, other requests for table locks or transactions will also wait until read_only has been set.

Older MariaDB Versions

If you are using an older MariaDB version with read-only replicas and binary logging enabled on the replica, and you need to do some changes but don't want to have them logged to the binary log, the easiest way to avoid the logging is to disable binary logging while running as root during maintenance:

```
SET sql_log_bin=0;
ALTER TABLE test ENGINE=rocksdb;
```

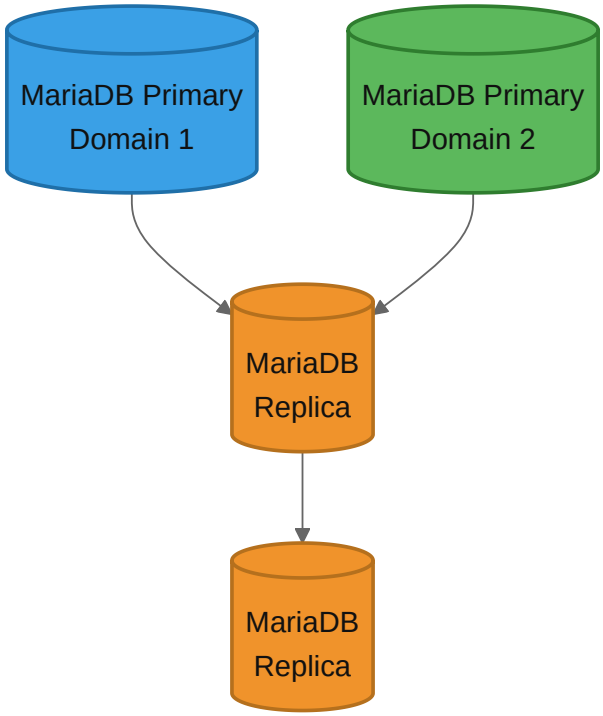
The above changes the test table on the replica to rocksdb without registering the change in the binary log.

This page is licensed: CC BY-SA / Gnu FDL

Multi-Source Replication

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Multi-source replication means that one server has many primaries from which it replicates.



Multi-source replication: one replica pulls from two primaries on separate domains and applies them in parallel.

New Syntax

You specify which primary connection you want to work with by either specifying the connection name in the command or setting default_master_connection to the connection you want to work with.

The connection name may include any characters and should be less than 64 characters. Connection names are compared without regard to case (case insensitive). You should preferably keep the connection name short, as it will be used as a suffix for relay logs and primary info index files.

The new syntax was introduced to handle many connections:

- CHANGE MASTER ['connection_name'] TO ….
  Creates or modifies a connection to a primary.
- FLUSH RELAY LOGS ['connection_name']
- MASTER_POS_WAIT(….,['connection_name'])

- [RESET SLAVE ['connection_name'] [ALL]
  Used to reset a replica's replication position or to remove a replica permanently.
- SHOW RELAYLOG ['connection_name'] EVENTS
- SHOW SLAVE ['connection_name'] STATUS
- SHOW ALL SLAVES STATUS
- [START SLAVE ['connection_name']]
- START ALL SLAVES …
- STOP SLAVE ['connection_name'] …
- STOP ALL SLAVES …

The original old-style connection is an empty string `''`. You don't have to use this connection if you don't want to.

You create new primary connections with CHANGE MASTER. You delete the connection permanently with RESET SLAVE 'connection_name' ALL.

Replication Variables for Multi-Source

The new replication variable default_master_connection specifies which connection will be used for commands and variables if you don't specify a connection. By default, this is `''` (the default connection name).

The following replication variables are local for the connection. (In other words, they show the value for the `@@default_master_connection` connection). We are working on making all the important ones local for the connection.

Type	Name	Description
Variable	max_relay_log_size	Maximum size of relay log. Is set at startup to max_binlog_size if 0
Variable	replicate_do_db	Tell the replica to restrict replication to updates of tables whose names appear in the comma-separated list. For statement-based replication, only the default database (that is, the one selected by USE) is considered, not any explicitly mentioned tables in the query. For row-based replication, the actual names of the table(s) being updated are checked.
Variable	replicate_do_table	Tells the replica to restrict replication to tables in the comma-separated list.
Variable	replicate_ignore_db	Tell the replica to restrict replication to updates of tables whose names do not appear in the comma-separated list. For statement-based replication, only the default database (that is, the one selected by USE) is considered, not any explicitly mentioned tables in the query. For row-based replication, the actual names of the table(s) being updated are checked.
Variable	replicate_ignore_table	Tells the replica thread not to replicate any statement that updates the specified table, even if any other tables might be updated by the same statement.
Variable	replicate_rewrite_db	Starting with MariaDB 10.11, one can configure a replica to rewrite database names. It uses the format primary_database->replica_database. If a replica encounters a binary log event in which the default database (i.e., the one selected by the USE statement) is primary_database, then the replica will apply the event in replica_database instead.
Variable	replicate_wild_do_table	Tells the replica thread to restrict replication to statements where any of the updated tables match the specified database and table name patterns.
Variable	replicate_wild_ignore_table	Tells the replica thread not to replicate to the tables that match the given wildcard pattern.
Status	Slave_heartbeat_period	How often to request a heartbeat packet from the primary (in seconds).

Type	Name	Description
Status	Slave_received_heartbeats	How many heartbeats do we have from the primary.
Status	Slave_running	Shows if the replica is running. YES means that the sql thread and the IO thread are active. No means either one is not running. '' means that @@default_master_connection doesn't exist.
Variable	Sql_slave_skip_counter	How many entries in the replication log that should be skipped (mainly used in case of errors in the log).

You can access all of the above variables with either `SESSION` or `GLOBAL`.

Note that in contrast to MySQL, all variables always show the correct active value!

Example:

```
set @@default_master_connection='';
show status like 'Slave_running';
set @@default_master_connection='other_connection';
show status like 'Slave_running';
```

If `@@default_master_connection` contains a non existing name, you will get a warning.

All other primary-related variables are global and affect either only the default (`''`) connections or all connections. For example, Slave_retried_transactions now shows the total number of retried transactions over all replicas.

If you need to set gtid_slave_pos, you need to set this for all primaries at the same time.

New status variables:

Name	Description
Com_start_all_slaves	Number of executed START ALL SLAVES commands.
Com_start_slave	Number of executed START SLAVE commands. This replaces Com_slave_start.
Com_stop_slave	Number of executed STOP SLAVE commands. This replaces Com_slave_stop.
Com_stop_all_slaves	Number of executed STOP ALL SLAVES commands.

SHOW ALL SLAVES STATUS has the following new columns:

Name	Description
Connection_name	Name of the primary connection. This is the first variable.
Slave_SQL_State	State of SQL thread.
Retried_transactions	Number of retried transactions for this connection.
Max_relay_log_size	Max relay log size for this connection.
Executed_log_entries	How many log entries the replica has executed.
Slave_received_heartbeats	How many heartbeats we have got from the primary.
Slave_heartbeat_period	How often to request a heartbeat packet from the primary (in seconds).

New Files

The basic principle of the new files used by multi-source replication is that they have the same name as the original relay log files, suffixed with `connection_name` before the extension. The main exception is that the file that holds all connections is named as the normal `master-info-file` with a `multi-` prefix.

When you are using multi-source, the following new files are created:

Name	Description
multi-master-info-file	The master-info-file (normally master.info) with a multi- prefix. This contains all primary connections in use.
master-info-file-connection_name.extension	Contains the current primary position for what's applied to in the replica. Extension is normally .info
relay-log-connection_name.xxxxx	The relay-log name with a connection_name suffix. The xxxxx is the relay log number. This contains the replication data read from the primary.
relay-log-index-connection_name.extension	Contains the name of the active relay-log-connection_name.xxxxx files. Extension is normally .index
relay-log-info-file-connection_name.extension	Contains the current primary position for the relay log. Extension is normally .info

When creating the file, the connection name is converted to lowercase, and all special characters in the connection name are converted, the same way as MySQL table names are converted. This is done to make the file name portable across different systems.

> **Note:**
>
> Instead of specifying names for `mysqld` with –relay-log, –relay-log-index, –general-log-file, –slow-query-log-file,–log-bin, and –log-bin-index, you can just specify –log-basename, and all the other variables are set with this as a prefix.

Other Things

- All error messages from a replica with a connection name that are written to the error log are prefixed with `Master 'connection_name':`. This makes it easy to see from where an error originated.
- Errors `ER_MASTER_INFO` and `WARN_NO_MASTER_INFO` now includes connection_name.
- There is no conflict resolution. The assumption is that there are no conflicts in data between the different primaries.
- All executed commands are stored in the normal binary log (nothing new here).
- If the server variable `log_warnings` > 1, then you will get some information in the log about how the multi-master-info file is updated (mainly for debugging).
- The output of SHOW ALL SLAVES STATUS has one more column than `SHOW SLAVE STATUS`, since it includes the `connection_name` column.
- RESET SLAVE now deletes all relay-log files.

replicate-… Variables

- One can set the values for the `replicate-...` variables from the command line or in `my.cnf` for a given connection by prefixing the variable with the connection name.
- If one doesn't use any connection name prefix for a `replicate..` variable, then the value will be used as the default value for all connections that don't have a value set for this variable.

Example:

```
mysqld --main_connection.replicate_do_db=main_database --replicate_do_db=other_database
```

This sets the `replicate_do_db` variable to `main_database` for the connection named `main_connection`. All other connections will use the value `other_database`.

One can also use this syntax to set `replicate-rewrite-db` for a given connection.

Typical Use Cases

- You are partitioning your data over many primaries and would like to get it all together on one machine to do analytical queries on all the data.
- You have many databases spread over many MariaDB/MySQL servers and would like to have all of them on one machine as an extra backup.
- In a Galera cluster, the default replication filter rules, like `replicate-do-db`, do not apply to replication connections, but also to Galera write set applier threads. By using a named multi-primary replication connection instead, even when replicating from just one primary into the cluster, the primary-replica replication rules can be kept separate from the Galera intra-node replication traffic.

Limitations

- Each active connection will create 2 threads (as is normal for MariaDB replication).
- You should ensure that all primaries have different `server-id`'s. If you don't do this, you get into trouble if you try to replicate from the multi-source replica back to your primaries.
- One can change max_relay_log_size for any active connection, but new connections will always use the server startup value for `max_relay_log_size`, which can't be changed at runtime.
- Option innodb-recovery-update-relay-log (xtradb feature to store and restore relay log position for replicas) only works for the default connection ''. As this option is not really safe and can easily cause loss of data if you use storage engines other than InnoDB, we don't recommend using this option.
- slave_net_timeout affects all connections. We don't check anymore if it's less than Slave_heartbeat_period, as this doesn't make sense in a multi-source setup.

Incompatibilities with MariaDB/MySQL 5.5

- max_relay_log_size is now (almost) a normal variable and not automatically changed if max_binlog_size is changed. To keep things compatible with old config files, we set it to `max_binlog_size` at startup if its value is 0.
- You can now access replication variables that depend on the active connection with either `GLOBAL` or `SESSION`.
- We only write information about relay log positions for recovery if innodb-recovery-update-relay-log is set.
- Slave_retried_transactions now shows the total count of retried transactions over all replicas.
- The status variable `Com_slave_start` is replaced with Com_start_slave.
- The status variable `Com_slave_stop` is replaced with Com_stop_slave.
- `FLUSH RELAY LOGS` are not replicated anymore. This is not safe as connection names may be different on the replica.

See Also

- Multi-master ring replication.
- Using multi-source with global transaction id.
- The work in MariaDB is based on the project description at MDEV-253.
- The original code base comes from Taobao, developed by Peng Lixun. A big thanks to them for this important feature!

This page is licensed: CC BY-SA / Gnu FDL

Multi-Master Ring Replication

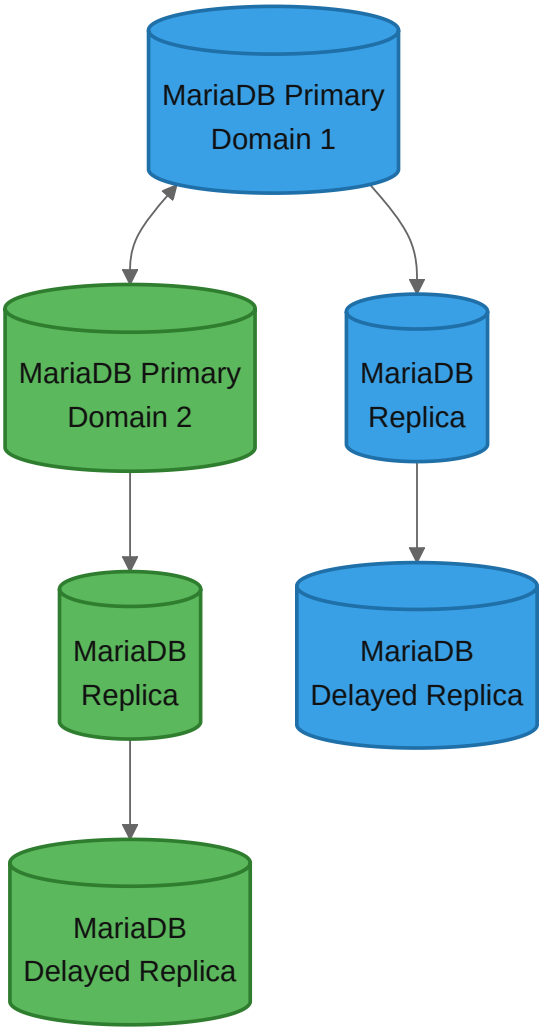
> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

What is Multi-Master Ring Replication

Multi-Master "Ring" Replication means that you have two or more masters where each master is replicating its data to another master asynchronously. This is not to be confused with MariaDB Galera Cluster which is a virtually synchronous multi-primary cluster for MariaDB.

The benefit of asynchronous replication compared to Galera Cluster, is that Ring Replication is resilient against bad network connections between the master servers. If a connection goes down, all masters will continue to serve its clients locally and data will automatically be synced when the link is available again.

The following picture shows one of the more advanced Multi-Master setups that is resilient against any master going down but can also handle 'human failures', like an accidental drop table, thanks to the addition of delayed slaves.



Multi-master ring with replicas: two primaries replicate to each other; each also has a replica and a delayed replica.

One should setup replication on each master like one does in standard MariaDB replication. The replication setup among the masters should be a ring. In other words, each master should replicate to one other master and each

master should only have one other master as a slave.

Each master can of course have one or more normal slaves. A master could also be a slave of another master that is not in the ring setup.

All MariaDB servers support Multi-Master Ring Replication. In fact, when MySQL replication was originally designed around the year 2000, it was to be a Multi-Master Ring Replication solution for Yahoo to replicate from the East Coast to the West Coast.

Configuring the Masters

First, follow the instructions in setup replication. The main thing to remember is to use the master_use_gtid=current_pos option for CHANGE MASTER.

The main things that are different for Multi-Master Ring Replication are:

- Give every master and slave in the replication setup a unique server_id. This can be a number from 1 to 4294967295 or 1-255 if one is using uuid_short(). It is a good practice to ensure that you do not have any servers in your system with the same server_id!
- Use global transaction id (as described above)
- Give each master a unique gtid_domain_id. This will allow replication to apply transactions from a different master in parallel independent from other masters.

Add the following into your my.cnf file for **all masters** and restart the servers.

```
[mariadb]
# Replace the following with a unique ID. All slaves of this master should have the same
# gtid_domain_id to allow easy failover to a slave if needed
gtid_domain_id=1
#
# Let us assume there will never be more than 10 masters in a Multi-Master ring setup
auto_increment_increment=10
# Set this to a different value 1-10 for each master. Could be the same as gtid_domain_id
# This is to ensure that all masters generate different values for AUTO_INCREMENT keys.
auto_increment_offset=1
#
# The following is needed to ensure the ALTER TABLE on another master will not
# break ring replication
slave_type_conversions=ALL_NON_LOSSY,ALL_LOSSY
#
# We cannot use semi-sync in Ring Replication as the masters need to be resilient against
# bad connections
rpl_semi_sync_master_enabled=0
#
# We have to log updates from other masters to the binary log.
log_slave_updates
```

Limitations when using Ring Replication

- MariaDB does not yet support conflict resolution for conflicting changes. It is up to the application to ensure that there is never a conflicting insert/update/delete between the masters. The easiest setup is having each master server work on a different database or table. If not, one must:
  - Ensure you have an id (master-unique-id) for each row that unequally identifies the master who is responsible for this row. This should preferably be short and part of the primary key in each table. A good value for this would be the `gtid_domain_id` as this is unique for each local cluster.
  - Never insert rows with `PRIMARY KEY` or `UNIQUE KEY` values that can be same on another master. This can be avoided by
    - Have the master-unique-id part of all primary and unique keys.
    - In case of AUTO_INCREMENT keys, have a different value for auto_increment_offset on each master.

        - Use uuid_short() to generate unique values, like in `create table t1 (a bigint unsigned default(uuid_short()) primary key)`. Note that if one is using uuid_short() in Multi-Master ring replication, one can only use `server_id` in the range 1-255!
    - Ensure that UPDATE and DELETE on each master only update rows generated by this master.
- If several masters are constantly generating and updating rows for common tables, one has to be extra careful with `ALTER TABLE` to ensure that any change one does will not cause conflicts when the `ALTER TABLE` is replicated to other servers. In particular one has to ensure that all masters and their slaves are configured with `slave_type_conversions=ALL_NON_LOSSY,ALL_LOSSY`.
- The `server_id` should be unique for each server. One should not change the `server_id` of an active master, as the ID is used by the master to recognize its own events and stop them from replicating endlessly around the ring (see replicate_same_server_id).

How does Multi-Master Ring Replication work

- The main difference between Multi-Master Ring Replication and normal replication is that a change done by a master will eventually replicate back to it. When this happens, the master will see that the binary log event has the same server_id as the master has and will ignore the event. This is why it is critical to ensure that all server_id's are unique and that one does not change the server id.
- When doing ALTER TABLE in a Multi-Master Ring replication setup, you should be aware the while you are running an ALTER TABLE on one master, another master can generate events that uses the old table definition. You should especially take care to not drop columns that could be in use by any application or still available in the upcoming replication stream.

How to resolve things if they go wrong in Multi-Master Ring Replication

When used correctly, Multi-Master Ring Replication is as resilient to errors as normal MariaDB master-slave replication. If the connection goes down, the replication will stop and will automatically continue when the connection resumes.

What to do when one of the masters dies and has to be replaced by a slave.

- Ensure that the slave is up to date (has digested all relay events).
- Check if there are any events on the old master that have not been sent to the slave. You can use mariadb-binlog to read the old master binary log files and apply them to the slave.
- You can now treat the slave as a new master and put it back in the replication ring. The new master will use its replication GTID position to continue replication from the other master in the ring.

If the slave is **not up to date** and one cannot access any information of the old master, then one can continue the following way:

- Enable the option –gtid-ignore-duplicates on the servers.
- Add the slave to the replication ring.
- The two masters (one of which is the old slave now added to the ring) will each replicate the events they are missing from one another. The `--gtid-ignore-duplicates` option is needed to allow the two masters in the ring to start replicating from each other when each server is ahead of the other in one domain and behind in another.

Error applying events

As long as each master handles their own set of data, as described above, there should not be any conflicting data coming from the other master.

If there are conflicts, one should resolve them as one resolves issues with normal replication.
The most common way to solve issues is to skip the conflicting log events with SET GLOBAL

SQL_SLAVE_SKIP_COUNTER.

Handling duplicate key errors and other conflicts

If things are setup correctly as described earlier, one should never get duplicate key errors in Multi-Master ring replication. Any duplicate key error or data mismatch is usually an application error where it inserts/updates or deletes something it should not have the right to do.

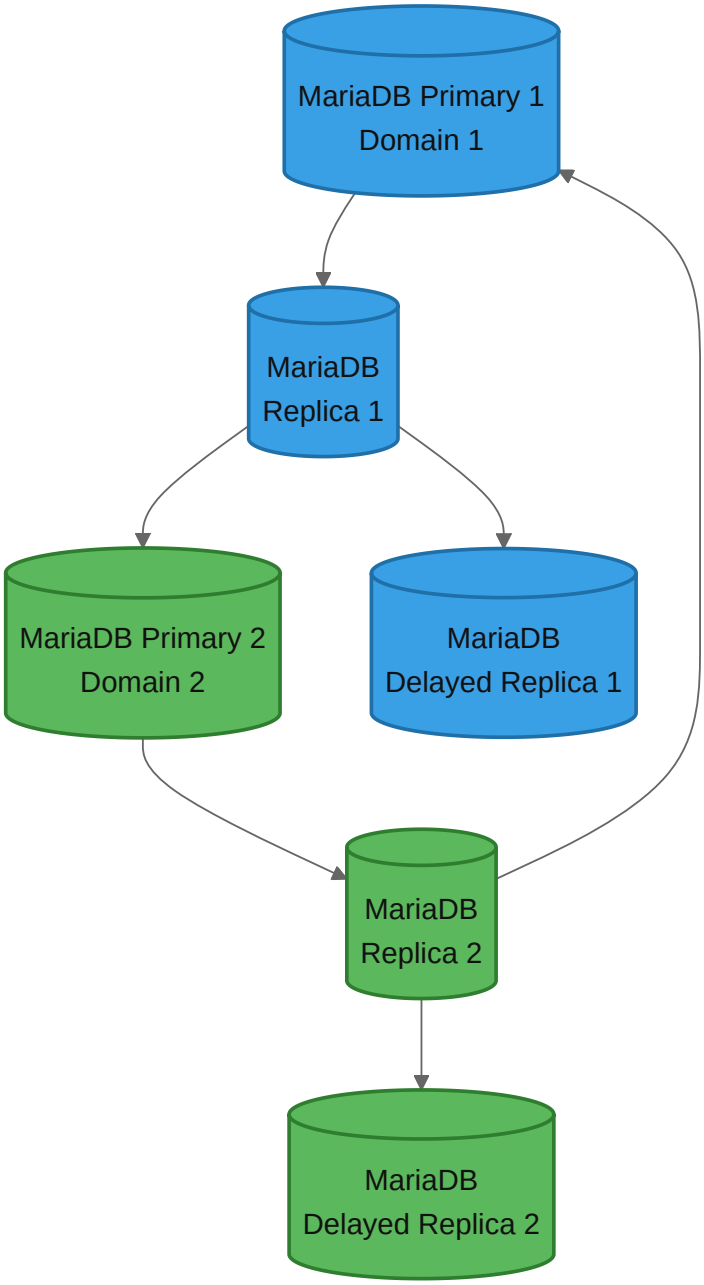
To fix this:

- Use SET GLOBAL SQL_SLAVE_SKIP_COUNTER to skip the error.
- Use `mariadb-binlog –base64-output=decode-rows –verbose –start-position=

binlog-name` to see what is missing and apply the changes missing on the server (minus the conflict).

Multi-Master Ring Replication through slaves

An alternative setup to use for Multi-master ring replications is to replicate to the other masters through slaves. The following setup shows how this can be done.



Multi-master ring relayed through replicas: each replica forwards to the other domain's primary, closing the ring.

Benefits of replication through slaves

- The slave will never be out of sync compared to other master. This makes failover from master to slave much easier as one does not have to check if slave has all data.
- Slightly less load on the master
- One can use semisynchronous-replication between master and slave.
- Slave configuration is consolidated to one place. That is, if a master's immediate slave and replicating master should share the same configuration (e.g. rewrite rules, filters, etc), only the immediate slave needs them configured, as the replicating master will pull in the effects inherently.

Disadvantages of replication through slaves

- There will be a slightly longer delay for the data to hit the next master as it has to go through the slave. This can be notable if there is a very large transaction executed on the master.

- If the master OR the slave dies, the replication to other masters will stop.
- A replicating master is subject to the configuration of a slave (e.g. transactions may be incorrectly filtered out).
- Re-setting replication after failover is a bit more complex.

Setting things up

- Setup is identical to Multi-Master Ring Replication, except that rpl_semi_sync_master_enabled=0 is not required.

What to do when one of the masters dies and has to be replaced by a slave.

Let assume that master1 in the above picture has failed.

What needs to be done is to replace master1 with slave1 and add a new slave to replace slave1.
Here follows a step by step description of how to do this.

The new slave that will be added to replace slave1 place will below be called slave3.
The new master will be called master3 (to simplify explanations).
Note that in some cases, the failed master can be reused as the new slave if it did recover properly. If this is the case, reset all replications setups on the failed master.

Note that when one sets up a master->slave replication, all configurations are done only on the slave!

- Promote slave1 as the new master3. Applications should now be moved to use master3. Note MaxScale can do this step automatically.
- Setup master3 as a slave of slave2.
  - The ring replication is now active (at least temporarily, until we have slave3 in place).
- Delete the old replication setup in master3 that pointed to the deleted master1.
- Prepare the new slave3 (which can take a while if it based on a backup of master3).
- Setup slave3 as a slave of master3.
- If delayed slave1 exists, redirect it to be a slave of slave3.
- Update master2 to be slave of slave3 (from being a slave of master3).

Some other options:

- For semi-sync setups, the old master1 can be reused as slave3 if re-started with `--init-rpl-role=SLAVE` during recovery
- For non-semi-synchronous setups, one can use option CHANGE MASTER TO MASTER_DEMOTE_TO_SLAVE=1 (requires MariaDB 10.11 or higher).

See also

- Multi-source replication

This page is licensed: CC BY-SA / Gnu FDL

Delayed Replication

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Delayed replication allows specifying that a replica should lag behind the primary by (at least) a specified amount of time (specified in seconds). Before executing an event, the replica will first wait, if necessary, until the given time has

passed since the event was created on the primary. The result is that the replica will reflect the state of the primary some time back in the past.

The default is zero, or no delay, and the maximum value is 2147483647, or about 68 years.

Delayed replication is enabled using the MASTER_DELAY option to CHANGE MASTER:

```
CHANGE MASTER TO master_delay=3600;
```

A zero delay disables delayed replication. The replica must be stopped when changing the delay value.

> **Tip:**
>
> New Variables to Measure Replication Lag
>
> MariaDB 10.6 ES and 11.6 introduced new variables to measure replication lag. The variables are recommended to monitor the time difference between the master and slave.
>
> - `Master_last_event_time` : The timestamp of the last event read from the master by the IO thread.
> - `Slave_last_event_time` : The timestamp of the last event committed on the slave (from the master's perspective).
> - `Master_Slave_time_diff` : The difference in seconds between `Master_last_event_time` and `Slave_last_event_time`.

Fields in SHOW SLAVE STATUS are associated with delayed replication

1. `SQL_Delay` : This is the value specified by MASTER_DELAY in CHANGE MASTER (or 0 if none).
2. `SQL_Remaining_Delay` : When the replica is delaying the execution of an event due to MASTER_DELAY, this is the number of seconds of delay remaining before the event will be applied. Otherwise, the value is NULL.
3. `Seconds_Behind_Master` :

- If replication is not active, the value is NULL.
- If all events has been executed and slave is now idle (waiting for new events), the value is 0.
- If using parallel replication:
  - When starting to execute a new delayed event on an idle slave the value is `clock_time_on_slave - clock_when_started_execution_of_the_event_on_master - clock_difference_between_master_and_slave`
  - When event is committed the value is updated to: `clock_time_on_slave - clock_when_ending_execution_of_the_event_on_master - clock_difference_between_master_and_slave`
  - As Seconds_Behind_Master value depends on the timestamp of the last executed event, its value can be much bigger than SQL_Delay. For example if SQL_delay is 10 hours and the master sends a new event after not having executed anything for 20 hours, Seconds_behind_master will be 30 hours. When event starts to execute, Seconds_behind_master will still be growing until it commits. Then it will be changed to about 10 hours (more precisely to 10 hours + execution_time_on_slave - execution_time_on_master which corresponds to the above bullet). Since that the value will be growing until next event is executed and Seconds_behind_master is rebased on its completion time again.
    To expand more on this check a practical example in the comments.
- If not using parallel replication the value is updated when starting to execute an event. The value is set to: `clock_time_on_slave - clock_when_started_execution_of_the_event_on_master - clock_difference_between_master_and_slave`. Note that in this case the clock is at start of event, not at end of event like in parallel replication.

1. `Slave_SQL_Running_State` : This shows the state of the SQL driver threads, same as in SHOW PROCESSLIST. When the replica is delaying the execution of an event due to MASTER_DELAY, this fields displays: "Waiting

until MASTER_DELAY seconds after master executed event".

When using older versions prior to MariaDB 10.2.3, a 3rd party tool called pt-slave-delay can be used. It is part of the Percona Toolkit. Note that pt-slave-delay does not support MariaDB multi-channel replication syntax.

clock_difference_between_master_and_slave

When setting up a master and slaves it is important that the internal clock has the same configuration. In almost all unix systems the internal clock is in UTC by default. On Windows system it may be in local time. This should be changed to be in UTC if one uses MariaDB with replication!

In addition to being in UTC, one should ensure that the clocks are synchronized. If not properly synchronized, there is a gap between the values of the clocks between different machines. MariaDB replication takes this into account by comparing the master and slave clocks when the slave connects to the master. The difference is `clock_difference_between_master_and_slave`.

This page is licensed: CC BY-SA / Gnu FDL

Parallel Replication

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Some writes, replicated from the primary can be executed in parallel (simultaneously) on the replica. Note that for parallel replication to work, both the primary and replica need to be MariaDB 10.0.5 or later.

Parallel Replication Overview

MariaDB replication in general takes place in three parts:

- Replication events are read from the primary by the IO thread and queued in the relay log.
- Replication events are fetched one at a time by the SQL thread from the relay log
- Each event is applied on the replica to replicate all changes done on the primary.

Before MariaDB 10, the third step was also performed by the SQL thread; this meant that only one event could execute at a time, and replication was essentially single-threaded. Since MariaDB 10, the third step can optionally be performed by a pool of separate replication worker threads, and thereby potentially increase replication performance by applying multiple events in parallel.

How to Enable Parallel Replica

To enable, specify slave-parallel-threads=# in your my.cnf file as an argument to mysql.
Parallel replication can in addition be disabled on a per-multi-source connection by setting @@connection_name.slave-parallel-mode to "none".

The value (#) of slave_parallel_threads specifies how many threads will be created in a pool of worker threads used to apply events in parallel for *all* your replicas (this includes multi-source replication). If the value is zero, then no worker threads are created, and old-style replication is used where events are applied inside the SQL thread. Usually the value, if non-zero, should be at least two times the number of multi-source primary connections used. It makes little sense to use only a single worker thread for one connection; this will incur some overhead in inter-thread communication between the SQL thread and the worker thread, but with just a single worker thread event cannot be applied in parallel anyway.

`slave-parallel-threads=#` is a dynamic variable that can be changed without restarting mysqld. All replicas' connections must however be stopped when changing the value.

Configuring the Replica Parallel Mode

Parallel replication can be in-order or out-of-order:

- In-order executes transactions in parallel but orders the commit step of the transactions to happen in the exact same order as on the primary. Transactions are only executed in parallel to the extent that this can be automatically verified to be possible without any conflicts. This means that the use of parallelism is completely transparent to the application.
- Out-of-order can execute and commit transactions in different order on the replica than originally on the primary. This means that the application must be tolerant to seeing updates occur in different order. The application is also responsible for ensuring that there are no conflicts between transactions that are replicated out-of-order. Out-of-order is only used in GTID mode and only when explicitly enabled by the application, using the replication domain that is part of the GTID.

In-Order Parallel Replication

Optimistic Mode of In-Order Parallel Replication

Optimistic mode of in-order parallel replication provides a lot of opportunities for parallel apply on the replica while still preserving exact transaction semantics from the point of view of applications. It is the default mode from MariaDB 10.5.1.

Optimistic mode of in-order parallel replication can be configured by setting the slave_parallel_mode system variable to `optimistic` on the replica.

Any transactional DML (INSERT/UPDATE/DELETE) is allowed to run in parallel, up to the limit of @@slave_domain_parallel_threads. This may cause conflicts on the replica, e.g. if two transactions try to modify the same row. Any such conflict is detected, and the latter of the two transactions is rolled back, allowing the former to proceed. The latter transaction is then re-tried once the former has completed.

The term "optimistic" is used for this mode, because the server optimistically assumes that few conflicts will occur, and that the extra work spent rolling back and retrying conflicting transactions is justified from the gain from running most transactions in parallel.

There are a few heuristics to try to avoid needless conflicts. If a transaction executed a row lock wait on the primary, it will not be run in parallel on the replica. Transactions can also be marked explicitly as potentially conflicting on the primary, by setting the variable @@skip_parallel_replication. More such heuristics may be added in later MariaDB versions. There is a further –slave-parallel-mode called "aggressive", where these heuristics are disabled, allowing even more transactions to be applied in parallel.

Non-transactional DML and DDL is not safe to optimistically apply in parallel, as it cannot be rolled back in case of conflicts. Thus, in optimistic mode, non-transactional (such as MyISAM) updates are not applied in parallel with earlier events (it is however possible to apply a MyISAM update in parallel with a later InnoDB update). DDL statements are not applied in parallel with any other transactions, earlier or later.

The different kind of transactions can be identified in the output of mariadb-binlog. For example:

```
#150324 13:06:26 server id 1  end_log_pos 6881  GTID 0-1-42 ddl
...
#150324 13:06:26 server id 1  end_log_pos 7816  GTID 0-1-47
...
#150324 13:06:26 server id 1  end_log_pos 8177  GTID 0-1-49 trans
/*!100101 SET @@session.skip_parallel_replication=1*//*!*/;
...
#150324 13:06:26 server id 1  end_log_pos 9836  GTID 0-1-59 trans waited
```

GTID 0-1-42 is marked as being DDL. GTID 0-1-47 is marked as being non-transactional DML, while GTID 0-1-49 is transactional DML (seen on the “trans” keyword). GTID 0-1-49 was additionally run with @@skip_parallel_replication set on the primary. GTID 0-1-59 is transactional DML that had a row lock wait when run on the primary (the “waited” keyword).

Aggressive Mode of In-Order Parallel Replication

Aggressive mode of in-order parallel replication is very similar to optimistic mode. The main difference is that the replica does not consider whether transactions conflicted on the primary when deciding whether to apply the transactions in parallel.

Aggressive mode of in-order parallel replication can be configured by setting the slave_parallel_mode system variable to `aggressive` on the replica.

Conservative Mode of In-Order Parallel Replication

Conservative mode of in-order parallel replication uses the group commit on the primary to discover potential for parallel apply of events on the replica. If two transactions commit together in a group commit on the primary, they are written into the binlog with the same commit id. Such events are certain to not conflict with each other, and they can be scheduled by the parallel replication to run in different worker threads.

Conservative mode of in-order parallel replication is the default mode until MariaDB 10.5.0, but it can also be configured by setting the slave_parallel_mode system variable to `conservative` on the replica.

Two transactions that were committed separately on the primary can potentially conflict (eg. modify the same row of a table). Thus, the worker that applies the second transaction will not start immediately but wait until the first transaction begins the commit step; at this point it is safe to start the second transaction, as it can no longer disrupt the execution of the first one.

Here is example output from mariadb-binlog that shows how GTID events are marked with commit id. The GTID 0-1-47 has no commit id and cannot run in parallel. The GTIDs 0-1-48 and 0-1-49 have the same commit id 630, and can thus replicate in parallel with one another on a replica:

```
#150324 12:54:24 server id 1  end_log_pos 20052     GTID 0-1-47 trans
...
#150324 12:54:24 server id 1  end_log_pos 20212     GTID 0-1-48 cid=630 trans
...
#150324 12:54:24 server id 1  end_log_pos 20372     GTID 0-1-49 cid=630 trans
```

In either case, when the two transactions reach the point where the low-level commit happens and commit order is determined, the two commits are sequenced to happen in the same order as on the primary, so that operation is transparent to applications.

The opportunities for parallel replication on replicas can be highly increased if more transactions are committed in a group commit on the primary. This can be tuned using the binlog_commit_wait_count and binlog_commit_wait_usec variables. If for example the application can tolerate up to 50 milliseconds extra delay for transactions on the primary, one can set `binlog_commit_wait_usec=50000` and `binlog_commit_wait_count=20` to get up to 20 transactions at a time available for replication in parallel. Care must however be taken to not set

`binlog_commit_wait_usec` too high, as this could cause significant slowdown for applications that run a lot of small transactions serially one after the other.

Note that even if there is no parallelism available from the primary group commit, there is still an opportunity for speedup from in-order parallel replication, since the actual commit steps of different transactions can run in parallel. This can be particularly effective on a replica with binlog enabled (log_slave_updates=1), and more so if replica is configured to be crash-safe (sync_binlog=1 and innodb_flush_log_at_trx_commit=1), as this makes group commit possible on the replica.

Minimal Mode of In-Order Parallel Replication

Minimal mode of in-order parallel replication _only_allows the commit step of transactions to be applied in parallel; all other steps are applied serially.

Minimal mode of in-order parallel replication can be configured by setting the slave_parallel_mode system variable to `minimal` on the replica.

Out-of-Order Parallel Replication

Out-of-order parallel replication happens (only) when using GTID mode, when GTIDs with different replication domains are used. The replication domain is set by the DBA/application using the variable `gtid_domain_id`.

Two transactions having GTIDs with different domain_id are scheduled to different worker threads by parallel replication and are allowed to execute completely independently from each other. It is the responsibility of the application to only set different domain_ids for transactions that are truly independent, and are guaranteed to not conflict with each other. The application must also be able to work correctly even though the transactions with different domain_id are seen as committing in different order between the replica and the primary, and between different replicas.

Out-of-order parallel replication can potentially give more performance gain than in-order parallel replication, since the application can explicitly give more opportunities for running transactions in parallel than what the server can determine on its own automatically.

One simple but effective usage is to run long-running statements, such as ALTER TABLE, in a separate replication domain. This allows replication of other transactions to proceed uninterrupted:

```
SET SESSION gtid_domain_id=1
ALTER TABLE t ADD INDEX myidx(b)
SET SESSION gtid_domain_id=0
```

Normally, a long-running ALTER TABLE or other query will stall all following transactions, causing the replica to become behind the primary as least as long time as it takes to run the long-running query. By using out-of-order parallel replication by setting the replication domain id, this can be avoided. The DBA/application must ensure that no conflicting transactions will be replicated while the ALTER TABLE runs.

Another common opportunity for out-of-order parallel replication comes in connection with multi-source replication. Suppose we have two different primaries M1 and M2, and we are using multi-source replication to have S1 as a replica of both M1 and M2. S1 will apply events received from M1 in parallel with events received from M2. If we now have a third-level replica S2 that replicates from S1 as primary, we want S2 to also be able to apply events that originated on M1 in parallel with events that originated on M2. This can be achieved with out-of-order parallel replication, by setting `gtid_domain_id` different on M1 and M2.

Note that there are no special restrictions on what operations can be replicated in parallel using out-of-order; such operations can be on the same database/schema or even on the same table. The only restriction is that the operations must not conflict, that is they must be able to be applied in any order and still end up with the same result.

When using out-of-order parallel replication, the current replica position in the primary's binlog becomes multi-dimensional - each replication domain can have reached a different point in the primary binlog at any one time. The current position can be seen from the variable `gtid_slave_pos`. When the replica is stopped, restarted, or switched to replicate from a different primary using CHANGE MASTER, MariaDB automatically handles restarting each replication domain at the appropriate point in the binlog.

Out-of-order parallel replication is disabled when –slave-parallel-mode=minimal (or none).

Checking Worker Thread Status in SHOW PROCESSLIST

The worker threads will be listed as "system user" in SHOW PROCESSLIST. Their state will show the query they are currently working on, or it can show one of these:

- "Waiting for work from main SQL threads". This means that the worker thread is idle, no work is available for it at the moment.
- "Waiting for prior transaction to start commit before starting next transaction". This means that the previous batch of transactions that committed together on the primary primary has to complete first. This worker thread is waiting for that to happen before it can start working on the following batch.
- "Waiting for prior transaction to commit". This means that the transaction has been executed by the worker thread. In order to ensure in-order commit, the worker thread is waiting to commit until the previous transaction is ready to commit before it.

Expected Performance Gain

Here is an article showing up to ten times improvement when using parallel replication: 18435.html.

Configuring the Maximum Size of the Parallel Replica Queue

The slave_parallel_max_queued system variable can be used to configure the maximum size of the parallel replica queue. This system variable is only meaningful when parallel replication is configured (i.e. when slave_parallel_threads > `0`).

When parallel replication is used, the SQL thread will read ahead in the relay logs, queueing events in memory while looking for opportunities for executing events in parallel. The slave_parallel_max_queued system variable sets a limit for how much memory it will use for this.

The configured value of the slave_parallel_max_queued system variable is actually allocated for each worker thread, so the total allocation is actually equivalent to the following:

slave_parallel_max_queued * slave_parallel_threads

If this value is set too high, and the replica is far (eg. gigabytes of binlog) behind the primary, then the SQL thread can quickly read all of that and fill up memory with huge amounts of binlog events faster than the worker threads can consume them.

On the other hand, if set too low, the SQL thread might not have sufficient space for queuing enough events to keep the worker threads busy, which could reduce performance. In this case, the SQL thread will have the thread state that states `Waiting for room in worker thread event queue`. For example:

```
+----+-------------+-----------+------+---------+--------+-----------------------------------------------
+------------------+----------+
| Id | User        | Host      | db   | Command | Time   | State
| Info             | Progress |
+----+-------------+-----------+------+---------+--------+-----------------------------------------------
+------------------+----------+
|  3 | system user |           | NULL | Connect |    139 | closing tables
| NULL             |    0.000 |
|  4 | system user |           | NULL | Connect |    139 | Waiting for work from SQL thread
| NULL             |    0.000 |
|  6 | system user |           | NULL | Connect | 264274 | Waiting for master to send event
| NULL             |    0.000 |
| 10 | root        | localhost | NULL | Sleep   |     43 |
| NULL             |    0.000 |
| 21 | system user |           | NULL | Connect |     45 | Waiting for room in worker thread event queue
| NULL             |    0.000 |
| 54 | root        | localhost | NULL | Query   |      0 | init
| SHOW PROCESSLIST |    0.000 |
+----+-------------+-----------+------+---------+--------+-----------------------------------------------
+------------------+----------+
```

The slave_parallel_max_queued system variable does not define a hard limit, since the binary log events that are currently executing always need to be held in-memory. This means that at least two events per worker thread can always be queued in-memory, regardless of the value of slave_parallel_threads.

Usually, the slave_parallel_threads system variable should be set large enough that the SQL thread is able to read far enough ahead in the binary log to exploit all possible parallelism. In normal operation, the replica will hopefully not be too far behind, so there will not be a need to queue much data in-memory. The slave_parallel_max_queued system variable could be set fairly high (e.g. a few hundred kilobytes) to not limit throughput. It should just be set low enough that total allocation of the parallel replica queue will not cause the server to run out of memory.

Configuration Variable slave_domain_parallel_threads

The pool of replication worker threads is shared among all multi-source primary connections, and among all replication domains that can replicate in parallel using out-of-order.

If one primary connection or replication domain is currently processing a long-running query, it is possible that it will allocate all the worker threads in the pool, only to have them wait for the long-running query to complete, stalling any other primary connection or replication domain, which will have to wait for a worker thread to become free.

This can be avoided by setting slave_domain_parallel_threads to a number that is lower than `slave_parallel_threads`. When set different from zero, each replication domain in one primary connection can reserve at most that many worker threads at any one time, leaving the rest (up to the value of slave_parallel_threads) free for other primary connections or replication domains to use in parallel.

The `slave_domain_parallel_threads` variable is dynamic and can be changed without restarting the server; all replicas must be stopped while changing it, though.

Implementation Details

The implementation is described in MDEV-4506.

See Also

- Better Parallel Replication for MariaDB and MySQL (MariaDB.com blog)
- Evaluating MariaDB & MySQL Parallel Replication Part 2: Slave Group Commit (MariaDB.com blog)

This page is licensed: CC BY-SA / Gnu FDL

Semisynchronous Replication

Description

Standard MariaDB replication is asynchronous, but MariaDB also provides a semisynchronous replication option. The feature is built into the server and is always available. In versions prior to MariaDB 10.3, it was a separate plugin that needed to be installed.

With regular asynchronous replication, replicas request events from the primary's binary log whenever the replicas are ready. The primary does not wait for a replica to confirm that an event has been received.

With fully synchronous replication, all replicas are required to respond that they have received the events. See Galera Cluster.

Semisynchronous replication waits for just one replica to acknowledge that it has received and logged the events.

Semisynchronous replication therefore comes with some negative performance impact, but increased data integrity. Since the delay is based on the roundtrip time to the replica and back, this delay is minimized for servers in close proximity over fast networks.

Semisynchronous replication is built into the server. See MDEV-13073 for more information.

Enabling Semisynchronous Replication

Semisynchronous replication can be enabled by setting the relevant system variables on the primary and the replica.

If a server needs to be able to switch between acting as a primary and a replica, then you can enable both the primary and replica system variables on the server. For example, you might need to do this if MariaDB MaxScale is being used to enable auto-failover or switchover with MariaDB Monitor.

Enabling Semisynchronous Replication on the Primary

Semisynchronous replication can be enabled on the primary by setting the rpl_semi_sync_master_enabled system variable to `ON`. It can be set dynamically with SET GLOBAL. For example:

```
SET GLOBAL rpl_semi_sync_master_enabled=ON;
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
rpl_semi_sync_master_enabled=ON
```

Enabling Semisynchronous Replication on the Replica

Semisynchronous replication can be enabled on the replica by setting the rpl_semi_sync_slave_enabled system variable to `ON`. It can be set dynamically with SET GLOBAL. For example:

```
SET GLOBAL rpl_semi_sync_slave_enabled=ON;
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
rpl_semi_sync_slave_enabled=ON
```

When switching between semisynchronous replication and asynchronous replication on a replica with replica IO threads already running, the replica I/O thread will need to be restarted. For example:

```
STOP SLAVE IO_THREAD;
START SLAVE IO_THREAD;
```

If this is not done, then the replica IO thread will continue to use the previous setting.

Configuring the Primary Timeout

In semisynchronous replication, only after the events have been written to the relay log and flushed does the replica acknowledge receipt of a transaction's events. If the replica does not acknowledge the transaction before a certain amount of time has passed, then a timeout occurs and the primary switches to asynchronous replication. This will be reflected in the primary's error log with messages like the following:

```
[Warning] Timeout waiting for reply of binlog (file: mariadb-1-bin.000002, pos: 538), semi-sync up to
file , position 0.
[Note] Semi-sync replication switched OFF.
```

When this occurs, the Rpl_semi_sync_master_status status variable will be switched to `OFF`.

When at least one semisynchronous replica catches up, semisynchronous replication is resumed. This will be reflected in the primary's error log with messages like the following:

```
[Note] Semi-sync replication switched ON with replica (server_id: 184137206) at (mariadb-1-bin.000002,
215076)
```

When this occurs, the Rpl_semi_sync_master_status status variable will be switched to `ON`.

The number of times that semisynchronous replication has been switched off can be checked by looking at the value of the Rpl_semi_sync_master_no_times status variable.

If you see a lot of timeouts like this in your environment, then you may want to change the timeout period. The timeout period can be changed by setting the rpl_semi_sync_master_timeout system variable. It can be set dynamically with SET GLOBAL. For example:

```
SET GLOBAL rpl_semi_sync_master_timeout=20000;
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
rpl_semi_sync_master_timeout=20000
```

To determine a good value for the rpl_semi_sync_master_timeout system variable, you may want to look at the values of the Rpl_semi_sync_master_net_avg_wait_time and Rpl_semi_sync_master_tx_avg_wait_time status variables.

Configuring the Primary Wait Point

In semisynchronous replication, there are two potential points at which the primary can wait for the replica acknowledge the receipt of a transaction's events. These two wait points have different advantages and disadvantages.

The wait point is configured by the rpl_semi_sync_master_wait_point system variable. The supported values are:

- `AFTER_SYNC`
- `AFTER_COMMIT`

> When using the InnoDB-based Binary Log (`--binary-storage-engine=innodb`), the `AFTER_SYNC` wait point is not supported. Only `AFTER_COMMIT` is available, since the traditional two-phase commit between the binary log

> and the InnoDB storage engine is no longer used.

It can be set dynamically with SET GLOBAL. For example:

```
SET GLOBAL rpl_semi_sync_master_wait_point='AFTER_SYNC';
```

It can also be set in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
rpl_semi_sync_master_wait_point=AFTER_SYNC
```

When this variable is set to `AFTER_SYNC`, the primary performs the following steps:

> The `AFTER_SYNC` wait point is only supported with the traditional binlog implementation and is not available when the
> InnoDB-based Binary Log is enabled.

1. Prepares the transaction in the storage engine.
2. Syncs the transaction to the binary log.
3. Waits for acknowledgement from the replica.
4. Commits the transaction to the storage engine.
5. Returns an acknowledgement to the client.

The effects of the `AFTER_SYNC` wait point are:

- All clients see the same data on the primary at the same time; after acknowledgement by the replica and after being committed to the storage engine on the primary.
- If the primary crashes, then failover should be lossless, because all transactions committed on the primary would have been replicated to the replica.
- However, if the primary crashes, then its binary log may also contain events for transactions that were prepared by the storage engine and written to the binary log, but that were never actually committed by the storage engine. As part of the server's automatic crash recovery process, the server may recover these prepared transactions when the server is restarted. This could cause the "old" crashed primary to become inconsistent with its former replicas when they have been reconfigured to replace the old primary with a new one. The old primary in such a scenario can be re-introduced only as a semisync replica. To recover it safely, start the server with `--init-rpl-role=SLAVE`. This tells the server its role in the replication topology, so that post-crash recovery discards transactions proven not to be committed and the server will not have extra transactions (MDEV-21117, MDEV-33465). Before MDEV-33465, this recovery was instead deduced from the semisync variables (`rpl_semi_sync_slave_enabled = ON`), which did not work reliably in mixed topologies; setting `--init-rpl-role=SLAVE` is now required. The server's binlog gets truncated to discard transactions proven not to be committed, in any of their branches if they are multi-engine. Truncation does not occur though when there exists a non-transactional group of events beyond the truncation position in which case recovery reports an error. When the semisync replica recovery can't be carried out, the crashed primary may need to be rebuilt.

When this variable is set to `AFTER_COMMIT`, the primary performs the following steps:

1. Prepares the transaction in the storage engine.
2. Syncs the transaction to the binary log.
3. Commits the transaction to the storage engine.
4. Waits for acknowledgement from the replica.
5. Returns an acknowledgement to the client.

The effects of the `AFTER_COMMIT` wait point are:

- Other clients may see the committed transaction before the committing client.
- If the primary crashes, then failover may involve some data loss, because the primary may have committed transactions that had not yet been acknowledged by the replicas.

Failover Implications

System administrators implementing a semi-synchronous replication fail-over strategy must understand the distinction between the `AFTER_SYNC` and `AFTER_COMMIT` wait points. This choice has irreversible implications for the server recovery process: if a pre-crash semi-sync master is brought back online as a post-crash semi-sync slave, the recovery process will truncate any transactions from the binary log that were not committed in the storage engine.

Conceptually, the two configurations differ in which server may be logically ahead of the other:

- `AFTER_SYNC` : The slave may be ahead of the master.
- `AFTER_COMMIT` : The master will always be ahead of the slave.

In either configuration, data loss is possible if the logically behind server is promoted to master after a crash.

Recommended Fail-Over Strategy

To prevent data loss, the fail-over strategy should align with the configured wait point:

- `AFTER_COMMIT` configurations: Always wait for a failed master to be brought back online to resume its role as master.
- `AFTER_SYNC` configurations: Demote the failed master to a slave of a newly-promoted master.

Recovery Procedure for Guaranteed Consistency

If neither of the above strategies is feasible, the following procedure ensures data consistency after a crash:

1. Configure all servers in the topology with `SET GLOBAL read_only=1`. This prevents the failed master from accidentally resuming its master role
2. After the master fails, select any node (hereafter node A) as the new master candidate. Choosing the most up-to-date node will minimize the time required to complete this process
3. On another node (hereafter node B), run `SELECT @@gtid_current_pos`.
4. Configure node A as a slave of node B using `CHANGE MASTER TO` (if this replication channel does not already exist).
5. On node A, run `START SLAVE UNTIL` with the GTID value retrieved in step 3
6. Repeat steps 3–5 for each remaining node in the topology, using node A as the slave in each iteration.
7. Once all nodes have been processed, node A is guaranteed to be the furthest-progressed node in the topology and is safe to promote as the new master.
8. On node A, run `SET GLOBAL read_only=0`.
9. Configure all other nodes as slaves of node A using `CHANGE MASTER TO` followed by `START SLAVE`.

System Variables

`rpl_semi_sync_master_enabled`

- Description: Set to `ON` to enable semi-synchronous replication primary. Disabled by default.
- Command line: `--rpl-semi-sync-master-enabled[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`

- Default Value: `OFF`

`rpl_semi_sync_master_timeout`

- Description: The timeout value, in milliseconds, for semi-synchronous replication in the primary. If this timeout is exceeded in waiting on a commit for acknowledgement from a replica, the primary will revert to asynchronous replication.
  - When a timeout occurs, the Rpl_semi_sync_master_status status variable will also be switched to `OFF`.
  - See Configuring the Primary Timeout for more information.
- Command line: `--rpl-semi-sync-master-timeout[=#]`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10000` (10 seconds)
- Range: `0` to `18446744073709551615`

`rpl_semi_sync_master_trace_level`

- Description: The tracing level for semi-sync replication. Four levels are defined:
  - `1`: General level, including for example time function failures.
  - `16`: More detailed level, with more verbose information.
  - `32`: Net wait level, including more information about network waits.
  - `64`: Function level, including information about function entries and exits.
- Command line: `--rpl-semi-sync-master-trace-level[=#]`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `32`
- Range: `0` to `18446744073709551615`

`rpl_semi_sync_master_wait_no_slave`

- Description: If all replicas have disconnected from the primary (i.e. Rpl_semi_sync_master_clients is 0), this variable controls whether or not the primary will still suspend the next transaction's (and any others that commit within Rpl_semi_sync_master_timeout duration) commit phase to wait for a replica to connect or reconnect, and send an ACK. If set to `ON`, the default, the replica count may drop to zero, and the primary will still wait for the timeout period for the next transaction, and any more that commit after this transaction within the semi-sync timeout duration. If no ACK is received in this time, the primary will revert to asynchronous replication. If set to `OFF`, the primary will revert to asynchronous replication as soon as the replica count drops to zero.
- Command line: `--rpl-semi-sync-master-wait-no-slave[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`rpl_semi_sync_master_wait_point`

- Description: Whether the transaction should wait for semi-sync acknowledgement after having synced the binlog (`AFTER_SYNC`), or after having committed in storage engine (`AFTER_COMMIT`, the default).
  - When this variable is set to `AFTER_SYNC`, the primary performs the following steps:

1. Prepares the transaction in the storage engine.
2. Syncs the transaction to the binary log.
3. Waits for acknowledgement from the replica.
4. Commits the transaction to the storage engine.
5. Returns an acknowledgement to the client.

- When this variable is set to `AFTER_COMMIT`, the primary performs the following steps:
  1. Prepares the transaction in the storage engine.
  2. Syncs the transaction to the binary log.
  3. Commits the transaction to the storage engine.
  4. Waits for acknowledgement from the replica.
  5. Returns an acknowledgement to the client.
- In MariaDB 10.1.2 and before, this system variable does not exist. However, in those versions, the primary waits for the acknowledgement from replicas at a point that is equivalent to `AFTER_COMMIT`.
- See Configuring the Primary Wait Point for more information.

- Command line: `--rpl-semi-sync-master-wait-point=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `AFTER_COMMIT`
- Valid Values: `AFTER_SYNC`, `AFTER_COMMIT`

`rpl_semi_sync_slave_delay_master`

- Description: Only write primary info file when ack is needed.
- Command line: `--rpl-semi-sync-slave-delay-master[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rpl_semi_sync_slave_enabled`

- Description: Set to `ON` to enable semi-synchronous replication replica. Disabled by default.
- Command line: `--rpl-semi-sync-slave-enabled[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`rpl_semi_sync_slave_kill_conn_timeout`

- Description: Timeout for the mysql connection used to kill the replica io_thread's connection on primary. This timeout comes into play when stop slave is executed.
- Command line: `--rpl-semi-sync-slave-kill-conn-timeout[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`

- Default Value: `5`
- Range: `0` to `4294967295`

`rpl_semi_sync_slave_trace_level`

- Description: The tracing level for semi-sync replication. The levels are the same as for rpl_semi_sync_master_trace_level.
- Command line: `--rpl-semi-sync-slave-trace_level[=#]`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `32`
- Range: `0` to `18446744073709551615`

Options

`init-rpl-role`

- From MariaDB 10.6.19, MariaDB 10.11.9, MariaDB 11.1.6, MariaDB 11.2.5, MariaDB 11.4.3 and MariaDB 11.5.2, changes the condition for semi-sync recovery to truncate the binlog to instead use this option, when set to SLAVE. This avoids a possible error state where the replica's state is ahead of the primaries. See -init-rpl-role.

`rpl-semi-sync_master`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--rpl-semi-sync-master=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`
- Removed: MariaDB 10.3.3

`rpl-semi-sync_slave`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL

PLUGIN while the server is running.

- See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.

- Command line: `--rpl-semi-sync-slave=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`
- Removed: MariaDB 10.3.3

Status Variables

For a list of status variables added when the plugin is installed, see Semisynchronous Replication Plugin Status Variables.

This page is licensed: CC BY-SA / Gnu FDL

Row-based Replication With No Primary Key

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

MariaDB improves on row-based replication (see binary log formats) of tables which have no primary key but do have some other index. This is based in part on the original Percona patch `row_based_replication_without_primary_key.patch`, with some additional fixes and enhancements.

When row-based replication is used with UPDATE or DELETE, the replica needs to locate each replicated row based on the value in columns. If the table contains at least one index; an index lookup will be used (otherwise a table scan is needed for each row, which is extremely inefficient for all but the smallest table and generally to be avoided).

In MariaDB, the slave will try to choose a good index among any available:

- The primary key is used, if there is one.
- Else, the first unique index without `NULL`-able columns is used, if there is one.
- Else, a choice is made among any normal indexes on the table (e.g. a FULLTEXT index is not considered).

The choice of which of several non-unique indexes to use is based on the cardinality of indexes; the one that is most selective (has the smallest average number of rows per distinct tuple of column values) is preferred. Note that for this choice to be effective, for most storage engines (like MyISAM, InnoDB) it is necessary to make sure ANALYZE TABLE has been run on the slave, otherwise statistics about index cardinality is not available. In the absence of index cardinality, the first unique index is chosen, if any, else the first non-unique index.

See Also

-

This page is licensed: CC BY-SA / Gnu FDL

Unsafe Statements for Statement-Based Replication

A safe statement is generally deterministic; in other words the statement will always produce the same result. For example, an INSERT statement producing a random number will most likely produce a different result on the primary than on the replica, and so cannot be replicated safely.

When an unsafe statement is run, the current binary logging format determines how the server responds.

- If the binary logging format is statement-based, unsafe statements generate a warning and are logged normally.
- If the binary logging format is mixed, unsafe statements are logged using the row-based format, while safe statements use the statement-based format.
- If the binary logging format is row-based, all statements are logged normally, and the distinction between safe and unsafe is not made.

MariaDB tries to detect unsafe statements. When an unsafe statement is issued, a warning similar to the following is produced:

```
Note (Code 1592): Unsafe statement written to the binary log using statement format since
  BINLOG_FORMAT = STATEMENT. The statement is unsafe because it uses a LIMIT clause. This
  is unsafe because the set of rows included cannot be predicted.
```

MariaDB also issues this warning for some classes of statements that are safe.

Unsafe Statements

The following statements are regarded as unsafe:

- INSERT … ON DUPLICATE KEY UPDATE statements using tables with multiple primary or unique keys, as the order that the keys are checked in, and which affect the rows chosen to update, is not deterministic. The warning about this was removed, because we always check keys in the same order on the primary and replica if the primary and replica are using the same storage engine.
- INSERT-DELAYED. These statements are inserted in an indeterminate order.
- INSERT … SELECT for a table that has an AUTO_INCREMENT column.
- INSERTs on tables with a composite primary key that has an AUTO_INCREMENT column that isn't the first column of the composite key.
- When a table has an AUTO_INCREMENT column and a trigger or stored procedure executes an UPDATE statement against the table.
- UPDATE statements that use LIMIT, since the order of the returned rows is unspecified. This applies even to statements using an ORDER BY clause, which are deterministic (a known bug). However, `LIMIT 0` is an exception to this rule (see MDEV-6170), and these statements are safe for replication.
- When using a user-defined function.
- Statements using any of the following functions, which can return different results on the replica:
  - CURRENT_ROLE()
  - CURRENT_USER()
  - FOUND_ROWS()
  - GET_LOCK()
  - IS_FREE_LOCK()
  - IS_USED_LOCK()
  - JSON_TABLE()
  - LOAD_FILE()
  - MASTER_POS_WAIT()
  - RAND()
  - RANDOM_BYTES()
  - RELEASE_ALL_LOCKS()
  - RELEASE_LOCK()
  - ROW_COUNT()
  - SESSION_USER()

    - SLEEP()
    - SYSDATE()
    - SYSTEM_USER()
    - USER()
    - UUID()
    - UUID_SHORT().
- Statements which refer to log tables, since these may differ across servers.
- Statements which refer to self-logging tables. Statements following a read or write to a self-logging table within a transaction are also considered unsafe.
- Statements which refer to system variables (there are a few exceptions).
- LOAD DATA INFILE statements.
- Non-transactional reads or writes that execute after transactional reads within a transaction.
- If row-based logging is used for a statement, and the session executing the statement has any temporary tables, row-based logging is used for the remaining statements until the temporary table is dropped. This is because temporary tables can't use row-based logging, so if it is used due to one of the above conditions, all subsequent statements using that table are unsafe. The server deals with this situation by treating all statements in the session as unsafe for statement-based logging until the temporary table is dropped.

Safe Statements

The following statements are not deterministic, but are considered safe for binary logging and replication:

- CONNECTION_ID()
- CURDATE()
- CURRENT_DATE()
- CURRENT_TIME()
- CURRENT_TIMESTAMP()
- CURTIME()
- LAST_INSERT_ID()
- LOCALTIME()
- LOCALTIMESTAMP()
- NOW()
- UNIX_TIMESTAMP()
- UTC_DATE()
- UTC_TIME()
- UTC_TIMESTAMP()

Isolation Levels

Even when using safe statements, not all transaction isolation levels are safe with statement-based or mixed binary logging. While the `REPEATABLE READ` and `SERIALIZABLE` isolation levels can be used with both statement- and row-based replication, the `READ COMMITTED` and `READ UNCOMMITTED` isolation levels only support row-based replication, as with them isolation between transactions is not guaranteed at all, and different transaction orders on a replica, or when doing point-in-time recovery, would lead to different results than on the original master.

This restriction does not apply if only non-transactional storage engines are used.

See Also

- Replication and Foreign Keys

- Replicating Temporary Tables

This page is licensed: CC BY-SA / Gnu FDL

Replication Filters

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Replication filters allow users to configure replicas to intentionally skip certain events.

Binary Log Filters for Replication Primaries

MariaDB provides options that can be used on a replication primary to restrict local changes to specific databases from getting written to the binary log, which also determines whether any replicas replicate those changes.

Binary Log Filter Options

The following options are available, and they are evaluated in the order that they are listed below. If there are conflicting settings, `binlog_do_db` prevails. Before MariaDB 11.2.0, they are only available as options; from MariaDB 11.2.0 they are also available as system variables.

`binlog_do_db`

The binlog_do_db option allows you to configure a replication primary to write statements and transactions affecting databases that match a specified name into its binary log. Since the filtered statements or transactions will not be present in the binary log, its replicas will not be able to replicate them.

This option will **not** work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.

This option cannot be set dynamically.

When setting it on the command-line or in a server option group in an option file, the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times. For example:

```
[mariadb]
...
binlog_do_db=db1
binlog_do_db=db2
```

This will tell the primary to do the following:

- Write statements and transactions affecting the database named db1 into the binary log.
- Write statements and transactions affecting the database named db2 into the binary log.
- Don’t write statements and transactions affecting any other databases into the binary log.

`binlog_ignore_db`

The binlog_ignore_db option allows you to configure a replication primary to **not** write statements and transactions affecting databases that match a specified name into its binary log. Since the filtered statements or transactions will not be present in the binary log, its replicas will not be able to replicate them.

This option will **not** work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.

This option cannot be set dynamically.

When setting it on the command-line or in a server option group in an option file, the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times. For example:

```
[mariadb]
...
binlog_ignore_db=db1
binlog_ignore_db=db2
```

This will tell the primary to do the following:

- Don't write statements and transactions affecting the database named db1 into the binary log.
- Don't write statements and transactions affecting the database named db2 into the binary log.
- Write statements and transactions affecting any other databases into the binary log.

The binlog_ignore_db option is effectively ignored if the binlog_do_db option is set, so those two options should not be set together.

Replication Filters for Replicas

MariaDB provides options and system variables that can be used on used on a replica to filter events replicated in the binary log.

Replication Filter Options

The following options and system variables are available, and they are evaluated in the order that they are listed below. If there are conflicting settings, the respective *replicate_do_* prevails.

`replicate_rewrite_db`

The replicate_rewrite_db option (and, from MariaDB 10.11, system variable), allows you to configure a replica to rewrite database names. It uses the format `primary_database->replica_database`. If a replica encounters a binary log event in which the default database (i.e. the one selected by the USE statement) is `primary_database`, then the replica will apply the event in `replica_database` instead.

This option will **not** work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.

This option only affects statements that involve tables. This option does not affect statements involving the database itself, such as CREATE DATABASE, ALTER DATABASE, and DROP DATABASE.

This option's rewrites are evaluated *before* any other replication filters configured by the `replicate_*` system variables.

Statements that use table names qualified with database names do not work with other replication filters such as replicate_do_table.

Until MariaDB 10.11, this option could not be set dynamically.

When setting it on the command-line or in a server option group in an option file, the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times. For example:

```
[mariadb]
...
replicate_rewrite_db=db1->db3
replicate_rewrite_db=db2->db4
```

This will tell the replica to do the following:

- If a binary log event is encountered in which the default database was db1, then apply the event in db3 instead.
- If a binary log event is encountered in which the default database was db2, then apply the event in db4 instead.

See Configuring Replication Filter Options with Multi-Source Replication for how to configure this system variable with multi-source replication.

`replicate_do_db`

The replicate_do_db system variable allows you to configure a replica to apply statements and transactions affecting databases that match a specified name.

This system variable will **not** work with cross-database updates with statement-based logging or when using mixed-based logging and the statement is logged statement based. For statement-based replication, only the default database (that is, the one selected by USE) is considered, not any explicitly mentioned tables in the query.
See the Statement-Based Logging section for more information.

When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.

When setting it dynamically, it is not possible to specify database names that contain commas. If you need to specify database names that contain commas, then you will need to specify them by either providing the command-line options or configuring them in a server option group in an option file when the server is started.

When setting it dynamically, the replica threads must be stopped. For example:

```
STOP SLAVE;
SET GLOBAL replicate_do_db='db1,db2';
START SLAVE;
```

When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times. For example:

```
[mariadb]
...
replicate_do_db=db1
replicate_do_db=db2
```

This will tell the replica to do the following:

- Replicate statements and transactions affecting the database named db1.
- Replicate statements and transactions affecting the database named db2.
- Ignore statements and transactions affecting any other databases.

See Configuring Replication Filter Options with Multi-Source Replication for how to configure this system variable with multi-source replication.

`replicate_ignore_db`

The replicate_ignore_db system variable allows you to configure a replica to ignore statements and transactions affecting databases that match a specified name.

This system variable will **not** work with cross-database updates with statement-based logging or when using mixed-based logging and the statement is logged statement based. For statement-based replication, only the default database (that is, the one selected by USE) is considered, not any explicitly mentioned tables in the query.
See the Statement-Based Logging section for more information.

When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.

When setting it dynamically, it is not possible to specify database names that contain commas. If you need to specify names or patterns that contain commas, then you will need to specify them by either providing the command-line options or configuring them in a server option group in an option file when the server is started.

When setting it dynamically, the replica threads must be stopped. For example:

```
STOP SLAVE;
SET GLOBAL replicate_ignore_db='db1,db2';
START SLAVE;
```

When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times. For example:

```
[mariadb]
...
replicate_ignore_db=db1
replicate_ignore_db=db2
```

This will tell the replica to do the following:

- Ignore statements and transactions affecting databases named db1.
- Ignore statements and transactions affecting databases named db2.
- Replicate statements and transactions affecting any other databases.

The replicate_ignore_db system variable is effectively ignored if the replicate_do_db system variable is set, so those two system variables should not be set together.

See Configuring Replication Filter Options with Multi-Source Replication for how to configure this system variable with multi-source replication.

`replicate_do_table`

The replicate_do_table system variable allows you to configure a replica to apply statements and transactions that affect tables that match a specified name. The table name is specified in the format: `dbname.tablename`.

This system variable will **not** work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.

This option only affects statements that involve tables. This option does not affect statements involving the database itself, such as CREATE DATABASE, ALTER DATABASE, and DROP DATABASE.

When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.

When setting it dynamically, it is not possible to specify database or table names or patterns that contain commas. If you need to specify database or table names that contain commas, then you will need to specify them by either providing the command-line options or configuring them in a server option group in an option file when the server is started.

When setting it dynamically, the replica threads must be stopped. For example:

```
STOP SLAVE;
SET GLOBAL replicate_do_table='db1.tab,db2.tab';
START SLAVE;
```

When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times. For example:

```
[mariadb]
...
replicate_do_table=db1.tab
replicate_do_table=db2.tab
```

This will tell the replica to do the following:

- Replicate statements and transactions affecting tables in databases named db1 and which are named tab.
- Replicate statements and transactions affecting tables in databases named db2 and which are named tab.
- Ignore statements and transactions affecting any other tables.

See Configuring Replication Filter Options with Multi-Source Replication for how to configure this system variable with multi-source replication.

`replicate_ignore_table`

The replicate_ignore_table system variable allows you to configure a replica to ignore statements and transactions that affect tables that match a specified name. The table name is specified in the format: `dbname.tablename`.

This system variable will **not** work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.

When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.

When setting it dynamically, it is not possible to specify database or table names that contain commas. If you need to specify database or table names that contain commas, then you will need to specify them by either providing the command-line options or configuring them in a server option group in an option file when the server is started.

When setting it dynamically, the replica threads must be stopped. For example:

```
STOP SLAVE;
SET GLOBAL replicate_ignore_table='db1.tab,db2.tab';
START SLAVE;
```

When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times. For example:

```
[mariadb]
...
replicate_ignore_table=db1.tab
replicate_ignore_table=db2.tab
```

This will tell the replica to do the following:

- Ignore statements and transactions affecting tables in databases named db1 and which are named tab.
- Ignore statements and transactions affecting tables in databases named db2 and which are named tab.
- Replicate statements and transactions affecting any other tables.

The replicate_ignore_table system variable is effectively ignored if either the replicate_do_table system variable or the replicate_wild_do_table system variable is set, so the replicate_ignore_table system variable should not be used with those two system variables.

See Configuring Replication Filter Options with Multi-Source Replication for how to configure this system variable with multi-source replication.

`replicate_wild_do_table`

The replicate_wild_do_table system variable allows you to configure a replica to apply statements and transactions that affect tables that match a specified wildcard pattern.

The wildcard pattern uses the same semantics as the LIKE operator. This means that the following characters have a special meaning:

- `_` - The `_` character matches any single character.
- `%` - The `%` character matches zero or more characters.
- `\` - The `\` character is used to escape the other special characters in cases where you need the literal character.

This system variable will work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.

The system variable does filter databases, tables, views and triggers.

The system variable does not filter stored procedures, stored functions, and events. The replicate_do_db system variable will need to be used to filter those.

If the table name pattern for a filter is just specified as `%`, then all tables in the database will be matched. In this case, the filter will also affect certain database-level statements, such as CREATE DATABASE, ALTER DATABASE and DROP DATABASE.

When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.

When setting it dynamically, it is not possible to specify database or table names or patterns that contain commas. If you need to specify database or table names or patterns that contain commas, then you will need to specify them by either providing the command-line options or configuring them in a server option group in an option file when the server is started.

When setting it dynamically, the replica threads must be stopped. For example:

```
STOP SLAVE;
SET GLOBAL replicate_wild_do_table='db%.tab%,app1.%';
START SLAVE;
```

When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times. For example:

```
[mariadb]
...
replicate_wild_do_table=db%.tab%
replicate_wild_do_table=app1.%
```

This will tell the replica to do the following:

- Replicate statements and transactions affecting tables in databases that start with db and whose table names start with tab.
- Replicate statements and transactions affecting the database named app1.
- Ignore statements and transactions affecting any other tables and databases.

See Configuring Replication Filter Options with Multi-Source Replication for how to configure this system variable with multi-source replication.

`replicate_wild_ignore_table`

The replicate_wild_ignore_table system variable allows you to configure a replica to ignore statements and transactions that affect tables that match a specified wildcard pattern.

The wildcard pattern uses the same semantics as the LIKE operator. This means that the following characters have a special meaning:

- `_` - The `_` character matches any single character.
- `%` - The `%` character matches zero or more characters.
- `\` - The `\` character is used to escape the other special characters in cases where you need the literal character.

This system variable will work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.

The system variable does filter databases, tables, views and triggers.

The system variable does not filter stored procedures, stored functions, and events. The replicate_ignore_db system variable will need to be used to filter those.

If the table name pattern for a filter is just specified as `%`, then all tables in the database will be matched. In this case, the filter will also affect certain database-level statements, such as CREATE DATABASE, ALTER DATABASE and DROP DATABASE.

When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.

When setting it dynamically, it is not possible to specify database or table names or patterns that contain commas. If you need to specify database or table names or patterns that contain commas, then you will need to specify them by either providing the command-line options or configuring them in a server option group in an option file when the server is started.

When setting it dynamically, the replica threads must be stopped. For example:

```
STOP SLAVE;
SET GLOBAL replicate_wild_ignore_table='db%.tab%,app1.%';
START SLAVE;
```

When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times. For example:

```
[mariadb]
...
replicate_wild_ignore_table=db%.tab%
replicate_wild_ignore_table=app1.%
```

This will tell the replica to do the following:

- Ignore statements and transactions affecting tables in databases that start with db and whose table names start with tab.
- Ignore statements and transactions affecting all the tables in the database named app1.
- Replicate statements and transactions affecting any other tables and databases.

The replicate_ignore_table system variable is effectively ignored if either the replicate_do_table system variable or the replicate_wild_do_table system variable is set, so the replicate_ignore_table system variable should not be used with those two system variables.

See Configuring Replication Filter Options with Multi-Source Replication for how to configure this system variable with multi-source replication.

Configuring Replication Filter Options with Multi-Source Replication

How you configure replication filters with multi-source replication depends on whether you are configuring them dynamically or whether you are configuring them in a server option group in an option file.

Setting Replication Filter Options Dynamically with Multi-Source Replication

The usage of dynamic replication filters changes somewhat when multi-source replication is in use. By default, the variables are addressed to the default connection, so in a multi-source environment, the required connection needs to be specified. There are two ways to do this.

Prefixing the Replication Filter Option with the Connection Name

One way to change a replication filter for a multi-source connection is to explicitly specify the name when changing the filter. For example:

```
STOP SLAVE 'gandalf';
SET GLOBAL gandalf.replicate_do_table='database1.table1,database1.table2,database1.table3';
START SLAVE 'gandalf';
```

Changing the Default Connection

Alternatively, the default connection can be changed by setting the default_master_connection system variable, and then the replication filter can be changed in the usual fashion. For example:

```
SET default_master_connection = 'gandalf';
STOP SLAVE;
SET GLOBAL replicate_do_table='database1.table1,database1.table2,database1.table3';
START SLAVE;
```

Setting Replication Filter Options in Option Files with Multi-Source Replication

If you are using multi-source replication and if you would like to make this filter persist server restarts by adding it to a server option group in an option file, then the option file can also include the connection name that each filter would apply to. For example:

```
[mariadb]
...
gandalf.replicate_do_db=database1
saruman.replicate_do_db=database2
```

CHANGE MASTER Options

The CHANGE MASTER statement has a few options that can be used to filter certain types of binary log events.

`IGNORE_SERVER_IDS`

The IGNORE_SERVER_IDS option for `CHANGE MASTER` can be used to configure a replica to ignore binary log events that originated from certain servers. Filtered binary log events will not get logged to the replica's relay log, and they will not be applied by the replica.

`DO_DOMAIN_IDS`

The DO_DOMAIN_IDS option for `CHANGE MASTER` can be used to configure a replica to only apply binary log events if the transaction's GTID is in a specific gtid_domain_id value. Filtered binary log events will not get logged to the replica's relay log, and they will not be applied by the replica.

`IGNORE_DOMAIN_IDS`

The IGNORE_DOMAIN_IDS option for `CHANGE MASTER` can be used to configure a replica to ignore binary log events if the transaction's GTID is in a specific gtid_domain_id value. Filtered binary log events will not get logged to the replica's relay log, and they will not be applied by the replica.

Replication Filters and Binary Log Formats

The way that a replication filter is interpreted can depend on the binary log format.

Statement-Based Logging

When an event is logged in its statement-based format, many replication filters that affect a database will test the filter against the default database (i.e. the one selected by the USE statement). This applies to the following replication filters:

- binlog_do_db
- binlog_ignore_db
- replicate_rewrite_db
- replicate_do_db
- replicate_ignore_db

When an event is logged in its statement-based format, many replication filters that affect a table will test the filter against the table in the default database (i.e. the one selected by the USE statement). This applies to the following replication filters:

- replicate_do_table
- replicate_ignore_table

This means that cross-database updates **do not work** with replication filters and statement-based binary logging. For example, if replicate_do_table=db2.tab were set, then the following would not replicate with statement-based binary logging:

```
USE db1;
INSERT INTO db2.tab VALUES (1);
```

If you need to be able to support cross-database updates with replication filters and statement-based binary logging, then you should use the following replication filters:

- replicate_wild_do_table
- replicate_wild_ignore_table

Row-Based Logging

When an event is logged in its row-based format, many replication filters that affect a database will test the filter against the database that is actually affected by the event.

Similarly, when an event is logged in its row-based format, many replication filters that affect a table will test the filter against the table in the database that is actually affected by the event.

This means that cross-database updates work with replication filters and statement-based binary logging.

Keep in mind that DDL statements are always logged to the binary log in statement-based format, even when the binlog_format system variable is set to `ROW`. This means that the notes mentioned in Statement-Based Logging always apply to DDL.

Replication Filters and Galera Cluster

When using Galera cluster, replication filters should be used with caution. See Configuring MariaDB Galera Cluster: Replication Filters for more details.

See Also

- Dynamic replication filters — our wheel will be square!

This page is licensed: CC BY-SA / Gnu FDL

Replication Threads

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

MariaDB's replication implementation requires several types of threads.

Threads on the Primary

The primary usually only has one type of replication-related thread: the binary log dump thread.

If semisynchronous replication is enabled, then the primary also has an ACK receiver thread.

Binary Log Dump Thread

The binary log dump thread runs on the primary and dumps the binary log to the replica. This thread can be identified by running the SHOW PROCESSLIST statement and finding the thread where the thread command is `"Binlog Dump"`.

The primary creates a separate binary log dump thread for each replica connected to the primary. You can identify which replicas are connected to the primary by executing the SHOW SLAVE HOSTS statement.

Binary Log Dump Threads and the Shutdown Process

When a primary server is shut down and it goes through the normal shutdown process, the primary kills client threads in random order. By default, the primary also considers its binary log dump threads to be regular client threads. As a consequence, the binary log dump threads can be killed while client threads still exist, and this means that data can be written on the primary during a normal shutdown that won't be replicated. This is true even if semi-synchronous replication is being used. Data is not lost; it is stored in the primary server's binary log. The replicas on reconnection, after the primary server restarts, will resume at the exact position they were killed off during the primary shutdown. No data is lost.

In MariaDB 10.4 and later, this problem can be solved by shutting down the server using either the mariadb-admin utility or the SHUTDOWN command and providing a special option.

For example, this problem can be solved by shutting down the server with the mariadb-admin utility and by providing the `--wait-for-all-slaves` option to the utility and by executing the `shutdown` command with the utility:

```
mariadb-admin --wait-for-all-slaves shutdown
```

Or this problem can be solved by shutting down the server with the SHUTDOWN command and by providing the `WAIT FOR ALL SLAVES` option to the command:

```
SHUTDOWN WAIT FOR ALL SLAVES;
```

When one of these special options is provided, the server only kills its binary log dump threads after all client threads have been killed, and it only completes the shutdown after the last binary log has been sent to all connected replicas.

In MariaDB 10.4 and later, it is still not possible to enable this behavior by default. This means that this behavior is currently inaccessible when shutting down the server using tools like systemd or sysVinit.

In MariaDB 10.3 and before, it is recommended to manually switchover replicas to a new primary before shutting down the old primary.

ACK Receiver Thread

When semisynchronous replication is enabled, semisynchronous replicas send acknowledgements (ACKs) to their primary to confirm that they have received some transaction. The primary creates an ACK receiver thread to receive these ACKs.

Threads on the Replica

The replica has three types of replication-related threads: the replica I/O thread, the replica SQL thread, and worker threads, which are only applicable when parallel replication is in use.

When multi-source replication is in use, each independent replication connection has its own replica threads of each type.

Replica I/O Thread

The replica's I/O thread receives the binary log events from the primary and writes them to its relay log.

Binary Log Position

The binary log position of the replica's I/O thread can be checked by executing the SHOW SLAVE STATUS statement. It will be shown as the `Master_Log_File` and `Read_Master_Log_Pos` columns.

The binary log position of the replica's I/O thread can be set by setting the MASTER_LOG_FILE and MASTER_LOG_POS options with the CHANGE MASTER statement.

The binary log position of the replica's I/O thread and the values of most other CHANGE MASTER options are written to either the default `master.info` file or the file that is configured by the master_info_file option. The replica's I/O thread keeps this binary log position updated as it downloads events only when the MASTER_USE_GTID option is set to `NO`. Otherwise the file is not updated on a per event basis. See CHANGE MASTER TO: Option Persistence for more information.

Replica SQL Thread

The replica's SQL thread reads events from the relay log. What it does with them depends on whether parallel replication is in use. If parallel replication is not in use, then the SQL thread applies the events to its local copy of the data. If parallel replication is in use, then the SQL thread hands off the events to its worker threads to apply in parallel.

Relay Log Position

The relay log position of the replica's SQL thread can be checked by executing the SHOW SLAVE STATUS statement. It will be shown as the `Relay_Log_File` and `Relay_Log_Pos` columns.

The relay log position of the replica's SQL thread can be set by setting the RELAY_LOG_FILE and RELAY_LOG_POS options with the CHANGE MASTER statement.

The relay log position of the replica's SQL thread is written to either the default `relay-log.info` file or the file that is configured by the relay_log_info_file system variable. The replica's SQL thread keeps this relay log position

updated as it applies events. See CHANGE MASTER TO: Option Persistence for more information.

Binary Log Position

The corresponding binary log position of the current relay log position of the replica's SQL thread can be checked by executing the SHOW SLAVE STATUS statement. It will be shown as the `Relay_Master_Log_File` and `Exec_Master_Log_Pos` columns.

GTID Position

If the replica is replicating binary log events that contain GTIDs, then the replica's's SQL thread will write every GTID that it applies to the mysql.gtid_slave_pos table. This GTID can be inspected and modified through the gtid_slave_pos system variable.

If the replica has the log_slave_updates system variable enabled and if the replica has the binary log enabled, then every write by the replica's SQL thread will also go into the replica's binary log. This means that GTIDs of replicated transactions would be reflected in the value of the gtid_binlog_pos system variable.

See CHANGE MASTER TO: GTID Persistence for more information.

Worker Threads

When parallel replication is in use, then the SQL thread hands off the events to its worker threads to apply in parallel.

This page is licensed: CC BY-SA / Gnu FDL

Replication and Foreign Keys

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Replication is based upon the binary log. However, cascading deletes or updates based on foreign key relations are an internal mechanism, and are not written to the binary log.

Because of this, an identical statement run on the master and the slave may result in different outcomes if the foreign key relations are not identical on both master and slave This could be the case if the storage engine on one supports cascading deletes (e.g. InnoDB) and the storage engine on the other does not (e.g. MyISAM), or the one has specified a foreign key relation, and the other hasn't.

Take the following example:

```
CREATE TABLE employees (
    x INT PRIMARY KEY,
    name VARCHAR(10)
) ENGINE = InnoDB;

CREATE TABLE children (
    y INT PRIMARY KEY,
    f INT,
    name VARCHAR(10),
    FOREIGN KEY fk (f) REFERENCES employees (x)
        ON DELETE CASCADE
) ENGINE = InnoDB;
```

The slave, however, has been set up without InnoDB support, and defaults to MyISAM, so the foreign key restrictions are not in place.

```
INSERT INTO employees VALUES (1, 'Yaser'), (2, 'Prune');

INSERT INTO children VALUES (1, 1, 'Haruna'), (2, 1, 'Hera'), (3, 2, 'Eva');
```

At this point, the slave and the master are in sync:

```
SELECT * FROM employees;
+---+-------+
| x | name  |
+---+-------+
| 1 | Yaser |
| 2 | Prune |
+---+-------+
2 rows in set (0.00 sec)

SELECT * FROM children;
+---+------+--------+
| y | f    | name   |
+---+------+--------+
| 1 |    1 | Haruna |
| 2 |    1 | Hera   |
| 3 |    2 | Eva    |
+---+------+--------+
```

However, after:

```
DELETE FROM employees WHERE x=1;
```

there are different outcomes on the slave and the master.

On the master, the cascading deletes have taken effect:

```
SELECT * FROM children;
+---+------+------+
| y | f    | name |
+---+------+------+
| 3 |    2 | Eva  |
+---+------+------+
```

On the slave, the cascading deletes did not take effect:

```
SELECT * FROM children;
+---+------+--------+
| y | f    | name   |
+---+------+--------+
| 1 |    1 | Haruna |
| 2 |    1 | Hera   |
| 3 |    2 | Eva    |
+---+------+--------+
```

This page is licensed: CC BY-SA / Gnu FDL

Replication and Binary Log System Variables

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Overview

This page lists system variables that are related to binary logging and replication.

See Server System Variables for a complete list of system variables and instructions on setting them, as well as System variables for global transaction ID.

Also see mariadbd replication options for related options that are not system variables (such as binlog_do_db and binlog_ignore_db).

Variable Descriptions

`auto_increment_increment`

- Description: The increment for all AUTO_INCREMENT values on the server, by default `1`. Intended for use in primary-to-primary replication.
- Command line: `--auto-increment-increment[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `65535`

`auto_increment_offset`

- Description: The offset for all AUTO_INCREMENT values on the server, by default `1`. Intended for use in primary-to-primary replication. Should be not be larger than auto_increment_increment. See AUTO_INCREMENT#Replication.
- Command line: `--auto-increment-offset[=#]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `65535`

`binlog_alter_two_phase`

- Description: When set, split ALTER at binary logging into two statements: `START ALTER` and `COMMIT` / `ROLLBACK ALTER`. The `ON` setting is recommended for long-running `ALTER TABLE` statements, so it could start on replica before its actual execution on primary.
- Command line: `--binlog-alter-two-phase[={0|1}]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.8.1

`binlog_annotate_row_events`

- Description: This option tells the primary to write annotate_rows_events to the binary log.
- Command line: `--binlog-annotate-row-events[={0|1}]`
- Scope: Global, Session

- Dynamic: Yes
- Data Type: boolean
- Default Value: `ON`

`binlog_cache_size`

- Description: If the binary log is active, this variable determines the size in bytes, per-connection, of the cache holding a record of binary log changes during a transaction. A separate variable, binlog_stmt_cache_size, sets the upper limit for the statement cache. The binlog_cache_disk_use and binlog_cache_use server status variables indicates whether this variable needs to be increased (you want a low ratio of binlog_cache_disk_use to binlog_cache_use).
- Command line: `--binlog-cache-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `32768`
- Range - 32 bit: `4096` to `4294967295`
- Range - 64 bit: `4096` to `18446744073709547520`

`binlog_checksum`

- Description: Specifies the type of `BINLOG_CHECKSUM_ALG` for log events in the binary log.
- Command line:
  - `--binlog-checksum=name`
  - `--binlog-checksum=[0|1]`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `CRC32`
- Valid Values: `NONE` (`0`), `CRC32` (`1`)

`binlog_commit_wait_count`

- Description: Configures the behavior of group commit for the binary log, which can help increase transaction throughput and is used to enable conservative mode of in-order parallel replication.
  With group commit for the binary log, the server can delay flushing a committed transaction into binary log until the given number of transactions are ready to be flushed as a group. However, the delay is not longer than the value set by binlog_commit_wait_usec. The default value of `0` means that no delay is introduced.
  Setting this value can reduce I/O on the binary log and give an increased opportunity for parallel apply on the replica when conservative mode of in-order parallel replication is enabled, but too high a value decreases the transaction throughput. By monitoring the status variable binlog_group_commit_trigger_count, it is possible to see how often this is occurring.
- If the server detects that one of the committing transactions T1 holds an InnoDB row lock that another transaction T2 is waiting for, the commit completes immediately without further delay. This helps avoid losing throughput when many transactions need conflicting locks. This often makes it safe to use this option without losing throughput on a replica with conservative mode of in-order parallel replication, provided the value of slave_parallel_threads is sufficiently high.
- Command line: `--binlog-commit-wait-count=#]`
- Scope: Global

- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`

`binlog_commit_wait_usec`

- Description: Configures the behavior of group commit for the binary log, which can help increase transaction throughput and is used to enable conservative mode of in-order parallel replication.
  With group commit for the binary log, the server can delay flushing a committed transaction into binary log until the transaction has waited the configured number of microseconds. By monitoring the status variable binlog_group_commit_trigger_timeout, it is possible to see how often group commits are made due to `binlog_commit_wait_usec`. As soon as the number of pending commits reaches binlog_commit_wait_count, the wait is terminated, though. Thus, this setting only takes effect if `binlog_commit_wait_count` is non-zero.
- Command line: `--binlog-commit-wait-usec#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `100000`
- Range: `0` to `18446744073709551615`

`binlog_direct_non_transactional_updates`

- Description: Replication inconsistencies can occur due when a transaction updates both transactional and non-transactional tables and the updates to the non-transactional tables are visible before being written to the binary log. This is because, to preserve causality, the non-transactional statements are written to the transaction cache, which is only flushed on commit. Setting `binlog_direct_non_transactional_updates` to `1` (`0` is default) causes non-transactional tables to be written straight to the binary log, rather than the transaction cache. This setting has no effect when row-based binary logging is used, as it requires statement-based logging. See binlog_format. Use with care, and only in situations where no dependencies exist between the non-transactional and transactional tables, for example, inserting into a non-transactional table based upon the results of a `SELECT` from a transactional table.
- Command line: `--binlog-direct-non-transactional-updates[=value]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF (0)`

`binlog_directory`

- Description: Specifies the directory in which to store binary log files.
- Command line: `--binlog-directory=/path/to/dir`
- Scope: Global
- Dynamic: No (requires server restart)
- Data type: `string`
- Default Value: `''` (binary log files are stored in the data directory)

`binlog_do_db`

- Description: This option allows you to configure a replication primary to write statements and transactions affecting databases that match a specified name into its binary log. Since the filtered statements or transactions are not be present in the binary log, its replicas are not be able to replicate them.
  - This option does not work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
  - Until MariaDB 11.2.0, only available as an option, not a system variable. This option can not be set dynamically.
  - When setting it on the command-line or in a server option group in an option file, the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times.
  - See Replication Filters for more information.
- Command line: `--binlog-do-db=#`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: NULL
- Introduced: MariaDB 11.2.0 (as a system variable)

`binlog_expire_logs_seconds`

- Description: If non-zero, binary logs are purged after `binlog_expire_logs_seconds` seconds. Possible purges happen at startup and at binary log rotation. From MariaDB 10.6.1, `binlog_expire_logs_seconds` and expire_logs_days are forms of aliases, such that changes to one automatically reflect in the other.
- Command line: `--binlog-expire-logs-seconds=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `8553600`
- Introduced: MariaDB 10.6.1

`binlog_file_cache_size`

- Description: Size of in-memory cache that is allocated when reading binary log and relay log files.
- Command line: `--binlog-file-cache-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `16384`
- Range: `8192` to `18446744073709551615`

`binlog_format`

- Description: Determines whether replication is row-based, statement-based or mixed. Statement-based was the default until MariaDB 10.2.3. Be careful of changing the binary log format when a replication environment is already running. See Binary Log Formats. A replica applies any events it gets from the primary, regardless of the binary log format. `binlog_format` only applies to normal (not replicated) updates.
- Command line: `--binlog-format=format`

- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `MIXED`
- Valid Values: `ROW`, `STATEMENT` or `MIXED`

`binlog_gtid_index`

- Description: Enable the creation of a GTID index for every binlog file, and the use of such index for speeding up GTID lookup in the binlog. See Binlog indexing.
- Command line: `--binlog-gtid-index{=0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 11.4

`binlog_gtid_index_page_size`

- Description: Page size to use for the binlog GTID index. See Binlog indexing.
- Command line: `--binlog-gtid-index-page-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `4096`
- Range: `64` to `16777216`
- Introduced: MariaDB 11.4

`binlog_gtid_index_span_min`

- Description: Control sparseness of the binlog GTID index. If set, at most one index record is added for every `N` bytes of binlog file written, to reduce the size of the index. Normally, this does not need tuning. See Binlog indexing.
- Command line: `--binlog-gtid-index-span-min=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `65536`
- Range: `1` to `1073741824`
- Introduced: MariaDB 11.4

`binlog_ignore_db`

- Description: This option allows you to configure a replication primary to not write statements and transactions affecting databases that match a specified name into its binary log. Since the filtered statements or transactions are not be present in the binary log, its replicas are not able to replicate them.
  - This option does not work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
  - When setting it on the command-line or in a server option group in an option file, the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple

times.
    - See Replication Filters for more information.
- Command line: `--binlog-ignore-db=name`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: NULL
- Introduced: MariaDB 11.2.0

`binlog_large_commit_threshold`

- Description: Increases transaction concurrency for large transactions (for instance, those with sizes larger than this value) by using the large transaction's cache file as a new binary log and rotating the active binary log to the large transaction's cache file at commit time. This avoids the default commit logic that copies the transaction cache data to the end of the active binary log file while holding a lock that prevents other transactions from binlogging.
- Command line: `--binlog-large-commit-threshold=val`
- Scope: Global
- Dynamic: Yes
- Data Type: `bigint unsigned`
- Default Value: `134217728`
- Range: `10485760` to `18446744073709551615`
- Introduced: MariaDB 11.7

`binlog_legacy_event_pos`

- Description: Fill in the `end_log_pos` field of *all* events in the binlog, even when doing so costs performance. Can be used in case some old application needs it for backwards compatibility. Setting this option can hurt binlog scalability.
  Limitations: Checksums cannot be pre-computed when binlog encryption is enabled, because encryption relies on correct `end_log_pos` to provide part of the nonce[31]/IV[32].

  Checksum pre-computation is also disabled for WSREP/Galera, as it uses events differently in its write sets.
- Command line: `--binlog-legacy-event-pos{=0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 11.4

`binlog_optimize_thread_scheduling`

- Description: Run fast part of group commit in a single thread, to optimize kernel thread scheduling. On by default. Disable to run each transaction in group commit in its own thread, which can be slower at very high concurrency. This option is mostly for testing one algorithm versus another, and it should not normally be necessary to change it. Deprecated in MariaDB 11.7, as the option was initially added to provide a safe alternative for the newly added binlog group commit logic, such that when `0`, it would disable a leader thread from performing the binlog write for all transactions that are a part of the group commit. Problems related to the

binlog group commit optimization are expected to be addressed by now, so the option has been deprecated and is going to be removed in future.

- Command line: `--binlog-optimize-thread-scheduling` or `--skip-binlog-optimize-thread-scheduling`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Deprecated: MariaDB 11.7

`binlog_row_event_fragment_threshold`

- Description: When a `Rows_log_event` exceeds this threshold, it is fragmented into multiple `Partial_rows_log_event` events in the binary log, each of it configured to maximum size. That is, all `Partial_rows_log_event` events up to the last in the group have this configured maximum size, and the last event takes the remaining size. This is relevant for events that would surpass the `slave_max_allowed_packet` length when sending to the replica, and thereby a sensible value would reflect the replica's configured `slave_max_allowed_packet` size.
- Command line: `--binlog-row-event-fragment-threshold`
- Scope: Global
- Dynamic: Yes
- Data Type: `INT unsigned`
- Default Value: 1GB
- Introduced: MariaDB 12.3

`binlog_row_event_max_size`

- Description: The maximum size of a row-based binary log event in bytes. Rows are grouped into events smaller than this size if possible. The value has to be a multiple of 256.
- Command line: `--binlog-row-event-max-size=val`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `8192`
- Range: `256` to `4294967040` (in multiples of 256)
- Introduced: MariaDB 11.2.0

`binlog_row_image`

- Description: Controls the logging format in row-based replication. In row-based replication (the variable has no effect with statement-based replication), each row change event contains an image for matching against when choosing the row to be updated, and another image containing the changes. Before the introduction of this variable, all columns were logged for both of these images. In certain circumstances, this is not necessary, and memory, disk and network resources can be saved by partial logging. Note that to safely change this setting from the default, the table being replicated to must contain identical primary key definitions, and columns must be present, in the same order, and use the same data types as the original table. If these conditions are not met, matches may not be correctly determined and updates and deletes may diverge on the replica, with no warnings or errors returned.
  - `FULL`: All columns in the before and after image are logged. This is the default, and the only behavior in earlier versions.

    - `NOBLOB`: mariadbd avoids logging blob and text columns whenever possible (eg, blob column was not changed or is not part of primary key).
    - `MINIMAL`: A PK equivalent (PK columns or full row if there is no PK in the table) is logged in the before image, and only changed columns are logged in the after image.
    - `FULL_NODUP`: All columns are logged in the before image but only changed columns or all columns of inserted record are logged in the after image. This is essentially the same as `FULL`, but takes less space. From MariaDB 11.4.
- Command line: `--binlog-row-image=value`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `FULL`
- Valid Values:
    - <= MariaDB 11.3: `FULL`, `NOBLOB` or `MINIMAL`
    - >=MariaDB 11.4: `FULL`, `NOBLOB`, `MINIMAL` or `FULL_NODUP`

`binlog_row_metadata`

- Description: Controls the format used for binlog metadata logging – *value* is one of the following:
    - `NO_LOG`: No metadata is logged (default).
    - `MINIMAL`: Only metadata required by a replica is logged.
    - `FULL`: All metadata is logged.
        - From MariaDB 12.3, row events are mapped to using column names. This addresses a potential issue in multi-master replication.
- Command line: `--binlog-row-metadata=*value*`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `NO_LOG`
- Valid Values: `NO_LOG`, `MINIMAL`, `FULL`
- Introduced: MariaDB 10.5.0

`binlog_space_limit`

- Description: Alias for max_binlog_total_size.
- Introduced: MariaDB 11.4

`binlog_stmt_cache_size`

- Description: If the binary log is active, this variable determines the size in bytes of the cache holding a record of binary log changes outside of a transaction. The variable binlog_cache_size, determines the cache size for binary log statements inside a transaction. The binlog_stmt_cache_disk_use and binlog_stmt_cache_use server status variables indicates whether this variable needs to be increased (you want a low ratio of `binlog_stmt_cache_disk_use` to `binlog_stmt_cache_use`).
- Command line: `--binlog-stmt-cache-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`

- Default Value: `32768`
- Range - 32 bit: `4096` to `4294967295`
- Range - 64 bit: `4096` to `18446744073709547520`

`binlog_storage_engine`

- Description: Specifies the storage engine that manages the binary log.
- Command line: `--binlog-storage-engine=name`
- Scope: Global
- Dynamic: No (requires server restart)
- Data type: `enum`
- Default Value: None

`create_tmp_table_binlog_formats`

- Description: The binary logging formats under which the primary logs `CREATE TEMPORARY` statements to the binary log. If `CREATE TEMPORARY` is not logged, all usage of the temporary table is logged in `ROW` format. Allowed values are `STATEMENT` or `MIXED`, `STATEMENT`.
- Command line: `--create-tmp-table-binlog-formats=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `STATEMENT`
- Valid Values: `STATEMENT` or `MIXED,STATEMENT`
- Introduced: MariaDB 12.0

`default_master_connection`

- Description: In multi-source replication, specifies which connection is used for commands and variables if you don't specify a connection.
- Command line: None
- Scope: Session
- Dynamic: Yes
- Data Type: `string`
- Default Value: `''` (empty string)

`encrypt_binlog`

- Description: Encrypt binary logs (including relay logs). See Data at Rest Encryption and Encrypting Binary Logs.
- Command line: `--encrypt-binlog[={0|1}]`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`expire_logs_days`

- Description: Number of days after which the binary log can be automatically removed. By default, 0, or no automatic removal. When using replication, should always be set higher than the maximum lag by any replica. Removals take place when the server starts up, when the binary log is flushed, when the next binary log is created after the previous one reaches the maximum size, or when running PURGE BINARY LOGS. Units are

1/1000000 precision (double). `expire_logs_days` and binlog_expire_logs_seconds are forms of aliases, such that changes to one automatically reflect in the other. Some container configs explicitly set `expire_logs_days` to 10, rather than leave it as the default, zero.

- Command line: `--expire-logs-days=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0.000000`
- Range: `0` to `99`

`init_rpl_role`

- Command line: `--init-rpl-role=name`
- Description: Sets the replication role for the server.

  In older versions of MariaDB, a primary server configured with both `rpl_semi_sync_master_enabled=1` and `rpl_semi_sync_slave_enabled=1` could experience data loss after a restart. Upon recovery, the primary might truncate its binary log, dropping transactions that replicas had already received and executed. This caused the replica's `gtid_slave_pos` to be ahead of the primary's `gtid_binlog_pos`, leading to an error state.

  Starting with MariaDB 10.6.19, MariaDB 10.11.9, MariaDB 11.1.6, MariaDB 11.2.5, MariaDB 11.4.3, and MariaDB 11.5.2, the condition for binary log truncation during semi-synchronous recovery has changed:

  - Preventing Data Loss: If you restart a primary server and do not set `--init-rpl-role` to `SLAVE`, the server will not truncate transactions required by the replicas.
  - Requirements: This protection allows you to keep both `rpl_semi_sync_master_enabled` and `rpl_semi_sync_slave_enabled` active on a primary to ensure no transactions are lost during a restart.
- Default Value: `MASTER`
- Valid values: `MASTER` or `SLAVE`
- Introduced (as a variable): MariaDB 13.0.1 (prior, it was only an option you could set in a configuration file). The variable can be queried with `SHOW VARIABLES LIKE 'init_rpl_role'` or `SELECT @@init_rpl_role` (MDEV-38202).

`init_slave`

- Description: Similar to init_connect, but the string contains one or more SQL statements (separated by semicolons) that are executed by a replica server each time the SQL thread starts. These statements are only executed after the acknowledgement is sent to the replica and START REPLICA completes.
- Command line: `--init-slave=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Related variables: init_connect

`innodb_binlog_state_interval`

- Description: The interval (in bytes) between writing GTID state records into the binary log.
- Command line: `--innodb-binlog-state-interval=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: 2097152 (2MB)

`log_bin`

- Description: Whether binary logging is enabled or not. If the `--log-bin` option is used, `log_bin` is set to `ON`, otherwise to `OFF`. If no `name` option is given for `--log-bin`, `datadir/` *`log-basename`* `-bin` or *`datadir`* `/mysql-bin` are used (the latter is used if –log-basename is not specified). We strongly recommend you use either `--log-basename`, or to specify a filename to ensure that replication doesn't stop if the real hostname of the computer changes. The name option can optionally include an absolute path. If no path is specified, the log is written to the data directory. The name can optionally include the file extension; if it does, it is stripped, and only the file basename is used.
- Command line: `--log-bin[=name]`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Related variables: sql_log_bin

`log_bin_basename`

- Description: The full path of the binary log file names, excluding the extension. Its value is derived from the rules specified in `log_bin` system variable. This is a read-only variable only, there is no corresponding configuration file setting or command line option by the same name, use `log_bin` to set the basename path instead.
- Command line: `No commandline option`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: None
- Dynamic: `No`

`log_bin_compress`

- Description: Whether or not the binary log can be compressed. `0` (the default) means no compression. See Compressing Events to Reduce Size of the Binary Log.
- Command line: `--log-bin-compress`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`log_bin_compress_min_len`

- Description: Minimum length of sql statement (in statement mode) or record (in row mode) that can be compressed. See Compressing Events to Reduce Size of the Binary Log.
- Command line: `--log-bin-compress-min-len`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `256`

- Range: `10` to `1024`

`log_bin_index`

- Description: File that holds the names for last binlog files. If –log-basename is also set, `log_bin_index` should be placed after in the config files. Later settings override earlier settings, so `log-basename` override any earlier log file name settings.
- Command line: `--log-bin-index=name`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: None

`log_bin_trust_function_creators`

- Description: Functions and triggers can be dangerous when used with replication. Certain types of functions and triggers may have unintended consequences when the statements are applied on a replica. For that reason, there are some restrictions on the creation of functions and triggers when the binary log is enabled by default, such as:
  - When `log_bin_trust_function_creators` is `OFF` and log_bin is `ON`, CREATE FUNCTION and ALTER FUNCTION statements trigger an error if the function is defined with any of the `NOT DETERMINISTIC`, `CONTAINS SQL` or `MODIFIES SQL DATA` characteristics.
  - This means that when `log_bin_trust_function_creators` is `OFF` and log_bin is `ON`, CREATE FUNCTION and ALTER FUNCTION statements only succeed if the function is defined with any of the `DETERMINISTIC`, `NO SQL`, or `READS SQL DATA` characteristics.
  - When `log_bin_trust_function_creators` is `OFF` and log_bin is `ON`, the SUPER privilege is also required to execute the following statements:
    - CREATE FUNCTION
    - CREATE TRIGGER
    - DROP TRIGGER
  - Setting `log_bin_trust_function_creators` to `ON` removes these requirements around functions characteristics and the SUPER privileges.
  - See Binary Logging of Stored Routines for more information.
- Command line: `--log-bin-trust-function-creators[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`log_slow_slave_statements`

- Description: Log slow statements executed by replica thread to the slow log if it is open.
- Command line: `--log-slow-slave-statements`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`

`log_slave_updates`

- Description: If set to `0`, the default, updates on a replica received from a primary during replication are not logged in the replica's binary log. If set to `1`, they are. The replica's binary log needs to be enabled for this to have an effect. Set to `1` if you want to daisy-chain the replicas.
- Command line: `--log-slave-updates`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`master_info_file`

- Description: The location and name of the file that remembers the primary and where the I/O replication thread is in the primary's binlogs. Defaults to primary.info.
- Command line: `--master-info-file=val`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `master.info`
- Introduced: MariaDB 12.0 (as a system variable, previously it was just an option)

`master_verify_checksum`

- Description: Verify binlog checksums when reading events from the binlog on the primary.
- Command line: `--master-verify-checksum=[0|1]`
- Scope: Global
- Access Type: Can be changed dynamically
- Data Type: `bool`
- Default Value: `OFF (0)`

`max_binlog_cache_size`

- Description: Restricts the size in bytes used to cache a multi-transactional query. If more bytes are required, a `Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage` error is generated. If the value is changed, current sessions are unaffected, only sessions started subsequently. See max_binlog_stmt_cache_size and binlog_cache_size.
- Command line: `--max-binlog-cache-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `18446744073709547520`
- Range: `4096` to `18446744073709547520`

`max_binlog_size`

- Description: If the binary log exceeds this size in bytes after a write, the server rotates it by closing it and opening a new binary log. Single transactions are always stored in the same binary log, so the server waits for open transactions to complete before rotating. This figure also applies to the size of relay logs if max_relay_log_size is set to zero.
- Command line: `--max-binlog-size=#`

- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1073741824` (1GB)
- Range: `4096` to `1073741824` (4KB to 1GB)

`max_binlog_stmt_cache_size`

- Description: Restricts the size used to cache non-transactional statements. See max_binlog_cache_size and binlog_stmt_cache_size.
- Command line: `--max-binlog-stmt-cache-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `18446744073709547520` (64 bit), `4294963200` (32 bit)
- Range: `4096` to `18446744073709547520`

`max_binlog_total_size`

- Description: Maximum space in bytes to use for all binary logs. Extra logs are deleted on server start, log rotation, `FLUSH LOGS` statements, or when writing to binlog. Default is `0`, which means no size restrictions. See also slave_connections_needed_for_purge.
- Command line: `--max-binlog-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `18446744073709551615`
- Introduced: MariaDB 11.4

`max_relay_log_size`

- Description: The replica rotates its relay log if it exceeds this size after a write. If set to `0`, the max_binlog_size setting is used instead. Previously global only, since the implementation of multi-source replication, it can be set per session as well.
- Command line: `--max-relay-log-size=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0`, or `4096 to 1073741824` (4KB to 1GB)

`read_binlog_speed_limit`

- Description: Used to restrict the speed at which a replica can read the binlog from the primary. This can be used to reduce the load on a primary if many replicas need to download large amounts of old binlog files at the same time. The network traffic is restricted to the specified number of kilobytes per second.
- Command line: `--read-binlog-speed-limit=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `0` (no limit)
- Range: `0` to `18446744073709551615`

`relay_log`

- Description: Relay log basename. If not set, the basename of the files is `hostname-relay-bin`, or derived from –log-basename. If –log-basename is also set, `relay_log` should be placed after in the config files. Later settings override earlier settings, so `log-basename` overrides any earlier log file name settings.
- Command line: `--relay-log=file_name`
- Scope: Global
- Dynamic: No
- Data Type: `filename`
- Default Value: `''` (none)

`relay_log_basename`

- Description: The full path of the relay log file names, excluding the extension. Its value is derived from the relay-log variable value.
- Command line: `No commandline option`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: None
- Dynamic: `No`

`relay_log_index`

- Description: Name and location of the relay log index file, the file that keeps a list of the last relay logs. Defaults to hostname-relay-bin.index, or derived from –log-basename. If –log-basename is also set, `relay_log_index` should be placed after in the config files. Later settings override earlier settings, so `log-basename` overrides any earlier log file name settings.
- Command line: `--relay-log-index=name`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: None

`relay_log_info_file`

- Description: Name and location of the file where the `RELAY_LOG_FILE` and `RELAY_LOG_POS` options (i.e. the relay log position) for the CHANGE MASTER statement are written. The replica's SQL thread keeps this relay log position updated as it applies events.
  - See CHANGE MASTER TO: Option Persistence for more information.
- Command line: `--relay-log-info-file=file_name`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `relay-log.info`

`relay_log_purge`

- Description: If set to `1` (the default), relay logs is purged as soon as they are no longer necessary.
- Command line: `--relay-log-purge={0|1}`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `ON`
- Note: In MySQL and in MariaDB before version 10.0.8 this variable was silently changed if you did CHANGE MASTER.

`relay_log_recovery`

- Description: If set to `1` (`0` is default), on startup the replica drops all relay logs that haven't yet been processed, and retrieve relay logs from the primary. Can be useful after the replica has crashed to prevent the processing of corrupt relay logs. relay_log_recovery should always be set together with relay_log_purge. Setting `relay-log-recovery=1` with `relay-log-purge=0` can cause the relay log to be read from files that were not purged, leading to data inconsistencies.
- Command line: `--relay-log-recovery`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`relay_log_space_limit`

- Description: Specifies the maximum space to be used for the relay logs. The IO thread stops until the SQL thread has cleared the backlog. By default `0`, or no limit.
- Command line: `--relay-log-space-limit=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range - 32 bit: `0` to `4294967295`
- Range - 64 bit: `0` to `18446744073709547520`

`replicate_annotate_row_events`

- Description: Tells the replica to reproduce annotate_rows_events received from the primary in its own binary log. This option is sensible only when used in tandem with the log_slave_updates option.
- Command line: `--replicate-annotate-row-events`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`

`replicate_do_db`

- Description: This system variable allows you to configure a replica to apply statements and transactions affecting databases that match a specified name.

    - This system variable does not work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
    - When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.
    - When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times.
    - See Replication Filters for more information.
- Command line: `--replicate-do-db=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `''` (empty)

`replicate_do_table`

- Description: This system variable allows you to configure a replica to apply statements and transactions that affect tables that match a specified name. The table name is specified in the format: `dbname.tablename`.
    - This system variable does not work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
    - When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.
    - When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times.
    - See Replication Filters for more information.
- Command line: `--replicate-do-table=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `''` (empty)

`replicate_events_marked_for_skip`

- Description: Tells the replica whether to replicate events that are marked with the `@@skip_replication` flag. See Selectively skipping replication of binlog events for more information.
- Command line: `--replicate-events-marked-for-skip`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `replicate`
- Valid Values: `REPLICATE`, `FILTER_ON_SLAVE`, `FILTER_ON_MASTER`

`replicate_ignore_db`

- Description: This system variable allows you to configure a replica to ignore statements and transactions affecting databases that match a specified name.
    - This system variable does not work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
    - When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.

    - When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times.
    - See Replication Filters for more information.

- Command line: `--replicate-ignore-db=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `''` (empty)

`replicate_ignore_table`

- Description: This system variable allows you to configure a replica to ignore statements and transactions that affect tables that match a specified name. The table name is specified in the format: `dbname.tablename`.
    - This system variable does not work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
    - When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.
    - When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times.
    - See Replication Filters for more information.
- Command line: `--replicate-ignore-table=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `''` (empty)

`replicate_rewrite_db`

- Description: This option allows you to configure a replica to rewrite database names. It uses the format `primary_database->replica_database`. If a replica encounters a binary log event in which the default database (i.e. the one selected by the USE statement) is `primary_database`, then the replica applies the event in `replica_database` instead.
    - This option does not work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
    - This option only affects statements that involve tables. This option does not affect statements involving the database itself, such as CREATE DATABASE, ALTER DATABASE, and DROP DATABASE.
    - When setting it on the command-line or in a server option group in an option file, the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times.
    - See Replication Filters for more information.
- Command line: `--replicate-rewrite-db=primary_database->replica_database`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `''` (empty)
- Introduced: MariaDB 10.11.0

`replicate_same_server_id`

- Description: In replication, if set to 1, do not skip events having our server id. Default value is `0` (to break infinite loops in circular replication). Can't be set to `1` if –log-slave-updates is used.
- Command line: `--replicate-same-server-id[={0|1}]`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 12.0 (as a system variable, previously just an option)

`replicate_wild_do_table`

- Description: This system variable allows you to configure a replica to apply statements and transactions that affect tables that match a specified wildcard pattern. The wildcard pattern uses the same semantics as the LIKE operator.
  - This system variable works with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
  - When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.
  - When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times.
  - See Replication Filters for more information.
- Command line: `--replicate-wild-do-table=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `''` (empty)

`replicate_wild_ignore_table`

- Description: This system variable allows you to configure a replica to ignore statements and transactions that affect tables that match a specified wildcard pattern. The wildcard pattern uses the same semantics as the LIKE operator.
  - This system variable work with cross-database updates with statement-based logging. See the Statement-Based Logging section for more information.
  - When setting it dynamically with SET GLOBAL, the system variable accepts a comma-separated list of filters.
  - When setting it on the command-line or in a server option group in an option file, the system variable does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the system variable multiple times.
  - See Replication Filters for more information.
- Command line: `--replicate-wild-ignore-table=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `string`
- Default Value: `''` (empty)

`report_host`

- Description: The host name or IP address the replica reports to the primary when it registers. If left unset, the replica does not register itself. Reported by SHOW REPLICA HOSTS. Note that it is not sufficient for the primary to simply read the IP of the replica from the socket once the replica connects. Due to NAT and other routing issues, that IP may not be valid for connecting to the replica from the primary or other hosts.
- Command line: `--report-host=host_name`
- Scope: Global
- Dynamic: No
- Data Type: `string`

`report_password`

- Description: Replica password reported to the primary when it registers. Reported by SHOW REPLICA HOSTS if `--show-slave-auth-info` is set. This password has no connection with user privileges or with the replication user account password.
- Command line: `--report-password=password`
- Scope: Global
- Dynamic: No
- Data Type: `string`

`report_port`

- Description: The command line option sets the TCP/IP port for connecting to the replica that is reported to the replicating primary during the replica's registration. Viewing the variable shows this value.
- Command line: `--report-port=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `65535`

`report_user`

- Description: Replica's account user name reported to the primary when it registers. Reported by SHOW REPLICA HOSTS if `--show-slave-auth-info` is set. This username has no connection with user privileges or with the replication user account.
- Command line: `--report-user=name`
- Scope: Global
- Dynamic: No
- Data Type: `string`

`server_id`

- Description: This system variable is used with MariaDB replication to identify unique primary and replica servers in a topology. This system variable is also used with the binary log to determine which server a specific transaction originated on.
  - When MariaDB replication is used with standalone MariaDB Server, each server in the replication topology must have a unique `server_id` value.
  - When MariaDB replication is used with MariaDB Galera Cluster, see Using MariaDB Replication with MariaDB Galera Cluster: Setting server_id on Cluster Nodes for more information on how to set the

`server_id` values.

- Command line: `--server-id =#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `1` to `4294967295`

`show_slave_auth_info`

- Description: Show user and password in SHOW REPLICA HOSTS on this primary.
- Command line: `--show-slave-auth-info[={0|1}]`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 12.0 (as a system variable, previously it was just an option)

`skip_parallel_replication`

- Description: If set when a transaction is written to the binlog, parallel apply of that transaction is avoided on a replica where slave_parallel_mode is not `aggressive`. Can be used to avoid unnecessary rollback and retry for transactions that are likely to cause a conflict if replicated in parallel. See parallel replication.
- Command line: None
- Scope: Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`skip_replication`

- Description: Changes are logged into the binary log with the @@skip_replication flag set. Such events are not be replicated by replica that run with `--replicate-events-marked-for-skip` set different from its default of `REPLICATE`. See Selectively skipping replication of binlog events for more information.
- Command line: None
- Scope: Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`slave_abort_blocking_timeout`

- Description: Maximum time a replica DDL waits for a blocking `SELECT` or other user query until that query is aborted. The argument is treated as a decimal value with nanosecond precision. The variable is intended to solve a problem where a long-running `SELECT` on a replica causes DDL to wait for that `SELECT` to complete, potentially causing massive replica lag.
- Command line: `--slave-abort-blocking-timeout=num`
- Scope: Global
- Dynamic: Yes

- Data Type: `double`
- Default Value: `31536000.000000`
- Range: `0` to `31536000`
- Introduced: MariaDB 11.7

`slave_compressed_protocol`

- Description: If set to `1` (`0` is the default), uses compression for the replica/primary protocol if both primary and replica support this.
- Command line: `--slave-compressed-protocol`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `0`

`slave_connections_needed_for_purge`

- Description: Minimum number of connected replicas required for automatic binary log purge with max_binlog_total_size, binlog_expire_logs_seconds or expire_logs_days.
  Change of the value triggers an attempt to purging, though without binlog rotation, with the purged set of files satisfying the above two parameters and the value that is set itself.
- Command line: `--slave-connections-needed-for-purge=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`; `0` on Galera cluster nodes.
- Range: `0` to `18446744073709551615`
- Introduced: MariaDB 11.4

`slave_ddl_exec_mode`

- Description: Modes for how replication of DDL events should be executed. Legal values are `STRICT` and `IDEMPOTENT` (default). In `IDEMPOTENT` mode, the replica does not stop for failed DDL operations that would not cause a difference between the primary and the replica. In particular CREATE TABLE is treated as CREATE OR REPLACE TABLE and DROP TABLE is treated as `DROP TABLE IF EXISTS`.
- Command line: `--slave-ddl-exec-mode=name`
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `IDEMPOTENT`
- Valid Values: `IDEMPOTENT`, `STRICT`

`slave_domain_parallel_threads`

- Description: When set to a non-zero value, each replication domain in one primary connection can reserve at most that many worker threads at any one time, leaving the rest (up to the value of slave_parallel_threads) free for other primary connections or replication domains to use in parallel. See Parallel Replication for details.
- Command line: `--slave-domain-parallel-threads=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `0`
- Valid Values: `0` to `16383`

`slave_exec_mode`

- Description: Determines the mode used for replication error checking and conflict resolution. `STRICT` mode is the default, and catches all errors and conflicts. `IDEMPOTENT` mode suppresses duplicate key or no key errors, which can be useful in certain replication scenarios, such as when there are Galera nodes, multiple primaries, or circular replication.
- Scope: Global
- Dynamic: Yes
- Data Type: `enumeration`
- Default Value: `STRICT`
- Valid Values: `IDEMPOTENT`, `STRICT`

`slave_load_tmpdir`

- Description: Directory where the replica stores temporary files for replicating LOAD DATA INFILE statements. If not set, the replica uses tmpdir. Should be set to a disk-based directory that survives restarts, or else replication can fail.
- Command line: `--slave-load-tmpdir=path`
- Scope: Global
- Dynamic: No
- Data Type: `file name`
- Default Value: `/tmp`

`slave_max_allowed_packet`

- Description: Maximum packet size in bytes for replica SQL and I/O threads. This value overrides max_allowed_packet for replication purposes. Set in multiples of 1024 (the minimum) up to 1GB
- Command line: `--slave-max-allowed-packet=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1073741824`
- Range: `1024` to `1073741824`

`slave_max_statement_time`

- Description: A query that has taken more than this in seconds to run on the replica is aborted. The argument is treated as a decimal value with microsecond precision. A value of `0` (default) means no timeout.
- Command line: `--slave-max-statement-time=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0.000000`
- Range: `0` to `31536000`
- Introduced: MariaDB 10.10

`slave_net_timeout`

- Description: Time in seconds for the replica to wait for more data from the primary before considering the connection broken, after which it aborts the read and attempt to reconnect. The retry interval is determined by the `MASTER_CONNECT_RETRY` open for the CHANGE MASTER statement, while the maximum number of reconnection attempts is set by the master-retry-count option. The first reconnect attempt takes place immediately.
- Command line: `--slave-net-timeout=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value:
  - `60 (1 minute)`
- Range: `1` to `31536000`

`slave_parallel_max_queued`

- Description: When parallel_replication is used, the SQL thread reads ahead in the relay logs, queueing events in memory while looking for opportunities for executing events in parallel. This system variable sets a limit for how much memory it uses for this.
  - The configured value of this system variable is actually allocated for each worker thread, so the total allocation is actually equivalent to the following:
    - slave_parallel_max_queued * slave_parallel_threads
  - This system variable is only meaningful when parallel replication is configured (i.e. when slave_parallel_threads > `0`).
  - See Parallel Replication: Configuring the Maximum Size of the Parallel Replica Queue for more information.
- Command line: `--slave-parallel-max-queued=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `131072`
- Range: `0` to `2147483647`

`slave_parallel_mode`

- Description: Controls what transactions are applied in parallel when using parallel replication.
  - `optimistic`: tries to apply most transactional DML in parallel and handles any conflicts with rollback and retry. See optimistic mode.
  - `conservative`: limits parallelism in an effort to avoid any conflicts. See conservative mode.
  - `aggressive`: tries to maximize the parallelism, possibly at the cost of increased conflict rate.
  - `minimal`: only parallelizes the commit steps of transactions.
  - `none` disables parallel apply completely.
- Command line: None
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `optimistic` (>= MariaDB 10.5.1), `conservative` (<= MariaDB 10.5.0)
- Valid Values: `conservative`, `optimistic`, `none`, `aggressive` and `minimal`

`slave_parallel_threads`

- Description: This system variable is used to configure parallel replication.
  - If this system variable is set to a value greater than `0`, then its value determines how many replica worker threads are created to apply binary log events in parallel.
  - If this system variable is set to `0` (which is the default value), no replica worker threads are created. Instead, when replication is enabled, binary log events are applied by the replica's SQL thread.
  - The replica threads must be stopped in order to change this option's value dynamically.
  - Events that were logged with GTIDs with different gtid_domain_id values can be applied in parallel in an out-of-order manner. Each gtid_domain_id can use the number of threads configured by slave_domain_parallel_threads.
  - Events that were group-committed on the primary can be applied in parallel in an in-order manner, and the specific behavior can be configured by setting slave_parallel_mode.
- Command line: `--slave-parallel-threads=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `16383`

`slave_parallel_workers`

- Description: Alias for slave_parallel_threads.
- Command line: `--slave-parallel-workers=#`

`slave_run_triggers_for_rbr`

- Description: See Running triggers on the replica for Row-based events for a description and use-case for this setting. In MariaDB Enterprise Server 12.3, this variable also enables Conflict Detection and Resolution (CDR) triggers on the replica.
- Command line: `--slave-run-triggers-for-rbr=value`
- Scope: Global
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `NO`
- Valid Values: `NO`, `YES`, `LOGGING`, or `ENFORCE` (>= MariaDB 10.5.2)

`slave_skip_errors`

- Description: When an error occurs on the replica, replication usually halts. This option permits a list of error codes to ignore, and for which replication continues. This option should never be needed in normal use, and careless use could lead to replica that are out of sync with primaries. Error codes are in the format of the number from the replica error log. Using `all` as an option permits the replica the keep replicating no matter what error it encounters, an option you would never normally need in production, and which could rapidly lead to data inconsistencies. A count of these is kept in slave_skipped_errors.
- Command line: `--slave-skip-errors=[error_code1,error_code2,...|all]`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `OFF`

- Valid Values: `[list of error codes]`, `ALL`, `OFF`

`slave_sql_verify_checksum`

- Description: Verify binlog checksums when the replica SQL thread reads events from the relay log.
- Command line: `--slave-sql-verify-checksum=[0|1]`
- Scope: Global
- Access Type: Can be changed dynamically
- Data Type: `bool`
- Default Value: `ON (1)`

`slave_transaction_retries`

- Description: Number of times a replication replica retries to execute an SQL thread after it fails due to InnDB deadlock or by exceeding the transaction execution time limit. If after this number of tries the SQL thread has still failed to execute, the replica stops with an error. See also the innodb_lock_wait_timeout system variable.
- Command line: `--slave-transaction-retries=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10`
- Range - 32 bit: `0` to `4294967295`
- Range - 64 bit: `0` to `18446744073709547520`

`slave_transaction_retry_errors`

- Description: When an error occurs during a transaction on the replica, replication usually halts. By default, transactions that caused a deadlock or elapsed lock wait timeout is retried. One can add other errors to the list of errors that should be retried by adding a comma-separated list of error numbers to this variable. This is particularly useful in some Spider setups. Some recommended errors to retry for Spider are 1020, 1158, 1159, 1160, 1161, 1429, 2013, 12701 (these are in the default value in recent versions).
- Command line: `--slave-transaction-retry-errors=[error_code1,error_code2,...]`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value:
  - `1158,1159,1160,1161,1205,1213,1020,1429,2013,12701` (>= MariaDB 10.6.18, MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5, MariaDB 11.2.4, MariaDB 11.4.2)
  - `1158,1159,1160,1161,1205,1213,1429,2013,12701` (>= MariaDB 10.4.5)
- Valid Values: *`comma-separated list of error codes`*
- Introduced: MariaDB 10.3.3

`slave_transaction_retry_interval`

- Description: Interval in seconds for the replica SQL thread to retry a failed transaction due to a deadlock, elapsed lock waits timeout or an error listed in slave_transaction_retry_errors. The interval is calculated as `max(slave_transaction_retry_interval, min(retry_count, 5))`.
- Command line: `--slave-transaction-retry-interval=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `3600`

`slave_type_conversions`

- Description: Determines the type conversion mode on the replica when using row-based replication, including replications in MariaDB Galera cluster. Multiple options can be set, delimited by commas. If left empty, the default, type conversions are disallowed. The variable is dynamic and a change in its value takes effect immediately. This variable tells the server what to do if the table definition is different between the primary and replica (for example a column has a data type of `INT` on the primary and `BIGINT` on the replica).
  - `ALL_NON_LOSSY` means that all safe conversions (no data loss) are allowed.
  - `ALL_LOSSY` means that all lossy conversions are allowed (for example 'bigint' to 'int'). This, however, does not imply that safe conversions (non-lossy) are allowed as well. In order to allow all conversions, one needs to allow both lossy as well as non-lossy conversions by setting this variable to `ALL_NON_LOSSY,ALL_LOSSY`.
  - Empty (default) means that the server gives an error and replication stops if the table definition is different between the primary and replica.
  - `ERROR_IF_MISSING_FIELD`. This value is available from MariaDB 12.3. When provided, the replica aborts replication if it is missing a field that exists on the primary server. When absent, replication continues, although if `log_warnings` is set to `1` or greater, a warning is logged to the error log.
- Command line: `--slave-type-conversions=set`
- Scope: Global
- Dynamic: Yes
- Data Type: `set`
- Default Value: `Empty variable`
- Valid Values: `ALL_LOSSY`, `ALL_NON_LOSSY`, empty

`sql_log_bin`

- Description: If set to 0 (1 is the default), no logging to the binary log is done for the client. Only clients with the `SUPER` privilege can update this variable. Does not affect the replication of events in a Galera cluster. Note that `sql_log_bin` has no effect if log_bin is not set.
- Scope: Session
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `1`
- Related variables: log_bin

`sql_slave_skip_counter`

- Description: Number of events that a replica skips from the primary. If this would cause the replica to begin in the middle of an event group, the replica instead starts from the beginning of the next event group. See SET GLOBAL sql_slave_skip_counter.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`

`sync_binlog`

- Description: MariaDB synchronizes its binary log file to disk after this many events. The default is `0`, in which case the operating system handles flushing the file to disk. `1` is the safest, but slowest, choice, since the file is flushed after each write. If autocommit is enabled, there is one write per statement, otherwise there's one writes per transaction. If the disk has cache backed by battery, synchronization is fast, and a more conservative number can be chosen.
- Command line: `--sync-binlog=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4294967295`

Note: When you enable the InnoDB-based Binary Log (`--binary-storage-engine=innodb`), the `sync_binlog` option is ignored. Instead, the durability of the binary log is controlled by `--innodb-flush-log-at-trx-commit`, which applies to both InnoDB data and binary log writes.

`sync_master_info`

- Description: A replication replica synchronizes its primary.info file to disk after this many events. If set to 0, the operating system handles flushing the file to disk.
- Command line: `--sync-master-info=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10000`

`sync_relay_log`

- Description: The MariaDB server synchronizes its relay log to disk after the specified number of writes to the log. `1` is the safest, but slowest, choice, since the file is flushed after each write. If autocommit is enabled, there is one write per statement, otherwise there's one write per transaction. If the disk has cache backed by battery, synchronization is fast and a more conservative number can be chosen.
- Command line: `--sync-relay-log=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10000`

`sync_relay_log_info`

- Description: A replication replica synchronizes its `relay-log.info` file to disk after the specified number of transactions. `1` is the most secure choice, because at most one event can be lost in the event of a crash, but it's also the slowest.
- Command line: `--sync-relay-log-info=#`
- Scope: Global,
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `10000`
- Range: `0` to `4294967295`

See Also

- Full list of MariaDB options, system and status variables

This page is licensed: CC BY-SA / Gnu FDL

Replication and Binary Log Status Variables

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

The following status variables are useful in binary logging and replication. See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

See also the Full list of MariaDB options, system and status variables.

`Binlog_bytes_written`

- Description: The number of bytes written to the binary log.
- Scope: Global
- Data Type: `numeric`

`Binlog_cache_disk_use`

- Description: Number of transactions which used a temporary disk cache because they could not fit in the regular binary log cache, being larger than binlog_cache_size. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Binlog_cache_use`

- Description: Number of transaction which used the regular binary log cache, being smaller than binlog_cache_size. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Binlog_commits`

- Description: Total number of transactions committed to the binary log.
- Scope: Global
- Data Type: `numeric`

`Binlog_disk_use`

- Description: If max-binlog-total_size is not set to zero, shows the space usage of the binary log in bytes.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.4

`Binlog_group_commit_trigger_count`

- Description: Total number of group commits triggered because of the number of binary log commits in the group reached the limit set by the variable binlog_commit_wait_count. See Group commit for the binary log.

- Scope: Global
- Data Type: `numeric`

`Binlog_group_commit_trigger_lock_wait`

- Description: Total number of group commits triggered because a binary log commit was being delayed because of a lock wait where the lock was held by a prior binary log commit. When this happens the later binary log commit is placed in the next group commit. See Group commit for the binary log.
- Scope: Global
- Data Type: `numeric`

`Binlog_group_commit_trigger_timeout`

- Description: Total number of group commits triggered because of the time since the first binary log commit reached the limit set by the variable binlog_commit_wait_usec. See Group commit for the binary log.
- Scope: Global
- Data Type: `numeric`

`Binlog_group_commits`

- Description: Total number of group commits done to the binary log. See Group commit for the binary log.
- Scope: Global
- Data Type: `numeric`

`Binlog_gtid_index_hit`

- Description: Incremented for each successful lookup in a GTID index.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.4

`Binlog_gtid_index_miss`

- Description: Incremented when a GTID index lookup is not possible, which indicates that the index file is missing (eg. binlog written by old server version without GTID index support), or corrupt.
- Scope: Global
- Data Type: `numeric`
- Introduced: MariaDB 11.4

`Binlog_snapshot_file`

- Description: The binary log file. Unlike SHOW MASTER STATUS, can be queried in a transactionally consistent way, irrespective of which other transactions have been committed since the snapshot was taken. See Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT.
- Scope: Global
- Data Type: `string`

`Binlog_snapshot_position`

- Description: The binary log position. Unlike SHOW MASTER STATUS, can be queried in a transactionally consistent way, irrespective of which other transactions have been committed since the snapshot was taken. See Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT.
- Scope: Global

- Data Type: `numeric`

`Binlog_stmt_cache_disk_use`

- Description: Number of non-transaction statements which used a temporary disk cache because they could not fit in the regular binary log cache, being larger than binlog_stmt_cache_size. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Binlog_stmt_cache_use`

- Description: Number of non-transaction statement which used the regular binary log cache, being smaller than binlog_stmt_cache_size. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Com_change_master`

- Description: Number of CHANGE MASTER TO statements executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_binlog_status`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.5.2

`Com_show_master_status`

- Description: Number of SHOW MASTER STATUS commands executed.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.5.2

`Com_show_new_master`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 5.5

`Com_show_slave_hosts`

- Description: Number of SHOW SLAVE HOSTS commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_show_slave_status`

- Description: Number of SHOW SLAVE STATUS commands executed.
- Scope: Global, Session

- Data Type: `numeric`

`Com_slave_start`

- Description: Number of START SLAVE commands executed. Removed in MariaDB 10.0, see Com_start_slave.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.0

`Com_slave_stop`

- Description: Number of STOP SLAVE commands executed. Removed in MariaDB 10.0, see Com_stop_slave.
- Scope: Global, Session
- Data Type: `numeric`
- Removed: MariaDB 10.0

`Com_start_all_slaves`

- Description: Number of START ALL SLAVES commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_start_slave`

- Description: Number of START SLAVE commands executed. Replaces the old Com_slave_start.
- Scope: Global, Session
- Data Type: `numeric`

`Com_stop_all_slaves`

- Description: Number of STOP ALL SLAVES commands executed.
- Scope: Global, Session
- Data Type: `numeric`

`Com_stop_slave`

- Description: Number of STOP SLAVE commands executed. Replaces the old Com_slave_stop.
- Scope: Global, Session
- Data Type: `numeric`

`Master_gtid_wait_count`

- Description: Number of times MASTER_GTID_WAIT called.
- Scope: Global, Session
- Data Type: `numeric`

`Master_gtid_wait_time`

- Description: Total number of time spent in MASTER_GTID_WAIT.
- Scope: Global, Session
- Data Type: `numeric`

`Master_gtid_wait_timeouts`

- Description: Number of timeouts occurring in MASTER_GTID_WAIT.
- Scope: Global, Session
- Data Type: `numeric`

`Rpl_status`

- Description: For showing the status of fail-safe replication. Removed in MySQL 5.6, still present in MariaDB 10.0.

`Rpl_transactions_multi_engine`

- Description: Number of replicated transactions that involved changes in multiple (transactional) storage engines, before considering the update of `mysql.gtid_slave_pos`. These are transactions that were already cross-engine, independent of the GTID position update introduced by replication. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Slave_connections`

- Description: Number of REGISTER_SLAVE attempts. In practice the number of times slaves has tried to connect to the master.
- Scope: Global
- Data Type: `numeric`

`Slave_heartbeat_period`

- Description: Time in seconds that a heartbeat packet is requested from the master by a slave.
- Scope: Global
- Data Type: `numeric`

`Slave_open_temp_tables`

- Description: Number of temporary tables the slave has open.
- Scope: Global
- Data Type: `numeric`

`Slave_received_heartbeats`

- Description: Number of heartbeats the slave has received from the master.
- Scope: Global
- Data Type: `numeric`

`Slave_retried_transactions`

- Description: Number of times the slave has retried transactions since the server started. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Slave_running`

- Description: Whether the default connection slave is running (both I/O and SQL threads are running) or not.
- Scope: Global

- Data Type: `numeric`

`Slave_skipped_errors`

- Description: The number of times a slave has skipped errors defined by slave-skip-errors.
- Scope: Global
- Data Type: `numeric`

`Slaves_connected`

- Description: Number of slaves connected.
- Scope: Global
- Data Type: `numeric`

`Slaves_running`

- Description: Number of slave SQL threads running.
- Scope: Global
- Data Type: `numeric`

`Transactions_gtid_foreign_engine`

- Description: Number of replicated transactions where the update of the `gtid_slave_pos` table had to choose a storage engine that did not otherwise participate in the transaction. This can indicate that setting gtid_pos_auto_engines might be useful. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

`Transactions_multi_engine`

- Description: Number of transactions that changed data in multiple (transactional) storage engines. If this is significantly larger than Rpl_transactions_multi_engine, it indicates that setting gtid_pos_auto_engines could reduce the need for cross-engine transactions. The global value can be flushed by FLUSH STATUS.
- Scope: Global
- Data Type: `numeric`

This page is licensed: CC BY-SA / Gnu FDL

Binlog Event Checksum Interoperability

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

The introduction of checksums on binlog events changes the format that events are stored in binary log files and sent over the network to replicas. This raises the question on what happens when replicating between different versions of the server, where one server is a newer version that has the binlog checksum feature implemented, while the other server is an older version that does not know about binlog checksums.

When checksums are disabled on the primary (or the primary has the old version with no checksums implemented), there is no problem. In this case the binlog format is backwards compatible, and replication works fine.

When the primary is a newer version with checksums enabled in the binlog, but the replica is an old version that does not understand checksums, replication will fail. The primary will disconnect the replica with an error, and also log a warning in its own error log. This prevents sending events to the replica that it will be unable to interpret correctly, but means that binlog checksums can not be used with older replicas. (With the recommended upgrade path, where replicas are upgraded before primaries, this is not a problem of course).

Replicating from a new MySQL primary with checksums enabled to a new MariaDB which also understands checksums works, and the MariaDB replica will verify checksums on replicated events.

There is however a problem when a newer MySQL replica replicates against a newer MariaDB primary with checksums enabled. The replica server looks at the primary server version to know whether events include checksums or not, and MySQL has not yet been updated to learn that MariaDB does this already from version 5.3.0 (as of the time of writing, MySQL 5.6.2). Thus, if MariaDB at least version
5.3.0 but less that 5.6.1 is used as a primary with binlog checksums enabled, a MySQL replica will interpret the received events incorrectly as it does not realise the last part of the events is the checksum. So replication will fail with an error about corrupt events or even silent corruption of replicated data in unlucky cases. This requires changes to the MySQL server to fix.

Here is a summary table of the status of replication between different combination of primary and replica servers and checksum enabled/disabled:

- OLD: MySQL <5.6.1 or MariaDB < 5.3.0 with no checksum capabilities
- NEW-MARIA: MariaDB >= 5.3.0 with checksum capabilities
- NEW-MYSQL: MySQL >= 5.6.1 with checksum capabilities

Primary mariadb-lbinlog	Replica / enabled?	Checksums	Status
OLD	OLD	-	Ok
OLD	NEW-MARIA	-	Ok
OLD	MYSQL	-	Ok
NEW-MARIA	OLD	No	Ok
NEW-MARIA	OLD	Yes	Primary will refuse with error
NEW-MARIA	NEW-MARIA	Yes/No	Ok
NEW-MARIA	NEW-MYSQL	No	Ok
NEW-MARIA	NEW-MYSQL	Yes	Fail. Requires changes in MySQL, otherwise it will not realise MariaDB < 5.6.1 does checksums and will be confused.
NEW-MYSQL	OLD	No	Ok
NEW-MYSQL	OLD	Yes	Primary will refuse with error
NEW-MYSQL	NEW-MARIA	Yes/No	Ok
NEW-MYSQL	NEW-MYSQL	Yes/No	Ok

Checksums and mariadb-binlog

When using the mariadb-binlog client program, there are similar issues.

A version of `mariadb-binlog` which understands checksums can read binlog files from either old or new servers, with or without checksums enabled.

An old version of `mariadb-binlog` can read binlog files produced by a new server version **if** checksums were disabled when the log was produced. Old versions of `mariadb-binlog` reading a new binlog file containing checksums will be confused, and output will be garbled, with the added checksums being interpreted as extra garbage at the end of query strings and similar entries. No error will be reported in this case, just wrong output.

A version of `mysqlbinlog` (the MySQL equivalent to mariadb-binlog and the old MariaDB name for the binary) from MySQL >= 5.6.1 will have similar problems as a replica until this is fixed in MySQL. When reading a binlog file with checksums produced by MariaDB >= 5.3.0 but < 5.6.1, it will not realise that checksums are included, and will produce garbled output just like an old version of `mysqlbinlog`. The MariaDB version of `mariadb-binlog` can read binlog files produced by either MySQL or MariaDB just fine.

See Also

- Binlog Event Checksums

This page is licensed: CC BY-SA / Gnu FDL

Binlog Event Checksums

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

MariaDB includes a feature to include a checksum in binary log events.

Checksums are enabled with the binlog_checksum option, which is set to `CRC32` by default.

The variable can be changed dynamically without restarting the server. Setting the variable in any way (even to the existing value) forces a rotation of the binary log (the intention is to avoid having a single binlog where some events are checksummed and others are not).

When checksums are enabled, replicas will check events received over the network for checksum errors, and will stop with an error if a corrupt event is detected.

In addition, the server can be configured to verify checksums in two other places.

One is when reading events from the binlog on the primary, for example when sending events to a replica or for something like SHOW BINLOG EVENTS. This is controlled by the master_verify_checksum option, and is thus used to detect file system corruption of the binlog files.

The other is when the replica SQL thread reads events from the relay log. This is controlled by the slave_sql_verify_checksum option, and is used to detect file system corruption of replica relay log files.

MariaDB starting with 11.4

From MariaDB 11.4, binlog checksums are computed when writing events into the statement or transaction caches, where before this was done when the caches were copied to the real binlog file. This moves the checksum computation outside of holding LOCK_log, improving scalability. See MDEV-31273.

`master_verify_checksum`

- Description: Verify binlog checksums when reading events from the binlog on the primary.
- Command line: `--master_verify_checksum=[0|1]`
- Scope: Global
- Access Type: Can be changed dynamically
- Data Type: `bool`
- Default Value: `OFF (0)`

`slave_sql_verify_checksum`

- Description: Verify binlog checksums when the replica SQL thread reads events from the relay log.

- Command line: `--slave_sql_verify_checksum=[0|1]`
- Scope: Global
- Access Type: Can be changed dynamically
- Data Type: `bool`
- Default Value: `ON (1)`

The mariadb-binlog client program by default does not verify checksums when reading a binlog file, however it can be instructed to do so with the option `verify-binlog-checksum`:

- Variable Name: `verify-binlog-checksum`
- Data Type: `bool`
- Default Value: `OFF`

See Also

- Binlog Event Checksum Interoperability
- What is MariaDB 5.3

This page is licensed: CC BY-SA / Gnu FDL

Changing a Replica to Become the Primary

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

This article describes how to change a replica to become a primary and optionally to set the old primary as a replica for the new primary.

A typical scenario of when this is useful is if you have set up a new version of MariaDB as a replica, for example for testing, and want to upgrade your primary to the new version.

In MariaDB replication, a replica should be of a version same or newer than the primary. Because of this, one should first upgrades all replicas to the latest version before changing a replica to be a primary. In some cases one can have a replica to be of an older version than the primary, as long as one doesn't execute on the primary any SQL commands that the replica doesn't understand. This is however not guaranteed between all major MariaDB versions.

Note that in the examples below, `[connection_name]` is used as the name of the connection. If you are not using named connections you can ignore this.

Stopping the Original Master.

First one needs to take down the original primary in such a way that the replica has all information on the primary.

If you are using Semisynchronous Replication you can just stop the server with the SHUTDOWN command as the replicas should be automatically up to date.

If you are using MariaDB MaxScale proxy, then you can use MaxScale to handle the whole process of taking down the primary and replacing it with one of the replicas.

If neither of the above is true, you have to do this step manually:

Manually Take Down the Primary

First we have to set the primary to read only to ensure that there are no new updates on the primary:

```
FLUSH TABLES WITH READ LOCK;
```

Note that you should not disconnect this session as otherwise the read lock will disappear and you have to start from the beginning.

Then you should check the current position of the primary:

```
SHOW MASTER STATUS;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| mariadb-bin.000003 |      343 |              |                  |
+--------------------+----------+--------------+------------------+
SELECT @@global.gtid_binlog_pos;
+--------------------------+
| @@global.gtid_binlog_pos |
+--------------------------+
| 0-1-2                    |
+--------------------------+
```

And wait until you have the same position on the replica:
(The following should be expected on the replica)

```
SHOW SLAVE [connection_name] STATUS;
+------------------+------------------+
Master_Log_File     | narttu-bin.000003 +
Read_Master_Log_Pos | 343               +
Exec_Master_Log_Pos | 343               +
...
Gtid_IO_Pos          0-1-2              +
+------------------+------------------+
```

The most important information to watch are `Master_Log_File` and `Exec_Master_Log_Pos` as when this matches the primary, it signals that all transactions have been committed on the replica.

Note that `Gtid_IO_Pos` on replica can contain many different positions separated with ‘,’ if the replica has been connected to many different primaries. What is important is that all the sequences that are on the primary is also on the replica.

When replica is up to date, you can then take the **PRIMARY** down. This should be on the same connection where you executed FLUSH TABLES WITH READ LOCK.

```
SHUTDOWN;
```

Preparing the Replica to be a Primary

Stop all old connections to the old primary(s) and reset **read only mode**, if you had it enabled. You also want to save the values of SHOW MASTER STATUS and `gtid_binlog_pos`, as you may need these to setup new replicas.

```
STOP ALL SLAVES;
RESET SLAVE ALL;
SHOW MASTER STATUS;
SELECT @@global.gtid_binlog_pos;
SET @@global.read_only=0;
```

Reconnect Other Replicas to the New Primary

On the other replicas you have point them to the new primary (the replica you promoted to a primary).

```
STOP SLAVE [connection_name];
CHANGE MASTER [connection_name] TO
 MASTER_HOST="new_master_name",
 MASTER_PORT=3306,
 MASTER_USER='root',
 MASTER_USE_GTID=current_pos,
 MASTER_LOG_FILE="XXX",
 MASTER_LOG_POS=XXX;
START SLAVE;
```

The `XXX` values for `MASTER_LOG_FILE` and `MASTER_LOG_POS` should be the values you got from the `SHOW MASTER STATUS` command you did when you finished setting up the replica.

Changing the Old Primary to be a Replica

Now you can upgrade the old primary to a newer version of MariaDB and then follow the same procedure to connect it as a replica.

When starting the original primary, it's good to start the `mysqld` executable with the `--with-skip-slave-start` and `--read-only` options to ensure that no old replica configurations could cause any conflicts.

For the same reason it's also good to execute the following commands on the old primary (same as for other replicas, but with some extra security). The `read_only` option below is there to ensure that old applications doesn't by accident try to update the old primary by mistake.
It only affects normal connections to the replica, not changes from the new primary.

```
SET @@global.read_only=1;
STOP ALL SLAVES;
RESET MASTER;
RESET SLAVE ALL;
CHANGE MASTER [connection_name] TO
 MASTER_HOST="new_master_name",
 MASTER_PORT=3306,
 MASTER_USER='root',
 MASTER_USE_GTID=current_pos,
 MASTER_LOG_FILE="XXX",
 MASTER_LOG_POS=XXX;
START SLAVE;
```

Moving Applications to Use New Primary

You should now point your applications to use the new primary.
If you are using the MariaDB MaxScale proxy, then you don't have to do this step as MaxScale will take care of sending write request to the new primary.

See Also

- CHANGE MASTER TO command
- MaxScale Blog about using Switchover to swap a primary and replica
- Percona blog about how to upgrade replica to primary

This page is licensed: CC BY-SA / Gnu FDL

Replication When the Primary and Replica Have Different Table Definitions

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

While replication is usually meant to take place between primaries and replicas with the same table definitions and this is recommended, in certain cases replication can still take place even if the definitions are identical.

Tables on the replica and the primary do not need to have the same definition in order for replication to take place. There can be differing numbers of columns, or differing data definitions and, in certain cases, replication can still proceed.

Different Column Definitions - Attribute Promotion and Demotion

It is possible in some cases to replicate to a replica that has a column of a different type on the replica and the primary. This process is called attribute promotion (to a larger type) or attribute demotion (to a smaller type).

The conditions differ depending on whether statement-based or row-based replication is used.

Statement-Based Replication

When using statement-based replication, generally, if a statement can run successfully on the replica, it will be replicated. If a column definition is the same or a larger type on the replica than on the primary, it can replicate successfully. For example, a column defined as VARCHAR(10) will successfully be replicated on a replica with a definition of `VARCHAR(12)`.

Replicating to a replica where the column is defined as smaller than on the primary can also work. For example, given the following table definitions:

Master:

```
DESC r;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | tinyint(4)  | YES  |     | NULL    |       |
| v     | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

Slave

```
DESC r;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | tinyint(4)  | YES  |     | NULL    |       |
| v     | varchar(8) | YES   |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

the statement

```
INSERT INTO r VALUES (6,'hi');
```

would successfully replicate because the value inserted into the `v` field can successfully be inserted on both the primary and the smaller replica equivalent.

However, the following statement would fail:

```
INSERT INTO r VALUES (7,'abcdefghi')
```

In this case, the value fits in the primary definition, but is too long for the replica field, and so replication will fail.

```
SHOW SLAVE STATUS\G
*************************** 1. row ***************************
...
Slave_IO_Running: Yes
Slave_SQL_Running: No
...
Last_Errno: 1406
Last_Error: Error 'Data too long for column 'v' at row 1' on query.
   Default database: 'test'. Query: 'INSERT INTO r VALUES (7,'abcdefghi')'
...
```

Row-Based Replication

When using row-based replication, the value of the slave_type_conversions variable is important. The default value of this variable is empty, in which case MariaDB will not perform attribute promotion or demotion. If the column definitions do not match, replication will stop. If set to `ALL_NON_LOSSY`, safe replication is permitted. If set to `ALL_LOSSY` as well, replication will be permitted even if data loss takes place.

For example:

Master:

```
DESC r;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | smallint(6) | YES  |     | NULL    |       |
| v     | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

Slave:

```
SHOW VARIABLES LIKE 'slave_ty%';
+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| slave_type_conversions |       |
+------------------------+-------+

 DESC r;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| id    | tinyint(4) | YES  |     | NULL    |       |
| v     | varchar(1) | YES  |     | NULL    |       |
+-------+------------+------+-----+---------+-------+
```

The following query fails:

```
INSERT INTO r VALUES (3,'c');
```

```
SHOW SLAVE STATUS\G;
...
Slave_IO_Running: Yes
Slave_SQL_Running: No
...
Last_Errno: 1677
Last_Error: Column 0 of table 'test.r' cannot be converted from
  type 'smallint' to type 'tinyint(4)'
...
```

By changing the value of the slave_type_conversions, replication can proceed:

```
SET GLOBAL slave_type_conversions='ALL_NON_LOSSY,ALL_LOSSY';

START SLAVE;
```

```
SHOW SLAVE STATUS\G;
*************************** 1. row ***************************
...
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
...
```

Supported Conversions

- Between TINYINT, SMALLINT, MEDIUMINT, INT and BIGINT. If lossy conversion is supported, the value from the primary will be converted to the maximum or minimum permitted on the replica, which non-lossy conversions require the replica column to be large enough. For example, `SMALLINT UNSIGNED` can be converted to `MEDIUMINT`, but not `SMALLINT SIGNED`.

Different Number or Order of Columns

Replication can also take place when the primary and replica have a different number of columns if the following criteria are met:

- Columns are in the same order on the primary and replica.
- Common columns are defined with the same data type.
- Extra columns are defined after the common columns.
- The primary is configured with `binlog_row_metadata=FULL` and the replica runs on MariaDB 12.3 or higher (note option `slave_type_conversions=ERROR_IF_MISSING_FIELD`).

Row-Based

The following example replicates incorrectly (replication proceeds, but the data is corrupted), as the columns are not in the same order.

Master:

```
CREATE OR REPLACE TABLE r (i1 INT, i2 INT);
```

Slave:

```
ALTER TABLE r ADD i3 INT AFTER i1;
```

Master:

```
INSERT INTO r (i1,i2) VALUES (1,1);

SELECT * FROM r;
+------+------+
| i1   | i2   |
+------+------+
|    1 |    1 |
+------+------+
```

Slave:

```
SELECT * FROM r;
+------+------+------+
| i1   | i3   | i2   |
+------+------+------+
|    1 |    1 | NULL |
+------+------+------+
```

Statement-Based

Using statement-based replication, the same example may work, even though the columns are not in the same order.

Master:

```
CREATE OR REPLACE TABLE r (i1 INT, i2 INT);
```

Slave:

```
ALTER TABLE r ADD i3 INT AFTER i1;
```

Master:

```
INSERT INTO r (i1,i2) VALUES (1,1);

SELECT * FROM r;
+------+------+
| i1   | i2   |
+------+------+
|    1 |    1 |
+------+------+
```

Slave:

```
SELECT * FROM r;
+------+------+------+
| i1   | i3   | i2   |
+------+------+------+
|    1 | NULL |    1 |
+------+------+------+
```

ALTER TABLE Issues

Row-Based Replication

In row-based multi-master replication, `ALTER TABLE` statements can make replication fail.

`ALTER TABLE` works fine when columns are added to the end of the table definition. When columns are deleted or added in the middle of a table, though, this causes subsequent DML[33] queries to fail.

> **Note:**
>
> The following functionality is available from MariaDB 12.3.

The solution to this problem is to **start master servers with `--binlog_row_metadata=full`**. With that setting, the master writes column names to the binary log. The replicas receiving binlog events can then look up the columns using column names.

The solution to the other problem, deleting columns in the middle of a table, is addressed by **configuring the replica with `--slave_type_conversions=ignore_missing_columns`**.

This page is licensed: CC BY-SA / Gnu FDL

START TRANSACTION … WITH CONSISTENT SNAPSHOT

The `START TRANSACTION WITH CONSISTENT SNAPSHOT` statement begins a new transaction and, for the InnoDB storage engine, immediately establishes a consistent read view of the database.

This differs from a standard `START TRANSACTION` or `BEGIN` statement, which creates its read view *lazily* only when the first read operation is performed. Using `WITH CONSISTENT SNAPSHOT` is essential for transactions where the snapshot's timing must be precisely aligned with the start of the transaction itself, not a later read query.

Syntax

`START TRANSACTION` and its alias `BEGIN` can be modified with one or more characteristics.

```
START TRANSACTION [transaction_characteristic [, transaction_characteristic] ...]

transaction_characteristic:
    WITH CONSISTENT SNAPSHOT
  | READ WRITE | READ ONLY
  | [NOT] CHAIN
```

For example:

```
BEGIN WITH CONSISTENT SNAPSHOT;
START TRANSACTION READ ONLY, WITH CONSISTENT SNAPSHOT;
```

The InnoDB Read View

MariaDB's InnoDB storage engine uses a mechanism called MVCC (Multi-Version Concurrency Control) to handle concurrent data access. A core component of MVCC is the read view.

A read view can be thought of as an instantaneous snapshot of the database. When a transaction uses a read view, it sees only the data that was committed at the moment the "snapshot" was taken. It ignores any changes made by transactions that had not yet committed, as well as any changes from transactions that started after the read view was created.

The key difference addressed by this command is the timing of this snapshot:

Transaction Type	Creation of Read View	Time of Read View Creation
`START TRANSACTION`	Created lazily	At the first read operation (e.g., a `SELECT`)
`START TRANSACTION WITH CONSISTENT SNAPSHOT`	Created immediately	At the moment the statement is executed, before other actions

Behavior with Transaction Isolation Levels

The behavior of `WITH CONSISTENT SNAPSHOT` is dependent on the transaction isolation level.

Isolation Level	Default Read View Behavior	Effect of WITH CONSISTENT SNAPSHOT
REPEATABLE READ	A single, stable read view is created and used for the entire transaction.	Guarantees the read view is established immediately at the start of the transaction.
SERIALIZABLE	Same as REPEATABLE READ	Provides a predictable snapshot for all subsequent reads. This is its most common use case.
READ COMMITTED	A new read view is created for each individual `SELECT` statement.	Affects the first read statement only, ensuring its snapshot is taken at the transaction's start time.
READ UNCOMMITTED	Does not use read views. Reads include uncommitted data ("dirty reads").	Not permitted and has no effect.

With Default Read View Behavior

Checking an Account Balance with a Transaction Delay

In this situation, there is a delay in the application logic after initiating a transaction to check an account balance.

Setup

```
CREATE TABLE accounts (
    id INT PRIMARY KEY,
    balance DECIMAL(10, 2)
) ENGINE=InnoDB;

INSERT INTO accounts VALUES (1, 1000.00);
```

Scenario: Two sessions are running concurrently.

Timeline	Session 1 (Application checking balance)	Session 2 (An external deposit)
T1	`START TRANSACTION;`	
T2	`– Application logic causes a 2-second delay DO SLEEP(2);`	`START TRANSACTION;`
T3		`UPDATE accounts SET balance = 1500.00 WHERE id = 1;`
T4		`COMMIT;`
T5	`SELECT balance FROM accounts WHERE id = 1;`	
T6	`COMMIT;`	

In the scenario above, the read view for Session 1 is created at T5. Since Session 2's `COMMIT` happened at T4, the `SELECT` in Session 1 will see the new balance: `$1500.00`. This might not be the desired behavior if the goal was to see the balance as it was at T1.

With CONSISTENT SNAPSHOT

Timeline	Session 1 (Application checking balance)	Session 2 (An external deposit)
T1	`START TRANSACTION WITH CONSISTENT SNAPSHOT;`	
T2	`– Application logic causes a 2-second delay DO SLEEP(2);`	`START TRANSACTION;`
T3		`UPDATE accounts SET balance = 1500.00 WHERE id = 1;`
T4		`COMMIT;`
T5	`SELECT balance FROM accounts WHERE id = 1;`	

Timeline	Session 1 (Application checking balance)	Session 2 (An external deposit)
T6	`COMMIT;`	

In this second scenario, the read view for Session 1 is created immediately at T1. Even though Session 2 commits a change at T4, the `SELECT` at T5 uses the original snapshot. It will see the old balance: `$1000.00`, reflecting the state of the database when the transaction began.

Related System Variables

`innodb_snapshot_isolation`

This system variable influences the behavior of locking reads (for instance, `SELECT ... FOR UPDATE`). When innodb_snapshot_isolation is enabled (`ON`), locking reads reference the transaction's read view. If a transaction tries to lock a row modified by another transaction not visible in the current read view, MariaDB returns an `ER_CHECKREAD` error instead of waiting for a lock. This enforces stricter snapshot consistency, even for locking operations.

This page is licensed: CC BY-SA / Gnu FDL

Restricting Speed of Reading Binlog from Primary by a Replica

When a replica starts after being stopped for some time, or a new replica starts that was created from a backup from some time back, a lot of old binlog events may need to be downloaded from the primary. If this happens from many replicas simultaneously, it can put a lot of load on the primary.

The **read_binlog_speed_limit** option can be used to reduce such load, by limiting the speed at which events are downloaded. The limit is given as maximum kilobytes per second to download on one replica connection.

With this option set, the replication I/O thread will limit the rate of download. Since the I/O thread is often much faster to download events than the SQL thread is at applying them, an appropriate value for**read_binlog_speed_limit** may reduce load spikes on the primary without much limit in the speed of the replica.

The option **read_binlog_speed_limit** is available starting from MariaDB 10.2.3.

`read_binlog_speed_limit`

- Description: Maximum speed(KB/s) to read binlog from primary.
- Command line: `--read-binlog-speed-limit[=#]`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `4294967295`
- Introduced: MariaDB 10.2.3

This page is licensed: CC BY-SA / Gnu FDL

Running Triggers on the Replica for Row-based Events

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

MariaDB can force the replica thread to run triggers for row-based binlog events.

The setting is controlled by the slave_run_triggers_for_rbr global variable. It can be also specified as a command-line option or in my.cnf.

Possible values are:

Value	Meaning
NO (Default)	Don't invoke triggers for row-based events
YES	Invoke triggers for row-based events, don't log their effect into the binary log
LOGGING	Invoke triggers for row-based events, and log their effect into the binary log
ENFORCE	From MariaDB 10.5.2 only. Triggers will always be run on the replica, even if there are triggers on the master. ENFORCE implies LOGGING.

Note that if you just want to use triggers together with replication, you most likely don't need this option. Read below for details.

> **Note:**
>
> Starting with MariaDB Enterprise Server 12.3, enabling `slave_run_triggers_for_rbr` also activates Conflict Detection and Resolution (CDR) triggers — a dedicated trigger type for resolving replication conflicts on the replica instead of stopping the SQL thread.

When to Use slave_run_triggers_for_rbr

Background

Normally, MariaDB's replication system can replicate trigger actions automatically.

- When one uses statement-based replication, the binary log contains SQL statements. Replica server(s) execute the SQL statements. Triggers are run on the master and on each replica, independently.
- When one uses row-based replication, the binary log contains row changes. It will have both the changes made by the statement itself, and the changes made by the triggers that were invoked by the statement. Replica server(s) do not need to run triggers for row changes they are applying.

Target Usecase

One may want to have a setup where a replica has triggers that are not present on the master (Suppose the replica needs to update summary tables or perform some other ETL-like process).

If one uses statement-based replication, they can just create the required triggers on the replica. The replica will run the statements from the binary log, which will cause the triggers to be invoked.

However, there are cases where you have to use row-based replication. It could be because the master runs non-deterministic statements, or the master could be a node in a Galera cluster. In that case, you would want row-based events to invoke triggers on the replica. This is what the `slave_run_triggers_for_rbr` option is for. Setting the option to `YES` will cause the SQL replica thread to invoke triggers for row-based events; setting it to `LOGGING` will also cause the changes made by the triggers to be written into the binary log.

The following triggers are invoked:

- Update_row_event runs an UPDATE trigger
- Delete_row_event runs a DELETE trigger
- Write_row_event runs an INSERT trigger. Additionally it runs a DELETE trigger if there was a conflicting row that had to be deleted.

Preventing Multiple Trigger Invocations

There is a basic protection against triggers being invoked both on the master and replica.
If the master modifies a table that has triggers, it will produce row-based binlog events with the “triggers were invoked for this event” flag. The replica will not invoke any triggers for flagged events.

See Also

- Conflict Detection and Resolution (CDR) Triggers
- Task in Jira, MDEV-5095.

This page is licensed: CC BY-SA / Gnu FDL

Conflict Detection and Resolution (CDR) Triggers

> **Warning:**
>
> CDR triggers are available beginning with **MariaDB Enterprise Server 12.3** and are a **beta feature**. Read Limitations and Beta Caveats before deploying CDR triggers on a production replica. Behavior outside the supported configuration is unspecified and may change.

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Replica data can diverge from the primary in three ways:

1. **Direct local writes to the replica** — for example, an ETL job or a manual administrative fix.
2. **Multi-writer topologies** — such as ring or bidirectional replication, where every server legitimately accepts writes.
3. **Operational drift** — left behind by skipped events, imperfect restores, or past failovers.

In every case the replication stream itself remains correct — the divergence enters alongside it, and surfaces only when a later event touches a row that no longer matches. Conflict Detection and Resolution (CDR) Triggers let the replica resolve that moment according to a policy you define, instead of stopping replication.

Overview

In row-based replication (RBR), the replica applies the row events recorded in the primary’s binary log. Each event carries a *before-image* (the row as it existed on the primary) and/or an *after-image* (the row as the primary changed it). The applier locates the matching row on the replica and performs the same insert, update, or delete.

A **conflict** occurs when the replica’s local data has diverged from what the primary’s event expects:

- The primary inserts a row whose key **already exists** on the replica.
- The primary updates or deletes a row that is **missing** on the replica.
- The primary updates or deletes a row that exists on the replica but **holds different values**.

By default, a divergence that blocks the applier — a duplicate key, a missing row — raises a hard error and **stops the SQL thread**, requiring an operator to intervene manually or skip the event. A divergence that does not block it — a row that exists but holds different values — is silently overwritten (or deleted) by the incoming event and passes unnoticed unless CDR is enabled.

Conflict Detection and Resolution (CDR) Triggers let you encode the resolution policy as SQL, on the replica. When a conflict is detected, the applier diverts the failing row event into a user-defined trigger. Inside the trigger you

decide, per row, whether to overwrite, merge, ignore, or deliberately halt.

Enabling CDR Triggers

CDR triggers fire only on the replica's SQL (applier) thread, and they operate on row-based replication events.

Primary-Side Configuration

The only requirement on the primary is that it logs changes in row format with full row images, so that the replica receives the before- and after-images CDR triggers depend on:

```
[mariadbd]
binlog_format    = ROW
binlog_row_image = FULL
```

No other CDR-specific configuration is needed on the primary, and CDR triggers do **not** need to exist on the primary. (`STATEMENT` and `MIXED` binary log formats do not carry the row images CDR requires, so they are not supported for this feature.)

Replica-Side Configuration

CDR triggers are gated by the same global system variable that controls running triggers on the replica for row-based events, slave_run_triggers_for_rbr. There is no separate CDR switch. For CDR conflict handling, set the variable to `YES` while the SQL thread is stopped:

```
STOP REPLICA;
SET @@global.slave_run_triggers_for_rbr = YES;
START REPLICA;
```

Changing `slave_run_triggers_for_rbr` requires a privilege that permits setting this global variable, such as SUPER or the corresponding fine-grained global-variable grant.

The Five Conflict Types

A CDR trigger is bound to **one** conflict type. The type names encode *what the primary did* and *what the replica's local state implies*. The examples below use the table `t1 (a INT PRIMARY KEY, b INT, c CHAR(255))`, already in sync on both servers.

- **`INSERT_INSERT`**
  - Primary's operation: `INSERT`
  - Replica's local state: a row with the same key already exists
  - Typical underlying error: duplicate key

  ```
  -- On the replica (local write):
  INSERT INTO t1 VALUES (7, 1, 'local row');

  -- On the primary (replicated to the replica afterwards):
  INSERT INTO t1 VALUES (7, 2, 'primary row');
  -- The replicated insert collides with the replica's key 7
  ```

- **`UPDATE_UPDATE`**
  - Primary's operation: `UPDATE`
  - Replica's local state: the row exists but its values differ
  - Typical underlying error: before-image mismatch

```
-- Both servers hold (7, 300, 'in sync')

-- On the replica (local write):
UPDATE t1 SET b = 900 WHERE a = 7;

-- On the primary (replicated to the replica afterwards):
UPDATE t1 SET b = 500 WHERE a = 7;
-- The event's before-image says b = 300; the replica finds b = 900
```

- **`DELETE_UPDATE`**
  - Primary's operation: `DELETE`
  - Replica's local state: the row exists but has been changed locally
  - Typical underlying error: before-image mismatch

```
-- Both servers hold (7, 300, 'in sync')

-- On the replica (local write):
UPDATE t1 SET b = 900 WHERE a = 7;

-- On the primary (replicated to the replica afterwards):
DELETE FROM t1 WHERE a = 7;
-- The event's before-image says b = 300; the replica finds b = 900
```

- **`UPDATE_DELETE`**
  - Primary's operation: `UPDATE`
  - Replica's local state: the row is missing (already deleted locally)
  - Typical underlying error: record not found

```
-- On the replica (local write):
DELETE FROM t1 WHERE a = 7;

-- On the primary (replicated to the replica afterwards):
UPDATE t1 SET b = 500 WHERE a = 7;
-- The replica has no row with key 7 to update
```

- **`DELETE_DELETE`**
  - Primary's operation: `DELETE`
  - Replica's local state: the row is missing (already deleted locally)
  - Typical underlying error: record not found

```
-- On the replica (local write):
DELETE FROM t1 WHERE a = 7;

-- On the primary (replicated to the replica afterwards):
DELETE FROM t1 WHERE a = 7;
-- The replica has no row with key 7 to delete
```

Syntax

```
CREATE TRIGGER <trigger_name>
    FOR CONFLICT <conflict_type>
    ON <table_name>
    FOR EACH ROW
    <trigger_body>;
```

`<conflict_type>` is one of `INSERT_INSERT`, `UPDATE_UPDATE`, `DELETE_UPDATE`, `UPDATE_DELETE`, `DELETE_DELETE`.

A conflict trigger has **no** `BEFORE` / `AFTER` timing keyword — the `FOR CONFLICT` clause replaces it. The trigger fires only at conflict time, on the applier thread. A single `CREATE TRIGGER` statement binds to exactly one conflict type: conflict types cannot be chained with `OR`, and a conflict type cannot be combined with the regular `INSERT` / `UPDATE` / `DELETE` trigger events. To handle several conflict types on the same table, create a separate trigger for each.

Example skeleton:

```
DELIMITER //
CREATE TRIGGER cdr_upd_upd FOR CONFLICT UPDATE_UPDATE ON t1
FOR EACH ROW
BEGIN
    -- resolution logic here
END//
DELIMITER ;
```

Row Accessors: NEW, OLD, and ORG

CDR triggers introduce a third row accessor, `ORG`, alongside the familiar `NEW` and `OLD`:

Accessor	Represents	Writable
`NEW`	The **resolution image** — the row you want the replica to end up with.	Yes
`OLD`	The replica's **current local row** (its actual stored state).	No
`ORG`	The primary's **before-image** from the replication event — the state the primary expected.	No

`ORG` is the key to conflict resolution: it lets the trigger compare what the primary believed (`ORG`) against what the replica actually has (`OLD`), and construct what should result (`NEW`).

Accessor Availability per Conflict Type

Some images do not logically exist for certain conflicts, so the parser **rejects** triggers that reference an unavailable accessor at `CREATE TRIGGER` time:

Conflict type	`NEW`	`OLD`	`ORG`
`INSERT_INSERT`	Yes	Yes	No (an insert has no before-image)
`UPDATE_UPDATE`	Yes	Yes	Yes
`DELETE_UPDATE`	Yes	Yes	Yes
`UPDATE_DELETE`	Yes	No (row missing locally)	Yes
`DELETE_DELETE`	Yes	No (row missing locally)	Yes

Violations produce an error when the trigger is created:

```
-- ORG referenced in INSERT_INSERT:
CREATE TRIGGER trg1 FOR CONFLICT INSERT_INSERT ON t1
  FOR EACH ROW SET NEW.b = ORG.a;
-- ERROR 1363 (HY000): There is no ORG row in on INSERT_INSERT trigger

-- OLD referenced in a *_DELETE conflict:
CREATE TRIGGER trg1 FOR CONFLICT UPDATE_DELETE ON t1
  FOR EACH ROW SET NEW.b = OLD.a;
-- ERROR 1363 (HY000): There is no OLD row in on UPDATE_DELETE trigger

-- Attempting to assign to a read-only accessor:
CREATE TRIGGER trg1 FOR CONFLICT UPDATE_UPDATE ON t1
  FOR EACH ROW SET ORG.b = NEW.a;
-- ERROR 1362 (HY000): Updating of ORG row is not allowed in trigger
```

The Four Resolution Outcomes

Inside the trigger body, you choose one of four outcomes. A CDR trigger is not a general-purpose stored program for arbitrary DML against the conflicting table — these four outcomes are the supported ways to act on it.

Resolve by Assigning NEW (Apply the Row)

Set columns on `NEW` and let the trigger return normally. The applier writes your `NEW` image to the table:

- For `INSERT_INSERT`, `UPDATE_UPDATE`, `DELETE_UPDATE`: the existing local row is overwritten with `NEW`.
- For `UPDATE_DELETE`, `DELETE_DELETE`: a new row is inserted from `NEW`.

```
-- UPDATE_UPDATE: primary wins, keep an audit note
CREATE TRIGGER cdr_uu FOR CONFLICT UPDATE_UPDATE ON t1
FOR EACH ROW
  SET NEW.c = 'resolved: primary wins';
```

Do Nothing — Let the Default Operation Apply

If you do **not** assign `NEW` and return normally, the applier performs the operation's natural default for that conflict:

Conflict	Default action when `NEW` is left untouched
`INSERT_INSERT`	Overwrite the existing row with the primary's image.
`UPDATE_UPDATE`	Overwrite the local row with the primary's image.
`DELETE_UPDATE`	Delete the local row.
`UPDATE_DELETE`	Insert the primary's row.
`DELETE_DELETE`	No operation (the row is already absent).

Skip the Event — SIGNAL SQLSTATE '02TRG'

Raising the special SQLSTATE `02TRG` tells the applier to **gracefully ignore** this row event and continue replication. The table is left unmodified (the replica keeps `OLD`, or stays empty for a `*_DELETE` conflict). The SQL thread does **not** stop, and the replica's GTID position advances past the event.

```
CREATE TRIGGER cdr_skip FOR CONFLICT UPDATE_UPDATE ON t1
FOR EACH ROW
  SIGNAL SQLSTATE '02TRG';   -- keep the local row, drop the primary's change
```

`02TRG` is a "no data in trigger" subclass of the standard `02` ("no data") SQLSTATE class and is intentionally treated as *not an error*.

Halt Deliberately — SIGNAL SQLSTATE '51CDR'

When a conflict is genuinely unresolvable by policy, raise SQLSTATE `51CDR` with a custom error number to **stop the SQL thread on purpose**:

```
CREATE TRIGGER cdr_halt FOR CONFLICT DELETE_DELETE ON t1
FOR EACH ROW
  SIGNAL SQLSTATE '51CDR' SET MYSQL_ERRNO = 9001;
```

The SQL thread stops with the error number you supplied (`9001` above), letting an operator investigate. After fixing the underlying data, run `START REPLICA` to resume.

Any other unhandled error raised inside the trigger also stops the SQL thread, but `51CDR` is the intentional, documented way to do so.

How NEW Starts in Each Conflict Type

For `INSERT_INSERT` and `UPDATE_UPDATE`, `NEW` starts as the primary's incoming after-image, which the trigger can adjust before it is applied. The other three types differ:

- **`UPDATE_DELETE`** — the row is missing locally (no `OLD`), but the update event carries an after-image, so `NEW` starts **pre-filled** with it. Return without changes and the applier re-creates the row from `NEW` (the update becomes an insert); raise `SIGNAL SQLSTATE '02TRG'` to keep the row deleted instead.
- **`DELETE_UPDATE`** — a delete event carries no after-image, so the applier hands the trigger a blank `NEW` whose **primary key is set to `NULL`** as a sentinel. Leave the key `NULL` and the default action applies (the local row is deleted). Populate the key — for example from `OLD` — and the applier instead **converts the delete into an update**, writing `NEW` over the local row.
- **`DELETE_DELETE`** — the same blank-`NEW`, `NULL`-key sentinel, and the row is also missing locally (no `OLD`). Leave the key `NULL` for the default no-op, or populate `NEW` from `ORG` (the only image available) to re-create the row the primary tried to delete:

```
DELIMITER //
CREATE TRIGGER cdr_dd FOR CONFLICT DELETE_DELETE ON t1
FOR EACH ROW
BEGIN
    -- Re-create the row the primary tried to delete, from its before-image
    SET NEW.a = ORG.a;          -- setting the PK signals "apply this row"
    SET NEW.b = ORG.b;
    SET NEW.c = 'del_del: re-materialized';
END//
DELIMITER ;
```

If you instead leave `NEW.a` untouched, the conflict resolves as a no-op and the row stays absent.

The Before-Image Consistency Check

When a table has CDR triggers and an `UPDATE` or `DELETE` row event is applied, the applier locates the target row by primary key and compares the primary's before-image (`ORG`) against the row it actually found. A mismatch in the non-key columns means the replica's copy has diverged from what the primary expected, and is treated as a conflict.

What happens next depends on whether the mismatch is eligible to be routed to a CDR trigger:

- **Eligible → the trigger handles it.** The mismatch is delivered to your `UPDATE_UPDATE` trigger (for `UPDATE`) or `DELETE_UPDATE` trigger (for `DELETE`), exactly like the other conflict types.
- **Not eligible → the applier raises the consistency error and stops the SQL thread**, so an operator can reconcile rather than silently applying changes on top of unexpectedly diverged data:

```
ERROR 6001: Before image of replicated record does not match the slave local record
```

A mismatch is eligible for trigger routing only when all of these hold:

- the table has a **primary key**;
- the replica's slave_parallel_mode is `optimistic` or a more conservative setting (`conservative`, `minimal`, `none`); and
- the row event is not currently being applied as an optimistic speculative attempt. (This case is transient: the transaction rolls back and retries non-speculatively, and on retry the conflict is delivered to the trigger.)

So even on a table that has CDR triggers, error 6001 can still appear when the table lacks a primary key, or when the replica runs a parallel mode more aggressive than `optimistic`. This is why the recommended beta configuration keeps the replica at `optimistic` or below — see Limitations and Beta Caveats.

This check only exists when CDR triggers are present on the table. It is suppressed entirely when slave_exec_mode is `IDEMPOTENT` — but `IDEMPOTENT` mode is itself outside the supported beta configuration.

Worked Example: a Newest-Wins Policy

The following implements a *newest wins* policy on a table whose application sets `updated_at` on every write: whichever version of a row was written last — on any server — survives everywhere. Conflicts are also recorded in a local audit table, demonstrating that a trigger body may write to other tables.

```
-- ===== On the replica =====
STOP REPLICA;
SET @@global.slave_run_triggers_for_rbr = YES;
START REPLICA;

-- The replicated table (created on the primary):
--   CREATE TABLE customer_profile (
--       id         INT PRIMARY KEY,
--       email      VARCHAR(255),
--       updated_at DATETIME(6) NOT NULL   -- set by the application on every write
--   ) ENGINE=InnoDB;

-- Local audit table (created directly on the replica):
CREATE TABLE cdr_conflict_log (
    logged_at     DATETIME(6),
    conflict_type VARCHAR(16),
    id            INT,
    local_time    DATETIME(6),
    incoming_time DATETIME(6)
);

DELIMITER //

-- INSERT_INSERT: two servers created the same profile — the newer one survives
CREATE TRIGGER cdr_ins_ins FOR CONFLICT INSERT_INSERT ON customer_profile
FOR EACH ROW
BEGIN
    IF OLD.updated_at > NEW.updated_at THEN
        SIGNAL SQLSTATE '02TRG';   -- the local row is newer: keep it, skip the event
    END IF;
    -- otherwise return normally: the incoming row overwrites the local one (default)
END//

-- UPDATE_UPDATE: both servers updated the same row — the newer update survives
CREATE TRIGGER cdr_upd_upd FOR CONFLICT UPDATE_UPDATE ON customer_profile
FOR EACH ROW
BEGIN
    INSERT INTO cdr_conflict_log
        VALUES (NOW(6), 'UPDATE_UPDATE', ORG.id, OLD.updated_at, NEW.updated_at);
    IF OLD.updated_at > NEW.updated_at THEN
        SIGNAL SQLSTATE '02TRG';   -- the local update is newer: keep it
    END IF;
END//

-- DELETE_UPDATE: the incoming delete was decided against an older version of
-- the row than the local update — the newer local update survives
CREATE TRIGGER cdr_del_upd FOR CONFLICT DELETE_UPDATE ON customer_profile
FOR EACH ROW
BEGIN
    IF OLD.updated_at > ORG.updated_at THEN
        SIGNAL SQLSTATE '02TRG';   -- local update is newer than the state the
                                   -- primary deleted: keep the local row
    END IF;
    -- otherwise return normally: the local row is deleted (default)
END//

-- UPDATE_DELETE: the incoming update is newer than the local delete (a deleted
-- row keeps no timestamp, so this policy treats the update as newer):
-- return normally and NEW re-creates the row (default).
CREATE TRIGGER cdr_upd_del FOR CONFLICT UPDATE_DELETE ON customer_profile
FOR EACH ROW
BEGIN
    INSERT INTO cdr_conflict_log
        VALUES (NOW(6), 'UPDATE_DELETE', ORG.id, NULL, NEW.updated_at);
    -- To make deletes win instead, use: SIGNAL SQLSTATE '02TRG';
END//

-- DELETE_DELETE: both servers deleted the row — nothing to reconcile
```

```
CREATE TRIGGER cdr_del_del FOR CONFLICT DELETE_DELETE ON customer_profile
FOR EACH ROW
BEGIN
    INSERT INTO cdr_conflict_log
        VALUES (NOW(6), 'DELETE_DELETE', ORG.id, NULL, NULL);
    -- return normally: the row stays absent (default no-op)
END//

DELIMITER ;
```

Warning:

A timestamp-based policy converges only if every writer maintains the `updated_at` column on every write and the servers' clocks are synchronized. See Using CDR in Multi-Writer Topologies.

Using CDR in Multi-Writer Topologies

CDR is especially useful in multi-primary and ring topologies, where every server accepts writes and the goal is for all servers to **converge to the same consistent state**. In these setups:

- **Every participating server needs CDR triggers.** Each server is a replica of another, so each applier can encounter conflicts. A server without triggers stops on the first conflict its peers would have resolved.
- **The resolution policy must be deterministic and symmetric** — written so that every server reaches the **same end state** regardless of which side of the conflict it sees. *Newest wins* (as in the worked example above), *oldest wins*, or *the change from the server with the lowest server ID wins* are all convergent policies. A policy such as "my local row always wins" is **not**: each server would keep its own version, and the servers would disagree indefinitely.
- **Timestamp-based policies depend on the application and the clocks.** Every writer must maintain the timestamp column on every write, and the servers' clocks must be synchronized; otherwise "newest" is not well defined across servers.

Operational Notes

- **One trigger per conflict type per table.** Define separate triggers for the conflict types you want to handle. Conflict types you don't define fall back to the normal applier behavior (hard error, SQL thread stops).
- **Define triggers on servers that act as replicas.** A server that only acts as a primary does not need CDR triggers — a CDR trigger only fires on a replica's applier thread. For topologies where every server accepts writes, see Using CDR in Multi-Writer Topologies.
- **You can write to other tables.** The `NEW` / `OLD` / `ORG` accessors act on the conflicting table, but the trigger body may run DML against other tables (for example, an audit or exception log). Keep this lightweight — it runs inline on the applier thread.
- **Monitor the SQL thread.** A `51CDR` halt, an unhandled trigger error, or a before-image mismatch (error 6001) stops the SQL thread. Watch `Last_SQL_Errno` and `Last_SQL_Error` in SHOW REPLICA STATUS.
- **The Executed_triggers status variable** increments for each trigger invocation (CDR triggers included), which is useful for confirming the feature is firing.

Limitations and Beta Caveats

CDR triggers are **beta**. The current implementation is validated only within the configuration below. Outside it, behavior is unsupported, unspecified, or known-incomplete:

Area	Current state in beta
Row image format	Supported **only** with `binlog_row_image = FULL`. Other row-image settings are not supported.
System-versioned tables	**Not supported.** Do not create CDR triggers on tables with system versioning.

Area	Current state in beta
Parallel replication mode	Supported up to and including `slave_parallel_mode = optimistic` (that is, `none`, `minimal`, `conservative`, `optimistic`). `aggressive` is **not supported**.
`slave_exec_mode = IDEMPOTENT`	Behavior in combination with `IDEMPOTENT` is **unspecified**. `IDEMPOTENT` also disables the before-image consistency check.
Multiple triggers per conflict type	Creating more than one CDR trigger for the same conflict type on the same table is not prevented, but the effect is **unspecified**.
Coexistence with regular RBR triggers	A CDR trigger alongside a normal replica-side RBR trigger on the same table is intended to work but **may produce unexpected results**. Test thoroughly.
Mixing with multi-event triggers	A single `CREATE TRIGGER` statement is either a CDR trigger or a regular trigger, never both — see below.

No Mixing of CDR and Regular Trigger Syntax

MariaDB supports defining a single regular trigger that fires on more than one event by chaining events with `OR` (MDEV-10164). CDR triggers and this multi-event syntax are mutually exclusive within a single `CREATE TRIGGER` statement:

- A CDR trigger binds to exactly **one** conflict type — `FOR CONFLICT INSERT_INSERT OR UPDATE_UPDATE` is rejected.
- A conflict type cannot be mixed with regular events in the same statement — `FOR CONFLICT` combined with `INSERT`, `UPDATE`, or `DELETE` is rejected.
- The `FOR CONFLICT` clause replaces the `BEFORE` / `AFTER` timing keyword entirely; a regular trigger cannot carry a `FOR CONFLICT` clause.

Recommended Beta Configuration

```
[mariadbd]
binlog_row_image    = FULL
# On the replica, keep parallel mode at optimistic or below:
slave_parallel_mode = optimistic
# Do not use IDEMPOTENT exec mode with CDR triggers:
slave_exec_mode     = STRICT
```

Validate CDR triggers in a staging environment that mirrors your production topology before enabling them on a production replica, and watch the SQL thread closely during the initial rollout.

See Also

- Running Triggers on the Replica for Row-based Events
- Replication and Binary Log System Variables
- Trigger Overview
- Parallel Replication
- Multi-Master Ring Replication

This page is: Copyright © 2026 MariaDB. All rights reserved.

Benchmarking

Published Results

- Benchmark Results

Systemtap

- Useful scripts: keyword-index.html#DISK

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Memory Allocation

Quick Recommendations

> **Note:**
>
> Size MariaDB against **available RAM**, not total installed RAM.
>
> On a dedicated host, available RAM means total RAM minus operating system needs and other running services. In containers or VMs, use the effective memory limit and leave headroom for the OS, filesystem cache, and temporary per-connection allocations.

Start with these defaults, depending on your choice of storage engines:

- **Mostly InnoDB:** Set innodb_buffer_pool_size to about 60% to 70% of available RAM. Keep key_buffer_size small, such as `16M` to `64M`, unless you use MyISAM indexes.
- **Mostly MyISAM:** Set key_buffer_size to about 20% of available RAM. Set innodb_buffer_pool_size to `0` only if you do not use InnoDB at all.
- **Mixed engines:** Prioritize InnoDB first. Then size key_buffer_size to actual MyISAM index usage.
- Hosts with less than 4 GB of RAM: Use lower percentages and leave extra headroom.

Use these as starting points:

- Start from the packaged `my.cnf` or `my.ini`.
- Change only the major cache settings first.
- Fix slow queries with indexes, schema changes, or query changes before tuning smaller buffers.
- Keep the settings in the `[mariadb]` section.
- Restart the server after file-based changes.

Memory problems usually come from one of two causes:

- Global caches are too large.
- Per-connection buffers are too large for the current concurrency.

How MariaDB Uses Memory

MariaDB uses three broad kinds of memory:

- **Fixed global allocations**
  - Storage engine caches such as innodb_buffer_pool_size, key_buffer_size and aria_pagecache_buffer_size.
  - Optional caches such as query_cache_size.
- **Dynamic global structures**
  - Metadata- and connection-related structures such as table_open_cache, table_definition_cache, and thread_cache_size.
  - Limits such as max_connections do not allocate memory by themselves, but they raise the possible peak memory footprint.
- **Per-connection and per-query allocations**
  - Execution buffers such as join_buffer_size, mrr_buffer_size, read_buffer_size, and sort_buffer_size.

- Temporary in-memory tables controlled by max_heap_table_size and tmp_memory_table_size.
- Temporary blob storage and engine-specific work areas.

> **Warning:**
>
> On containers, cgroups[34], or managed environments, use the memory limit seen by MariaDB. Do not size caches against the physical RAM of the host.

Step 1

Confirm memory pressure.

Check whether the host is swapping or being killed by the OOM killer[35].

On the MariaDB side, inspect overall memory use:

```
SHOW GLOBAL STATUS LIKE 'memory_used';
```

If the server is swapping, reduce major caches before tuning anything else.

Step 2

Review non-default variables.

List variables that differ from the compiled defaults:

```
SELECT information_schema.system_variables.variable_name,
       information_schema.system_variables.default_value,
       information_schema.global_variables.variable_value
FROM information_schema.system_variables
JOIN information_schema.global_variables
  ON information_schema.system_variables.variable_name =
     information_schema.global_variables.variable_name
WHERE information_schema.system_variables.default_value <>
      information_schema.global_variables.variable_value
  AND information_schema.system_variables.default_value <> 0;
```

This can yield a result like this:

```
+-----------------------------+---------------+----------------+
| variable_name               | default_value | variable_value |
+-----------------------------+---------------+----------------+
| BACK_LOG                    | 150           | 80             |
| INNODB_IO_CAPACITY_MAX      | 4294967295    | 2000           |
| INNODB_LRU_FLUSH_SIZE       | 32            | 0              |
| HOST_CACHE_SIZE             | 128           | 279            |
| THREAD_CACHE_SIZE           | 256           | 151            |
| THREAD_POOL_SIZE            | 8             | 2              |
| INNODB_LOG_WRITE_AHEAD_SIZE | 512           | 4096           |
+-----------------------------+---------------+----------------+
```

If the server is running out of memory, the cause is often in this list.

> **Step 3**
>
> **Reduce the biggest consumers first.**
>
> Start with the variables that move total memory the most:
>
> - innodb_buffer_pool_size
> - key_buffer_size
> - aria_pagecache_buffer_size
> - query_cache_size
>
> Then review per-connection buffers, especially if concurrency is high:
>
> - join_buffer_size
> - mrr_buffer_size
> - sort_buffer_size
> - read_buffer_size
> - max_heap_table_size
>
> Large per-connection buffers are safe only when few sessions use them at the same time.

> **Step 4**
>
> **Re-test under real concurrency.**
>
> After each change, recheck swapping, memory use, and query latency.
>
> Avoid tuning many memory variables at once. Change the largest settings first, then re-measure.

What Is the Key Buffer?

MyISAM does two different things for caching:

- Index blocks (1KB each, BTree structured, from `.MYI` file) live in the “key buffer”.
- Data block caching (from .MYD file) is left to the OS, so be sure to leave a bunch of free space for this. Caveat: Some flavors of OS always claim to be using over 90%, even when there is really lots of free space.

To inspect the key buffer, run this statement:

```
SHOW GLOBAL STATUS LIKE 'Key%';
```

The result is something like this:

```
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| Key_blocks_not_flushed | 0      |
| Key_blocks_unused      | 107163 |
| Key_blocks_used        | 0      |
| Key_blocks_warm        | 0      |
| Key_read_requests      | 0      |
| Key_reads              | 0      |
| Key_write_requests     | 0      |
| Key_writes             | 0      |
+------------------------+--------+
```

Calculate Key_read_requests / Key_reads. If it is high (over 10), the key buffer is big enough, otherwise you should adjust the key_buffer_size value.

What Is the Buffer Pool?

InnoDB performs all its caching in the buffer pool, whose size is controlled by innodb_buffer_pool_size. By default, it contains 128MiB of data and index blocks from the open tables (see innodb_page_size), plus some maintenance overhead. Focus on total buffer pool size.

Another Algorithm

Use this to set the main cache settings to the minimum. This can be important for systems with lots of other processes, and/or where RAM is 2GB or smaller.

Run SHOW TABLE STATUS for all the tables in all the databases.

Add up index length for all MyISAM tables. Set key_buffer_size no larger than that size.

Add up `Data_length` + `Index_length` for all the InnoDB tables. Set innodb_buffer_pool_size to no more than 110% of that total.

If that leads to swapping, cut both settings back. Suggest cutting them down proportionately.

Run this to see the values for your system. (This query may run a while if you have a large set of tables.)

```
SELECT  ENGINE,
        ROUND(SUM(data_length) /1024/1024, 1) AS "Data MB",
        ROUND(SUM(index_length)/1024/1024, 1) AS "Index MB",
        ROUND(SUM(data_length + index_length)/1024/1024, 1) AS "Total MB",
        COUNT(*) "Num Tables"
    FROM  INFORMATION_SCHEMA.TABLES
    WHERE  table_schema NOT IN (
             "information_schema",
             "PERFORMANCE_SCHEMA",
             "SYS_SCHEMA",
             "ndbinfo")
    GROUP BY  ENGINE;
```

Query Memory Allocation

There are two variables that dictate how memory is allocated while MariaDB parses and executes a query. query_prealloc_size defines the standard buffer for query execution, and query_alloc_block_size defines the size of extra memory blocks if `query_prealloc_size` is not large enough. Getting these variables right can reduce memory fragmentation in the server.

Legacy and Specialized Scenarios

The sections below cover older platforms, specialized hardware, and edge cases. Use them only if they match your environment.

Mutex Bottleneck

This is mostly a historical concern.

Older MySQL and MariaDB releases could hit mutex contention around a few shared structures:

- MyISAM's `key_buffer`
- The Query Cache
- InnoDB's `buffer_pool`

On current MariaDB releases, this is much less important than:

- sizing the buffer pool correctly
- avoiding unnecessary query cache usage

- fixing slow queries and bad indexes
- sizing connection concurrency realistically

Legacy Platform Notes

These notes only apply to older or unusual deployments.

32-Bit Operating Systems

On a 32-bit OS, MariaDB cannot effectively use large amounts of RAM.

- RAM above 4 GB is not usable by a 32-bit process.
- The operating system may impose a much lower per-process limit.
- Large cache settings can fail with out-of-memory errors even when the host still has free RAM.

Example: FreeBSD's `maxdsiz`, which defaults to 512MB.

```
$ ulimit -a
...
max memory size (kbytes, -m) 524288
```

If you must run MariaDB on a 32-bit platform, keep cache settings conservative.

If you get an error like `[ERROR] /usr/libexec/mysqld: Out of memory (Needed xxx bytes)`, it probably means that MariaDB exceeded what the OS is willing to give it. Decrease the cache settings.

64-Bit OS with 32-Bit MariaDB

The OS can use more than 4 GB, but the MariaDB process still cannot.

If you are in this configuration, upgrade MariaDB to 64-bit if possible.

Until then, keep cache sizes below the effective 32-bit process limit.

64-Bit OS and MariaDB

This is the standard deployment target these days.

Use the recommendations at the top of this page. They are more accurate than the old platform-specific rules of thumb.

To find the values of the max_connections and thread_stack variable, issue this:

```
SELECT @@max_connections, @@thread_stack;
```

```
+-------------------+----------------+
| @@max_connections | @@thread_stack |
+-------------------+----------------+
|               151 |         299008 |
+-------------------+----------------+
```

Each thread takes some amount of RAM. This used to be about 200KB; 100 threads would be 20MB, not a significant size. If you have max_connections > 1000, you are talking about 200MB, maybe more. Having that many connections, though, likely implies other issues that should be addressed.

Thread pooling can change how concurrency behaves, but it does not remove the need to budget memory for peak active sessions.

Thread stack overruns are rare. If they occur, adjust `thread_stack` carefully and test.

table_open_cache

The OS has some limit on the number of open files it will let a process have. Each table needs 1 to 3 open files. Each partition is effectively a table. Most operations on a partitioned table open *all* partitions.

In Unix, ulimit tells you what the file limit is. The maximum value is in the tens of thousands, but sometimes it is set to only 1024. This limits you to about 300 tables. More discussion on ulimit

You can see how well your system is performing via SHOW GLOBAL STATUS; and computing the opens/second via Opened_files / Uptime If this is more than, say, 5, table_open_cache should be increased. If it is less than, say, 1, you might get improvement by decreasing table_open_cache.

table_open_cache defaults to `2000`.

Query Cache

Set query_cache_type to `OFF` and query_cache_size to `0` unless measurements show a clear benefit.

The query cache can be very fast for repeated identical `SELECT` statements, but it has significant trade-offs:

- Every eligible `SELECT` must consult it.
- A write to a table invalidates cached results for that table.
- Contention around the shared cache can hurt concurrency.

If you decide to use it:

- Keep query_cache_size small, usually no more than `50M` .
- Prefer query_cache_type = `DEMAND` .
- Use `SQL_CACHE` only on queries that benefit from reuse.

To see whether it helps, issue this statement:

```
SHOW GLOBAL STATUS LIKE 'Qc%';
```

Review the result:

```
+-------------------------+---------+
| Variable_name           | Value   |
+-------------------------+---------+
| Qcache_free_blocks      | 1       |
| Qcache_free_memory      | 1031248 |
| Qcache_hits             | 0       |
| Qcache_inserts          | 0       |
| Qcache_lowmem_prunes    | 0       |
| Qcache_not_cached       | 0       |
| Qcache_queries_in_cache | 0       |
| Qcache_total_blocks     | 1       |
+-------------------------+---------+
```

A rough read hit rate is:

- `Qcache_hits / Qcache_inserts`

If that ratio stays low, disable the query cache.

thread_cache_size

The thread_cache_size rarely needs manual tuning. Keep it small but non-zero unless you have evidence that connection creation is a bottleneck.

This setting controls how many disconnected threads are kept ready for reuse. It does not limit concurrent connections; max_connections does.

A very high value usually wastes memory without helping throughput.

Binary Logs

If you enable binary logging for replication or point-in-time recovery, logs will accumulate until they are purged.

Set a retention policy explicitly. On current MariaDB releases, prefer binlog_expire_logs_seconds. Older setups may still use expire_logs_days.

Choose a retention period that matches your recovery and replication requirements.

Swapping

Swapping usually hurts MariaDB badly.

For dedicated MariaDB hosts on Linux, a low `vm.swappiness` value often works best because it reduces the chance that large caches are pushed to disk under pressure.

In practice:

- Avoid heavy swapping.
- Leave enough free RAM for the operating system.
- Use a low `vm.swappiness` value if your platform and workload support it.

> **Note:**
>
> Do not use this as a substitute for proper memory sizing.

NUMA

NUMA[36] can cause uneven memory allocation on multi-socket systems.

To see if your machine is a NUMA host, run this command (available on Linux):

```
numactl --hardware
```

If the output yields `available: 1 nodes (0)`, all CPUs share memory equally. If it indicates 2 or mode nodes, the machine is a NUMA host.

If you see swapping or memory imbalance on a NUMA host, check the platform configuration before changing MariaDB settings. In some environments, interleaving memory allocation can avoid one NUMA node filling early while others stay underused.

Possible mitigations include:

- BIOS or firmware memory interleaving.
- OS-level NUMA configuration.
- Setting innodb_numa_interleave where supported.

Only tune NUMA after you confirm it is part of the problem.

Huge Pages

Huge pages can reduce page table overhead for large memory allocations.

For InnoDB-heavy systems with large buffer pools, explicit huge pages can help. Transparent huge pages are often less predictable and can hurt latency, so review your OS defaults and current MariaDB release behavior before changing anything.

If you use huge pages:

- Size them to match the intended buffer pool usage.
- Validate startup and allocation behavior.
- Measure latency and throughput after enabling them.

Do not enable huge pages blindly on small or memory-constrained systems.

ENGINE=MEMORY

The Memory Storage Engine is a little-used alternative to MyISAM and InnoDB. The data is not persistent, so it has limited uses. The size of a MEMORY table is limited to max_heap_table_size, which defaults to 16MB. I mention it in case you have changed the value to something huge; this would be stealing from other possible uses of RAM.

How to Set Variables

In your configuration file (for instance, `my.cnf`), set a variable like this (`variable_name = value`), in the `[mariadb]` section, then restart the server for the changes to take effect:

```
innodb_buffer_pool_size = 5G
```

For better readability of large values, use `M` for million and `G` for billion (note that the multiplier is `1024`):

- **1M** (Megabyte) = 1024^2 (1048576 bytes).
- **1G** (Gigabyte) = 1024^3 (1073741824 bytes).

Most settings can also be changed on the live system, by running a query like this:

```
SET @@global.key_buffer_size = 77000000;
```

> **Note:**
>
> Setting variables like that isn't persistent across server starts.
>
> However, you can do both – set the variable with a `SET` statement (so that it becomes immediately available), and in the configuration file (so that it's persistent when the server is restarted).
>
> Note that you cannot use `M` or `G` when setting variable values with a `SET` statement.

To see the setting of a global variable, run a statement like this:

```
SHOW GLOBAL VARIABLES LIKE "key_buffer_size";
+-----------------+----------+
| Variable_name   | Value    |
+-----------------+----------+
| key_buffer_size | 76996608 |
+-----------------+----------+
```

You may want to do both (SET and modify my.cnf) in order to make the change immediately and have it so that the next restart (for whatever reason) will again get the value.

Application Connection Pools

Application servers and web servers can create far more database connections than MariaDB can use efficiently.

- Size the application pool explicitly.
- Keep max_connections high enough for real traffic, but not so high that worst-case per-connection buffers can exhaust RAM.
- Prefer connection pooling over one-connection-per-request patterns.

Tools

- MySQLTuner
- TUNING-PRIMER

These tools can help identify obvious configuration problems, but they often assume worst-case memory use.

One common warning looks like this:

```
Maximum possible memory usage: 31.3G (266% of installed RAM)
```

Treat that as an upper bound, not a prediction. It usually assumes all max_connections are active and all are using large per-query buffers at the same time.

Another warning can look like this:

```
Total fragmented tables: 23
```

This implies that OPTIMIZE TABLE *might* help. Use it for tables with either a high percentage of "free space" (see SHOW TABLE STATUS) or where you know you run a lot of `DELETE` and/or `UPDATE` statements. Still, don't bother to optimize too often. Once a month should suffice.

See Also

- Configuring MariaDB for Optimal Performance
- InnoDB Buffer Pool
- Server System Variables
- What to Do if MariaDB Doesn't Start

Attribution

Rick James wrote the original version of this page. Rick James' site, original source: random.

This page is licensed: CC BY-SA / Gnu FDL

Hardware Optimization

Better hardware is one of the easiest ways to improve performance.

As a general rule of thumb, hardware should be improved in the following order:

Memory

Memory is the most important factor as it allows you to adjust the Server System Variables. More memory means larger key and table caches can be stored in memory so that disk access, an order of magnitude slower, is reduced.

Simply adding more memory may not result in drastic improvements if the server variables are not set to make use of the extra available memory.

Using more RAM slots on the motherboard increases the bus frequency, and there will be more latency between the RAM and the CPU. So, using the highest RAM size per slot is preferable.

Storage

Fast storage access is critical, as ultimately it's where the data resides. The key figure is the random seek time, a measurement of how fast the physical can retrieve data to access the data, so choose storage with as low a random seek time as possible.

Memory can make up for slow storage, but only that table data is accessed.

You can also add dedicated storage for temporary files and transaction logs.

Network

CPU

Although hardware bottlenecks often fall elsewhere, faster processors allow calculations to be performed more quickly, and the results sent back to the client more quickly. Besides processor speed, the processor's bus speed and cache size are also important factors to consider. Pipelining/ prefetching of instructions/data and other aspects of CPU architecture also have a significant effect.

If in doubt, ways conduct a benchmark that resembles the database workload of intended use.

See Also

- Configuring Linux for MariaDB

This page is licensed: CC BY-SA / Gnu FDL

Connection Redirection Mechanism in the MariaDB Client/Server Protocol

> **Note:**
>
> This functionality is available from MariaDB 11.3.

A connection redirection mechanism was added via MDEV-15935.

Redirection mechanisms are widely used in proxy-based scenarios.

Previously, when multiple servers shared one proxy, the proxy forwarded all packets between servers and clients. Thus, the proxy added latency, consuming computing resources and impacting overall performance. For scenarios with many short connections, such as WordPress, latency can be a critical issue.

With a redirection mechanism, much like HTTP redirects or Oracle redirected connections, clients get the servers' address from proxies and connect to servers transparently, without latency and without wasting resources.

Usage

Redirection is handled through a new system variable, redirect_url. The value defaults to an empty string, but can also contain a connection string in the conventional format (in the style of a Connector/C etc. connection url), like this:

```
{mysql,mariadb}://host[:port]
```

Here, *`host`* is an arbitrary string not containing colons, and *`port`* is a number between `0` and `65535` inclusive.

This variable is appended to the default value of the session_track_system_variables system variable. Unless empty, clients are redirected to the specified server.

Use Cases

- Always redirect all clients to a new location:
  - Set `@@global.redirect_url`, or start the server with `--redirect-url=` *`url`*, or add that setting to the server configuration file.
- Redirect to a group of servers randomly:

    - Create a table with connection URLs, one per row.
    - Use an SQL script that selects a random row from that table, and sets `@@redirect_url` to this value.
    - Specify this script in the `--init-connect` server parameter.
- Dynamically redirect from the primary server to one of the replicas:
    - Same as before, but use INFORMATION_SCHEMA.PROCESSLIST to get the list of active replicas.

Example

```
SET GLOBAL redirect_url="mysql://mariadb.org:12345";
```

Invalid formats are not permitted:

```
SET GLOBAL redirect_url="mysql://mariadb.org:";
ERROR 1231 (42000): Variable 'redirect_url' can't be set to the value of 'mysql://mariadb.org:'
```

This page is licensed: CC BY-SA / Gnu FDL

Reference

See SQL Structure.

Explore the SQL structure in MariaDB Server. This section provides a fundamental overview of SQL syntax, components, and concepts that form the basis for interacting with your database.

See SQL Statements.

Reference for MariaDB Server SQL statements. Find the syntax, clauses, and options for data definition, data manipulation, administrative, and other SQL statements.

See SQL Functions.

Discover functions and procedures in MariaDB.

See Data Types.

Comprehensive MariaDB data types reference. Complete guide for numeric, string, date/time, spatial, and JSON types with storage specifications.

See Plugins.

Explore plugins in MariaDB Server. This section details how to extend database functionality, security, and performance by leveraging various loadable plugins, from authentication to storage engines.

See System Tables.

Explore system tables in MariaDB Server. These internal tables store metadata and configuration information about the database, essential for administration, monitoring, and advanced querying.

See Error Codes.

Reference for MariaDB Server error codes. Look up error numbers, their symbolic names, and the conditions that trigger them, plus operating-system error codes.

See Client/Server Protocol.

Understand the MariaDB client/server protocol. This section details how clients communicate with the server, including connection, authentication, query execution, and result set handling.

See Product Development.

Provides background information and high-level resources aimed at engineers developing new MariaDB features and contributing to the ecosystem.

This page is licensed: CC BY-SA / Gnu FDL

SQL Structure

See SQL Language Structure.

Explore the SQL language structure in MariaDB Server. This section provides fundamental concepts, syntax rules, and common elements that form the building blocks of SQL queries and commands.

See Joins, Subqueries, and Set.

Explore joins, subqueries, and set operations in MariaDB SQL. This section details how to combine data from multiple tables, nest queries, and perform set-based operations for complex data retrieval.

See Geometry.

Explore geometric data types and functions in MariaDB Server. This section details how to store, query, and manipulate spatial data, enabling geospatial applications within your database.

See NoSQL.

Explore MariaDB Server's NoSQL capabilities. This section details how to store and query schemaless data, including JSON, and how to integrate with other NoSQL data sources, enhancing flexibility.

See Operators.

Learn about operators in MariaDB Server SQL. This section details arithmetic, comparison, logical, and bitwise operators used in expressions and conditions for data manipulation and querying.

See Sequences.

Learn about sequences in MariaDB Server. This section details how to create and manage sequences for generating unique numbers, often used for primary keys and other auto-incrementing values.

See Temporal Tables.

Explore temporal tables in MariaDB Server. This section details how to manage data with system-versioning and application-time periods, enabling historical data tracking and time-aware queries.

See Vectors.

Explore vector data types. This section details how to store and manage numerical arrays, enabling efficient vector similarity search and machine learning applications within your database.

This page is licensed: CC BY-SA / Gnu FDL

SQL Language Structure

See Binary Literals.

Binary literal syntax in MariaDB: b'value', B'value', and 0bvalue.

See Date and Time Literals.

Syntax for DATE, TIME, DATETIME, and TIMESTAMP literals, including SQL- standard and ODBC forms.

See Hexadecimal Literals.

Hexadecimal literal syntax in MariaDB, such as x'value' and 0xvalue.

See Identifier Case Sensitivity.

How identifier case sensitivity depends on the object type and the operating system.

See Identifier Names.

Rules for naming identifiers such as databases, tables, columns, and other objects.

See Identifier Qualifiers.

Using qualifiers (such as db.table.column) to reference database objects in SQL.

See Identifier to File Name Mapping.

How MariaDB maps identifiers to file names on disk.

See Numeric Literals.

Numeric literal syntax for integers and decimals, including signs and exponents.

See Reserved Words.

The list of reserved words in MariaDB.

See Boolean Literals.

Boolean literals: TRUE and FALSE as synonyms for 1 and 0.

See Schema Qualifiers.

Schema qualifiers manage SQL mode dependent behavior for data types and functions.

See String Literals.

String literal syntax, quoting, and escape sequences.

See Table Value Constructors.

The VALUES table value constructor, which produces a set of rows.

See User-Defined Variables.

Session-scoped user-defined variables (@var), and how to set and use them.

This page is licensed: CC BY-SA / Gnu FDL

Binary Literals

Binary literals can be written in one of the following formats: `b'value'`, `B'value'` or `0bvalue`, where `value` is a string composed by `0` and `1` digits.

Binary literals are interpreted as binary strings, and are convenient to represent VARBINARY, BINARY or BIT values.

To convert a binary literal into an integer, just add `0`.

Examples

Printing the value as a binary string:

```
SELECT 0b1000001;
+-----------+
| 0b1000001 |
+-----------+
| A         |
+-----------+
```

Converting the same value into a number:

```
SELECT 0b1000001+0;
+-------------+
| 0b1000001+0 |
+-------------+
|          65 |
+-------------+
```

See Also

- BIN()

This page is licensed: CC BY-SA / Gnu FDL

Date and Time Literals

Standard syntaxes

MariaDB supports the SQL standard and ODBC syntaxes for DATE, TIME, and TIMESTAMP literals.

SQL standard syntax:

- `DATE` 'string'
- `TIME` 'string'
- `TIMESTAMP` 'string'

ODBC syntax:

- {d 'string'}
- {t 'string'}
- {ts 'string'}

The timestamp literals are treated as DATETIME literals, because in MariaDB the range of `DATETIME` is closer to the `TIMESTAMP` range in the SQL standard.

`string` is a string in a proper format, as explained below.

`DATE` literals

A `DATE` string is a string in one of the following formats: `'YYYY-MM-DD'` or `'YY-MM-DD'`. Note that any punctuation character can be used as delimiter. All delimiters must consist of 1 character. Different delimiters can be used in the same string. Delimiters are optional (but if one delimiter is used, all delimiters must be used).

A `DATE` literal can also be an integer, in one of the following formats: `YYYYMMDD` or `YYMMDD`.

All the following `DATE` literals are valid, and they all represent the same value:

```
'19940101'
'940101'
'1994-01-01'
'94/01/01'
'1994-01/01'
'94:01!01'
19940101
940101
```

`DATETIME` literals

A `DATETIME` string is a string in one of the following formats: `'YYYY-MM-DD HH:MM:SS'` or `'YY-MM-DD HH:MM:SS'`. Note that any punctuation character can be used as delimiter for the date part and for the time part. All delimiters must consist of 1 character. Different delimiters can be used in the same string. The hours, minutes and seconds parts can consist of one character. For this reason, delimiters are mandatory for `DATETIME` literals.

The delimiter between the date part and the time part can be a `T` or any sequence of space characters (including tabs, new lines and carriage returns).

A `DATETIME` literal can also be a number, in one of the following formats: `YYYYMMDDHHMMSS`, `YYMMDDHHMMSS`, `YYYYMMDD` or `YYMMDD`. In this case, all the time subparts must consist of 2 digits.

All the following `DATE` literals are valid, and they all represent the same value:

```
'1994-01-01T12:30:03'
'1994/01/01\n\t 12+30+03'
'1994/01\\01\n\t 12+30-03'
'1994-01-01 12:30:3'
```

`TIME` literals

A `TIME` string is a string in one of the following formats: `'D HH:MM:SS'`, `'HH:MM:SS`, `'D HH:MM'`, `'HH:MM'`, `'D HH'`, or `'SS'`. `D` is a value from `0` to `34` which represents days. `:` is the only allowed delimiter for `TIME` literals. Delimiters are mandatory, with an exception: the `'HHMMSS'` format is allowed. When delimiters are used, each part of the literal can consist of one character.

A `TIME` literal can also be a number in one of the following formats: `HHMMSS`, `MMSS`, or `SS`.

The following literals are equivalent:

```
'09:05:00'
'9:05:0'
'9:5:0'
'090500'
```

2-digit years

The year part in `DATE` and `DATETIME` literals is determined as follows:

- `70` - `99` = `1970` - `1999`
- `00` - `69` = `2000` - `2069`

Microseconds

`DATETIME` and `TIME` literals can have an optional microseconds part. For both string and numeric forms, it is expressed as a decimal part. Up to 6 decimal digits are allowed. Examples:

```
'12:30:00.123456'
123000.123456
```

See Microseconds in MariaDB for details.

Date and time literals and the `SQL_MODE`

Unless the SQL_MODE `NO_ZERO_DATE` flag is set, some special values are allowed: the `'0000-00-00'` `DATE`, the `'00:00:00'` `TIME`, and the `0000-00-00 00:00:00` `DATETIME`.

If the `ALLOW_INVALID_DATES` flag is set, the invalid dates (for example, 30th February) are allowed. If not, if the `NO_ZERO_DATE` is set, an error is produced; otherwise, a zero-date is returned.

Unless the `NO_ZERO_IN_DATE` flag is set, each subpart of a date or time value (years, hours…) can be set to `0`.

See also

- Date and time units

This page is licensed: CC BY-SA / Gnu FDL

Hexadecimal Literals

Hexadecimal literals can be written using any of the following syntaxes:

- x'`value`'
- X'`value`' (SQL standard)
- 0x`value` (ODBC)

`value` is a sequence of hexadecimal digits (from `0` to `9` and from `A` to `F`). The case of the digits does not matter. With the first two syntaxes, `value` must consist of an even number of digits. With the last syntax, digits can be even, and they are treated as if they had an extra 0 at the beginning.

Normally, hexadecimal literals are interpreted as binary string, where each pair of digits represents a character. When used in a numeric context, they are interpreted as integers. (See the example below). In no case can a hexadecimal literal be a decimal number.

The first two syntaxes; `X' value '` and `x' value`, follow the SQL standard, and behave as a string in all contexts in MariaDB. The latter syntax, `0x value`, is a MySQL/MariaDB extension for hex hybrids and behaves as a string or as a number depending on context. MySQL treats all syntaxes the same, so there may be different results in MariaDB and MySQL (see below).

Examples

Representing the `a` character with the three syntaxes explained above:

```
SELECT x'61', X'61', 0x61;
+-------+-------+------+
| x'61' | X'61' | 0x61 |
+-------+-------+------+
| a     | a     | a    |
+-------+-------+------+
```

Hexadecimal literals in a numeric context:

```
SELECT 0 + 0xF, -0xF;
+---------+------+
| 0 + 0xF | -0xF |
+---------+------+
|      15 |  -15 |
+---------+------+
```

Fun With Types

```
CREATE TABLE t1 (a INT, b VARCHAR(10));
INSERT INTO t1 VALUES (0x31, 0x61),(COALESCE(0x31), COALESCE(0x61));

SELECT * FROM t1;
+------+------+
| a    | b    |
+------+------+
|   49 | a    |
|    1 | a    |
+------+------+
```

The reason for the differing results above is that when 0x31 is inserted directly to the column, it's treated as a number, while when 0x31 is passed to COALESCE(), it's treated as a string, because:

- `HEX` values have a string data type by default.
- `COALESCE()` has the same data type as the argument.

Differences Between MariaDB and MySQL

```
SELECT x'0a'+0;
+---------+
| x'0a'+0 |
+---------+
|       0 |
+---------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1292): Truncated incorrect DOUBLE value: '\x0A'

SELECT X'0a'+0;
+---------+
| X'0a'+0 |
+---------+
|       0 |
+---------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1292): Truncated incorrect DOUBLE value: '\x0A'

SELECT 0x0a+0;
+--------+
| 0x0a+0 |
+--------+
|     10 |
+--------+
```

In MySQL (up until at least MySQL 8.0.26):

```
SELECT x'0a'+0;
+---------+
| x'0a'+0 |
+---------+
|      10 |
+---------+

SELECT X'0a'+0;
+---------+
| X'0a'+0 |
+---------+
|      10 |
+---------+

SELECT 0x0a+0;
+--------+
| 0x0a+0 |
+--------+
|     10 |
+--------+
```

See Also

- HEX()
- UNHEX()

This page is licensed: CC BY-SA / Gnu FDL

Identifier Case Sensitivity

Whether objects are case sensitive or not is partly determined by the underlying operating system. Unix-based systems are case sensitive, Windows is not, while Mac OS X is usually case insensitive by default, but devices can be configured as case sensitive using Disk Utility.

Database, table, table aliases, and trigger names are affected by the system's case sensitivity, while index, column, column aliases, stored routine, and event names are never case sensitive.

Log file group names are case sensitive.

> **Note:**
>
> In some cases, the table exists but that you are referring to it incorrectly:
>
> - Because MariaDB uses directories and files to store databases and tables, database and table names are case-sensitive if they are located on a file system that has case-sensitive file names.
> - Even for file systems that are not case-sensitive, such as on Windows, all references to a given table within a query must use the same lettercase.

The lower_case_table_names server system variable plays a key role. It determines whether table names, aliases, and database names are compared in a case-sensitive manner. If set to `0` (the default on Unix-based systems), table names and aliases, and database names are compared in a case-sensitive manner. If set to `1` (the default on Windows), names are stored in lowercase and not compared in a case-sensitive manner. If set to `2` (the default on Mac OS X), names are stored as declared, but compared in lowercase.

It is thus possible to make Unix-based systems behave like Windows and ignore case-sensitivity. The reverse is not true before Windows 10, as the underlying Windows filesystem could not support this. It is possible since Windows 10, although case insensitivity is still the default operating system setting.

Even on case-insensitive systems, you are required to use the same case consistently within the same statement. The following statement fails, as it refers to the table name in a different case.

```
SELECT * FROM a_table WHERE A_table.id>10;
```

For a full list of identifiers naming rules, see Identifier Names.

> **Note:**
>
> Please note that lower_case_table_names is a database initialization parameter. This means that, along with innodb_page_size, this variable must be set before running mariadb-install-db, and will not change the behavior of servers unless applied before the creation of core system databases.

This page is licensed: CC BY-SA / Gnu FDL

Identifier Names

Databases, tables, indexes, columns, aliases, views, stored routines, triggers, events, variables, partitions, tablespaces, savepoints, labels, users, roles, are collectively known as identifiers, and have certain rules for naming.

Identifiers may be quoted using the backtick character (`` ` ``). Quoting is optional for identifiers that don't contain special characters, or for identifiers that are not reserved words. If the `ANSI_QUOTES` SQL_MODE flag is set, double quotes (`"`) can also be used to quote identifiers. If the MSSQL flag is set, square brackets (`[` and `]`) can be used for quoting.

Even when using reserved words as names, fully qualified names do not need to be quoted. For example, `test.select` has only one possible meaning, so it is correctly parsed even without quotes.

Unquoted

The following characters are valid, and allow identifiers to be unquoted:

- ASCII: [0-9,a-z,A-Z$_] (numerals 0-9, basic Latin letters, both lowercase and uppercase, dollar sign, underscore)
- Extended: U+0080 .. U+FFFF

Quoted

The following characters are valid, but identifiers using them must be quoted:

- ASCII: U+0001 .. U+007F (full Unicode Basic Multilingual Plane (BMP) except for U+0000)
- Extended: U+0080 .. U+FFFF
- Identifier quotes can themselves be used as part of an identifier, as long as they are quoted.

Further Rules

There are a number of other rules for identifiers:

- Identifiers are stored as Unicode (UTF-8)
- Identifiers may or may not be case-sensitive. See Identifier Case-sensitivity.
- Database, table and column names can't end with space characters
- Identifier names may begin with a numeral, but can't only contain numerals unless quoted.
- An identifier starting with a numeral, followed by an 'e', may be parsed as a floating point number, and needs to be quoted.
- Identifiers are not permitted to contain the ASCII NUL character (U+0000) and supplementary characters (U+10000 and higher).
- Names such as 5e6, 9e are not prohibited, but it's strongly recommended not to use them, as they could lead to ambiguity in certain contexts, being treated as a number or expression.
- User variables cannot be used as part of an identifier, or as an identifier in an SQL statement.

Quote Character

The regular quote character is the backtick character - ```, but if the `ANSI_QUOTES` SQL_MODE option is specified, a regular double quote - `"` may be used as well.

The backtick character can be used as part of an identifier. In that case the identifier needs to be quoted. The quote character can be the backtick, but in that case, the backtick in the name must be escaped with another backtick.

Maximum Length

- Databases, tables, columns, indexes, constraints, stored routines, triggers, events, views, tablespaces, servers and log file groups have a maximum length of 64 characters.
- Compound statement labels have a maximum length of 16 characters.
- Aliases have a maximum length of 256 characters, except for column aliases in CREATE VIEW statements, which are checked against the maximum column length of 64 characters (not the maximum alias length of 256 characters).
- Users have a maximum length of 80 characters.
- Roles have a maximum length of 128 characters.
- Multi-byte characters do not count extra towards the character limit.

Multiple Identifiers

MariaDB allows the column name to be used on its own if the reference will be unambiguous, or the table name to be used with the column name, or all three of the database, table and column names. A period is used to separate the identifiers, and the period can be surrounded by spaces.

Examples

Using the period to separate identifiers:

```
CREATE TABLE t1 (i int);

INSERT INTO t1(i) VALUES (10);

SELECT i FROM t1;
+------+
| i    |
+------+
|   10 |
+------+

SELECT t1.i FROM t1;
+------+
| i    |
+------+
|   10 |
+------+

SELECT test.t1.i FROM t1;
+------+
| i    |
+------+
|   10 |
+------+
```

The period can be separated by spaces:

```
SELECT test . t1 . i FROM t1;
+------+
| i    |
+------+
|   10 |
+------+
```

Resolving ambiguity:

```
CREATE TABLE t2 (i int);

SELECT i FROM t1 LEFT JOIN t2 ON t1.i=t2.i;
ERROR 1052 (23000): Column 'i' in field list is ambiguous

SELECT t1.i FROM t1 LEFT JOIN t2 ON t1.i=t2.i;
+------+
| i    |
+------+
|   10 |
+------+
```

Creating a table with characters that require quoting:

```
CREATE TABLE 123% (i int);
ERROR 1064 (42000): You have an error in your SQL syntax;
  check the manual that corresponds to your MariaDB server version for the right syntax
  to use near '123% (i int)' at line 1

CREATE TABLE `123%` (i int);
Query OK, 0 rows affected (0.85 sec)

CREATE TABLE `TABLE` (i int);
Query OK, 0 rows affected (0.36 sec)
```

Using double quotes as a quoting character:

```
CREATE TABLE "SELECT" (i int);
ERROR 1064 (42000): You have an error in your SQL syntax;
  check the manual that corresponds to your MariaDB server version for the right syntax
  to use near '"SELECT" (i int)' at line 1

SET sql_mode='ANSI_QUOTES';
Query OK, 0 rows affected (0.03 sec)

CREATE TABLE "SELECT" (i int);
Query OK, 0 rows affected (0.46 sec)
```

Using an identifier quote as part of an identifier name:

```
SHOW VARIABLES LIKE 'sql_mode';
+---------------+-------------+
| Variable_name | Value       |
+---------------+-------------+
| sql_mode      | ANSI_QUOTES |
+---------------+-------------+

CREATE TABLE "fg`d" (i int);
Query OK, 0 rows affected (0.34 sec)
```

Creating the table named `*` (Unicode number: U+002A) requires quoting.

```
CREATE TABLE `*` (a INT);
```

Floating point ambiguity:

```
CREATE TABLE 8984444cce5d (x INT);
Query OK, 0 rows affected (0.38 sec)

CREATE TABLE 8981e56cce5d (x INT);
ERROR 1064 (42000): You have an error in your SQL syntax;
  check the manual that corresponds to your MariaDB server version for the right syntax
  to use near '8981e56cce5d (x INT)' at line 1

CREATE TABLE `8981e56cce5d` (x INT);
Query OK, 0 rows affected (0.39 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Identifier Qualifiers

Qualifiers are used within SQL statements to reference data structures, such as databases, tables, or columns. For example, typically a `SELECT` query contains references to some columns and at least one table.

Qualifiers can be composed by one or more identifiers, where the initial parts affect the context within which the final identifier is interpreted:

- For a database, only the database identifier needs to be specified.
- For objects which are contained in a database (like tables, views, functions, etc.) the database identifier can be specified. If no database is specified, the current database is assumed (see USE and DATABASE() for more details). If there is no default database and no database is specified, an error is issued.
- For column names, the table and the database are generally obvious from the context of the statement. It is however possible to specify the table identifier, or the database identifier plus the table identifier.
- An identifier is fully-qualified if it contains all possible qualifiers, for example, the following column is fully qualified: `db_name.tbl_name.col_name`.

If a qualifier is composed by more than one identifier, a dot (.) must be used as a separator. All identifiers can be quoted individually. Extra spacing (including new lines and tabs) is allowed.

All the following examples are valid:

- db_name.tbl_name.col_name
- tbl_name
- `db_name`.`tbl_name`.`col_name`
- `db_name` . `tbl_name`
- db_name. tbl_name

If a table identifier is prefixed with a dot (.), the default database is assumed. This syntax is supported for ODBC compliance, but has no practical effect on MariaDB. These qualifiers are equivalent:

- tbl_name
- . tbl_name
- .`tbl_name`
- . `tbl_name`

For DML statements, it is possible to specify a list of the partitions using the PARTITION clause. See Partition Pruning and Selection for details.

See Also

- Identifier Names
- USE
- DATABASE()

This page is licensed: CC BY-SA / Gnu FDL

Identifier to File Name Mapping

Some identifiers map to a file name on the filesystem. Databases each have their own directory, while, depending on the storage engine, table names and index names may map to a file name.

Not all characters that are allowed in table names can be used in file names. Every filesystem has its own rules of what characters can be used in file names. To let the user create tables using all characters allowed in the SQL Standard and to not depend on whatever particular filesystem a particular database resides, MariaDB encodes “potentially unsafe” characters in the table name to derive the corresponding file name.

This is implemented using a special character set. MariaDB converts a table name to the “filename” character set to get the file name for this table. And it converts the file name from the “filename” character set to, for example, utf8 to get the table name for this file name.

The conversion rules are as follows: if the identifier is made up only of basic Latin numbers, letters and/or the underscore character, the encoding matches the name (see however Identifier Case Sensitivity). Otherwise they are encoded according to the following table:

Code Range	Pattern	Number	Used	Unused	Blocks
00C0..017F	[@][0..4][g..z]	5*20= 100	97	3	Latin-1 Supplement + Latin Extended-A
0370..03FF	[@][5..9][g..z]	5*20= 100	88	12	Greek and Coptic
0400..052F	[@][g..z][0..6]	20*7= 140	137	3	Cyrillic + Cyrillic Supplement
0530..058F	[@][g..z][7..8]	20*2= 40	38	2	Armenian
2160..217F	[@][g..z][9]	20*1= 20	16	4	Number Forms

Code Range	Pattern	Number	Used	Unused	Blocks
0180..02AF	[@][g..z][a..k]	20*11=220	203	17	Latin Extended-B + IPA Extensions
1E00..1EFF	[@][g..z][l..r]	20*7= 140	136	4	Latin Extended Additional
1F00..1FFF	[@][g..z][s..z]	20*8= 160	144	16	Greek Extended
....	[@][a..f][g..z]	6*20= 120	0	120	RESERVED
24B6..24E9	[@][@][a..z]	26	26	0	Enclosed Alphanumerics
FF21..FF5A	[@][a..z][@]	26	26	0	Halfwidth and Fullwidth forms

Code Range values are UCS-2.

All of this encoding happens transparently at the filesystem level with one exception.

Examples

Find the file name for a table with a non-Latin1 name:

```
SELECT CAST(CONVERT("this_is_таблица" USING filename) AS BINARY);
+-----------------------------------------------------------------+
| CAST(CONVERT("this_is_таблица" USING filename) AS BINARY)       |
+-----------------------------------------------------------------+
| this_is_@y0@g0@h0@r0@o0@i1@g0                                   |
+-----------------------------------------------------------------+
```

Find the table name for a file name:

```
SELECT CONVERT(_filename "this_is_@y0@g0@h0@r0@o0@i1@g0" USING utf8);
+-------------------------------------------------------------+
| CONVERT(_filename "this_is_@y0@g0@h0@r0@o0@i1@g0" USING utf8) |
+-------------------------------------------------------------+
| this_is_таблица                                             |
+-------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Numeric Literals

Numeric literals are written as a sequence of digits from `0` to `9`. Initial zeros are ignored. A sign can always precede the digits, but it is optional for positive numbers. In decimal numbers, the integer part and the decimal part are divided with a dot (`.`).

If the integer part is zero, it can be omitted, but the literal must begin with a dot.

The notation with exponent can be used. The exponent is preceded by an `E` or `e` character. The exponent can be preceded by a sign and must be an integer. A number `N` with an exponent part `X`, is calculated as `N * POW(10, X)`.

In some cases, adding zeroes at the end of a decimal number can increment the precision of the expression where the number is used. For example, PI() by default returns a number with 6 decimal digits. But the `PI()+0.0000000000` expression (with 10 zeroes) returns a number with 10 decimal digits.

Hexadecimal literals are interpreted as numbers when used in numeric contexts.

Examples

```
10
+10
-10
```

All these literals are equivalent:

```
0.1
.1
+0.1
+.1
```

With exponents:

```
0.2E3 -- 0.2 * POW(10, 3) = 200
.2e3
.2e+2
1.1e-10 -- 0.00000000011
-1.1e10 -- -11000000000
```

This page is licensed: CC BY-SA / Gnu FDL

Reserved Words

The following is a list of all reserved words in MariaDB.

Reserved words cannot be used as Identifiers, unless they are quoted.

The definitive list of reserved words for each version can be found by examining the `sql/lex.h` and `sql/sql_yacc.yy` files.

Reserved Words

Keyword
ACCESSIBLE
ADD
ALL
ALTER
ANALYZE
AND
AS
ASC
ASENSITIVE
BEFORE
BETWEEN
BIGINT
BINARY
BLOB
BOTH
BY
CALL
CASCADE
CASE
CHANGE
CHAR

Keyword
CHARACTER
CHECK
COLLATE
COLUMN
CONDITION
CONSTRAINT
CONTINUE
CONVERSION (> 12.2)
CONVERT
CREATE
CROSS
CURRENT_DATE
CURRENT_ROLE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR
DATABASE
DATABASES
DAY_HOUR
DAY_MICROSECOND
DAY_MINUTE
DAY_SECOND
DEC
DECIMAL
DECLARE
DEFAULT
DELAYED
DELETE
DELETE_DOMAIN_ID
DESC
DESCRIBE
DETERMINISTIC
DISTINCT
DISTINCTROW
DIV
DO_DOMAIN_IDS
DOUBLE
DROP

Keyword
DUAL
EACH
ELSE
ELSEIF
ENCLOSED
ESCAPED
EXCEPT
EXISTS
EXIT
EXPLAIN
FALSE
FETCH
FLOAT
FLOAT4
FLOAT8
FOR
FORCE
FOREIGN
FROM
FULLTEXT
GENERAL
GRANT
GROUP
HAVING
HIGH_PRIORITY
HOUR_MICROSECOND
HOUR_MINUTE
HOUR_SECOND
IF
IGNORE
IGNORE_DOMAIN_IDS
IGNORE_SERVER_IDS
IN
INDEX
INFILE
INNER
INOUT
INSENSITIVE
INSERT

Keyword
INT
INT1
INT2
INT3
INT4
INT8
INTEGER
INTERSECT
INTERVAL
INTO
IS
ITERATE
JOIN
KEY
KEYS
KILL
LEADING
LEAVE
LEFT
LIKE
LIMIT
LINEAR
LINES
LOAD
LOCALTIME
LOCALTIMESTAMP
LOCK
LONG
LONGBLOB
LONGTEXT
LOOP
LOW_PRIORITY
MASTER_HEARTBEAT_PERIOD
MASTER_SSL_VERIFY_SERVER_CERT
MATCH
MAXVALUE
MEDIUMBLOB
MEDIUMINT
MEDIUMTEXT

Keyword
MIDDLEINT
MINUTE_MICROSECOND
MINUTE_SECOND
MOD
MODIFIES
NATURAL
NOT
NO_WRITE_TO_BINLOG
NULL
NUMERIC
OFFSET (> 10.6)
ON
OPTIMIZE
OPTION
OPTIONALLY
OR
ORDER
OUT
OUTER
OUTFILE
OVER
PAGE_CHECKSUM
PARSE_VCOL_EXPR
PARTITION
PRECISION
PRIMARY
PROCEDURE
PURGE
RANGE
READ
READS
READ_WRITE
REAL
RECURSIVE
REF_SYSTEM_ID
REFERENCES
REGEXP
RELEASE
RENAME

Keyword
REPEAT
REPLACE
REQUIRE
RESIGNAL
RESTRICT
RETURN
RETURNING
REVOKE
RIGHT
RLIKE
ROW_NUMBER (> 10.7)
ROWS
SCHEMA
SCHEMAS
SECOND_MICROSECOND
SELECT
SENSITIVE
SEPARATOR
SET
SHOW
SIGNAL
SLOW
SMALLINT
SPATIAL
SPECIFIC
SQL
SQLEXCEPTION
SQLSTATE
SQLWARNING
SQL_BIG_RESULT
SQL_CALC_FOUND_ROWS
SQL_SMALL_RESULT
SSL
STARTING
STATS_AUTO_RECALC
STATS_PERSISTENT
STATS_SAMPLE_PAGES
STRAIGHT_JOIN
ST_COLLECT (> 11.8)

Keyword
TABLE
TERMINATED
THEN
TINYBLOB
TINYINT
TINYTEXT
TO
TO_DATE (> 12.2)
TRAILING
TRIGGER
TRUE
UNDO
UNION
UNIQUE
UNLOCK
UNSIGNED
UPDATE
USAGE
USE
USING
UTC_DATE
UTC_TIME
UTC_TIMESTAMP
VALUES
VARBINARY
VARCHAR
VARCHARACTER
VARYING
VECTOR (> 11.6)
WHEN
WHERE
WHILE
WINDOW
WITH
WRITE
XOR
YEAR_MONTH
ZEROFILL

Exceptions

Some keywords are exceptions for historical reasons, and are permitted as unquoted identifiers. These include:

Keyword
ACTION
BIT
DATE
ENUM
NO
TEXT
TIME
TIMESTAMP

Oracle Mode

In Oracle mode, there are a number of extra reserved words:

Keyword
BODY
ELSIF
GOTO
HISTORY
MINUS (> 10.6.0)
OTHERS
PACKAGE
PERIOD
RAISE
ROWNUM
ROWTYPE
SYSDATE
SYSTEM
SYSTEM_TIME
VERSIONING
WITHOUT

Contextual Keywords and Parser Limitations

The lists above indicate words that are always reserved and must be quoted. Some keywords, though, are reserved for specific contexts. In most SQL statements, they operate normally, but in some cases, they activate unique parser rules.

The Oracle mode term `SYSTEM` is a prime example. Although it is not a fully reserved word, the parser expects the VERSIONING keyword to follow (as part of the `SYSTEM VERSIONING` clause for system-versioned tables), which results in a syntax error in an `ALTER TABLE ... ADD` command.

Example of unexpected behavior

The following examples show how the keyword SYSTEM behaves as an identifier:

```
CREATE TABLE t1 (system INT);
```

This statement works because SYSTEM is a valid column name in this situation.

Invalid usage

```
ALTER TABLE t1 ADD system VARCHAR(64);
```

This statement fails due to a syntax error.

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MariaDB server version for the right syntax to use near 'VARCHAR(64)' at line 1
```

To avoid any conflict, always quote such keywords when using them as identifiers in statements.

```
ALTER TABLE t1 ADD `system` VARCHAR(64);
```

It is recommended that all identifiers listed as keywords (including those indicated as non-reserved or version-specific) be quoted for maximum compatibility and to prevent unexpected parser errors, especially in `ALTER` statements.

Function Names

If the `IGNORE_SPACE` SQL_MODE flag is set, function names become reserved words.

See Also

- Information Schema KEYWORDS Table

This page is licensed: CC BY-SA / Gnu FDL

Boolean Literals

In MariaDB, `FALSE` is a synonym of `0` and `TRUE` is a synonym of `1`. These constants are case insensitive, so `TRUE`, `True`, and `true` are equivalent.

These terms are not synonyms of `0` and `1` when used with the IS operator. So, for example, `10 IS TRUE` returns `1`, while `10 = TRUE` returns `0` (because 1 != 10).

The `IS` operator accepts a third constant exists: `UNKNOWN`. It is always a synonym of NULL.

`TRUE` and `FALSE` are reserved words, while `UNKNOWN` is not.

See Also

- BOOLEAN type

This page is licensed: CC BY-SA / Gnu FDL

Schema Qualifiers

> **Note:**
>
> - `mariadb_schema` and `oracle_schema` are not actual databases or schemas.
> - They do not appear in `SHOW DATABASES`.
> - These are prefix qualifiers that manage SQL-mode dependent behavior for function and data types.

Description

MariaDB offers schema qualifiers that allow you to define how data types and functions are managed when SQL modes result in operational differences.

These qualifiers are used as prefixes and are part of the SQL language structure. These are not system databases or tables.

The schema qualifiers listed below are supported:

- mariadb_schema
- oracle_schema

Both prefixes describe the same mechanism, however the behavior differs based on the desired compatibility.

Certain SQL modes, such as SQL_MODE=ORACLE, affect how MariaDB understands data type and functions. MariaDB translates several data type names and function behaviors according to Oracle Database standards when SQL_MODE=ORACLE is active. This translation may lead to ambiguity: depending on the SQL mode in use, a data type or function name written without a qualifier could be interpreted differently.

For example, in Oracle mode, the data type DATE is interpreted as DATETIME. This behavior can be explicitly overridden at the statement level using schema qualifiers, without modifying the SQL mode session.

Version Support

- `mariadb_schema` for data type qualification is supported since:
  - MariaDB 10.3.24
  - MariaDB 10.4.14
  - MariaDB 10.5.5
- `oracle_schema` is supported as part of the same schema qualifier mechanism.
- Functional qualification using schema qualifiers is supported since:
  - MariaDB 10.6.17 (see MDEV-27744)

Why Schema Qualifiers are Needed

Specific SQL modes, for example `SQL_MODE=ORACLE`, determine how MariaDB handles data types and functions.

This may result in ambiguity when:

- Switching between SQL modes
- Evaluating table definitions using `SHOW CREATE TABLE`
- Syncing data across servers

Regardless of the SQL mode that is currently in use, schema qualifiers provide explicit control over interpretation, ensuring reliable and consistent behavior.

mariadb_schema

The `mariadb_schema` qualifier forces MariaDB-native interpretation of the data type or function, regardless of the active SQL mode.

When the server sees the `mariadb_schema` qualifier, it disables sql_mode-specific data type translation and interprets the data type literally. For example, `mariadb_schema.DATE` is interpreted as the traditional MariaDB `DATE` data type, regardless of the current SQL mode.

Data Type Qualification

In Oracle mode, the DATE type is translated to DATETIME:

```
SET sql_mode=ORACLE;

CREATE TABLE t1 (
d DATE
);
```

To specifically use the MariaDB-native DATE type while in Oracle mode:

```
CREATE TABLE t1 (
  d mariadb_schema.DATE
);
```

Example

```
SET sql_mode=ORACLE;

CREATE TABLE t1 (
  d DATE,
  d_m mariadb_schema.DATE
);

DESCRIBE t1;
```

Output

```
+-------+----------+
| Field | Type     |
+-------+----------+
| d     | datetime |
| d_m   | date     |
+-------+----------+
```

oracle_schema

The `oracle_schema` qualifier enforces Oracle-compatible behavior of the data type or function, regardless of the active SQL mode. It supports Oracle semantics but acts similarly to `mariadb_schema`.

For example:

- In Oracle mode, the `DATE` data types behaves like `DATETIME`.
- Using `oracle_schema.DATE` ensures Oracle-compatible interpretation even when Oracle mode is not enabled.

Data Type Qualification

```
SET sql_mode=ORACLE;

CREATE TABLE t1 (
  d DATE,
  d_m mariadb_schema.DATE,
  d_o oracle_schema.DATE
);

DESCRIBE t1;
```

Output

```
+-------+----------+
| Field | Type     |
+-------+----------+
| d     | datetime |
| d_m   | date     |
| d_o   | datetime |
+-------+----------+
```

Using Both Qualifiers Together

The following example demonstrates how to use both qualifiers together in a single table definition to control data type behavior.

To explicitly use Oracle-compatible behavior outside Oracle mode:

```
SET sql_mode=DEFAULT;

CREATE TABLE t1 (
d oracle_schema.DATE
);
```

When running in Oracle mode, both qualifiers can be used to explicitly control behavior.

```
SET sql_mode=ORACLE;

CREATE TABLE t1 (
d DATE,                  # Oracle-compatible (stored as DATETIME)
d_m mariadb_schema.DATE, # MariaDB-native DATE
d_o oracle_schema.DATE   # Oracle-compatible DATE
);

DESCRIBE t1;
```

Output

```
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| d     | datetime | YES  |     | NULL    |       |
| d_m   | date     | YES  |     | NULL    |       |
| d_o   | datetime | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
```

Function Qualification

Starting with MariaDB 10.6.17, schema qualifiers can also be added to functions.

Regardless of the current SQL mode, this enables the explicit selection of MariaDB-native or Oracle-compatible behavior.

Example syntax:

```
# MariaDB-native function behavior in any SQL mode
mariadb_schema.function_name(...)

# Oracle-compatible function behavior in any SQL mode
oracle_schema.function_name(...)
```

Behavior

- `mariadb_schema.function_name(...)`: MariaDB-native behavior
- `oracle_schema.function_name(...)`: Oracle-compatible behavior

Note: Function behavior varies depending on the SQL mode and function.
Detailed function-level differences are not fully listed here and may differ by version.

SHOW CREATE TABLE

To prevent ambiguity in data type interpretation, MariaDB may display schema qualifiers when `SHOW CREATE TABLE` is executed.

```
SET sql_mode=DEFAULT;
CREATE OR REPLACE TABLE t1 (
  d DATE
);
SET SQL_mode=ORACLE;
SHOW CREATE TABLE t1;
+-------+-----------------------------------------------------------------+
| Table | Create Table                                                    |
+-------+-----------------------------------------------------------------+
| t1    | CREATE TABLE "t1" (                                             |
|       |   "d" mariadb_schema.DATE DEFAULT NULL                          |
|       |  )                                                              |
+-------+-----------------------------------------------------------------+
```

The prefix is displayed to indicate that the column uses MariaDB's native DATE type rather than the Oracle-compatible translated version.

The `mariadb_schema` prefix is only displayed when the data type is ambiguous. When the type is non-ambiguous or the SQL mode is set to default, it is not displayed.

This makes it easier to determine whether a column uses MariaDB-native or Oracle-compatible semantics.

History

When running with SQL_MODE=ORACLE, MariaDB server translates the `DATE` data type to `DATETIME`, for Oracle compatibility.

```
SET SQL_mode=ORACLE;
CREATE OR REPLACE TABLE t1 (
  d DATE
);
SHOW CREATE TABLE t1;
+-------+-----------------------------------------------------------------+
| Table | Create Table                                                    |
+-------+-----------------------------------------------------------------+
| t1    | CREATE TABLE "t1" (                                             |
|       |   "d" datetime DEFAULT NULL                        |
|       |  )                                                              |
+-------+-----------------------------------------------------------------+
```

Note: `DATE` was converted to `DATETIME`.

This translation can cause ambiguity. For example, if a table is created in default SQL mode and then evaluated in Oracle mode:

```
SET sql_mode=DEFAULT;
CREATE OR REPLACE TABLE t1 (
  d DATE
);
SET SQL_mode=ORACLE;
SHOW CREATE TABLE t1;
```

Before schema qualifiers were introduced, this would display:

```
CREATE TABLE "t1" (
  "d" DATE DEFAULT NULL
);
```

This created two problems:

- It was not clear whether DATE referred to the MariaDB DATE type or the Oracle-compatible DATETIME.
- It broke replication, causing a data type mismatch between the primary and the replica. (see MDEV-19632).

To address this problem, MariaDB included schema qualifiers to explicitly qualify data types:

```
SET sql_mode=DEFAULT;
CREATE OR REPLACE TABLE t1 (
  d DATE
);
SET SQL_mode=ORACLE;
SHOW CREATE TABLE t1;
+-------+-----------------------------------------------------------------+
| Table | Create Table                                                    |
+-------+-----------------------------------------------------------------+
| t1    | CREATE TABLE "t1" (                                             |
|       |   "d" mariadb_schema.DATE DEFAULT NULL                          |
|       |  )                                                              |
+-------+-----------------------------------------------------------------+
```

This enables consistent and unambiguous behavior across all SQL modes.

See Also

- SQL Mode
- SQL_MODE=ORACLE
- Data Types
- SHOW CREATE TABLE

This page is licensed: CC BY-SA / Gnu FDL

String Literals

Strings are sequences of characters and enclosed with quotes.

The syntax is:

```
[_charset_name]'string' [COLLATE collation_name]
```

For example:

```
'The MariaDB Foundation'
_utf8 'Foundation' COLLATE utf8_unicode_ci;
```

Strings can either be enclosed in single quotes or in double quotes (the same character must be used to both open and close the string).

The ANSI SQL-standard does not permit double quotes for enclosing strings, and although MariaDB does by default, if the MariaDB server has enabled the ANSI_QUOTES_SQL SQL_MODE, double quotes will be treated as being used for identifiers instead of strings.

Strings that are next to each other are automatically concatenated. The following are equivalent:

```
'The ' 'MariaDB ' 'Foundation'
```

```
'The MariaDB Foundation'
```

The `\` (backslash character) is used to escape characters (unless the SQL_MODE hasn't been set to NO_BACKSLASH_ESCAPES):

```
'MariaDB's new features'
```

That is not a valid string because of the single quote in the middle of the string, which is treated as if it closes the string, but is actually meant as part of the string, an apostrophe. The backslash character helps in situations like this:

```
'MariaDB\'s new features'
```

That is now a valid string, and if displayed, will appear without the backslash.

```
SELECT 'MariaDB\'s new features';
+------------------------+
| MariaDB's new features |
+------------------------+
| MariaDB's new features |
+------------------------+
```

Another way to escape the quoting character is repeating it twice:

```
SELECT 'I''m here', """Double""";
+----------+----------+
| I'm here | "Double" |
+----------+----------+
| I'm here | "Double" |
+----------+----------+
```

Escape Sequences

There are other escape sequences:

Escape sequence	Character
\0	ASCII NUL (0x00).
'	Single quote.
"	Double quote.
\b	Backspace.
\n	New line / linefeed.
\r	Carriage return.
\t	Tab.
\Z	ASCII 26 (Control+Z). See note following the table.
\	Backslash.
%	Percentage character. See note following the table.
_	Underscore character. See note following the table.

Escaping the `%` and `_` characters can be necessary when using the LIKE operator, which treats them as special characters.

The ASCII 26 character (`\Z`) needs to be escaped when included in a batch file which needs to be executed in Windows. The reason is that ASCII 26, in Windows, is the end of file (EOF).

Backslash (`\`), if not used as an escape character, must always be escaped. When followed by a character that is not in the above table, backslashes will simply be ignored.

This page is licensed: CC BY-SA / Gnu FDL

Table Value Constructors

Syntax

```
VALUES ( row_value[, row_value...]), (...)...
```

Description

In Unions, Views, and subqueries, a Table Value Constructor (TVC) allows you to inject arbitrary values into the result set. The given values must have the same number of columns as the result set, otherwise it returns Error 1222.

Examples

Using TVC's with UNION operations:

```
CREATE TABLE test.t1 (val1 INT, val2 INT);
INSERT INTO test.t1 VALUES(5, 8), (3, 4), (1, 2);

SELECT * FROM test.t1
UNION
VALUES (70, 90), (100, 110);

+------+------+
| val1 | val2 |
+------+------+
|    5 |    8 |
|    3 |    4 |
|    1 |    2 |
|   70 |   90 |
|  100 |  110 |
+------+------+
```

Using TVCs with a CREATE VIEW statement:

```
CREATE VIEW v1 AS VALUES (7, 9), (9, 10);

SELECT * FROM v1;
+---+----+
| 7 |  9 |
+---+----+
| 7 |  9 |
| 9 | 10 |
+---+----+
```

Using TVC with an ORDER BY clause:

```
SELECT * FROM test.t1
UNION
VALUES (10, 20), (30, 40), (50, 60), (70, 80)
ORDER BY val1 DESC;
```

Using TVC with LIMIT clause:

```
SELECT * FROM test.t1
UNION
VALUES (10, 20), (30, 40), (50, 60), (70, 80)
LIMIT 2 OFFSET 4;

+------+------+
| val1 | val2 |
+------+------+
|   30 |   40 |
|   50 |   60 |
+------+------+
```

This page is licensed: CC BY-SA / Gnu FDL

User-Defined Variables

User-defined variables are variables which can be created by the user and exist in the session. This means that no one can access user-defined variables that have been set by another user, and when the session is closed these variables expire. However, these variables can be shared between several queries and stored programs.

User-defined variables names must be preceded by a single *at* character (`@`). While it is safe to use a reserved word as a user-variable name, the only allowed characters are ASCII letters, digits, dollar sign (`$`), underscore (`_`) and dot (`.`). If other characters are used, the name can be quoted in one of the following ways:

- @`var_name`
- @'var_name'
- @"var_name"

These characters can be escaped as usual.

User-variables names are case insensitive.

User-defined variables cannot be declared. They can be read even if no value has been set yet; in that case, they are `NULL`. To set a value for a user-defined variable you can use:

- SET statement;
- := operator within a SQL statement;
- SELECT ... INTO.

Since user-defined variables type cannot be declared, the only way to force their type is using CAST() or CONVERT():

```
SET @str = CAST(123 AS CHAR(5));
```

If a variable has not been used yet, its value is `NULL`:

```
SELECT @x IS NULL;
+------------+
| @x IS NULL |
+------------+
|          1 |
+------------+
```

It is unsafe to read a user-defined variable and set its value in the same statement (unless the command is `SET`), because the order of these actions is undefined.

User-defined variables can be used in most MariaDB's statements and clauses which accept an SQL expression. However there are some exceptions, like the LIMIT clause.

They must be used to PREPARE a prepared statement:

```
@sql = 'DELETE FROM my_table WHERE c>1;';
PREPARE stmt FROM @sql;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

Another common use is to include a counter in a query:

```
SET @var = 0;
SELECT a, b, c, (@var:=@var+1) AS counter FROM my_table;
```

Viewing

User-defined variables can be viewed by either querying the USER_VARIABLES, or by running `SHOW USER_VARIABLES`.

Flushing User-Defined Variables

User-defined variables are reset and the Information Schema table emptied with the FLUSH USER_VARIABLES statement.

Examples

```
SET @v1 = 0;
SET @v2 = 'abc';
SET @v3 = CAST(123 AS CHAR(5));

SHOW USER_VARIABLES;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| v3            | 123   |
| v2            | abc   |
| v1            | 0     |
+---------------+-------+

SELECT * FROM information_schema.USER_VARIABLES ORDER BY VARIABLE_NAME;
+---------------+----------------+---------------+--------------------+
| VARIABLE_NAME | VARIABLE_VALUE | VARIABLE_TYPE | CHARACTER_SET_NAME |
+---------------+----------------+---------------+--------------------+
| v1            | 0              | INT           | latin1             |
| v2            | abc            | VARCHAR       | utf8               |
| v3            | 123            | VARCHAR       | utf8               |
+---------------+----------------+---------------+--------------------+

FLUSH USER_VARIABLES;

SELECT * FROM information_schema.USER_VARIABLES ORDER BY VARIABLE_NAME;
Empty set (0.000 sec)
```

See Also

- DECLARE VARIABLE
- Performance Schema user_variables_by_thread Table
- Information Schema USER_VARIABLES Table

This page is licensed: CC BY-SA / Gnu FDL

Joins, Subqueries, and Set

SQL (Structured Query Language) is a highly potent language used in the realm of databases and data analytics. It allows us to create, manipulate, and retrieve data stored in relational databases. While simple SQL commands can

handle straightforward tasks, one must delve into advanced aspects such as multiple joins, subqueries, and set operations to unravel the full potential of SQL.

Multiple Joins

Joins are fundamental to SQL, as they enable us to combine rows from two or more tables based on related columns. Several types of joins exist, including `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, and `FULL JOIN`.

In real-world scenarios, finding database schemas with multiple tables linked through relationships is common. In such cases, we often need to write SQL queries with multiple joins to fetch the required data.

For instance, consider a database for a bookstore with three tables: Books, Authors, and Publishers. If we want to get a list of books along with their author names and publishers, you would require multiple joins.

```
SELECT Books.title, Authors.name, Publishers.name
FROM Books
INNER JOIN Authors ON Books.author_id = Authors.id
INNER JOIN Publishers ON Books.publisher_id = Publishers.id;
```

In that query, we used two `INNER JOIN` operations to combine data from the three tables.

Subqueries

A subquery, also known as a nested query or inner query, is a query embedded within another SQL query. The result of the subquery can be used in the outer query. Subqueries can be used in `SELECT`, `INSERT`, `UPDATE`, and `DELETE` statements and also in conjunction with WHERE and HAVING clauses.

Subqueries can prove powerful in many scenarios. For example, suppose we want to find books that are above the average price in our bookstore database. We could achieve this using a subquery.

```
SELECT title, price
FROM Books
WHERE price > (SELECT AVG(price) FROM Books);
```

In this query, the subquery calculates the average book price, which the outer query then uses to filter books that are priced above average.

Set Operations

SQL also provides several set operations to combine rows from two or more tables, including `UNION`, `UNION ALL`, `INTERSECT`, and `EXCEPT`. These operations are instrumental when dealing with large databases where data is spread across multiple tables and we need to perform set-based operations.

For example, suppose we have two tables, *Books_2019* and *Books_2020*, listing the books sold in the respective years. If we want to find books sold in both years, we can use the `INTERSECT` operation.

```
SELECT title FROM Books_2019
INTERSECT
SELECT title FROM Books_2020;
```

In the above query, INTERSECT returns the common book titles sold in both 2019 and 2020.

Conclusion

Advanced SQL queries that involve multiple joins, subqueries, and set operations can be daunting at first glance. However, these incredibly powerful tools in our SQL toolkit enable us to craft complex data retrieval and manipulation commands. To truly excel in data analytics, mastering these aspects of SQL is not only beneficial but essential.

Remember, practice is the key to mastering these advanced SQL concepts, like any other technical skill. Write queries, solve problems, make mistakes, and learn from them. Over time, you'll find yourself not only comfortable but proficient in creating intricate SQL queries.

See Also

- Joining Tables with JOIN Clauses Guide
- More Advanced Joins
- JOIN Syntax
- Comma vs JOIN

This page is licensed: CC BY-SA / Gnu FDL

Geometry

See Geometry Hierarchy.

The geometry class hierarchy — the abstract Geometry base class and its instantiable point, line, and polygon subclasses.

See Geometry Types.

Create spatial columns for geometry types in MariaDB, supported on MyISAM, InnoDB, and ARCHIVE tables.

See GIS Features.

An overview of GIS support in MariaDB, based on the OpenGIS Simple Feature Access standard.

See GIS Resources.

External resources and further reading on GIS and spatial data in MariaDB.

See SPATIAL INDEX.

Create SPATIAL (R-tree) indexes on geometry columns in MyISAM, Aria, and InnoDB tables.

This page is licensed: CC BY-SA / Gnu FDL

Geometry Hierarchy

Description

Geometry is the base class. It is an abstract class. The instantiable subclasses of Geometry are restricted to zero-, one-, and two-dimensional geometric objects that exist in two-dimensional coordinate space. All instantiable geometry classes are defined so that valid instances of a geometry class are topologically closed (that is, all defined geometries include their boundary).

The base Geometry class has subclasses for Point, Curve, Surface, and GeometryCollection:

- Point represents zero-dimensional objects.
- Curve represents one-dimensional objects, and has a subclass LineString, with sub-subclasses Line and LinearRing.
- Surface is designed for two-dimensional objects and has a subclass Polygon.
- GeometryCollection has specialized zero-, one-, and two-dimensional collection classes named MultiPoint, MultiLineString, and MultiPolygon for modeling geometries corresponding to collections of Points, LineStrings, and Polygons, respectively. MultiCurve and MultiSurface are introduced as abstract superclasses that generalize the collection interfaces to handle Curves and Surfaces.

Geometry, Curve, Surface, MultiCurve, and MultiSurface are defined as non-instantiable classes. They define a common set of methods for their subclasses and are included for extensibility.

Point, LineString, Polygon, GeometryCollection, MultiPoint,MultiLineString, and MultiPolygon are instantiable classes.

This page is licensed: GPLv2, originally from fill_help_tables.sql

Geometry Types

Description

MariaDB provides a standard way of creating spatial columns for geometry types, for example, with CREATE TABLE or ALTER TABLE. Spatial columns are supported for MyISAM, InnoDB and ARCHIVE tables. See also SPATIAL INDEX.

The basic geometry type is `GEOMETRY`, but the type can be more specific. The following types are supported:

- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION
- GEOMETRY

Examples

Note: For clarity, only one type is listed per table in the examples below, but a table row can contain multiple types.

```
CREATE TABLE object (shapeA POLYGON, shapeB LINESTRING);
```

```
INSERT INTO geometry_example VALUES
  (Point(0, 0)),
  (ST_PolygonFromText('POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))')),
  (ST_LineFromText('LINESTRING(0 0, 1 1, 2 2)')),
  (ST_MultiLineStringFromText(
    'MULTILINESTRING((0 1, 0 2, 0 3),
    (1 0, 2 0, 3 0))')),
  (ST_MultiPointFromText('MULTIPOINT(0 0, 1 1, 0 1, 1 0)')),
  (ST_MultiPolygonFromText(
    'MULTIPOLYGON(((0 40, 0 20, 6 30, 12 20, 12 40, 0 40),
    (15 40, 15 20, 25 20, 30 25, 30 35, 25 40, 15 40)))'));
```

```
SELECT ST_AsWKT(g) FROM geometry_example;

+-----------------------------------------------------------------------------------------------+
| ST_AsWKT(g)                                                                                   |
+-----------------------------------------------------------------------------------------------+
| POINT(0 0)                                                                                    |
| POLYGON((0 0,0 1,1 1,1 0,0 0))                                                                |
| LINESTRING(0 0,1 1,2 2)                                                                       |
| MULTILINESTRING((0 1,0 2,0 3),(1 0,2 0,3 0))                                                  |
| MULTIPOINT(0 0,1 1,0 1,1 0)                                                                   |
| MULTIPOLYGON(((0 40,0 20,6 30,12 20,12 40,0 40),(15 40,15 20,25 20,30 25,30 35,25 40,15 40))) |
+-----------------------------------------------------------------------------------------------+
```

POINT

```
CREATE TABLE gis_point  (g POINT);
SHOW FIELDS FROM gis_point;
INSERT INTO gis_point VALUES
    (PointFromText('POINT(10 10)')),
    (PointFromText('POINT(20 10)')),
    (PointFromText('POINT(20 20)')),
    (PointFromWKB(AsWKB(PointFromText('POINT(10 20)'))));
```

LINESTRING

```
CREATE TABLE gis_line  (g LINESTRING);
SHOW FIELDS FROM gis_line;
INSERT INTO gis_line VALUES
    (LineFromText('LINESTRING(0 0,0 10,10 0)')),
    (LineStringFromText('LINESTRING(10 10,20 10,20 20,10 20,10 10)')),
    (LineStringFromWKB(AsWKB(LineString(Point(10, 10), Point(40, 10)))));
```

POLYGON

```
CREATE TABLE gis_polygon   (g POLYGON);
SHOW FIELDS FROM gis_polygon;
INSERT INTO gis_polygon VALUES
    (PolygonFromText('POLYGON((10 10,20 10,20 20,10 20,10 10))')),
    (PolyFromText('POLYGON((0 0,50 0,50 50,0 50,0 0), (10 10,20 10,20 20,10 20,10 10))')),
    (PolyFromWKB(AsWKB(Polygon(LineString(Point(0, 0), Point(30, 0), Point(30, 30), Point(0, 0)))));
```

MULTIPOINT

```
CREATE TABLE gis_multi_point (g MULTIPOINT);
SHOW FIELDS FROM gis_multi_point;
INSERT INTO gis_multi_point VALUES
    (MultiPointFromText('MULTIPOINT(0 0,10 10,10 20,20 20)')),
    (MPointFromText('MULTIPOINT(1 1,11 11,11 21,21 21)')),
    (MPointFromWKB(AsWKB(MultiPoint(Point(3, 6), Point(4, 10)))));
```

MULTILINESTRING

```
CREATE TABLE gis_multi_line (g MULTILINESTRING);
SHOW FIELDS FROM gis_multi_line;
INSERT INTO gis_multi_line VALUES
    (MultiLineStringFromText('MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48))')),
    (MLineFromText('MULTILINESTRING((10 48,10 21,10 0))')),
    (MLineFromWKB(AsWKB(MultiLineString(LineString(Point(1, 2), Point(3, 5)), LineString(Point(2, 5),
Point(5, 8), Point(21, 7)))));
```

MULTIPOLYGON

```
CREATE TABLE gis_multi_polygon  (g MULTIPOLYGON);
SHOW FIELDS FROM gis_multi_polygon;
INSERT INTO gis_multi_polygon VALUES
    (MultiPolygonFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),
((59 18,67 18,67 13,59 13,59 18)))')),
    (MPolyFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67
18,67 13,59 13,59 18)))')),
    (MPolyFromWKB(AsWKB(MultiPolygon(Polygon(LineString(Point(0, 3), Point(3, 3), Point(3, 0), Point(0,
3)))))));
```

GEOMETRYCOLLECTION

```
CREATE TABLE gis_geometrycollection  (g GEOMETRYCOLLECTION);
SHOW FIELDS FROM gis_geometrycollection;
INSERT INTO gis_geometrycollection VALUES
    (GeomCollFromText('GEOMETRYCOLLECTION(POINT(0 0), LINESTRING(0 0,10 10))')),
    (GeometryFromWKB(AsWKB(GeometryCollection(Point(44, 6), LineString(Point(3, 6), Point(7, 9))))),
    (GeomFromText('GeometryCollection()')),
    (GeomFromText('GeometryCollection EMPTY'));
```

GEOMETRY

```
CREATE TABLE gis_geometry (g GEOMETRY);
SHOW FIELDS FROM gis_geometry;
INSERT INTO gis_geometry SELECT * FROM gis_point;
INSERT INTO gis_geometry SELECT * FROM gis_line;
INSERT INTO gis_geometry SELECT * FROM gis_polygon;
INSERT INTO gis_geometry SELECT * FROM gis_multi_point;
INSERT INTO gis_geometry SELECT * FROM gis_multi_line;
INSERT INTO gis_geometry SELECT * FROM gis_multi_polygon;
INSERT INTO gis_geometry SELECT * FROM gis_geometrycollection;
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

GIS Features

GIS stands for Geographic Information System.

MySQL operates on spatial data based on the OpenGIS standards, particularly the OpenGIS SFS (Simple feature access, SQL option).

Initial support was based on version 05-134 of the standard. MariaDB implements a subset of the 'SQL with Geometry Types' environment proposed by the OGC. And the SQL environment was extended with a set of geometry types.

MariaDB supports spatial extensions to operate on spatial features. These features are available for Aria, MyISAM, InnoDB, NDB, and ARCHIVE tables.

For spatial columns, Aria and MyISAM supports both SPATIAL and non-SPATIAL indexes. Other storage engines support non-SPATIAL indexes.

We aim at meeting the OpenGIS requirements. One thing missed in previous MariaDB versions is that the functions which check spatial relations didn't consider the actual shape of an object, instead they operate only on their bounding rectangles. These legacy functions have been left as they are and new, properly-working functions are named with an '`ST_`' prefix, in accordance with the latest OpenGIS requirements. Also, operations over geometry features were added.

Spatial Operators

Spatial operators produce new geometries.

Name	Description
ST_UNION(A, B)	union of A and B
ST_INTERSECTION(A, B)	intersection of A and B
ST_SYMDIFFERENCE(A, B)	symdifference, notintersecting parts of A and B
ST_BUFFER(A, radius)	returns the shape of the area that lies in 'radius' distance from the shape A.

Predicates

Predicates return a boolean result of the relationship.

Name	Description
ST_INTERSECTS(A, B)	if A and B have an intersection
ST_CROSSES(A, B)	if A and B cross
ST_EQUALS(A, B)	if A and B are equal
ST_WITHIN(A, B)	if A lies within B
ST_CONTAINS(A,B)	if B lies within A
ST_DISJOINT(A,B)	if A and B have no intersection
ST_TOUCHES(A,B)	if A touches B

This page is licensed: CC BY-SA / Gnu FDL

GIS Resources

GIS stands for Geographic Information System.

Further reading:

- OGC Simple Feature Access - the Open Geospatial Consortium's OpenGIS Simple Features Specifications For SQL.
- Geo/Spatial Search with MySQL - a presentation by Alexander Rubin, from the MySQL Conference in 2006.

There are no differences between GIS in stable versions of MariaDB and GIS in MySQL. There are, however, some extensions and enhancements being worked on.

This page is licensed: CC BY-SA / Gnu FDL

SPATIAL INDEX

Description

On MyISAM, Aria and InnoDB tables, MariaDB can create spatial indexes (an R-tree index) using syntax similar to that for creating regular indexes, but extended with the `SPATIAL` keyword. Columns in spatial indexes must be declared `NOT NULL`.

Spatial indexes can be created when the table is created, or added after the fact:

```
CREATE TABLE geom (g GEOMETRY NOT NULL, SPATIAL INDEX(g));
```

```
ALTER TABLE geom ADD SPATIAL INDEX(g);
```

```
CREATE SPATIAL INDEX sp_index ON geom (g);
```

`SPATIAL INDEX` creates an `R-tree` index. For storage engines that support non-spatial indexing of spatial columns, the engine creates a `B-tree` index. A `B-tree` index on spatial values is useful for exact-value lookups, but not for range scans.

For more information on indexing spatial columns, see CREATE INDEX.

To drop spatial indexes, use ALTER TABLE or DROP INDEX:

```
ALTER TABLE geom DROP INDEX g;
```

```
DROP INDEX sp_index ON geom;
```

Data-at-Rest Encryption

If innodb_checksum_algorithm is set to `full_crc32` or `strict_full_crc32`, and if the table does not use ROW_FORMAT=COMPRESSED, InnoDB spatial indexes are encrypted if the table is encrypted.

See MDEV-12026 for more information.

This page is licensed: GPLv2, originally from fill_help_tables.sql

NoSQL

See Dynamic Columns.

Dynamic columns let each row store a different set of columns inside a blob, for schema-flexible (NoSQL-style) data.

See Dynamic Column API.

The client-side API for reading and writing dynamic-column blobs, as an alternative to the in-server dynamic-column functions.

See HANDLER.

Explore the HANDLER statement in MariaDB Server for direct table access. This section details how to bypass the SQL optimizer for low-level row operations, useful for specific NoSQL-like interactions.

See HandlerSocket.

Explore HandlerSocket for MariaDB Server. This plugin enables high-performance NoSQL-like access directly to InnoDB tables, bypassing SQL parsing for fast key-value operations.

See JSON Functions.

Complete JSON Functions reference: JSON_EXTRACT(), JSON_SET(), JSON_REPLACE(), JSON_SEARCH() syntax for path queries, document updates, and value retrieval.

See CONNECT.

The CONNECT storage engine.

This page is licensed: CC BY-SA / Gnu FDL

Dynamic Columns

Overview

Dynamic columns allow one to store different sets of columns for each row in a table. It works by storing a set of columns in a blob and having a small set of functions to manipulate it.

Dynamic columns should be used when it is not possible to use regular columns.

A typical use case is when one needs to store items that may have many different attributes (like size, color, weight, etc), and the set of possible attributes is very large and/or unknown in advance. In that case, attributes can be put into dynamic columns.

Dynamic Columns Basics

The table should have a blob column which will be used as storage for dynamic columns:

```
main
CREATE TABLE assets (
  item_name VARCHAR(32) PRIMARY KEY, -- A common attribute for all items
  dynamic_cols  BLOB  -- Dynamic columns will be stored here
);
```

Once created, you can access dynamic columns via dynamic column functions.

Insert a row with two dynamic columns: color=blue, size=XL:

```
INSERT INTO assets VALUES
  ('MariaDB T-shirt', COLUMN_CREATE('color', 'blue', 'size', 'XL'));
```

Insert another row with dynamic columns: color=black, price=500:

```
INSERT INTO assets VALUES
  ('Thinkpad Laptop', COLUMN_CREATE('color', 'black', 'price', 500));
```

Select the dynamic column ‘color’ for all items:

```
SELECT item_name, COLUMN_GET(dynamic_cols, 'color' as char)
  AS color FROM assets;
+-----------------+-------+
| item_name       | color |
+-----------------+-------+
| MariaDB T-shirt | blue  |
| Thinkpad Laptop | black |
+-----------------+-------+
```

It is possible to add and remove dynamic columns from a row:

```
-- Remove a column:
UPDATE assets SET dynamic_cols=COLUMN_DELETE(dynamic_cols, "price")
WHERE COLUMN_GET(dynamic_cols, 'color' AS CHAR)='black';

-- Add a column:
UPDATE assets SET dynamic_cols=COLUMN_ADD(dynamic_cols, 'warranty', '3 years')
WHERE item_name='Thinkpad Laptop';
```

You can also list all columns, or get them together with their values in JSON format:

```
SELECT item_name, column_list(dynamic_cols) FROM assets;
+-----------------+--------------------------+
| item_name       | column_list(dynamic_cols) |
+-----------------+--------------------------+
| MariaDB T-shirt | `size`,`color`            |
| Thinkpad Laptop | `color`,`warranty`        |
+-----------------+--------------------------+

SELECT item_name, COLUMN_JSON(dynamic_cols) FROM assets;
+-----------------+---------------------------------------+
| item_name       | COLUMN_JSON(dynamic_cols)             |
+-----------------+---------------------------------------+
| MariaDB T-shirt | {"size":"XL","color":"blue"}          |
| Thinkpad Laptop | {"color":"black","warranty":"3 years"} |
+-----------------+---------------------------------------+
```

Dynamic Columns Reference

This is a complete reference of dynamic columns in MariaDB.

> **Tip:**
>
> **Column can be referred to by name (column_name in the following code blocks). This is the preferred method.** If you need support for old (< 10.0) MariaDB versions, you have to refer to columns by number. In that case, replace *column_name* with *column_nr*. This method is not recommended.

Dynamic Columns Functions

COLUMN_CREATE

```
COLUMN_CREATE(column_name, value [as type], [column_name, value
  [as type]]...);
```

Returns a dynamic columns blob that stores the specified columns with values. The return value is suitable for

- storing in a table,
- further modification with other dynamic columns functions.

The `as type` part allows one to specify the value type. In most cases, this is redundant because MariaDB will be able to deduce the type of the value. Explicit type specification may be needed when the type of the value is not apparent. For example, a literal `'2012-12-01'` has a `CHAR` type by default, one will need to specify `'2012-12-01' AS DATE` to have it stored as a date. See the Datatypes section for further details. Note also MDEV-597.

Typical usage:

```
INSERT INTO tbl SET dyncol_blob=COLUMN_CREATE("column_name", "value");
```

COLUMN_ADD

```
COLUMN_ADD(dyncol_blob, column_name, value [as type],
  [column_name, value [as type]]...);
```

Adds or updates dynamic columns.

- `dyncol_blob` must be either a valid dynamic columns blob (for example, `COLUMN_CREATE` returns such blob), or an empty string.
- `column_name` specifies the name of the column to be added. If `dyncol_blob` already has a column with this name, it will be overwritten.
- `value` specifies the new value for the column. Passing a NULL value will cause the column to be deleted.
- `as type` is optional. See the datatypes section for a discussion about types.

The return value is a dynamic column blob after the modifications.

Typical usage:

```
UPDATE t1 SET dyncol_blob=COLUMN_ADD(dyncol_blob, "column_name", "value")
  WHERE id=1;
```

Note: `COLUMN_ADD()` is a regular function (just like CONCAT()), hence, in order to update the value in the table you have to use the `UPDATE ... SET dynamic_col=COLUMN_ADD(dynamic_col, ....)` pattern.

COLUMN_GET

```
COLUMN_GET(dyncol_blob, column_name as type);
```

Retrieves the value of a dynamic column by its name. If no column with the given name exists, `NULL` is returned.

`column_name as type` requires that one specify the datatype of the dynamic column they are reading.

This may seem counter-intuitive: Why would you need to specify which datatype they're retrieving? Can't the dynamic columns system figure the datatype from the data being stored?

The answer is: SQL is a statically typed language. The SQL interpreter needs to know the datatypes of all expressions before the query is run (for example, when one is using prepared statements and runs `"select COLUMN_GET(...)"`, the prepared statement API requires the server to inform the client about the datatype of the column being read before the query is executed and the server can see what datatype the column actually has).

See the Datatypes section for more information about datatypes.

COLUMN_DELETE

```
COLUMN_DELETE(dyncol_blob, column_name, column_name...);
```

Deletes a dynamic column with the specified name. Multiple names can be given.

The return value is a dynamic column blob after the modification.

COLUMN_EXISTS

```
COLUMN_EXISTS(dyncol_blob, column_name);
```

Checks if a column with name `column_name` exists in `dyncol_blob`. If yes, return `1`, otherwise return `0`.

COLUMN_LIST

```
COLUMN_LIST(dyncol_blob);
```

Returns a comma-separated list of column names. The names are quoted with backticks.

```
SELECT column_list(column_create('col1','val1','col2','val2'));
+-------------------------------------------------------+
| column_list(column_create('col1','val1','col2','val2')) |
+-------------------------------------------------------+
| `col1`,`col2`                                         |
+-------------------------------------------------------+
```

COLUMN_CHECK

```
COLUMN_CHECK(dyncol_blob);
```

Checks if `dyncol_blob` is a valid packed dynamic columns blob. A return value of `1` means the blob is valid, a return value of `0` means it is not.

Rationale:
Normally, you work with valid dynamic column blobs. Functions like `COLUMN_CREATE`, `COLUMN_ADD`, `COLUMN_DELETE` always return valid dynamic column blobs. However, if a dynamic column blob is accidentally truncated, or transcoded from one character set to another, it is corrupted. This function can be used to check if a value in a blob field is a valid dynamic column blob.

> **Warning:**
>
> It is possible that a truncation cuts a Dynamic Column "clearly" so that `COLUMN_CHECK` does not notice the corruption, but, in any case of truncation, a warning is issued during value storing.

COLUMN_JSON

```
COLUMN_JSON(dyncol_blob);
```

Returns a JSON representation of data in `dyncol_blob`:

```
SELECT item_name, COLUMN_JSON(dynamic_cols) FROM assets;
+-----------------+---------------------------------------+
| item_name       | COLUMN_JSON(dynamic_cols)             |
+-----------------+---------------------------------------+
| MariaDB T-shirt | {"size":"XL","color":"blue"}          |
| Thinkpad Laptop | {"color":"black","warranty":"3 years"} |
+-----------------+---------------------------------------+
```

Limitation: `COLUMN_JSON` decodes nested dynamic columns at a nesting level of not more than 10 levels deep. Dynamic columns that are nested deeper than 10 levels are shown as a `BINARY` string, without encoding.

Nesting Dynamic Columns

It is possible to use nested dynamic columns by putting one dynamic column blob inside another. The `COLUMN_JSON` function will display nested columns.

```
SET @tmp= column_create('parent_column',
  column_create('child_column', 12345));
Query OK, 0 rows affected (0.00 sec)

SELECT column_json(@tmp);
+----------------------------------------+
| column_json(@tmp)                      |
+----------------------------------------+
| {"parent_column":{"child_column":12345}} |
+----------------------------------------+

SELECT column_get(column_get(@tmp, 'parent_column' AS char),
  'child_column' AS int);
+-------------------------------------------------------------------------------+
| column_get(column_get(@tmp, 'parent_column' as char), 'child_column' as int) |
+-------------------------------------------------------------------------------+
|                                                                         12345 |
+-------------------------------------------------------------------------------+
```

If you are trying to get a nested dynamic column as a string, use `AS BINARY` as the last argument of `COLUMN_GET`. Otherwise, problems with character set conversion and illegal symbols are possible:

```
SELECT column_json( column_get(
  column_create('test1',
    column_create('key1','value1','key2','value2','key3','value3')),
  'test1' AS BINARY));
```

Datatypes

In SQL, one needs to define the type of each column in a table. Dynamic columns do not provide any way to declare a type in advance (“whenever there is a column ‘weight’, it should be integer” is not possible). However, each particular dynamic column value is stored together with its datatype.

The set of possible datatypes is mostly the same as that used by the CAST and CONVERT functions. However, note that there are currently some differences - see MDEV-597.

Type	Dynamic column internal type	Description
BINARY[(N)]	DYN_COL_STRING	(variable length string with binary charset)
CHAR[(N)]	DYN_COL_STRING	(variable length string with charset)

Type	Dynamic column internal type	Description
DATE	DYN_COL_DATE	(date - 3 bytes)
DATETIME[(D)]	DYN_COL_DATETIME	(date and time (with microseconds) - 9 bytes)
DECIMAL[(M[,D])]	DYN_COL_DECIMAL	(variable length binary decimal representation with MariaDB limitation)
DOUBLE[(M,D)]	DYN_COL_DOUBLE	(64 bit double-precision floating point)
INTEGER	DYN_COL_INT	(variable length, up to 64-bit signed integer)
SIGNED [INTEGER]	DYN_COL_INT	(variable length, up to 64-bit signed integer)
TIME[(D)]	DYN_COL_TIME	(time (with microseconds, may be negative) - 6 bytes)
UNSIGNED [INTEGER]	DYN_COL_UINT	(variable length, up to 64bit unsigned integer)

A Note About Lengths

If you're running queries without specifying a maximum length (i.e. using #as CHAR#, not `as CHAR(n)`), MariaDB reports the maximum length of the result set column to be `53,6870,911` (bytes or characters?). This may cause excessive memory usage in some client libraries, because they try to pre-allocate a buffer of maximum result set width. If you suspect you're hitting this problem, use `CHAR(n)` whenever you're using `COLUMN_GET` in the select list.

```
SELECT COLUMN_GET(blob, 'colname' as CHAR) ...
```

Client-side API

It is also possible to create or parse dynamic columns blobs on the client side. `libmysql` client library now includes an API for writing/reading dynamic column blobs. See dynamic-columns-api for details.

Limitations

Description	Limit
Max number of columns	65535
Maximum total length of packed dynamic column	max_allowed_packet (1G)

See Also

- Dynamic Column API

This page is licensed: CC BY-SA / Gnu FDL

Dynamic Column API

This page describes the client-side API for reading and writing Dynamic Columns blobs.

Normally, you should use Dynamic column functions which are run inside the MariaDB server and allow one to access Dynamic Columns content without any client-side libraries.

If you need to read/write dynamic column blobs **on the client** for some reason, this API enables that.

Where to get it

The API is a part of `libmysql` C client library. In order to use it, you need to include this header file and link against `libmysql`:

```
#include <mysql/ma_dyncol.h>
```

Data structures

DYNAMIC_COLUMN

`DYNAMIC_COLUMN` represents a packed dynamic column blob. It is essentially a string-with-length and is defined as follows:

```
/* A generic-purpose arbitrary-length string defined in MySQL Client API */
typedef struct st_dynamic_string
{
  char *str;
  size_t length,max_length,alloc_increment;
} DYNAMIC_STRING;

...

typedef DYNAMIC_STRING DYNAMIC_COLUMN;
```

DYNAMIC_COLUMN_VALUE

Dynamic columns blobs store {name, value} pairs. The `DYNAMIC_COLUMN_VALUE` structure is used to represent the value in accessible form.

```
struct st_dynamic_column_value
{
  DYNAMIC_COLUMN_TYPE type;
  union
  {
    long long long_value;
    unsigned long long ulong_value;
    double double_value;
    struct {
      MYSQL_LEX_STRING value;
      CHARSET_INFO *charset;
    } string;
    struct {
      decimal_digit_t buffer[DECIMAL_BUFF_LENGTH];
      decimal_t value;
    } decimal;
    MYSQL_TIME time_value;
  } x;
};
typedef struct st_dynamic_column_value DYNAMIC_COLUMN_VALUE;
```

Every value has a type, which is determined by the `type` member.

Type	Structure field
DYN_COL_NULL	-
DYN_COL_INT	value.x.long_value
DYN_COL_UINT	value.x.ulong_value
DYN_COL_DOUBLE	value.x.double_value
DYN_COL_STRING	value.x.string.value, value.x.string.charset
DYN_COL_DECIMAL	value.x.decimal.value
DYN_COL_DATETIME	value.x.time_value
DYN_COL_DATE	value.x.time_value
DYN_COL_TIME	value.x.time_value
DYN_COL_DYNCOL	value.x.string.value

Notes

- Values with type `DYN_COL_NULL` do not ever occur in dynamic columns blobs.
- Type `DYN_COL_DYNCOL` means that the value is a packed dynamic blob. This is how nested dynamic columns are done.
- Before storing a value to `value.x.decimal.value`, you must call `mariadb_dyncol_prepare_decimal()` to initialize the space for storage.

enum_dyncol_func_result

`enum enum_dyncol_func_result` is used as return value.

Value	Name	Comments
0	ER_DYNCOL_OK	OK
0	ER_DYNCOL_NO	(the same as ER_DYNCOL_OK but for functions which return a YES/NO)
1	ER_DYNCOL_YES	`YES` response or success
2	ER_DYNCOL_TRUNCATED	Operation succeeded but the data was truncated
-1	ER_DYNCOL_FORMAT	Wrong format of the encoded string
-2	ER_DYNCOL_LIMIT	A limit of implementation reached
-3	ER_DYNCOL_RESOURCE	Out of resources
-4	ER_DYNCOL_DATA	Incorrect input data
-5	ER_DYNCOL_UNKNOWN_CHARSET	Unknown character set

Result codes that are less than zero represent error conditions.

Function reference

Functions come in pairs:

- `xxx_num()` operates on the old (pre-MariaDB-10.0.1) dynamic column blob format, where columns were identified by numbers.
- `xxx_named()` can operate on both old or new data format. If it modifies the blob, it converts it to the new data format.

You should use `xxx_named()` functions, unless you need to keep the data compatible with MariaDB versions before 10.0.1.

mariadb_dyncol_init

First, define mariadb_dyncol_init(A) memset((A), 0, sizeof(*(A))). It is the correct initialization for an empty packed dynamic blob.

mariadb_dyncol_free

Copy where str is IN. Packed dynamic blob which memory should be freed.

```
void mariadb_dyncol_free(DYNAMIC_COLUMN *str);
```

mariadb_dyncol_create_many

Create a packed dynamic blob from arrays of values and names.

```
enum enum_dyncol_func_result
mariadb_dyncol_create_many(DYNAMIC_COLUMN *str,
                           uint column_count,
                           uint *column_numbers,
                           DYNAMIC_COLUMN_VALUE *values,
                           my_bool new_string);
enum enum_dyncol_func_result
mariadb_dyncol_create_many_named(DYNAMIC_COLUMN *str,
                                 uint column_count,
                                 MYSQL_LEX_STRING *column_keys,
                                 DYNAMIC_COLUMN_VALUE *values,
                                 my_bool new_string);
```

Here are the names and values:

Name	Value	Comments
str	OUT	Packed dynamic blob will be put here
column_count	IN	Number of columns
column_numbers	IN	Column numbers array (old format)
column_keys	IN	Column names array (new format)
values	IN	Column values array
new_string	IN	If TRUE then the str will be reinitialized (not freed) before usage

mariadb_dyncol_update_many

Add or update columns in a dynamic columns blob. To delete a column, update its value to a “non-value” of type `DYN_COL_NULL`:

```
enum enum_dyncol_func_result
mariadb_dyncol_update_many(DYNAMIC_COLUMN *str,
                           uint column_count,
                           uint *column_numbers,
                           DYNAMIC_COLUMN_VALUE *values);
enum enum_dyncol_func_result
mariadb_dyncol_update_many_named(DYNAMIC_COLUMN *str,
                                 uint column_count,
                                 MYSQL_LEX_STRING *column_keys,
                                 DYNAMIC_COLUMN_VALUE *values);
```

Name	Value	Comments
str	IN/OUT	Dynamic columns blob to be modified
column_count	IN	Number of columns in following arrays
column_numbers	IN	Column numbers array (old format)
column_keys	IN	Column names array (new format)
values	IN	Column values array

mariadb_dyncol_exists

Check if column with given names exist in the blob:

```
enum enum_dyncol_func_result
mariadb_dyncol_exists(DYNAMIC_COLUMN *str, uint column_number);
enum enum_dyncol_func_result
mariadb_dyncol_exists_named(DYNAMIC_COLUMN *str, MYSQL_LEX_STRING *column_key);
```

Name	Value	Comments
str	IN	Packed dynamic columns string
column_number	IN	Column number (old format)
column_key	IN	Column name (new format)

The function returns `YES`, or `NO` or Error code.

mariadb_dyncol_column_count

Get the number of columns in a dynamic column blob:

```
enum enum_dyncol_func_result
mariadb_dyncol_column_count(DYNAMIC_COLUMN *str, uint *column_count);
```

Name	Value	Comments
str	IN	Packed dynamic columns string
column_count	OUT	Number of not NULL columns in the dynamic columns string

mariadb_dyncol_list

List columns in a dynamic column blob:

```
enum enum_dyncol_func_result
mariadb_dyncol_list(DYNAMIC_COLUMN *str, uint *column_count, uint **column_numbers);
enum enum_dyncol_func_result
mariadb_dyncol_list_named(DYNAMIC_COLUMN *str, uint *column_count,
                          MYSQL_LEX_STRING **column_keys);
```

str	IN	Packed dynamic columns string
column_count	OUT	Number of columns in following arrays
column_numbers	OUT	Column numbers array (old format). Caller should free this array
column_keys	OUT	Column names array (new format). Caller should free this array

mariadb_dyncol_get

Get a value of one column:

```
enum enum_dyncol_func_result
mariadb_dyncol_get(DYNAMIC_COLUMN *org, uint column_number,
                   DYNAMIC_COLUMN_VALUE *value);
enum enum_dyncol_func_result
mariadb_dyncol_get_named(DYNAMIC_COLUMN *str, MYSQL_LEX_STRING *column_key,
                         DYNAMIC_COLUMN_VALUE *value);
```

Name	Value	Comments
str	IN	Packed dynamic columns string
column_number	IN	Column numbers array (old format)
column_key	IN	Column names array (new format)
value	OUT	Value of the column

If the column is not found, `NULL` is returned as the value of the column.

mariadb_dyncol_unpack

Get the value of all columns:

```
enum enum_dyncol_func_result
mariadb_dyncol_unpack(DYNAMIC_COLUMN *str,
                      uint *column_count,
                      MYSQL_LEX_STRING **column_keys,
                      DYNAMIC_COLUMN_VALUE **values);
```

Name	Value	Comments
str	IN	Packed dynamic columns string to unpack
column_count	OUT	Number of columns in following arrays
column_keys	OUT	Column names array (should be free by caller)
values	OUT	Values of the columns array (should be free by caller)

mariadb_dyncol_has_names

Check whether the dynamic columns blob uses the new data format (the one where columns are identified by names):

```
my_bool mariadb_dyncol_has_names(DYNAMIC_COLUMN *str);
```

Name	Value	Comments
str	IN	Packed dynamic columns string

mariadb_dyncol_check

Check whether the dynamic column blob has the correct data format:

```
enum enum_dyncol_func_result
mariadb_dyncol_check(DYNAMIC_COLUMN *str);
```

Name	Value	Comments
str	IN	Packed dynamic columns string

mariadb_dyncol_json

Get contents of a dynamic columns blob in a JSON form:

```
enum enum_dyncol_func_result
mariadb_dyncol_json(DYNAMIC_COLUMN *str, DYNAMIC_STRING *json);
```

Name	Value	Comments
str	IN	Packed dynamic columns string
json	OUT	JSON representation

mariadb_dyncol_val_TYPE

Get the dynamic column value as one of the base types:

```
enum enum_dyncol_func_result
mariadb_dyncol_val_str(DYNAMIC_STRING *str, DYNAMIC_COLUMN_VALUE *val,
                       CHARSET_INFO *cs, my_bool quote);
enum enum_dyncol_func_result
mariadb_dyncol_val_long(longlong *ll, DYNAMIC_COLUMN_VALUE *val);
enum enum_dyncol_func_result
mariadb_dyncol_val_double(double *dbl, DYNAMIC_COLUMN_VALUE *val);
```

Name	Value	Comments
str or ll or dbl	OUT	value of the column
val	IN	Value

mariadb_dyncol_prepare_decimal

Initialize `DYNAMIC_COLUMN_VALUE` before setting the value of `value.x.decimal.value`:

```
void mariadb_dyncol_prepare_decimal(DYNAMIC_COLUMN_VALUE *value);
```

Name	Value	Comments
value	OUT	Value of the column

This function links `value.x.decimal.value` to `value.x.decimal.buffer`.

mariadb_dyncol_value_init

Initialize a `DYNAMIC_COLUMN_VALUE` structure to a safe default:

```
#define mariadb_dyncol_value_init(V) (V)->type= DYN_COL_NULL
```

mariadb_dyncol_column_cmp_named

Compare two column names (column names are compared with memcmp()):

```
int mariadb_dyncol_column_cmp_named(const MYSQL_LEX_STRING *s1,
                                    const MYSQL_LEX_STRING *s2);
```

This page is licensed: CC BY-SA / Gnu FDL

HANDLER

See HANDLER.

Access storage engine interfaces directly for key lookups and key or table scans with the HANDLER statement's OPEN, READ, and CLOSE commands.

See HANDLER for MEMORY Tables.

Use HANDLER commands efficiently with MEMORY/HEAP tables, including creating BTREE keys for range scans and the limitations of HASH keys, BTREE keys, and table scans.

This page is licensed: CC BY-SA / Gnu FDL

HANDLER

Syntax

```
HANDLER tbl_name OPEN [ [AS] alias]
HANDLER tbl_name READ index_name { = | >= | <= | < } (value1,value2,...)
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ index_name { FIRST | NEXT | PREV | LAST }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ { FIRST | NEXT }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name CLOSE
```

Description

The `HANDLER` statement provides direct access to table storage engine interfaces for key lookups and key or table scans. It is available for at least Aria, Memory, MyISAM and InnoDB tables (and should work with most 'normal' storage engines, but not with system tables, MERGE or views).

`HANDLER ... OPEN` opens a table, allowing it to be accessible to subsequent `HANDLER ... READ` statements. The table can either be opened using an alias, or a table name. If opened with an alias, references to this table by further HANDLER statements must use this alias, and not the table name. If opened with a table name qualified by database name, further references to this table must use the unqualified table name. For example, if a table is opened with `db1.t1`, further references must use `t1`.

The table object is only closed when `HANDLER ... CLOSE` is called by the session, or the session closes, and is not shared by other sessions.

Prepared statements work with `HANDLER READ`, which gives a much higher performance (50% speedup) as there is no parsing and all data is transformed in binary (without conversions to text, as with the normal protocol).

Key Lookup

A key lookup is started with:

```
HANDLER tbl_name READ index_name { = | >= | <= | < }  (value,value) [LIMIT...]
```

The values stands for the value of each of the key columns. For most key types, except for `HASH` keys in `MEMORY` storage engine, you can use a prefix subset of its columns.

If you are using LIMIT, then in case of >= or > then there is an implicit NEXT implied, while if you are using <= or < then there is an implicit PREV implied.

After the initial read, you can use the following to scan rows in key order:

```
HANDLER tbl_name READ index_name NEXT [ LIMIT ... ]
or
HANDLER tbl_name READ index_name PREV [ LIMIT ... ]
```

> **Warning:**
>
> Note that the row order is not defined for keys with duplicated values, and will vary from engine to engine.

Key Scans

You can scan a table in key order by doing this:

```
HANDLER tbl_name READ index_name FIRST [ LIMIT ... ]
HANDLER tbl_name READ index_name NEXT  [ LIMIT ... ]
```

Alternatively, if the handler supports backwards key scans (which most do), you can use this:

```
HANDLER tbl_name READ index_name LAST [ LIMIT ... ]
HANDLER tbl_name READ index_name PREV [ LIMIT ... ]
```

Table Scans

You can scan a table in row order by doing this:

```
HANDLER tbl_name READ FIRST [ LIMIT ... ]
HANDLER tbl_name READ NEXT  [ LIMIT ... ]
```

Limitations

As this is a direct interface to the storage engine, some limitations may apply for what you can do and what happens if the table changes. Here are some of the common limitations.

Finding 'Old Rows'

`HANDLER READ` is not transaction-safe, consistent or atomic. It's okay for the storage engine to return rows that existed when you started the scan, but that were later deleted. This can happen as the storage engine may cache rows as part of the scan from a previous read.

You may also find rows committed since the scan originally started.

Invisible Columns

`HANDLER ... READ` also reads the data of invisible columns.

System-Versioned Tables

`HANDLER ... READ` reads everything from system-versioned tables, and so includes `row_start` and `row_end` fields, as well as all rows that have since been deleted or changed, including when history partitions are used.

Other Limitations

- If you do an ALTER TABLE, all your `HANDLER`s for that table are automatically closed.
- If you do an `ALTER TABLE` for a table that is used by some other connection with `HANDLER`, the `ALTER TABLE` query waits for the `HANDLER` to be closed.
- For `HASH` keys, you must use all key parts when searching for a row.
- For `HASH` keys, you can't do a key scan of all values. You can only find all rows with the same key value.
- While each `HANDLER READ` command is atomic, if you do a scan in many steps, some engines may give you error 1020 if the table changed between the commands. Please refer to the specific engine handler page if this happens.

Error Codes

- Error 1031 (`ER_ILLEGAL_HA`) Table storage engine for 't1' doesn't have this option
  - If you get this for `HANDLER OPEN` it means the storage engine doesn't support `HANDLER` calls.
  - If you get this for `HANDLER READ`, it means you are trying to use an incomplete `HASH` key.
- Error 1020 (`ER_CHECKREAD`) Record has changed since last read in table '…'
  - This means that the table changed between two reads, and the handler can't handle this case for the given scan.

Examples

```
CREATE TABLE t1 (f1 INT);

INSERT INTO t1 VALUES (1),(2),(3);

HANDLER t1 OPEN;

HANDLER t1 READ NEXT;
+------+
| f1   |
+------+
|    1 |
+------+

HANDLER t1 READ NEXT;
+------+
| f1   |
+------+
|    2 |
+------+
```

In the previous example, the `HANDLER` was opened with the `t1` table name. Since `HANDLER`s use unqualified table names, trying to access another table with this same name, even though it's in another database, will result in ambiguity. An alias needs to be used to avoid the ambiguity, resulting in Error 1066: Not unique table/alias:

```
CREATE DATABASE db_new;

CREATE TABLE db_new.t1 (id INT);

INSERT INTO db_new.t1 VALUES (4),(5),(6);

HANDLER db_new.t1 OPEN;
ERROR 1066 (42000): Not unique table/alias: 't1'

HANDLER db_new.t1 OPEN AS db_new_t1;

HANDLER db_new_t1 READ NEXT LIMIT 3;
+------+
| id   |
+------+
|    4 |
|    5 |
|    6 |
+------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

HANDLER for MEMORY Tables

This article explains how to use HANDLER commands efficiently with MEMORY/HEAP tables.

If you want to scan a table for different key values, not just search for exact key values, you should create your keys with `USING BTREE`:

```
CREATE TABLE t1 (a INT, b INT, KEY(a), KEY b USING BTREE (b)) ENGINE=memory;
```

In the above table, `a` is a HASH key that only supports exact matches (=) while `b` is a BTREE key that you can use to scan the table in key order, starting from start or from a given key value.

The limitations for `HANDLER READ` with `MEMORY|HEAP` tables are:

Limitations for HASH keys

- You must use all key parts when searching for a row.
- You can't do a key scan of all values. You can only find all rows with the same key value.
- `READ NEXT` gives an error 1031 if the tables changed since last read.

Limitations for BTREE keys

- `READ NEXT` gives an error 1031 if the tables changed since last read. This limitation can be lifted in the future.

Limitations for table scans

- `READ NEXT` gives an error 1031 if the table was truncated since last `READ` call.

See also

See also the limitations listed in HANDLER commands.

This page is licensed: CC BY-SA / Gnu FDL

HandlerSocket

See HandlerSocket Installation.

Install and configure the HandlerSocket plugin with INSTALL PLUGIN and the required my.cnf address and port options, then restart MariaDB.

See HandlerSocket Client Libraries.

List the HandlerSocket client libraries available for C++, Perl, PHP, Java, Python, Ruby, JavaScript, Scala, and Haskell applications.

See HandlerSocket Configuration Options.

Describe the HandlerSocket plugin configuration options for the my.cnf [mysqld] section, including addresses, ports, thread counts, buffers, and timeouts.

See HandlerSocket External Resources.

Collect external resources and documentation about the HandlerSocket plugin, including its home repository, background articles, and presentations.

This page is licensed: CC BY-SA / Gnu FDL

HandlerSocket Installation

After MariaDB is installed, use the INSTALL PLUGIN command (as the root user) to install the HandlerSocket plugin. This command only needs to be run once, like so:

```
INSTALL PLUGIN handlersocket SONAME 'handlersocket.so';
```

After installing the plugin, SHOW PROCESSLIST shows you first need to configure some settings. All HandlerSocket configuration options are placed in the `[mysqld]` section of your `my.cnf` file.

At least the handlersocket_address, handlersocket_port and handlersocket_port_wr options need to be set:

```
handlersocket_address="127.0.0.1"
handlersocket_port="9998"
handlersocket_port_wr="9999"
```

After updating the configuration options, restart MariaDB.

On the client side, to make use of the plugin you will need to install the appropriate client library (that is, libhsclient for C++ applications and perl-Net-HandlerSocket for perl applications).

This page is licensed: CC BY-SA / Gnu FDL

HandlerSocket Client Libraries

In order to make use of the HandlerSocket plugin in your applications, you will need to use the appropriate client library. The following client libraries are available:

- C++
  - libhsclient (included with the HandlerSocket plugin source)
- Perl
  - perl-Net-HandlerSocket (included with the HandlerSocket plugin source)
- PHP
  - Net_HandlerSocket
  - HSPHP
  - php-ext-handlersocketi
- Java
  - hs4j
  - handlersocketforjava
- Python
  - python-handler-socket
  - pyhandlersocket
- Ruby
  - ruby-handlersocket
  - handlersocket
- JavaScript
  - node-handlersocket
- Scala
  - hs2client
- Haskell
  - HandlerSocket-Haskell-Client

This page is licensed: CC BY-SA / Gnu FDL

HandlerSocket Configuration Options

The HandlerSocket plugin has the following options.

See also the Full list of MariaDB options, system and status variables.

Add the options to the `[mysqld]` section of your `my.cnf` file.

`handlersocket_accept_balance`

- Description: When set to a value other than zero (‘`0`’), `handlersocket` will try to balance accepted connections among threads. Default is `0` , but if you use persistent connections (for example if you use client-side connection pooling), a non-zero value is recommended.
- Command line: `--handlersocket-accept-balance="value"`
- Scope: Global

- Dynamic: No
- Type: number
- Range: `0` to `10000`
- Default Value: `0`

`handlersocket_address`

- Description: Specify the IP address to bind to.
- Command line: `--handlersocket-address="value"`
- Scope: Global
- Dynamic: No
- Type: IP Address
- Default Value: Empty, previously `0.0.0.0`

`handlersocket_backlog`

- Description: Specify the listen backlog length.
- Command line: `--handlersocket-backlog="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Range: `5` to `1000000`
- Default Value: `32768`

`handlersocket_epoll`

- Description: Specify whether to use epoll for I/O multiplexing.
- Command line: `--handlersocket-epoll="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Valid values:
  - Min: `0`
  - Max: `1`
- Default Value: `1`

`handlersocket_plain_secret`

- Description: When set, enables plain-text authentication for the listener for read requests, with the value of the option specifying the secret authentication key.
- Command line: `--handlersocket-plain-secret="value"`
- Dynamic: No
- Type: string
- Default Value: Empty

`handlersocket_plain_secret_wr`

- Description: When set, enables plain-text authentication for the listener for write requests, with the value of the option specifying the secret authentication key.
- Command line: `--handlersocket-plain-secret-wr="value"`

- Dynamic: No
- Type: string
- Default Value: Empty

`handlersocket_port`

- Description: Specify the port to bind to for reads. An empty value disables the listener.
- Command line: `--handlersocket-port="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Default Value: Empty, previously `9998`

`handlersocket_port_wr`

- Description: Specify the port to bind to for writes. An empty value disables the listener.
- Command line: `--handlersocket-port-wr="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Default Value: Empty, previously `9999`

`handlersocket_rcvbuf`

- Description: Specify the maximum socket receive buffer (in bytes). If ‘0’ then the system default is used.
- Command line: `--handlersocket-rcvbuf="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Range: `0` to `1677216`
- Default Value: `0`

`handlersocket_readsize`

- Description: Specify the minimum length of the request buffer. Larger values consume available memory but can make handlersocket faster for large requests.
- Command line: `--handlersocket-readsize="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Range: `0` to `1677216`
- Default Value: `0` (possibly `4096`)

`handlersocket_sndbuf`

- Description: Specify the maximum socket send buffer (in bytes). If ‘0’ then the system default is used.
- Command line: `--handlersocket-sndbuf="value"`
- Scope: Global
- Dynamic: No
- Type: number

- Range: `0` to `1677216`
- Default Value: `0`

`handlersocket_threads`

- Description: Specify the number of worker threads for reads.
  Recommended value = number of CPU cores * 2.
- Command line: `--handlersocket-threads="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Range: `1` to `3000`
- Default Value: `16`

`handlersocket_threads_wr`

- Description: Specify the number of worker threads for writes. Recommended value = 1.
- Command line: `--handlersocket-threads-wr="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Range: `1` to `3000`
- Default Value: `1`

`handlersocket_timeout`

- Description: Specify the socket timeout in seconds.
- Command line: `--handlersocket-timeout="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Range: `30` to `3600`
- Default Value: `300`

`handlersocket_verbose`

- Description: Specify the logging verbosity.
- Command line: `--handlersocket-verbose="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Valid values:
  - Min: `0`
  - Max: `10000`
- Default Value: `10`

`handlersocket_wrlock_timeout`

- Description: The write lock timeout in seconds. When acting on write requests, handlersocket locks an advisory lock named ‘handlersocket_wr’ and this option sets the timeout for it.

- Command line: `--handlersocket-wrlock-timeout="value"`
- Scope: Global
- Dynamic: No
- Type: number
- Range: `0` to `3600`

This page is licensed: CC BY-SA / Gnu FDL

HandlerSocket External Resources

Some resources and documentation about HandlerSocket.

- The home of HandlerSocket is here.
- The story of handlersocket can be found here.
- Comparison of HANDLER and HandlerSocket can be found here.
- HandlerSocket plugin for MySQL presentation by Akira Higuchi of DeNA - June 29 2010 - DeNA Technology Seminar
- HandlerSocket plugin for MySQL presentation by Akira Higuchi of DeNA - June 29 2011 - in Japanese

This page is licensed: CC BY-SA / Gnu FDL

Operators

See Operator Precedence.

Understand operator precedence in MariaDB Server SQL. This section details the order in which operators are evaluated within expressions, crucial for writing accurate and predictable queries.

See Arithmetic Operators.

Learn about arithmetic operators in MariaDB Server SQL. This section details how to perform mathematical calculations like addition, subtraction, multiplication, and division within your queries.

See Assignment Operators.

Learn about assignment operators in MariaDB Server SQL. This section details how to assign values to variables and columns, essential for data manipulation and programmatic logic.

See Comparison Operators.

Learn about comparison operators in MariaDB Server SQL. This section details operators like =, >, <, and LIKE used to compare values in conditions, essential for filtering and joining data.

See Logical Operators.

Learn about logical operators in MariaDB Server SQL. This section details operators like AND, OR, and NOT used to combine or negate conditions, essential for complex filtering and data selection.

This page is licensed: CC BY-SA / Gnu FDL

Operator Precedence

The precedence is the order in which the SQL operators are evaluated.

The following list shows the SQL operator precedence. **Operators that appear first in the list have a higher precedence.** Operators which are listed together have the same precedence.

- INTERVAL
- BINARY, COLLATE

- !
- - (unary minus), bitwise not (unary bit inversion)
- `||` (string concatenation)
- ^ (bitwise XOR)
- *, /, DIV, %, MOD (multiplication, division, modulo)
- -, + (subtraction, addition)
- <<, >>
- & (bitwise AND)
- | (bitwise OR)
- LIKE, REGEXP, IN
- BETWEEN
- = (comparison), <=>, >=, >, <=, <, <>, !=, IS
- NOT
- &&, AND
- XOR
- || (logical or), OR
- = (assignment), :=

> **Warning:**
>
> Functions precedence is always higher than operators precedence.

If the `HIGH_NOT_PRECEDENCE` SQL_MODE is set, `NOT` has the same precedence as `!`.

The `||` operator's precedence, as well as its meaning, depends on the `PIPES_AS_CONCAT` SQL_MODE flag: if it is on, `||` can be used to concatenate strings (like the CONCAT() function) and has a higher precedence.

The `=` operator's precedence depends on the context - it is higher when `=` is used as a comparison operator.

Parentheses can be used to modify the operators precedence in an expression.

Short-Circuit Evaluation

The `AND`, `OR`, `&&` and `||` operators support short-circuit evaluation. This means that, in some cases, the expression on the right of those operators is not evaluated, because its result cannot affect the result. In the following cases, short-circuit evaluation is used and `x()` is not evaluated:

- `FALSE AND x()`
- `FALSE && x()`
- `TRUE OR x()`
- `TRUE || x()`
- `NULL BETWEEN x() AND x()`

Note however that the short-circuit evaluation does *not* apply to `NULL AND x()`. Also, `BETWEEN`'s right operands are not evaluated if the left operand is `NULL`, but in all other cases all the operands are evaluated.

This is a speed optimization. Also, since functions can have side-effects, this behavior can be used to choose whether execute them or not using a concise syntax:

```
SELECT some_function() OR log_error();
```

This page is licensed: CC BY-SA / Gnu FDL

Arithmetic Operators

See Addition Operator (+).

Add two operands with the + operator, calculating integer results at BIGINT precision and letting the highest-precision operand determine the result type for real or string values.

See Subtraction Operator (-).

Subtract one operand from another with the - operator, which also serves as the unary minus for changing sign, computing integer results at BIGINT precision.

See Multiplication Operator (*).

Multiply two operands with the * operator.

See Division Operator (/).

Divide operands with the / operator, returning NULL on division by zero and controlling decimal digits with the div_precision_increment system variable.

See Modulo Operator (%).

Return the remainder of N divided by M with the % modulo operator.

This page is licensed: CC BY-SA / Gnu FDL

Addition Operator (+)

Syntax

```
+
```

Description

Addition.

If both operands are integers, the result is calculated with BIGINT precision. If either integer is unsigned, the result is also an unsigned integer.

For real or string operands, the operand with the highest precision determines the result precision.

Examples

```
SELECT 3+5;
+-----+
| 3+5 |
+-----+
|   8 |
+-----+
```

See Also

- Type Conversion
- Subtraction Operator (-)
- Multiplication Operator (*)
- Division Operator (/)
- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

Subtraction Operator (-)

Syntax

```
-
```

Description

Subtraction. The operator is also used as the unary minus for changing sign.

If both operands are integers, the result is calculated with BIGINT precision. If either integer is unsigned, the result is also an unsigned integer, unless the `NO_UNSIGNED_SUBTRACTION` SQL_MODE is enabled, in which case the result is always signed.

For real or string operands, the operand with the highest precision determines the result precision.

Examples

```
SELECT 96-9;
+------+
| 96-9 |
+------+
|   87 |
+------+

SELECT 15-17;
+-------+
| 15-17 |
+-------+
|    -2 |
+-------+

SELECT 3.66 + 1.333;
+--------------+
| 3.66 + 1.333 |
+--------------+
|        4.993 |
+--------------+
```

Unary minus:

```
SELECT - (3+5);
+---------+
| - (3+5) |
+---------+
|      -8 |
+---------+
```

See Also

- Type Conversion
- Addition Operator (+)
- Multiplication Operator (*)
- Division Operator (/)
- Operator Precedence

This page is licensed: CC BY-SA / Gnu FDL

Multiplication Operator (*)

Syntax

```
*
```

Description

Multiplication operator.

Examples

```
SELECT 7*6;
+-----+
| 7*6 |
+-----+
|  42 |
+-----+

SELECT 1234567890*9876543210;
+-----------------------+
| 1234567890*9876543210 |
+-----------------------+
|  -6253480962446024716 |
+-----------------------+

SELECT 18014398509481984*18014398509481984.0;
+---------------------------------------+
| 18014398509481984*18014398509481984.0 |
+---------------------------------------+
|   324518553658426726783156020576256.0 |
+---------------------------------------+

SELECT 18014398509481984*18014398509481984;
+-------------------------------------+
| 18014398509481984*18014398509481984 |
+-------------------------------------+
|                                   0 |
+-------------------------------------+
```

See Also

- Type Conversion
- Addition Operator (+)
- Subtraction Operator (-)
- Division Operator (/)
- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

Division Operator (/)

Syntax

```
/
```

Description

Division operator. Dividing by zero will return `NULL`. By default, returns four digits after the decimal. This is determined by the server system variable div_precision_increment which by default is four. It can be set from 0 to

30.

Dividing by zero returns `NULL`. If the default `ERROR_ON_DIVISION_BY_ZERO` SQL_MODE is used, a division by zero also produces a warning.

Examples

```
SELECT 4/5;
+--------+
| 4/5    |
+--------+
| 0.8000 |
+--------+

SELECT 300/(2-2);
+-----------+
| 300/(2-2) |
+-----------+
|      NULL |
+-----------+

SELECT 300/7;
+---------+
| 300/7   |
+---------+
| 42.8571 |
+---------+
```

Changing div_precision_increment for the session from the default of four to six:

```
SET div_precision_increment = 6;

SELECT 300/7;
+-----------+
| 300/7     |
+-----------+
| 42.857143 |
+-----------+

SELECT 300/7;
+-----------+
| 300/7     |
+-----------+
| 42.857143 |
+-----------+
```

See Also

- Type Conversion
- Module operator (%)
- Addition Operator (+)
- Subtraction Operator (-)
- Multiplication Operator (*)
- truncate()
- Operator Precedence
- DIV function

This page is licensed: GPLv2, originally from fill_help_tables.sql

Modulo Operator (%)

Syntax

```
N % M
```

Description

Modulo operator. Returns the remainder of `N` divided by `M`. See also MOD.

Examples

```
SELECT 1042 % 50;
+-----------+
| 1042 % 50 |
+-----------+
|        42 |
+-----------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

Assignment Operators

See Assignment Operator (:=).

Assign a value to a user-defined or local variable with the := operator, which, unlike =, can be used in any context.

See Assignment Operator (=).

Assign a value with the = operator, which doubles as a comparison operator and is valid for assignment only in SET statements and the SET clause of UPDATE.

This page is licensed: CC BY-SA / Gnu FDL

Assignment Operator (:=)

Syntax

```
var_name := expr
```

Description

Assignment operator for assigning a value. The value on the right is assigned to the variable on left.

Unlike the = operator, `:=` can always be used to assign a value to a variable.

This operator works with both user-defined variables and local variables.

When assigning the same value to several variables, LAST_VALUE() can be useful.

Examples

```
SELECT @x := 10;
+----------+
| @x := 10 |
+----------+
|       10 |
+----------+

SELECT @x, @y := @x;
+------+----------+
| @x   | @y := @x |
+------+----------+
|   10 |       10 |
+------+----------+
```

See Also

- Operator Precedence

This page is licensed: CC BY-SA / Gnu FDL

Assignment Operator (=)

Syntax

```
identifier = expr
```

Description

The equal sign is used as both an assignment operator in certain contexts, and as a comparison operator. When used as assignment operator, the value on the right is assigned to the variable (or column, in some contexts) on the left.

Since its use can be ambiguous, unlike the := assignment operator, the `=` assignment operator cannot be used in all contexts, and is only valid as part of a SET statement, or the SET clause of an UPDATE statement

This operator works with both user-defined variables and local variables.

Examples

```
UPDATE table_name SET x = 2 WHERE x > 100;
```

```
SET @x = 1, @y := 2;
```

See Also

- Operator Precedence

This page is licensed: CC BY-SA / Gnu FDL

Comparison Operators

- BETWEEN AND
- COALESCE
- =
- >=
- >

- GREATEST
- IN
- INTERVAL
- IS JSON
- IS NOT NULL
- IS NOT
- IS NULL
- IS
- ISNULL
- LEAST
- <=
- <
- NOT BETWEEN
- !=
- NOT IN
- <=>

This page is licensed: CC BY-SA / Gnu FDL

BETWEEN AND

Syntax

```
expr BETWEEN min AND max
```

Description

If `expr` is greater than or equal to `min` and `expr` is less than or equal to `max`, `BETWEEN` returns `1`, otherwise it returns `0`. This is equivalent to the expression (`min <= expr AND expr <= max`) if all the arguments are of the same type. Otherwise type conversion takes place according to the rules described at Type Conversion, but applied to all the three arguments.

Examples

```
SELECT 1 BETWEEN 2 AND 3;
+-------------------+
| 1 BETWEEN 2 AND 3 |
+-------------------+
|                 0 |
+-------------------+
```

```
SELECT 'b' BETWEEN 'a' AND 'c';
+-------------------------+
| 'b' BETWEEN 'a' AND 'c' |
+-------------------------+
|                       1 |
+-------------------------+
```

```
SELECT 2 BETWEEN 2 AND '3';
+---------------------+
| 2 BETWEEN 2 AND '3' |
+---------------------+
|                   1 |
+---------------------+
```

```
SELECT 2 BETWEEN 2 AND 'x-3';
+-----------------------+
| 2 BETWEEN 2 AND 'x-3' |
+-----------------------+
|                     0 |
+-----------------------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1292): Truncated incorrect DOUBLE value: 'x-3'
```

`NULL`:

```
SELECT 1 BETWEEN 1 AND NULL;
+----------------------+
| 1 BETWEEN 1 AND NULL |
+----------------------+
|                 NULL |
+----------------------+
```

`DATE`, `DATETIME` and `TIMESTAMP` examples. Omitting the time component compares against `00:00`, so later times on the same date are not returned:

```
CREATE TABLE `x` (
  a date ,
  b datetime,
  c timestamp
)

INSERT INTO x VALUES
 ('2018-11-11', '2018-11-11 05:15', '2018-11-11 05:15'),
 ('2018-11-12', '2018-11-12 05:15', '2018-11-12 05:15');

SELECT * FROM x WHERE a BETWEEN '2018-11-11' AND '2018-11-12';
+------------+---------------------+---------------------+
| a          | b                   | c                   |
+------------+---------------------+---------------------+
| 2018-11-11 | 2018-11-11 05:15:00 | 2018-11-11 05:15:00 |
| 2018-11-12 | 2018-11-12 05:15:00 | 2018-11-12 05:15:00 |
+------------+---------------------+---------------------+

SELECT * FROM x WHERE b BETWEEN '2018-11-11' AND '2018-11-12';
+------------+---------------------+---------------------+
| a          | b                   | c                   |
+------------+---------------------+---------------------+
| 2018-11-11 | 2018-11-11 05:15:00 | 2018-11-11 05:15:00 |
+------------+---------------------+---------------------+

SELECT * FROM x WHERE c BETWEEN '2018-11-11' AND '2018-11-12';
+------------+---------------------+---------------------+
| a          | b                   | c                   |
+------------+---------------------+---------------------+
| 2018-11-11 | 2018-11-11 05:15:00 | 2018-11-11 05:15:00 |
+------------+---------------------+---------------------+
```

Common Pitfall When Using Strings as Conditions

The following query doesn't show countries whose name starts with 'D':

```
SELECT name
FROM countries
WHERE name BETWEEN "B" AND "D"
ORDER BY name ASC;
+-------------------------------+
| name                          |
+-------------------------------+
| Bahamas                       |
| Bahrain                       |
| Bangladesh                    |
...
| Cuba                          |
| Cyprus                        |
| Czech Republic                |
+-------------------------------+
41 rows in set (0.001 sec)
```

When using `WHERE name BETWEEN 'B' AND 'D'`, the condition includes all names that start with 'B' because 'B' is the lower bound and is inclusive. However, names starting with 'D' are excluded because the upper bound 'D' is treated as a string, and the comparison stops at the character level. For example, a name like 'D' is included if it matches exactly, but a name like 'Denmark' is not included because 'Denmark' is lexicographically greater than 'D'.

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

COALESCE

Syntax

```
COALESCE(value,...)
```

Description

Returns the first non-`NULL` value in the list, or `NULL` if there are no non-`NULL` values. At least one parameter must be passed.

The function is useful when substituting a default value for null values when displaying data.

See also NULL Values in MariaDB.

Examples

```
SELECT COALESCE(NULL,1);
+------------------+
| COALESCE(NULL,1) |
+------------------+
|                1 |
+------------------+
```

```
SELECT COALESCE(NULL,NULL,NULL);
+--------------------------+
| COALESCE(NULL,NULL,NULL) |
+--------------------------+
|                     NULL |
+--------------------------+
```

When two arguments are given, `COALESCE()` is the same as IFNULL():

```
SET @a=NULL, @b=1;

SELECT COALESCE(@a, @b), IFNULL(@a, @b);
+------------------+----------------+
| COALESCE(@a, @b) | IFNULL(@a, @b) |
+------------------+----------------+
|                1 |              1 |
+------------------+----------------+
```

Hex type confusion:

```
CREATE TABLE t1 (a INT, b VARCHAR(10));
INSERT INTO t1 VALUES (0x31, 0x61),(COALESCE(0x31), COALESCE(0x61));

SELECT * FROM t1;
+------+------+
| a    | b    |
+------+------+
|   49 | a    |
|    1 | a    |
+------+------+
```

The reason for the differing results above is that when `0x31` is inserted directly to the column, it's treated as a number (see Hexadecimal Literals), while when `0x31` is passed to `COALESCE()`, it's treated as a string, because:

- `HEX` values have a string data type by default.
- `COALESCE()` has the same data type as the argument.

Substituting zero for `NULL` (in this case when the aggregate function returns `NULL` after finding no rows):

```
SELECT SUM(score) FROM student;
+------------+
| SUM(score) |
+------------+
|       NULL |
+------------+

SELECT COALESCE(SUM(score),0) FROM student;
+------------------------+
| COALESCE(SUM(score),0) |
+------------------------+
|                      0 |
+------------------------+
```

See also

- NULL values
- IS NULL operator
- IS NOT NULL operator
- IFNULL function
- NULLIF function
- CONNECT data types

This page is licensed: GPLv2, originally from fill_help_tables.sql

=

Syntax

```
left_expr = right_expr
```

Description

Equal operator. Evaluates both SQL expressions and returns `1` if they are equal, `0` if they are not equal, or NULL if either expression is `NULL`. If the expressions return different data types (for example, a number and a string), a type conversion is performed.

When used in row comparisons these two queries are synonymous and return the same results:

```
SELECT (t1.a, t1.b) = (t2.x, t2.y) FROM t1 INNER JOIN t2;

SELECT (t1.a = t2.x) AND (t1.b = t2.y) FROM t1 INNER JOIN t2;
```

To perform a `NULL`-safe comparison, use the <=> operator.

`=` can also be used as an assignment operator.

Examples

```
SELECT 1 = 0;
+-------+
| 1 = 0 |
+-------+
|     0 |
+-------+

SELECT '0' = 0;
+---------+
| '0' = 0 |
+---------+
|       1 |
+---------+

SELECT '0.0' = 0;
+-----------+
| '0.0' = 0 |
+-----------+
|         1 |
+-----------+

SELECT '0.01' = 0;
+------------+
| '0.01' = 0 |
+------------+
|          0 |
+------------+

SELECT '.01' = 0.01;
+--------------+
| '.01' = 0.01 |
+--------------+
|            1 |
+--------------+

SELECT (5 * 2) = CONCAT('1', '0');
+----------------------------+
| (5 * 2) = CONCAT('1', '0') |
+----------------------------+
|                          1 |
+----------------------------+

SELECT 1 = NULL;
+----------+
| 1 = NULL |
+----------+
|     NULL |
+----------+

SELECT NULL = NULL;
+-------------+
| NULL = NULL |
+-------------+
|        NULL |
+-------------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

>=

Syntax

```
>=
```

Description

Greater than or equal operator. Evaluates both SQL expressions and returns `1` if the left value is greater than or equal to the right value and `0` if it is not, or `NULL` if either expression is `NULL`. If the expressions return different data types, (for instance, a number and a string), performs type conversion.

When used in row comparisons, these two queries return the same results:

```
SELECT (t1.a, t1.b) >= (t2.x, t2.y)
FROM t1 INNER JOIN t2;

SELECT (t1.a > t2.x) OR ((t1.a = t2.x) AND (t1.b >= t2.y))
FROM t1 INNER JOIN t2;
```

Examples

```
SELECT 2 >= 2;
+--------+
| 2 >= 2 |
+--------+
|      1 |
+--------+

SELECT 'A' >= 'a';
+------------+
| 'A' >= 'a' |
+------------+
|          1 |
+------------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

>

Syntax

```
>
```

Description

Greater than operator. Evaluates both SQL expressions and returns `1` if the left value is greater than the right value and `0` if it is not, or `NULL` if either expression is `NULL`. If the expressions return different data types, (for instance, a number and a string), performs type conversion.

When used in row comparisons these two queries return the same results:

```
SELECT (t1.a, t1.b) > (t2.x, t2.y)
FROM t1 INNER JOIN t2;

SELECT (t1.a > t2.x) OR ((t1.a = t2.x) AND (t1.b > t2.y))
FROM t1 INNER JOIN t2;
```

Examples

```
SELECT 2 > 2;
+-------+
| 2 > 2 |
+-------+
|     0 |
+-------+

SELECT 'b' > 'a';
+-----------+
| 'b' > 'a' |
+-----------+
|         1 |
+-----------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

GREATEST

Syntax

```
GREATEST(value1,value2,...)
```

Description

With two or more arguments, returns the largest (maximum-valued) argument. The arguments are compared using the same rules as for LEAST().

Examples

```
SELECT GREATEST(2,0);
+---------------+
| GREATEST(2,0) |
+---------------+
|             2 |
+---------------+
```

```
SELECT GREATEST(34.0,3.0,5.0,767.0);
+-----------------------------+
| GREATEST(34.0,3.0,5.0,767.0) |
+-----------------------------+
|                       767.0 |
+-----------------------------+
```

```
SELECT GREATEST('B','A','C');
+-----------------------+
| GREATEST('B','A','C') |
+-----------------------+
| C                     |
+-----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

IN

Syntax

```
expr IN (value,...)
```

Description

Returns `1` if *`expr`* is equal to any of the values in the `IN` list, else returns `0`. If all values are constants, they are evaluated according to the type of *`expr`* and sorted. The search for the item then is done using a binary search. This means IN is very quick if the IN value list consists entirely of constants. Otherwise, type conversion takes place according to the rules described at Type Conversion, but applied to all the arguments.

If *`expr`* is `NULL`, `IN` always returns `NULL`. If at least one of the values in the list is `NULL`, and one of the comparisons is true, the result is `1`. If at least one of the values in the list is `NULL` and none of the comparisons is true, the result is `NULL`.

Examples

```
SELECT 2 IN (0,3,5,7);
+----------------+
| 2 IN (0,3,5,7) |
+----------------+
|              0 |
+----------------+
```

```
SELECT 'wefwf' IN ('wee','wefwf','weg');
+---------------------------------+
| 'wefwf' IN ('wee','wefwf','weg') |
+---------------------------------+
|                               1 |
+---------------------------------+
```

Type conversion:

```
SELECT 1 IN ('1', '2', '3');
+----------------------+
| 1 IN ('1', '2', '3') |
+----------------------+
|                    1 |
+----------------------+
```

```
SELECT NULL IN (1, 2, 3);
+-------------------+
| NULL IN (1, 2, 3) |
+-------------------+
|              NULL |
+-------------------+

SELECT 1 IN (1, 2, NULL);
+-------------------+
| 1 IN (1, 2, NULL) |
+-------------------+
|                 1 |
+-------------------+

SELECT 5 IN (1, 2, NULL);
+-------------------+
| 5 IN (1, 2, NULL) |
+-------------------+
|              NULL |
+-------------------+
```

See Also

- Conversion of Big IN Predicates Into Subqueries
- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

INTERVAL

Syntax

```
INTERVAL(N0,N1,N2,N3,...)
```

Description

Returns the index of the last argument that is less than or equal to the first argument, or is NULL.

Returns `0` if `N0 < N1`, `1` if `N1 <= N0 < N2`, `2` if `N2 <= N0 < N3` and so on or `-1` if `N0` is `NULL`. All arguments are treated as integers. It is required that `N1 <= N2 <= N3 <= ... <= Nn` for this function to work correctly because a fast binary search is used.

Examples

```
SELECT INTERVAL(22, 24, 26, 28);
```

```
+-------------------------+
| INTERVAL(22, 24, 26, 28) |
+-------------------------+
|                       0 |
+-------------------------+
```

```
SELECT INTERVAL(22, 22, 22, 22, 23);
```

```
+-----------------------------+
| INTERVAL(22, 22, 22, 22, 23) |
+-----------------------------+
|                           3 |
+-----------------------------+
```

```
SELECT INTERVAL(25, 24, 26, 28);
```

```
+--------------------------+
| interval(25, 24, 26, 28) |
+--------------------------+
|                        1 |
+--------------------------+
```

```
SELECT INTERVAL(27, 24, 26, 28);
```

```
+--------------------------+
| interval(27, 24, 26, 28) |
+--------------------------+
|                        2 |
+--------------------------+
```

```
SELECT INTERVAL(27, 25, 26, 27);
```

```
+--------------------------+
| interval(27, 25, 26, 27) |
+--------------------------+
|                        3 |
+--------------------------+
```

```
SELECT INTERVAL(23, 1, 15, 17, 30, 44, 200);
```

```
+-------------------------------------+
| INTERVAL(23, 1, 15, 17, 30, 44, 200) |
+-------------------------------------+
|                                   3 |
+-------------------------------------+
```

```
SELECT INTERVAL(10, 1, 10, 100, 1000);
```

```
+--------------------------------+
| INTERVAL(10, 1, 10, 100, 1000) |
+--------------------------------+
|                              2 |
+--------------------------------+
```

```
SELECT INTERVAL(22, 23, 30, 44, 200);
```

```
+-------------------------------+
| INTERVAL(22, 23, 30, 44, 200) |
+-------------------------------+
|                             0 |
+-------------------------------+
```

```
SELECT INTERVAL(10, 2, NULL);
```

```
+----------------------+
| INTERVAL(10, 2, NULL) |
+----------------------+
|                    2 |
+----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

IS JSON

Syntax

```
<expr> IS [ NOT ] JSON
       [ { VALUE | ARRAY | OBJECT | SCALAR } ]
       [ { WITH | WITHOUT } UNIQUE [ KEYS ] ]
```

Description

The `IS JSON` predicate checks whether a given string expression evaluates to valid JSON data in accordance with the SQL:2016 standard (based on RFC 8259).

The predicate returns:

- `1` (**TRUE**) if the JSON expression is valid and satisfies any type constraint that is specified (e.g., `OBJECT`, `ARRAY`)
- `0` (**FALSE**) if the expression is not valid JSON or does not match the specified type constraints
- `NULL` (**UNKNOWN**) if the JSON expression itself evaluates to `NULL`

When the value is invalid JSON, the predicate does not show errors. Any valid JSON value is accepted since VALUE is considered when no type constraint is mentioned.

Type Constraints

To limit the permitted top-level JSON value, use the optional type constraints listed below:

Type	Description
`VALUE`	Any valid JSON value (object, array, number, string, `true`, `false`, `null`)
`ARRAY`	Only JSON arrays (e.g., `[1,2,3]`
`OBJECT`	Only JSON objects (e.g., {“key”: “value”} = {“a”: “42”})
`SCALAR`	A JSON scalar value (strings, numbers, Boolean, or `null`)

Unique Keys

The object keys must be unique in order to use `WITH UNIQUE KEYS` option. `WITHOUT UNIQUE KEYS`, duplicate keys are accepted.

Notes

- JSON Validation is based on RFC 8259.
- JSON literal names must be all lowercase: `true`, `false`, `null`.
- Other literal names are not permitted.
- `IS JSON` works in generated columns, CHECK constraints, and DEFAULT expressions.

Examples

- **Basic JSON Validation**

```
SELECT '"abc"' IS JSON;
```

Output

```
+-----------------+
| '"abc"' IS JSON |
+-----------------+
|               1|
+-----------------+
```

- **Type-specific Checking**

```
SELECT ''{"a": "42"}' IS JSON OBJECT, '42' IS JSON SCALAR, '[1,2,3]' IS JSON ARRAY;
```

Output

```
+---------------------------+----------------------+-------------------------+
| '{"a": "42"}' IS JSON OBJECT |  '42' IS JSON SCALAR  | '[1,2,3]' IS JSON ARRAY  |
+---------------------------+----------------------+-------------------------+
|                          1|                     1|                        1|
+---------------------------+----------------------+-------------------------+
```

- **NULL Handling**

```
SELECT 'NULL' IS JSON;
```

Output

```
+-----------------+
| 'NULL' IS JSON  |
+-----------------+
|               0|
+-----------------+
```

- **JSON Literal Names**

```
SELECT 'null' IS JSON, 'NULL' IS JSON;
```

Output

```
+-----------------+--------------+
| 'null' IS JSON  | NULL IS JSON |
+-----------------+--------------+
|               1|          NULL|
+-----------------+--------------+
```

- **Negation with `IS NOT JSON`**

```
SELECT 'invalid' IS NOT JSON;
```

Output

```
+------------------------+
| 'invalid' IS NOT JSON  |
+------------------------+
|                       1|
+------------------------+
```

- **Unique Keys Validation**

```
SELECT '{"a": 42, "a":1}' IS JSON;
```

Output

```
+----------------------------+
| '{"a": 42, "a":1}' IS JSON  |
+----------------------------+
|                           1|
+----------------------------+
```

With Unique Keys (reject duplicates)

```
SELECT '{"a": 42, "a":1}' IS JSON WITH UNIQUE KEYS;
```

Output

```
+-------------------------------------------+
| '{"a": 42, "a":1}' IS JSON WITH UNIQUE KEYS  |
+-------------------------------------------+
|                                          0|
+-------------------------------------------+
```

See Also

- IS Operator
- JSON_VALID()
- JSON Data Type
- JSON_TYPE

This page is licensed: CC BY-SA / Gnu FDL

IS NOT NULL

Syntax

```
IS NOT NULL
```

Description

Tests whether a value is not `NULL`. See also NULL Values in MariaDB.

Examples

```
SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;
+--------------+--------------+------------------+
| 1 IS NOT NULL | 0 IS NOT NULL | NULL IS NOT NULL |
+--------------+--------------+------------------+
|            1 |            1 |                0 |
+--------------+--------------+------------------+
```

See also

- NULL values
- IS NULL operator
- COALESCE function
- IFNULL function
- NULLIF function
- CONNECT data types

This page is licensed: GPLv2, originally from fill_help_tables.sql

IS NOT

Syntax

```
IS NOT boolean_value
```

Description

Tests a value against a boolean value, where `boolean_value` can be `TRUE`, `FALSE`, or `UNKNOWN`.

Examples

```
SELECT 1 IS NOT UNKNOWN, 0 IS NOT UNKNOWN, NULL IS NOT UNKNOWN;
+------------------+------------------+---------------------+
| 1 IS NOT UNKNOWN | 0 IS NOT UNKNOWN | NULL IS NOT UNKNOWN |
+------------------+------------------+---------------------+
|                1 |                1 |                   0 |
+------------------+------------------+---------------------+
```

```
SELECT NULL IS NOT TRUE, NULL IS NOT FALSE;
+------------------+-------------------+
| NULL IS NOT TRUE | NULL IS NOT FALSE |
+------------------+-------------------+
|                1 |                 1 |
+------------------+-------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

IS NULL

Syntax

```
IS NULL
```

Description

Tests whether a value is `NULL`. See also NULL Values in MariaDB.

Examples

```
SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;
+-----------+-----------+--------------+
| 1 IS NULL | 0 IS NULL | NULL IS NULL |
+-----------+-----------+--------------+
|         0 |         0 |            1 |
+-----------+-----------+--------------+
```

Compatibility

Some `ODBC` applications use the syntax `auto_increment_field IS NOT NULL` to find the latest row that was inserted with an autogenerated key value. If your applications need this, you can set the sql_auto_is_null variable to `1`.

```
SET @@sql_auto_is_null=1;
CREATE TABLE t1 (auto_increment_column INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
INSERT INTO t1 VALUES (NULL);
SELECT * FROM t1 WHERE auto_increment_column IS NULL;

+-----------------------+
| auto_increment_column |
+-----------------------+
|                     1 |
+-----------------------+
```

See also

- NULL values
- IS NOT NULL operator
- COALESCE function
- IFNULL function
- NULLIF function
- CONNECT data types

This page is licensed: GPLv2, originally from fill_help_tables.sql

IS

Syntax

```
IS boolean_value
```

Description

Tests a value against a boolean value, where `boolean_value` can be `TRUE`, `FALSE`, or `UNKNOWN`.

There is an important difference between using `IS TRUE` or comparing a value with `TRUE` using `=`. When using `=`, only `1` equals to `TRUE`. But when using `IS TRUE`, all values which are logically true (like a number > 1) return `TRUE`.

Examples

```
SELECT 1 IS TRUE, 0 IS FALSE, NULL IS UNKNOWN;
+-----------+------------+-----------------+
| 1 IS TRUE | 0 IS FALSE | NULL IS UNKNOWN |
+-----------+------------+-----------------+
|         1 |          1 |               1 |
+-----------+------------+-----------------+
```

Difference between `=` and `IS TRUE`:

```
SELECT 2 = TRUE, 2 IS TRUE;
+----------+-----------+
| 2 = TRUE | 2 IS TRUE |
+----------+-----------+
|        0 |         1 |
+----------+-----------+
```

See Also

- Boolean Literals
- BOOLEAN Data Type

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

ISNULL

Syntax

```
ISNULL(expr)
```

Description

If `expr` is `NULL`, `ISNULL()` returns `1`, otherwise it returns 0.

See also NULL Values in MariaDB.

Examples

```
SELECT ISNULL(1+1);
+-------------+
| ISNULL(1+1) |
+-------------+
|           0 |
+-------------+

SELECT ISNULL(1/0);
+-------------+
| ISNULL(1/0) |
+-------------+
|           1 |
+-------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LEAST

Syntax

```
LEAST(value1,value2,...)
```

Description

With two or more arguments, returns the smallest (minimum-valued) argument. The arguments are compared using the following rules:

- If the return value is used in an `INTEGER` context or all arguments are integer-valued, they are compared as integers.
- If the return value is used in a `REAL` context or all arguments are real-valued, they are compared as reals.
- If any argument is a case-sensitive string, the arguments are compared as case-sensitive strings.
- In all other cases, the arguments are compared as case-insensitive strings.

`LEAST()` returns `NULL` if any argument is `NULL`.

Examples

```
SELECT LEAST(2,0);
+------------+
| LEAST(2,0) |
+------------+
|          0 |
+------------+
```

```
SELECT LEAST(34.0,3.0,5.0,767.0);
+--------------------------+
| LEAST(34.0,3.0,5.0,767.0) |
+--------------------------+
|                      3.0 |
+--------------------------+
```

```
SELECT LEAST('B','A','C');
+--------------------+
| LEAST('B','A','C') |
+--------------------+
| A                  |
+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

<=

Syntax

```
<=
```

Description

Less than or equal operator. Evaluates both SQL expressions and returns `1` if the left value is less than or equal to the right value and `0` if it is not, or `NULL` if either expression is `NULL`. If the expressions return different data types, (for instance, a number and a string), performs type conversion.

When used in row comparisons these two queries return the same results:

```
SELECT (t1.a, t1.b) <= (t2.x, t2.y)
FROM t1 INNER JOIN t2;

SELECT (t1.a < t2.x) OR ((t1.a = t2.x) AND (t1.b <= t2.y))
FROM t1 INNER JOIN t2;
```

Examples

```
SELECT 0.1 <= 2;
+----------+
| 0.1 <= 2 |
+----------+
|        1 |
+----------+
```

```
SELECT 'a'<='A';
+----------+
| 'a'<='A' |
+----------+
|        1 |
+----------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

<

Syntax

```
<
```

Description

Less than operator. Evaluates both SQL expressions and returns `1` if the left value is less than the right value and `0` if it is not, or `NULL` if either expression is `NULL`. If the expressions return different data types, (for instance, a number and a string), performs type conversion.

When used in row comparisons, these two queries return the same results:

```
SELECT (t1.a, t1.b) < (t2.x, t2.y)
FROM t1 INNER JOIN t2;

SELECT (t1.a < t2.x) OR ((t1.a = t2.x) AND (t1.b < t2.y))
FROM t1 INNER JOIN t2;
```

Examples

```
SELECT 2 < 2;
+-------+
| 2 < 2 |
+-------+
|     0 |
+-------+
```

Type conversion:

```
SELECT 3<'4';
+-------+
| 3<'4' |
+-------+
|     1 |
+-------+
```

Case insensitivity - see Character Sets and Collations:

```
SELECT 'a'<'A';
+---------+
| 'a'<'A' |
+---------+
|       0 |
+---------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

NOT BETWEEN

Syntax

```
expr NOT BETWEEN min AND max
```

Description

This is the same as `NOT ( expr` BETWEEN `min AND max )`.

Note that the meaning of the alternative form `NOT expr BETWEEN min AND max` is affected by the `HIGH_NOT_PRECEDENCE` SQL_MODE flag.

Examples

```
SELECT 1 NOT BETWEEN 2 AND 3;
+----------------------+
| 1 NOT BETWEEN 2 AND 3 |
+----------------------+
|                     1 |
+----------------------+
```

```
SELECT 'b' NOT BETWEEN 'a' AND 'c';
+----------------------------+
| 'b' NOT BETWEEN 'a' AND 'c' |
+----------------------------+
|                          0 |
+----------------------------+
```

`NULL`:

```
SELECT 1 NOT BETWEEN 1 AND NULL;
+--------------------------+
| 1 NOT BETWEEN 1 AND NULL |
+--------------------------+
|                     NULL |
+--------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

!=

Syntax

```
<>, !=
```

Description

Not equal operator. Evaluates both SQL expressions and returns `1` if they are not equal, and `0` if they are equal, or `NULL` if either expression is `NULL`. If the expressions return different data types, (for instance, a number and a string), performs type conversion.

When used in row comparisons these two queries return the same results:

```
SELECT (t1.a, t1.b) != (t2.x, t2.y)
FROM t1 INNER JOIN t2;

SELECT (t1.a != t2.x) OR (t1.b != t2.y)
FROM t1 INNER JOIN t2;
```

Examples

```
SELECT '.01' <> '0.01';
+-----------------+
| '.01' <> '0.01' |
+-----------------+
|               1 |
+-----------------+

SELECT .01 <> '0.01';
+---------------+
| .01 <> '0.01' |
+---------------+
|             0 |
+---------------+

SELECT 'zapp' <> 'zappp';
+-------------------+
| 'zapp' <> 'zappp' |
+-------------------+
|                 1 |
+-------------------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

NOT IN

Syntax

```
expr NOT IN (value,...)
```

Description

This is the same as `NOT` (`expr` IN (value,...)).

Examples

```
SELECT 2 NOT IN (0,3,5,7);
+--------------------+
| 2 NOT IN (0,3,5,7) |
+--------------------+
|                  1 |
+--------------------+
```

```
SELECT 'wefwf' NOT IN ('wee','wefwf','weg');
+--------------------------------------+
| 'wefwf' NOT IN ('wee','wefwf','weg') |
+--------------------------------------+
|                                    0 |
+--------------------------------------+
```

```
SELECT 1 NOT IN ('1', '2', '3');
+--------------------------+
| 1 NOT IN ('1', '2', '3') |
+--------------------------+
|                        0 |
+--------------------------+
```

`NULL`:

```
SELECT NULL NOT IN (1, 2, 3);
+----------------------+
| NULL NOT IN (1, 2, 3) |
+----------------------+
|                 NULL |
+----------------------+

SELECT 1 NOT IN (1, 2, NULL);
+----------------------+
| 1 NOT IN (1, 2, NULL) |
+----------------------+
|                    0 |
+----------------------+

SELECT 5 NOT IN (1, 2, NULL);
+----------------------+
| 5 NOT IN (1, 2, NULL) |
+----------------------+
|                 NULL |
+----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

<=>

Syntax

```
<=>
```

Description

`NULL`-safe equal operator. It performs an equality comparison like the = operator, but returns `1` rather than `NULL` if both operands are `NULL`, and `0` rather than `NULL` if one operand is `NULL`.

`a <=> b` is equivalent to `a = b OR (a IS NULL AND b IS NULL)`.

When used in row comparisons, these two queries return the same results:

```
SELECT (t1.a, t1.b) <=> (t2.x, t2.y)
FROM t1 INNER JOIN t2;

SELECT (t1.a <=> t2.x) AND (t1.b <=> t2.y)
FROM t1 INNER JOIN t2;
```

See also NULL Values in MariaDB.

Examples

```
SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;
+---------+---------------+------------+
| 1 <=> 1 | NULL <=> NULL | 1 <=> NULL |
+---------+---------------+------------+
|       1 |             1 |          0 |
+---------+---------------+------------+

SELECT 1 = 1, NULL = NULL, 1 = NULL;
+-------+-------------+----------+
| 1 = 1 | NULL = NULL | 1 = NULL |
+-------+-------------+----------+
|     1 |        NULL |     NULL |
+-------+-------------+----------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

Logical Operators

See &&.

Evaluate logical AND with AND or &&, returning 1 when all operands are non-zero and not NULL, 0 when any operand is 0, and NULL otherwise.

See !.

Negate an operand with logical NOT or !, returning 1 for 0, 0 for non-zero values, and NULL for NULL, with precedence affected by the HIGH_NOT_PRECEDENCE SQL_MODE.

See ||.

Evaluate logical OR with OR or ||, returning 1 when any operand is non-zero and handling NULL operands per SQL logic; || concatenates strings under PIPES_AS_CONCAT.

See XOR.

Evaluate exclusive OR with XOR, returning NULL if either operand is NULL and otherwise 1 when an odd number of operands is non-zero.

This page is licensed: CC BY-SA / Gnu FDL

&&

Syntax

```
AND, &&
```

Description

Logical `AND`. Evaluates to `1` if all operands are non-zero and not `NULL`, to `0` if one or more operands are `0`, otherwise `NULL` is returned.

For this operator, short-circuit evaluation can be used.

Examples

```
SELECT 1 && 1;
+--------+
| 1 && 1 |
+--------+
|      1 |
+--------+

SELECT 1 && 0;
+--------+
| 1 && 0 |
+--------+
|      0 |
+--------+

SELECT 1 && NULL;
+-----------+
| 1 && NULL |
+-----------+
|      NULL |
+-----------+

SELECT 0 && NULL;
+-----------+
| 0 && NULL |
+-----------+
|         0 |
+-----------+

SELECT NULL && 0;
+-----------+
| NULL && 0 |
+-----------+
|         0 |
+-----------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

!

Syntax

```
NOT, !
```

Description

Logical `NOT`. Evaluates to `1` if the operand is `0`, to `0` if the operand is non-zero, and `NOT NULL` returns `NULL`.

By default, the `!` operator has a higher precedence. If the `HIGH_NOT_PRECEDENCE` SQL_MODE flag is set, `NOT` and `!` have the same precedence.

Examples

```
SELECT NOT 10;
+--------+
| NOT 10 |
+--------+
|      0 |
+--------+

SELECT NOT 0;
+-------+
| NOT 0 |
+-------+
|     1 |
+-------+

SELECT NOT NULL;
+----------+
| NOT NULL |
+----------+
|     NULL |
+----------+

SELECT ! (1+1);
+---------+
| ! (1+1) |
+---------+
|       0 |
+---------+

SELECT ! 1+1;
+-------+
| ! 1+1 |
+-------+
|     1 |
+-------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

||

Syntax

```
OR, ||
```

Description

Logical `OR`. When both operands are non-`NULL`, the result is `1` if any operand is non-zero, and `0` otherwise. With a `NULL` operand, the result is `1` if the other operand is non-zero, and `NULL` otherwise. If both operands are `NULL`, the result is `NULL`.

For this operator, short-circuit evaluation can be used.

> **Warning:**
>
> Note that, if the `PIPES_AS_CONCAT` SQL_MODE is set, `||` is used as a string concatenation operator. This means that `a || b` is the same as `CONCAT(a,b)`. See CONCAT() for details.

Oracle Mode

In Oracle mode, `||` ignores null.

Examples

```
SELECT 1 || 1;
+--------+
| 1 || 1 |
+--------+
|      1 |
+--------+

SELECT 1 || 0;
+--------+
| 1 || 0 |
+--------+
|      1 |
+--------+

SELECT 0 || 0;
+--------+
| 0 || 0 |
+--------+
|      0 |
+--------+

SELECT 0 || NULL;
+-----------+
| 0 || NULL |
+-----------+
|      NULL |
+-----------+

SELECT 1 || NULL;
+-----------+
| 1 || NULL |
+-----------+
|         1 |
+-----------+
```

In Oracle mode:

```
SELECT 0 || NULL;
+-----------+
| 0 || NULL |
+-----------+
| 0         |
+-----------+
```

See Also

See Also

- Operator Precedence
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

XOR

Syntax

```
XOR
```

Description

`XOR` stands for eXclusive OR. Returns `NULL` if either operand is `NULL`. For non-`NULL` operands, evaluates to `1` if an odd number of operands is non-zero, otherwise `0` is returned.

Examples

```
SELECT 1 XOR 1;
+---------+
| 1 XOR 1 |
+---------+
|       0 |
+---------+

SELECT 1 XOR 0;
+---------+
| 1 XOR 0 |
+---------+
|       1 |
+---------+

SELECT 1 XOR NULL;
+------------+
| 1 XOR NULL |
+------------+
|       NULL |
+------------+
```

In the following example, the right `1 XOR 1` is evaluated first, and returns `0`. Then, `1 XOR 0` is evaluated, and `1` is returned.

```
SELECT 1 XOR 1 XOR 1;
+---------------+
| 1 XOR 1 XOR 1 |
+---------------+
|             1 |
+---------------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

Sequences

See Sequence Overview.

An overview of sequence objects, which generate a sequence of numeric values as defined by CREATE SEQUENCE.

See CREATE SEQUENCE.

Create a sequence generator. This statement initializes a sequence object that produces a series of unique numeric values on demand.

See SHOW CREATE SEQUENCE.

View the SQL used to create a sequence. This statement displays the CREATE SEQUENCE statement with current parameter values.

See ALTER SEQUENCE.

ALTER SEQUENCE changes the properties of an existing sequence object.

See DROP SEQUENCE.

DROP SEQUENCE removes one or more sequence objects.

See SEQUENCE Functions.

Learn about sequence functions in MariaDB Server. This section details SQL functions for retrieving the next or current value from a sequence, crucial for generating unique identifiers.

See Information Schema SEQUENCES Table.

The Information Schema SEQUENCES table provides metadata about sequence objects, including their minimum, maximum, and current values.

See SHOW TABLES.

Complete guide to listing tables in MariaDB. Complete SHOW TABLES syntax reference with LIKE patterns, WHERE conditions, and filtering options.

See Error 4084: Sequence has run out.

Error 4084: raised when a sequence without CYCLE reaches its MAXVALUE and can generate no more values.

This page is licensed: CC BY-SA / Gnu FDL

Sequence Overview

This page is about sequence objects. For details about the storage engine, see Sequence Storage Engine.

Introduction

A sequence is an object that generates a sequence of numeric values, as specified by the CREATE SEQUENCE statement.

`CREATE SEQUENCE` creates a sequence that generates new values when called with `NEXT VALUE FOR` sequence_name. It's an alternative to AUTO INCREMENT when one wants to have more control of how the numbers are generated. As the `SEQUENCE` caches values (up to the `CACHE` value in the CREATE SEQUENCE statement, by default 1000) it can in some cases be much faster than AUTO INCREMENT. Another benefit is that one can access the last value generated by all used sequences, which solves one of the limitations with LAST_INSERT_ID().

> **Warning:**
>
> Sequences should not be used with statement-based logging; see this section for details.

Creating a Sequence

The CREATE SEQUENCE statement is used to create a sequence. Here is an example of a sequence starting at 100, incrementing by 10 each time:

```
CREATE SEQUENCE s START WITH 100 INCREMENT BY 10;
```

The CREATE SEQUENCE statement, along with defaults, can be viewd with the SHOW CREATE SEQUENCE STATEMENT, for example:

```
SHOW CREATE SEQUENCE s\G
*************************** 1. row ***************************
       Table: s
Create Table: CREATE SEQUENCE `s` start with 100 minvalue 1 maxvalue 9223372036854775806
  increment by 10 cache 1000 nocycle ENGINE=InnoDB
```

Using Sequence Objects

To get the next value from a sequence, use

```
NEXT VALUE FOR sequence_name
```

or

```
NEXTVAL(sequence_name)
```

or in Oracle mode (SQL_MODE=ORACLE)

```
sequence_name.nextval
```

For retrieving the last value used by the current connection from a sequence use:

```
PREVIOUS VALUE FOR sequence_name
```

or

```
LASTVAL(sequence_name)
```

or in Oracle mode (SQL_MODE=ORACLE)

```
sequence_name.currval
```

For example:

```
SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|        100 |
+------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|        110 |
+------------+

SELECT LASTVAL(s);
+------------+
| LASTVAL(s) |
+------------+
|        110 |
+------------+
```

Using Sequences in DEFAULT

Sequences can be used in `DEFAULT`:

```
CREATE SEQUENCE s1;
CREATE TABLE t1 (a INT PRIMARY KEY DEFAULT (NEXT VALUE FOR s1), b INT);
INSERT INTO t1 (b) VALUES (1),(2);
SELECT * FROM t1;
+---+------+
| a | b    |
+---+------+
| 1 |    1 |
| 2 |    2 |
+---+------+
```

Changing a Sequence

The ALTER SEQUENCE statement is used for changing sequences. For example, to restart the sequence at another value:

```
ALTER SEQUENCE s RESTART 50;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|         50 |
+------------+
```

The SETVAL function can also be used to set the next value to be returned for a `SEQUENCE`, for example:

```
SELECT SETVAL(s, 100);
+----------------+
| SETVAL(s, 100) |
+----------------+
|            100 |
+----------------+
```

`SETVAL` can only be used to increase the sequence value. Attempting to set a lower value will fail, returning `NULL`:

```
SELECT SETVAL(s, 50);
+---------------+
| SETVAL(s, 50) |
+---------------+
|          NULL |
+---------------+
```

Dropping a Sequence

The DROP SEQUENCE statement is used to drop a sequence, for example:

```
DROP SEQUENCE s;
```

Replication

If you want to use Sequences in a master-master setup or with Galera, you should use `INCREMENT=0`. This tells the Sequence to use auto_increment_increment and auto_increment_offset to generate unique values for each server.

Statement-Based Replication Limitation

Using `SELECT NEXT VALUE` causes an error for statement-based logging:

```
ERROR 1665 (HY000): Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT =
STATEMENT and at least one table uses a storage engine limited to row-based logging.
```

This happens because `SELECT` modifies objects.

Standards Compliance

MariaDB supports both ANSI SQL and Oracle syntax for sequences.

However as `SEQUENCE` is implemented as a special kind of table, it uses the same namespace as tables. The benefits are that sequences show up in SHOW TABLES, and one can also create a sequence with CREATE TABLE and drop it with DROP TABLE. One can SELECT from it as from any other table. This ensures that all old tools that work with tables should work with sequences.

Since sequence objects act as regular tables in many contexts, they will be affected by LOCK TABLES. This is not the case in other DBMS, such as Oracle, where LOCK TABLE does not affect sequences.

Notes

One of the goals with the Sequence implementation is that all old tools, such as mariadb-dump (previously mysqldump), should work unchanged, while still keeping the normal usage of sequence standard compatibly.

To make this possible, `sequence` is currently implemented as a table with a few exclusive properties.

The special properties for sequence tables are:

- A sequence table has always one row.
- When one creates a sequence, either with CREATE TABLE or CREATE SEQUENCE, one row will be inserted.
- If one tries to insert into a sequence table, the single row will be updated. This allows mariadb-dump to work but also gives the additional benefit that one can change all properties of a sequence with a single insert. New applications should of course also use `ALTER SEQUENCE`.
- UPDATE or DELETE can't be performed on Sequence objects.
- Doing a select on the sequence shows the current state of the sequence, except the values that are reserved in the cache. The `next_value` column shows the next value not reserved by the cache.
- FLUSH TABLES will close the sequence and the next sequence number generated will be according to what's stored in the Sequence object. In effect, this will discard the cached values.
- A number of normal table operations work on Sequence tables. See next section.

Table Operations that Work with Sequences

- SHOW CREATE TABLE sequence_name. This shows the table structure that is behind the `SEQUENCE` including the field names that can be used with SELECT or even CREATE TABLE.
- CREATE TABLE sequence-structure ... SEQUENCE=1
- ALTER TABLE sequence RENAME TO sequence2
- RENAME TABLE sequence_name TO new_sequence_name
- DROP TABLE sequence_name. This is allowed mainly to get old tools like mariadb-dump to work with sequence tables.
- SHOW TABLES

Implementation

Internally, sequence tables are created as a normal table without rollback (the InnoDB, Aria and MySAM engines support this), wrapped by a sequence engine object. This allowed us to create sequences with almost no performance impact for normal tables. (The cost is one 'if' per insert if the binary log is enabled).

Underlying Table Structure

The following example shows the table structure of sequences and how it can be used as a table.
(Output of results are slightly edited to make them easier to read.)

```
CREATE SEQUENCE t1;
SHOW CREATE SEQUENCE t1\G
*************************** 1. row ***************************
  CREATE SEQUENCE `t1` start with 1 minvalue 1 maxvalue 9223372036854775806
  increment by 1 cache 1000 nocycle ENGINE=InnoDB

SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
Create Table: CREATE TABLE `t1` (
  `next_not_cached_value` bigint(21) NOT NULL,
  `minimum_value` bigint(21) NOT NULL,
  `maximum_value` bigint(21) NOT NULL,
  `start_value` bigint(21) NOT NULL COMMENT 'start value when sequences is created or value if RESTART is
used',
  `increment` bigint(21) NOT NULL COMMENT 'increment value',
  `cache_size` bigint(21) unsigned NOT NULL,
  `cycle_option` tinyint(1) unsigned NOT NULL COMMENT '0 if no cycles are allowed, 1 if the sequence
should begin a new cycle when maximum_value is passed',
  `cycle_count` bigint(21) NOT NULL COMMENT 'How many cycles have been done'
) ENGINE=InnoDB SEQUENCE=1

SELECT * FROM t1\G
next_not_cached_value: 1
 minimum_value: 1
 maximum_value: 9223372036854775806
  start_value: 1
  increment: 1
  cache_size: 1000
  cycle_option: 0
  cycle_count: 0
```

The `cycle_count` column is incremented every time the sequence wraps around.

Credits

- Thanks to Jianwe Zhao from Aliyun for his work on `SEQUENCE` in AliSQL, which gave ideas and inspiration for this work.
- Thanks to Peter Gulutzan, who helped test and gave useful comments about the implementation.

See Also

- CREATE SEQUENCE
- ALTER SEQUENCE
- DROP SEQUENCE
- NEXT VALUE FOR
- PREVIOUS VALUE FOR
- SETVAL(). Set next value for the sequence.
- AUTO INCREMENT
- Sequence Storage Engine
- Information Schema SEQUENCES Table
- Error 4084: Sequence has run out

This page is licensed: CC BY-SA / Gnu FDL

ALTER SEQUENCE

Syntax

```
ALTER SEQUENCE [IF EXISTS] sequence_name
[AS { TINYINT | SMALLINT | |MEDIUMINT | INT | INTEGER | BIGINT } [SIGNED | UNSIGNED]]
[ INCREMENT [ BY | = ] number ]
[ MINVALUE [=] number | NO MINVALUE | NOMINVALUE ]
[ MAXVALUE [=] number | NO MAXVALUE | NOMAXVALUE ]
[ START [ WITH | = ] number ] [ CACHE [=] number ] [ [ NO ] CYCLE ]
[ RESTART [[WITH | =] number]
```

`ALTER SEQUENCE` allows one to change any values for a `SEQUENCE` created with CREATE SEQUENCE.

The options for `ALTER SEQUENCE` can be given in any order.

Description

`ALTER SEQUENCE` changes the parameters of an existing sequence generator. Any parameters not specifically set in the `ALTER SEQUENCE` command retain their prior settings.

`ALTER SEQUENCE` requires the ALTER privilege.

Options to `ALTER SEQUENCE`

Option	Default value	Description
INCREMENT	1	Increment to use for values. May be negative.
MINVALUE	1 if INCREMENT > 0 and -9223372036854775807 if INCREMENT < 0	Minimum value for the sequence.
MAXVALUE	9223372036854775806 if INCREMENT > 0 and -1 if INCREMENT < 0	Max value for sequence.
START	MINVALUE if INCREMENT > 0 and MAX_VALUE if INCREMENT< 0	First value that the sequence will generate.
CACHE	1000	Number of values that should be cached. 0 if no CACHE. The underlying table will be updated first time a new sequence number is generated and each time the cache runs out.
CYCLE	0 (= NO CYCLE)	1 if the sequence should start again from MINVALUE

The optional clause `RESTART [ WITH restart ]` sets the next value for the sequence. This is equivalent to calling the SETVAL() function with the `is_used` argument as `0`. The specified value will be returned by the next call of `nextval`. Using `RESTART` with no restart value is equivalent to supplying the start value that was recorded by CREATE SEQUENCE or last set by `ALTER SEQUENCE START WITH`.

`ALTER SEQUENCE` does not allow to change the sequence so that it's inconsistent:

```
CREATE SEQUENCE s1;
ALTER SEQUENCE s1 MINVALUE 10;
ERROR 4061 (HY000): Sequence 'test.t1' values are conflicting

ALTER SEQUENCE s1 MINVALUE 10 RESTART 10;
ERROR 4061 (HY000): Sequence 'test.t1' values are conflicting

ALTER SEQUENCE s1 MINVALUE 10 START 10 RESTART 10;
```

INSERT

To allow `SEQUENCE` objects to be backed up by old tools, like mariadb-dump, one can use `SELECT` to read the current state of a `SEQUENCE` object and use an `INSERT` to update the `SEQUENCE` object. `INSERT` is only allowed if all fields are specified:

```
CREATE SEQUENCE s1;
INSERT INTO s1 VALUES(1000,10,2000,1005,1,1000,0,0);
SELECT * FROM s1;

+------------+-----------+-----------+-------+-----------+-------+-------+-------+
| next_value | min_value | max_value | start | increment | cache | cycle | round |
+------------+-----------+-----------+-------+-----------+-------+-------+-------+
|       1000 |        10 |      2000 |  1005 |         1 |  1000 |     0 |     0 |
+------------+-----------+-----------+-------+-----------+-------+-------+-------+

SHOW CREATE SEQUENCE s1;
+-------+-----------------------------------------------------------------------------------------------------------
--------------+
| Table | Create Table
|
+-------+-----------------------------------------------------------------------------------------------------------
--------------+
| s1    | CREATE SEQUENCE `s1` start with 1005 minvalue 10 maxvalue 2000 increment by 1 cache 1000
nocycle ENGINE=Aria |
+-------+-----------------------------------------------------------------------------------------------------------
--------------+
```

Notes

`ALTER SEQUENCE` will instantly affect all future `SEQUENCE` operations. This is in contrast to some other databases where the changes requested by `ALTER SEQUENCE` will not be seen until the sequence cache has run out.

`ALTER SEQUENCE` will take a full table lock of the sequence object during its (brief) operation. This ensures that `ALTER SEQUENCE` is replicated correctly. If you only want to set the next sequence value to a higher value than current, then you should use SETVAL() instead, as this is not blocking.

If you want to change the storage engine or sequence comment, or rename the sequence, you can use ALTER TABLE for this.

See Also

- Sequence Overview
- CREATE SEQUENCE
- DROP SEQUENCE
- NEXT VALUE FOR
- PREVIOUS VALUE FOR
- SETVAL(). Set next value for the sequence.
- AUTO INCREMENT
- ALTER TABLE
- Information Schema SEQUENCES Table

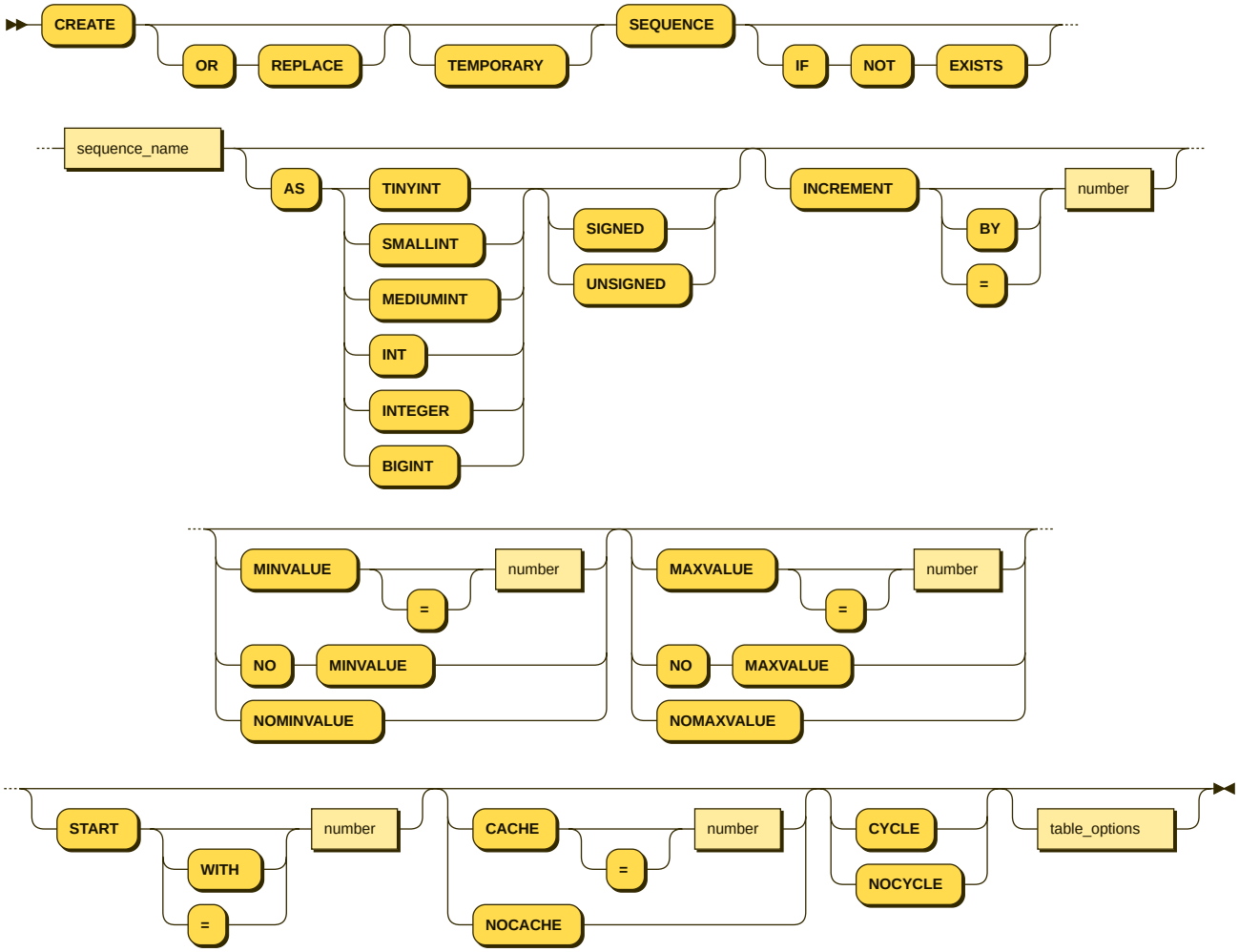
This page is licensed: CC BY-SA / Gnu FDL

CREATE SEQUENCE

Syntax

```
CURRENT
```

```
CREATE [OR REPLACE] [TEMPORARY] SEQUENCE [IF NOT EXISTS] sequence_name
[AS { TINYINT | SMALLINT | MEDIUMINT | INT | INTEGER | BIGINT } [SIGNED | UNSIGNED]]
[ INCREMENT [ BY | = ] number ]
[ MINVALUE [=] number | NO MINVALUE | NOMINVALUE ]
[ MAXVALUE [=] number | NO MAXVALUE | NOMAXVALUE ]
[ START [ WITH | = ] number ]
[ CACHE [=] number | NOCACHE ] [ CYCLE | NOCYCLE]
[table_options]
```



Railroad diagram of CREATE SEQUENCE — equivalent to the BNF above

< 11.5

```
CREATE [OR REPLACE] [TEMPORARY] SEQUENCE [IF NOT EXISTS] sequence_name
[ INCREMENT [ BY | = ] number ]
[ MINVALUE [=] number | NO MINVALUE | NOMINVALUE ]
[ MAXVALUE [=] number | NO MAXVALUE | NOMAXVALUE ]
[ START [ WITH | = ] number ]
[ CACHE [=] number | NOCACHE ] [ CYCLE | NOCYCLE]
[table_options]
```

The options for `CREATE SEQUENCE` can be given in any order, optionally followed by table_options.

`table_options` can be any of the normal table options in CREATE TABLE — the most used ones are `ENGINE=...` and `COMMENT=`.

`NOMAXVALUE` and `NOMINVALUE` are there to allow one to create `SEQUENCE`s using the Oracle syntax.

Description

`CREATE SEQUENCE` creates a sequence that generates new values when called with `NEXT VALUE FOR sequence_name`. It's an alternative to AUTO INCREMENT if you want to have more control of how the numbers are generated. As the `SEQUENCE` caches values (up to `CACHE`), it can in some cases be much faster than AUTO INCREMENT. Another benefit is that you can access the last value generated by all used sequences, which solves one of the limitations with LAST_INSERT_ID().

`CREATE SEQUENCE` requires the CREATE privilege.

DROP SEQUENCE can be used to drop a sequence, and ALTER SEQUENCE to change it.

CREATE Options

AS

> **CURRENT**
>
> `INT` type, that is, one of TINYINT, SMALLINT, MEDIUMINT, INT, INTEGER, BIGINT. Can be signed or unsigned. Maximum value is based on the data type. The use of `BIGINT UNSIGNED` with this option extends the possible maximum value from `9223372036854775806` to `18446744073709551614`. Default is `BIGINT`.
>
> **< 11.5**
>
> The `AS` option is not available.

INCREMENT

Increment to use for values. May be negative. Setting an increment of `0` causes the sequence to use the value of the auto_increment_increment system variable at the time of creation, which is always a positive number. (see MDEV-16035). Default `1`.

MINVALUE

Minimum value for the sequence. From MariaDB 11.5, the parser permits much smaller numbers, such as `-9999999999999999999999999999`, but converts to the minimum permitted for the `INT` type, with a note. Default `1` if `INCREMENT` > `0`, and `-9223372036854775807` (or based on int type) if `INCREMENT` < `0`.

MAXVALUE

Maximum value for sequence. From MariaDB 11.5, the parser permits much larger numbers, such as `9999999999999999999999999999` used in Oracle examples, but converts to the maximum permitted for the `INT` type, with a note. Default `9223372036854775806` (or based on int type) if `INCREMENT` > `0`, and `-1` if `INCREMENT` < `0`.

START

First value the sequence will generate. Default `MINVALUE` if `INCREMENT` > `0`, and `MAX_VALUE` if `INCREMENT` < `0`.

CACHE / NOCACHE

Number of values that should be cached. `0` if no `CACHE`. The underlying table will be updated first time a new sequence number is generated and each time the cache runs out. Default `1000`. FLUSH TABLES, shutting down the server, etc. will discard the cached values, and the next sequence number generated will be according to what's stored in the Sequence object. In effect, this will discard the cached values.

> **Warning:**
>
> Note that setting the cache to `1` from `1000` can make inserts to tables using sequences for default values 2x slower and increase the binary log sizes up to 7x.

CYCLE / NOCYCLE

If `CYCLE` is used, then the sequence should start again from `MINVALUE` after it has run out of values. Default value is `NOCYCLE`.

Constraints on Create Arguments

To be able to create a legal sequence, the following must hold:

- `MAXVALUE` >= start
- `MAXVALUE` > `MINVALUE`
- `START` >= `MINVALUE`
- `MAXVALUE` <= `9223372036854775806` (`LONGLONG_MAX`-1). From MariaDB 11.5, the parser accepts values beyond this, and converts based on the int type.
- `MINVALUE` >= `-9223372036854775807` (`LONGLONG_MIN`+1). From MariaDB 11.5, the parser accepts values beyond this, and converts based on the int type.

> **Warning:**
>
> Note that sequences can't generate the maximum/minimum 64 bit number because of the constraint of `MINVALUE` and `MAXVALUE`.

Atomic DDL

> **CURRENT**
>
> MariaDB supports Atomic DDL and `CREATE SEQUENCE` is atomic.
>
> **< 10.6.1**
>
> MariaDB does **not** support Atomic DDL and `CREATE SEQUENCE` is atomic.

Examples

```
CREATE SEQUENCE s START WITH 100 INCREMENT BY 10;

CREATE SEQUENCE s2 START WITH -100 INCREMENT BY -10;
```

The following statement fails, as the increment conflicts with the defaults:

```
CREATE SEQUENCE s3 START WITH -100 INCREMENT BY 10;
ERROR 4085 (HY000): Sequence 'test.s3' has out of range value for options
```

The sequence can be created by specifying workable minimum and maximum values:

```
CREATE SEQUENCE s3 START WITH -100 INCREMENT BY 10 MINVALUE=-100 MAXVALUE=1000;
```

From MariaDB 11.5:

```
CREATE SEQUENCE s3 AS BIGINT UNSIGNED START WITH 10;
```

Parser accepting larger or smaller values:

```
CREATE OR REPLACE SEQUENCE s AS TINYINT SIGNED
  MINVALUE=-99999999999999999999999999999999
  MAXVALUE=99999999999999999999999999999999
  START WITH 100 INCREMENT BY 10;
Query OK, 0 rows affected, 2 warnings (0.037 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------------+
| Level | Code | Message                                       |
+-------+------+-----------------------------------------------+
| Note  | 1292 | Truncated incorrect INTEGER value: 'MINVALUE' |
| Note  | 1292 | Truncated incorrect INTEGER value: 'MAXVALUE' |
+-------+------+-----------------------------------------------+

SELECT * FROM INFORMATION_SCHEMA.SEQUENCES\G
*************************** 1. row ***************************
       SEQUENCE_CATALOG: def
        SEQUENCE_SCHEMA: test
          SEQUENCE_NAME: s
              DATA_TYPE: tinyint
      NUMERIC_PRECISION: 8
NUMERIC_PRECISION_RADIX: 2
          NUMERIC_SCALE: 0
            START_VALUE: 100
          MINIMUM_VALUE: -127
          MAXIMUM_VALUE: 126
              INCREMENT: 10
           CYCLE_OPTION: 0
```

Cache

Flushing the cache:

```
CREATE OR REPLACE SEQUENCE s START WITH 1 INCREMENT BY 1 MAXVALUE=10 CACHE=5;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          1 |
+------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          2 |
+------------+

FLUSH TABLES s;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          6 |
+------------+

FLUSH TABLES s;

SELECT NEXTVAL(s);
ERROR 4084 (HY000): Sequence 'test.s' has run out
```

Create table with a sequence as a default value

You can use sequences instead of `AUTO_INCREMENT` to generate values for a table:

```
CREATE SEQUENCE s1;
CREATE TABLE t1 (a INT PRIMARY KEY DEFAULT nextval(s1), b INT);
INSERT INTO t1 (b) VALUES(1);
SELINT * FROM t1;
+---+------+
| a | b    |
+---+------+
| 1 |    1 |
+---+------+
```

See Also

- Sequence Overview
- ALTER SEQUENCE
- DROP SEQUENCE
- NEXT VALUE FOR
- PREVIOUS VALUE FOR
- SETVAL(). Set next value for the sequence.
- AUTO INCREMENT
- SHOW CREATE SEQUENCE
- Information Schema SEQUENCES Table

This page is licensed: CC BY-SA / Gnu FDL

DROP SEQUENCE

Syntax

```
DROP [TEMPORARY] SEQUENCE [IF EXISTS] [/*COMMENT TO SAVE*/]
    sequence_name [, sequence_name] ...
```

Description

`DROP SEQUENCE` removes one or more sequences created with CREATE SEQUENCE. You must have the `DROP` privilege for each sequence. MariaDB returns an error indicating by name which non-existing tables it was unable to drop, but it also drops all of the tables in the list that do exist.

Important: When a table is dropped, user privileges on the table are not automatically dropped. See GRANT.

If another connection is using the sequence, a metadata lock is active, and this statement will wait until the lock is released. This is also true for non-transactional tables.

For each referenced sequence, DROP SEQUENCE drops a temporary sequence with that name, if it exists. If it does not exist, and the `TEMPORARY` keyword is not used, it drops a non-temporary sequence with the same name, if it exists. The `TEMPORARY` keyword ensures that a non-temporary sequence will not accidentally be dropped.

Use `IF EXISTS` to prevent an error from occurring for sequences that do not exist. A `NOTE` is generated for each non-existent sequence when using `IF EXISTS`. See SHOW WARNINGS.

`DROP SEQUENCE` requires the DROP privilege.

Notes

`DROP SEQUENCE` only removes sequences, not tables. However, DROP TABLE can remove both sequences and tables.

See Also

- Sequence Overview
- CREATE SEQUENCE
- ALTER SEQUENCE
- DROP TABLE
- Information Schema SEQUENCES Table

This page is licensed: CC BY-SA / Gnu FDL

SEQUENCE Functions

See LASTVAL.

LASTVAL is a synonym for PREVIOUS VALUE FOR, returning the most recent value generated from a sequence in the current connection.

See NEXT VALUE FOR.

Generate the next value of a sequence with NEXT VALUE FOR (ANSI SQL), NEXTVAL() (PostgreSQL), or sequence_name.nextval in Oracle mode.

See NEXTVAL.

NEXTVAL is a synonym for NEXT VALUE FOR, generating the next value of a sequence.

See PREVIOUS VALUE FOR.

Get the most recent value generated from a sequence in the current connection with PREVIOUS VALUE FOR (DB2), LASTVAL() (PostgreSQL), or sequence_name.currval in Oracle mode.

See SETVAL.

Set the next value returned by a sequence with SETVAL(), a PostgreSQL-compatible function extended with is_used and round arguments that never lowers the sequence below its current value.

This page is licensed: CC BY-SA / Gnu FDL

LASTVAL

`LASTVAL` is a synonym for PREVIOUS VALUE for sequence_name.

This page is licensed: CC BY-SA / Gnu FDL

NEXT VALUE FOR

Syntax

```
NEXT VALUE FOR sequence
```

or

```
NEXTVAL(sequence_name)
```

or in Oracle mode (SQL_MODE=ORACLE)

```
sequence_name.nextval
```

`NEXT VALUE FOR` is ANSI SQL syntax while `NEXTVAL()` is PostgreSQL syntax.

Description

Generate next value for a `SEQUENCE`.

- You can greatly speed up `NEXT VALUE` by creating the sequence with the `CACHE` option. If not, every `NEXT VALUE` usage will cause changes in the stored `SEQUENCE` table.
- When using `NEXT VALUE` the value will be reserved at once and will not be reused, except if the `SEQUENCE` was created with `CYCLE`. This means that when you are using `SEQUENCE`s you have to expect gaps in the generated sequence numbers.
- If one updates the `SEQUENCE` with SETVAL() or ALTER SEQUENCE … RESTART, `NEXT VALUE FOR` will notice this and start from the next requested value.
- FLUSH TABLES will close the sequence and the next sequence number generated will be according to what's stored in the `SEQUENCE` object. In effect, this will discard the cached values.
- A server restart (or closing the current connection) also causes a drop of all cached values. The cached sequence numbers are reserved only for the current connection.
- `NEXT VALUE` requires the `INSERT` privilege.
- You can also use `NEXT VALUE FOR sequence` for column `DEFAULT`.

Once the sequence is complete, unless the sequence has been created with the CYCLE attribute (not the default), calling the function will result in Error 4084: Sequence has run out.

Examples

```
CREATE OR REPLACE SEQUENCE s MAXVALUE=2;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          1 |
+------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          2 |
+------------+

SELECT NEXTVAL(s);
ERROR 4084 (HY000): Sequence 'test.s' has run out

ALTER SEQUENCE s MAXVALUE=2 CYCLE;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          1 |
+------------+
```

See Also

- Sequence Overview
- CREATE SEQUENCE

- ALTER SEQUENCE
- PREVIOUS VALUE FOR
- SETVAL(). Set next value for the sequence.
- AUTO_INCREMENT
- Information Schema SEQUENCES Table

This page is licensed: CC BY-SA / Gnu FDL

NEXTVAL

`NEXTVAL` is a synonym for NEXT VALUE for sequence_name.

This page is licensed: CC BY-SA / Gnu FDL

PREVIOUS VALUE FOR

Syntax

```
PREVIOUS VALUE FOR sequence_name
```

or

```
LASTVAL(sequence_name)
```

or in Oracle mode (SQL_MODE=ORACLE)

```
sequence_name.currval
```

`PREVIOUS VALUE FOR` is IBM DB2 syntax while `LASTVAL()` is PostgreSQL syntax.

Description

Gets the most recent value in the current connection generated from a sequence.

- If the sequence has not yet been used by the connection, `PREVIOUS VALUE FOR` returns `NULL` (the same thing applies with a new connection which doesn't see a last value for an existing sequence).
- If a `SEQUENCE` has been dropped and re-created then it's treated as a new `SEQUENCE` and `PREVIOUS VALUE FOR` will return `NULL`.
- Returns `NULL` if the sequence is complete.
- FLUSH TABLES has no effect on `PREVIOUS VALUE FOR`.
- Previous values for all used sequences are stored per connection until connection ends.
- `PREVIOUS VALUE FOR` requires the SELECT privilege.

Examples

```
CREATE SEQUENCE s START WITH 100 INCREMENT BY 10;

SELECT PREVIOUS VALUE FOR s;
+----------------------+
| PREVIOUS VALUE FOR s |
+----------------------+
|                 NULL |
+----------------------+

# The function works for sequences only, if the table is used an error is generated
SELECT PREVIOUS VALUE FOR t;
ERROR 4089 (42S02): 'test.t' is not a SEQUENCE

# Call the NEXT VALUE FOR s:
SELECT NEXT VALUE FOR s;
+------------------+
| NEXT VALUE FOR s |
+------------------+
|              100 |
+------------------+

SELECT PREVIOUS VALUE FOR s;
+----------------------+
| PREVIOUS VALUE FOR s |
+----------------------+
|                  100 |
+----------------------+
```

Now try to start the new connection and check that the last value is still `NULL`, before updating the value in the new connection after the output of the new connection gets current value (110 in the example below). Note that first connection cannot see this change and the result of last value still remains the same (100 in the example above).

```
$ .mysql -uroot test -e"SELECT PREVIOUS VALUE FOR s; SELECT NEXT VALUE FOR s; SELECT PREVIOUS VALUE FOR
s;"
+----------------------+
| PREVIOUS VALUE FOR s |
+----------------------+
|                 NULL |
+----------------------+
+------------------+
| NEXT VALUE FOR s |
+------------------+
|              110 |
+------------------+
+----------------------+
| PREVIOUS VALUE FOR s |
+----------------------+
|                  110 |
+----------------------+
```

Returns `NULL` if the sequence has run out:

```
CREATE OR REPLACE SEQUENCE s MAXVALUE=2;

SELECT NEXTVAL(s), LASTVAL(s);
+------------+------------+
| NEXTVAL(s) | LASTVAL(s) |
+------------+------------+
|          1 |          1 |
+------------+------------+

SELECT NEXTVAL(s), LASTVAL(s);
+------------+------------+
| NEXTVAL(s) | LASTVAL(s) |
+------------+------------+
|          2 |          2 |
+------------+------------+

SELECT NEXTVAL(s), LASTVAL(s);
ERROR 4084 (HY000): Sequence 'test.s' has run out

SELECT LASTVAL(s);
+------------+
| LASTVAL(s) |
+------------+
|       NULL |
+------------+
```

See Also

- Sequence Overview
- CREATE SEQUENCE
- ALTER SEQUENCE
- NEXT VALUE FOR
- SETVAL(). Set next value for the sequence.
- AUTO_INCREMENT
- Information Schema SEQUENCES Table
- Error 4084: Sequence has run out

This page is licensed: CC BY-SA / Gnu FDL

SETVAL

Syntax

```
SETVAL(sequence_name, next_value, [is_used, [round]])
```

Description

Set the next value to be returned for a `SEQUENCE`.

This function is compatible with PostgreSQL syntax, extended with the `round` argument.

If the `is_used` argument is not given or is `1` or `true`, then the next used value will one after the given value. If `is_used` is `0` or `false` then the next generated value will be the given value.

If `round` is used then it will set the `round` value (or the internal cycle count, starting at zero) for the sequence. If `round` is not used, it's assumed to be 0.

`next_value` must be an integer literal.

For `SEQUENCE` tables defined with `CYCLE` (see CREATE SEQUENCE) one should use both `next_value` and `round` to define the next value. In this case the current sequence value is defined to be `round`, `next_value`.

The result returned by `SETVAL()` is `next_value` or NULL if the given `next_value` and `round` is smaller than the current value.

`SETVAL()` will not set the `SEQUENCE` value to a something that is less than its current value. This is needed to ensure that `SETVAL()` is replication-safe. If you want to set the `SEQUENCE` to a smaller number, use ALTER SEQUENCE.

If `CYCLE` is used, first `round` and then `next_value` are compared to see if the value is bigger than the current value.

Internally, in the MariaDB server, `SETVAL()` is used to inform replicas that a `SEQUENCE` has changed value. The replica may get `SETVAL()` statements out of order, but this is ok as only the biggest one will have an effect.

`SETVAL` requires the INSERT privilege.

Examples

```
SELECT setval(foo, 42);           -- Next nextval will return 43
SELECT setval(foo, 42, TRUE);     -- Same as above
SELECT setval(foo, 42, FALSE);    -- Next nextval will return 42
```

`SETVAL` setting higher and lower values on a sequence with an increment of 10:

```
SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|         50 |
+------------+

SELECT SETVAL(s, 100);
+----------------+
| SETVAL(s, 100) |
+----------------+
|            100 |
+----------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|        110 |
+------------+

SELECT SETVAL(s, 50);
+---------------+
| SETVAL(s, 50) |
+---------------+
|          NULL |
+---------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|        120 |
+------------+
```

Example demonstrating `round`:

```
CREATE OR REPLACE SEQUENCE s1
  START WITH 1
  MINVALUE 1
  MAXVALUE 99
  INCREMENT BY 1
  CACHE 20
  CYCLE;

SELECT SETVAL(s1, 99, 1, 0);
+----------------------+
| SETVAL(s1, 99, 1, 0) |
+----------------------+
|                   99 |
+----------------------+

SELECT NEXTVAL(s1);
+-------------+
| NEXTVAL(s1) |
+-------------+
|           1 |
+-------------+
```

The following statement returns `NULL`, as the given `next_value` and `round` is smaller than the current value:

```
SELECT SETVAL(s1, 99, 1, 0);
+----------------------+
| SETVAL(s1, 99, 1, 0) |
+----------------------+
|                 NULL |
+----------------------+

SELECT NEXTVAL(s1);
+-------------+
| NEXTVAL(s1) |
+-------------+
|           2 |
+-------------+
```

Increasing the round from zero to 1 allows `next_value` to be returned:

```
SELECT SETVAL(s1, 99, 1, 1);
+----------------------+
| SETVAL(s1, 99, 1, 1) |
+----------------------+
|                   99 |
+----------------------+

SELECT NEXTVAL(s1);
+-------------+
| NEXTVAL(s1) |
+-------------+
|           1 |
+-------------+
```

See Also

- Sequence Overview
- ALTER SEQUENCE
- CREATE SEQUENCE
- NEXT VALUE FOR
- PREVIOUS VALUE FOR
- Information Schema SEQUENCES Table

- Error 4084: Sequence has run out

This page is licensed: CC BY-SA / Gnu FDL

Temporal Tables

See Application-Time Periods.

Application-time period tables define a time period between two temporal columns, for application-level data versioning.

See Bitemporal Tables.

Bitemporal tables combine system-versioning and application-time periods, versioning data at both levels.

See System-Versioned Tables.

Complete system-versioned tables: WITH SYSTEM VERSIONING syntax, FOR SYSTEM_TIME AS OF/BETWEEN/ALL queries, and ROW_START/ROW_END columns.

This page is licensed: CC BY-SA / Gnu FDL

Application-Time Periods

Extending system-versioned tables, MariaDB supports application-time period tables. Time periods are defined by a range between two temporal columns. The columns must be of the same temporal data type, i.e. DATE, TIMESTAMP or DATETIME (TIME and YEAR are not supported), and of the same width.

Using time periods implicitly defines the two columns as `NOT NULL`. It also adds a constraint to check whether the first value is less than the second value. The constraint is invisible to SHOW CREATE TABLE statements. The name of this constraint is prefixed by the time period name, to avoid conflict with other constraints.

Creating Tables with Time Periods

To create a table with a time period, use a CREATE TABLE statement with the `PERIOD` table option.

```
CREATE TABLE t1(
   name VARCHAR(50),
   date_1 DATE,
   date_2 DATE,
   PERIOD FOR date_period(date_1, date_2));
```

This creates a table with a `time_period` period and populates the table with some basic temporal values.

Examples are available in the MariaDB Server source code, at `mysql-test/suite/period/r/create.result`.

Adding and Removing Time Periods

The ALTER TABLE statement now supports syntax for adding and removing time periods from a table. To add a period, use the `ADD PERIOD` clause:

```
CREATE OR REPLACE TABLE rooms (
 room_number INT,
 guest_name VARCHAR(255),
 checkin DATE,
 checkout DATE
 );

ALTER TABLE rooms ADD PERIOD FOR p(checkin,checkout);
```

To remove a period, use the `DROP PERIOD` clause:

```
ALTER TABLE rooms DROP PERIOD FOR p;
```

Both `ADD PERIOD` and `DROP PERIOD` clauses include an option to handle whether the period already exists:

```
ALTER TABLE rooms ADD PERIOD IF NOT EXISTS FOR p(checkin,checkout);

ALTER TABLE rooms DROP PERIOD IF EXISTS FOR p;
```

Deletion by Portion

You can also remove rows that fall within certain time periods.

When MariaDB executes a `DELETE FOR PORTION` statement, it removes the row:

- When the row period falls completely within the delete period, it removes the row.
- When the row period overlaps the delete period, it shrinks the row, removing the overlap from the first or second row period value.
- When the delete period falls completely within the row period, it splits the row into two rows. The first row runs from the starting row period to the starting delete period. The second runs from the ending delete period to the ending row period.

To test this, first populate the table with some data to operate on:

```
CREATE TABLE t1(
   name VARCHAR(50),
   date_1 DATE,
   date_2 DATE,
   PERIOD FOR date_period(date_1, date_2));

INSERT INTO t1 (name, date_1, date_2) VALUES
    ('a', '1999-01-01', '2000-01-01'),
    ('b', '1999-01-01', '2018-12-12'),
    ('c', '1999-01-01', '2017-01-01'),
    ('d', '2017-01-01', '2019-01-01');

SELECT * FROM t1;
+------+------------+------------+
| name | date_1     | date_2     |
+------+------------+------------+
| a    | 1999-01-01 | 2000-01-01 |
| b    | 1999-01-01 | 2018-12-12 |
| c    | 1999-01-01 | 2017-01-01 |
| d    | 2017-01-01 | 2019-01-01 |
+------+------------+------------+
```

Then, run the `DELETE FOR PORTION` statement:

```
DELETE FROM t1
FOR PORTION OF date_period
    FROM '2001-01-01' TO '2018-01-01';
Query OK, 3 rows affected (0.028 sec)

SELECT * FROM t1 ORDER BY name;
+------+------------+------------+
| name | date_1     | date_2     |
+------+------------+------------+
| a    | 1999-01-01 | 2000-01-01 |
| b    | 1999-01-01 | 2001-01-01 |
| b    | 2018-01-01 | 2018-12-12 |
| c    | 1999-01-01 | 2001-01-01 |
| d    | 2018-01-01 | 2019-01-01 |
+------+------------+------------+
```

Here:

- `a` is unchanged, as the range falls entirely out of the specified portion to be deleted.
- `b`, with values ranging from 1999 to 2018, is split into two rows, 1999 to 2000 and 2018-01 to 2018-12 (i.e. one extra row has been inserted).
- `c`, with values ranging from 1999 to 2017, where only the upper value falls within the portion to be deleted, has been shrunk to 1999 to 2001.
- `d`, with values ranging from 2017 to 2019, where only the lower value falls within the portion to be deleted, has been shrunk to 2018 to 2019.

The `DELETE FOR PORTION` statement has the following restrictions

- The `FROM...TO` clause must be constant.
- Multi-delete is not supported.

If there are `DELETE` or `INSERT` triggers, any matched row is deleted, and then one or two rows are inserted. If the record is deleted completely, nothing is inserted.

Updating by Portion

The UPDATE syntax now supports `UPDATE FOR PORTION`, which modifies rows based on their occurrence in a range:

To test it, first populate the table with some data:

```
TRUNCATE t1;

INSERT INTO t1 (name, date_1, date_2) VALUES
    ('a', '1999-01-01', '2000-01-01'),
    ('b', '1999-01-01', '2018-12-12'),
    ('c', '1999-01-01', '2017-01-01'),
    ('d', '2017-01-01', '2019-01-01');

SELECT * FROM t1;
+------+------------+------------+
| name | date_1     | date_2     |
+------+------------+------------+
| a    | 1999-01-01 | 2000-01-01 |
| b    | 1999-01-01 | 2018-12-12 |
| c    | 1999-01-01 | 2017-01-01 |
| d    | 2017-01-01 | 2019-01-01 |
+------+------------+------------+
```

Then run the update:

```
UPDATE t1 FOR PORTION OF date_period
  FROM '2000-01-01' TO '2018-01-01'
SET name = CONCAT(name,'_original');

SELECT * FROM t1 ORDER BY name;
+------------+------------+------------+
| name       | date_1     | date_2     |
+------------+------------+------------+
| a          | 1999-01-01 | 2000-01-01 |
| b          | 1999-01-01 | 2000-01-01 |
| b          | 2018-01-01 | 2018-12-12 |
| b_original | 2000-01-01 | 2018-01-01 |
| c          | 1999-01-01 | 2000-01-01 |
| c_original | 2000-01-01 | 2017-01-01 |
| d          | 2018-01-01 | 2019-01-01 |
| d_original | 2017-01-01 | 2018-01-01 |
+------------+------------+------------+
```

- `a` is unchanged, as the range falls entirely out of the specified portion to be updated.
- For `b`, with years ranging from 1999 to 2018, two extra rows are inserted, with ranges 1999-01 to 2000-01 and 2018-01 to 2018-12. The original row's period has been shrunk to years 2000 and 2018, and the `name` field has got "_original" appended.
- `c`, with values ranging from 1999 to 2017, where only the upper value falls within the portion to be updated, has been shrunk to 1999 to 2001.
- `d`, with values ranging from 2017 to 2019, where only the lower value falls within the portion to be updated, has been shrunk to 2018 to 2019.
- Original rows affected by the update have "_original" appended to the name.

The `UPDATE FOR PORTION` statement has the following limitations:

- The operation cannot modify the two temporal columns used by the time period.
- The operation cannot reference period values in the `SET` expression.
- `FROM...TO` expressions must be constant.

WITHOUT OVERLAPS

> **Note:**
>
> This clause is available from MariaDB 10.5.3.

`WITHOUT OVERLAPS` allows to create an index specifying that application time periods should not overlap. An index constrained by `WITHOUT OVERLAPS` is required to be either a primary key or a unique index.

Take the following example, an application time period table for a booking system:

```
CREATE OR REPLACE TABLE rooms (
 room_number INT,
 guest_name VARCHAR(255),
 checkin DATE,
 checkout DATE,
 PERIOD FOR p(checkin,checkout)
 );

INSERT INTO rooms VALUES
 (1, 'Regina', '2020-10-01', '2020-10-03'),
 (2, 'Cochise', '2020-10-02', '2020-10-05'),
 (1, 'Nowell', '2020-10-03', '2020-10-07'),
 (2, 'Eusebius', '2020-10-04', '2020-10-06');
```

Our system is not intended to permit overlapping bookings, so the fourth record above should not have been inserted. Using `WITHOUT OVERLAPS` in a unique index (in this case based on a combination of room number and the application time period) allows to specify this constraint in the table definition.

```
CREATE OR REPLACE TABLE rooms (
 room_number INT,
 guest_name VARCHAR(255),
 checkin DATE,
 checkout DATE,
 PERIOD FOR p(checkin,checkout),
 UNIQUE (room_number, p WITHOUT OVERLAPS)
 );

INSERT INTO rooms VALUES
 (1, 'Regina', '2020-10-01', '2020-10-03'),
 (2, 'Cochise', '2020-10-02', '2020-10-05'),
 (1, 'Nowell', '2020-10-03', '2020-10-07'),
 (2, 'Eusebius', '2020-10-04', '2020-10-06');
ERROR 1062 (23000): Duplicate entry '2-2020-10-06-2020-10-04' for key 'room_number'
```

Information Schema

> **Note:**
>
> Information Schema support for application time period tables is available from MariaDB 11.4.

Information Schema contains the following support for application time period tables:

- INFORMATION_SCHEMA.PERIODS view.
- INFORMATION_SCHEMA.KEY_PERIOD_USAGE view.
- Additional columns `IS_SYSTEM_TIME_PERIOD_START` and `IS_SYSTEM_TIME_PERIOD_END` in the INFORMATION_SCHEMA.COLUMNS view.

Further Examples

The implicit change from `NULL` to `NOT NULL`:

```
CREATE TABLE `t2` (
  `id` int(11) DEFAULT NULL,
  `d1` datetime DEFAULT NULL,
  `d2` datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

ALTER TABLE t2 ADD PERIOD FOR p(d1,d2);

SHOW CREATE TABLE t2\G
*************************** 1. row ***************************
       Table: t2
Create Table: CREATE TABLE `t2` (
* //a// is *unchanged*, as the range falls entirely out of the specified portion to be updated.
* For //b//, with years ranging from 1999 to 2018, two extra rows are *inserted*, with ranges 1999-01 to
2000-01 and 2018-01 to 2018-12. The original row's period has been *shrunk* to years 2000 and 2018, and
the _name_ field has got "_original" appended.
* //c//, with values ranging from 1999 to 2017, where only the upper value falls within the portion to be
updated, has been *shrunk* to 1999 to 2001.
* //d//, with values ranging from 2017 to 2019, where only the lower value falls within the portion to be
updated, has been *shrunk* to 2018 to 2019.
* Original rows affected by the update have "_original" appended to the ##name## field.
  `id` int(11) DEFAULT NULL,
  `d1` datetime NOT NULL,
  `d2` datetime NOT NULL,
  PERIOD FOR `p` (`d1`, `d2`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Due to this constraint, trying to add a time period where `NULL` data already exist fails:

```
CREATE OR REPLACE TABLE `t2` (
  `id` int(11) DEFAULT NULL,
  `d1` datetime DEFAULT NULL,
  `d2` datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO t2(id) VALUES(1);

ALTER TABLE t2 ADD PERIOD FOR p(d1,d2);
ERROR 1265 (01000): Data truncated for column 'd1' at row 1
```

See Also

- System-versioned Tables
- Bitemporal Tables

This page is licensed: CC BY-SA / Gnu FDL

Bitemporal Tables

Bitemporal tables are tables that use versioning both at the system and application-time period levels.

Using Bitemporal Tables

To create a bitemporal table, use:

```
CREATE TABLE test.t3 (
   date_1 DATE,
   date_2 DATE,
   row_start TIMESTAMP(6) AS ROW START INVISIBLE,
   row_end TIMESTAMP(6) AS ROW END INVISIBLE,
   PERIOD FOR application_time(date_1, date_2),
   PERIOD FOR system_time(row_start, row_end))
WITH SYSTEM VERSIONING;
```

Note that, while `system_time` here is also a time period, it cannot be used in `DELETE FOR PORTION` or `UPDATE FOR PORTION` statements:

```
DELETE FROM test.t3
FOR PORTION OF system_time
    FROM '2000-01-01' TO '2018-01-01';
ERROR 42000: You have an error in your SQL syntax; check the manual that corresponds
  to your MariaDB server version for the right syntax to use near
  'of system_time from '2000-01-01' to '2018-01-01'' at line 1
```

See Also

- System-versioned Tables
- Application-time Periods

This page is licensed: CC BY-SA / Gnu FDL

System-Versioned Tables

MariaDB supports temporal data tables in the form of system-versioning tables (allowing you to query and operate on historic data, discussed below), application-time periods (allow you to query and operate on a temporal range of data), and bitemporal tables (which combine both system-versioning and application-time periods).

System-Versioned Tables

System-versioned tables store the history of all changes, not only data which is currently applicable. This allows data analysis for any point in time, auditing of changes and comparison of data from different points in time. Typical uses cases are:

- Forensic analysis & legal requirements to store data for N years.
- Data analytics (retrospective, trends etc.), e.g. to get your staff information as of one year ago.
- Point-in-time recovery - recover a table state as of particular point in time.

System-versioned tables were first introduced in the SQL:2011 standard.

> **Note:**
>
> The keyword `SYSTEM` is used in the `SYSTEM VERSIONING` clause. Using `SYSTEM` as an unquoted column name in an `ALTER TABLE` is not permitted by some parser rules. Even though `SYSTEM` is not a fully reserved word in Oracle mode, the `ADD` statement will result in a syntax error.
>
> Always provide `SYSTEM` when using it as a column name: `system`. See the Reserved Words page for additional examples and a detailed explanation.

Creating a System-Versioned Table

The CREATE TABLE syntax has been extended to permit creating a system-versioned table. To be system-versioned, according to SQL:2011, a table must have two generated columns, a period, and a special table option clause:

```
CREATE TABLE t(
   x INT,
   start_timestamp TIMESTAMP(6) GENERATED ALWAYS AS ROW START,
   end_timestamp TIMESTAMP(6) GENERATED ALWAYS AS ROW END,
   PERIOD FOR SYSTEM_TIME(start_timestamp, end_timestamp)
) WITH SYSTEM VERSIONING;
```

In MariaDB, you can use a simplified syntax:

```
CREATE TABLE t (
   x INT
) WITH SYSTEM VERSIONING;
```

In the latter case, no extra columns will be created, and they won't clutter the output of, say, `SELECT * FROM t`. The versioning information will still be stored, and it can be accessed via the pseudo-columns `ROW_START` and `ROW_END`:

```
SELECT x, ROW_START, ROW_END FROM t;
```

Adding or Removing System Versioning To/From a Table

An existing table can be altered to enable system versioning for it.

```
CREATE TABLE t(
  x INT
);
```

```
ALTER TABLE t ADD SYSTEM VERSIONING;
```

```
SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE `t` (
  `x` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 WITH SYSTEM VERSIONING
```

Similarly, system versioning can be removed from a table:

```
ALTER TABLE t DROP SYSTEM VERSIONING;
```

```
SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE `t` (
  `x` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

You can also add system versioning with all columns created explicitly:

```
ALTER TABLE t ADD COLUMN ts TIMESTAMP(6) GENERATED ALWAYS AS ROW START,
              ADD COLUMN te TIMESTAMP(6) GENERATED ALWAYS AS ROW END,
              ADD PERIOD FOR SYSTEM_TIME(ts, te),
              ADD SYSTEM VERSIONING;
```

```
SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE `t` (
  `x` int(11) DEFAULT NULL,
  `ts` timestamp(6) GENERATED ALWAYS AS ROW START,
  `te` timestamp(6) GENERATED ALWAYS AS ROW END,
  PERIOD FOR SYSTEM_TIME (`ts`, `te`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 WITH SYSTEM VERSIONING
```

CURRENT

It is possible to convert a versioned table from implicit to explicit `row_start` / `row_end` columns. Note that, in order to do any `ALTER` on a system versioned table, system_versioning_alter_history must be set to `KEEP`.

```
CREATE OR REPLACE TABLE t1 (x INT) WITH SYSTEM VERSIONING;

SET system_versioning_alter_history = keep;

ALTER TABLE t1 ADD COLUMN rs TIMESTAMP(6) AS ROW START,
  ADD COLUMN re TIMESTAMP(6) AS ROW END, ADD PERIOD FOR SYSTEM_TIME (rs,re)
```

< 11.7

It is **not** possible to convert a versioned table from implicit to explicit `row_start` / `row_end` columns. Doing so results in a duplicate row error:

```
CREATE OR REPLACE TABLE t1 (x INT) WITH SYSTEM VERSIONING;

SET system_versioning_alter_history = keep;

ALTER TABLE t1 ADD COLUMN rs TIMESTAMP(6) AS ROW START,
  ADD COLUMN re TIMESTAMP(6) AS ROW END, ADD PERIOD FOR SYSTEM_TIME (rs,re);

ERROR 4134 (HY000): Duplicate ROW START column `rs`
```

Inserting Data

When data is inserted into a system-versioned table, it is given a *row_start* value of the current timestamp, and a *row_end* value of FROM_UNIXTIME (`2147483647.999999`). The current timestamp can be adjusted by setting the timestamp system variable:

```
SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2022-10-24 23:09:38 |
+---------------------+

INSERT INTO t VALUES(1);

SET @@timestamp = UNIX_TIMESTAMP('2033-10-24');

INSERT INTO t VALUES(2);

SET @@timestamp = default;

INSERT INTO t VALUES(3);

SELECT a,row_start,row_end FROM t;
+------+----------------------------+----------------------------+
| a    | row_start                  | row_end                    |
+------+----------------------------+----------------------------+
|    1 | 2022-10-24 23:09:38.951347 | 2038-01-19 05:14:07.999999 |
|    2 | 2033-10-24 00:00:00.000000 | 2038-01-19 05:14:07.999999 |
|    3 | 2022-10-24 23:09:38.961857 | 2038-01-19 05:14:07.999999 |
+------+----------------------------+----------------------------+
```

Updating Data

When an `UPDATE` statement modifies a row in a system-versioned table, the server writes a new history row capturing the previous state of the row.

A history row is created for every `UPDATE` statement that includes a versioned column in its `SET` list, regardless of whether any column value actually changes. Running `UPDATE t SET x = x` against an existing row of a versioned table still produces a new history row — what determines history-row creation is the presence of a versioned column in the `SET` list, not whether the supplied values differ from the stored values. `UPDATE` statements whose `SET` list references only columns declared `WITHOUT SYSTEM VERSIONING` do not produce a history row.

```
CREATE OR REPLACE TABLE t (x INT) WITH SYSTEM VERSIONING;
INSERT INTO t VALUES (1);

-- This UPDATE changes nothing about the stored value of x,
-- but still produces a new history row:
UPDATE t SET x = 1;

SELECT x, ROW_START, ROW_END FROM t FOR SYSTEM_TIME ALL;
```

This statement-level behavior applies to the default timestamp-based versioning. For transaction-precise versioning, history generation is handled at the storage-engine level by InnoDB's MVCC and is tied to actual row-level changes.

Reasoning for the behavior: Logging updates that do not change data values ensures a complete audit trail and maintains consistent server behavior. In MariaDB, an `UPDATE` statement executes its associated triggers regardless of whether the data values actually change. Recording a historical row for these statements ensures that system-versioned tables follow this standard query logic. Furthermore, capturing the exact timestamp of these updates provides vital context for auditing and troubleshooting. It proves that an application explicitly targeted and processed a specific record at that exact moment—valuable information that would be lost if the server silently ignored the event.

See MDEV-23446 for related discussion.

Querying Historical Data

SELECT

To query the historical data one uses the clause `FOR SYSTEM_TIME` directly after the table name (before the table alias, if any). SQL:2011 provides three syntactic extensions:

- `AS OF` is used to see the table as it was at a specific point in time in the past:

```
SELECT * FROM t FOR SYSTEM_TIME AS OF TIMESTAMP'2016-10-09 08:07:06';
```

- `BETWEEN start AND end` will show all rows that were visible at any point between two specified points in time. It works inclusively, a row visible exactly at start or exactly at end will be shown too.

```
SELECT * FROM t FOR SYSTEM_TIME BETWEEN (NOW() - INTERVAL 1 YEAR) AND NOW();
```

- `FROM start TO end` will also show all rows that were visible at any point between two specified points in time, including start, but excluding end.

```
SELECT * FROM t FOR SYSTEM_TIME FROM '2016-01-01 00:00:00' TO '2017-01-01 00:00:00';
```

Additionally, MariaDB implements a non-standard extension.

- `ALL` will show all rows, historical and current.

```
SELECT * FROM t FOR SYSTEM_TIME ALL;
```

CURRENT

If the `FOR SYSTEM_TIME` clause is not used, the table shows the *current* data. This is usually the same as if you had specified `FOR SYSTEM_TIME AS OF CURRENT_TIMESTAMP`, unless you've adjusted the *row_start* value:

```
CREATE OR REPLACE TABLE t (a int) WITH SYSTEM VERSIONING;

SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2022-10-24 23:43:37 |
+---------------------+

INSERT INTO t VALUES (1);

SET @@timestamp = UNIX_TIMESTAMP('2033-03-03');

INSERT INTO t VALUES (2);

DELETE FROM t;

SET @@timestamp = default;

SELECT a, row_start, row_end FROM t FOR SYSTEM_TIME ALL;
+------+----------------------------+----------------------------+
| a    | row_start                  | row_end                    |
+------+----------------------------+----------------------------+
|    1 | 2022-10-24 23:43:37.192725 | 2033-03-03 00:00:00.000000 |
|    2 | 2033-03-03 00:00:00.000000 | 2033-03-03 00:00:00.000000 |
+------+----------------------------+----------------------------+
2 rows in set (0.000 sec)

SELECT a, row_start, row_end FROM t FOR SYSTEM_TIME AS OF CURRENT_TIMESTAMP;
+------+----------------------------+----------------------------+
| a    | row_start                  | row_end                    |
+------+----------------------------+----------------------------+
|    1 | 2022-10-24 23:43:37.192725 | 2033-03-03 00:00:00.000000 |
+------+----------------------------+----------------------------+
1 row in set (0.000 sec)

SELECT a, row_start, row_end FROM t;
Empty set (0.001 sec)
```

< 10.11

If the `FOR SYSTEM_TIME` clause is not used, the table shows the *current* data. This is usually the same as if you had specified `FOR SYSTEM_TIME AS OF CURRENT_TIMESTAMP`, unless you've adjusted the *row_start* value (only possible by setting the secure_timestamp variable):

```
CREATE OR REPLACE TABLE t (a int) WITH SYSTEM VERSIONING;

SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2022-10-24 23:43:37 |
+---------------------+

INSERT INTO t VALUES (1);

SET @@timestamp = UNIX_TIMESTAMP('2033-03-03');

INSERT INTO t VALUES (2);

DELETE FROM t;

SET @@timestamp = default;

SELECT a, row_start, row_end FROM t FOR SYSTEM_TIME ALL;
+------+----------------------------+----------------------------+
| a    | row_start                  | row_end                    |
+------+----------------------------+----------------------------+
|    1 | 2022-10-24 23:43:37.192725 | 2033-03-03 00:00:00.000000 |
|    2 | 2033-03-03 00:00:00.000000 | 2033-03-03 00:00:00.000000 |
+------+----------------------------+----------------------------+
2 rows in set (0.000 sec)

SELECT a, row_start, row_end FROM t FOR SYSTEM_TIME AS OF CURRENT_TIMESTAMP;
+------+----------------------------+----------------------------+
| a    | row_start                  | row_end                    |
+------+----------------------------+----------------------------+
|    1 | 2022-10-24 23:43:37.192725 | 2033-03-03 00:00:00.000000 |
+------+----------------------------+----------------------------+
1 row in set (0.000 sec)

SELECT a, row_start, row_end FROM t;
Empty set (0.001 sec)
```

Views and Subqueries

When a system-versioned table is used in a view or in a subquery in the from clause, `FOR SYSTEM_TIME` can be used directly in the view or subquery body, or (non-standard) applied to the whole view when it's being used in a `SELECT`:

```
CREATE VIEW v1 AS SELECT * FROM t FOR SYSTEM_TIME AS OF TIMESTAMP'2016-10-09 08:07:06';
```

Or:

```
CREATE VIEW v1 AS SELECT * FROM t;
SELECT * FROM v1 FOR SYSTEM_TIME AS OF TIMESTAMP'2016-10-09 08:07:06';
```

Use in Replication and Binary Logs

Tables that use system-versioning implicitly add the `row_end` column to the Primary Key. While this is generally not an issue for most use cases, it can lead to problems when re-applying write statements from the binary log or in replication environments, where a primary retries an SQL statement on the replica.

Specifically, these writes include a value on the `row_end` column containing the timestamp from when the write was initially made. The re-occurrence of the Primary Key with the old system-versioning columns raises an error due to the duplication.

To mitigate this with MariaDB Replication, set the secure_timestamp system variable to `YES` on the replica. When set, the replica uses its own system clock when applying to the row log, meaning that the primary can retry as many times as needed without causing a conflict. The retries generate new historical rows with new values for the `row_start` and `row_end` columns.

Transaction-Precise History in InnoDB

A point in time when a row was inserted or deleted does not necessarily mean that a change became visible at the same moment. With transactional tables, a row might have been inserted in a long transaction, and became visible hours after it was inserted.

For some applications — for example, when doing data analytics on one-year-old data — this distinction does not matter much. For others — forensic analysis — it might be crucial.

MariaDB supports transaction-precise history (only for the InnoDB storage engine) that allows seeing the data exactly as it would've been seen by a new connection doing a `SELECT` at the specified point in time — rows inserted *before* that point, but committed *after* will not be shown.

To use transaction-precise history, InnoDB needs to remember not timestamps, but transaction identifier per row. This is done by creating generated columns as `BIGINT UNSIGNED`, not `TIMESTAMP(6)`:

```
CREATE TABLE t(
   x INT,
   start_trxid BIGINT UNSIGNED GENERATED ALWAYS AS ROW START,
   end_trxid BIGINT UNSIGNED GENERATED ALWAYS AS ROW END,
   PERIOD FOR SYSTEM_TIME(start_trxid, end_trxid)
) WITH SYSTEM VERSIONING;
```

These columns must be specified explicitly, but they can be made INVISIBLE to avoid cluttering `SELECT *` output.

Note that if you are using an engine that does not support system versioning with transaction ids, you will get an error like "`start_trxid` must be of type `TIMESTAMP(6)` for system-versioned table `t`".

When one uses transaction-precise history, one can optionally use transaction identifiers in the `FOR SYSTEM_TIME` clause:

```
SELECT * FROM t FOR SYSTEM_TIME AS OF TRANSACTION 12345;
```

This shows the data, exactly as it was seen by the transaction with the identifier 12345.

Data for this feature is stored in the mysql.transaction_registry table.

Limitations

`PARTITION BY SYSTEM_TIME` is not supported when using transaction-precise system versioning. Transaction-precise history tracks row versions using transaction identifiers instead of timestamps, so, `SYSTEM_TIME` cannot be used to partition transaction-precise system history. If partitioning is required, use the default timestamp-based versioning.

Storing the History Separately

When the history is stored together with the current data, it increases the size of the table, so current data queries — table scans and index searches — will take more time, because they will need to skip over historical data. If most queries on that table use only current data, it might make sense to store the history separately, to reduce the overhead from versioning.

This is done by partitioning the table by `SYSTEM_TIME`. Because of the partition pruning optimization, all current data queries will only access one partition, the one that stores current data.

This example shows how to create such a partitioned table:

```
CREATE TABLE t (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME (
    PARTITION p_hist HISTORY,
    PARTITION p_cur CURRENT
  );
```

In this example, all history will be stored in the partition `p_hist` while all current data will be in the partition `p_cur`. The table must have exactly one current partition and at least one historical partition.

Partitioning by `SYSTEM_TIME` also supports automatic partition rotation. You can rotate historical partitions by time or by size. This example shows how to rotate partitions by size:

```
CREATE TABLE t (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME LIMIT 100000 (
    PARTITION p0 HISTORY,
    PARTITION p1 HISTORY,
    PARTITION pcur CURRENT
  );
```

MariaDB starts writing history rows into partition `p0`, and at the end of the statement that wrote the 100000th row, MariaDB will switch to partition `p1`. There are only two historical partitions, so when `p1` overflows, MariaDB will issue a warning, but will continue writing into it.

Note on `LIMIT` precision: When using `LIMIT` to specify partition rotation, the specified value is just an estimate of the partition size. Although the partition's intended size is specified by the `LIMIT` setting, the actual number of rows may differ. This occurs because partition rotation is triggered during internal operations rather than when the specified row count is met. Partitions may therefore contain significantly more or fewer rows than the set `LIMIT` value.

Similarly, one can rotate partitions by time:

```
CREATE TABLE t (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME INTERVAL 1 WEEK (
    PARTITION p0 HISTORY,
    PARTITION p1 HISTORY,
    PARTITION p2 HISTORY,
    PARTITION pcur CURRENT
  );
```

This means that the history for the first week after the table was created will be stored in `p0`. The history for the second week — in `p1`, and all later history will go into `p2`. One can see the exact rotation time for each partition in the INFORMATION_SCHEMA.PARTITIONS table.

It is possible to combine partitioning by `SYSTEM_TIME` and subpartitions:

```
CREATE TABLE t (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME
    SUBPARTITION BY KEY (x)
    SUBPARTITIONS 4 (
    PARTITION ph HISTORY,
    PARTITION pc CURRENT
  );
```

Default Partitions

> **Note:**
>
> Default Partitions are available from MariaDB 10.5.

Since partitioning by current and historical data is such a typical use case, it is possible to use a simplified statement to do so. Instead of the following statement:

```
CREATE TABLE t (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME (
    PARTITION p0 HISTORY,
    PARTITION pn CURRENT
);
```

You can use:

```
CREATE TABLE t (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME;
```

You can also specify the number of partitions, which is useful if you want to rotate history by time, for example:

```
CREATE TABLE t (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME
    INTERVAL 1 MONTH
    PARTITIONS 12;
```

Specifying the number of partitions without specifying a rotation condition results in a warning:

```
CREATE OR REPLACE TABLE t (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME PARTITIONS 12;
Query OK, 0 rows affected, 1 warning (0.518 sec)

Warning (Code 4115): Maybe missing parameters: no rotation condition for multiple HISTORY partitions.
```

While specifying only one partition results in an error:

```
CREATE OR REPLACE TABLE t (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME PARTITIONS 1;
ERROR 4128 (HY000): Wrong partitions for `t`: must have at least one HISTORY and exactly one last CURRENT
```

Automatically Creating Partitions

> **Note:**
>
> Creating partitions with `AUTO` is available from MariaDB 10.9.1.

The `AUTO` keyword can be used to automatically create history partitions:

```
CREATE TABLE t1 (x INT) WITH SYSTEM VERSIONING
    PARTITION BY SYSTEM_TIME INTERVAL 1 HOUR AUTO;

CREATE TABLE t1 (x INT) WITH SYSTEM VERSIONING
   PARTITION BY SYSTEM_TIME INTERVAL 1 MONTH
   STARTS '2021-01-01 00:00:00' AUTO PARTITIONS 12;

CREATE TABLE t1 (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME LIMIT 1000 AUTO;
```

Or with explicit partitions:

```
CREATE TABLE t1 (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME INTERVAL 1 HOUR AUTO
  (PARTITION p0 HISTORY, PARTITION pn CURRENT);
```

To disable or enable auto-creation, one can use `ALTER TABLE` by adding or removing `AUTO` from the partitioning specification:

```
CREATE TABLE t1 (x INT) WITH SYSTEM VERSIONING
  PARTITION BY SYSTEM_TIME INTERVAL 1 HOUR AUTO;

# Disables auto-creation:
ALTER TABLE t1 PARTITION BY SYSTEM_TIME INTERVAL 1 HOUR;

# Enables auto-creation:
ALTER TABLE t1 PARTITION BY SYSTEM_TIME INTERVAL 1 HOUR AUTO;
```

If the rest of the partitioning specification is identical to `CREATE TABLE`, no repartitioning will be done (for details see MDEV-27328).

> Partitioning by SYSTEM_TIME is not available for tables that use transaction-precise history. If partitioning is required, use the default timestamp-based system versioning instead.

Removing Old History

Because it stores all the history, a system-versioned table might grow very large over time. There are many options to trim down the space and remove the old history.

One can completely drop the versioning from the table and add it back again, this deletes all the history:

```
ALTER TABLE t DROP SYSTEM VERSIONING;
ALTER TABLE t ADD SYSTEM VERSIONING;
```

It might be a rather time-consuming operation, though, as the table needs to be rebuilt, possibly twice (depending on the storage engine).

Another option would be to use partitioning and drop some of historical partitions:

```
ALTER TABLE t DROP PARTITION p0;
```

> **Warning:**
>
> You cannot drop a current partition or the only historical partition.

As a third option; you can use a variant of the DELETE statement to prune the history:

```
DELETE HISTORY FROM t;
```

Or only old history up to a specific point in time:

```
DELETE HISTORY FROM t BEFORE SYSTEM_TIME '2016-10-09 08:07:06';
```

Or to a specific transaction (with `BEFORE SYSTEM_TIME TRANSACTION xxx`).

To protect the integrity of the history, this statement requires a special DELETE HISTORY privilege.

Currently, using the `DELETE HISTORY` statement with a `BEFORE SYSTEM_TIME` greater than the `ROW_END` of the active records (as a TIMESTAMP, this has a maximum value of `'2038-01-19 03:14:07'` UTC) results in the historical records being dropped, and the active records being deleted and moved to history. See MDEV-25468.

The TRUNCATE TABLE statement drops all historical records from a system-versioned table.

Historic data is protected from `TRUNCATE` statements, as per the SQL standard, and an Error 4137 is instead raised:

```
TRUNCATE t;
ERROR 4137 (HY000): System-versioned tables do not support TRUNCATE TABLE
```

Excluding Columns From Versioning

Another MariaDB extension allows one to version only a subset of columns in a table. This is useful, for example, if you have a table with user information that should be versioned, but one column is, let's say, a login counter that is incremented often and is not interesting to version. Such a column can be excluded from versioning by declaring it `WITHOUT VERSIONING`

```
CREATE TABLE t (
   x INT,
   y INT WITHOUT SYSTEM VERSIONING
) WITH SYSTEM VERSIONING;
```

A column can also be declared `WITH VERSIONING`, that will automatically make the table versioned. The statement below is equivalent to the one above:

```
CREATE TABLE t (
   x INT WITH SYSTEM VERSIONING,
   y INT
);
```

System Variables

A number of system variables are related to system-versioned tables:

system_versioning_alter_history

- Description: SQL:2011 does not allow ALTER TABLE on system-versioned tables. When this variable is set to `ERROR`, an attempt to alter a system-versioned table will result in an error. When this variable is set to `KEEP`, `ALTER TABLE` will be allowed, but the history will become incorrect — querying historical data will show the new table structure. This mode is still useful, for example, when adding new columns to a table. Note that if historical data contains or would contain nulls, attempting to `ALTER` these columns to be `NOT NULL` will return an error (or warning if strict_mode is not set).
- Command line: `--system-versioning-alter-history=value`
- Scope: Global, Session
- Dynamic: Yes
- Type: Enum
- Default Value: `ERROR`
- Valid Values: `ERROR`, `KEEP`

`system_versioning_asof`

- Description: If set to a specific timestamp value, an implicit `FOR SYSTEM_TIME AS OF` clause will be applied to all queries. This is useful if one wants to do many queries for history at the specific point in time. Set it to `'DEFAULT'` to restore the default behavior. Has no effect on DML, so queries such as INSERT .. SELECT and REPLACE .. SELECT need to state AS OF explicitly.

Note: You need to use quotes around the name `'DEFAULT'` when setting the session value, unquoted literal `DEFAULT` will restore the current global value instead.

- Command line: None
- Scope: Global, Session
- Dynamic: Yes
- Type: Varchar
- Default Value: `DEFAULT`

system_versioning_innodb_algorithm_simple

- Description: Never fully implemented and removed in the following release.
- Command line: `--system-versioning-innodb-algorithm-simple[={0|1}]`
- Scope: Global, Session
- Dynamic: Yes
- Type: Boolean
- Default Value: `ON`
- Introduced: MariaDB 10.3.4
- Removed: MariaDB 10.3.5

system_versioning_insert_history

- Description: Allows direct inserts into ROW_START and ROW_END columns if secure_timestamp allows changing timestamp.
- Command line: `--system-versioning-insert-history[={0|1}]`
- Scope: Global, Session
- Dynamic: Yes
- Type: Boolean
- Default Value: `OFF`
- Introduced: MariaDB 10.11.0

Limitations

- Versioning clauses cannot be applied to generated (virtual and persistent) columns.
- mariadb-dump did not read historical rows from versioned tables, and so historical data would not be backed up. Also, a restore of the timestamps would not be possible as they cannot be defined by an insert/a user. From MariaDB 10.11, use the `-H` or `--dump-history` options to include the history.
- Transaction-precise system-versioned tables do not support partitioning by `SYSTEM_TIME`. See Transaction-Precise History in InnoDB for details.

See Also

- Application-Time Periods
- Bitemporal Tables
- mysql.transaction_registry Table
- MariaDB Temporal Tables (video)

This page is licensed: CC BY-SA / Gnu FDL

Vectors

See Vector Overview.

Official MariaDB Vector reference: VECTOR(n) data type, VECTOR INDEX (M, DISTANCE=euclidean|cosine), VEC_FromText() inserts, VEC_DISTANCE() queries.

See CREATE TABLE with Vectors.

Create tables optimized for vector storage. Learn to define columns with the VECTOR data type and configure vector indexes for similarity search.

See Vector System Variables.

System variables that control MariaDB's vector storage and similarity-search features.

See Vector Framework Integrations.

MariaDB Vector integrations with popular AI and application frameworks.

See Vector Functions.

Explore vector functions. This section details SQL functions for manipulating and querying vector data types, enabling efficient similarity search and AI/ML applications within your database.

See VECTOR.

The VECTOR data type, available from MariaDB 11.7.1, for storing fixed- length numeric arrays used in vector search.

See Optimizing Hybrid Search Query with Reciprocal Rank Fusion (RRF).

Combine full-text (keyword) and vector search using Reciprocal Rank Fusion (RRF) for higher-quality hybrid search results.

This page is licensed: CC BY-SA / Gnu FDL

Vector Overview

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

MariaDB Vector is a feature that allows MariaDB Server to perform as a relational vector database. Vectors generated by an AI model can be stored and searched in MariaDB.

MariaDB Vector was introduced in MariaDB Community Server 11.7, a rolling release, and is generally available from MariaDB Community Server 11.8 onwards, the first long-term support (LTS) release to include it.

In MariaDB Enterprise Server, vector search was backported to the 11.4 release series and is available from MariaDB Enterprise Server 11.4.5-3 onwards. It is also included in MariaDB Enterprise Server 11.8, which is generally available from 11.8.3-1 onwards.

The initial implementation uses the modified HNSW[37] algorithm for searching in the vector index (to solve the so-called Approximate Nearest Neighbor problem), and defaults to Euclidean distance. Concurrent reads/writes and all transaction isolation levels are supported.

MariaDB uses `int16` for indexes, which gives 15 bits to store the value, rather than 10 bits for float16.

Creating

Vectors can be defined using `VECTOR INDEX` for the index definition, and using the VECTOR data type in the CREATE TABLE statement.

```
CREATE TABLE v (
     id INT PRIMARY KEY,
     v VECTOR(5) NOT NULL,
     VECTOR INDEX (v)
);
```

The distance function used to build the vector index can be `euclidean` (the default) or `cosine`. An additional option, `M`, can be used to configure the vector index. Larger values mean slower `SELECT` and `INSERT` statements, larger index size and higher memory consumption but more accurate results. The valid range is from `3` to `200`.

```
CREATE TABLE embeddings (
        doc_id BIGINT UNSIGNED PRIMARY KEY,
        embedding VECTOR(1536) NOT NULL,
        VECTOR INDEX (embedding) M=8 DISTANCE=cosine
);
```

Inserting

Vector columns store 32-bit IEEE 754 floating point numbers.

```
INSERT INTO v VALUES
     (1, x'e360d63ebe554f3fcdbc523f4522193f5236083d'),
     (2, x'f511303f72224a3fdd05fe3eb22a133ffae86a3f'),
     (3,x'f09baa3ea172763f123def3e0c7fe53e288bf33e'),
     (4,x'b97a523f2a193e3eb4f62e3f2d23583e9dd60d3f'),
     (5,x'f7c5df3e984b2b3e65e59d3d7376db3eac63773e'),
     (6,x'de01453ffa486d3f10aa4d3fdd66813c71cb163f'),
     (7,x'76edfc3e4b57243f10f8423fb158713f020bda3e'),
     (8,x'56926c3fdf098d3e2c8c5e3d1ad4953daa9d0b3e'),
     (9,x'7b713f3e5258323f80d1113d673b2b3f66e3583f'),
     (10,x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e');
```

Alternatively, you can use `VEC_FromText()` function:

```
INSERT INTO v VALUES
  (1,Vec_FromText('[0.418708,0.809902,0.823193,0.598179,0.0332549]')),
  (2,Vec_FromText('[0.687774,0.789588,0.496138,0.57487,0.917617]')),
  (3,Vec_FromText('[0.333221,0.962687,0.467263,0.448235,0.475671]')),
  (4,Vec_FromText('[0.822185,0.185643,0.683452,0.211072,0.554056]')),
  (5,Vec_FromText('[0.437057,0.167281,0.0770977,0.428638,0.241591]')),
  (6,Vec_FromText('[0.76956,0.926895,0.803376,0.0157961,0.589042]')),
  (7,Vec_FromText('[0.493999,0.641957,0.761598,0.94276,0.425865]')),
  (8,Vec_FromText('[0.924108,0.275466,0.0543329,0.0731585,0.136344]')),
  (9,Vec_FromText('[0.186956,0.69666,0.0356002,0.668875,0.84722]')),
  (10,Vec_FromText('[0.415294,0.609278,0.426765,0.988832,0.475556]'));
```

Querying

For vector indexes built with the `euclidean` function, VEC_DISTANCE_EUCLIDEAN can be used. It calculates a Euclidean (L2) distance between two points:

```
SELECT id FROM v ORDER BY
  VEC_DISTANCE_EUCLIDEAN(v, x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e');
+----+
| id |
+----+
| 10 |
|  7 |
|  3 |
|  9 |
|  2 |
|  1 |
|  5 |
|  4 |
|  6 |
|  8 |
+----+
```

Most commonly, this kind of query is done with a limit, for example to return vectors that are closest to a given vector, such as from a user search query, image or a song fragment:

```
SELECT id FROM v
  ORDER BY VEC_DISTANCE_EUCLIDEAN(v, x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e')
  LIMIT 2;
+----+
| id |
+----+
| 10 |
|  7 |
+----+
```

For vector indexes built with the `cosine` function, VEC_DISTANCE_COSINE can be used. It calculates a Cosine distance between two vectors:

```
SELECT VEC_DISTANCE_COSINE(VEC_FROMTEXT('[1,2,3]'), VEC_FROMTEXT('[3,5,7]'));
```

The VEC_DISTANCE function is a generic function that behaves either as VEC_DISTANCE_EUCLIDEAN or VEC_DISTANCE_COSINE, depending on the underlying index type:

```
SELECT id FROM v
  ORDER BY VEC_DISTANCE(v, x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e');
+----+
| id |
+----+
| 10 |
|  7 |
|  3 |
|  9 |
|  2 |
|  1 |
|  5 |
|  4 |
|  6 |
|  8 |
+----+
```

Note:

There is no function for *dot product* (also called *inner product*) distance available in many other vector databases. Dot product is not a proper distance measure (for example, vector's closest match is not necessarily itself) and is only used for performance reasons, because it is often faster than cosine or euclidean and produces the same results if vectors are normalized. In MariaDB optimized implementation euclidean and cosine measures are the fastest, and dot product, if implemented, would not provide any performance benefits. Use euclidean or cosine (they are equally fast) for normalized vectors.

Using the Vector Index Efficiently

The optimizer uses the vector index only when the `ORDER BY` is the literal `VEC_DISTANCE_*(column, vector)` call (or its alias) sorted ascending, together with a `LIMIT`. Two common patterns defeat the index and fall back to a full table scan.

Wrapping the distance in an expression. To sort by a similarity score (for example `1 - cosine distance`), compute the score in an outer query and keep the inner `ORDER BY` on the bare distance:

```
SELECT t.id, 1 - t.distance AS score FROM (
    SELECT id, VEC_DISTANCE_COSINE(embedding, @query) AS distance
    FROM documents ORDER BY distance LIMIT 10
) AS t ORDER BY score DESC;
```

Filtering by a distance threshold. A bare `WHERE VEC_DISTANCE(...) < threshold` with no `ORDER BY ... LIMIT` is a range predicate that the index cannot drive, so it falls back to a full table scan. To use the index, retrieve an

indexed top-K and apply the threshold to it, for example in a subquery:

```
SELECT * FROM (
    SELECT content, VEC_DISTANCE_COSINE(embedding, @query) AS distance
    FROM documents ORDER BY distance LIMIT 100
) AS t WHERE t.distance < 0.5;
```

The `LIMIT` is a hard cap on how many rows the index returns *before* the threshold is applied, so choose it generously if many rows may match, otherwise qualifying rows beyond the limit are silently dropped. (Adding the `WHERE` directly to an `ORDER BY ... LIMIT` query also uses the index, but with the same hard cap.)

System Variables

There are a number of system variables used for vectors. See Vector System Variables.

Vector Framework Integrations

MariaDB Vector is integrated in several frameworks, see Vector Framework Integrations.

What is a Vector?

https://www.youtube.com/shorts/VrG8H53KJZY

What exactly is a vector in AI and RAG (1 minute • 2026)

> **Note:**
>
> **Video summary**
>
> - A vector (an *embedding*) is an ordered list of numbers.
> - AI models map content (like text) into vectors.
> - Similar meanings end up close together in vector space.
> - RAG and semantic search retrieve relevant items by finding the nearest vectors.

See Also

- MariaDB Vector project page — feature summary, benchmark results, talks, tutorials, and FAQ
- Big Vector Search Benchmark: 10 databases comparison (blog post • 2026) — MariaDB 11.8 and 12.3 measured against pgvector, pgvectorscale, Qdrant, Milvus, Weaviate, OpenSearch and others on one million OpenAI embeddings
- Vector indexes, large server, dbpedia-openai dataset: MariaDB, Qdrant and pgvector (blog post • 2025) — independent third-party benchmark by Mark Callaghan (Small Datum)
- MariaDB Vector: How it works (blog posts • 2026) — Part I, Part II, Part III, Part IV
- Everything you need to know to start building apps with AI and RAG (video • 40 minutes • 2026)
- What exactly is a vector in AI and RAG? (video • 1 minute • 2026)
- Get to know MariaDB's Rocket-Fast Native Vector Search - Sergei Golubchyk (video • 37 minutes • 2025)
- MariaDB Vector, a new Open Source vector database that you are already familiar with - Sergei Golubchik (video • 27 minutes • 2024)
- AI first applications with MariaDB Vector - Vicentiu Ciorbaru (video • 22 minutes • 2025)
- MariaDB Vector: A storage engine for LLMs - Kaj Arnö and Jonah Harris (video • 12 minutes • 2025)
- Try RAG with MariaDB Vector on your own MariaDB data! (blog post • 5 minutes • 2024)
  - The post, written by Robert Silén, explains how to build a Retrieval-Augmented Generation (RAG) system using MariaDB's native vector storage. It covers:
    1. Preparation: Setting up a MariaDB 11.7+ environment and creating a table with the `VECTOR` data type.
    2. Indexing: Using OpenAI's embedding model to vectorize documentation and store it in MariaDB.

3. Search & Generation: Performing a nearest-neighbor search (`VEC_DISTANCE_EUCLIDEAN`) to find relevant context for a user's question and feeding that context into an LLM for an accurate response.

- MariaDB Knowledge Base RAG demo (runnable demo • Python)
- MariaDB RAG demo with OpenAI (runnable demo • Java)
- MariaDB skills for AI agents — includes `mariadb-vector/SKILL.md` , with usage and tuning guidance for Claude Code and other AI agents

This page is licensed: CC BY-SA / Gnu FDL

CREATE TABLE with Vectors

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

MariaDB has a dedicated VECTOR(N) data type with a built-in data validation. `N` is the number of dimensions that all vector values in the column have.

> **Note:**
>
> - Vector indexes are dimensionality-specific.
> - All vectors inserted into an indexed column must match the index's target dimensionality.
> - Inserting vectors with different dimensionalities will result in an error.

Consider the following table:

```
CREATE TABLE embeddings (
        doc_id BIGINT UNSIGNED PRIMARY KEY,
        embedding VECTOR(1536)
);
```

To have a fast vector search, you have to index the vector column, creating a `VECTOR` index:

```
CREATE TABLE embeddings (
        doc_id BIGINT UNSIGNED PRIMARY KEY,
        embedding VECTOR(1536) NOT NULL,
        VECTOR INDEX (embedding)
);
```

> **Warning:**
>
> Note that there can be only one vector index in the table, and the indexed vector column must be `NOT NULL` .

There are two options that can be used to configure the vector index:

- `M` — Larger values mean slower `SELECT` and `INSERT` statements, larger index size and higher memory consumption, but more accurate results. The valid range is from `3` to `200` .
- `DISTANCE` — Distance function to build the vector index for. Searches using a different distance function will not be able to use a vector index. Valid values are `cosine` and `euclidean` (the default).

```
CREATE TABLE embeddings (
        doc_id BIGINT UNSIGNED PRIMARY KEY,
        embedding VECTOR(1536) NOT NULL,
        VECTOR INDEX (embedding) M=8 DISTANCE=cosine
);
```

> **Note:**
>
> Declare `DISTANCE` explicitly. The default is `euclidean`, and a query using a different distance function than the one the index was built for cannot use the index, it falls back to a full table scan. Match the distance function to the metric your embedding model recommends.

This page is licensed: CC BY-SA / Gnu FDL

Vector System Variables

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

This page documents system variables related to Vectors.

See Server System Variables for instructions on setting them.

Also see the Full list of MariaDB options, system and status variables.

`mhnsw_default_distance`

- Description: Specifies the default distance metric for MHNSW vector indexing. This is used when the `DISTANCE` option is not explicitly defined during index creation.
- Command line: `--mhnsw-default-distance=val`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `euclidean`
- Valid Values:
  - `euclidean` Calculates straight-line distance between vectors. Best for spatial data, images, etc, when absolute magnitude matters.
  - `cosine` Measures directional similarity between vectors. Ideal for text embeddings, semantic search, and when vector magnitude is less important.
- Introduced: MariaDB 11.7.1

`mhnsw_default_m`

- Description: Defines the default value for the M parameter in MHNSW vector indexing. The `M` parameter controls the number of connections per layer in the graph structure, influencing the balance between search performance and index size.
  - Larger `M` → Better search accuracy, but larger index size and slower updates and searches.
  - Smaller `M` → Faster updates and searches, smaller index, but potentially less accurate search.
- Command line: `--mhnsw-default-m=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `int unsigned`
- Default Value: `6`
- Range: `3` to `200`
- Introduced: MariaDB 11.7.1

`mhnsw_ef_search`

- Description: Defines the minimal number of result candidates to look for in the vector index for ORDER BY … LIMIT N queries. The search will never search for less rows than that, even if LIMIT is smaller. This notably improves the search quality at low LIMIT values, at the expense of search time. Higher values may increase search quality but will also impact query performance.
- Command line: `--mhnsw-ef-search=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `int unsigned`
- Default Value: `20`
- Range: `1` to `10000`
- Introduced: MariaDB 11.7.1

`mhnsw_max_cache_size`

- Description: Upper limit for one MHNSW vector index cache. This limits the amount of memory that can be used for caching the index, ensuring efficient memory utilization.
- Command line: `--mhnsw-max-cache-size=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `bigint unsigned`
- Default Value: `16777216` (16 MB)
- Range: `1048576` to `18446744073709551615`
- Introduced: MariaDB 11.7.1

This page is licensed: CC BY-SA / Gnu FDL

Vector Framework Integrations

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

MariaDB Vector has integrations in several frameworks. For a general overview of MariaDB Vector — feature summary, benchmarks, tutorials, and demos — see the MariaDB Vector project page.

AI Framework Integrations

- LangChain, MariaDB Vector Store - Python (PyPI package)
- LangChain4j, MariaDB Embedding Store - Java
- LlamaIndex, MariaDB Vector Store - Python
- MCP (Model Context Protocol), MariaDB MCP server - Python
- Spring AI, MariaDB Vector Store - Java
- VectorDBBench - benchmarking for vector databases

Web Framework and ORM Integrations

- Laravel, vector columns and vector indexes in the schema builder - PHP, since Laravel 13; `$table->vectorIndex('embedding')` compiles to a MariaDB `VECTOR INDEX` with `M=6 DISTANCE=cosine` (laravel/framework#60334)
- laravel-mariadb-vector, vector casts and similarity search for Eloquent models - PHP, community package

- TypeORM, MariaDB vector columns - TypeScript/Node.js, since TypeORM 0.3.28
- Hibernate ORM, MariaDB vector type - Java, since Hibernate ORM 7.0

For a worked example of picking an embedding model for MariaDB Vector in a Laravel application, see MariaDB Vector in Laravel: insights on choosing an embedding model.

Potential Future Vector or AI Integrations

- AutoGen - Agent to agent, Python
- DB-GPT - private LLM, vector search and text2sql, see integration docs, Python
- Feast - machine learning (not GenAI), Python
- Firebase Studio template for MariaDB Vector - visit link to vote for suggestion
- LangGraph - Agentic workflow, Python
- MindSQL - Text-to-SQL RAG Library simplifying database interactions, Python
- Open WebUI - AI Interface, Python & Javascript
- Vanna AI - Vector search and text2sql, Python

For the full, continuously updated catalog of AI frameworks, platforms, and tools that work with MariaDB, see the AI & Application Development category of the MariaDB Ecosystem Hub.

This page is licensed: CC BY-SA / Gnu FDL

Vector Functions

See VEC_DISTANCE.

Calculate distance between vectors. This function computes the distance between two vectors using either Euclidean or Cosine metric, depending on the index.

See VEC_DISTANCE_COSINE.

Calculate Cosine distance. This function computes the Cosine distance between two vectors, measuring the cosine of the angle between them.

See VEC_DISTANCE_EUCLIDEAN.

Calculate Euclidean distance. This function computes the Euclidean (L2) distance between two vectors, representing the straight-line distance.

See VEC_FromText.

Convert text to vector. This function parses a JSON array string representation of a vector and converts it into the binary VECTOR data type.

See VEC_ToText.

Convert vector to text. This function takes a binary VECTOR data type and returns its JSON array string representation.

This page is licensed: CC BY-SA / Gnu FDL

VECTOR

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

Syntax

```
VECTOR(N)
```

Description

The `VECTOR` data type was added as part of the vectors feature, which permits MariaDB Server to perform as a relational vector database. `N` represents the fixed number of dimensions of the vector up to a maximum of 16383. The `N` dimension will be determined by the embedding algorithm.

Example

```
CREATE TABLE t1 (id INT AUTO_INCREMENT PRIMARY KEY, v VECTOR(5) NOT NULL,
 VECTOR INDEX (v));
```

See Also

- CREATE TABLE with Vectors

This page is licensed: CC BY-SA / Gnu FDL

Optimizing Hybrid Search Query with Reciprocal Rank Fusion (RRF)

Hybrid search combines the keyword precision of full-text search with the conceptual understanding of vector search to produce a single, superior set of results.

Full-Text Search

Full-text search is the traditional keyword-based search, excelling at finding documents that contain the exact words from your query. Behind the scenes, it relies on a data structure called an inverted index—a dictionary that maps each word to a list of documents it appears in, allowing for very fast lookups. For instance, a search for ‘apple pie recipe’ will instantly find all documents indexed under those three words: ‘apple‘, ‘pie‘ & ‘recipe‘.

Vector Search

Vector search is a modern search method based on meaning. It finds documents that are conceptually similar to your query, even if they don’t share any keywords. It works by converting both the query and the documents into numerical representations called “vector embeddings.” These vectors exist as points in a high-dimensional conceptual space. A search then finds the “nearest neighbors”—the document points that are closest to the query point. For instance, a vector search for “apple pie recipe” might also return a document titled “how to bake a Granny Smith tart,” because the model understands that “tart” is similar to “pie” and “Granny Smith” is a type of “apple.”

The Power of Hybrid Search

Full-text search offers precise keyword matching, while vector search provides nuanced understanding of concepts. Together, they handle ambiguity and ensure critical queries aren’t missed, resulting in a robust, intelligent search experience.

Combining Full-Text and Vector Search

A Search for “Sustainable Coffee Pods”

When a user searches for “sustainable coffee pods,” the two search systems return the following ranked lists.

Full-Text Search Results

Rank	ID	Title
1	1	‘Eco Coffee Pods - 100 Count’
2	3	‘Recyclable Coffee Capsules’
3	4	‘Morning Roast Coffee Beans’

Vector Search Results

Rank	ID	Title
1	2	‘Compostable Espresso Pods’
2	3	‘Recyclable Coffee Capsules’
3	6	‘Bamboo Reusable Coffee Filter’

When two search systems produce distinct lists of titles, a traditional merging approach would prioritize titles that appear in both lists, similar to an INNER JOIN operation.

Why a Simple INNER JOIN Is Insufficient

An INNER JOIN only returns results found by both search systems, discarding valuable items that appear in only one list.

If we were to INNER JOIN the two lists from our “sustainable coffee pods” search, the result would be:

ID	Title
3	‘Recyclable Coffee Capsules’

This result is severely incomplete. It correctly finds the one common item but completely discards the #1 ranked result from both searches (‘Eco Coffee Pods’ and ‘Compostable Espresso Pods’) simply because they were specialists found by only one system.

Reciprocal Rank Fusion (RRF)

To solve the problem of discarded results, we use Reciprocal Rank Fusion (RRF). The power of RRF lies in its simplicity: it operates on the rank of an item (1st, 2nd, 3rd) in a list, not its raw, non-comparable score. This makes it highly effective for merging lists from different systems without needing to normalize their scores.

A helpful way to think of this is to imagine your different search systems as a “panel of expert advisors”; RRF intelligently combines their opinions using the formula:

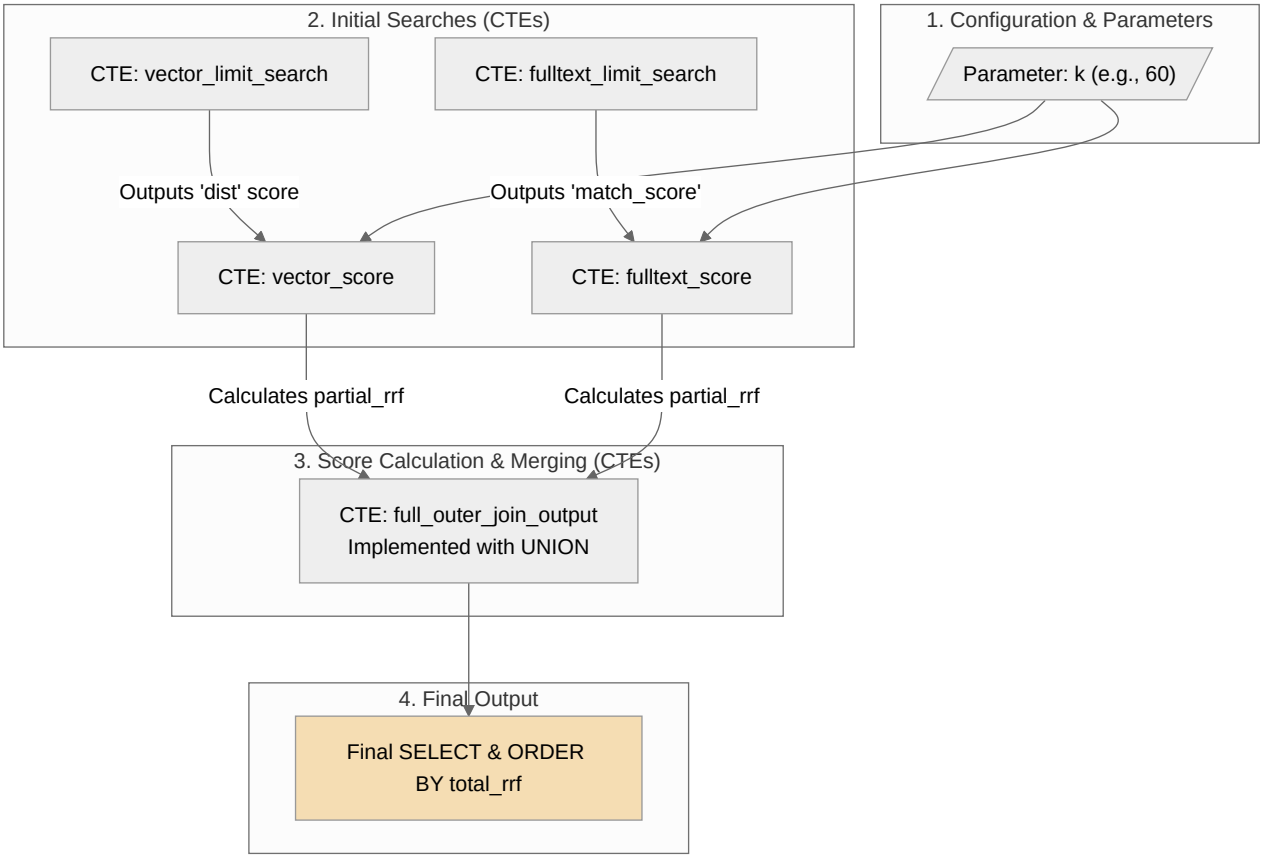
$$ RRF_{score} = \sum_{i} \frac{1}{k + \text{rank}_i} $$

Where `rank` is the position of a document in a list, and `k` is a tuning constant that moderates the influence of that rank.

> **Note:**
>
> **RRF’s k Parameter**
>
> The k parameter is the primary tuning knob for the RRF algorithm, acting as a smoothing factor that controls how results are weighted. A low k gives immense power to a top-ranked result, while a high k is more skeptical of a single top pick and rewards items found by multiple systems (consensus). A value of k=60 is a robust and effective baseline for general use.

A key advantage of RRF is that it ignores the raw scores from the search systems and uses only the rank. The table below shows the partial RRF score for the top items in our “sustainable coffee pods” search, calculated with k=60.

Building the Hybrid Query Optimization model with CTEs



To build the model, our query uses the following Common Table Expressions (CTEs).

CTE Name	Purpose
vector_limit_search	Runs the initial vector search to retrieve the top N conceptually similar items.
fulltext_limit_search	Runs the initial full-text search to retrieve the top N keyword matches.
vector_score	Calculates a partial_rrf score for each item from the vector search based on its rank.
fulltext_score	Calculates a partial_rrf score for each item from the full-text search based on its rank.
full_outer_join_output	Merges the two score lists and sums the partial scores into a total_rrf.

The annotated SQL query uses the CTEs to perform the hybrid search:

```
-- Set the parameters for our query.
SET @k = 60; -- The RRF smoothing constant.
SET @search_term = "sustainable coffee pods";
-- The vector for the search term would be set here.
-- SET @search_term_vector = VEC_FromText("...");

-- Use CTEs to break the logic into sequential steps.
WITH
-- STEP 1: Get top results from each search method.
vector_limit_search AS (
  SELECT id, title, VEC_DISTANCE_EUCLIDEAN(embedding, @search_term_vector) AS dist
  FROM products ORDER BY dist ASC LIMIT 10
),
fulltext_limit_search AS (
  SELECT id, title, MATCH(title) AGAINST (@search_term) AS match_score
  FROM products WHERE MATCH(title) AGAINST (@search_term)
  ORDER BY match_score DESC LIMIT 10
),
-- STEP 2: Calculate partial RRF scores for each list.
vector_score AS (
  SELECT id, title, 1 / (@k + RANK() OVER (ORDER BY dist ASC)) AS partial_rrf
  FROM vector_limit_search
),
fulltext_score AS (
  SELECT id, title, 1 / (@k + RANK() OVER (ORDER BY match_score DESC)) AS partial_rrf
  FROM fulltext_limit_search
),
-- STEP 3: Merge the two lists and sum the scores.
full_outer_join_output AS (
  SELECT v.id, v.title, (v.partial_rrf + IFNULL(f.partial_rrf, 0)) AS total_rrf
  FROM vector_score v LEFT JOIN fulltext_score f USING (id)
  UNION
  SELECT f.id, f.title, (IFNULL(v.partial_rrf, 0) + f.partial_rrf) AS total_rrf
  FROM fulltext_score f LEFT JOIN vector_score v USING (id)
)
-- STEP 4: Select the final, unified list.
SELECT id, title, total_rrf FROM full_outer_join_output
ORDER BY total_rrf DESC
LIMIT 10;
```

Final Calculation for “Sustainable Coffee Pods”

With k=60, the final CTE merges and sums the partial scores. The consensus item, found in both lists, rises to the top.

ID	Title	fulltext_rrf	vector_rrf	total_rrf	Final Rank
3	Recyclable Coffee Capsules	0.01613	0.01613	0.03226	1
1	Eco Coffee Pods	0.01639	0	0.01639	2
2	Compostable Espresso Pods	0	0.01639	0.01639	2

Tuning the k Parameter

As introduced earlier, the k parameter can be fine-tuned for specific situations. Here are three common scenarios to consider.

1. **Combining Diverse, Specialist Systems**

For merging results from different methods like keyword and vector search, a higher k (such as 60) is ideal to balance their contributions. Our main “sustainable coffee pods” search is a perfect illustration of this, where the consensus item wins but the high-quality specialists are ranked immediately after.

Final Rank	ID	Title	Total RRF	Note
1	3	Recyclable Coffee Capsules	0.03226	Consensus Winner
2	1	Eco Coffee Pods	0.01639	Specialist
2	2	Compostable Espresso Pods	0.01639	Specialist

2. **Handling Mixed-Quality or "Noisy" Systems**

If you are fusing results from reliable systems and one experimental, less predictable system, a higher k (60 or more) is the safest choice to prevent an outlier from disproportionately influencing the final rank.

Consider our search with a third, "noisy" system that incorrectly ranks 'Eco-Friendly Car Wax' (ID 8) at #1. A high k value minimizes the impact of this error.

Product	FT Rank	Vector Rank	Noisy Rank	Total RRF (k=60)
Recyclable Coffee Capsules	2	2	-	1/62 + 1/62 = 0.03226
Eco-Friendly Car Wax	-	-	1	1/61 = 0.01639

The high k value correctly ensures that the consensus result from the two reliable systems easily beats the single, erroneous result from the noisy system.

3. **Fusing High-Quality, Similar Systems**

If you are combining lists from two very similar, high-performing algorithms, you can experiment with a slightly lower k (30-50) to give more weight to a top-ranked document.

Consider a case where we fuse two similar vector models (Vector_A, Vector_B) that both rank 'Compostable Espresso Pods' as #1 and 'Recyclable Coffee Capsules' as #2. A lower k makes the winner more decisive.

Product	Ranks	Total RRF (k=30)	Total RRF (k=60)
Compostable Pods	1, 1	1/31 + 1/31 = 0.0645	1/61 + 1/61 = 0.0328
Recyclable Capsules	2, 2	1/32 + 1/32 = 0.0625	1/62 + 1/62 = 0.0322
Score Difference		0.0020	0.0006

With k=30, the score separation between the #1 and #2 results is more than three times larger, showing higher confidence in the top result, which is desirable when you trust both systems.

Determining the Best k (The Experimental Method)

A formal, 3-step process can be used to scientifically determine the best k value for your data through offline evaluation. To illustrate this process, which requires a pre-judged "ground truth" set with graded relevance, we will use a separate, self-contained case study.

1. **Gather Prerequisites**

You need: Multiple Ranked Lists, a "Ground Truth" Set, and an Evaluation Metric (like NDCG). With a query like "healthy breakfast", your ground truth might look like this:

Doc ID	Title	Relevance
A	'Oatmeal with Berries'	3 (High)
B	'Green Smoothie'	3 (High)
C	'Avocado Toast'	2 (Medium)
D	'Bacon and Eggs'	1 (Low)
E	'Cinnamon Roll'	0 (None)

And your raw ranked lists might be:

- Full-Text Results: [D, A, E]
- Vector Results: [B, C, A]

2. **Run the Experiment**

You iterate through k values, applying the RRF formula to the raw lists to generate a final ranking for each k.

- Final Ranked List (k=10):

Rank	ID	Title
1	B	‘Green Smoothie’
2	D	‘Bacon and Eggs’
3	A	‘Oatmeal with Berries’
4	C	‘Avocado Toast’
5	E	‘Cinnamon Roll’

- Final Ranked List (k=60):

Rank	ID	Title
1	A	‘Oatmeal with Berries’
2	B	‘Green Smoothie’
3	C	‘Avocado Toast’
4	D	‘Bacon and Eggs’
5	E	‘Cinnamon Roll’

3. **Analyze and Select**

You use your metric to “grade” each list against the Ground Truth and choose the k with the highest score. The k=60 list is clearly better as it placed the two “Highly Relevant” documents (A and B) at the top.

k Value	Performance Score (NDCG)
10	0.85
60	0.92

In this experiment, k=60 is the winner. A key advantage of RRF is that its performance is “not critically sensitive to the choice of k, making it a robust and reliable method” for improving search relevance.

Further Reading

- Reciprocal Rank Fusion for IR (SIGIR ’09): The original research paper that proposed the RRF method.

This page is licensed: CC BY-SA / Gnu FDL

SQL Statements

See Data Definition (DDL).

Learn data definition language (DDL) statements in MariaDB Server. This section covers SQL commands for creating, altering, and dropping databases, tables, indexes, and other schema objects.

See Data Manipulation (DML).

Learn data manipulation language (DML) statements in MariaDB Server. This section covers SQL commands for inserting, updating, deleting, and selecting data within your databases.

See Account Management.

Learn account management SQL statements for MariaDB Server. This section covers commands like CREATE USER, GRANT, and REVOKE to securely manage user access and privileges within your database.

See Administrative Statements.

Explore administrative SQL statements for MariaDB Server. This section covers commands for server management, maintenance, and diagnostics, including BINLOG, KILL, SHUTDOWN, and SHOW.

See SQL Functions.

Discover functions and procedures in MariaDB.

See Prepared Statements.

Learn about prepared statements in MariaDB Server. This section details how to use them for efficient and secure execution of repetitive SQL queries, preventing SQL injection vulnerabilities.

See Programmatic & Compound Statements.

Explore programmatic compound statements in MariaDB Server. This section covers BEGIN…END blocks, loops, and conditional logic for writing complex stored routines and event definitions.

See Stored Routines.

Automate tasks in MariaDB Server with stored routines. Learn to create and manage stored procedures and functions for enhanced database efficiency and code reusability.

See Table Statements.

Learn about table-related SQL statements in MariaDB Server. This section covers commands for creating, altering, dropping, and manipulating tables, essential for managing your database schema.

See Transactions.

Learn about transactions in MariaDB Server. This section covers SQL statements for managing atomic operations (START TRANSACTION, COMMIT, ROLLBACK), ensuring data integrity and consistency.

See Comment Syntax.

Complete SQL comment syntax: single-line # and – (space required), multi-line /* / *blocks, MySQL* /!##### / *and MariaDB* /M!##### */ executable comments.

See HELP Command.

Displays help information from the server's help tables. Useful for looking up SQL syntax and command descriptions directly from the client.

This page is licensed: CC BY-SA / Gnu FDL

Comment Syntax

There are three supported comment styles in MariaDB:

1. From a hash sign (`#`) to the end of a line:

```
SELECT * FROM users;

# This is a comment
```

1. From a double dash (`--`) to the end of a line. The space after the two dashes is required (as in MySQL).

```
SELECT * FROM users; -- This is a comment
```

1. C style comments from an opening ' `/*` ' to a closing ' `*/` '. Comments of this form can span multiple lines:

```
SELECT * FROM users; /* This is a
multi-line
comment */
```

Nested comments are possible in some situations, but they are not supported or recommended.

Executable Comments

As an aid to portability between different databases, MariaDB supports executable comments. These special comments allow you to embed SQL code which will not execute when run on other databases, but will execute when run on MariaDB.

MariaDB supports both MySQL's executable comment format, and a slightly modified version specific to MariaDB. This way, if you have SQL code that works on MySQL and MariaDB, but not other databases, you can wrap it in a MySQL executable comment, and if you have code that specifically takes advantage of features only available in MariaDB you can use the MariaDB specific format to hide the code from MySQL.

Executable Comment Syntax

MySQL and MariaDB executable comment syntax:

```
/*! MySQL or MariaDB-specific code */
```

Code that should be executed only starting from a specific MySQL or MariaDB version:

```
/*!##### MySQL or MariaDB-specific code */
```

The numbers, represented by ' `######` ' in the syntax examples above specify the specific the minimum versions of MySQL and MariaDB that should execute the comment. The first number is the major version, the second 2 numbers are the minor version and the last 2 is the patch level.

For example, if you want to embed some code that should only execute on MySQL or MariaDB starting from 5.1.0, you would do the following:

```
/*!50100 MySQL and MariaDB 5.1.0 (and above) code goes here. */
```

MariaDB-only executable comment syntax (starting from MariaDB 5.3.1):

```
/*M! MariaDB-specific code */
/*M!###### MariaDB-specific code */
```

MariaDB ignores MySQL-style executable comments that have a version number in the range `50700..99999`. This is needed to skip features introduced in MySQL-5.7 that are not ported to MariaDB 10.x yet.

```
/*!50701 MariaDB-10.x ignores MySQL-5.7 specific code */
```

Note: comments which have a version number in the range `50700..99999` that use MariaDB-style executable comment syntax are still executed.

```
/*M!50701 MariaDB-10.x does not ignore this */
```

Statement delimiters cannot be used within executable comments.

Examples

In MySQL all the following will return 2:
In MariaDB, the last 2 queries would return 3.

```
SELECT 2 /* +1 */;
SELECT 1 /*! +1 */;
SELECT 1 /*!50101 +1 */;
SELECT 2 /*M! +1 */;
SELECT 2 /*M!50301 +1 */;
```

The following executable statement will not work due to the delimiter inside the executable portion:

```
/*M!100100 select 1 ; */
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MariaDB server version for the right syntax to use near '' at line 1
```

Instead, the delimiter should be placed outside the executable portion:

```
/*M!100100 select 1 */;
+---+
| 1 |
+---+
| 1 |
+---+
```

This page is licensed: CC BY-SA / Gnu FDL

Account Management

See ALTER USER.

Complete ALTER USER guide for MariaDB. Complete syntax for modifying authentication, passwords, and account security settings with comprehensive examples and.

See CREATE ROLE.

Create new roles to simplify privilege management. Learn how to define a role that can be assigned to multiple users or other roles.

See CREATE USER.

Complete guide to creating MariaDB user accounts. Complete CREATE USER syntax for authentication methods and password policies with comprehensive examples.

See DENY.

Block a privilege outright. Learn how a deny overrides every grant at every level, and how to lift one again with REVOKE DENY.

See DROP ROLE.

Remove a role from the system. Learn the syntax to delete defined roles and revoke them from any users or roles that currently hold them.

See DROP USER.

Complete DROP statement reference for MariaDB. Complete guide for safely removing database objects with CASCADE options with comprehensive examples and best.

See GRANT.

Complete privilege management guide for MariaDB. Complete GRANT syntax for database, table, and column permissions with roles with comprehensive examples and.

See RENAME USER.

Rename existing database accounts. This guide explains how to change a user's name while preserving their current privileges and properties.

See REVOKE.

Remove privileges or roles. Learn how to withdraw previously granted permissions from users or roles to restrict access and secure the database.

See SET DEFAULT ROLE.

Define the default role for a user. Learn how to configure which role is automatically active when a user connects to the server.

See SET PASSWORD.

Complete reference for SET PASSWORD in MariaDB. Complete syntax guide with all options, clauses, and practical examples with comprehensive examples and best.

See SET SESSION AUTHORIZATION.

Impersonate another user for the current session. Learn how to assume the identity and privileges of another account for testing or administration.

See SET ROLE.

Sets the current role for the session. Learn how to enable none, or a specific role to change your current privileges dynamically.

This page is licensed: CC BY-SA / Gnu FDL

ALTER USER

Syntax

```
ALTER USER [IF EXISTS]
 user_specification [,user_specification] ...
  [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
  [WITH resource_option [resource_option] ...]
  [lock_option] [password_option]

user_specification:
  username [authentication_option]

authentication_option:
  IDENTIFIED BY 'password'
  | IDENTIFIED BY PASSWORD 'password_hash'
  | IDENTIFIED {VIA|WITH} authentication_rule [OR authentication_rule] ...

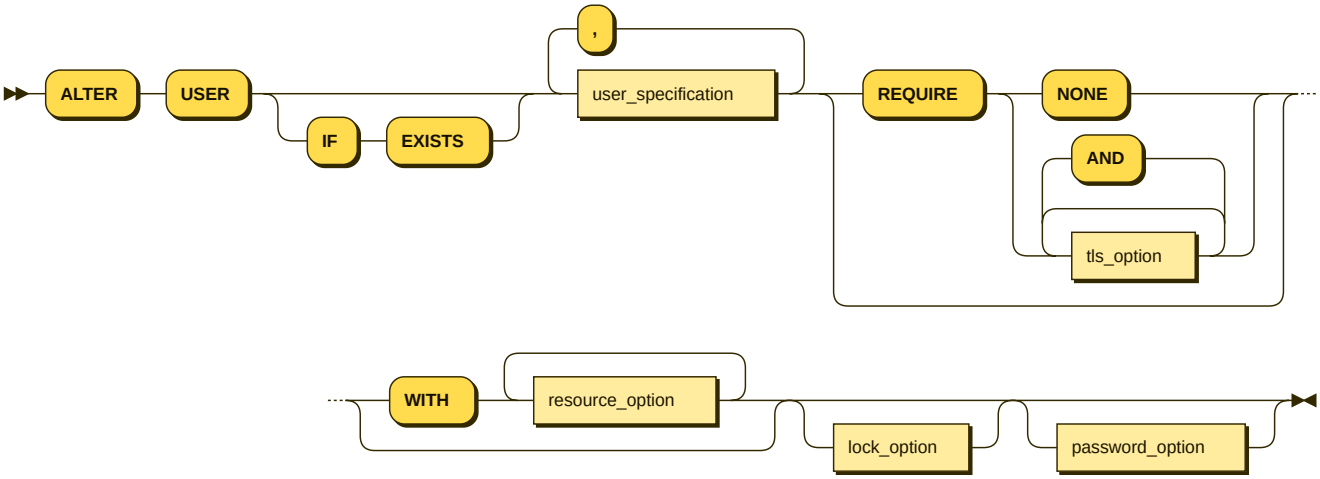
authentication_rule:
  authentication_plugin
  | authentication_plugin {USING|AS} 'authentication_string'
  | authentication_plugin {USING|AS} PASSWORD('password')

tls_option:
  SSL
  | X509
  | CIPHER 'cipher'
  | ISSUER 'issuer'
  | SUBJECT 'subject'

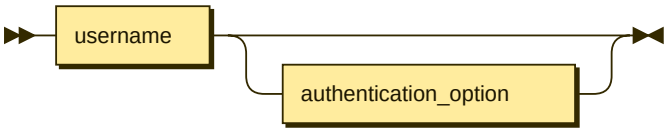
resource_option:
  MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count
  | MAX_STATEMENT_TIME time

password_option:
  PASSWORD EXPIRE
  | PASSWORD EXPIRE DEFAULT
  | PASSWORD EXPIRE NEVER
  | PASSWORD EXPIRE INTERVAL N DAY

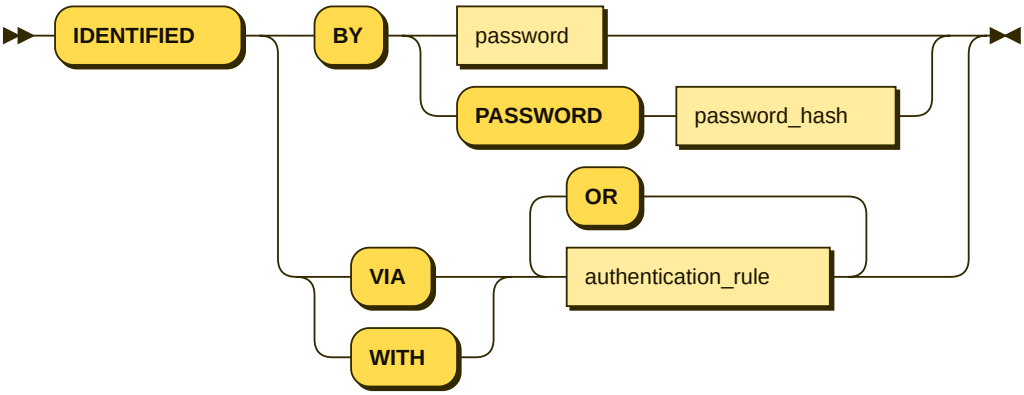
lock_option:
    ACCOUNT LOCK
  | ACCOUNT UNLOCK
```



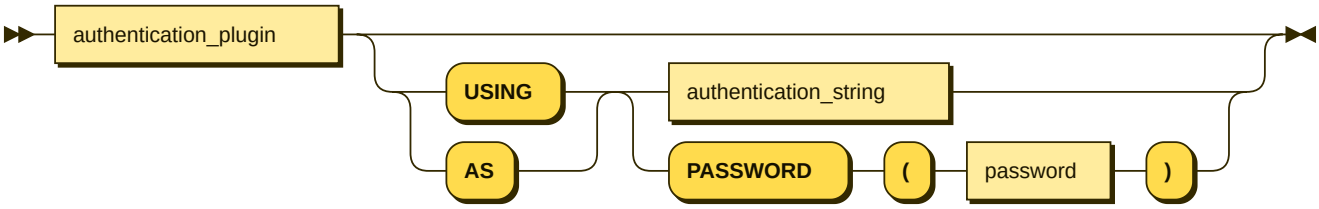
Railroad diagram of ALTER USER — equivalent to the BNF above



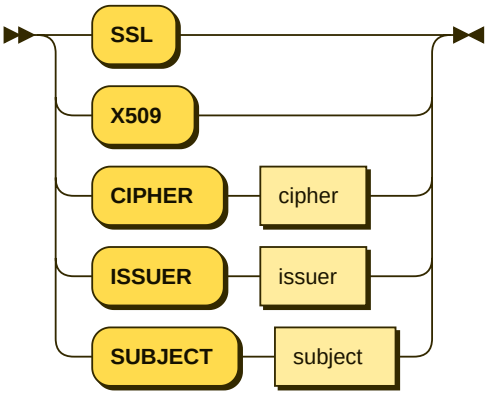
Railroad diagram of user_specification



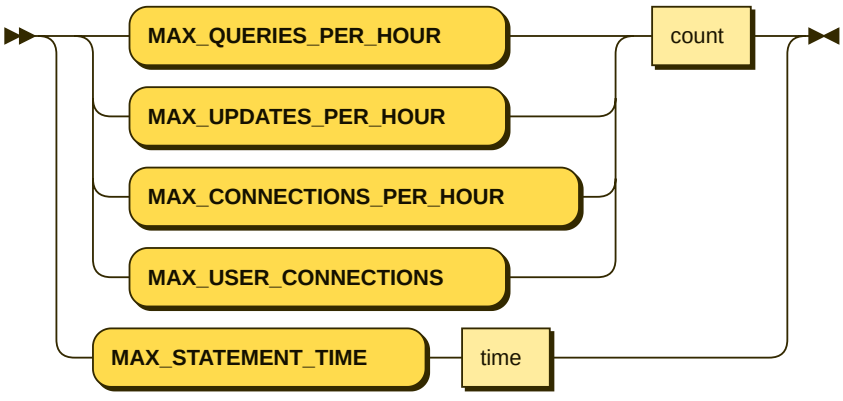
Railroad diagram of authentication_option



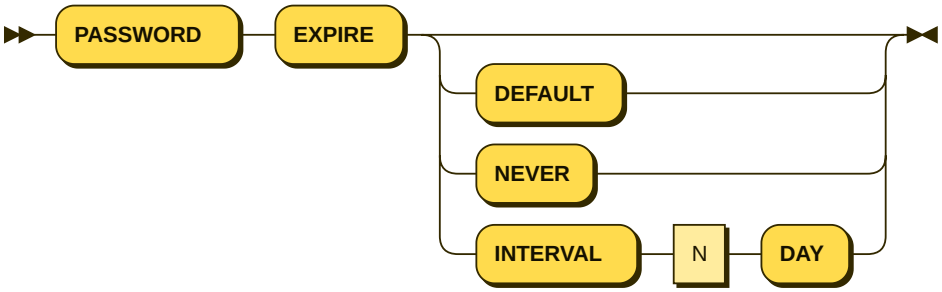
Railroad diagram of authentication_rule



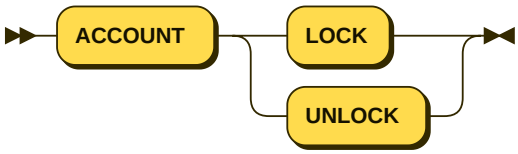
Railroad diagram of tls_option



Railroad diagram of resource_option



Railroad diagram of password_option



Railroad diagram of lock_option

Description

The `ALTER USER` statement modifies existing MariaDB accounts. To use it, you must have the global CREATE USER privilege or the UPDATE privilege for the mysql database. The global READ_ONLY ADMIN privilege is also required if the read_only system variable is enabled.

If any of the specified user accounts do not yet exist, an error results. If an error occurs, `ALTER USER` will still modify the accounts that do not result in an error. Only one error is produced for all users which have not been modified.

For renaming an existing account (user name and/or host), see RENAME USER.

IF EXISTS

When the `IF EXISTS` clause is used, MariaDB will return a warning instead of an error for each specified user that does not exist.

Account Names

For `ALTER USER` statements, account names are specified as the `username` argument in the same way as they are for CREATE USER statements. See account names from the `CREATE USER` page for details on how account names are specified.

CURRENT_USER or `CURRENT_USER()` can also be used to alter the account logged into the current session. For example, to change the current user's password to `mariadb`:

```
ALTER USER CURRENT_USER() IDENTIFIED BY 'mariadb';
```

Authentication Options

From MariaDB 10.4, it is possible to use more than one authentication plugin for each user account. For example, this can be useful to slowly migrate users to the more secure ed25519 authentication plugin over time, while allowing the old mysql_native_password authentication plugin as an alternative for the transitional period. See Authentication from MariaDB 10.4 for more.

When running `ALTER USER`, not specifying an authentication option in the IDENTIFIED VIA clause will remove that authentication method. (However this was not the case before MariaDB 10.4.13, see MDEV-21928)

For example, a user is created with the ability to authenticate via both a password and unix_socket:

```
CREATE USER 'bob'@'localhost'
  IDENTIFIED VIA mysql_native_password USING PASSWORD('pwd')
  OR unix_socket;

SHOW CREATE USER 'bob'@'localhost'\G
```

```
*************************** 1. row ***************************
CREATE USER for bob@localhost: CREATE USER `bob`@`localhost`
  IDENTIFIED VIA mysql_native_password
  USING '*975B2CD4FF9AE554FE8AD33168FBFC326D2021DD'
  OR unix_socket
```

If the user's password is updated, but unix_socket authentication is not specified in the `IDENTIFIED VIA` clause, unix_socket authentication will no longer be permitted.

```
ALTER USER 'bob'@'localhost' IDENTIFIED VIA mysql_native_password
  USING PASSWORD('pwd2');

SHOW CREATE USER 'bob'@'localhost'\G
```

```
*************************** 1. row ***************************
CREATE USER for bob@localhost: CREATE USER `bob`@`localhost`
  IDENTIFIED BY PASSWORD '*38366FDA01695B6A5A9DD4E428D9FB8F7EB75512'
```

IDENTIFIED BY 'password'

The optional `IDENTIFIED BY` clause can be used to provide an account with a password. The password should be specified in plain text. It will be hashed by the PASSWORD function prior to being stored in the mysql.user view.

For example, if our password is `mariadb`, then we can set the account's password with:

```
ALTER USER foo2@test IDENTIFIED BY 'mariadb';
```

If you do not specify a password with the `IDENTIFIED BY` clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

The only authentication plugins that this clause supports are mysql_native_password and mysql_old_password.

IDENTIFIED BY PASSWORD 'password_hash'

The optional `IDENTIFIED BY PASSWORD` clause can be used to provide an account with a password that has already been hashed. The password should be specified as a hash that was provided by the PASSWORD#function. It will be stored in the mysql.user view as-is.

For example, if our password is `mariadb`, then we can find the hash with:

```
SELECT PASSWORD('mariadb');
```

```
+-------------------------------------------+
| PASSWORD('mariadb')                       |
+-------------------------------------------+
| *54958E764CE10E50764C2EECBB71D01F08549980 |
+-------------------------------------------+
```

And then we can set an account's password with the hash:

```
ALTER USER foo2@test
  IDENTIFIED BY PASSWORD '*54958E764CE10E50764C2EECBB71D01F08549980';
```

If you do not specify a password with the `IDENTIFIED BY` clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

The only authentication plugins that this clause supports are mysql_native_password and mysql_old_password.

IDENTIFIED {VIA|WITH} authentication_plugin

The optional `IDENTIFIED VIA authentication_plugin` allows you to specify that the account should be authenticated by a specific authentication plugin. The plugin name must be an active authentication plugin as per SHOW PLUGINS. If it doesn't show up in that output, then you will need to install it with INSTALL PLUGIN or INSTALL SONAME.

For example, this could be used with the PAM authentication plugin:

```
ALTER USER foo2@test IDENTIFIED VIA pam;
```

Some authentication plugins allow additional arguments to be specified after a `USING` or `AS` keyword. For example, the PAM authentication plugin accepts a service name:

```
ALTER USER foo2@test IDENTIFIED VIA pam USING 'mariadb';
```

The exact meaning of the additional argument would depend on the specific authentication plugin.

> **CURRENT**
>
> The `USING` or `AS` keyword can also be used to provide a plain-text password to a plugin if it's provided as an argument to the PASSWORD() function. This is only valid for authentication plugins that have implemented a hook for the PASSWORD() function. For example, the ed25519 authentication plugin supports this:
>
> ```
> ALTER USER safe@'%' IDENTIFIED VIA ed25519 USING PASSWORD('secret');
> ```
>
> **< 10.4**
>
> The `USING` or `AS` keyword **cannot** be used to provide a plain-text password to a plugin if it's provided as an argument to the PASSWORD() function.

TLS Options

By default, MariaDB transmits data between the server and clients without encrypting it. This is generally acceptable when the server and client run on the same host or in networks where security is guaranteed through other means. However, in cases where the server and client exist on separate networks or they are in a high-risk network, the lack of encryption does introduce security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

To mitigate this concern, MariaDB allows you to encrypt data in transit between the server and clients using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix ssl_, but internally, MariaDB only supports its secure successors.

See Secure Connections Overview for more information about how to determine whether your MariaDB server has TLS support.

You can set certain TLS-related restrictions for specific user accounts. For instance, you might use this with user accounts that require access to sensitive data while sending it across networks that you do not control. These

restrictions can be enabled for a user account with the CREATE USER, ALTER USER, or GRANT statements. The following options are available:

Option	Description
REQUIRE NONE	TLS is not required for this account, but can still be used.
REQUIRE SSL	The account must use TLS, but no valid X509 certificate is required. This option cannot be combined with other TLS options.
REQUIRE X509	The account must use TLS and must have a valid X509 certificate. This option implies REQUIRE SSL. This option cannot be combined with other TLS options.
REQUIRE ISSUER 'issuer'	The account must use TLS and must have a valid X509 certificate. Also, the Certificate Authority must be the one specified via the string issuer. This option implies REQUIRE X509. This option can be combined with the SUBJECT, and CIPHER options in any order.
REQUIRE SUBJECT 'subject'	The account must use TLS and must have a valid X509 certificate. Also, the certificate's Subject must be the one specified via the string subject. This option implies REQUIRE X509. This option can be combined with the ISSUER, and CIPHER options in any order.
REQUIRE CIPHER 'cipher'	The account must use TLS, but no valid X509 certificate is required. Also, the encryption used for the connection must use a specific cipher method specified in the string cipher. This option implies REQUIRE SSL. This option can be combined with the ISSUER, and SUBJECT options in any order.

The `REQUIRE` keyword must be used only once for all specified options, and the `AND` keyword can be used to separate individual options, but it is not required.

For example, you can alter a user account to require these TLS options with the following:

```
ALTER USER 'alice'@'%'
 REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=Portland' AND
 ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter Parker/emailAddress=p.parker@marvel.com'
 AND CIPHER 'SHA-DES-CBC3-EDH-RSA';
```

If any of these options are set for a specific user account, then any client who tries to connect with that user account will have to be configured to connect with TLS.

See Securing Connections for Client and Server for information on how to enable TLS on the client and server.

Resource Limit Options

It is possible to set per-account limits for certain server resources. The following table shows the values that can be set per account:

Limit Type	Description
MAX_QUERIES_PER_HOUR	Number of statements that the account can issue per hour (including updates)
MAX_UPDATES_PER_HOUR	Number of updates (not queries) that the account can issue per hour
MAX_CONNECTIONS_PER_HOUR	Number of connections that the account can start per hour
MAX_USER_CONNECTIONS	Number of simultaneous connections that can be accepted from the same account; if it is 0, max_connections will be used instead; if max_connections is 0, there is no limit for this account's simultaneous connections.
MAX_STATEMENT_TIME	Timeout, in seconds, for statements executed by the user. See also Aborting Statements that Exceed a Certain Time to Execute.

If any of these limits are set to `0`, then there is no limit for that resource for that user.

Here is an example showing how to set an account's resource limits:

```
ALTER USER 'someone'@'localhost' WITH
    MAX_USER_CONNECTIONS 10
    MAX_QUERIES_PER_HOUR 200;
```

The resources are tracked per account, which means `'user'@'server'`; not per user name or per connection.

The count can be reset for all users using FLUSH USER_RESOURCES, FLUSH PRIVILEGES or mysqladmin reload.

Per account resource limits are stored in the user table, in the mysql database. Columns used for resources limits are named `max_questions`, `max_updates`, `max_connections` (for `MAX_CONNECTIONS_PER_HOUR`), and `max_user_connections` (for `MAX_USER_CONNECTIONS`).

Password Expiry

Besides automatic password expiry, as determined by default_password_lifetime, password expiry times can be set on an individual user basis, overriding the global setting, for example:

```
ALTER USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY;
ALTER USER 'monty'@'localhost' PASSWORD EXPIRE NEVER;
ALTER USER 'monty'@'localhost' PASSWORD EXPIRE DEFAULT;
```

See User Password Expiry for more details.

Account Locking

Account locking permits privileged administrators to lock/unlock user accounts. No new client connections will be permitted if an account is locked (existing connections are not affected). For example:

```
ALTER USER 'marijn'@'localhost' ACCOUNT LOCK;
```

See Account Locking for more details.

> **CURRENT**
>
> The *lock_option* and *password_option* clauses can occur in either order.
>
> **< 10.5.8, < 10.4.7**
>
> The *lock_option* must be placed before the *password_option*.

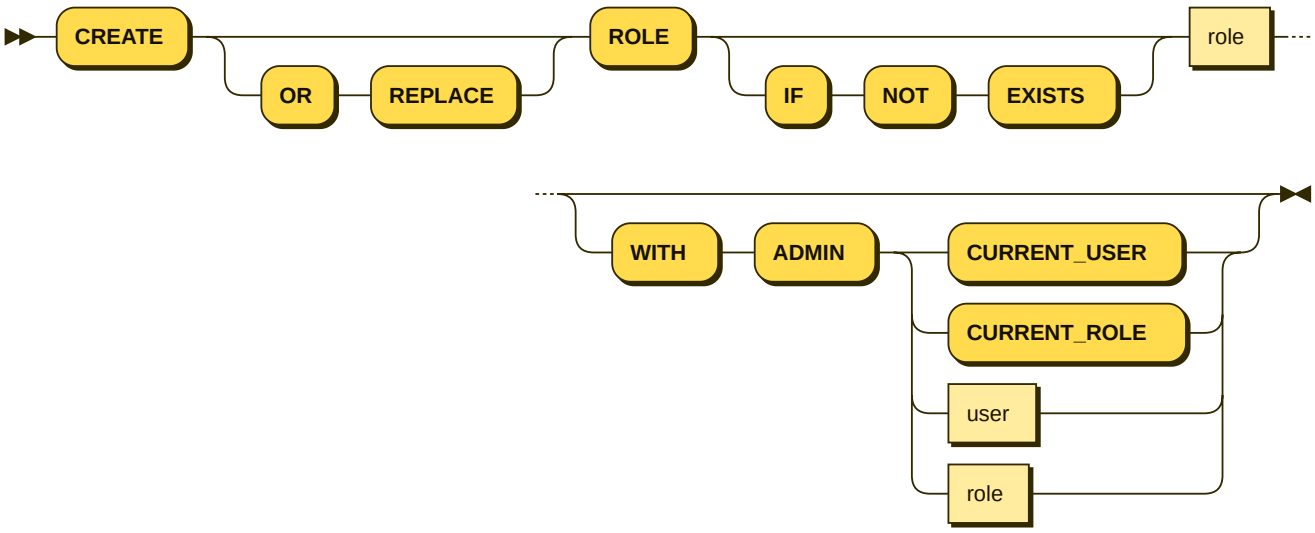
- Authentication from MariaDB 10.4
- GRANT
- CREATE USER
- DROP USER
- SET PASSWORD
- SHOW CREATE USER
- mysql.user
- Password Validation Plugins - permits the setting of basic criteria for passwords
- Authentication Plugins - allow various authentication methods to be used, and new ones to be developed.

This page is licensed: CC BY-SA / Gnu FDL

CREATE ROLE

Syntax

```
CREATE [OR REPLACE] ROLE [IF NOT EXISTS] role
  [WITH ADMIN
    {CURRENT_USER | CURRENT_ROLE | user | role}]
```



Railroad diagram of CREATE ROLE — equivalent to the BNF above

Description

The `CREATE ROLE` statement creates one or more MariaDB roles. To use it, you must have the global CREATE USER privilege or the INSERT privilege for the mysql database. For each account, `CREATE ROLE` creates a new row in the mysql.user table that has no privileges, and with the corresponding `is_role` field set to `Y`. It also creates a record in the mysql.roles_mapping table.

If any of the specified roles already exist, `ERROR 1396 (HY000)` results. If an error occurs, `CREATE ROLE` will still create the roles that do not result in an error. The maximum length for a role is 128 characters. Role names can be quoted, as explained in the Identifier names page. Only one error is produced for all roles which have not been created:

```
ERROR 1396 (HY000): Operation CREATE ROLE failed for 'a','b','c'
```

Failed `CREATE` or `DROP` operations, for both users and roles, produce the same error code.

`PUBLIC` and `NONE` are reserved, and cannot be used as role names. `NONE` is used to unset a role and `PUBLIC` has a special use in other systems, such as Oracle, so is reserved for compatibility purposes.

For valid identifiers to use as role names, see Identifier Names.

WITH ADMIN

The optional `WITH ADMIN` clause determines whether the current user, the current role or another user or role has use of the newly created role. If the clause is omitted, `WITH ADMIN CURRENT_USER` is treated as the default, which means that the current user will be able to GRANT this role to users. ###### Example: Using WITH ADMIN

The WITH ADMIN option allows a specific user or role to manage (grant or revoke) the newly created role. For example:

```
CREATE ROLE developer WITH ADMIN lorinda@localhost;
```

Here, the `developer` role is created, and the user `lorinda@localhost` is given permission to grant or revoke this role to other users.

If another user without administrative privileges attempts to grant the role, the operation fails:

```
GRANT developer TO ian@localhost;
-- ERROR: Access denied
```

However, when executed by `lorinda@localhost`, the operation succeeds:

```
GRANT developer TO ian@localhost;
```

OR REPLACE

If the optional `OR REPLACE` clause is used, it acts as a shortcut for:

```
DROP ROLE IF EXISTS name;
CREATE ROLE name ...;
```

IF NOT EXISTS

When the `IF NOT EXISTS` clause is used, MariaDB will return a warning instead of an error if the specified role already exists. Cannot be used together with the `OR REPLACE` clause.

Examples

```
CREATE ROLE journalist;

CREATE ROLE developer WITH ADMIN lorinda@localhost;
```

Granting the role to another user. Only user `lorinda@localhost` has permission to grant the `developer` role:

```
SELECT USER();
```

```
+-------------------+
| USER()            |
+-------------------+
| henning@localhost |
+-------------------+
...
```

```
GRANT developer TO ian@localhost;
```

```
Access denied for user 'henning'@'localhost'
```

```
SELECT USER();
```

```
+-------------------+
| USER()            |
+-------------------+
| lorinda@localhost |
+-------------------+
```

```
GRANT m_role TO ian@localhost;
```

The `OR REPLACE` and `IF NOT EXISTS` clauses. The `journalist` role already exists:

```
CREATE ROLE journalist;
ERROR 1396 (HY000): Operation CREATE ROLE failed for 'journalist'

CREATE OR REPLACE ROLE journalist;
Query OK, 0 rows affected (0.00 sec)

CREATE ROLE IF NOT EXISTS journalist;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
SHOW WARNINGS;
```

```
+-------+------+-----------------------------------------------------+
| Level | Code | Message                                             |
+-------+------+-----------------------------------------------------+
| Note  | 1975 | Can't create role 'journalist'; it already exists |
+-------+------+-----------------------------------------------------+
```

See Also

- Identifier Names
- Roles Overview
- DROP ROLE

This page is licensed: CC BY-SA / Gnu FDL

CREATE USER

Syntax

```
CREATE [OR REPLACE] USER [IF NOT EXISTS]
 user_specification [,user_specification ...]
  [REQUIRE {NONE | tls_option [[AND] tls_option ...] }]
  [WITH resource_option [resource_option ...] ]
  [lock_option] [password_option]

user_specification:
  username [authentication_option]

authentication_option:
  IDENTIFIED BY 'password'
  | IDENTIFIED BY PASSWORD 'password_hash'
  | IDENTIFIED {VIA|WITH} authentication_rule [OR authentication_rule  ...]

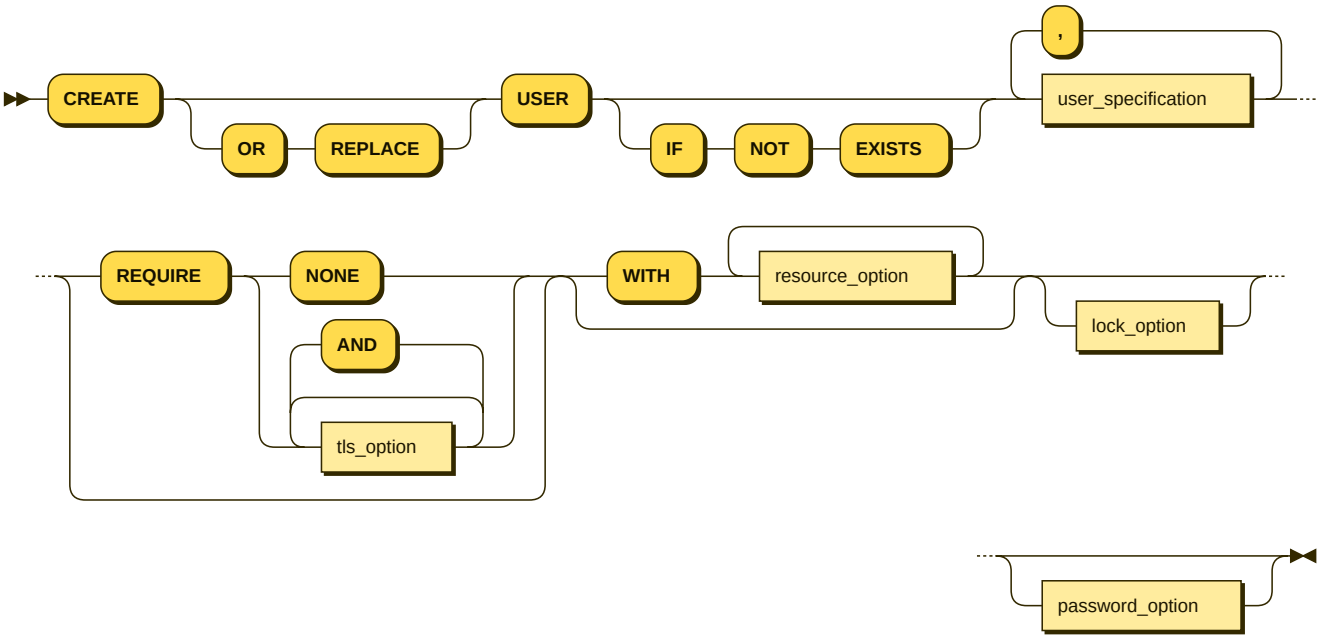
authentication_rule:
    authentication_plugin
  | authentication_plugin {USING|AS} 'authentication_string'
  | authentication_plugin {USING|AS} PASSWORD('password')

tls_option:
  SSL
  | X509
  | CIPHER 'cipher'
  | ISSUER 'issuer'
  | SUBJECT 'subject'

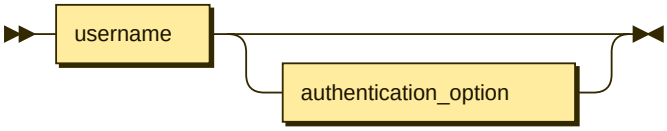
resource_option:
  MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count
  | MAX_STATEMENT_TIME time

password_option:
  PASSWORD EXPIRE
  | PASSWORD EXPIRE DEFAULT
  | PASSWORD EXPIRE NEVER
  | PASSWORD EXPIRE INTERVAL N DAY

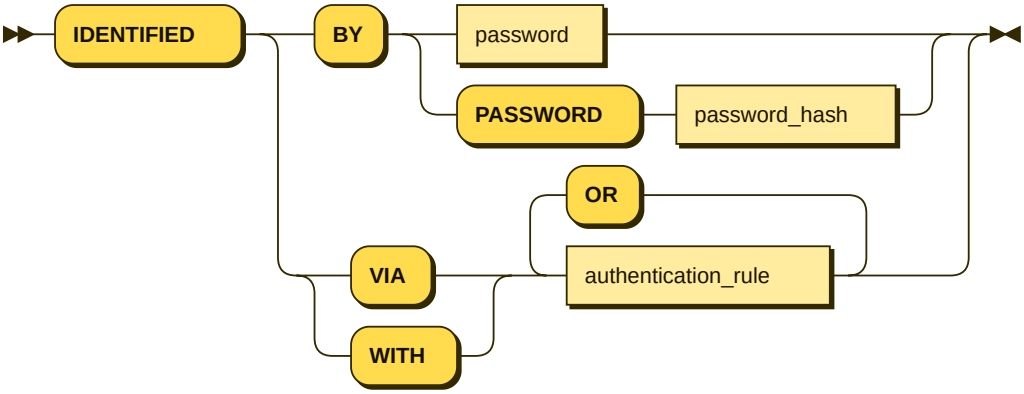
lock_option:
    ACCOUNT LOCK
  | ACCOUNT UNLOCK
```



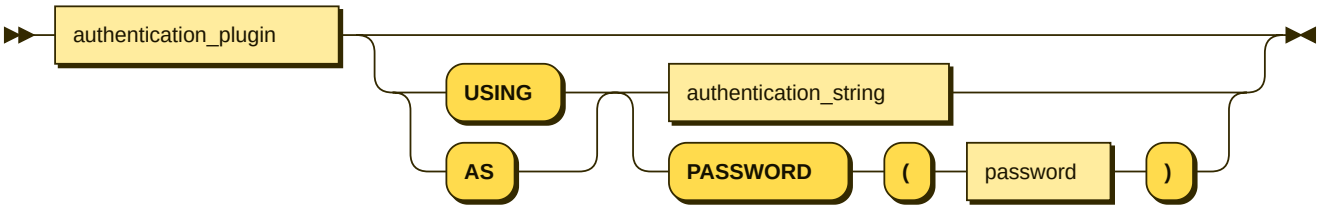
Railroad diagram of CREATE USER — equivalent to the BNF above



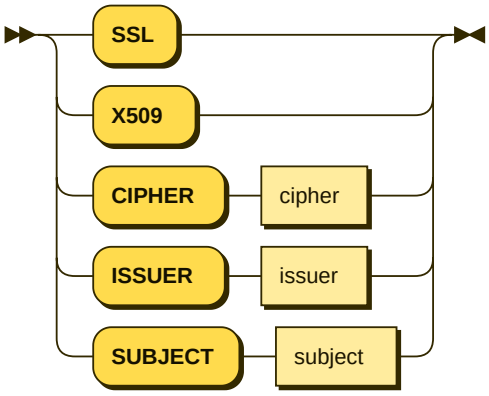
Railroad diagram of user_specification



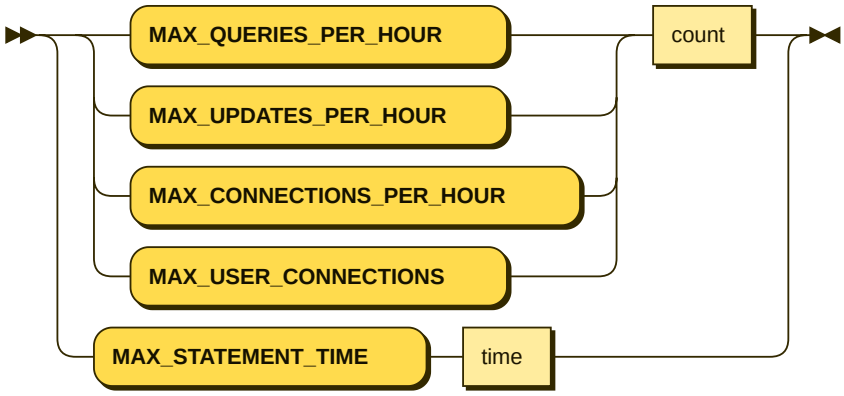
Railroad diagram of authentication_option



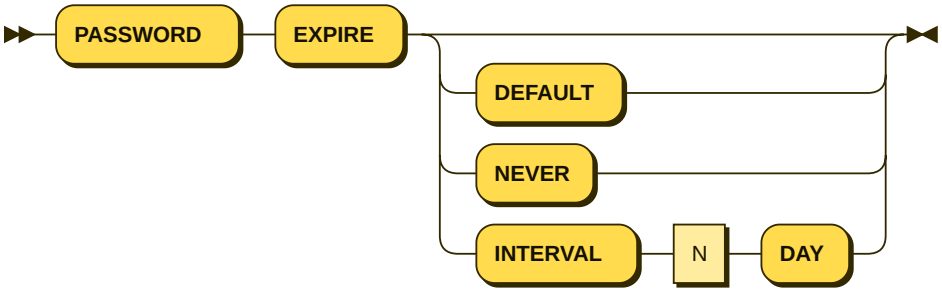
Railroad diagram of authentication_rule



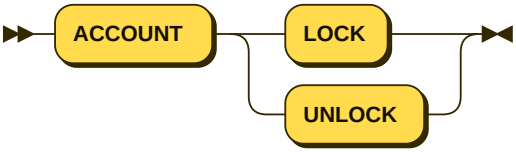
Railroad diagram of tls_option



Railroad diagram of resource_option



Railroad diagram of password_option



Railroad diagram of lock_option

Description

The `CREATE USER` statement creates new MariaDB accounts. To use it, you must have the global CREATE USER privilege or the INSERT privilege for the mysql database.

> **CURRENT**
>
> For each account, `CREATE USER` creates a new row in the mysql.user view (and the underlying mysql.global_priv table) that has no privileges.
>
> **< 10.4**
>
> For each account, `CREATE USER` creates a new row in mysql.user table that has no privileges.

If any of the specified accounts, or any permissions for the specified accounts, already exist, then the server returns `ERROR 1396 (HY000)`. If an error occurs, `CREATE USER` will still create the accounts that do not result in an error. Only one error is produced for all users which have not been created:

```
ERROR 1396 (HY000):
  Operation CREATE USER failed for 'u1'@'%','u2'@'%'
```

CREATE USER, DROP USER, CREATE ROLE, and DROP ROLE all produce the same error code when they fail.

See Account Names below for details on how account names are specified.

One can also create users with GRANT if SQL_MODE does not have NO_AUTO_CREATE_USER set. `NO_AUTO_CREATE_USER` is set by default.

OR REPLACE

If the optional `OR REPLACE` clause is used, it is basically a shortcut for:

```
DROP USER IF EXISTS name;
CREATE USER name ...;
```

For example:

```
CREATE USER foo2@test IDENTIFIED BY 'password';
ERROR 1396 (HY000): Operation CREATE USER failed for 'foo2'@'test'

CREATE OR REPLACE USER foo2@test IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.00 sec)
```

IF NOT EXISTS

When the `IF NOT EXISTS` clause is used, MariaDB will return a warning instead of an error if the specified user already exists.

For example:

```
CREATE USER foo2@test IDENTIFIED BY 'password';
ERROR 1396 (HY000): Operation CREATE USER failed for 'foo2'@'test'

CREATE USER IF NOT EXISTS foo2@test IDENTIFIED BY 'password';
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------------------+
| Level | Code | Message                                             |
+-------+------+-----------------------------------------------------+
| Note  | 1973 | Can't create user 'foo2'@'test'; it already exists |
+-------+------+-----------------------------------------------------+
```

Authentication Options

If more than one authentication mechanism is declared using the `OR` keyword, the mechanisms are attempted in the order they are declared in the `CREATE USER` statement. As soon as one of the authentication mechanisms is successful, authentication is complete. If none of them is successful, the authentication has failed.

IDENTIFIED BY 'password'

The optional `IDENTIFIED BY` clause can be used to provide an account with a password. The password should be specified in plain text. It will be hashed by the PASSWORD function prior to being stored in the mysql.user/mysql.global_priv_table table.

For example, if our password is `mariadb`, then we can create the user with:

```
CREATE USER foo2@test IDENTIFIED BY 'mariadb';
```

If you do not specify a password with the `IDENTIFIED BY` clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

The only authentication plugins that this clause supports are mysql_native_password and mysql_old_password.

IDENTIFIED BY PASSWORD 'password_hash'

The optional `IDENTIFIED BY PASSWORD` clause can be used to provide an account with a password that has already been hashed. The password should be specified as a hash that was provided by the PASSWORD function. It will be stored in the mysql.user/mysql.global_priv_table table as-is.

For example, if our password is `mariadb`, then we can find the hash with:

```
SELECT PASSWORD('mariadb');
+-------------------------------------------+
| PASSWORD('mariadb')                       |
+-------------------------------------------+
| *54958E764CE10E50764C2EECBB71D01F08549980 |
+-------------------------------------------+
1 row in set (0.00 sec)
```

And then we can create a user with the hash:

```
CREATE USER foo2@test IDENTIFIED BY PASSWORD '*54958E764CE10E50764C2EECBB71D01F08549980';
```

If you do not specify a password with the `IDENTIFIED BY` clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

The only authentication plugins that this clause supports are mysql_native_password and mysql_old_password.

IDENTIFIED {VIA|WITH} authentication_plugin

The optional `IDENTIFIED VIA authentication_plugin` allows you to specify that the account should be authenticated by a specific authentication plugin. The plugin name must be an active authentication plugin as per SHOW PLUGINS. If it doesn't show up in that output, then you will need to install it with INSTALL PLUGIN or INSTALL SONAME.

`VIA` and `WITH` are synonyms.

For example, this could be used with the PAM authentication plugin:

```
CREATE USER foo2@test IDENTIFIED VIA pam;
```

Some authentication plugins allow additional arguments to be specified after a `USING` or `AS` keyword. For example, the PAM authentication plugin accepts a service name:

```
CREATE USER foo2@test IDENTIFIED VIA pam USING 'mariadb';
```

The exact meaning of the additional argument would depend on the specific authentication plugin.

The `USING` or `AS` keyword can also be used to provide a plain-text password to a plugin if it's provided as an argument to the PASSWORD() function. This is only valid for authentication plugins that have implemented a hook for the PASSWORD() function. For example, the ed25519 authentication plugin supports this:

```
CREATE USER safe@'%' IDENTIFIED VIA ed25519 USING PASSWORD('secret');
```

One can specify many authentication plugins, they all work as alternatives ways of authenticating a user:

```
CREATE USER safe@'%' IDENTIFIED VIA ed25519 USING PASSWORD('secret') OR unix_socket;
```

By default, when you create a user without specifying an authentication plugin, MariaDB uses the mysql_native_password plugin.

TLS Options

CURRENT

MariaDB allows you to encrypt data in transit between the server and clients using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix ssl_, but internally, MariaDB only supports its secure successors.

< 11.4

By default, MariaDB transmits data between the server and clients **without encrypting it**. This is generally acceptable when the server and client run on the same host or in networks where security is guaranteed through other means. However, in cases where the server and client exist on separate networks or they are in a high-risk network, the lack of encryption does introduce security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

To mitigate this concern, MariaDB allows you to encrypt data in transit between the server and clients using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix ssl_, but internally, MariaDB only supports its secure successors.

See Secure Connections Overview for more information about how to determine whether your MariaDB server has TLS support.

You can set certain TLS-related restrictions for specific user accounts. For instance, you might use this with user accounts that require access to sensitive data while sending it across networks that you do not control. These restrictions can be enabled for a user account with the CREATE USER, ALTER USER, or GRANT statements. The following options are available:

Option	Description
REQUIRE NONE	TLS is not required for this account, but can still be used.
REQUIRE SSL	The account must use TLS, but no valid X509 certificate is required. This option cannot be combined with other TLS options.
REQUIRE X509	The account must use TLS and must have a valid X509 certificate. This option implies REQUIRE SSL. This option cannot be combined with other TLS options.
REQUIRE ISSUER 'issuer'	The account must use TLS and must have a valid X509 certificate. Also, the Certificate Authority must be the one specified via the string issuer. This option implies REQUIRE X509. This option can be combined with the SUBJECT, and CIPHER options in any order.
REQUIRE SUBJECT 'subject'	The account must use TLS and must have a valid X509 certificate. Also, the certificate's Subject must be the one specified via the string subject. This option implies REQUIRE X509. This option can be combined with the ISSUER, and CIPHER options in any order.

Option	Description
REQUIRE CIPHER 'cipher'	The account must use TLS, but no valid X509 certificate is required. Also, the encryption used for the connection must use a specific cipher method specified in the string cipher. This option implies REQUIRE SSL. This option can be combined with the ISSUER, and SUBJECT options in any order.

The `REQUIRE` keyword must be used only once for all specified options, and the `AND` keyword can be used to separate individual options, but it is not required.

For example, you can create a user account that requires these TLS options with the following:

```
CREATE USER 'alice'@'%'
 REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=Portland'
 AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter Parker/emailAddress=p.parker@marvel.com'
 AND CIPHER 'SHA-DES-CBC3-EDH-RSA';
```

If any of these options are set for a specific user account, then any client who tries to connect with that user account will have to be configured to connect with TLS.

See Securing Connections for Client and Server for information on how to enable TLS on the client and server.

Resource Limit Options

It is possible to set per-account limits for certain server resources. The following table shows the values that can be set per account:

Limit Type	Description
MAX_QUERIES_PER_HOUR	Number of statements that the account can issue per hour (including updates)
MAX_UPDATES_PER_HOUR	Number of updates (not queries) that the account can issue per hour
MAX_CONNECTIONS_PER_HOUR	Number of connections that the account can start per hour
MAX_USER_CONNECTIONS	Number of simultaneous connections that can be accepted from the same account; if it is 0, max_connections will be used instead; if max_connections is 0, there is no limit for this account's simultaneous connections.
MAX_STATEMENT_TIME	Timeout, in seconds, for statements executed by the user. See also Aborting Statements that Exceed a Certain Time to Execute.

If any of these limits are set to `0`, then there is no limit for that resource for that user.

Here is an example showing how to create a user with resource limits:

```
CREATE USER 'someone'@'localhost' WITH
    MAX_USER_CONNECTIONS 10
    MAX_QUERIES_PER_HOUR 200;
```

The resources are tracked per account, which means `'user'@'server'`; not per user name or per connection.

The count can be reset for all users using FLUSH USER_RESOURCES, FLUSH PRIVILEGES or mariadb-admin reload.

Per account resource limits are stored in the user table, in the mysql database. Columns used for resources limits are named `max_questions`, `max_updates`, `max_connections` (for `MAX_CONNECTIONS_PER_HOUR`), and `max_user_connections` (for `MAX_USER_CONNECTIONS`).

Account Names

Account names have both a user name component and a host name component, and are specified as `'user_name'@'host_name'`.

The user name and host name may be unquoted, quoted as strings using double quotes (`"`) or single quotes (`'`), or quoted as identifiers using backticks (```). You must use quotes when using special characters (such as a hyphen) or wildcard characters. If you quote, you must quote the user name and host name separately (for example `'user_name'@'host_name'`).

Host Name Component

If the host name is not provided, it is assumed to be `'%'`.

Host names may contain the wildcard characters `%` and `_`. They are matched as if by the LIKE clause. If you need to use a wildcard character literally (for example, to match a domain name with an underscore), prefix the character with a backslash. See `LIKE` for more information on escaping wildcard characters.

Before MariaDB 10.4.6, when multiple host patterns could match a connecting client, the sort order among wildcard patterns was determined only by the position of the first wildcard character. This approach often produced incorrect results or made the outcome dependent on insertion order.

Starting with MariaDB 10.4.6 (MDEV-14735), the matching algorithm correctly ranks host patterns by specificity, the number of hosts a pattern can match, ensuring deterministic and accurate privilege resolution.

Host name matches are case-insensitive. Host names can match either domain names or IP addresses. Use `'localhost'` as the host name to allow only local client connections. On Linux, the loopback interface (127.0.0.1) will not match 'localhost' as it is not considered a local connection: this means that only connections via UNIX-domain sockets will match 'localhost'.

You can use a netmask to match a range of IP addresses using `'base_ip/netmask'` as the host name. A user with an IP address *ip_addr* will be allowed to connect if the following condition is true:

```
ip_addr & netmask = base_ip
```

For example, given a user:

```
CREATE USER 'maria'@'247.150.130.0/255.255.255.0';
```

the IP addresses satisfying this condition range from 247.150.130.0 to 247.150.130.255.

Using `255.255.255.255` is equivalent to not using a netmask at all. Netmasks cannot be used for IPv6 addresses.

Note that the credentials added when creating a user with the `'%'` wildcard host will not grant access in all cases. For example, some systems come with an anonymous localhost user, and when connecting from localhost this will take precedence.

Before MariaDB 10.6, the host name component could be up to 60 characters in length. Starting from MariaDB 10.6, it can be up to 255 characters.

User Name Component

User names must match exactly, including case. A user name that is empty is known as an anonymous account and is allowed to match a login attempt with any user name component. These are described more in the next section.

For valid identifiers to use as user names, see Identifier Names.

It is possible for more than one account to match when a user connects. MariaDB selects the first matching account after sorting according to the following criteria:

- Accounts with an exact host name are sorted before accounts using a wildcard in the host name. Host names using a netmask are considered to be exact for sorting.
- Accounts with a wildcard in the host name are sorted by specificity: a hostname that can match fewer hosts is considered more specific and is sorted first. Exact hostnames (no wildcards) are most specific; a bare `%` (matches any host) is least specific. Among patterns with wildcards, those that can match a narrower set of hosts sort before those that match a broader set. For example, `%.foo.bar` sorts before `%.bar` because it matches fewer hosts.

  Starting with MariaDB 10.4.6, this ordering is handled correctly by the improved `get_sort()` algorithm (MDEV-14735). In earlier versions, sorting was based only on the length of the prefix before the first wildcard, which led to indeterminate ordering for patterns such as `%.bar` versus `%.foo.bar`.
- Accounts with a non-empty user name sort before accounts with an empty user name.
- Accounts with an empty user name are sorted last. As mentioned previously, these are known as anonymous accounts. These are described more in the next section.

The following table shows a list of example account as sorted by these criteria:

```
+---------+-------------+
| User    | Host        |
+---------+-------------+
| joffrey | 192.168.0.3 |
|         | 192.168.0.% |
| joffrey | 192.168.%   |
|         | 192.168.%   |
+---------+-------------+
```

Once connected, you only have the privileges granted on a particular object to the account that matched, not all accounts that could have matched. For example, consider the following commands:

```
CREATE USER 'joffrey'@'192.168.0.3';
CREATE USER 'joffrey'@'%';
GRANT SELECT ON test.t1 TO 'joffrey'@'192.168.0.3';
GRANT INSERT ON test.t1 TO 'joffrey'@'%';
```

If you connect as joffrey from `192.168.0.3`, you will have the `SELECT` privilege on the table `test.t1`, but not INSERT. If you connect as joffrey from any other IP address, you will have the `INSERT` privilege on the table `test.t1`, but not `SELECT`.

Usernames can be up to 80 characters long before 10.6 and starting from 10.6 it can be 128 characters long.

Starting with MariaDB 10.4.6, patterns are ranked according to how many hosts they can match; those that match fewer hosts are considered more specific and take precedence in the ordering. The following example shows how domain-name wildcard patterns are sorted by specificity.

```
+---------+-------------+
| User    | Host        |
+---------+-------------+
| alice   | db.foo.bar  |  <- exact, matched first
| alice   | %.foo.bar   |  <- more specific wildcard
| alice   | %.bar       |  <- less specific wildcard
| alice   | %           |  <- least specific, matched last
+---------+-------------+
```

Note: The ordering of wildcard host patterns shown above reflects the behavior introduced in MariaDB 10.4.6 (MDEV-14735). In earlier versions, `%.foo.bar` and `%.bar` could sort indeterminately because the algorithm only compared the length of the prefix before the first wildcard character, both patterns have an empty prefix, so their relative order was undefined and could depend on insertion order in `mysql.user`.

Anonymous Accounts

Anonymous accounts are accounts where the user name portion of the account name is empty. These accounts act as special catch-all accounts. If a user attempts to log into the system from a host, and an anonymous account exists with a host name portion that matches the user's host, then the user will log in as the anonymous account if there is no more specific account match for the user name that the user entered.

For example, here are some anonymous accounts:

```
CREATE USER ''@'localhost';
CREATE USER ''@'192.168.0.3';
```

Fixing a Legacy Default Anonymous Account

On some systems, the mysql.db table has some entries for the `''@'%'` anonymous account by default. Unfortunately, there is no matching entry in the mysql.user/mysql.global_priv_table table, which means that this anonymous account doesn't exactly exist, but it does have privileges–usually on the default `test` database created by mariadb-install-db. These account-less privileges are a legacy that is leftover from a time when MySQL's privilege system was less advanced.

This situation means that you will run into errors if you try to create a `''@'%'` account. For example:

```
CREATE USER ''@'%';
ERROR 1396 (HY000): Operation CREATE USER failed for ''@'%'
```

The fix is to DELETE the row in the mysql.db table and then execute FLUSH PRIVILEGES:

```
DELETE FROM mysql.db WHERE User='' AND Host='%';
FLUSH PRIVILEGES;
```

Note that `FLUSH PRIVILEGES` is only needed if one modifies the `mysql` tables directly. It is not needed when using `CREATE USER`, `DROP USER`, `GRANT` etc.

And then the account can be created:

```
CREATE USER ''@'%';
Query OK, 0 rows affected (0.01 sec)
```

See MDEV-13486 for more information.

Password Expiry

Besides automatic password expiry, as determined by default_password_lifetime, password expiry times can be set on an individual user basis, overriding the global setting, for example:

```
CREATE USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY;
```

See User Password Expiry for more details.

Account Locking

Account locking permits privileged administrators to lock/unlock user accounts. No new client connections will be permitted if an account is locked (existing connections are not affected). For example:

```
CREATE USER 'marijn'@'localhost' ACCOUNT LOCK;
```

See Account Locking for more details.

CURRENT

The *lock_option* and *password_option* clauses can occur in either order.

<10.4.7, <10.5.8

Prior to MariaDB 10.4.7 and MariaDB 10.5.8, the *lock_option* must be placed before the *password_option*.

From MariaDB 10.4.7 and MariaDB 10.5.8, the *lock_option* and *password_option* clauses can occur in either order.

See Also

- GRANT
- ALTER USER
- RENAME USER
- DROP USER
- CREATE ROLE
- SET PASSWORD
- SHOW CREATE USER
- Troubleshooting Connection Issues
- Authentication from MariaDB 10.4
- Identifier Names
- mysql.user table
- mysql.global_priv_table
- Password Validation Plugins - permits the setting of basic criteria for passwords
- Authentication Plugins - allow various authentication methods to be used, and new ones to be developed.

This page is licensed: GPLv2, originally from fill_help_tables.sql

DENY

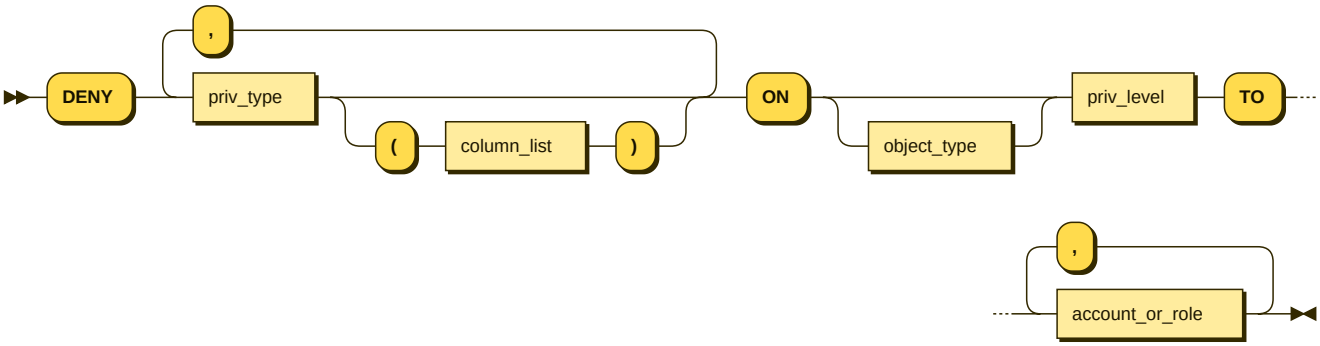
Note:

`DENY` and `REVOKE DENY` were added in MariaDB 13.1.

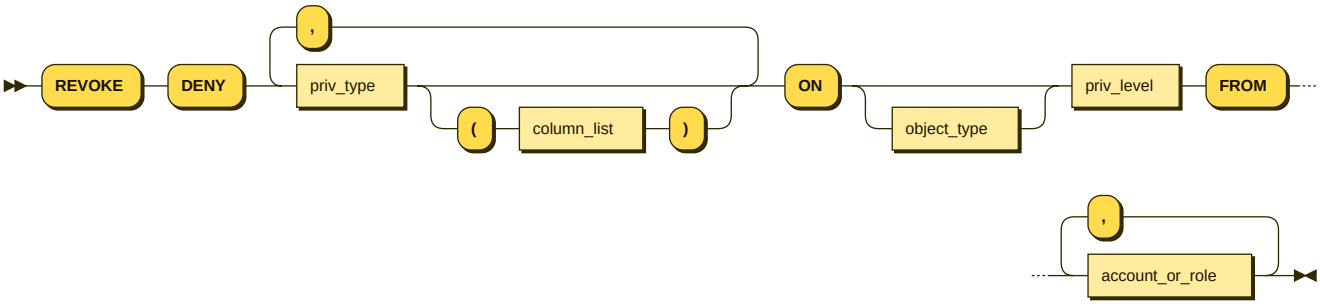
Syntax

```
/* 1. Denying Privileges */
DENY
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    TO account_or_role [, account_or_role] ...

/* 2. Removing a Deny */
REVOKE DENY
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM account_or_role [, account_or_role] ...
```



Railroad diagram of DENY — equivalent to the BNF above



Railroad diagram of REVOKE DENY — equivalent to the BNF above

`account_or_role`, `priv_type`, `object_type`, and `priv_level` accept the same values as they do for GRANT. Sub-rule diagrams are not repeated here — see GRANT for those productions.

Description

`DENY` records a privilege that an account must never have. Where GRANT adds a privilege and REVOKE takes a granted privilege away, `DENY` stores a separate, negative entry that is consulted on every privilege check. As long as the deny exists, the privilege is refused, no matter how many `GRANT` statements say otherwise:

```
GRANT SELECT ON *.* TO alice;
DENY SELECT ON secrets.payroll TO alice;
```

`alice` can now read every table on the server except `secrets.payroll`.

The typical use is the “everything except” case that `GRANT` and `REVOKE` alone handle awkwardly: grant broadly, then carve out the objects or columns that must stay out of reach.

Use `REVOKE DENY` to remove a deny. Nothing else restores the privilege — there is no privilege that lets an account bypass a deny, and a later `GRANT` does not cancel one.

To issue `DENY` or `REVOKE DENY`, you need the `GRANT OPTION` privilege plus the privileges you are denying, exactly as you would to `GRANT` them.

`DENY` applies to privileges only. It cannot be used to deny a role or proxy access, so there is no `DENY role` or `DENY PROXY` form.

Deny Levels

A deny is recorded at the level named by `priv_level`, matching the privilege levels used by `GRANT`:

Level	Syntax	Example
Global	`*.*`	`DENY SELECT ON *.* TO alice;`
Database	`db_name.*`	`DENY INSERT ON hr.* TO alice;`
Table	`db_name.tbl_name`	`DENY DELETE ON hr.staff TO alice;`

Level	Syntax	Example
Column	`priv_type (column)` on a table	`DENY SELECT (salary) ON hr.staff TO alice;`
Routine	`PROCEDURE`, `FUNCTION`, `PACKAGE`, or `PACKAGE BODY`	`DENY EXECUTE ON FUNCTION hr.bonus TO alice;`

Global denies cover administrative privileges as well as data privileges. Denying `RELOAD`, `SHUTDOWN`, `PROCESS`, `FILE`, or `CONNECTION ADMIN` on `*.*` blocks the matching operations even for an account that holds `ALL PRIVILEGES`:

```
GRANT ALL PRIVILEGES ON *.* TO opsuser@localhost;
DENY SHUTDOWN ON *.* TO opsuser@localhost;
```

```
SHUTDOWN;
ERROR 42000: Access denied; you need (at least one of) the SHUTDOWN privilege(s) for this operation
```

As with `GRANT`, a routine deny must state the routine type. Leaving it out makes MariaDB read the name as a table:

```
DENY EXECUTE ON hr.bonus TO alice;
ERROR 42000: Illegal GRANT/REVOKE command; please consult the manual to see which privileges can be used
```

Precedence

Two rules cover every case:

- **A deny beats a grant.** If a privilege is denied anywhere in the hierarchy that applies to an object, the privilege is refused.
- **Order does not matter.** `GRANT` then `DENY` and `DENY` then `GRANT` give the same result.

A deny also propagates downwards: a global deny masks database, table, and column grants; a database deny masks table and column grants; a table deny masks column grants.

```
DENY SELECT ON hr.* TO alice;
GRANT SELECT ON hr.staff TO alice;
```

```
SELECT * FROM hr.staff;
ERROR 42000: SELECT command denied to user 'alice'@'localhost' for table `hr`.`staff`
```

A deny only affects the privileges it names. Other privileges on the same object are unaffected:

```
GRANT SELECT, INSERT ON hr.staff TO alice;
DENY SELECT ON hr.staff TO alice;
```

`alice` can still insert into `hr.staff`; only reading it is refused.

Denies also reach statements that read a table implicitly. An `UPDATE ... WHERE` needs `SELECT` on the columns in the `WHERE` clause, so a `SELECT` deny stops it, while the same `UPDATE` without a `WHERE` clause succeeds.

A deny follows the account into definer context as well. A view defined with `SQL SECURITY DEFINER` fails if the definer is denied the privileges the view needs, even when the invoker holds them.

Roles and PUBLIC

Denies can be granted to a role, and they travel through the role graph the same way privileges do. A deny on any role an account holds wins over a grant on any other role, at any depth:

```
CREATE ROLE reader, blocked, combined;
GRANT SELECT ON hr.staff TO reader;
DENY SELECT ON hr.staff TO blocked;
GRANT reader TO combined;
GRANT blocked TO combined;
GRANT combined TO alice@localhost;
```

With `combined` active, `alice` cannot read `hr.staff`.

Denying to PUBLIC blocks a privilege for every account on the server:

```
DENY SELECT ON hr.staff TO PUBLIC;
```

Removing a Deny

`REVOKE DENY` removes the named privileges from an existing deny entry at that exact level. It fails if there is no matching deny, even when a positive grant exists:

```
GRANT INSERT ON hr.* TO alice;
REVOKE DENY INSERT ON hr.* FROM alice;
ERROR 42000: There is no such grant defined for user 'alice' on host '%'
```

`REVOKE DENY` matches by level, so a table-level revoke does not touch column-level denies on the same table. Remove those by naming the columns:

```
DENY DELETE ON hr.staff TO alice;
DENY SELECT (salary) ON hr.staff TO alice;
REVOKE DENY ALL PRIVILEGES ON hr.staff FROM alice;   -- clears the DELETE deny only
REVOKE DENY SELECT (salary) ON hr.staff FROM alice;  -- clears the column deny
```

REVOKE ALL PRIVILEGES, GRANT OPTION clears an account's denies along with its grants:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM alice;
```

Dropping the object a deny points at also removes the deny. Dropping a stored procedure, for example, deletes the deny entries recorded against it.

Viewing Denies

SHOW GRANTS reports denies as separate `DENY` lines beside the `GRANT` lines:

```
SHOW GRANTS FOR alice@localhost;
+-------------------------------------------------------------+
| Grants for alice@localhost                                  |
+-------------------------------------------------------------+
| GRANT USAGE ON *.* TO `alice`@`localhost`                    |
| DENY SELECT ON *.* TO `alice`@`localhost`                    |
| DENY INSERT ON `hr`.* TO `alice`@`localhost`                 |
| GRANT SELECT ON `hr`.`staff` TO `alice`@`localhost`          |
| DENY UPDATE (`salary`) ON `hr`.`staff` TO `alice`@`localhost` |
+-------------------------------------------------------------+
```

`DENY` lines are only visible to a connection that has the `SELECT` privilege on the `mysql` database. Accounts without it see their own `GRANT` lines but none of their denies, so a deny is not self-advertising. The same applies to `SHOW GRANTS FOR CURRENT_ROLE`, `SHOW GRANTS FOR role`, and `SHOW GRANTS FOR PUBLIC`.

> **Note:**
>
> The visibility check is itself subject to denies. An account with `SELECT` on `mysql.*` but a deny on `mysql.global_priv` sees no denies, and cannot run `SHOW GRANTS` for another account.

Effect on Metadata

Because metadata visibility follows privileges, a deny also hides what the account may not read:

- A global `SELECT` deny hides databases from SHOW DATABASES.
- A table-level `SELECT` deny hides the table from SHOW TABLES, `SHOW TABLE STATUS`, `INFORMATION_SCHEMA.TABLES`, and `INFORMATION_SCHEMA.COLUMNS`, and makes `SHOW CREATE TABLE`, `SHOW COLUMNS`, and `SHOW INDEX` fail.
- A column-level `SELECT` deny hides the denied columns from `SHOW COLUMNS` and `INFORMATION_SCHEMA.COLUMNS`, and drops index rows that use them from `INFORMATION_SCHEMA.STATISTICS` and `INFORMATION_SCHEMA.KEY_COLUMN_USAGE`. The remaining columns stay visible, and `SHOW CREATE TABLE` still prints the full table definition.
- A routine-level `EXECUTE` deny hides that routine from `INFORMATION_SCHEMA.ROUTINES` while leaving the other routines in the database visible.

If every privilege an account holds in a database is denied, USE is refused as well:

```
USE hr;
ERROR 42000: Access denied for user 'alice'@'localhost' to database 'hr'
```

Storage and Persistence

Denies are stored in the `Priv` JSON document of the mysql.global_priv table, under a `denies` key, and not in `mysql.db`, `mysql.tables_priv`, or `mysql.columns_priv`. Each array element carries the level in `type`, the object names, and the denied privilege bits:

```
SELECT User, Host, JSON_PRETTY(JSON_EXTRACT(Priv, '$.denies')) AS denies
  FROM mysql.global_priv WHERE User = 'alice';
+-------+------+---------------------------+
| User  | Host | denies                    |
+-------+------+---------------------------+
| alice | %    | [                         |
|       |      |     {                     |
|       |      |         "type": "column", |
|       |      |         "db": "hr",       |
|       |      |         "table": "staff", |
|       |      |         "column": "salary",|
|       |      |         "bits": 1         |
|       |      |     }                     |
|       |      | ]                         |
+-------+------+---------------------------+
```

`type` is one of `global`, `db`, `table`, `column`, `function`, `procedure`, `package`, or `package body`. `bits` uses the same privilege bit values as the `access` field, listed in Mapping the access Field Values to Grants.

Denies survive FLUSH PRIVILEGES and a server restart, and replicate to replicas like any other account management statement.

Column names and routine names in denies are matched case insensitively. Database and table names follow lower_case_table_names: with the default `0` on Linux, `DENY SELECT ON DbCase.T` and `DENY SELECT ON dbcase.t` are two distinct denies.

> **Warning:**
>
> Editing the `denies` array directly with `UPDATE` is not supported. A malformed entry is skipped at startup and on `FLUSH PRIVILEGES`, with a `Malformed DENY entry in mysql.global_priv` warning in the error log — leaving the privilege silently allowed. Always use `DENY` and `REVOKE DENY`.

Comparison with REVOKE

`REVOKE` and `DENY` are not interchangeable:

	`REVOKE`	`DENY`
What it does	Removes a granted privilege	Records a permanent block
Can a later `GRANT` restore access?	Yes	No
Effect of a broader grant	A grant at a higher level still applies	Refused regardless of level
How to undo	`GRANT`	`REVOKE DENY`

If an account holds `SELECT ON *.*` and you revoke `SELECT` on one table, the global grant still covers that table. A deny is what actually excludes it.

Examples

Grant broad access, then exclude one table:

```
CREATE USER analyst@'%' IDENTIFIED BY 'secret';
GRANT SELECT ON sales.* TO analyst@'%';
DENY SELECT ON sales.customer_pii TO analyst@'%';
```

Hide two columns while leaving the rest of the table readable:

```
GRANT SELECT ON hr.staff TO analyst@'%';
DENY SELECT (salary, ssn) ON hr.staff TO analyst@'%';
```

```
SELECT id, name FROM hr.staff;   -- succeeds
SELECT * FROM hr.staff;
ERROR 42000: SELECT command denied to user 'analyst'@'%' for column 'salary' in table 'staff'
```

Stop an account from writing to a column it can otherwise insert into:

```
GRANT SELECT, INSERT ON hr.audit_log TO app@localhost;
DENY INSERT (modified_by) ON hr.audit_log TO app@localhost;
```

Block a function while leaving a procedure callable:

```
GRANT EXECUTE ON PROCEDURE hr.refresh TO app@localhost;
GRANT EXECUTE ON FUNCTION hr.bonus TO app@localhost;
DENY EXECUTE ON FUNCTION hr.bonus TO app@localhost;
```

Lift the deny again:

```
REVOKE DENY EXECUTE ON FUNCTION hr.bonus FROM app@localhost;
```

See Also

- GRANT
- REVOKE
- SHOW GRANTS
- Roles
- mysql.global_priv Table

This page is licensed: CC BY-SA / Gnu FDL

DROP ROLE

Syntax

```
DROP ROLE [IF EXISTS] role_name [,role_name ...]
```

Description

The `DROP ROLE` statement removes one or more MariaDB roles. To use this statement, you must have the global CREATE USER privilege or the DELETE privilege for the mysql database.

`DROP ROLE` does not disable roles for connections which selected them with SET ROLE. If a role has previously been set as a default role, `DROP ROLE` does not remove the record of the default role from the mysql.user table. If the role is subsequently recreated and granted, it will again be the user's default. Use SET DEFAULT ROLE NONE to explicitly remove this.

If any of the specified user accounts do not exist, `ERROR 1396 (HY000)` results. If an error occurs, `DROP ROLE` will still drop the roles that do not result in an error. Only one error is produced for all roles which have not been dropped:

```
ERROR 1396 (HY000): Operation DROP ROLE failed for 'a','b','c'
```

Failed `CREATE` or `DROP` operations, for both users and roles, produce the same error code.

IF EXISTS

If the `IF EXISTS` clause is used, MariaDB will return a warning instead of an error if the role does not exist.

Examples

```
DROP ROLE journalist;
```

The same thing using the optional `IF EXISTS` clause:

```
DROP ROLE journalist;
ERROR 1396 (HY000): Operation DROP ROLE failed for 'journalist'

DROP ROLE IF EXISTS journalist;
Query OK, 0 rows affected, 1 warning (0.00 sec)

Note (Code 1975): Can't drop role 'journalist'; it doesn't exist
```

See Also

- Roles Overview
- CREATE ROLE

This page is licensed: CC BY-SA / Gnu FDL

DROP USER

Syntax

CURRENT

```
DROP USER [IF EXISTS] user_name [, user_name] ... [FORCE]
```

< 12.1

```
DROP USER [IF EXISTS] user_name [, user_name] ...
```

Description

The `DROP USER` statement removes one or more MariaDB accounts. It removes privilege rows for the account from all grant tables. To use this statement, you must have the global CREATE USER privilege or the DELETE privilege for the `mysql` database. Each account is named using the same format as for the `CREATE USER` statement; for example, `'jeffrey'@'localhost'`. If you specify only the user name part of the account name, a host name part of `'%'` is used. For additional information about specifying account names, see CREATE USER.

CURRENT

If you specify an account that is currently connected, it is deleted, but the statement completes with a warning:

```
Dropped users 'user'@'host[,...]' have active connections. Use KILL CONNECTION if they should not be
used anymore.
```

This means that the user account with all its privileges is still active until the connection is terminated. As the warning implies, use a `KILL CONNECTION` statement to terminate a connection, or (more proactively) use the `FORCE` clause to forcibly close connections of the users named in the `DROP USER` statement. This ends connections, and immediately deletes the users.

In Oracle mode, if a user is connected, the `DROP USER` statement fails with an error:

```
Operation DROP USER failed for 'foo'@'localhost'.
```

Again, use the `FORCE` clause to prevent that error.

> **Warning:**
>
> The `FORCE` clause is not available in MariaDB Community Server. In Community Server, use a `KILL CONNECTION` statement to end the live sessions of a dropped account.

< 12.1

If you specify an account that is currently connected, it is not deleted until the connection is closed. The connection is not automatically closed.

However, a deleted user cannot initiate new connections any more.

If any of the specified user accounts do not exist, `ERROR 1396 (HY000)` results. If an error occurs, `DROP USER` still drops the accounts that do not result in an error. Only one error is produced for all users which have not been dropped:

```
ERROR 1396 (HY000): Operation DROP USER failed for 'u1'@'%','u2'@'%'
```

Failed `CREATE` or `DROP` operations, for both users and roles, produce the same error.

IF EXISTS

If the `IF EXISTS` clause is used, MariaDB returns a note instead of an error if the user does not exist.

The `CREATE USER` statement creates new MariaDB accounts. To use it, you must have the global CREATE USER privilege or the INSERT privilege for the mysql database.

If the `IF EXISTS` clause is used, MariaDB returns a note instead of an error if the user does not exist.

Examples

```
DROP USER bob;

DROP USER foo2@localhost,foo2@'127.%';
```

`IF EXISTS`:

```
DROP USER bob;
ERROR 1396 (HY000): Operation DROP USER failed for 'bob'@'%'

DROP USER IF EXISTS bob;
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------+
| Level | Code | Message                                 |
+-------+------+-----------------------------------------+
| Note  | 1974 | Can't drop user 'bob'@'%'; it doesn't exist |
+-------+------+-----------------------------------------+
```

See Also

- CREATE USER
- ALTER USER
- GRANT
- SHOW CREATE USER
- mysql.user table

This page is licensed: GPLv2, originally from fill_help_tables.sql

GRANT

Syntax

```
/* 1. Granting Privileges */
GRANT
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    TO account_or_role [, account_or_role] ...
    [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
    [WITH grant_option_list]

/* 2. Granting Proxy Access */
GRANT PROXY ON user_or_role
    TO account_or_role [, account_or_role] ...
    [WITH GRANT OPTION]

/* 3. Granting Roles */
GRANT role [, role] ...
    TO account_or_role [, account_or_role] ...
    [WITH ADMIN OPTION]

/* Variable Definitions */
account_or_role:
    username [authentication_option]
  | role
  | PUBLIC
  | CURRENT_USER [()]
  | CURRENT_ROLE [()]

authentication_option:
    IDENTIFIED BY 'password'
  | IDENTIFIED BY PASSWORD 'password_hash'
  | IDENTIFIED {VIA | WITH} authentication_rule [OR authentication_rule ...]

authentication_rule:
    authentication_plugin
  | authentication_plugin {USING | AS} 'authentication_string'
  | authentication_plugin {USING | AS} PASSWORD('password')

priv_type:
    ALL [PRIVILEGES]
  | ALTER | ALTER ROUTINE | BINLOG ADMIN | BINLOG MONITOR | BINLOG REPLAY
  | CONNECTION ADMIN | CREATE | CREATE ROUTINE | CREATE TABLESPACE
  | CREATE TEMPORARY TABLES | CREATE USER | CREATE VIEW
  | DELETE | DELETE HISTORY | DROP | EVENT | EXECUTE | FEDERATED ADMIN
  | FILE | GRANT OPTION | INDEX | INSERT | LOCK TABLES | PROCESS
  | READ ONLY ADMIN | RELOAD | REPLICATION CLIENT | REPLICATION MASTER ADMIN
  | REPLICATION SLAVE | REPLICATION SLAVE ADMIN | REFERENCES
  | SELECT | SET USER | SHOW CREATE ROUTINE | SHOW DATABASES | SHOW VIEW
  | SHUTDOWN | SLAVE MONITOR | SUPER | TRIGGER | UPDATE | USAGE

object_type:
    TABLE
  | FUNCTION
  | PROCEDURE
  | PACKAGE
  | PACKAGE BODY

priv_level:
    *
  | *.*
  | db_name.*
  | db_name.tbl_name
  | tbl_name
  | db_name.routine_name
```

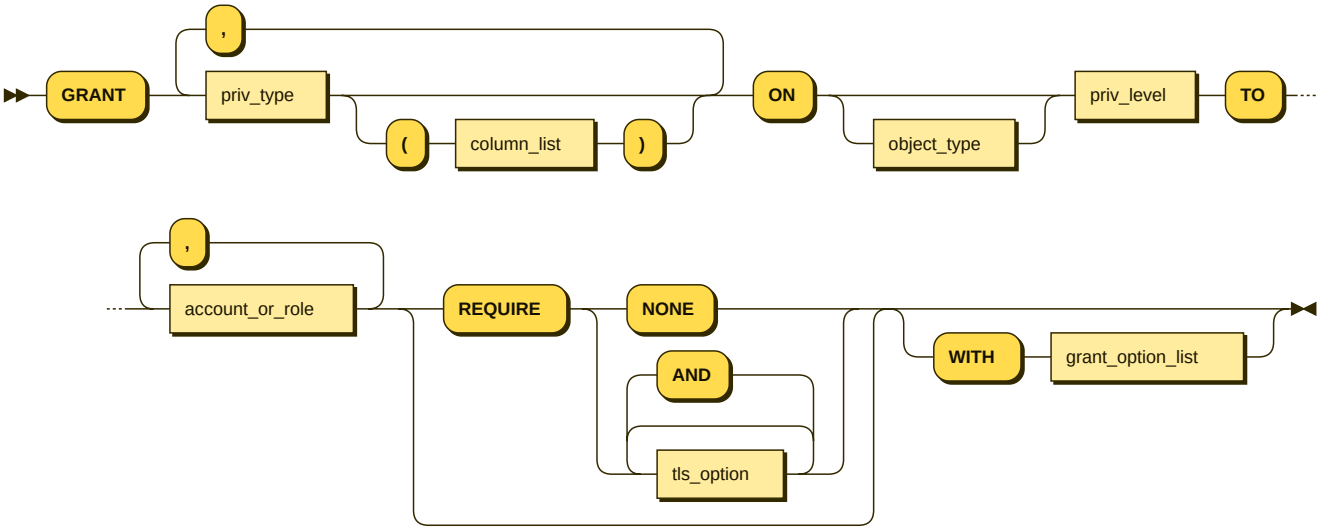
```
grant_option_list:
    grant_option [grant_option] ...

grant_option:
    GRANT OPTION
  | resource_option

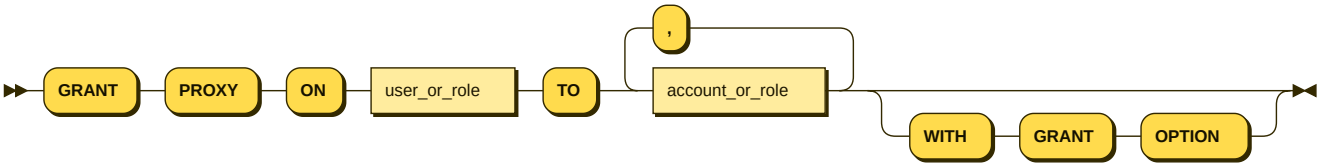
resource_option:
    MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count
  | MAX_STATEMENT_TIME time

tls_option:
    SSL
  | X509
  | CIPHER 'cipher'
  | ISSUER 'issuer'
  | SUBJECT 'subject'
```

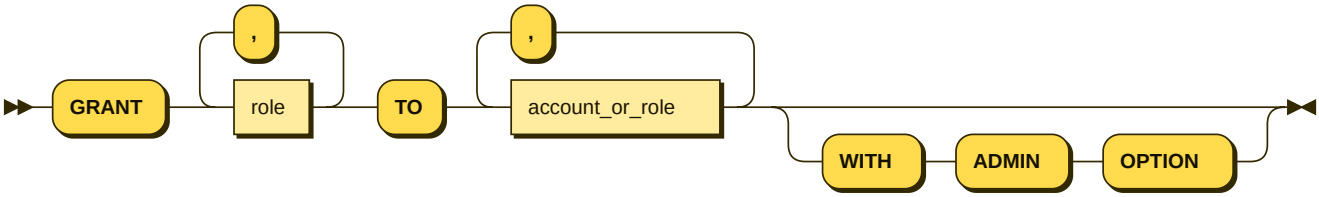
GRANT has three top-level forms. Sub-rule diagrams are not included here — the `priv_type` production alone is a ~60-row alternative list; readers can consult the BNF above for the sub-rule expansions.



Railroad diagram of GRANT (privileges form)



Railroad diagram of GRANT PROXY



Railroad diagram of GRANT (roles form)

Description

The `GRANT` statement allows you to grant privileges or roles to accounts. To use `GRANT`, you must have the `GRANT OPTION` privilege, and you must have the privileges that you are granting.

Use the REVOKE statement to revoke privileges granted with the `GRANT` statement.

Use the SHOW GRANTS statement to determine what privileges an account has.

> **Note:**
>
> From MariaDB 13.1, a privilege blocked with DENY cannot be granted back. A deny takes precedence over every `GRANT` at every privilege level, whatever order the statements are issued in, and is only lifted by `REVOKE DENY`.

Account Names

For `GRANT` statements, account names are specified as the `username` argument in the same way as they are for CREATE USER statements. See account names from the `CREATE USER` page for details on how account names are specified.

Database Name Wildcard Matching Order

When multiple `GRANT` entries match a database name (since database names in `GRANT` can contain `%` and `_` wildcards), MariaDB applies the most specific matching grant. Specificity is determined by how many database names a pattern can match, a pattern matching fewer databases is more specific and takes precedence.

From MariaDB 10.4.6 (MDEV-14735), this ordering is managed correctly. In previous versions (before 10.4.6), the sort order among wildcard database patterns was determined only by the position of the first wildcard, which could generate insertion-order-dependent results.

Example

```
CREATE USER 'jtest'@localhost IDENTIFIED BY 'jtest';
GRANT SELECT ON `%test`.* TO 'jtest'@localhost;
GRANT SELECT, INSERT, DELETE ON `j-%`.* TO 'jtest'@localhost;
```

Starting with MariaDB 10.4.6, `` `%test`.* `` is considered more specific than `` `j-%`.* `` because it matches fewer database names. Since only the first matching grant is applied, the `SELECT`-only grant on `` `%test`.* `` takes precedence, and `INSERT` is not granted for the database `j-test`.

> This behavior changed in MariaDB 10.4.6. In earlier versions, `j-%` was incorrectly treated as more specific because sorting considered only the prefix before the first wildcard (`j-` is longer than the empty prefix of `%test`). If you are upgrading from MariaDB 10.3 or earlier, review any `GRANT` statements that use overlapping wildcard database patterns, as privilege resolution may differ.

Implicit Account Creation

The `GRANT` statement also allows you to implicitly create accounts in some cases.

If the account does not yet exist, then `GRANT` can implicitly create it. To implicitly create an account with `GRANT`, a user is required to have the same privileges that would be required to explicitly create the account with the `CREATE USER` statement.

If the `NO_AUTO_CREATE_USER` SQL_MODE is set, then accounts can only be created if authentication information is specified, or with a CREATE USER statement. If no authentication information is provided, `GRANT` will produce an error when the specified account does not exist, for example:

```
SHOW VARIABLES LIKE '%sql_mode%' ;
```

```
+---------------+-------------------------------------------+
| Variable_name | Value                                     |
+---------------+-------------------------------------------+
| sql_mode      | NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+---------------+-------------------------------------------+
```

```
GRANT USAGE ON *.* TO 'user123'@'%' IDENTIFIED BY '';
```

```
ERROR 1133 (28000): Can't find any matching row in the user table
```

```
GRANT USAGE ON *.* TO 'user123'@'%'
  IDENTIFIED VIA PAM using 'mariadb' require ssl ;
Query OK, 0 rows affected (0.00 sec)
```

```
SELECT host, user FROM mysql.user WHERE user='user123' ;
```

```
+------+----------+
| host | user     |
+------+----------+
| %    | user123 |
+------+----------+
```

Privilege Levels

Privileges can be set globally, for an entire database, for a table or routine, or for individual columns in a table. Certain privileges can only be set at certain levels.

Global privileges do not take effect immediately and are only applied to connections created after the `GRANT` statement was executed.

- Global privileges priv_type are granted using `*.*` for priv_level. Global privileges include privileges to administer the database and manage user accounts, as well as privileges for all tables, functions, and procedures. Global privileges are stored in mysql.global_priv table.
- Database privileges priv_type are granted using `db_name.*` for priv_level, or using just `*` to use the current database. Database privileges include privileges to create tables and functions, as well as privileges for all tables, functions, and procedures in the database. Database privileges are stored in the mysql.db table.
- Table privileges priv_type are granted using `db_name.tbl_name` for priv_level, or using just `tbl_name` to specify a table in the current database. The `TABLE` keyword is optional. Table privileges include the ability to select and change data in the table. Certain table privileges can be granted for individual columns.
- Column privileges priv_type are granted by specifying a table for priv_level and providing a column list after the privilege type. They allow you to control exactly which columns in a table users can select and change.
- Function privileges priv_type are granted using `FUNCTION db_name.routine_name` for priv_level, or using just `FUNCTION routine_name` to specify a function in the current database.
- Procedure privileges priv_type are granted using `PROCEDURE db_name.routine_name` for priv_level, or using just `PROCEDURE routine_name` to specify a procedure in the current database.

The `USAGE` Privilege

The `USAGE` privilege grants no real privileges. The SHOW GRANTS statement will show a global `USAGE` privilege for a newly-created user. You can use `USAGE` with the `GRANT` statement to change options like `GRANT OPTION` and `MAX_USER_CONNECTIONS` without changing any account privileges.

The `ALL PRIVILEGES` Privilege

The `ALL PRIVILEGES` privilege grants all available privileges. Granting all privileges only affects the given privilege level. For example, granting all privileges on a table does not grant any privileges on the database or globally.

Using `ALL PRIVILEGES` does not grant the special `GRANT OPTION` privilege.

You can use `ALL` instead of `ALL PRIVILEGES`.

The `GRANT OPTION` Privilege

Use the `WITH GRANT OPTION` clause to give users the ability to grant privileges to other users at the given privilege level. Users with the `GRANT OPTION` privilege can only grant privileges they have. They cannot grant privileges at a higher privilege level than they have the `GRANT OPTION` privilege.

The `GRANT OPTION` privilege cannot be set for individual columns. If you use `WITH GRANT OPTION` when specifying column privileges, the `GRANT OPTION` privilege will be granted for the entire table.

Using the `WITH GRANT OPTION` clause is equivalent to listing `GRANT OPTION` as a privilege.

Global Privileges

The following table lists the privileges that can be granted globally. You can also grant all database, table, and function privileges globally. When granted globally, these privileges apply to all databases, tables, or functions, including those created later.

To set a global privilege, use `*.*` for *priv_level*.

BINLOG ADMIN

> **CURRENT**
>
> Enables administration of the binary log, including the PURGE BINARY LOGS statement and setting the system variables:
>
> - binlog_annotate_row_events
> - binlog_cache_size
> - binlog_commit_wait_count
> - binlog_commit_wait_usec
> - binlog_direct_non_transactional_updates
> - binlog_expire_logs_seconds
> - binlog_file_cache_size
> - binlog_format
> - binlog_row_image
> - binlog_row_metadata
> - binlog_stmt_cache_size
> - expire_logs_days
> - log_bin_compress
> - log_bin_compress_min_len
> - log_bin_trust_function_creators
> - max_binlog_cache_size
> - max_binlog_size
> - max_binlog_stmt_cache_size
> - sql_log_bin and

- sync_binlog.

< 10.5

BINLOG ADMIN isn't available.

BINLOG MONITOR

CURRENT

New name for REPLICATION CLIENT. `REPLICATION CLIENT` can still be used, though.

< 10.5

Use REPLICATION CLIENT instead. SHOW REPLICA STATUS isn't included in this privilege, and REPLICA MONITOR is required.

Permits running SHOW commands related to the binary log, in particular the SHOW BINLOG STATUS and SHOW BINARY LOGS statements.

BINLOG REPLAY

CURRENT

Enables replaying the binary log with the BINLOG statement (generated by mariadb-binlog), executing SET timestamp when secure_timestamp is set to `replication`, and setting the session values of system variables usually included in BINLOG output, in particular:

- gtid_domain_id
- gtid_seq_no
- pseudo_thread_id
- server_id.

< 10.5

`BINLOG REPLAY` isn't available.

CONNECTION ADMIN

Enables administering connection resource limit options. This includes ignoring the limits specified by max_user_connections and max_password_errors, and allowing one extra connection over max_connections

The statements specified in init_connect are not executed, killing connections and queries owned by other users is permitted. The following connection-related system variables can be changed:

- connect_timeout
- disconnect_on_expired_password
- extra_max_connections
- init_connect
- max_connections
- max_connect_errors
- max_password_errors
- proxy_protocol_networks
- secure_auth
- slow_launch_time

- thread_pool_exact_stats
- thread_pool_dedicated_listener
- thread_pool_idle_timeout
- thread_pool_max_threads
- thread_pool_min_threads
- thread_pool_oversubscribe
- thread_pool_prio_kickup_timer
- thread_pool_priority
- thread_pool_size, and
- thread_pool_stall_limit.

CREATE USER

Create a user using the CREATE USER statement, or implicitly create a user with the `GRANT` statement.

FEDERATED ADMIN

> **CURRENT**
>
> Execute CREATE SERVER, ALTER SERVER, and DROP SERVER statements.
>
> **< 10.5**
>
> `FEDERATED ADMIN` is not available.

FILE

Read and write files on the server, using statements like LOAD DATA INFILE or functions like LOAD_FILE(). Also needed to create CONNECT outward tables. MariaDB server must have the permissions to access those files.

GRANT OPTION

Grant global privileges. You can only grant privileges that you have.

PROCESS

Show information about the active processes, for example via SHOW PROCESSLIST or mariadb-admin processlist. If you have the PROCESS privilege, you can see all threads. Otherwise, you can see only your own threads (that is, threads associated with the MariaDB account that you are using).

READ_ONLY ADMIN

> **CURRENT**
>
> User ignores the read_only system variable, and can perform write operations even when the `read_only` option is active.
>
> A user with that privilege can also change the (global) value of `read_only`.
>
> The `READ_ONLY ADMIN` privilege has been removed from SUPER. The benefit of this is that one can remove the `READ_ONLY ADMIN` privilege from all users and ensure that no one can make any changes on any non-temporary tables. This is useful on replicas when one wants to ensure that the replica is kept identical to the primary.
>
> **< 10.11**

User ignores the read_only system variable, and can perform write operations even when the `read_only` option is active.

A user with that privilege can also change the (global) value of `read_only`.

The `READ_ONLY ADMIN` privilege is included in SUPER.

< 10.5

`READ\_ONLY ADMIN` isn't available.

RELOAD

Execute FLUSH statements or equivalent mariadb-admin commands.

REPLICATION CLIENT

CURRENT

Execute SHOW MASTER STATUS and SHOW BINARY LOGS informative statements. Renamed to BINLOG MONITOR (but still supported as an alias for compatibility reasons).

< 10.5

Execute SHOW MASTER STATUS and SHOW BINARY LOGS informative statements. SHOW SLAVE STATUS is part of REPLICATION CLIENT.

CURRENT

Execute SHOW MASTER STATUS and SHOW BINARY LOGS informative statements. Using BINLOG MONITOR instead is still supported as an alias.

< 10.6

Execute SHOW MASTER STATUS and SHOW BINARY LOGS informative statements. Renamed to BINLOG MONITOR in MariaDB 10.5.2 (but still supported as an alias for compatibility reasons). SHOW SLAVE STATUS was part of REPLICATION CLIENT prior to MariaDB 10.5.

REPLICATION MASTER ADMIN

CURRENT

Permits administration of primary servers, including the SHOW REPLICA HOSTS statement, and setting the gtid_binlog_state, gtid_domain_id, master_verify_checksum and server_id system variables.

< 10.5

`REPLICATION MASTER ADMIN` is not available.

REPLICA MONITOR

CURRENT

Permit SHOW REPLICA STATUS and SHOW RELAYLOG EVENTS.

See *Reasoning* tab as to why this was implemented.

REASONING

When a user would upgrade from an older major release to a MariaDB 10.5 minor release prior to MariaDB 10.5.9, certain user accounts would lose capabilities. For example, a user account that had the REPLICATION CLIENT privilege in older major releases could run SHOW REPLICA STATUS, but after upgrading to a MariaDB 10.5 minor release prior to MariaDB 10.5.9, they could no longer run SHOW REPLICA STATUS, because that statement was changed to require the REPLICATION REPLICA ADMIN privilege.

This issue is fixed in MariaDB 10.5.9 with this new privilege, which now grants the user the ability to execute `SHOW [ALL] (SLAVE | REPLICA) STATUS`.

When a database is upgraded from an older major release to MariaDB Server 10.5.9 or later, any user accounts with the `REPLICATION CLIENT` or `REPLICATION SLAVE` privileges will automatically be granted the new `REPLICA MONITOR` privilege. The privilege fix occurs when the server is started up, not when mariadb-upgrade is performed.

However, when a database is upgraded from an early 10.5 minor release to 10.5.9 and later, the user will have to fix any user account privileges manually.

< 10.5.9

`REPLICA MONITOR` is not available.

REPLICATION REPLICA

CURRENT

Synonym for REPLICATION SLAVE.

< 10.5

`REPLICATION REPLICA` is not available.

REPLICATION SLAVE

CURRENT

Accounts used by replica servers on the primary need this privilege. This is needed to get the updates made on the master. REPLICATION REPLICA is an alias for `REPLICATION SLAVE`.

< 10.5

Accounts used by replica servers on the primary need this privilege. This is needed to get the updates made on the master.

REPLICATION SLAVE ADMIN

CURRENT

Permits administering replica servers, including START REPLICA/SLAVE, STOP REPLICA/SLAVE, CHANGE MASTER, SHOW REPLICA/SLAVE STATUS, SHOW RELAYLOG EVENTS statements, replaying the binary log with the BINLOG statement (generated by mariadb-binlog), and setting the system variables:

- gtid_cleanup_batch_size
- gtid_ignore_duplicates
- gtid_pos_auto_engines
- gtid_slave_pos
- gtid_strict_mode
- init_slave

- read_binlog_speed_limit
- relay_log_purge
- relay_log_recovery
- replicate_do_db
- replicate_do_table
- replicate_events_marked_for_skip
- replicate_ignore_db
- replicate_ignore_table
- replicate_wild_do_table
- replicate_wild_ignore_table
- slave_compressed_protocol
- slave_ddl_exec_mode
- slave_domain_parallel_threads
- slave_exec_mode
- slave_max_allowed_packet
- slave_net_timeout
- slave_parallel_max_queued
- slave_parallel_mode
- slave_parallel_threads
- slave_parallel_workers
- slave_run_triggers_for_rbr
- slave_sql_verify_checksum
- slave_transaction_retry_interval
- slave_type_conversions
- sync_master_info
- sync_relay_log, and
- sync_relay_log_info.

< 10.5

`REPLICATION SLAVE ADMIN` is not available.

SET USER

CURRENT

Enables setting the `DEFINER` when creating triggers, views, stored functions and stored procedures.

< 10.5

SET USER isn't available.

SHOW DATABASES

List all databases using the SHOW DATABASES statement. Without the `SHOW DATABASES` privilege, you can still issue the `SHOW DATABASES` statement, but it will only list databases containing tables on which you have privileges.

SHUTDOWN

Shut down the server using SHUTDOWN or the mariadb-admin shutdown command.

SUPER

Execute superuser statements: CHANGE MASTER TO, KILL (users who do not have this privilege can only `KILL` their own threads), PURGE LOGS, SET global system variables, or the mariadb-admin debug command. Also, this permission allows the user to write data even if the read_only startup option is set, enable or disable logging, enable or disable replication on replica, specify a `DEFINER` for statements that support that clause, connect once reaching the `MAX_CONNECTIONS`. If a statement has been specified for the init-connect mariadbd option, that command will not be executed when a user with `SUPER` privileges connects to the server.

CURRENT

The SUPER privilege has been split into multiple smaller privileges to allow for more fine-grained privileges (MDEV-21743). The privileges are:

- SET USER
- FEDERATED ADMIN
- CONNECTION ADMIN
- REPLICATION SLAVE ADMIN
- BINLOG ADMIN
- BINLOG REPLAY
- REPLICA MONITOR
- BINLOG MONITOR
- REPLICATION MASTER ADMIN
- READ_ONLY ADMIN

These grants are no longer a part of SUPER and need to be granted separately.

The READ_ONLY ADMIN privilege has been removed from `SUPER`. The benefit of this is that one can remove the READ_ONLY ADMIN privilege from all users and ensure that no one can make any changes on any non-temporary tables. This is useful on replicas when one wants to ensure that the replica is kept identical to the primary (MDEV-29596).

< 11.0.1

The SUPER privilege has been split into multiple smaller privileges to allow for more fine-grained privileges (MDEV-21743). The privileges are:

- SET USER
- FEDERATED ADMIN
- CONNECTION ADMIN
- REPLICATION SLAVE ADMIN
- BINLOG ADMIN
- BINLOG REPLAY
- REPLICA MONITOR
- BINLOG MONITOR
- REPLICATION MASTER ADMIN
- READ_ONLY ADMIN

These grants are part of SUPER and don't need to be granted separately.

< 10.5

Use the SUPER privilege.

Database Privileges

The following table lists the privileges that can be granted at the database level. You can also grant all table and function privileges at the database level. Table and function privileges on a database apply to all tables or functions in that database, including those created later.

To set a privilege for a database, specify the database using `db_name.*` for *priv_level*, or just use `*` to specify the current. database.

Privilege	Description
CREATE	Create a database using the CREATE DATABASE statement, when the privilege is granted for a database. You can grant the CREATE privilege on databases that do not yet exist. This also grants the CREATE privilege on all tables in the database.
CREATE ROUTINE	Create Stored Programs using the CREATE PROCEDURE and CREATE FUNCTION statements.
CREATE TEMPORARY TABLES	Create temporary tables with the CREATE TEMPORARY TABLE statement. This privilege enable writing and dropping those temporary tables
DROP	Drop a database using the DROP DATABASE statement, when the privilege is granted for a database. This also grants the DROP privilege on all tables in the database.
EVENT	Create, drop and alter EVENTs.
GRANT OPTION	Grant database privileges. You can only grant privileges that you have.
LOCK TABLES	Acquire explicit locks using the LOCK TABLES statement; you also need to have the SELECT privilege on a table, in order to lock it.
SHOW CREATE ROUTINE	Permit viewing the SHOW CREATE definition statement of a routine, for example SHOW CREATE FUNCTION, even if not the routine owner. From MariaDB 11.3.0.

Table Privileges

Privilege	Description
ALTER	Change the structure of an existing table using the ALTER TABLE statement.
CREATE	Create a table using the CREATE TABLE statement. You can grant the CREATE privilege on tables that do not yet exist.
CREATE VIEW	Create a view using the CREATE_VIEW statement.
DELETE	Remove rows from a table using the DELETE statement.
DELETE HISTORY	Remove historical rows from a table using the DELETE HISTORY statement. Displays as DELETE VERSIONING ROWS when running SHOW PRIVILEGES until MariaDB 10.5.2 (MDEV-20382). If a user has the SUPER privilege but not this privilege, running mariadb-upgrade will grant this privilege as well.
DROP	Drop a table using the DROP TABLE statement or a view using the DROP VIEW statement. Also required to execute the TRUNCATE TABLE statement.
GRANT OPTION	Grant table privileges. You can only grant privileges that you have.
INDEX	Create an index on a table using the CREATE INDEX statement. Without the INDEX privilege, you can still create indexes when creating a table using the CREATE TABLE statement if the you have the CREATE privilege, and you can create indexes using the ALTER TABLE statement if you have the ALTER privilege.
INSERT	Add rows to a table using the INSERT statement. The INSERT privilege can also be set on individual columns; see Column Privileges below for details.
REFERENCES	Unused.
SELECT	Read data from a table using the SELECT statement. The SELECT privilege can also be set on individual columns; see Column Privileges below for details.

Privilege	Description
SHOW VIEW	Show the CREATE VIEW statement to create a view using the SHOW CREATE VIEW statement.
TRIGGER	Required to run the CREATE TRIGGER, DROP TRIGGER, and SHOW CREATE TRIGGER statements. When another user activates a trigger (running INSERT, UPDATE, or DELETE statements on the associated table), for the trigger to execute, the user that defined the trigger should have the TRIGGER privilege for the table. The user running the INSERT, UPDATE, or DELETE statements on the table is not required to have the TRIGGER privilege.
UPDATE	Update existing rows in a table using the UPDATE statement. UPDATE statements usually include a WHERE clause to update only certain rows. You must have SELECT privileges on the table or the appropriate columns for the WHERE clause. The UPDATE privilege can also be set on individual columns; see Column Privileges below for details.

Column Privileges

Some table privileges can be set for individual columns of a table. To use column privileges, specify the table explicitly and provide a list of column names after the privilege type. For example, the following statement would allow the user to read the names and positions of employees, but not other information from the same table, such as salaries.

```
GRANT SELECT (name, position) ON Employee TO 'jeffrey'@'localhost';
```

Privilege	Description
INSERT (column_list)	Add rows specifying values in columns using the INSERT statement. If you only have column-level `INSERT` privileges, you must specify the columns you are setting in the `INSERT` statement. All other columns will be set to their default values, or `NULL`.
REFERENCES (column_list)	Unused.
SELECT (column_list)	Read values in columns using the SELECT statement. You cannot access or query any columns for which you do not have `SELECT` privileges, including in `WHERE`, `ON`, `GROUP BY`, and `ORDER BY` clauses.
UPDATE (column_list)	Update values in columns of existing rows using the UPDATE statement. `UPDATE` statements usually include a `WHERE` clause to update only certain rows. You must have `SELECT` privileges on the table or the appropriate columns for the `WHERE` clause.

Function Privileges

Privilege	Description
ALTER ROUTINE	Change the characteristics of a stored function using the ALTER FUNCTION statement.
EXECUTE	Use a stored function. You need `SELECT` privileges for any tables or columns accessed by the function.
GRANT OPTION	Grant function privileges. You can only grant privileges that you have.

Procedure Privileges

Privilege	Description
ALTER ROUTINE	Change the characteristics of a stored procedure using the ALTER PROCEDURE statement.
EXECUTE	Execute a stored procedure using the CALL statement. The privilege to call a procedure may allow you to perform actions you wouldn't otherwise be able to do, such as insert rows into a table.
GRANT OPTION	Grant procedure privileges. You can only grant privileges that you have.

```
GRANT EXECUTE ON PROCEDURE mysql.create_db TO maintainer;
```

Package Privileges

Privilege	Description
ALTER ROUTINE	Change the characteristics of a stored package.
EXECUTE	Execute a stored package or package body.
GRANT OPTION	Grant package privileges. You can only grant privileges that you have.

Proxy Privileges

Privilege	Description
PROXY	Permits one user to be a proxy for another.

The `PROXY` privilege allows one user to proxy as another user, which means their privileges change to that of the proxy user, and the CURRENT_USER() function returns the user name of the proxy user.

The `PROXY` privilege only works with authentication plugins that support it. The default mysql_native_password authentication plugin does not support proxy users.

The pam authentication plugin is the only plugin included with MariaDB that currently supports proxy users. The `PROXY` privilege is commonly used with the pam authentication plugin to enable user and group mapping with PAM.

For example, to grant the `PROXY` privilege to an anonymous account that authenticates with the pam authentication plugin, you could execute the following:

```
CREATE USER 'dba'@'%' IDENTIFIED BY 'strongpassword';
GRANT ALL PRIVILEGES ON *.* TO 'dba'@'%' ;

CREATE USER ''@'%' IDENTIFIED VIA pam USING 'mariadb';
GRANT PROXY ON 'dba'@'%' TO ''@'%';
```

A user account can only grant the `PROXY` privilege for a specific user account if the granter also has the `PROXY` privilege for that specific user account, and if that privilege is defined `WITH GRANT OPTION`. For example, the following example fails because the granter does not have the `PROXY` privilege for that specific user account at all:

```
SELECT USER(), CURRENT_USER();
```

```
+-----------------+-----------------+
| USER()          | CURRENT_USER()  |
+-----------------+-----------------+
| alice@localhost | alice@localhost |
+-----------------+-----------------+
```

```
SHOW GRANTS
```

```
+-----------------------------------------------------------------------------------------------------------------
---------------+
| Grants for alice@localhost
|
+-----------------------------------------------------------------------------------------------------------------
---------------+
| GRANT ALL PRIVILEGES ON *.* TO 'alice'@'localhost' IDENTIFIED BY PASSWORD
'*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' |
+-----------------------------------------------------------------------------------------------------------------
---------------+
```

```
GRANT PROXY ON 'dba'@'localhost' TO 'bob'@'localhost';
ERROR 1698 (28000): Access denied for user 'alice'@'localhost'
```

And the following example fails because the granter does have the `PROXY` privilege for that specific user account, but it is not defined `WITH GRANT OPTION`:

```
SELECT USER(), CURRENT_USER();
```

```
+-----------------+-----------------+
| USER()          | CURRENT_USER()  |
+-----------------+-----------------+
| alice@localhost | alice@localhost |
+-----------------+-----------------+
```

```
SHOW GRANTS;
```

```
+-----------------------------------------------------------------------------------------------------------------
--------------+
| Grants for alice@localhost
|
+-----------------------------------------------------------------------------------------------------------------
--------------+
| GRANT ALL PRIVILEGES ON *.* TO 'alice'@'localhost' IDENTIFIED BY PASSWORD
'*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' |
| GRANT PROXY ON 'dba'@'localhost' TO 'alice'@'localhost'
|
+-----------------------------------------------------------------------------------------------------------------
--------------+
```

```
GRANT PROXY ON 'dba'@'localhost' TO 'bob'@'localhost';
ERROR 1698 (28000): Access denied for user 'alice'@'localhost'
```

But the following example succeeds because the granter does have the `PROXY` privilege for that specific user account, and it is defined `WITH GRANT OPTION`:

```
SELECT USER(), CURRENT_USER();
```

```
+-----------------+-----------------+
| USER()          | CURRENT_USER()  |
+-----------------+-----------------+
| alice@localhost | alice@localhost |
+-----------------+-----------------+
```

```
SHOW GRANTS;
```

```
+-----------------------------------------------------------------------------------------------------------------
--------------------------------+
| Grants for alice@localhost
|
+-----------------------------------------------------------------------------------------------------------------
--------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'alice'@'localhost' IDENTIFIED BY PASSWORD
'*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' WITH GRANT OPTION |
| GRANT PROXY ON 'dba'@'localhost' TO 'alice'@'localhost' WITH GRANT OPTION
|
+-----------------------------------------------------------------------------------------------------------------
--------------------------------+
```

```
GRANT PROXY ON 'dba'@'localhost' TO 'bob'@'localhost';
```

A user account can grant the `PROXY` privilege for any other user account if the granter has the `PROXY` privilege for the `''@'%'` anonymous user account, like this:

```
GRANT PROXY ON ''@'%' TO 'dba'@'localhost' WITH GRANT OPTION;
```

For example, the following example succeeds because the user can grant the `PROXY` privilege for any other user account:

```
SELECT USER(), CURRENT_USER();
```

```
+-----------------+-----------------+
| USER()          | CURRENT_USER()  |
+-----------------+-----------------+
| alice@localhost | alice@localhost |
+-----------------+-----------------+
```

```
SHOW GRANTS;
```

```
+-----------------------------------------------------------------------------------------------------------------
--------------------------------+
| Grants for alice@localhost
|
+-----------------------------------------------------------------------------------------------------------------
--------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'alice'@'localhost' IDENTIFIED BY PASSWORD
'*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' WITH GRANT OPTION |
| GRANT PROXY ON ''@'%' TO 'alice'@'localhost' WITH GRANT OPTION
|
+-----------------------------------------------------------------------------------------------------------------
--------------------------------+
```

```
GRANT PROXY ON 'app1_dba'@'localhost' TO 'bob'@'localhost';
Query OK, 0 rows affected (0.004 sec)
```

```
GRANT PROXY ON 'app2_dba'@'localhost' TO 'carol'@'localhost';
Query OK, 0 rows affected (0.004 sec)
```

The default `root` user accounts created by mariadb-install-db have this privilege. For example:

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION;
GRANT PROXY ON ''@'%' TO 'root'@'localhost' WITH GRANT OPTION;
```

This allows the default `root` user accounts to grant the `PROXY` privilege for any other user account, and it also allows the default `root` user accounts to grant others the privilege to do the same.

Authentication Options

The authentication options for the `GRANT` statement are the same as those for the CREATE USER statement.

IDENTIFIED BY 'password'

The optional `IDENTIFIED BY` clause can be used to provide an account with a password. The password should be specified in plain text. It will be hashed by the PASSWORD function prior to being stored.

For example, if our password is `mariadb`, then we can create the user with:

```
GRANT USAGE ON *.* TO foo2@test IDENTIFIED BY 'mariadb';
```

If you do not specify a password with the `IDENTIFIED BY` clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

If the user account already exists and if you provide the `IDENTIFIED BY` clause, then the user's password will be changed. You must have the privileges needed for the SET PASSWORD statement to change a user's password with `GRANT`.

The only authentication plugins that this clause supports are mysql_native_password and mysql_old_password.

IDENTIFIED BY PASSWORD 'password_hash'

The optional `IDENTIFIED BY PASSWORD` clause can be used to provide an account with a password that has already been hashed. The password should be specified as a hash that was provided by the PASSWORD function. It will be stored as-is.

For example, if our password is `mariadb`, then we can find the hash with:

```
SELECT PASSWORD('mariadb');
```

```
+-------------------------------------------+
| PASSWORD('mariadb')                       |
+-------------------------------------------+
| *54958E764CE10E50764C2EECBB71D01F08549980 |
+-------------------------------------------+
1 row in set (0.00 sec)
```

And then we can create a user with the hash:

```
GRANT USAGE ON *.* TO foo2@test IDENTIFIED BY
  PASSWORD '*54958E764CE10E50764C2EECBB71D01F08549980';
```

If you do not specify a password with the `IDENTIFIED BY` clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

If the user account already exists and if you provide the `IDENTIFIED BY` clause, then the user's password will be changed. You must have the privileges needed for the SET PASSWORD tastatement to change a user's password with `GRANT`.

The only authentication plugins that this clause supports are mysql_native_password and mysql_old_password.

IDENTIFIED {VIA|WITH} authentication_plugin

The optional `IDENTIFIED VIA authentication_plugin` allows you to specify that the account should be authenticated by a specific authentication plugin. The plugin name must be an active authentication plugin as per SHOW PLUGINS. If it doesn't show up in that output, then you will need to install it with INSTALL PLUGIN or INSTALL SONAME.

For example, this could be used with the PAM authentication plugin:

```
GRANT USAGE ON *.* TO foo2@test IDENTIFIED VIA pam;
```

Some authentication plugins allow additional arguments to be specified after a `USING` or `AS` keyword. For example, the PAM authentication plugin accepts a service name:

```
GRANT USAGE ON *.* TO foo2@test IDENTIFIED VIA pam USING 'mariadb';
```

The exact meaning of the additional argument would depend on the specific authentication plugin.

The `USING` or `AS` keyword can also be used to provide a plain-text password to a plugin if it's provided as an argument to the PASSWORD() function. This is only valid for authentication plugins that have implemented a hook for the PASSWORD() function. For example, the ed25519 authentication plugin supports this:

```
CREATE USER safe@'%' IDENTIFIED VIA ed25519
  USING PASSWORD('secret');
```

One can specify many authentication plugins, they all work as alternative ways of authenticating a user:

```
CREATE USER safe@'%' IDENTIFIED VIA ed25519
  USING PASSWORD('secret') OR unix_socket;
```

By default, when you create a user without specifying an authentication plugin, MariaDB uses the mysql_native_password plugin.

Resource Limit Options

It is possible to set per-account limits for certain server resources. The following table shows the values that can be set per account:

Limit Type	Description
MAX_QUERIES_PER_HOUR	Number of statements that the account can issue per hour (including updates)
MAX_UPDATES_PER_HOUR	Number of updates (not queries) that the account can issue per hour
MAX_CONNECTIONS_PER_HOUR	Number of connections that the account can start per hour
MAX_USER_CONNECTIONS	Number of simultaneous connections that can be accepted from the same account; if it is 0, max_connections will be used instead; if max_connections is 0, there is no limit for this account's simultaneous connections.
MAX_STATEMENT_TIME	Timeout, in seconds, for statements executed by the user. See also Aborting Statements that Exceed a Certain Time to Execute.

If any of these limits are set to `0`, then there is no limit for that resource for that user.

To set resource limits for an account, if you do not want to change that account's privileges, you can issue a `GRANT` statement with the `USAGE` privilege, which has no meaning. The statement can name some or all limit types, in any order.

Here is an example showing how to set resource limits:

```
GRANT USAGE ON *.* TO 'someone'@'localhost' WITH
    MAX_USER_CONNECTIONS 0
    MAX_QUERIES_PER_HOUR 200;
```

The resources are tracked per account, which means `'user'@'server'`; not per user name or per connection.

The count can be reset for all users using FLUSH USER_RESOURCES, FLUSH PRIVILEGES or mariadb-admin reload.

> **CURRENT**
>
> Users with the `CONNECTION ADMIN` privilege or the `SUPER` privilege are not restricted by `max_user_connections` or `max_password_errors`, and they are allowed one additional connection when `max_connections` is reached.

< 10.5

Users with the `CONNECTION ADMIN` privilege or the `SUPER` privilege are restricted by `max_user_connections` or `max_password_errors`, and they are not allowed one additional connection when `max_connections` is reached.

Per account resource limits are stored in the user table, in the mysql database. Columns used for resources limits are named `max_questions`, `max_updates`, `max_connections` (for `MAX_CONNECTIONS_PER_HOUR`), and `max_user_connections` (for `MAX_USER_CONNECTIONS`).

TLS Options

By default, MariaDB transmits data between the server and clients without encrypting it. This is generally acceptable when the server and client run on the same host or in networks where security is guaranteed through other means. However, in cases where the server and client exist on separate networks or they are in a high-risk network, the lack of encryption does introduce security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

To mitigate this concern, MariaDB allows you to encrypt data in transit between the server and clients using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix ssl_, but internally, MariaDB only supports its secure successors.

See Secure Connections Overview for more information about how to determine whether your MariaDB server has TLS support.

You can set certain TLS-related restrictions for specific user accounts. For instance, you might use this with user accounts that require access to sensitive data while sending it across networks that you do not control. These restrictions can be enabled for a user account with the CREATE USER, ALTER USER, or GRANT statements. The following options are available:

Option	Description
REQUIRE NONE	TLS is not required for this account, but can still be used.
REQUIRE SSL	The account must use TLS, but no valid X509 certificate is required. This option cannot be combined with other TLS options.
REQUIRE X509	The account must use TLS and must have a valid X509 certificate. This option implies REQUIRE SSL. This option cannot be combined with other TLS options.
REQUIRE ISSUER 'issuer'	The account must use TLS and must have a valid X509 certificate. Also, the Certificate Authority must be the one specified via the string issuer. This option implies REQUIRE X509. This option can be combined with the SUBJECT, and CIPHER options in any order.
REQUIRE SUBJECT 'subject'	The account must use TLS and must have a valid X509 certificate. Also, the certificate's Subject must be the one specified via the string subject. This option implies REQUIRE X509. This option can be combined with the ISSUER, and CIPHER options in any order.
REQUIRE CIPHER 'cipher'	The account must use TLS, but no valid X509 certificate is required. Also, the encryption used for the connection must use a specific cipher method specified in the string cipher. This option implies REQUIRE SSL. This option can be combined with the ISSUER, and SUBJECT options in any order.

The `REQUIRE` keyword must be used only once for all specified options, and the `AND` keyword can be used to separate individual options, but it is not required.

For example, you can create a user account that requires these TLS options with the following:

```
GRANT USAGE ON *.* TO 'alice'@'%'
  REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=Portland'
  AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter Parker/emailAddress=p.parker@marvel.com'
  AND CIPHER 'SHA-DES-CBC3-EDH-RSA';
```

If any of these options are set for a specific user account, then any client who tries to connect with that user account will have to be configured to connect with TLS.

See Securing Connections for Client and Server for information on how to enable TLS on the client and server.

Roles

```
GRANT role TO grantee [, grantee ... ]
[ WITH ADMIN OPTION ]

grantee:
    rolename
    username [authentication_option]
```

The `GRANT` statement is also used to grant the use of a role to one or more users or other roles. In order to be able to grant a role, the grantor doing so must have permission to do so (see `WITH ADMIN` in the CREATE ROLE article).

Specifying the `WITH ADMIN OPTION` permits the grantee to in turn grant the role to another.

For example, the following commands show how to grant the same role to a couple different users.

```
GRANT journalist TO hulda;

GRANT journalist TO berengar WITH ADMIN OPTION;
```

If a user has been granted a role, they do not automatically obtain all permissions associated with that role. These permissions are only in use when the user activates the role with the SET ROLE statement.

Warning:

Be careful to avoid conflicting role and user names. In case of a conflict, the role name takes precedence, as shown in the following example. The `GRANT` statement assigns privileges to the role, not to the user:

```
CREATE USER alice IDENTIFIED BY 'password';
CREATE ROLE alice;
GRANT select, insert on db.* TO alice;
```

TO PUBLIC

CURRENT

Syntax

```
GRANT <privilege> ON <db_name>.<object> TO PUBLIC;
REVOKE <privilege> ON <db_name>.<object> FROM PUBLIC;
```

`GRANT ... TO PUBLIC` grants privileges to all users with access to the server. The privileges also apply to users created after the privileges are granted. This can be useful when you only want to state once that all users need to have a certain set of privileges. When running SHOW GRANTS, a user also sees all privileges inherited from `PUBLIC`. SHOW GRANTS FOR PUBLIC only shows `TO PUBLIC` grants.

Example

The following example shows the difference between granting privileges to particular users and granting privileges to `PUBLIC`.

```
-- ... (connect as user root) ...
MariaDB [(none)]> CREATE USER developer;
MariaDB [(none)]> CREATE DATABASE dev_db;
MariaDB [(none)]> GRANT ALL ON dev_db.* TO PUBLIC;
MariaDB [(none)]> GRANT ALL ON mysql.* TO developer;
-- ... (connect as user developer) ...
MariaDB [(none)]> SHOW GRANTS;
+-----------------------------------------------+
| Grants for developer@%                        |
+-----------------------------------------------+
| GRANT USAGE ON . TO developer@%               |
| GRANT ALL PRIVILEGES ON mysql.* TO developer@%  |
| GRANT ALL PRIVILEGES ON dev_db.* TO PUBLIC      |
+-----------------------------------------------+
MariaDB [(none)]> SHOW GRANTS FOR PUBLIC;
+-----------------------------------------------+
| Grants for PUBLIC                             |
+-----------------------------------------------+
| GRANT ALL PRIVILEGES ON `dev_db`.* TO `PUBLIC` |
+-----------------------------------------------+
```

For more details, and information on the background of this feature, refer to this blog post.

< 10.11.0

TO PUBLIC is unavailable.

Grant Examples

Granting Root-like Privileges

You can create a user that has privileges similar to the default `root` accounts by executing the following:

```
CREATE USER 'alexander'@'localhost';
GRANT ALL PRIVILEGES ON  *.* TO 'alexander'@'localhost' WITH GRANT OPTION;
```

See Also

- Troubleshooting Connection Issues
- Authentication from MariaDB 10.4
- –skip-grant-tables allows you to start MariaDB without `GRANT`. This is useful if you lost your root password.
- CREATE USER
- ALTER USER
- DROP USER
- DENY blocks a privilege so that no `GRANT` can restore it.
- SET PASSWORD
- SHOW CREATE USER
- mysql.global_priv table
- mysql.user table
- Password Validation Plugins - permits the setting of basic criteria for passwords
- Authentication Plugins - allow various authentication methods to be used, and new ones to be developed.

This page is licensed: GPLv2, originally from fill_help_tables.sql

RENAME USER

Syntax

```
RENAME USER old_user TO new_user
    [, old_user TO new_user] ...
```

Description

The RENAME USER statement renames existing MariaDB accounts. To use it, you must have the global CREATE USER privilege or the UPDATE privilege for the `mysql` database. Each account is named using the same format as for the CREATE USER statement; for example, `'jeffrey'@'localhost'`.
If you specify only the user name part of the account name, a host name part of `'%'` is used.

If any of the old user accounts do not exist or any of the new user accounts already exist, `ERROR 1396 (HY000)` results. If an error occurs, `RENAME USER` will still rename the accounts that do not result in an error.

For modifying an existing account, see ALTER USER.

Examples

```
CREATE USER 'donald', 'mickey';
RENAME USER 'donald' TO 'duck'@'localhost', 'mickey' TO 'mouse'@'localhost';
```

Renaming the host component of a user:

```
RENAME USER 'foo'@'1.2.3.4' TO 'foo'@'10.20.30.40';
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

REVOKE

Privileges

Syntax

```
/* 1. Revoking Privileges */
REVOKE
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM account_or_role [, account_or_role] ...

/* 2. Revoking All Privileges */
REVOKE ALL [PRIVILEGES], GRANT OPTION
    FROM account_or_role [, account_or_role] ...

/* 3. Revoking a Deny (from MariaDB 13.1) */
REVOKE DENY
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM account_or_role [, account_or_role] ...

/* 4. Revoking Proxy Access */
REVOKE PROXY ON user_or_role
    FROM account_or_role [, account_or_role] ...

/* 5. Revoking Roles */
REVOKE role [, role] ...
    FROM account_or_role [, account_or_role] ...

/* 6. Revoking Admin Option for Roles */
REVOKE ADMIN OPTION FOR role [, role] ...
    FROM account_or_role [, account_or_role] ...

/* Variable Definitions */

account_or_role:
    username
  | role
  | PUBLIC
  | CURRENT_USER [()]
  | CURRENT_ROLE [()]

priv_type:
    ALL [PRIVILEGES]
  | ALTER | ALTER ROUTINE | BINLOG ADMIN | BINLOG MONITOR | BINLOG REPLAY
  | CONNECTION ADMIN | CREATE | CREATE ROUTINE | CREATE TABLESPACE
  | CREATE TEMPORARY TABLES | CREATE USER | CREATE VIEW
  | DELETE | DELETE HISTORY | DROP | EVENT | EXECUTE | FEDERATED ADMIN
  | FILE | GRANT OPTION | INDEX | INSERT | LOCK TABLES | PROCESS
  | READ ONLY ADMIN | RELOAD | REPLICATION CLIENT | REPLICATION MASTER ADMIN
  | REPLICATION SLAVE | REPLICATION SLAVE ADMIN | REFERENCES
  | SELECT | SET USER | SHOW CREATE ROUTINE | SHOW DATABASES | SHOW VIEW
  | SHUTDOWN | SLAVE MONITOR | SUPER | TRIGGER | UPDATE | USAGE

object_type:
    TABLE
  | FUNCTION
  | PROCEDURE
  | PACKAGE
  | PACKAGE BODY

priv_level:
    *
  | *.*
  | db_name.*
```

```
  | db_name.tbl_name
  | tbl_name
  | db_name.routine_name
```

Description

The `REVOKE` statement enables system administrators to revoke privileges (or roles - see section below) from MariaDB accounts. Each account is named using the same format as for the `GRANT` statement; for example, `jeffrey@localhost`. If you specify only the user name part of the account name, a host name part of `%` is used. For details on the levels at which privileges exist, the available `priv_type` and `priv_level` values, and the syntax for specifying users and passwords, see GRANT.

To use the first syntax (`REVOKE ... ON ... FROM ...`), you must have the `GRANT OPTION` privilege, and you must have the privileges that you are revoking. Also, remember to specify the `ON` clause (in many cases, `ON *.*` to revoke privileges for all objects):

```
REVOKE ALL ON *.* FROM 'myuser'@'localhost';
```

This leaves the `USAGE` privilege, and can leave other privileges, too. For that reason, the second syntax (`REVOKE ALL PRIVILEGES, GRANT OPTION FROM ...`) is preferable, which drops all global, database, table, column, and routine privileges for the named user or users:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

To use this `REVOKE` syntax, you must have the global CREATE USER privilege or the UPDATE privilege for the mysql database. See GRANT. For that syntax, the `ON *.*` clause must not be used (it yields an error).

> **Note:**
>
> Revoking all privileges doesn't remove *all* privileges, not even with the second syntax. The user still keeps the `USAGE` privilege, making it possible to connect to the server (but nothing else). To remove that privilege, too, remove the user entirely with a `DROP USER` statement.

Revoking a privilege is not the same as blocking it. A privilege that a broader grant still covers remains in effect after the narrower grant is revoked, and any later `GRANT` restores it. To block a privilege outright, use DENY instead.

Examples

Revoking a particular privilege (`SUPER`):

```
REVOKE SUPER ON *.* FROM 'myuser'@'localhost';
```

Revoking all privileges (on all objects – note you must specify `ON *.*`):

```
REVOKE ALL ON *.* FROM 'myuser'@'localhost';
```

Revoking all privileges and the `GRANT` option (note that `ON *.*` cannot be specified here):

```
REVOKE ALL, GRANT OPTION FROM 'myuser'@'localhost';
```

> **Note:**
>
> The `PRIVILEGES` keyword is optional for `ALL PRIVILEGES`.

Denies

From MariaDB 13.1, `REVOKE DENY` removes a negative privilege created with DENY. It is the only way to restore a denied privilege.

```
REVOKE DENY SELECT ON hr.staff FROM 'alice'@'localhost';
```

The privileges, object type, and privilege level must match the deny that is being removed, and the statement fails if no such deny exists — even when a positive grant does. `REVOKE ALL PRIVILEGES, GRANT OPTION` clears an account's denies together with its grants. See DENY for details.

Roles

Syntax

```
REVOKE role  [, role ...]
    FROM grantee [, grantee2 ... ]

REVOKE ADMIN OPTION FOR role FROM grantee [, grantee2]
```

Description

`REVOKE` is also used to remove a role from a user or another role that it's previously been assigned to. If a role has previously been set as a default role, `REVOKE` does not remove the record of the default role from the mysql.user table. If the role is subsequently granted again, it will again be the user's default. Use SET DEFAULT ROLE NONE to explicitly remove this.

> **CURRENT**
>
> `REVOKE role` is also permitted in prepared statements.
>
> **< 10.1.13**
>
> `REVOKE role` is not permitted in prepared statements.

Example

```
REVOKE journalist FROM hulda
```

Revoking Proxy

The `REVOKE PROXY` syntax removes the ability for one user to proxy as another.

```
REVOKE PROXY ON 'dba_user'@'localhost' FROM 'app_user'@'localhost';
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SET DEFAULT ROLE

Syntax

```
SET DEFAULT ROLE { role | NONE } [ FOR user@host ]
```

Description

The `SET DEFAULT ROLE` statement sets a **default role** for a specified (or current) user. A default role is automatically enabled when a user connects (an implicit SET ROLE statement is executed immediately after a

connection is established).

To be able to set a role as a default, the role must already have been granted to that user, and one needs the privileges to enable this role (if you cannot do `SET ROLE X`, you won't be able to do `SET DEFAULT ROLE X`). To set a default role for another user one needs to have write access to the `mysql` database.

To remove a user's default role, use `SET DEFAULT ROLE NONE [ FOR user@host ]`. The record of the default role is not removed if the role is dropped or revoked, so if the role is subsequently re-created or granted, it will again be the user's default role.

The default role is stored in the `default_role` column in the mysql.user table/view, as well as in the Information Schema APPLICABLE_ROLES table, so these can be viewed to see which role has been assigned to a user as the default.

Examples

Setting a default role for the current user:

```
SET DEFAULT ROLE journalist;
```

Removing a default role from the current user:

```
SET DEFAULT ROLE NONE;
```

Setting a default role for another user. The role has to have been granted to the user before it can be set as default:

```
CREATE ROLE journalist;
CREATE USER taniel;

SET DEFAULT ROLE journalist FOR taniel;
ERROR 1959 (OP000): Invalid role specification `journalist`

GRANT journalist TO taniel;
SET DEFAULT ROLE journalist FOR taniel;
```

Viewing mysql.user:

```
SELECT * FROM mysql.user WHERE user='taniel'\G
```

```
*************************** 1. row ***************************
                  Host: %
                  User: taniel
...
               is_role: N
          default_role: journalist
...
```

Removing a default role for another user

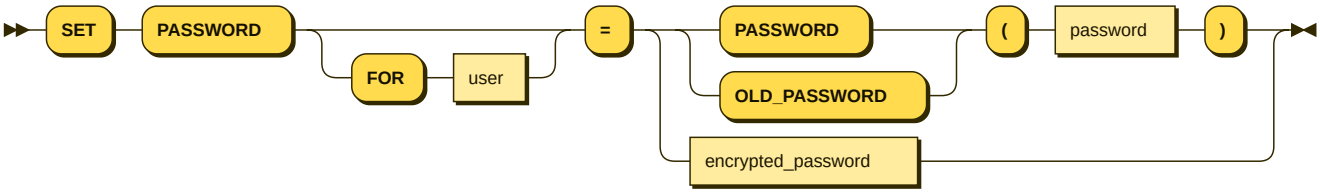
```
SET DEFAULT ROLE NONE FOR taniel;
```

This page is licensed: CC BY-SA / Gnu FDL

SET PASSWORD

Syntax

```
SET PASSWORD [FOR user] =
    {
        PASSWORD('some password')
      | OLD_PASSWORD('some password')
      | 'encrypted password'
    }
```



Railroad diagram of SET PASSWORD — equivalent to the BNF above

Description

The `SET PASSWORD` statement assigns a password to an existing MariaDB user account.

If the password is specified using the PASSWORD() or OLD_PASSWORD() function, the literal text of the password should be given. If the password is specified without using either function, the password should be the already-encrypted password value as returned by PASSWORD().

> **Note:**
>
> OLD_PASSWORD() should only be used if your MariaDB/MySQL clients are very old (< 4.0.0).

With no `FOR` clause, this statement sets the password for the current user. Any client that has connected to the server using a non-anonymous account can change the password for that account.

With a `FOR` clause, this statement sets the password for a specific account on the current server host. Only clients that have the `UPDATE` privilege for the `mysql` database can do this. The user value should be given in `user_name@host_name` format, where `user_name` and `host_name` are exactly as they are listed in the User and Host columns of the mysql.user table (or view in current versions) entry.

The argument to PASSWORD() and the password given to MariaDB clients can be of arbitrary length.

Authentication Plugin Support

`SET PASSWORD` (with or without `PASSWORD()`) works for accounts authenticated via any authentication plugin that supports passwords stored in the mysql.global_priv table.

The ed25519, mysql_native_password, and mysql_old_password authentication plugins store passwords in the mysql.global_priv table.

If you run `SET PASSWORD` on an account that authenticates with one of these authentication plugins that stores passwords in the mysql.global_priv table, then the `PASSWORD()` function is evaluated by the specific authentication plugin used by the account. The authentication plugin hashes the password with a method that is compatible with that specific authentication plugin.

The unix_socket, named_pipe, gssapi, and pam authentication plugins do **not** store passwords in the mysql.global_priv table. These authentication plugins rely on other methods to authenticate the user.

If you attempt to run `SET PASSWORD` on an account that authenticates with one of these authentication plugins that doesn't store a password in the mysql.global_priv table, then MariaDB Server will issue an error like the following:

```
SET PASSWORD IS ignored FOR users authenticating via unix_socket plugin
```

Note:

See Authentication from MariaDB 10.4 for an overview of authentication changes in MariaDB.

Passwordless User Accounts

User accounts do not always require passwords to login.

The unix_socket , named_pipe and gssapi authentication plugins do not require a password to authenticate the user.

The pam authentication plugin may or may not require a password to authenticate the user, depending on the specific configuration.

The mysql_native_password and mysql_old_password authentication plugins require passwords for authentication, but the password can be blank. In that case, no password is required.

If you provide a password while attempting to log into the server as an account that doesn't require a password, then MariaDB server will simply ignore the password.

A user account can be defined to use multiple authentication plugins in a specific order of preference. This specific scenario may be more noticeable in these versions, since an account could be associated with some authentication plugins that require a password, and some that do not.

Example

For example, if you had an entry with User and Host column values of '`bob`' and '`%.loc.gov`', you would write the statement like this:

```
SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('newpass');
```

If you want to delete a password for a user, you would do:

```
SET PASSWORD FOR 'bob'@localhost = PASSWORD("");
```

See Also

- Password Validation Plugins - permits the setting of basic criteria for passwords
- ALTER USER

This page is licensed: GPLv2, originally from fill_help_tables.sql

SET SESSION AUTHORIZATION

Note:

This statement is available from MariaDB 12.0.

Certain users can perform server actions as another user:

- This is implemented through the `SET SESSION AUTHORIZATION` statement.
- This permits everything that can be done in a stored procedure with an arbitrary definer.
- In particular, this bypasses account lock, expired password, authentication, `REQUIRE SSL` checks, and so on.
- Users are required to have the SET USER privilege.
- Does not work inside transactions, prepared statements and stored procedures.

Examples

```
SELECT USER(), CURRENT_USER(), DATABASE();
```

```
+--------------------+--------------------+------------+
| user()             | current_user()     | database() |
+--------------------+--------------------+------------+
| msandbox@localhost | msandbox@localhost | test       |
+--------------------+--------------------+------------+
1 row in set (0.000 sec)
```

```
SET SESSION AUTHORIZATION foo@localhost;
SELECT USER(), CURRENT_USER(), DATABASE();
```

```
+---------------+----------------+------------+
| user()        | current_user() | database() |
+---------------+----------------+------------+
| foo@localhost | foo@%          | NULL       |
+---------------+----------------+------------+
```

This page is licensed: CC BY-SA / Gnu FDL

SET ROLE

Syntax

```
SET ROLE { role | NONE }
```

Description

> **Note:**
>
> Only one role can be current at a time. Executing `SET ROLE` replaces the current role; it does not add to a list of current roles. This is SQL Standard compliant behavior which differs from MySQL, where you may have several current roles at a time.

The `SET ROLE` statement switches the current role for the session, enabling its associated permissions. To have no current role, set `NONE`.

If a role that doesn't exist, or to which the user has not been assigned, is specified, an `ERROR 1959 (OP000): Invalid role specification` error occurs.

An automatic SET ROLE is implicitly performed when a user connects if that user has been assigned a default role. See SET DEFAULT ROLE.

Example

```
--Checking the current role status
SELECT CURRENT_ROLE;
```

`No role active`

```
+--------------+
| CURRENT_ROLE |
+--------------+
| NULL         |
+--------------+
```

```
--Setting the staff role, and verifying the switch
SET ROLE staff;
SELECT CURRENT_ROLE;
```

``

```
+--------------+
| CURRENT_ROLE |
+--------------+
| staff        |
+--------------+
```

```
--switching to 'admin' role, and verifying the switch
SET ROLE admin;
SELECT CURRENT_ROLE;
```

``

```
+--------------+
| CURRENT_ROLE |
+--------------+
| admin        |
+--------------+
```

```
--Removing the active role
SET ROLE NONE;
SELECT CURRENT_ROLE();
```

```
+----------------+
| CURRENT_ROLE() |
+----------------+
| NULL           |
+----------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Administrative Statements

See BINLOG.

Executes binary log events directly using base64-encoded data. Primarily used by the mariadb-binlog utility to re-apply events to the server.

See CACHE INDEX.

Assigns specific table indices to a named key cache. Optimizes server performance by preloading or dedicating memory to frequently accessed keys.

See DESCRIBE.

Provides information about a table's columns. Acts as a shortcut for SHOW COLUMNS, displaying field names, types, and other attributes.

See HELP Command.

Displays help information from the server's help tables. Useful for looking up SQL syntax and command descriptions directly from the client.

See KILL.

Terminates a specific connection or query. Allows administrators to stop runaway threads or disconnect users to free up server resources.

See PURGE BINARY LOGS.

Official PURGE BINARY LOGS syntax: delete binlogs using TO ‘log_name’ or BEFORE datetime_expr, replica read constraints, and SHOW BINARY LOGS commands.

See RESET.

Clears internal server buffers, caches, and status variables. Resets state information like the query cache or replication status without a restart.

See SHUTDOWN.

Stops the MariaDB server process. Allows a client with the SHUTDOWN privilege to cleanly power down the database instance.

See ANALYZE and EXPLAIN Statements.

Learn commands for query analysis. This section covers ANALYZE TABLE and EXPLAIN, used to view execution plans and optimize query performance.

See BACKUP Statements.

Learn about backup statements for MariaDB Server. This section details SQL statements and utilities for creating consistent database backups, essential for disaster recovery and data protection.

See FLUSH Statements.

Explore commands to clear internal caches. Learn to use FLUSH to reload privileges, clear the query cache, or close open tables.

See Plugin Statements.

Manage server plugins. This section covers INSTALL PLUGIN, UNINSTALL PLUGIN, and SHOW PLUGINS for extending server functionality.

See Replication Statements.

Control replication topologies. Learn statements like CHANGE MASTER TO and START SLAVE to configure primaries and replicas.

See SET Statements.

Assign values to system variables. Learn to use the SET statement to configure GLOBAL and SESSION variables for tuning server behavior.

See SHOW Statements.

View server metadata and status. This section lists SHOW statements for inspecting databases, tables, variables, and performance metrics.

This page is licensed: CC BY-SA / Gnu FDL

BINLOG

Syntax

```
BINLOG 'str'
```

Description

`BINLOG` is an internal-use statement. It is generated by the mariadb-binlog program as the printable representation of certain events in binary log files. The `'str'` value is a base 64-encoded string that the server decodes to determine the data change indicated by the corresponding event.

> **CURRENT**
>
> This statement requires the BINLOG REPLAY privilege.
>
> **< 10.5.1**
>
> This statement requires the SUPER privilege.

See Also

- MariaDB replication

This page is licensed: GPLv2, originally from fill_help_tables.sql

CACHE INDEX

Syntax

```
CACHE INDEX
  tbl_index_list [, tbl_index_list] ...
  IN key_cache_name

tbl_index_list:
  tbl_name [[INDEX|KEY] (index_name[, index_name] ...)]
```

Description

The `CACHE INDEX` statement assigns table indexes to a specific key cache. It is used only for MyISAM tables.

A default key cache exists and cannot be destroyed. To create more key caches, the key_buffer_size server system variable.

The associations between tables indexes and key caches are lost on server restart. To recreate them automatically, it is necessary to configure caches in a configuration file and include some `CACHE INDEX` (and optionally LOAD INDEX) statements in the init file.

Examples

The following statement assigns indexes from the tables t1, t2, and t3 to the key cache named hot_cache:

```
CACHE INDEX t1, t2, t3 IN hot_cache;
+---------+--------------------+----------+----------+
| Table   | Op                 | Msg_type | Msg_text |
+---------+--------------------+----------+----------+
| test.t1 | assign_to_keycache | status   | OK       |
| test.t2 | assign_to_keycache | status   | OK       |
| test.t3 | assign_to_keycache | status   | OK       |
+---------+--------------------+----------+----------+
```

Implementation (for MyISAM)

Normally CACHE INDEX should not take a long time to execute. Internally it's implemented the following way:

- Find the right key cache (under LOCK_global_system_variables)
- Open the table with a TL_READ_NO_INSERT lock.

- Flush the original key cache for the given file (under key cache lock)
- Flush the new key cache for the given file (safety)
- Move the file to the new key cache (under file share lock)

The only possible long operations are getting the locks for the table and flushing the original key cache, if there were many key blocks for the file in it.

We plan to also add CACHE INDEX for Aria tables if there is a need for this.

This page is licensed: GPLv2, originally from fill_help_tables.sql

DESCRIBE

Syntax

```
{DESCRIBE | DESC} tbl_name [col_name | wild]
```

Description

`DESCRIBE` provides information about the columns in a table.It is a shortcut for SHOW COLUMNS FROM.These statements also display information for views.

`col_name` can be a column name, or a string containing theSQL "`%`" and "`_`" wildcard characters toobtain output only for the columns with names matching the string. There is noneed to enclose the string within quotes unless it contains spaces or otherspecial characters.

```
DESCRIBE city;
+------------+----------+------+-----+---------+----------------+
| Field      | Type     | Null | Key | Default | Extra          |
+------------+----------+------+-----+---------+----------------+
| Id         | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name       | char(35) | YES  |     | NULL    |                |
| Country    | char(3)  | NO   | UNI |         |                |
| District   | char(20) | YES  | MUL |         |                |
| Population | int(11)  | YES  |     | NULL    |                |
+------------+----------+------+-----+---------+----------------+
```

The description for SHOW COLUMNS providesmore information about the output columns.

See Also

- SHOW COLUMNS
- INFORMATION_SCHEMA.COLUMNS Table
- mysqlshow

This page is licensed: GPLv2, originally from fill_help_tables.sql

HELP Command

Syntax

```
HELP search_string
```

Description

The `HELP` command can be used in any MariaDB client, such as the mariadb command-line client, to get basic syntax help and a short description for most commands and functions.

If you provide an argument to the `HELP` command, the mariadb client uses it as a search string to access server-side help. The proper operation of this command requires that the help tables in the `mysql` database be initialized with help topic information.

If there is no match for the search string, the search fails. Use `HELP contents` to see a list of the help categories:

```
HELP contents
You asked for help about help category: "Contents"
For more information, type 'help <item>', where <item> is one of the following
categories:
   Account Management
   Administration
   Compound Statements
   Data Definition
   Data Manipulation
   Data Types
   Functions
   Functions and Modifiers for Use with GROUP BY
   Geographic Features
   Help Metadata
   Language Structure
   Plugins
   Procedures
   Sequences
   Table Maintenance
   Transactions
   User-Defined Functions
   Utility
```

If a search string matches multiple items, MariaDB shows a list of matching topics:

```
HELP drop
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following
topics:
   ALTER TABLE
   DROP DATABASE
   DROP EVENT
   DROP FUNCTION
   DROP FUNCTION UDF
   DROP INDEX
   DROP PACKAGE
   DROP PACKAGE BODY
   DROP PROCEDURE
   DROP ROLE
   DROP SEQUENCE
   DROP SERVER
   DROP TABLE
   DROP TRIGGER
   DROP USER
   DROP VIEW
```

Then you can enter a topic as the search string to see the help entry for that topic.

The help is provided with the MariaDB server and makes use of four help tables found in the `mysql` database: help_relation, help_topic, help_category and help_keyword. These tables are populated by the mariadb-install-db or `fill_help_table.sql` scripts.

This page is licensed: GPLv2, originally from fill_help_tables.sql

KILL

Syntax

```
KILL [HARD | SOFT] { {CONNECTION|QUERY} thread_id | QUERY ID query_id | USER user_name }
```

Description

Each connection to mariadbd runs in a separate thread. You can see which threads are running with the `SHOW PROCESSLIST` statement and kill a thread with the `KILL thread_id` statement. `KILL` allows the optional `CONNECTION` or `QUERY` modifier:

- `KILL CONNECTION` is the same as `KILL` with no modifier: It terminates the connection associated with the given thread or query id.
- `KILL QUERY` terminates the statement that the connection `thread_id` is currently executing, but leaves the connection itself intact.
- `KILL QUERY ID` terminates the query by `query_id`, leaving the connection intact.

If a connection is terminated that has an active transaction, the transaction is rolled back. If only a query is killed, the current transaction stays active. See also idle_transaction_timeout.

If you have the PROCESS privilege, you can see all threads.

CURRENT

If you have the CONNECTION ADMIN privilege, you can kill all threads and statements. Otherwise, you can see and kill only your own threads and statements.

< 10.5.2

If you have the SUPER privilege, the CONNECTION ADMIN privilege, you can kill all threads and statements. Otherwise, you can see and kill only your own threads and statements.

Warning:

Killing queries that repair or create indexes on MyISAM and Aria tables may result in corrupted tables. Use the `SOFT` option to avoid this.

The `HARD` option (default) kills a statement as soon as possible. If you use `SOFT`, critical operations that may leave a table in an inconsistent state are not interrupted. Such operations include `REPAIR` and `INDEX` creation for MyISAM and Aria tables (REPAIR TABLE, OPTIMIZE TABLE).

`KILL ... USER username` kills all connections and queries for a given user. `USER` can be specified in one of the following ways:

- username (kill without regard to hostname)
- username@hostname
- CURRENT_USER or CURRENT_USER()

If you specify a thread id and that thread does not exist, you get the following error:

```
ERROR 1094 (HY000): Unknown thread id: <thread_id>
```

If you specify a query id that doesn't exist, you get the following error:

```
ERROR 1957 (HY000): Unknown query id: <query_id>
```

However, if you specify a user name, no error is issued for non-connected (or even non-existing) users. To check if the connection/query has been killed, you can use the ROW_COUNT() function.

A client whose connection is killed receives the following error:

```
ERROR 1317 (70100): Query execution was interrupted
```

To obtain a list of existing sessions, use the SHOW PROCESSLIST statement or query the Information Schema PROCESSLIST table.

> **Note:**
>
> You cannot use `KILL` with the Embedded MariaDB Server library because the embedded server merely runs inside the threads of the host application. It does not create any connection threads of its own.

Note: You can also use `mariadb-admin kill thread_id [,thread_id...]` to kill connections. To get a list of running queries,use `mariadb-admin processlist`. See mariadb-admin.

See Also

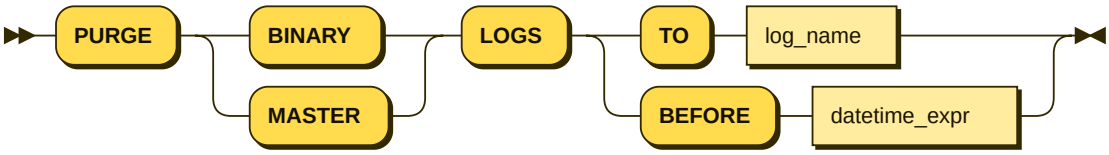
- Query limits and timeouts
- Aborting statements that exceed a certain time to execute
- idle_transaction_timeout

This page is licensed: GPLv2, originally from fill_help_tables.sql

PURGE BINARY LOGS

Syntax

```
PURGE { BINARY | MASTER } LOGS
    { TO 'log_name' | BEFORE datetime_expr }
```



Railroad diagram of PURGE BINARY LOGS — equivalent to the BNF above

Description

The `PURGE BINARY LOGS` statement deletes all the binary log files listed in the log index file prior to the specified log file name ordate. `BINARY` and `MASTER` are synonyms.Deleted log files also are removed from the list recorded in the index file, sothat the given log file becomes the first in the list.

The datetime expression is in the format 'YYYY-MM-DD hh:mm:ss'.

If a replica is active but has yet to read from a binary log file you attempt to delete, the statement will fail with an error. However, if the replica is not connected and has yet to read from a log file you delete, the file will be deleted, but the replica will be unable to continue replicating once it connects again.

This statement has no effect if the server was not started with the –log-bin option to enable binary logging.

> **CURRENT**
>
> To list the binary log files on the server, use SHOW BINARY LOGS. To see which files they are reading, use SHOW REPLICA STATUS. You can only delete the files that are older than the oldest file that is used by the

slaves.

< 10.5.1

To list the binary log files on the server, use SHOW BINARY LOGS. To see which files they are reading, use SHOW SLAVE STATUS. You can only delete the files that are older than the oldest file that is used by the slaves.

To delete all binary log files, use RESET MASTER.To move to a new log file (for example if you want to remove the current log file), use FLUSH LOGS before you execute `PURGE LOGS`.

If the expire_logs_days server system variable is not set to 0, the server automatically deletes binary log files after the given number of days. From MariaDB 10.6, the binlog_expire_logs_seconds variable allows more precise control over binlog deletion, and takes precedence if both are non-zero.

CURRENT

Requires the BINLOG ADMIN privilege.

< 10.5.2

Requires the SUPER privilege.

Examples

```
PURGE BINARY LOGS TO 'mariadb-bin.000063';
```

```
PURGE BINARY LOGS BEFORE '2013-04-21';
```

```
PURGE BINARY LOGS BEFORE '2013-04-22 09:55:22';
```

See Also

- Using and Maintaining the Binary Log
- FLUSH LOGS.

This page is licensed: GPLv2, originally from fill_help_tables.sql

RESET

Syntax

```
RESET reset_option [, reset_option] ...
```

Description

The `RESET` statement is used to clear the state of various server operations. You must have the RELOAD privilege to execute `RESET`.

`RESET` acts as a stronger version of the FLUSH statement.

The different `RESET` options are:

Option	Description
SLAVE [“connection_name”] [ALL]	Deletes all relay logs from the slave and reset the replication position in the master binary log.

Option	Description
MASTER	Deletes all old binary logs, makes the binary index file (–log-bin-index) empty and creates a new binary log file. This is useful when you want to reset the master to an initial state. If you want to just delete old, not used binary logs, you should use the PURGE BINARY LOGS command.
QUERY CACHE	Removes all queries from the query cache. See also FLUSH QUERY CACHE.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHUTDOWN

Syntax

```
SHUTDOWN [WAIT FOR ALL { SLAVES | REPLICAS } ]
```

Description

The `SHUTDOWN` command shuts the server down.

WAIT FOR ALL SLAVES / REPLICAS

> **CURRENT**
>
> The `WAIT FOR ALL REPLICAS` statement can be used as an alias for `WAIT FOR ALL SLAVES`.
>
> **< 10.5.1**
>
> The `WAIT FOR ALL SLAVES` option was first added in MariaDB 10.4.4.

When a primary server is shutdown and it goes through the normal shutdown process, the primary kills client threads in random order. By default, the primary also considers its binary log dump threads to be regular client threads. As a consequence, the binary log dump threads can be killed while client threads still exist, and this means that data can be written on the primary during a normal shutdown that won't be replicated. This is true even if semi-synchronous replication is being used.

This problem can be solved by shutting down the server with the SHUTDOWN command and by providing the `WAIT FOR ALL SLAVES` / `WAIT FOR ALL REPLICAS` option to the command. For example:

```
SHUTDOWN WAIT FOR ALL SLAVES;
```

When the `WAIT FOR ALL SLAVES` option is provided, the server only kills its binary log dump threads after all client threads have been killed, and it only completes the shutdown after the last binary log has been sent to all connected replicas.

See Replication Threads: Binary Log Dump Threads and the Shutdown Process for more information.

Required Permissions

One must have a `SHUTDOWN` privilege (see GRANT) to use this command. It is the same privilege one needs to use the mariadb-admin shutdown command.

Shutdown for Upgrades

If you are doing a shutdown to migrate to another major version of MariaDB, please ensure that the innodb_fast_shutdown variable is not 2 (fast crash shutdown). The default of this variable is 1.

Example

The following example shows how to create an event which turns off the server at a certain time:

```
CREATE EVENT `test`.`shutd`
    ON SCHEDULE
        EVERY 1 DAY
        STARTS '2014-01-01 20:00:00'
    COMMENT 'Shutdown Maria when the office is closed'
DO BEGIN
    SHUTDOWN;
END;
```

Other Ways to Stop mariadbd

You can use the mariadb-admin shutdown command to take down mariadbd cleanly.

You can also use the system kill command on Unix with signal SIGTERM (15)

```
kill -SIGTERM pid-of-mariadbd-process
```

You can find the process number of the server process in the file that ends with `.pid` in your data directory.

The above is identical to `mariadb-admin shutdown`.

On windows you should use:

```
NET STOP MariaDB
```

See Also

- mariadb-admin shutdown.
- InnoDB fast shutdown option

This page is licensed: CC BY-SA / Gnu FDL

USE [DATABASE]

Syntax

```
USE db_name
```

Description

The `'USE db_name'` statement tells MariaDB to use the `db_name` database as the default (current) database for subsequent statements. The database remains the default until the end of the session, or until another `USE` statement is issued:

```
USE db1;
SELECT COUNT(*) FROM mytable;   # selects FROM db1.mytable
USE db2;
SELECT COUNT(*) FROM mytable;   # selects FROM db2.mytable
```

The DATABASE() function (SCHEMA() is a synonym) returns the default database.

Another way to set the default database is by specifying its name when starting the mariadb command line client:

```
mariadb db1    # starts the mariadb command-line client with db1 as the current database
```

One cannot use `USE` to a database one has no privileges to. The reason is that a user with no privileges to a database should not be able to know if a database exists or not.

See Also

- Identifier Qualifiers
- USE CATALOG

This page is licensed: GPLv2, originally from fill_help_tables.sql

ANALYZE and EXPLAIN Statements

See ANALYZE FORMAT=JSON.

Gain deep insight into query execution with JSON-formatted analysis. This command combines optimizer estimates with actual runtime statistics for precise performance tuning.

See ANALYZE FORMAT=JSON Examples.

Review practical examples of ANALYZE FORMAT=JSON output. Learn to identify performance bottlenecks by comparing estimated costs against actual execution metrics.

See ANALYZE: Interpreting rows and filtered members.

Understand the r_rows and r_filtered fields in analysis output. Learn how these actual runtime counters compare to the optimizer's rows and filtered estimates.

See ANALYZE Statement.

Learn to use the ANALYZE statement to execute a query and produce a performance report. This command reveals how close the optimizer's plan was to the actual execution.

See EXPLAIN ANALYZE.

Understand the historical context of EXPLAIN ANALYZE in MariaDB. Learn how this syntax maps to the modern ANALYZE statement for profiling query execution.

See EXPLAIN FORMAT=JSON.

Get comprehensive query plans in JSON format. This output provides detailed optimizer data, including costs and attached conditions, not found in the tabular view.

See EXPLAIN.

Complete EXPLAIN statement reference: SELECT/UPDATE/DELETE syntax, EXTENDED and PARTITIONS options, FORMAT=JSON output, EXPLAIN FOR CONNECTION usage.

See Using Buffer UPDATE Algorithm.

Understand the 'Using buffer' strategy for UPDATE operations. Learn how MariaDB prevents infinite update loops when modifying indexed columns during a range scan.

This page is licensed: CC BY-SA / Gnu FDL

ANALYZE FORMAT=JSON

`ANALYZE FORMAT=JSON` is a mix of the EXPLAIN FORMAT=JSON and ANALYZE statement features. The `ANALYZE FORMAT=JSON $statement` will execute `$statement`, and then print the output of `EXPLAIN FORMAT=JSON`, amended with data from the query execution.

Basic Execution Data

You can get the following also from tabular `ANALYZE` statement form:

- `r_rows` is provided for any node that reads rows. It shows how many rows were read, on average.
- `r_filtered` is provided whenever there is a condition that is checked. It shows the percentage of rows left after checking the condition.

Advanced Execution Data

The most important data not available in the regular tabular `ANALYZE` statement are:

- `r_loops` field. This shows how many times the node was executed. Most query plan elements have this field.
- `r_total_time_ms` field. It shows how much time in total, in milliseconds, was spent executing this node. If the node has subnodes, their execution time is included.
  - For UPDATE and DELETE statements, top-level `query_block.r_total_time_ms` does include the time to make row deletions/updates but does NOT include the time to commit the changes.
- `r_buffer_size` field. Query plan nodes that make use of buffers report the size of buffer that was used.

InnoDB engine statistics

Starting from MariaDB 10.6.15, MariaDB 10.8.8, MariaDB 10.9.8, MariaDB 10.10.6, MariaDB 10.11.5, MariaDB 11.0.3, MariaDB 11.1.2 and MariaDB 11.2.1 (MDEV-31577), the following statistics are reported for InnoDB tables:

```
"r_engine_stats": {
        "pages_accessed":  integer,
        "pages_updated": integer,
        "pages_read_count": integer,
        "pages_prefetch_read_count": integer,
        "pages_read_time_ms": double,
        "old_rows_read": integer
      }
```

Only non-zero members are printed.

- `pages_accessed` is the total number of buffer pool pages accessed when reading this table.
- `pages_updated` is the total number of buffer pool pages that were modified during the execution of the statement.
- `pages_read_count` is the number of pages that InnoDB had to read from disk for this table. If the query touches “hot” data in the InnoDB buffer pool, this value will be 0 and not present.
- `pages_prefetch_read_count` Number of pages for which read-ahead was initiated. Not all such pages will necessarily be accessed.
- `pages_read_time_ms` is the total time spent reading the table.
- `old_rows_read` is the number of old row versions that InnoDB had to read. Old row version is the version of the row that is not visible to this transaction.

SHOW ANALYZE FORMAT=JSON

MariaDB starting with 10.9

`SHOW ANALYZE FORMAT=JSON for <connection_id>`
extends `ANALYZE [FORMAT=JSON] <select>` to allow one to analyze a query currently running in another connection.

Data About Individual Query Plan Nodes

- `filesort` node reports whether sorting was done with `LIMIT n` parameter, and how many rows were in the sort result.
- `block-nl-join` node has `r_loops` field, which allows to tell whether `Using join buffer` was efficient.
- `range-checked-for-each-record` reports counters that show the result of the check.
- `expression-cache` is used for subqueries, and it reports how many times the cache was used, and what cache hit ratio was.
- `union_result` node has `r_rows` so one can see how many rows were produced after UNION operation and so forth.

Use Cases

See Examples of ANALYZE FORMAT=JSON.

This page is licensed: CC BY-SA / Gnu FDL

ANALYZE FORMAT=JSON Examples

Example #1

Customers who have ordered more than 1M goods.

```
ANALYZE FORMAT=JSON
SELECT COUNT(*)
FROM customer
WHERE
  (SELECT SUM(o_totalprice) FROM orders WHERE o_custkey=c_custkey) > 1000*1000;
```

The query takes 40 seconds over cold cache.

```
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "r_loops": 1,
    "r_total_time_ms": 39872,
    "table": {
      "table_name": "customer",
      "access_type": "index",
      "key": "i_c_nationkey",
      "key_length": "5",
      "used_key_parts": ["c_nationkey"],
      "r_loops": 1,
      "rows": 150303,
      "r_rows": 150000,
      "r_total_time_ms": 270.3,
      "filtered": 100,
      "r_filtered": 60.691,
      "attached_condition": "((subquery#2) > <cache>((1000 * 1000)))",
      "using_index": true
    },
    "subqueries": [
      {
        "query_block": {
          "select_id": 2,
          "r_loops": 150000,
          "r_total_time_ms": 39531,
          "table": {
            "table_name": "orders",
            "access_type": "ref",
            "possible_keys": ["i_o_custkey"],
            "key": "i_o_custkey",
            "key_length": "5",
            "used_key_parts": ["o_custkey"],
            "ref": ["dbt3sf1.customer.c_custkey"],
            "r_loops": 150000,
            "rows": 7,
            "r_rows": 10,
            "r_total_time_ms": 39208,
            "filtered": 100,
            "r_filtered": 100
          }
        }
      }
    ]
  }
}
```


`ANALYZE` shows that 39.208 seconds were spent in the subquery, which was executed 150K times (for every row of outer table).

This page is licensed: CC BY-SA / Gnu FDL

ANALYZE: Interpreting rows and filtered members

This article describes how to interpret `r_rows` and `r_filtered` members in `ANALYZE FORMAT=JSON` when an index-based access method is used.

Index-based access method

Index-based access method may employ some or all of the following:

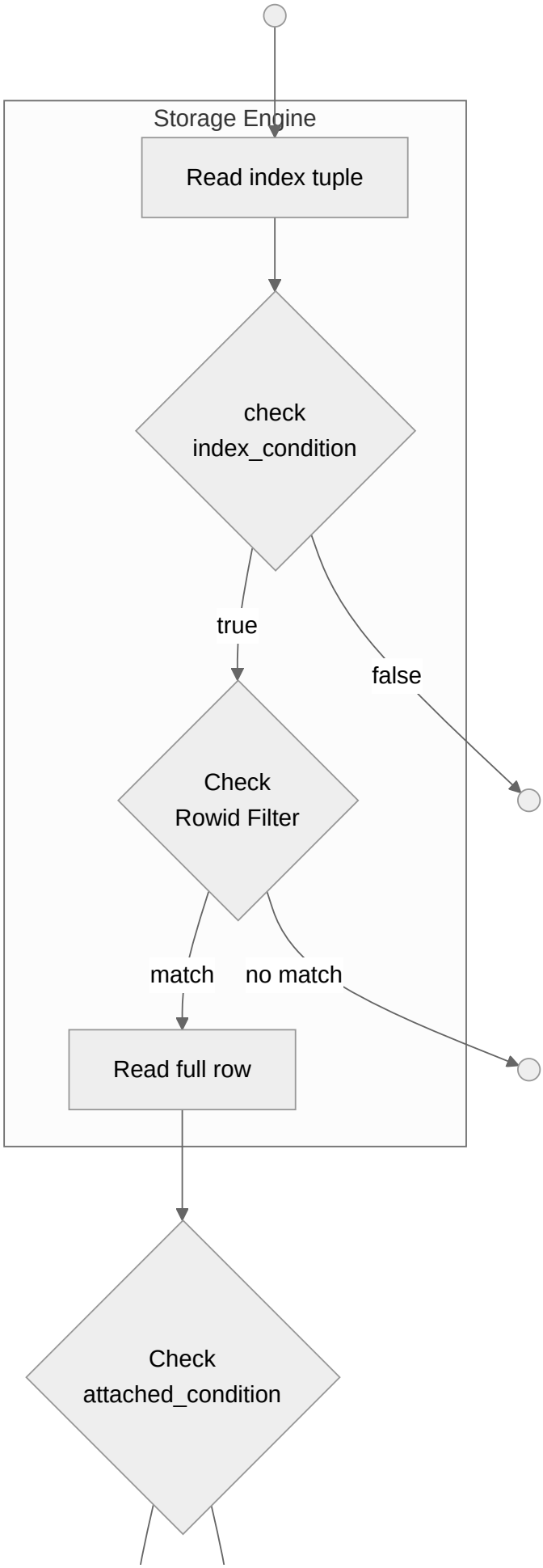
- Index Condition Pushdown
- Rowid Filtering
- attached_condition checking

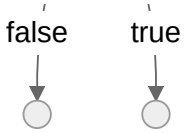
Consider a table access which does all three:

```
"table": {
    "table_name": "t1",
    "access_type": "range",
    "possible_keys": ...,
    "key": "INDEX1",
    ...
    "rowid_filter": {
      ...
      "r_selectivity_pct": n.nnn,
    },
    ...
    "rows": 123,
    "r_rows": 125,
    ...
    "filtered": 8.476269722,
    "r_filtered": 100,
    "index_condition": "cond1",
    "attached_condition": "cond2"
  }
```

The access is performed as follows:

Access diagram

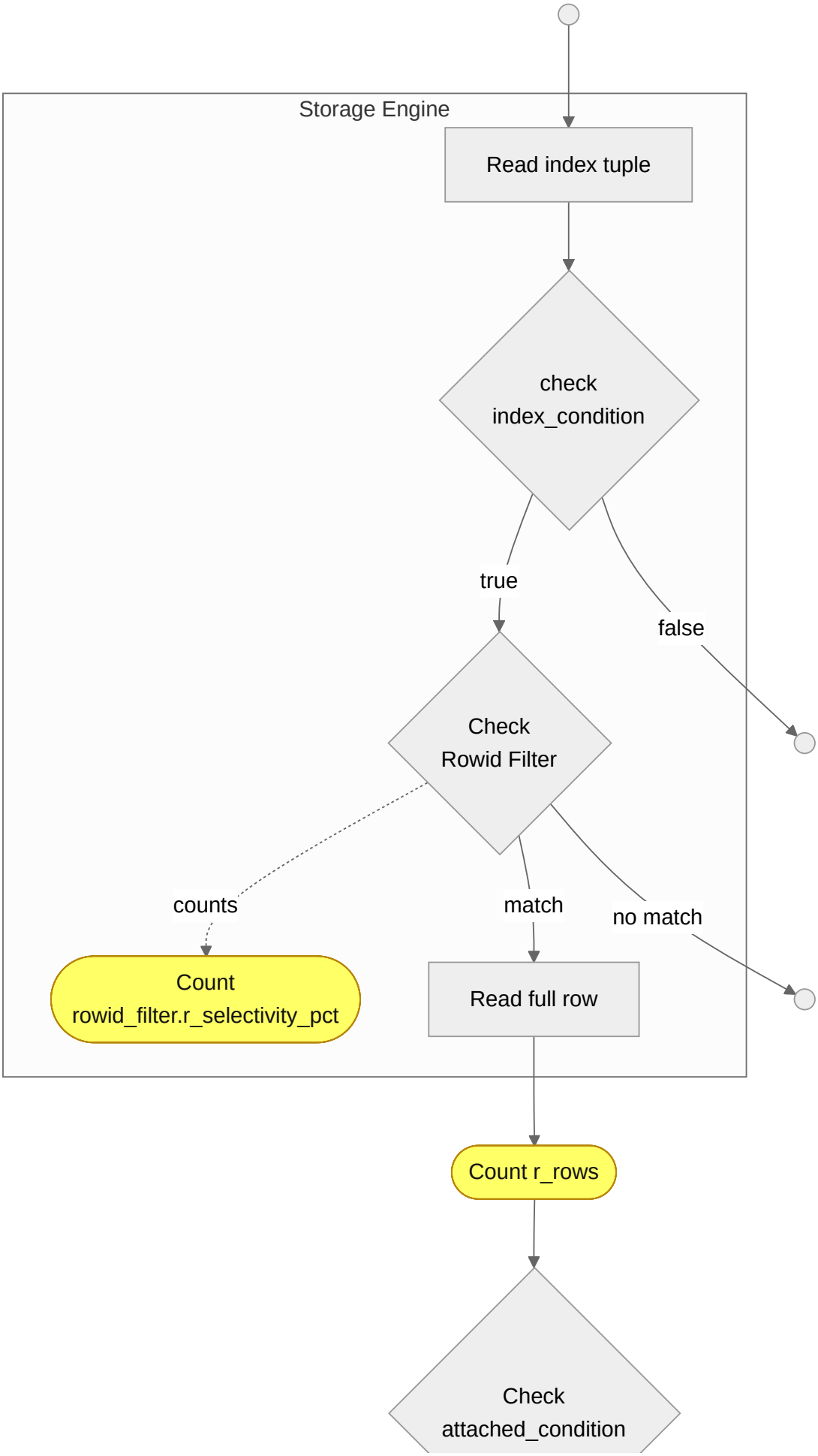

Storage Engine
Read index tuple
check
index_condition
true
false
Check
Rowid Filter
match
no match
Read full row
Check
attached_condition

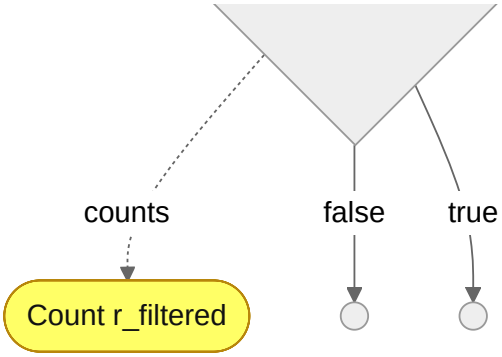


The storage engine applies Index Condition Pushdown and the Rowid Filter internally; the server checks `attached_condition` *afterwards, outside the storage engine.*

Statistics values in MariaDB before 11.5

In MariaDB versions before 11.5, the counters were counted as follows:


Storage Engine
Read index tuple
check
index_condition
true
false
Check
Rowid Filter
counts
match
no match
Count
rowid_filter.r_selectivity_pct
Read full row
Count r_rows
Check
attached_condition



Before MariaDB 11.5, `r_rows` is counted after the index and Rowid Filter checks, while `r_filtered` counts only the `attached_condition` selectivity.

that is,

- `r_rows` is counted after Index Condition Pushdown check and Rowid Filter check.
- `r_filtered` only counts selectivity of the `attached_condition`.
- selectivity of the Rowid Filter is in `rowid_filter.r_selectivity_pct`.

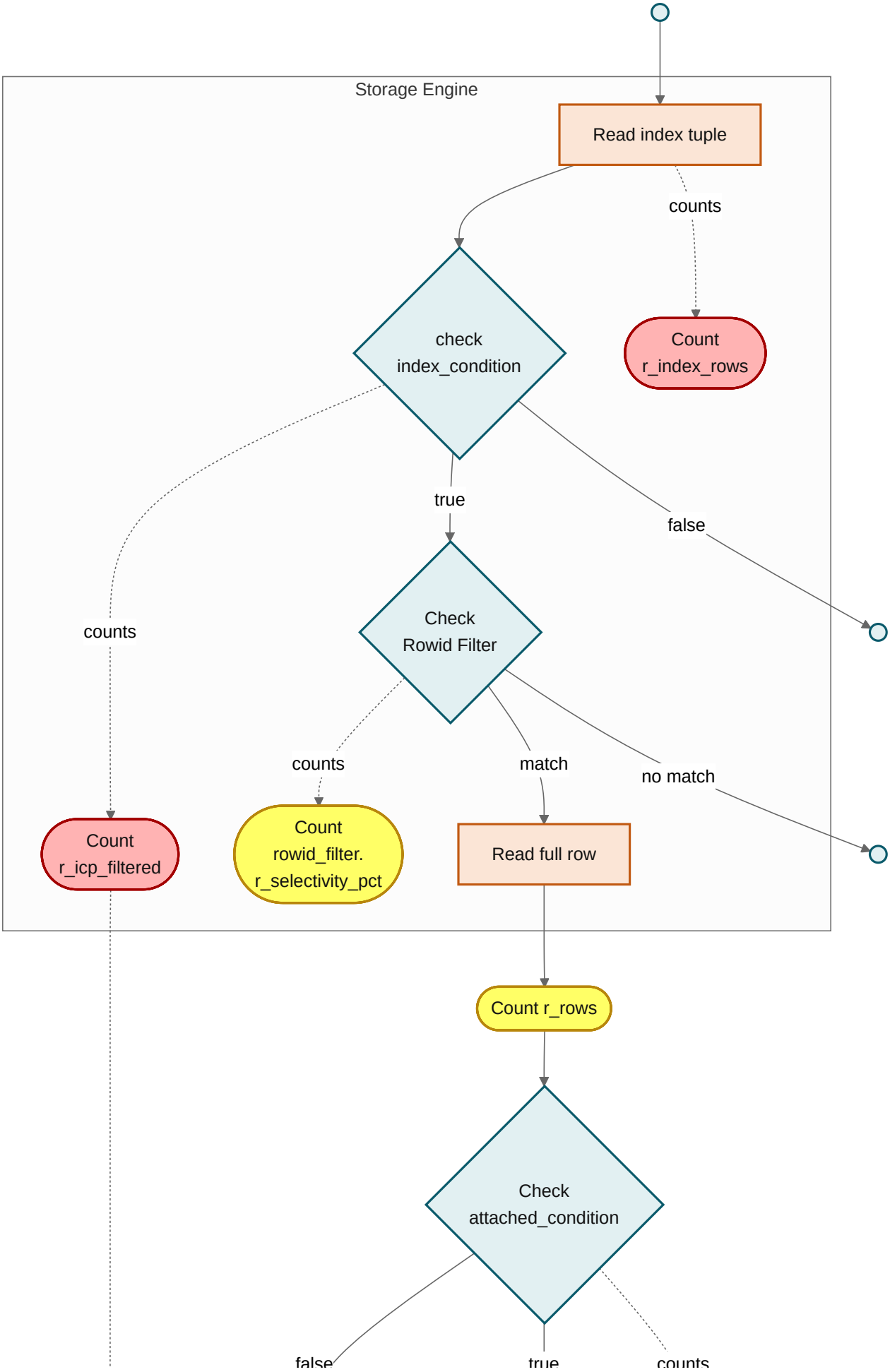
Statistics values in MariaDB 11.5 and later versions

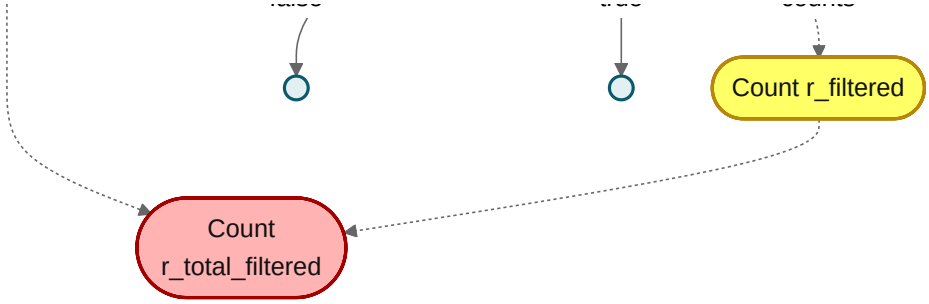
Starting from MariaDB 11.5 (MDEV-18478), the row counters are:

- `r_index_rows` counts the number of enumerated index tuples, before any checks are made
- `r_rows` is the same as before - number of rows after index checks.

The selectivity counters are:

- `r_icp_filtered` is the percentage of records left after pushed index condition check.
- `rowid_filter.r_selectivity_pct` shows selectivity of Rowid Filter, as before.
- `r_filtered` is the selectivity of `attached_condition` check, as before.
- `r_total_filtered` is the combined selectivity of all checks.


Storage Engine
Read index tuple
counts
check
index_condition
Count
r_index_rows
true
false
counts
Check
Rowid Filter
counts
match
no match
Count
r_icp_filtered
Count
rowid_filter.
r_selectivity_pct
Read full row
Count r_rows
Check
attached_condition
false
true
counts



MariaDB 11.5 and later add `r_index_rows`, `r_icp_filtered`, and `r_total_filtered` (highlighted in red) alongside the existing `rowid_filter.r_selectivity_pct`, `r_rows`, and `r_filtered` counters (highlighted in yellow).

ANALYZE output members

in ANALYZE FORMAT=JSON output these members are placed as follows:

```
"table": {
    "table_name": ...,

    "rows": 426,
    "r_index_rows": 349,
    "r_rows": 34,
```

Whenever applicable, `r_index_rows` is shown. It is comparable with `rows` - both are numbers of rows to enumerate before any filtering is done.
If `r_index_rows` is not shown, `r_rows` shows the number of records enumerated.

Then, filtering members:

```
...
    "filtered": 8.476269722,
    "r_total_filtered": 9.742120344,
```

`filtered` is comparable with `r_total_filtered`: both show total amount of filtering.

```
...
    "index_condition": "lineitem.l_quantity > 47",
    "r_icp_filtered": 100,
```

ICP and its observed filtering. The optimizer doesn't compute an estimate for this currently.

```
...
    "attached_condition": "lineitem.l_shipDATE between '1997-01-01' and '1997-06-30'",
    "r_filtered": 100
```

`attached_condition` and its observed filtering.

This page is licensed: CC BY-SA / Gnu FDL

ANALYZE Statement

Description

The `ANALYZE statement` is similar to the `EXPLAIN statement`. `ANALYZE statement` will invoke the optimizer, execute the statement, and then produce `EXPLAIN` output instead of the result set. The `EXPLAIN` output will be annotated with statistics from statement execution.

This lets one check how close the optimizer's estimates about the query plan are to the reality. `ANALYZE` produces an overview, while the ANALYZE FORMAT=JSON command provides a more detailed view of the query plan and the query execution.

The syntax is

```
ANALYZE explainable_statement;
```

where the statement is any statement for which one can run EXPLAIN.

Command Output

Consider an example:

```
ANALYZE SELECT * FROM tbl1
WHERE key1
  BETWEEN 10 AND 200 AND
  col1 LIKE 'foo%'\G
```

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: tbl1
         type: range
possible_keys: key1
          key: key1
      key_len: 5
          ref: NULL
         rows: 181
       r_rows: 181
     filtered: 100.00
   r_filtered: 10.50
        Extra: Using index condition; Using where
```

Compared to `EXPLAIN`, `ANALYZE` produces two extra columns:

- `r_rows` is an observation-based counterpart of the rows column. It shows how many rows were actually read from the table.
- `r_filtered` is an observation-based counterpart of the filtered column. It shows which fraction of rows was left after applying the WHERE condition.

> These `r_rows` and `r_filtered` columns are also included in the EXPLAIN output written to the slow query log in MariaDB 10.1.0 and later. This explains why, even though they are not displayed by a standard EXPLAIN statement, some columns may appear in slow query log data.

Interpreting the Output

Joins

Let's consider a more complicated example.

```
ANALYZE SELECT *
FROM orders, customer
WHERE
  customer.c_custkey=orders.o_custkey AND
  customer.c_acctbal < 0 AND
  orders.o_totalprice > 200*1000
```

```
+----+-------------+----------+------+---------------+-------------+---------+--------------------+------
--+--------+----------+------------+-------------+
| id | select_type | table    | type | possible_keys | key         | key_len | ref                | rows
| r_rows | filtered | r_filtered | Extra       |
+----+-------------+----------+------+---------------+-------------+---------+--------------------+------
--+--------+----------+------------+-------------+
|  1 | SIMPLE      | customer | ALL  | PRIMARY,...   | NULL        | NULL    | NULL               |
149095 | 150000 |    18.08 |       9.13 | Using where |
|  1 | SIMPLE      | orders   | ref  | i_o_custkey   | i_o_custkey | 5       | customer.c_custkey |
7 |     10 |   100.00 |      30.03 | Using where |
+----+-------------+----------+------+---------------+-------------+---------+--------------------+------
--+--------+----------+------------+-------------+
```

Here, one can see that

- For table customer, customer.rows=149095, customer.r_rows=150000. The estimate for number of rows we will read was fairly precise
- customer.filtered=18.08, customer.r_filtered=9.13. The optimizer somewhat overestimated the number of records that will match selectivity of condition attached to `customer` table (in general, when you have a full scan and r_filtered is less than 15%, it's time to consider adding an appropriate index).
- For table orders, orders.rows=7, orders.r_rows=10. This means that on average, there are 7 orders for a given c_custkey, but in our case there were 10, which is close to the expectation (when this number is consistently far from the expectation, it may be time to run ANALYZE TABLE, or even edit the table statistics manually to get better query plans).
- orders.filtered=100, orders.r_filtered=30.03. The optimizer didn't have any way to estimate which fraction of records will be left after it checks the condition that is attached to table orders (it's orders.o_totalprice > 200*1000). So, it used 100%. In reality, it is 30%. 30% is typically not selective enough to warrant adding new indexes. For joins with many tables, it might be worth to collect and use column statistics for columns in question, this may help the optimizer to pick a better query plan.

Meaning of NULL in r_rows and r_filtered

Let's modify the previous example slightly

```
ANALYZE SELECT *
FROM orders, customer
WHERE
  customer.c_custkey=orders.o_custkey AND
  customer.c_acctbal < -0 AND
  customer.c_comment LIKE '%foo%' AND
  orders.o_totalprice > 200*1000;
```

```
+----+-------------+----------+------+---------------+-------------+---------+--------------------+------
--+--------+----------+------------+-------------+
| id | select_type | table    | type | possible_keys | key         | key_len | ref                | rows
| r_rows | filtered | r_filtered | Extra       |
+----+-------------+----------+------+---------------+-------------+---------+--------------------+------
--+--------+----------+------------+-------------+
|  1 | SIMPLE      | customer | ALL  | PRIMARY,...   | NULL        | NULL    | NULL               |
149095 | 150000 |    18.08 |       0.00 | Using where |
|  1 | SIMPLE      | orders   | ref  | i_o_custkey   | i_o_custkey | 5       | customer.c_custkey |
7 |   NULL |   100.00 |       NULL | Using where |
+----+-------------+----------+------+---------------+-------------+---------+--------------------+------
--+--------+----------+------------+-------------+
```

The output of **orders.r_rows=NULL** and **orders.r_filtered=NULL** shows that the table `orders` was never scanned. Indeed, we can also see customer.r_filtered=0.00. This shows that a part of WHERE attached to table `customer` was never satisfied (or, satisfied in less than 0.01% of cases).

ANALYZE FORMAT=JSON

ANALYZE FORMAT=JSON produces JSON output. It produces much more information than tabular `ANALYZE`.

Notes

- `ANALYZE UPDATE` or `ANALYZE DELETE` will actually make updates/deletes (`ANALYZE SELECT` will perform the select operation and then discard the resultset).
- PostgreSQL has a similar command, `EXPLAIN ANALYZE`.
- The EXPLAIN in the slow query log feature allows MariaDB to have `ANALYZE` output of slow queries printed into the slow query log (see MDEV-6388).

See Also

- ANALYZE FORMAT=JSON
- SHOW ANALYZE
- ANALYZE TABLE
- JIRA task for ANALYZE statement, MDEV-406

This page is licensed: CC BY-SA / Gnu FDL

EXPLAIN ANALYZE

> **Warning:**
>
> Outdated syntax

The syntax for the `EXPLAIN ANALYZE` feature was changed to `ANALYZE statement`, available since MariaDB 10.1.0. See ANALYZE statement.

This page is licensed: CC BY-SA / Gnu FDL

EXPLAIN FORMAT=JSON

Synopsis

`EXPLAIN FORMAT=JSON` is a variant of EXPLAIN command that produces output in JSON form. The output always has one row which has only one column titled "`JSON`". The contents are a JSON representation of the query plan:

```
EXPLAIN FORMAT=JSON SELECT * FROM t1 WHERE col1=1\G
```

```
*************************** 1. row ***************************
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "table": {
      "table_name": "t1",
      "access_type": "ALL",
      "rows": 1000,
      "filtered": 100,
      "attached_condition": "(t1.col1 = 1)"
    }
  }
}
```

Output is different from MySQL

The output of MariaDB's `EXPLAIN FORMAT=JSON` is different from `EXPLAIN FORMAT=JSON` in MySQL. The reasons for that are:

- MySQL's output has deficiencies.
- The output of MySQL's `EXPLAIN FORMAT=JSON` is not defined. Even MySQL Workbench has trouble parsing it (see this blog post).
- MariaDB has query optimizations that MySQL does not have. This means that MariaDB generates query plans that MySQL does not generate.

Output Format

TODO: MariaDB's output format description.

See Also

- ANALYZE FORMAT=JSON produces output like `EXPLAIN FORMAT=JSON`, but amended with the data from query execution.

This page is licensed: CC BY-SA / Gnu FDL

EXPLAIN

Syntax

```
EXPLAIN tbl_name [col_name | wildcard]
```

or

```
EXPLAIN [EXTENDED | PARTITIONS | FORMAT=JSON]
  {SELECT select_options | UPDATE update_options | DELETE delete_options}
```

or

```
EXPLAIN [FORMAT=JSON] FOR CONNECTION <connection_id>
```

Overview

The `EXPLAIN` statement can be used either as a synonym for DESCRIBE, or as a way to obtain information about how MariaDB executes a `SELECT`, `UPDATE` or `DELETE` statement:

- `'EXPLAIN tbl_name'` is synonymous with DESCRIBE or SHOW COLUMNS … FROM *mytable*.
- When you precede a `SELECT`, `UPDATE` or a `DELETE` statement with the keyword `EXPLAIN`, MariaDB displays information from the optimizer about the query execution plan. That is, MariaDB explains how it would process the `SELECT`, `UPDATE` or `DELETE`, including information about how tables are joined and in which order. `EXPLAIN EXTENDED` can be used to provide additional information.
- `EXPLAIN PARTITIONS` is useful only when examining queries involving partitioned tables. For details, see Partition pruning and selection.
- The ANALYZE statement is similar to `EXPLAIN`, but it runs the query rather than only estimating the execution path. Hence, the `ANALYZE` statement provides both estimated and real execution statistics. This includes the `r_rows` and `r_filtered` columns, which reflect the actual number of rows evaluated and filtered during query execution.
- `EXPLAIN` output can be printed in the slow query log. See EXPLAIN in the Slow Query Log for details.
- `EXPLAIN FOR CONNECTION` is an alias for `SHOW EXPLAIN FOR`.

SHOW EXPLAIN shows the output of a running statement. In some cases, its output can be closer to reality than `EXPLAIN`.

The ANALYZE statement runs a statement and returns information about its execution plan. It also shows additional columns, to check how much the optimizer's estimation about filtering and found rows are close to reality.

There is an online EXPLAIN Analyzer that you can use to share `EXPLAIN` and `EXPLAIN EXTENDED` output with others.

`EXPLAIN` can acquire metadata locks in the same way that `SELECT` does, as it needs to know table metadata and, sometimes, data as well.

Columns in EXPLAIN ... SELECT

Column name	Description
id	Sequence number that shows in which order tables are joined.
select_type	What kind of `SELECT` the table comes from.
table	Alias name of table. Materialized temporary tables for sub queries are named `<subquery#>`
type	How rows are found from the table (join type).
possible_keys	keys in table that could be used to find rows in the table
key	The name of the key that is used to retrieve rows. `NULL` means no key was used.
key_len	How many bytes of the key that was used (shows if we are using only parts of the multi-column key).
ref	The reference that is used as the key value.
rows	An estimate of how many rows we will find in the table for each key lookup.
Extra	Extra information about this join.

Here are descriptions of the values for some of the more complex columns in `EXPLAIN ... SELECT`:

Select_type Column

The `select_type` column can have the following values:

Value	Description	Comment
DEPENDENT SUBQUERY	The `SUBQUERY` is `DEPENDENT`.	
DEPENDENT UNION	The `UNION` is `DEPENDENT`.	
DERIVED	The `SELECT` is `DERIVED` from the `PRIMARY`.	
MATERIALIZED	The `SUBQUERY` is MATERIALIZED.	Materialized tables will be populated at first access and will be accessed by the primary key (= one key lookup). Number of rows in `EXPLAIN` shows the cost of populating the table
PRIMARY	The `SELECT` is in the outermost query, but there is also a `SUBQUERY` within it.	
SIMPLE	It is a simple `SELECT` query without any `SUBQUERY` or `UNION`.	
SUBQUERY	The `SELECT` is a `SUBQUERY` of the `PRIMARY`.	
UNCACHEABLE SUBQUERY	The `SUBQUERY` is `UNCACHEABLE`.	
UNCACHEABLE UNION	The `UNION` is `UNCACHEABLE`.	

Value	Description	Comment
UNION	The `SELECT` is a `UNION` of the `PRIMARY`.	
UNION RESULT	The result of the `UNION`.	
LATERAL DERIVED	The `SELECT` uses a Lateral Derived optimization	

Type Column

This column contains information on how the table is accessed.

Value	Description
ALL	A full table scan is done for the table (all rows are read). This is bad if the table is large and the table is joined against a previous table! This happens when the optimizer could not find any usable index to access rows.
const	There is only one possibly matching row in the table. The row is read before the optimization phase and all columns in the table are treated as constants.
eq_ref	A unique index is used to find the rows. This is the best possible plan to find the row.
filter	A second index is being used with the Rowid Filtering Optimization.
fulltext	A fulltext index is used to access the rows.
index_merge	A 'range' access is done for several index and the found rows are merged. The key column shows which keys are used.
index_subquery	This is similar as ref, but used for sub queries that are transformed to key lookups.
index	A full scan over the used index. Better than `ALL` but still bad if index is large and the table is joined against a previous table.
range	The table will be accessed with a key over one or more value ranges.
ref_or_null	Like 'ref' but in addition another search for the 'null' value is done if the first value was not found. This happens usually with sub queries.
ref	A non unique index or prefix of an unique index is used to find the rows. Good if the prefix doesn't match many rows.
system	The table has 0 or 1 rows.
unique_subquery	This is similar as eq_ref, but used for sub queries that are transformed to key lookups

Extra Column

This column consists of one or more of the following values, separated by ';'

Note that some of these values are detected after the optimization phase.

The optimization phase can do the following changes to the `WHERE` clause:

- Add the expressions from the `ON` and `USING` clauses to the `WHERE` clause.
- Constant propagation: If there is `column=constant`, replace all column instances with this constant.
- Replace all columns from '`const`' tables with their values.
- Remove the used key columns from the `WHERE` (as this will be tested as part of the key lookup).
- Remove impossible constant sub expressions. For example `WHERE '(a=1 and a=2) OR b=1'` becomes `'b=1'`.
- Replace columns with other columns that has identical values: Example: `WHERE a=b` and `a=c` may be treated as `'WHERE a=b and a=c and b=c'`.
- Add extra conditions to detect impossible row conditions earlier. This happens mainly with `OUTER JOIN` where we in some cases add detection of `NULL` values in the `WHERE` (Part of '`Not exists`' optimization). This can

cause an unexpected '`Using where`' in the Extra column.

- For each table level we remove expressions that have already been tested when we read the previous row. Example: When joining tables `t1` with `t2` using the following `WHERE 't1.a=1 and t1.a=t2.b'`, we don't have to test `'t1.a=1'` when checking rows in `t2` as we already know that this expression is true.

Value	Description
const row not found	The table was a system table (a table with should exactly one row), but no row was found.
Distinct	If distinct optimization (remove duplicates) was used. This is marked only for the last table in the `SELECT`.
Full scan on NULL key	The table is a part of the sub query and if the value that is used to match the sub query will be `NULL`, we will do a full table scan.
Impossible HAVING	The used `HAVING` clause is always false so the `SELECT` will return no rows.
Impossible WHERE noticed after reading const tables.	The used `WHERE` clause is always false so the `SELECT` will return no rows. This case was detected after we had read all 'const' tables and used the column values as constant in the `WHERE` clause. For example: `WHERE const_column=5 and const_column` had a value of 4.
Impossible WHERE	The used `WHERE` clause is always false so the `SELECT` will return no rows. For example: `WHERE 1=2`
No matching min/max row	During early optimization of `MIN()` / `MAX()` values it was detected that no row could match the `WHERE` clause. The `MIN()` / `MAX()` function will return `NULL`.
no matching row in const table	The table was a const table (a table with only one possible matching row), but no row was found.
No tables used	The `SELECT` was a sub query that did not use any tables. For example a there was no `FROM` clause or a `FROM DUAL` clause.
Not exists	Stop searching after more row if we find one single matching row. This optimization is used with `LEFT JOIN` where one is explicitly searching for rows that doesn't exists in the `LEFT JOIN TABLE`. Example: `SELECT * FROM t1 LEFT JOIN t2 on (...) WHERE t2.not_null_column IS NULL`. As `t2.not_null_column` can only be `NULL` if there was no matching row for on condition, we can stop searching if we find a single matching row.
Open_frm_only	For information_schema tables. Only the frm (table definition file was opened) was opened for each matching row.
Open_full_table	For information_schema tables. A full table open for each matching row is done to retrieve the requested information. (Slow)
Open_trigger_only	For information_schema tables. Only the trigger file definition was opened for each matching row.
Range checked for each record (index map: …)	This only happens when there was no good default index to use but there may some index that could be used when we can treat all columns from previous table as constants. For each row combination the optimizer will decide which index to use (if any) to fetch a row from this table. This is not fast, but faster than a full table scan that is the only other choice. The index map is a bitmask that shows which index are considered for each row condition.
Scanned 0/1/all databases	For information_schema tables. Shows how many times we had to do a directory scan.
Select tables optimized away	All tables in the join was optimized away. This happens when we are only using `COUNT(*)`, `MIN()` and `MAX()` functions in the `SELECT` and we where able to replace all of these with constants.
Skip_open_table	For information_schema tables. The queried table didn't need to be opened.
unique row not found	The table was detected to be a const table (a table with only one possible matching row) during the early optimization phase, but no row was found.

Value	Description
Using filesort	Filesort is needed to resolve the query. This means an extra phase where we first collect all columns to sort, sort them with a disk based merge sort and then use the sorted set to retrieve the rows in sorted order. If the column set is small, we store all the columns in the sort file to not have to go to the database to retrieve them again.
Using index	Only the index is used to retrieve the needed information from the table. There is no need to perform an extra seek to retrieve the actual record.
Using index condition	Like 'Using where' but the where condition is pushed down to the table engine for internal optimization at the index level.
Using index condition(BKA)	Like 'Using index condition' but in addition we use batch key access to retrieve rows.
Using index for group-by	The index is being used to resolve a `GROUP BY` or `DISTINCT` query. The rows are not read. This is very efficient if the table has a lot of identical index entries as duplicates are quickly jumped over.
Using intersect(…)	For index_merge joins. Shows which index are part of the intersect.
Using join buffer	We store previous row combinations in a row buffer to be able to match each row against all of the rows combinations in the join buffer at one go.
Using sort_union(…)	For index_merge joins. Shows which index are part of the union.
Using temporary	A temporary table is created to hold the result. This typically happens if you are using `GROUP BY`, `DISTINCT` or `ORDER BY`.
Using where	A `WHERE` expression (in additional to the possible key lookup) is used to check if the row should be accepted. If you don't have 'Using where' together with a join type of `ALL`, you are probably doing something wrong!
Using where; FirstMatch(*mytable*)	When one matching record combination has been found, shortcut the execution and jump back to the *mytable* table. See this page for details on the `FirstMatch` Strategy.
Using where with pushed condition	Like 'Using where' but the where condition is pushed down to the table engine for internal optimization at the row level.
Using buffer	The `UPDATE` statement will first buffer the rows, and then run the updates, rather than do updates on the fly. See Using Buffer UPDATE Algorithm for a detailed explanation.

EXPLAIN EXTENDED

The `EXTENDED` keyword adds another column, *filtered*, to the output. This is a percentage estimate of the table rows that will be filtered by the condition.

An `EXPLAIN EXTENDED` will always throw a warning, as it adds extra *Message* information to a subsequent SHOW WARNINGS statement. This includes what the `SELECT` query would look like after optimizing and rewriting rules are applied and how the optimizer qualifies columns and tables.

Examples

As synonym for `DESCRIBE` or `SHOW COLUMNS FROM`:

```
DESCRIBE city;
+------------+----------+------+-----+---------+----------------+
| Field      | Type     | Null | Key | Default | Extra          |
+------------+----------+------+-----+---------+----------------+
| Id         | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name       | char(35) | YES  |     | NULL    |                |
| Country    | char(3)  | NO   | UNI |         |                |
| District   | char(20) | YES  | MUL |         |                |
| Population | int(11)  | YES  |     | NULL    |                |
+------------+----------+------+-----+---------+----------------+
```

A simple set of examples to see how `EXPLAIN` can identify poor index usage:

```
CREATE TABLE IF NOT EXISTS `employees_example` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `first_name` varchar(30) NOT NULL,
  `last_name` varchar(40) NOT NULL,
  `position` varchar(25) NOT NULL,
  `home_address` varchar(50) NOT NULL,
  `home_phone` varchar(12) NOT NULL,
  `employee_code` varchar(25) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `employee_code` (`employee_code`),
  KEY `first_name` (`first_name`,`last_name`)
) ENGINE=Aria;

INSERT INTO `employees_example`
  (`first_name`, `last_name`, `position`, `home_address`, `home_phone`, `employee_code`)
  VALUES
  ('Mustapha', 'Mond', 'Chief Executive Officer', '692 Promiscuous Plaza', '326-555-3492', 'MM1'),
  ('Henry', 'Foster', 'Store Manager', '314 Savage Circle', '326-555-3847', 'HF1'),
  ('Bernard', 'Marx', 'Cashier', '1240 Ambient Avenue', '326-555-8456', 'BM1'),
  ('Lenina', 'Crowne', 'Cashier', '281 Bumblepuppy Boulevard', '328-555-2349', 'LC1'),
  ('Fanny', 'Crowne', 'Restocker', '1023 Bokanovsky Lane', '326-555-6329', 'FC1'),
  ('Helmholtz', 'Watson', 'Janitor', '944 Soma Court', '329-555-2478', 'HW1');

SHOW INDEXES FROM employees_example;
+------------------+------------+---------------+--------------+---------------+-----------+------------
-+----------+--------+------+------------+---------+---------------+
| Table            | Non_unique | Key_name      | Seq_in_index | Column_name   | Collation | Cardinality
| Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+------------------+------------+---------------+--------------+---------------+-----------+------------
-+----------+--------+------+------------+---------+---------------+
| employees_example |          0 | PRIMARY       |            1 | id            | A         |           7
|     NULL | NULL   |      | BTREE      |         |               |
| employees_example |          0 | employee_code |            1 | employee_code | A         |           7
|     NULL | NULL   |      | BTREE      |         |               |
| employees_example |          1 | first_name    |            1 | first_name    | A         |        NULL
|     NULL | NULL   |      | BTREE      |         |               |
| employees_example |          1 | first_name    |            2 | last_name     | A         |        NULL
|     NULL | NULL   |      | BTREE      |         |               |
+------------------+------------+---------------+--------------+---------------+-----------+------------
-+----------+--------+------+------------+---------+---------------+
```

`SELECT` on a primary key:

```
EXPLAIN SELECT * FROM employees_example WHERE id=1;
+------+-------------+------------------+-------+---------------+---------+---------+-------+------+----
---+
| id   | select_type | table            | type  | possible_keys | key     | key_len | ref   | rows |
Extra |
+------+-------------+------------------+-------+---------------+---------+---------+-------+------+----
---+
|    1 | SIMPLE      | employees_example | const | PRIMARY       | PRIMARY | 4       | const |    1 |
|
+------+-------------+------------------+-------+---------------+---------+---------+-------+------+----
---+
```

The type is *const*, which means that only one possible result could be returned.
Now, returning the same record but searching by their phone number:

```
EXPLAIN SELECT * FROM employees_example WHERE home_phone='326-555-3492';
+------+-------------+------------------+------+---------------+------+---------+------+------+---------
----+
| id   | select_type | table            | type | possible_keys | key  | key_len | ref  | rows | Extra
|
+------+-------------+------------------+------+---------------+------+---------+------+------+---------
----+
|    1 | SIMPLE      | employees_example | ALL  | NULL          | NULL | NULL    | NULL |    6 | Using
where |
+------+-------------+------------------+------+---------------+------+---------+------+------+---------
----+
```

Here, the type is *All*, which means no index could be used. Looking at the rows count, a full table scan (all six rows) had to be performed in order to retrieve the record. If it's a requirement to search by phone number, an index will have to be created.

SHOW EXPLAIN example:

```
SHOW EXPLAIN FOR 1;
+------+-------------+-------+-------+---------------+------+---------+------+---------+-------------+
| id   | select_type | table | type  | possible_keys | key  | key_len | ref  | rows    | Extra       |
+------+-------------+-------+-------+---------------+------+---------+------+---------+-------------+
|    1 | SIMPLE      | tbl   | index | NULL          | a    | 5       | NULL | 1000107 | Using index |
+------+-------------+-------+-------+---------------+------+---------+------+---------+-------------+
1 row in set, 1 warning (0.00 sec)
```

Example of `ref_or_null` Optimization

```
SELECT * FROM table_name
  WHERE key_column=expr OR key_column IS NULL;
```

`ref_or_null` is something that often happens when you use subqueries with `NOT IN` as then one has to do an extra check for `NULL` values if the first value didn't have a matching row.

See Also

- SHOW EXPLAIN
- Ignored Indexes

This page is licensed: GPLv2, originally from fill_help_tables.sql

Using Buffer UPDATE Algorithm

This article explains the UPDATE statement's *Using Buffer* algorithm.

Consider the following table and query:

Name	Salary
Babatunde	1000
Jolana	1050
Pankaja	1300

```
UPDATE employees SET salary = salary+100 WHERE salary < 2000;
```

Suppose the *employees* table has an index on the *salary* column, and the optimizer decides to use a range scan on that index.

The optimizer starts a range scan on the *salary* index. We find the first record *Babatunde, 1000*. If we do an on-the-fly update, we immediately instruct the storage engine to change this record to be *Babatunde, 1000+100=1100*.

Then we proceed to search for the next record, and find *Jolana, 1050*. We instruct the storage engine to update it to be *Jolana, 1050+100=1150*.

Then we proceed to search for the next record … and what happens next depends on the storage engine. In some storage engines, data changes are visible immediately, so we will find the *Babatunde, 1100* record that we wrote at the first step, modifying it again, giving Babatunde an undeserved raise. Then we will see Babatunde again and again, looping continually.

In order to prevent such situations, the optimizer checks whether the UPDATE statement is going to change key values for the keys it is using. In that case, it will use a different algorithm:

1. Scan everyone with “salary<2000”, remembering the rowids of the rows in a buffer.
2. Read the buffer and apply the updates.

This way, each row will be updated only once.

The `Using buffer` EXPLAIN output indicates that the buffer as described above will be used.

This page is licensed: CC BY-SA / Gnu FDL

BACKUP Statements

See BACKUP LOCK.

Protect table files during backups. This command blocks DDL operations like ALTER TABLE while allowing read/write activity, ensuring file consistency for backup tools.

See BACKUP STAGE.

Control backup phases for external tools. Learn how to cycle through stages like START, BLOCK_DDL, and BLOCK_COMMIT to perform consistent backups with minimal locking.

See Storage Snapshots and BACKUP STAGE Statements.

Combine database commands with storage-level snapshots. Learn the correct sequence of BACKUP STAGE commands to freeze writes safely while taking a disk snapshot.

This page is licensed: CC BY-SA / Gnu FDL

BACKUP LOCK

`BACKUP LOCK` blocks a table from DDL statements. This is mainly intended to be used by tools like mariadb-backup that need to ensure there are no DDL[38] statements on a table while the table files are opened. For example, for an Aria table that stores data in 3 files with extensions `.frm`, `.MAI` and `.MAD`. Normal read/write operations can continue as normal.

Syntax

To lock a table:

```
BACKUP LOCK table_name
```

To unlock a table:

```
BACKUP UNLOCK
```

Usage in a Backup Tool

```
BACKUP LOCK [database.]table_name;
 - Open all files related to a table (for example, t.frm, t.MAI and t.MYD)
BACKUP UNLOCK;
- Copy data
- Close files
```

This ensures that all files are from the same generation, that is created at the same time by the MariaDB server. This works, because the open files will point to the original table files which will not be affected if there is any ALTER TABLE while copying the files.

Privileges

> **CURRENT**
>
> BACKUP LOCK requires the database LOCK TABLES privileges.
>
> **< 11.4.1 / 11.3.2 / 11.2.3 / 11.1.4 / 11.0.5**
>
> BACKUP LOCK requires the RELOAD privilege.
>
> **< 10.11.7 / 10.6.17 / 10.5.24**
>
> BACKUP LOCK requires the RELOAD privilege.

Notes

- The idea is that the `BACKUP LOCK` should be held for as short a time as possible by the backup tool. The time to take an uncontested lock is very short! One can easily do 50,000 locks/unlocks per second on low end hardware.
- One should use different connections for BACKUP STAGE commands and `BACKUP LOCK`.

Implementation

- Internally, BACKUP LOCK is implemented by taking an `MDLSHARED_HIGH_PRIO` MDL lock on the table object, which protects the table from any DDL operations.

See Also

- BACKUP STAGE
- MDEV-17309 - BACKUP LOCK: DDL locking of tables during backup

This page is licensed: CC BY-SA / Gnu FDL

BACKUP STAGE

The `BACKUP STAGE` commands are a set of commands to make it possible to make an efficient external backup tool.

Syntax

```
BACKUP STAGE [START | FLUSH | BLOCK_DDL | BLOCK_COMMIT | END ]
```

In the following text, a transactional table means InnoDB, or specifically an InnoDB-like engine with redo log that can lock redo purges and can be copied without locks by an outside process.

Goals with BACKUP STAGE Commands

- To be able to do a majority of the backup with the minimum possible server locks. Especially for transactional tables (InnoDB, MyRocks etc) there is only need for a very short block of new commits while copying statistics and log tables.
- DDL are only needed to be blocked for a very short duration of the backup while mariadb-backup is copying the tables affected by DDL during the initial part of the backup.
- Most non transactional tables (those that are not in use) will be copied during `BACKUP STAGE START`. The exceptions are system statistic and log tables that are not blocked during the backup until `BLOCK_COMMIT`.
- Should work efficiently with backup tools that use disk snapshots.
- Should work as efficiently as possible for all table types that store data on the local disks.
- As little copying as possible under higher level stages/locks. For example, .frm (dictionary) and .trn (trigger) files should be copying while copying the table data.

`BACKUP STAGE` Commands

`BACKUP STAGE START`

The `START` stage is designed for the following tasks:

- Blocks purge of redo files for storage engines that needs this (Aria).
- Start logging of DDL commands into *`datadir`*`/ddl.log`. This may take a short time as the command has to wait until there are no active DDL commands.

`BACKUP STAGE FLUSH`

The `FLUSH` stage is designed for the following tasks:

- `FLUSH` all changes for inactive non-transactional tables, except for statistics and log tables.
- Close all tables that are not in use, to ensure they are marked as closed for the backup.
- `BLOCK` all new write locks for all non transactional tables (except statistics and log tables). The command will not wait for tables that are in use by read-only transactions.

DDLs don't have to be blocked at this stage as they can't cause the table to be in an inconsistent state. This is true also for non-transactional tables.

`BACKUP STAGE BLOCK_DDL`

The `BLOCK_DDL` stage is designed for the following tasks:

- Wait for all statements using write locked non-transactional tables to end.
- Blocks CREATE TABLE, DROP TABLE, TRUNCATE TABLE, and RENAME TABLE.
- Blocks also start off a new ALTER TABLE and the final rename phase of ALTER TABLE. Running ALTER TABLES are not blocked.

`BACKUP STAGE BLOCK_COMMIT`

The `BLOCK_COMMIT` stage is designed for the following tasks:

- Lock the binary log and commit/rollback to ensure that no changes are committed to any tables. If there are active commits or data to be copied to the binary log this will be allowed to finish. Active transactions will not affect `BLOCK_COMMIT`.
- This doesn't lock temporary tables that are not used by replication. However these will be blocked when it's time to write to the binary log.
- Lock system log tables and statistics tables, flush them and mark them closed.

When the `BLOCK_COMMIT`'s stages return, this is the 'backup time'. Everything committed will be in the backup and everything not committed will roll back.

Transactional engines will continue to do changes to the redo log during the `BLOCK COMMIT` stage, but this is not important as all of these will roll back later as the changes will not be committed.

`BACKUP STAGE END`

The `END` stage is designed for the following tasks:

- End DDL logging
- Free resources

Using `BACKUP STAGE` Commands with Backup Tools

Using `BACKUP STAGE` Commands with mariadb-backup

> **CURRENT**
>
> The `BACKUP STAGE` statements are a set of statements that make it possible to make an efficient external backup tool.
>
> **< 11.2.4 / 11.1.5 / 11.0.6**
>
> The `BACKUP STAGE` statements are a set of statements that make it possible to make an efficient external backup tool. How mariadb-backup uses these statements depends on which version you are using. It depends on whether you are using the version that is bundled with MariaDB Community Server or the version that is bundled with MariaDB Enterprise Server.
>
> **< 10.11.8**
>
> The `BACKUP STAGE` statements are a set of statements that make it possible to make an efficient external backup tool. How mariadb-backup uses these statements depends on which version you are using. It depends on whether you are using the version that is bundled with MariaDB Community Server or the version that is bundled with MariaDB Enterprise Server.

See mariadb-backup and BACKUP STAGE Commands for some examples on how mariadb-backup uses these commands.

Using `BACKUP STAGE` Commands with Storage Snapshots

The `BACKUP STAGE` commands are a set of commands to make it possible to make an efficient external backup tool. These commands could even be used by tools that perform backups by taking a snapshot of a file system, SAN, or some other kind of storage device. See Storage Snapshots and BACKUP STAGE Commands for some examples on how to use each `BACKUP STAGE` command in an efficient way.

Privileges

`BACKUP STAGE` requires the RELOAD privilege.

Notes

- Only one connection can run `BACKUP STAGE START`. If a second connection tries, it will wait until the first one has executed `BACKUP STAGE END`.
- If the user skips a `BACKUP STAGE`, then all intermediate backup stages will automatically be run. This will allow us to add new stages within the `BACKUP STAGE` hierarchy in the future with even more precise locks without causing problems for tools using an earlier version of the `BACKUP STAGE` implementation.

- One can use the max_statement_time or lock_wait_timeout system variables to ensure that a `BACKUP STAGE` command doesn't block the server too long.
- DDL logging is available from MariaDB 10.11.8, MariaDB 11.0.6, MariaDB 11.1.5 and MariaDB 11.2.4, and in MariaDB Enterprise Server.
- A disconnect automatically releases backup stages.
- There is no easy way to see what the current stage is.

See Also

- BACKUP LOCK Locking a table from DDL.
- MDEV-5336. Implement `BACKUP STAGE` for safe external backups.

This page is licensed: CC BY-SA / Gnu FDL

Storage Snapshots and BACKUP STAGE Statements

The BACKUP STAGE statements are a set of statements to make it possible to make an efficient external backup tool. These commands could even be used by tools that perform backups by taking a snapshot of a file system, SAN, or some other kind of storage device.

Generic Backup Process with Storage Snapshots

A tool that backs up MariaDB by taking a snapshot of a file system, SAN, or some other kind of storage device could use each `BACKUP STAGE` command in the following way:

- First, execute the following:

```
BACKUP STAGE START
BACKUP STAGE BLOCK_COMMIT
```

- Then, take the snapshot.
- Then, execute the following:

```
BACKUP STAGE END
```

The above ensures that all non-transactional tables are properly flushed to disk before the snapshot is done. Using `BACKUP STAGE` commands is also more efficient than using the FLUSH TABLES WITH READ LOCK command as the above set of commands will not block or be blocked by write operations to transactional tables.

Note that when the backup is completed, one should delete all files with the "#sql" prefix, as these are files used by concurrent running `ALTER TABLE`. Note that InnoDB will on server restart automatically delete any tables with the "#sql" prefix.

This page is licensed: CC BY-SA / Gnu FDL

FLUSH Statements

See FLUSH.

Complete FLUSH statement reference: NO_WRITE_TO_BINLOG/LOCAL syntax, FLUSH TABLES WITH READ LOCK/FOR EXPORT, FLUSH STATUS, and SSL/TLS certificate reload.

See FLUSH QUERY CACHE.

Defragment the query cache to optimize memory usage. This command reorganizes the cache to eliminate fragmentation without removing existing cached queries.

See FLUSH TABLES FOR EXPORT.

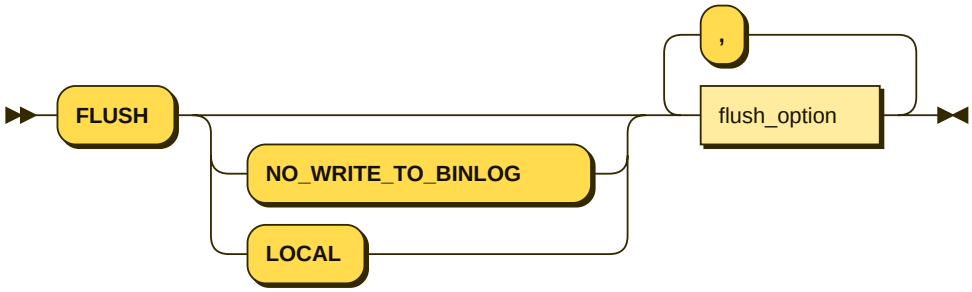
Prepare individual tables for binary backup. This command flushes changes to disk and locks tables, allowing safe copying of .ibd files while the server runs.

This page is licensed: CC BY-SA / Gnu FDL

FLUSH

Syntax

```
FLUSH [NO_WRITE_TO_BINLOG | LOCAL]
    flush_option [, flush_option] ...
```



Railroad diagram of FLUSH — equivalent to the BNF above

or when flushing tables:

```
FLUSH [NO_WRITE_TO_BINLOG | LOCAL] TABLES [table_list]  [table_flush_option]
```

where `table_list` is a list of tables separated by `,` (comma).

Description

The `FLUSH` statement clears or reloads various internal caches used by MariaDB. To execute `FLUSH`, you must have the `RELOAD` privilege. See GRANT.

The `RESET` statement is similar to `FLUSH`. See RESET.

You cannot issue a FLUSH statement from within a stored function or a trigger. Doing so within a stored procedure is permitted, as long as it is not called by a stored function or trigger. See Stored Routine Limitations, Stored Function Limitations and Trigger Limitations.

If a listed table is a view, an error like the following will be produced:

```
ERROR 1347 (HY000): 'test.v' is not BASE TABLE
```

By default, `FLUSH` statements are written to the binary log and will be replicated. The `NO_WRITE_TO_BINLOG` keyword (`LOCAL` is an alias) ensures that the statement is not written to the binary log. The different flush options are:

Option	Description
CHANGED_PAGE_BITMAPS	XtraDB only. Internal command used for backup purposes. See the Information Schema CHANGED_PAGE_BITMAPS Table.
CLIENT_STATISTICS	Reset client statistics (see SHOW CLIENT_STATISTICS).
DES_KEY_FILE	Reloads the DES key file (specified with the –des-key-file startup option). **Removed in MariaDB 13.0**, together with the `DES_ENCRYPT()` / `DES_DECRYPT()` functions.

Option	Description
HOSTS	Flush the hostname cache (used for converting ip to host names and for unblocking blocked hosts. See max_connect_errors and performance_schema.host_cache
INDEX_STATISTICS	Reset index statistics (see SHOW INDEX_STATISTICS).
ERROR	Closes and reopens the error log file to which the server is writing.
[ERROR | ENGINE | GENERAL | SLOW | RELAY] LOGS	Close and reopen the specified log type, or all log types if none are specified. `FLUSH RELAY LOGS [connection-name]` can be used to flush the relay logs for a specific connection. Only one connection can be specified per `FLUSH` command. See multi-source replication. `FLUSH ENGINE LOGS` deletes all unneeded Aria redo logs. `FLUSH BINARY LOGS DELETE_DOMAIN_ID=(list-of-domains)` can be used to discard obsolete GTID domains from the server's binary log state. In order for this to be successful, no event group from the listed GTID domains can be present in existing binary log files. If some still exist, then they must be purged prior to executing this statement. If the statement completes successfully, it also rotates the binary log.
BINARY LOGS	`FLUSH BINARY LOGS` rotates the current binary log.
BINARY LOGS DELETE_DOMAIN_ID *=list-of-domains*	`FLUSH BINARY LOGS DELETE_DOMAIN_ID` can be used to discard obsolete GTID domains from the server's binary log state. In order for this to be successful, no event group from the listed GTID domains can be present in existing binary log files. If some still exist, then they must be purged prior to executing this command. If the command completes successfully, then it also rotates the binary log.
MASTER	Deprecated option, use RESET MASTER instead.
PRIVILEGES	Reload all privileges from the privilege tables in the mysql database. If the server is started with –skip-grant-table option, this will activate the privilege tables again.
QUERY CACHE	Defragment the query cache to better utilize its memory. If you want to reset the query cache, you can do it with RESET QUERY CACHE.
QUERY_RESPONSE_TIME	See the QUERY_RESPONSE_TIME plugin.
QUERY_RESPONSE_TIME_READ	See the QUERY_RESPONSE_TIME plugin. From MariaDB 11.5.
QUERY_RESPONSE_TIME_READ_WRITE	See the QUERY_RESPONSE_TIME plugin. From MariaDB 11.5.
QUERY_RESPONSE_TIME_WRITE	See the QUERY_RESPONSE_TIME plugin. From MariaDB 11.5.
SLAVE	Deprecated option, use RESET REPLICA or RESET SLAVE instead.
SSL	Used to dynamically reinitialize the server's TLS context by reloading the files defined by several TLS system variables. See FLUSH SSL for more information.
[GLOBAL | SESSION] STATUS	Resets all server status variables that can be reset to 0. Not all global status variables support this, so not all global values are reset. From MariaDB 11.5, `GLOBAL` or `SESSION` can be specified. See FLUSH STATUS for more information.
TABLE[S]	Close tables given as options or all open tables if no table list was used. Using without any table list will only close tables not in use, and tables not locked by the `FLUSH TABLES` connection. If there are no locked tables, `FLUSH TABLES` will be instant and will not cause any waits, as it no longer waits for tables in use. When a table list is provided, the server will wait for the end of any transactions that are using the tables. Previously, `FLUSH TABLES` only waited for the statements to complete.

Option	Description
TABLE[S] … FOR EXPORT	For InnoDB tables, flushes table changes to disk to permit binary table copies while the server is running. See FLUSH TABLES … FOR EXPORT for more.
TABLE[S] WITH READ LOCK	Closes all open tables. New tables are only allowed to be opened with read locks until an UNLOCK TABLES is given.
TABLE[S] WITH READ LOCK AND DISABLE CHECKPOINT	As `TABLES WITH READ LOCK`, but also disable all checkpoint writes by transactional table engines. This is useful when doing a disk snapshot of all tables.
TABLE_STATISTICS	Reset table statistics (see SHOW TABLE_STATISTICS).
USER_RESOURCES	Resets all per hour user resources. This enables clients that have exhausted their resources to connect again.
USER_STATISTICS	Reset user statistics (see SHOW USER_STATISTICS).
USER_VARIABLES	Reset user variables (see User-defined variables).

You can also use the mariadb-admin client to flush things. Use `mariadb-admin --help` to examine what flush statements it supports.

FLUSH RELAY LOGS

```
FLUSH RELAY LOGS 'connection_name'
```

Compatibility with MySQL

> **CURRENT**
>
> The `FOR CHANNEL` keyword was added for MySQL compatibility. This is identical to using the channel_name directly after the `FLUSH command`. For example, one can now use:
>
> ```
> FLUSH RELAY LOGS FOR CHANNEL 'connection_name';
> ```
>
> **< 10.7.0**
>
> `FOR CHANNEL` isn't available.

FLUSH STATUS

Server status variables can be reset by executing the following:

```
FLUSH STATUS
```

This statement requires the RELOAD privilege.

> **CURRENT**

> Specify FLUSH GLOBAL or FLUSH SESSION. Flushing of global status variables has been moved to `FLUSH GLOBAL STATUS` which is a synonym for `FLUSH STATUS`.
> You can use `old-mode=OLD_FLUSH_STATUS` to restore the old behavior of the `FLUSH STATUS` statement.
>
> **< 11.5**
>
> The variables flushed are mainly session, but some are global. Not all session (or global) variables are flushed - the decision was made per variable.

Global Status Variables that Support `FLUSH STATUS`

Not all global status variables support being reset by `FLUSH STATUS`. The following is an incomplete list of status variables that are reset by `FLUSH GLOBAL STATUS` in MariaDB 11.5 or `FLUSH STATUS` in earlier versions:

- Aborted_clients
- Aborted_connects
- Binlog_cache_disk_use
- Binlog_cache_use
- Binlog_stmt_cache_disk_use
- Binlog_stmt_cache_use
- Connection_errors_accept
- Connection_errors_internal
- Connection_errors_max_connections
- Connection_errors_peer_address
- Connection_errors_select
- Connection_errors_tcpwrap
- Created_tmp_files
- Delayed_errors
- Delayed_writes
- Feature_check_constraint
- Feature_delay_key_write
- Max_used_connection_time
- Max_used_connections
- Opened_plugin_libraries
- Performance_schema_accounts_lost
- Performance_schema_cond_instances_lost
- Performance_schema_digest_lost
- Performance_schema_file_handles_lost
- Performance_schema_file_instances_lost
- Performance_schema_hosts_lost
- Performance_schema_locker_lost
- Performance_schema_mutex_instances_lost
- Performance_schema_rwlock_instances_lost
- Performance_schema_session_connect_attrs_lost
- Performance_schema_socket_instances_lost
- Performance_schema_stage_classes_lost
- Performance_schema_statement_classes_lost
- Performance_schema_table_handles_lost

- Performance_schema_table_instances_lost
- Performance_schema_thread_instances_lost
- Performance_schema_users_lost
- Qcache_hits
- Qcache_inserts
- Qcache_lowmem_prunes
- Qcache_not_cached
- Rpl_semi_sync_master_no_times
- Rpl_semi_sync_master_no_tx
- Rpl_semi_sync_master_timefunc_failures
- Rpl_semi_sync_master_wait_pos_backtraverse
- Rpl_semi_sync_master_yes_tx
- Rpl_transactions_multi_engine
- Server_audit_writes_failed
- Slave_retried_transactions
- Slow_launch_threads
- Ssl_accept_renegotiates
- Ssl_accepts
- Ssl_callback_cache_hits
- Ssl_client_connects
- Ssl_connect_renegotiates
- Ssl_ctx_verify_depth
- Ssl_ctx_verify_mode
- Ssl_finished_accepts
- Ssl_finished_connects
- Ssl_session_cache_hits
- Ssl_session_cache_misses
- Ssl_session_cache_overflows
- Ssl_session_cache_size
- Ssl_session_cache_timeouts
- Ssl_sessions_reused
- Ssl_used_session_cache_entries
- Subquery_cache_hit
- Subquery_cache_miss
- Table_locks_immediate
- Table_locks_waited
- Tc_log_max_pages_used
- Tc_log_page_waits
- Transactions_gtid_foreign_engine
- Transactions_multi_engine

FLUSH TABLES

> **CURRENT**

> `FLUSH TABLES` doesn't cause InnoDB statistics to be reloaded or recalculated. RENAME TABLE, however, triggers a reload of the statistics.
>
> **< 11.8.2 / 11.4.6**
>
> `FLUSH TABLES` causes InnoDB statistics to be reloaded or recalculated.
>
> **< 10.11.12**
>
> `FLUSH TABLES` causes InnoDB statistics to be reloaded or recalculated.

The purpose of `FLUSH TABLES` is to clean up the open table cache and table definition cache of tables that are not in use. This frees up memory and file descriptors. Normally this is not needed as the caches works on a first-in, first-out basis, but can be useful if the server seems to use too much memory for some reason.

In the following, the purpose of specific `FLUSH TABLES` statements is detailed.

FLUSH TABLES WITH READ LOCK

`FLUSH TABLES WITH READ LOCK` is useful if you want to take a backup of some tables. When the statement returns, all write access to tables is blocked and all tables are marked as "properly closed" on disk. The tables can still be used for read operations.

FLUSH TABLES *table_list*

`FLUSH TABLES`` *table_list*` is useful if you want to copy a table object or files to or from the server. This statement puts a lock that stops new users of the table and will wait until everyone has stopped using the table. The table is then removed from the table definition and table cache.

Note that it's up to the user to ensure that no one is accessing the table between issuing the `FLUSH TABLES` statement and the time the table is copied to or from the server. This can be secured by using LOCK TABLES.

If there are any tables locked by the connection that is using `FLUSH TABLES` all the locked tables will be closed as part of the flush and reopened and relocked before `FLUSH TABLES` returns. This allows one to copy the table after `FLUSH TABLES` returns without having any writes on the table. For now this works with most tables, except InnoDB as InnoDB may do background purges on the table even while it's write locked.

FLUSH TABLES *table_list* WITH READ LOCK

`FLUSH TABLES`` *table_list* ``WITH READ LOCK` works as `FLUSH TABLES WITH READ LOCK`, but only the tables listed are properly closed. In practice, this works exactly like `FLUSH TABLES WITH READ LOCK`, because the `FLUSH` statement must wait for all write operations to end – this is because MariaDB depends on a global read lock for this statement.

Implementation of FLUSH TABLES

- Free memory and file descriptors not in use

Implementation of FLUSH TABLES WITH READ LOCK

- Lock all tables read only for simple old style backup.
- All background writes are suspended and tables are marked as closed.
- No statement requiring table changes are allowed for any user until `UNLOCK TABLES`.

Instead of using `FLUSH TABLE WITH READ LOCK` one should in most cases instead use BACKUP STAGE BLOCK_COMMIT.

Implementation of FLUSH TABLES *table_list*

- Free memory and file descriptors for tables not in use from table list.
- Lock given tables as read only.
- Wait until all translations has ended that uses any of the given tables.
- Wait until all background writes are suspended and tables are marked as closed.

Implementation of FLUSH TABLES *table_list* FOR EXPORT

- Free memory and file descriptors for tables not in use from table list.
- Lock given tables as read.
- Wait until all background writes are suspended and tables are marked as closed.
- Check that all tables supports `FOR EXPORT`.
- No changes to these tables allowed until `UNLOCK TABLES`.

This is basically the same behavior as in older MariaDB versions if you first lock the tables, then do `FLUSH TABLES`. The tables will be copyable until you issue `UNLOCK TABLES`.

FLUSH SSL

The `FLUSH SSL` command can be used to dynamically reinitialize the server's TLS context. This is most useful if you need to replace a certificate that is about to expire without restarting the server.

This operation is performed by reloading the files defined by the following TLS system variables:

- ssl_cert
- ssl_key
- ssl_ca
- ssl_capath
- ssl_crl
- ssl_crlpath

These TLS system variables are not dynamic, so their values can **not** be changed without restarting the server.

If you want to dynamically reinitialize the server's TLS context, then you need to change the certificate and key files at the relevant paths defined by these TLS system variables, without actually changing the values of the variables. See MDEV-19341 for more information.

Reducing Memory Usage

To flush some of the global caches that take up memory, you could execute the following command:

```
FLUSH LOCAL HOSTS,
   QUERY CACHE,
   TABLE_STATISTICS,
   INDEX_STATISTICS,
   USER_STATISTICS;
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

FLUSH QUERY CACHE

Description

You can defragment the query cache to better utilize its memory with the `FLUSH QUERY CACHE` statement. The statement does not remove any queries from the cache.

The RESET QUERY CACHE statement removes all query results from the query cache.
The FLUSH TABLES statement also does this.

This page is licensed: GPLv2, originally from fill_help_tables.sql

FLUSH TABLES FOR EXPORT

Syntax

```
FLUSH TABLE[S] table_name [, table_name] FOR EXPORT
```

Description

`FLUSH TABLES ... FOR EXPORT` flushes changes to the specified tables to disk so that binary copies can be made while the server is still running. This works for Archive, Aria, CSV, InnoDB, MyISAM and MERGE tables.

The table is read locked until you issue UNLOCK TABLES.

If a storage engine does not support `FLUSH TABLES FOR EXPORT`, a 1031 error (SQLSTATE 'HY000') is produced.

If `FLUSH TABLES ... FOR EXPORT` is in effect in the session, the following statements will produce an error if attempted:

- `FLUSH TABLES WITH READ LOCK`
- `FLUSH TABLES ... WITH READ LOCK`
- `FLUSH TABLES ... FOR EXPORT`
- Any statement trying to update any table

If any of the following statements is in effect in the session, attempting `FLUSH TABLES ... FOR EXPORT` will produce an error.

- `FLUSH TABLES ... WITH READ LOCK`
- `FLUSH TABLES ... FOR EXPORT`
- `LOCK TABLES ... READ`
- `LOCK TABLES ... WRITE`

`FLUSH FOR EXPORT` is not written to the binary log.

This statement requires the RELOAD and the LOCK TABLES privileges.

If one of the specified tables cannot be locked, none of the tables will be locked.

If a table does not exist, an error like the following will be produced:

```
ERROR 1146 (42S02): Table 'test.xxx' doesn't exist
```

If a table is a view, an error like the following will be produced:

```
ERROR 1347 (HY000): 'test.v' is not BASE TABLE
```

Example

```
FLUSH TABLES test.t1 FOR EXPORT;
#  Copy files related to the table (see below)
UNLOCK TABLES;
```

For a full description, please see copying MariaDB tables.

See Also

- FLUSH TABLES
- Copying Tables Between Different MariaDB Databases and MariaDB Servers
- Copying Transportable InnoDB Tablespaces
- myisampack - Compressing the MyISAM data file for easier distribution.
- aria_pack - Compressing the Aria data file for easier distribution

This page is licensed: CC BY-SA / Gnu FDL

Plugin Statements

See INSTALL PLUGIN.

Install a specific plugin from a shared library. This statement adds the plugin to the mysql.plugin table and loads its code into the server memory.

See INSTALL SONAME.

Load all plugins contained within a shared library file. This statement automatically discovers and installs every valid plugin found in the specified library.

See UNINSTALL PLUGIN.

Remove a specific plugin from the server. This statement unloads the plugin code and deletes its entry from the mysql.plugin table to prevent reloading.

See UNINSTALL SONAME.

Uninstall all plugins loaded from a specific library. This statement removes every plugin associated with the library file and unloads the library itself.

This page is licensed: CC BY-SA / Gnu FDL

INSTALL PLUGIN

Syntax

```
INSTALL PLUGIN [IF NOT EXISTS] plugin_name SONAME 'plugin_library'
```

Description

This statement installs an individual plugin from the specified library. To install the whole library (which could be required), use INSTALL SONAME. See also Installing a Plugin.

`plugin_name` is the name of the plugin as defined in theplugin declaration structure contained in the library file. Plugin names arenot case sensitive. For maximal compatibility, plugin names should be limitedto ASCII letters, digits, and underscore, because they are used in C sourcefiles, shell command lines, M4 and Bourne shell scripts, and SQL environments.

`plugin_library` is the name of the shared library thatcontains the plugin code. The file name extension can be omitted (which makes the statement look the same on all architectures).

The shared library must be located in the plugin directory (that is,the directory named by the plugin_dir system variable). The library must be in the plugin directory itself, not in a subdirectory. Bydefault, `plugin_dir` is plugin directory under the directory named bythe `pkglibdir` configuration variable, but it can be changed by settingthe value of `plugin_dir` at server startup. For example, setits value in a `my.cnf` file:

```
[mariadbd]
plugin_dir=/path/to/plugin/directory
```

If the value of plugin_dir is a relative path name, it istaken to be relative to the base directory (the value of the basedir system variable).

`INSTALL PLUGIN` adds a line to the `mysql.plugin` table thatdescribes the plugin. This table contains the plugin name and library filename.

`INSTALL PLUGIN` causes the server to readoption (`my.cnf`) files just as during server startup. This enables the plugin topick up any relevant options from those files. It is possible to add pluginoptions to an option file even before loading a plugin (if the loose prefix isused). It is also possible to uninstall a plugin, edit `my.cnf`, and install theplugin again. Restarting the plugin this way enables it to the new optionvalues without a server restart.

`INSTALL PLUGIN` also loads and initializes the plugin code tomake the plugin available for use. A plugin is initialized by executing itsinitialization function, which handles any setup that the plugin must performbefore it can be used.

To use `INSTALL PLUGIN`, you must have the INSERT privilege for the `mysql.plugin` table.

At server startup, the server loads and initializes any plugin that islisted in the `mysql.plugin` table. This means that a plugin is installedwith `INSTALL PLUGIN` only once, not every time the serverstarts. Plugin loading at startup does not occur if the server is started with the `--skip-grant-tables` option.

When the server shuts down, it executes the de-initialization functionfor each plugin that is loaded so that the plugin has a chance toperform any final cleanup.

If you need to load plugins for a single server startup when the `--skip-grant-tables` option is given (which tells the servernot to read system tables), use the `--plugin-load` mariadbd option.

IF NOT EXISTS

When the `IF NOT EXISTS` clause is used, MariaDB will return a note instead of an error if the specified plugin already exists. See SHOW WARNINGS.

Examples

```
INSTALL PLUGIN sphinx SONAME 'ha_sphinx.so';
```

The extension can also be omitted:

```
INSTALL PLUGIN innodb SONAME 'ha_xtradb';
```

```
INSTALL PLUGIN IF NOT EXISTS example SONAME 'ha_example';
Query OK, 0 rows affected (0.104 sec)

INSTALL PLUGIN IF NOT EXISTS example SONAME 'ha_example';
Query OK, 0 rows affected, 1 warning (0.000 sec)

SHOW WARNINGS;
+-------+------+-------------------------------------+
| Level | Code | Message                             |
+-------+------+-------------------------------------+
| Note  | 1968 | Plugin 'example' already installed |
+-------+------+-------------------------------------+
```

See Also

- List of Plugins
- Plugin Overview

- INFORMATION_SCHEMA.PLUGINS Table
- mariadb-plugin
- SHOW PLUGINS
- INSTALL SONAME
- UNINSTALL PLUGIN
- UNINSTALL SONAME

This page is licensed: GPLv2, originally from fill_help_tables.sql

INSTALL SONAME

Syntax

```
INSTALL SONAME 'plugin_library'
```

Description

This statement is a variant of INSTALL PLUGIN. It installs **all** plugins from a given `plugin_library`. See INSTALL PLUGIN for details.

`plugin_library` is the name of the shared library thatcontains the plugin code. The file name extension (forexample, `libmyplugin.so` or `libmyplugin.dll`) can be omitted (which makes the statement look the same on all architectures).

The shared library must be located in the plugin directory (that is,the directory named by the plugin_dir system variable). The library must be in the plugin directory itself, not in a subdirectory. Bydefault, `plugin_dir` is plugin directory under the directory named bythe `pkglibdir` configuration variable, but it can be changed by settingthe value of `plugin_dir` at server startup. For example, setits value in a `my.cnf` file:

```
[mariadbd]
plugin_dir=/path/to/plugin/directory
```

If the value of plugin_dir is a relative path name, it istaken to be relative to the MySQL base directory (the value of the `basedir` system variable).

`INSTALL SONAME` adds one or more lines to the `mysql.plugin` table thatdescribes the plugin. This table contains the plugin name and library filename.

`INSTALL SONAME` causes the server to readoption (`my.cnf`) files just as during server startup. This enables the plugin topick up any relevant options from those files. It is possible to add pluginoptions to an option file even before loading a plugin (if the loose prefix isused). It is also possible to uninstall a plugin, edit `my.cnf`, and install theplugin again. Restarting the plugin this way enables it to the new optionvalues without a server restart.

`INSTALL SONAME` also loads and initializes the plugin code tomake the plugin available for use. A plugin is initialized by executing itsinitialization function, which handles any setup that the plugin must performbefore it can be used.

To use `INSTALL SONAME`, you must have the INSERT privilege for the `mysql.plugin` table.

At server startup, the server loads and initializes any plugin that islisted in the `mysql.plugin` table. This means that a plugin is installedwith `INSTALL SONAME` only once, not every time the serverstarts. Plugin loading at startup does not occur if the server is started with the `--skip-grant-tables` option.

When the server shuts down, it executes the de-initialization functionfor each plugin that is loaded so that the plugin has a chance toperform any final cleanup.

If you need to load plugins for a single server startup when the `--skip-grant-tables` option is given (which tells the servernot to read system tables), use the `--plugin-load` mariadbd option.

If you need to install only one plugin from a library, use the INSTALL PLUGIN statement.

Examples

To load the LOCALES plugin and all of its `information_schema` tables with one statement, use

```
INSTALL SONAME 'locales';
```

This statement can be used instead of `INSTALL PLUGIN` even when the library contains only one plugin:

```
INSTALL SONAME 'ha_sequence';
```

See Also

- List of Plugins
- Plugin Overview
- SHOW PLUGINS
- INSTALL PLUGIN
- UNINSTALL PLUGIN
- UNINSTALL SONAME
- SHOW PLUGINS
- INFORMATION_SCHEMA.PLUGINS Table
- mysql_plugin

This page is licensed: CC BY-SA / Gnu FDL

UNINSTALL PLUGIN

Syntax

```
UNINSTALL PLUGIN [IF EXISTS] plugin_name
```

Description

This statement removes a single installed plugin. To uninstall the whole library which contains the plugin, use UNINSTALL SONAME. You cannot uninstall a plugin if any table that uses it is open.

`plugin_name` must be the name of some plugin that is listedin the mysql.plugin table. The server executes the plugin's deinitializationfunction and removes the row for the plugin from the `mysql.plugin` table, so that subsequent server restarts will not load and initializethe plugin. `UNINSTALL PLUGIN` does not remove the plugin'sshared library file.

To use `UNINSTALL PLUGIN`, you must have the DELETE privilege for the mysql.plugin table.

IF EXISTS

If the `IF EXISTS` clause is used, MariaDB will return a note instead of an error if the plugin does not exist. See SHOW WARNINGS.

Examples

```
UNINSTALL PLUGIN example;
```

```
UNINSTALL PLUGIN IF EXISTS example;
Query OK, 0 rows affected (0.099 sec)

UNINSTALL PLUGIN IF EXISTS example;
Query OK, 0 rows affected, 1 warning (0.000 sec)

SHOW WARNINGS;
+-------+------+------------------------------+
| Level | Code | Message                      |
+-------+------+------------------------------+
| Note  | 1305 | PLUGIN example does not exist |
+-------+------+------------------------------+
```

See Also

- Plugin Overview
- mysql_plugin
- INSTALL PLUGIN
- List of Plugins

This page is licensed: GPLv2, originally from fill_help_tables.sql

UNINSTALL SONAME

Syntax

```
UNINSTALL SONAME  [IF EXISTS] 'plugin_library'
```

Description

This statement is a variant of UNINSTALL PLUGIN statement, that removes all plugins belonging to a specified `plugin_library`. See UNINSTALL PLUGIN for details.

`plugin_library` is the name of the shared library thatcontains the plugin code. The file name extension (forexample, `libmyplugin.so` or `libmyplugin.dll`) can be omitted (which makes the statement look the same on all architectures).

To use `UNINSTALL SONAME`, you must have the DELETE privilege for the `mysql.plugin` table.

IF EXISTS

If the `IF EXISTS` clause is used, MariaDB will return a note instead of an error if the plugin library does not exist. See SHOW WARNINGS.

Examples

To uninstall the XtraDB plugin and all of its `information_schema` tables with one statement, use

```
UNINSTALL SONAME 'ha_xtradb';
```

```
UNINSTALL SONAME IF EXISTS 'ha_example';
Query OK, 0 rows affected (0.099 sec)

UNINSTALL SONAME IF EXISTS 'ha_example';
Query OK, 0 rows affected, 1 warning (0.000 sec)

SHOW WARNINGS;
+-------+------+-------------------------------------+
| Level | Code | Message                             |
+-------+------+-------------------------------------+
| Note  | 1305 | SONAME ha_example.so does not exist |
+-------+------+-------------------------------------+
```

See Also

- INSTALL SONAME
- SHOW PLUGINS
- INSTALL PLUGIN
- UNINSTALL PLUGIN
- SHOW PLUGINS
- INFORMATION_SCHEMA.PLUGINS Table
- mysql_plugin
- List of Plugins

This page is licensed: CC BY-SA / Gnu FDL

Replication Statements

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

See CHANGE MASTER TO.

See RESET MASTER.

See RESET REPLICA.

See SET GLOBAL SQL_SLAVE_SKIP_COUNTER.

See START REPLICA.

See STOP REPLICA.

This page is licensed: CC BY-SA / Gnu FDL

CHANGE MASTER TO

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

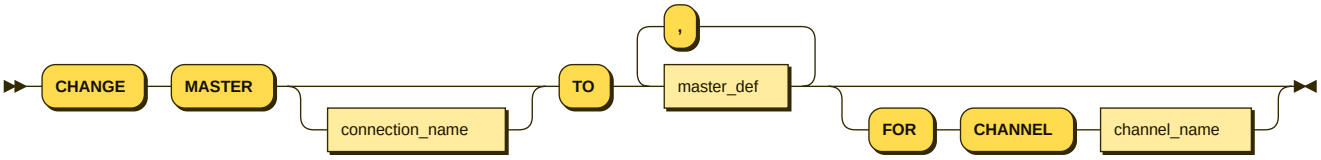
Description

The `CHANGE MASTER TO` statement sets up a replication replica server to connect to a specific primary server and defines the replication coordinates (binary log file and position or GTID) and connection parameters (host, user, password, port, SSL options, etc.). When executed, the replica updates its internal replication metadata accordingly.

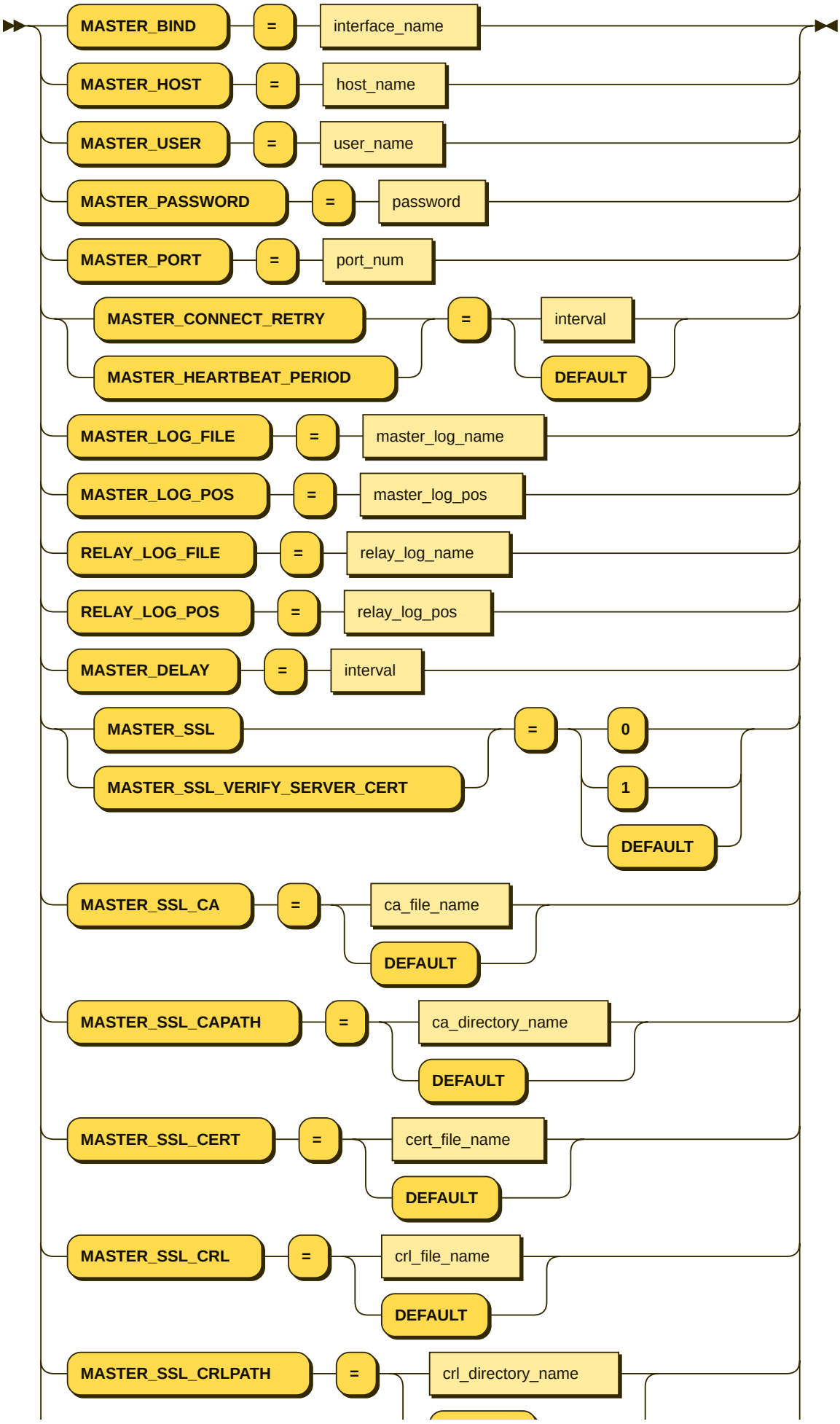
Syntax

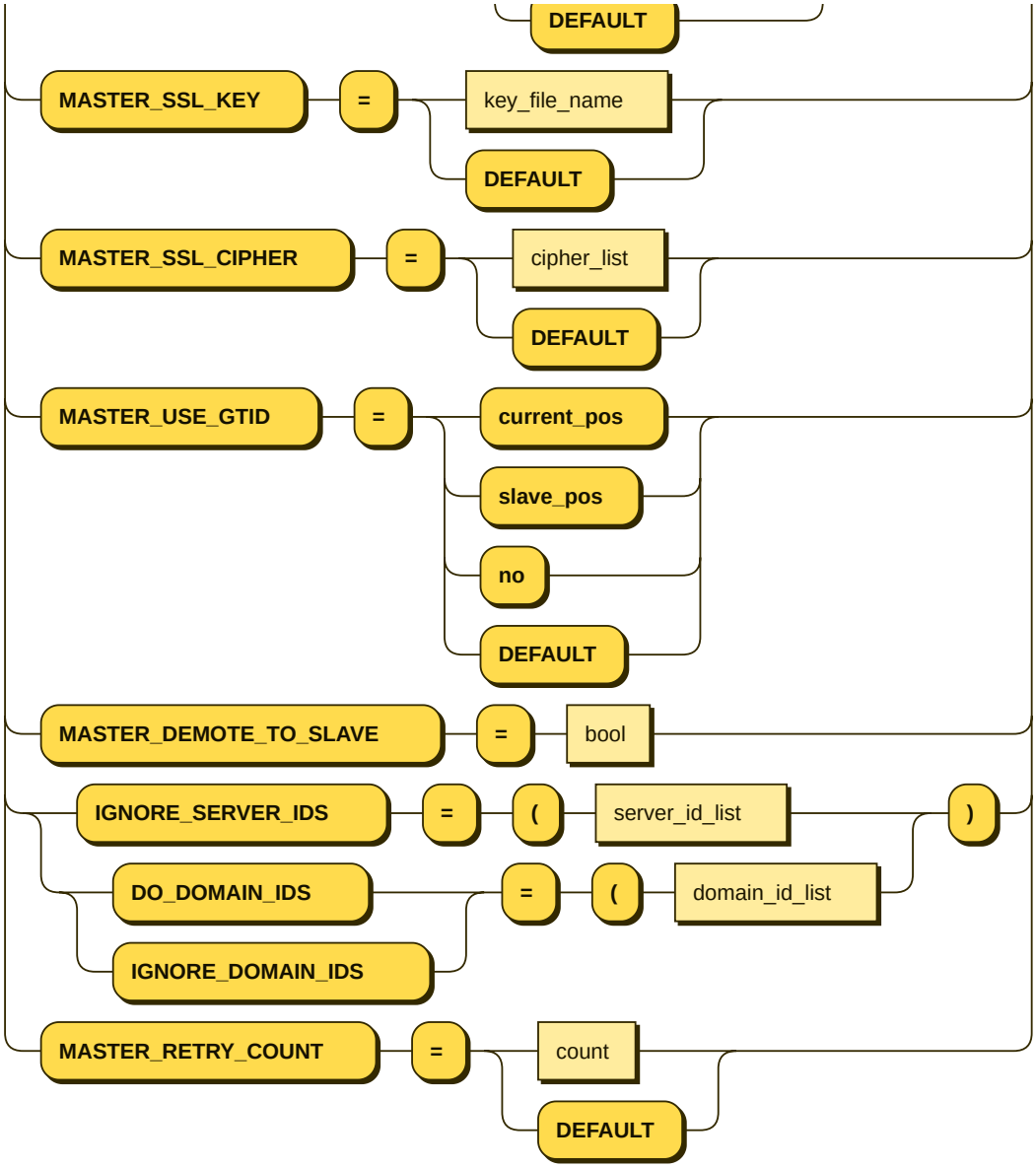
```
CHANGE MASTER ['connection_name'] TO master_def  [, master_def] ...
  [FOR CHANNEL 'channel_name']

master_def:
    MASTER_BIND = 'interface_name'
  | MASTER_HOST = 'host_name'
  | MASTER_USER = 'user_name'
  | MASTER_PASSWORD = 'password'
  | MASTER_PORT = port_num
  | MASTER_CONNECT_RETRY = {interval | DEFAULT}
  | MASTER_HEARTBEAT_PERIOD = {interval | DEFAULT}
  | MASTER_LOG_FILE = 'master_log_name'
  | MASTER_LOG_POS = master_log_pos
  | RELAY_LOG_FILE = 'relay_log_name'
  | RELAY_LOG_POS = relay_log_pos
  | MASTER_DELAY = interval
  | MASTER_SSL = {0 | 1 | DEFAULT}
  | MASTER_SSL_CA = {'ca_file_name' | DEFAULT}
  | MASTER_SSL_CAPATH = {'ca_directory_name' | DEFAULT}
  | MASTER_SSL_CERT = {'cert_file_name' | DEFAULT}
  | MASTER_SSL_CRL = {'crl_file_name' | DEFAULT}
  | MASTER_SSL_CRLPATH = {'crl_directory_name' | DEFAULT}
  | MASTER_SSL_KEY = {'key_file_name' | DEFAULT}
  | MASTER_SSL_CIPHER = {'cipher_list' | DEFAULT}
  | MASTER_SSL_VERIFY_SERVER_CERT = {0 | 1 | DEFAULT}
  | MASTER_USE_GTID = {current_pos | slave_pos | no | DEFAULT}
  | MASTER_DEMOTE_TO_SLAVE = bool
  | IGNORE_SERVER_IDS = (server_id_list)
  | DO_DOMAIN_IDS = ([N,..])
  | IGNORE_DOMAIN_IDS = ([N,..])
  | MASTER_RETRY_COUNT = {long | DEFAULT}
```



Railroad diagram of CHANGE MASTER TO — equivalent to the BNF above


MASTER_BIND
=
interface_name
MASTER_HOST
=
host_name
MASTER_USER
=
user_name
MASTER_PASSWORD
=
password
MASTER_PORT
=
port_num
MASTER_CONNECT_RETRY
MASTER_HEARTBEAT_PERIOD
=
interval
DEFAULT
MASTER_LOG_FILE
=
master_log_name
MASTER_LOG_POS
=
master_log_pos
RELAY_LOG_FILE
=
relay_log_name
RELAY_LOG_POS
=
relay_log_pos
MASTER_DELAY
=
interval
MASTER_SSL
MASTER_SSL_VERIFY_SERVER_CERT
=
0
1
DEFAULT
MASTER_SSL_CA
=
ca_file_name
DEFAULT
MASTER_SSL_CAPATH
=
ca_directory_name
DEFAULT
MASTER_SSL_CERT
=
cert_file_name
DEFAULT
MASTER_SSL_CRL
=
crl_file_name
DEFAULT
MASTER_SSL_CRLPATH
=
crl_directory_name



Railroad diagram of master_def

Note: The value is extracted from the corresponding server option or system variable that support `DEFAULT`. This allows you to reset a replication configuration parameter to its server-level configuration without providing an explicit value.

Replication Configuration

`CHANGE MASTER` is used on a replica to set up or change replication settings for connecting to the primary.

> **CURRENT**
>
> The `FOR CHANNEL` keyword was added for MySQL compatibility. This is identical to using the channel_name directly after `CHANGE MASTER`.
>
> **< 10.7.0**
>
> `FOR CHANNEL` is not available.

Multi-Source Replication

If you are using multi-source replication, then you need to specify a connection name when you execute `CHANGE MASTER`. There are two ways to do this:

- Setting the default_master_connection system variable prior to executing `CHANGE MASTER`.
- Setting the `connection_name` parameter when executing `CHANGE MASTER`.

default_master_connection

```
SET default_master_connection = 'gandalf';
STOP SLAVE;
CHANGE MASTER TO
   MASTER_PASSWORD='new3cret';
START SLAVE;
```

connection_name

```
STOP SLAVE 'gandalf';
CHANGE MASTER 'gandalf' TO
   MASTER_PASSWORD='new3cret';
START SLAVE 'gandalf';
```

Options

Using Configurable Defaults

Starting with MariaDB 12.3, replication connection parameters can be set at the server level (for example, in configuration files like `my.cnf` or via command-line options) and reused across replication channels.

When an option in `CHANGE MASTER TO` is set to `DEFAULT`, the value is obtained from the corresponding server option (e.g., options defined in configuration files like `my.cnf` or provided on the command line). Instead of keeping a fixed value, the replication channel derives the value from the appropriate server option when `DEFAULT` is selected.

CHANGE MASTER Option	System Variable	Description
`MASTER_CONNECT_RETRY`	`--master-connect-retry`	The interval to wait between connection retry attempts.
`MASTER_RETRY_COUNT`	`--master-retry-count`	Number of connection attempts before stopping.
`MASTER_HEARTBEAT_PERIOD`	`--master-heartbeat-period`	Interval between replication heartbeats.
`MASTER_SSL`	`--master-ssl`	Enables or disables TLS for the connection.
`MASTER_SSL_CA`	`--master-ssl-ca`	Path to the Certificate Authority (CA) file.
`MASTER_SSL_CERT`	`--master-ssl-cert`	Path to the client certificate file.
`MASTER_SSL_KEY`	`--master-ssl-key`	Path to the client private key file.
`MASTER_SSL_CAPATH`	`--master-ssl-capath`	Path to the directory containing CA certificates.
`MASTER_SSL_VERIFY_SERVER_CERT`	`--master-ssl-verify-server-cert`	Enable verification of the primary's certificate.
`MASTER_SSL_CRL`	`--master-ssl-crl`	Path to the Certificate Revocation List (CRL) file.
`MASTER_SSL_CRLPATH`	`--master-ssl-crlpath`	Path to the directory containing CRL files.
`MASTER_SSL_CIPHER`	`--master-ssl-cipher`	List of permitted TLS ciphers.
`MASTER_USE_GTID`	`--master-use-gtid`	Setting for Global Transaction ID (GTID) mode.

For example, a configuration file can specify default values:

```
# Defaults replication connection options
master_ssl_ca   = /etc/mysql/ssl/ca.pem
master_ssl_cert = /etc/mysql/ssl/client-cert.pem
master_ssl_key  = /etc/mysql/ssl/client-key.pem
master_use_gtid = slave_pos
```

These values can then be reused by specifying `DEFAULT`:

```
-- Setting up a specific channel using global defaults
CHANGE MASTER 'primary_node_1' TO
  MASTER_HOST = '10.0.0.5',
  MASTER_SSL_CA = DEFAULT,
  MASTER_SSL_CERT = DEFAULT,
  MASTER_SSL_KEY = DEFAULT,
  MASTER_USE_GTID = DEFAULT;

START SLAVE 'primary_node_1';
```

If an option is explicitly set in `CHANGE MASTER TO`, the value overrides the corresponding server option. The replication channel receives the current server-level value if `DEFAULT` is used.

Values set to `DEFAULT` are fixed dynamically, thus changes to server options (for example, after a restart) are automatically reflected, whereas explicitly defined values remain unchanged.

Behavior and Persistence

When `DEFAULT` is used, the option is stored in the replication metadata as `DEFAULT` instead of a specific value. The new value will be used automatically if the corresponding server option changes (for instance, after a server restart).

Changes to server settings have no effect on an explicit value that is provided instead of `DEFAULT`. When a `CHANGE MASTER` option is not explicitly set, it acts as `DEFAULT`.

Configuration Precedence

Values explicitly defined in `CHANGE MASTER TO` take precedence over server settings.

The replication channel can inherit values from server-level configuration rather than overriding them when the `DEFAULT` keyword is used.

Autoset Options

The following options can override manual defaults:

- `--autoset-master-heartbeat-period`
- `--autoset-master-use-gtid`

These options determine values automatically and may override `--master-*` settings.

Connection Options

MASTER_USER

The `MASTER_USER` option for `CHANGE MASTER` defines the user account that the replica will use to connect to the primary.

> **CURRENT**
>
> This user account will need the REPLICATION REPLICA privilege on the primary.
>
> **< 10.5.1**

This user account will need the REPLICATION SLAVE privilege on the primary.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_USER='repl',
   MASTER_PASSWORD='new3cret';
START SLAVE;
```

CURRENT

The maximum length of the `MASTER_USER` string is 128 characters.

< 10.6

The maximum length of the `MASTER_USER` string is 96 characters.

MASTER_PASSWORD

The `MASTER_PASSWORD` option for `CHANGE MASTER` defines the password that the replica will use to connect to the primary as the user account defined by the MASTER_USER option.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_PASSWORD='new3cret';
START SLAVE;
```

The maximum length of the `MASTER_PASSWORD` string is 32 characters. The effective maximum length of the string depends on how many bytes are used per character and can be up to 96 characters.

Warning:

Due to MDEV-29994, the password can be silently truncated to 41 characters when MariaDB is restarted. For this reason, it is recommended to use a password that is shorter than this.

MASTER_HOST

The `MASTER_HOST` option for `CHANGE MASTER` defines the hostname or IP address of the primary.

CURRENT

If you set the value of the `MASTER_HOST` option to the empty string, then that is not the same as not setting the option's value at all. If you set the value of the `MASTER_HOST` option to the empty string, then the `CHANGE MASTER` command will fail with an error.

< 5.4

If you set the value of the `MASTER_HOST` option to the empty string, then that is not the same as not setting the option's value at all. If you set the value of the `MASTER_HOST` option to the empty string, then the `CHANGE MASTER` command will fail with an error. In MariaDB 5.3 and before, if you set the value of the `MASTER_HOST` option to the empty string, then the `CHANGE MASTER` command would succeed, but the subsequent START REPLICA command would fail.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_HOST='dbserver1.example.com',
   MASTER_USER='repl',
   MASTER_PASSWORD='new3cret',
   MASTER_USE_GTID=slave_pos;
START SLAVE;
```

Note:

If you set the value of the `MASTER_HOST` option in a `CHANGE MASTER` command, then the replica assumes that the primary is different from before, even if you set the value of this option to the same value it had previously. In this scenario, the replica will consider the old values for the primary's binary log file name and position to be invalid for the new primary. As a side effect, if you do not explicitly set the values of the MASTER_LOG_FILE and MASTER_LOG_POS options in the statement, then the statement will be implicitly appended with `MASTER_LOG_FILE=''` and `MASTER_LOG_POS=4`. However, if you enable GTID mode for replication by setting the MASTER_USE_GTID option to some value other than `no` in the statement, then these values will effectively be ignored anyway.

Note:

Replicas cannot connect to primaries using Unix socket files or Windows named pipes. The replica must connect to the primary using TCP/IP.

CURRENT

The maximum length of the `MASTER_HOST` string is 255 characters.

< 10.6

The maximum length of the `MASTER_HOST` string is 60 characters.

MASTER_PORT

The `MASTER_PORT` option for `CHANGE MASTER` defines the TCP/IP port of the primary.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_HOST='dbserver1.example.com',
   MASTER_PORT=3307,
   MASTER_USER='repl',
   MASTER_PASSWORD='new3cret',
   MASTER_USE_GTID=slave_pos;
START SLAVE;
```

Note:

If you set the value of the `MASTER_PORT` option in a `CHANGE MASTER` command, then the replica assumes that the primary is different from before, even if you set the value of this option to the same value it had previously. In this scenario, the replica will consider the old values for the primary's binary log file name and position to be invalid for the new primary. As a side effect, if you do not explicitly set the values of the MASTER_LOG_FILE and MASTER_LOG_POS options in the statement, then the statement will be implicitly appended with `MASTER_LOG_FILE=''` and `MASTER_LOG_POS=4`. However, if you enable GTID mode for replication by setting the MASTER_USE_GTID option to some value other than `no` in the statement, then these values will effectively be ignored anyway.

> **Note:**
>
> Replicas cannot connect to primaries using Unix socket files or Windows named pipes. The replica must connect to the primary using TCP/IP.

MASTER_CONNECT_RETRY

The `MASTER_CONNECT_RETRY` option for `CHANGE MASTER` defines how many seconds that the replica will wait between connection retries. The default is `60`.

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_CONNECT_RETRY=20;
START SLAVE;
```

> Starting with MariaDB 12.3, this option accepts the `DEFAULT` keyword. When set to `DEFAULT`, the value is taken from the corresponding server option (for example, `--master-connect-retry` defined in the configuration files or on the command line).
>
> The option can now be set to 0, which disables the wait between connection retries.

MASTER_RETRY_COUNT

The `MASTER_RETRY_COUNT` option limits the number of connection attempts (i.e., `Connects_Tried` in SHOW REPLICA STATUS). For example:

```
STOP SLAVE;
CHANGE MASTER TO
  MASTER_RETRY_COUNT=1; # attempt only once; do not retry if it fails
START SLAVE;
```

Setting this option resets the `Connects_Tried` statistic in SHOW REPLICA STATUS to 0.

The default is the `--master-retry-count` option, which can be set either on the command-line or in a server option group in an option file prior to starting up the server. For example:

> **< 12.0**
>
> The `MASTER_RETRY_COUNT` option for `CHANGE MASTER` is only supported by MariaDB 12.0.1 and later and by MySQL. Please use the `--master-retry-count` option instead, which be set either on the command-line or in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
master_retry_count=4294967295
```

The maximum value is 18446744073709551615 (64-bit). Before MariaDB 12.3, the maximum value for Windows was 4294967295.

> Starting with MariaDB 12.3, this option accepts the `DEFAULT` keyword. When set to `DEFAULT`, the value is taken from the corresponding server option (for example, `--master-retry-count`).
>
> The option can now be set to `0`, allowing the replica to retry indefinitely.
> When it is set to 1, the connection attempt is only made once, hence disabling retries.

MASTER_BIND

> **Warning:**
>
> The `MASTER_BIND` option for `CHANGE MASTER` is only supported by MySQL 5.6.2 and later and by MySQL NDB Cluster 7.3.1 and later. This option is not supported by MariaDB. See MDEV-19248 for more information.

The `MASTER_BIND` option for `CHANGE MASTER` can be used on replicas that have multiple network interfaces to choose which network interface the replica will use to connect to the primary.

MASTER_HEARTBEAT_PERIOD

The `MASTER_HEARTBEAT_PERIOD` option for `CHANGE MASTER` can be used to set the interval in seconds between replication heartbeats. Whenever the primary's binary log is updated with an event, the waiting period for the next heartbeat is reset.

This option's *interval* argument has the following characteristics:

- It is a decimal value with a range of `0` to `4294967.295` seconds.
- It has a resolution of hundredths of a second.
- Its smallest valid non-zero value is `0.001`.
- Its default value is the value of the slave_net_timeout system variable divided by 2.
- If it's set to `0`, then heartbeats are disabled.

Heartbeats are sent by the primary only if there are no unsent events in the binary log file for a period longer than the interval.

> Starting with MariaDB 12.3, this option accepts the `DEFAULT` keyword. When set to `DEFAULT`, the value is taken from the corresponding server option (for example, `--master-heartbeat-period`).

If the RESET REPLICA statement is executed, then the heartbeat interval is reset to the default.

> **Note:**
>
> If the slave_net_timeout system variable is set to a value that is lower than the current heartbeat interval, then a warning will be issued.

TLS Options

The TLS options are used for providing information about TLS. The options can be set even on replicas that are compiled without TLS support. The TLS options are saved to either the default `master.info` file or the file that is configured by the master_info_file option, but these TLS options are ignored unless the replica supports TLS.

See Replication with Secure Connections for more information.

Starting with MariaDB 12.3, the `DEFAULT` keyword is accepted by the TLS options, enabling replication channels to share a central configuration specified in the server's option file. When `DEFAULT` is used, values are derived from the corresponding server options (e.g., `MASTER_SSL_CA` derives from `master_ssl_ca`). See Default System Variables for a full list of supported variables.

MASTER_SSL

The `MASTER_SSL` option for `CHANGE MASTER` tells the replica whether to force TLS for the connection. The valid values are `0`, `1`, or `DEFAULT` (available since MariaDB 12.3). The option must be set to `1` for the other `MASTER_SSL*` options to have any effect.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_SSL=1;
START SLAVE;
```

> Starting with MariaDB 12.3, this option accepts `DEFAULT` keyword. If `DEFAULT` is used, the value is derived from the corresponding server option (for example, `--master-ssl` defined in configuration files or on the command line).

MASTER_SSL_CA

The `MASTER_SSL_CA` option for `CHANGE MASTER` defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for TLS. This option requires that you use an absolute path, not a relative path.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
   MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
   MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
   MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;
```

See Secure Connections Overview: Certificate Authorities (CAs) for more information. The maximum length of `MASTER_SSL_CA` string is 511 characters.

> Starting with MariaDB 12.3, this option accepts a path or `DEFAULT` keyword. If `DEFAULT` is used, the value is inherited from the corresponding server option (for example, `--master-ssl-ca`).

MASTER_SSL_CAPATH

The `MASTER_SSL_CAPATH` option for `CHANGE MASTER` defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the openssl rehash command.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
   MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
   MASTER_SSL_CAPATH='/etc/my.cnf.d/certificates/ca/',
   MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;
```

See Secure Connections Overview: Certificate Authorities (CAs) for more information. The maximum length of `MASTER_SSL_CA_PATH` string is 511 characters.

> Starting with MariaDB 12.3, this option accepts a path or `DEFAULT` keyword. If `DEFAULT` is used, the value is taken from the corresponding server option (for example, `--master-ssl-capath`).

MASTER_SSL_CERT

The `MASTER_SSL_CERT` option for `CHANGE MASTER` defines a path to the X509 certificate file to use for TLS. This option requires that you use the absolute path, not a relative path.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
   MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
   MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
   MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;
```

The maximum length of `MASTER_SSL_CERT` string is 511 characters.

> Starting with MariaDB 12.3, this option accepts a path or `DEFAULT` keyword. When set to `DEFAULT`, the value is inherited from the corresponding server option (for example, `--master-ssl-cert`).

MASTER_SSL_CRL

The `MASTER_SSL_CRL` option for `CHANGE MASTER` defines a path to a PEM file that should contain one or more revoked X509 certificates to use for TLS. This option requires that you use the absolute path, not a relative path.

This option is only supported if the server was built with OpenSSL. If the server was built with yaSSL, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
   MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
   MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
   MASTER_SSL_VERIFY_SERVER_CERT=1,
   MASTER_SSL_CRL='/etc/my.cnf.d/certificates/crl.pem';
START SLAVE;
```

See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. The maximum length of `MASTER_SSL_CRL` string is 511 characters.

> Starting with MariaDB 12.3, this option also accepts the `DEFAULT` keyword. When set to `DEFAULT`, the value is derived from the corresponding server option (for example, `--master-ssl-crl`).

MASTER_SSL_CRLPATH

The `MASTER_SSL_CRLPATH` option for `CHANGE MASTER` defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for TLS. This option requires that you use the absolute path, not a relative path. The directory specified by this variable needs to be run through the openssl rehash command.

This option is only supported if the server was built with OpenSSL. If the server was built with yaSSL, then this option is not supported. See TLS and Cryptography Libraries Used by MariaDB for more information about which libraries are used on which platforms.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
   MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
   MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
   MASTER_SSL_VERIFY_SERVER_CERT=1,
   MASTER_SSL_CRLPATH='/etc/my.cnf.d/certificates/crl/';
START SLAVE;
```

See Secure Connections Overview: Certificate Revocation Lists (CRLs) for more information. The maximum length of `MASTER_SSL_CRL_PATH` string is 511 characters.

> Starting with MariaDB 12.3, this option also accepts the `DEFAULT` keyword. When set to `DEFAULT`, the value is taken from the corresponding server option (for example, `--master-ssl-crlpath`).

MASTER_SSL_KEY

The `MASTER_SSL_KEY` option for `CHANGE MASTER` defines a path to a private key file to use for TLS. This option requires that you use the absolute path, not a relative path.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
   MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
   MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
   MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;
```

The maximum length of `MASTER_SSL_KEY` string is 511 characters.

> Starting with MariaDB 12.3, this option also accepts the `DEFAULT` keyword. When set to `DEFAULT`, the value is derived from the corresponding server option (for example, `--master-ssl-key`).

MASTER_SSL_CIPHER

The `MASTER_SSL_CIPHER` option for `CHANGE MASTER` defines the list of permitted ciphers or cipher suites to use for TLS. Besides cipher names, if MariaDB was compiled with OpenSSL, this option could be set to “SSLv3” or “TLSv1.2” to allow all SSLv3 or all TLSv1.2 ciphers. Note that the TLSv1.3 ciphers cannot be excluded when using OpenSSL, even by using this option. See Using TLSv1.3 for details.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
   MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
   MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
   MASTER_SSL_VERIFY_SERVER_CERT=1,
   MASTER_SSL_CIPHER='TLSv1.2';
START SLAVE;
```

The maximum length of `MASTER_SSL_CIPHER` string is 511 characters.

> Starting with MariaDB 12.3, this option also accepts the `DEFAULT` keyword. When set to `DEFAULT`, the value is taken from the corresponding server option (for example, `--master-ssl-cipher`).

MASTER_SSL_VERIFY_SERVER_CERT

CURRENT

The `MASTER_SSL_VERIFY_SERVER_CERT` option for `CHANGE MASTER` enables server certificate verification. This option is enabled by default.

< 11.4

The `MASTER_SSL_VERIFY_SERVER_CERT` option for `CHANGE MASTER` enables server certificate verification. This option is disabled by default.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
   MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
   MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
   MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;
```

See Secure Connections Overview: Server Certificate Verification for more information.

> Starting with MariaDB 12.3, this option also accepts the `DEFAULT` keyword. When set to `DEFAULT`, the value is derived from the corresponding server option (for example, `--master-ssl-verify-server-cert`).

Binary Log Options

These options are related to the binary log position on the primary.

MASTER_LOG_FILE

The `MASTER_LOG_FILE` option for `CHANGE MASTER` can be used along with `MASTER_LOG_POS` to specify the coordinates at which the replica's I/O thread should begin reading from the primary's binary logs the next time the thread starts.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_LOG_FILE='master2-bin.001',
   MASTER_LOG_POS=4;
START SLAVE;
```

> **Note:**
>
> The MASTER_LOG_FILE and MASTER_LOG_POS options cannot be specified if the RELAY_LOG_FILE and RELAY_LOG_POS options were also specified.

> **Note:**
>
> The MASTER_LOG_FILE and MASTER_LOG_POS options are effectively ignored if you enable GTID mode for replication by setting the MASTER_USE_GTID option to some value other than `no` in the statement.

MASTER_LOG_POS

The `MASTER_LOG_POS` option for `CHANGE MASTER` can be used along with `MASTER_LOG_FILE` to specify the coordinates at which the replica's I/O thread should begin reading from the primary's binary logs the next time the thread starts.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_LOG_FILE='master2-bin.001',
   MASTER_LOG_POS=4;
START SLAVE;
```

> **Note:**
>
> The MASTER_LOG_FILE and MASTER_LOG_POS options cannot be specified if the RELAY_LOG_FILE and RELAY_LOG_POS options were also specified.

> **Note:**
>
> The MASTER_LOG_FILE and MASTER_LOG_POS options are effectively ignored if you enable GTID mode for replication by setting the MASTER_USE_GTID option to some value other than `no` in the statement.

Relay Log Options

These options are related to the relay log position on the replica.

RELAY_LOG_FILE

The `RELAY_LOG_FILE` option for `CHANGE MASTER` can be used along with the RELAY_LOG_POS option to specify the coordinates at which the replica's SQL thread should begin reading from the relay log the next time the thread starts.

The `CHANGE MASTER` statement usually deletes all relay log files. However, if the `RELAY_LOG_FILE` and/or `RELAY_LOG_POS` options are specified, then existing relay log files are kept.

When you want to change the relay log position, you only need to stop the replica's SQL thread. The replica's I/O thread can continue running. The STOP REPLICA and START REPLICA statements support the `SQL_THREAD` option for this scenario. For example:

```
STOP SLAVE SQL_THREAD;
CHANGE MASTER TO
   RELAY_LOG_FILE='slave-relay-bin.006',
   RELAY_LOG_POS=4025;
START SLAVE SQL_THREAD;
```

When the value of this option is changed, the metadata about the replica's SQL thread's position in the relay logs will also be changed in the `relay-log.info` file or the file that is configured by the relay_log_info_file system variable.

> **Note:**
>
> The RELAY_LOG_FILE and RELAY_LOG_POS options cannot be specified if the MASTER_LOG_FILE and MASTER_LOG_POS options were also specified.

RELAY_LOG_POS

The `RELAY_LOG_POS` option for `CHANGE MASTER` can be used along with the RELAY_LOG_FILE option to specify the coordinates at which the replica's SQL thread should begin reading from the relay log the next time the thread starts.

The `CHANGE MASTER` statement usually deletes all relay log files. However, if the `RELAY_LOG_FILE` and/or `RELAY_LOG_POS` options are specified, then existing relay log files are kept.

When you want to change the relay log position, you only need to stop the replica's SQL thread. The replica's I/O thread can continue running. The STOP REPLICA and START REPLICA statements support the `SQL_THREAD` option for this scenario. For example:

```
STOP SLAVE SQL_THREAD;
CHANGE MASTER TO
   RELAY_LOG_FILE='slave-relay-bin.006',
   RELAY_LOG_POS=4025;
START SLAVE SQL_THREAD;
```

When the value of this option is changed, the metadata about the replica's SQL thread's position in the relay logs will also be changed in the `relay-log.info` file or the file that is configured by the relay_log_info_file system variable.

> **Note:**
>
> The RELAY_LOG_FILE and RELAY_LOG_POS options cannot be specified if the MASTER_LOG_FILE and MASTER_LOG_POS options were also specified.

GTID Options

MASTER_USE_GTID

CURRENT

The `MASTER_USE_GTID` option for `CHANGE MASTER` can be used to configure the replica to use the global transaction ID (GTID) when connecting to a primary. The possible values are:

- `current_pos` - Replicate in GTID mode and use gtid_current_pos as the position to start downloading transactions from the primary. Using this on a replica server can break replication if the replica executes local transactions due to actively updating gtid_current_pos with gtid_binlog_pos and gtid_slave_pos. Use the new, safe, MASTER_DEMOTE_TO_SLAVE= option instead.
- `slave_pos` - Replicate in GTID mode and use gtid_slave_pos as the position to start downloading transactions from the primary. `replica_pos` is an alias for `slave_pos`.
- `no` - Don't replicate in GTID mode.

< 10.5.1

The `MASTER_USE_GTID` option for `CHANGE MASTER` can be used to configure the replica to use the global transaction ID (GTID) when connecting to a primary. The possible values are:

- `current_pos` - Replicate in GTID mode and use gtid_current_pos as the position to start downloading transactions from the primary. Using this on a replica server can break replication if the replica executes local transactions due to actively updating gtid_current_pos with gtid_binlog_pos and gtid_slave_pos. Use the new, safe, MASTER_DEMOTE_TO_SLAVE= option instead.
- `slave_pos` - Replicate in GTID mode and use gtid_slave_pos as the position to start downloading transactions from the primary.
- `no` - Don't replicate in GTID mode.

The `MASTER_USE_GTID` option for `CHANGE MASTER` can be used to configure the replica to use the global transaction ID (GTID) when connecting to a primary. The possible values are:

- `current_pos` - Replicate in GTID mode and use gtid_current_pos as the position to start downloading transactions from the primary. Using this on a replica server can break replication if the replica executes local transactions due to actively updating gtid_current_pos with gtid_binlog_pos and gtid_slave_pos. Use the new, safe, MASTER_DEMOTE_TO_SLAVE= option instead.
- `slave_pos` - Replicate in GTID mode and use gtid_slave_pos as the position to start downloading transactions from the primary. From MariaDB 10.5.1, `replica_pos` is an alias for `slave_pos`.

- `no` - Don't replicate in GTID mode.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_USE_GTID = current_pos;
START SLAVE;
```

Or:

```
STOP SLAVE;
SET GLOBAL gtid_slave_pos='0-1-153';
CHANGE MASTER TO
   MASTER_USE_GTID = slave_pos;
START SLAVE;
```

> Starting with MariaDB 12.3, this option also accepts the `DEFAULT` keyword. When set to `DEFAULT`, the value is derived from the corresponding server option (for example, `--master-use-gtid`).

MASTER_DEMOTE_TO_SLAVE

> **CURRENT**
>
> Used to transition a primary to become a replica. Replaces the old MASTER_USE_GTID=current_pos with a safe alternative by forcing users to set `Using_Gtid=Slave_Pos` and merging `gtid_binlog_pos` into `gtid_slave_pos` once at `CHANGE MASTER TO` time. If `gtid_slave_pos` is more recent than `gtid_binlog_pos` (as in the case of chain replication), the replication state should be preserved.
>
> For example:
>
> ```
> STOP SLAVE;
> CHANGE MASTER TO
>    MASTER_DEMOTE_TO_SLAVE = 1;
> START SLAVE;
> ```
>
> **< 10.10**
>
> `MASTER_DEMOTE_TO_SLAVE` is not available.

Replication Filter Options

Also see Replication filters.

IGNORE_SERVER_IDS

The `IGNORE_SERVER_IDS` option for `CHANGE MASTER` can be used to configure a replica to ignore binary log events that originated from certain servers. Filtered binary log events will not get logged to the replica's relay log, and they will not be applied by the replica.

The option's value can be specified by providing a comma-separated list of server_id values. For example:

```
STOP SLAVE;
CHANGE MASTER TO
   IGNORE_SERVER_IDS = (3,5);
START SLAVE;
```

If you would like to clear a previously set list, then you can set the value to an empty list. For example:

```
STOP SLAVE;
CHANGE MASTER TO
   IGNORE_SERVER_IDS = ();
START SLAVE;
```

DO_DOMAIN_IDS

The `DO_DOMAIN_IDS` option for `CHANGE MASTER` can be used to configure a replica to only apply binary log events if the transaction's GTID is in a specific gtid_domain_id value. Filtered binary log events will not get logged to the replica's relay log, and they will not be applied by the replica.

The option's value can be specified by providing a comma-separated list of gtid_domain_id values. Duplicate values are automatically ignored. For example:

```
STOP SLAVE;
CHANGE MASTER TO
   DO_DOMAIN_IDS = (1,2);
START SLAVE;
```

If you would like to clear a previously set list, then you can set the value to an empty list. For example:

```
STOP SLAVE;
CHANGE MASTER TO
   DO_DOMAIN_IDS = ();
START SLAVE;
```

Note:

The DO_DOMAIN_IDS option and the IGNORE_DOMAIN_IDS option cannot both be set to non-empty values at the same time. If you want to set the DO_DOMAIN_IDS option, and the IGNORE_DOMAIN_IDS option was previously set, then you need to clear the value of the IGNORE_DOMAIN_IDS option. For example:

```
STOP SLAVE;
CHANGE MASTER TO
   IGNORE_DOMAIN_IDS = (),
   DO_DOMAIN_IDS = (1,2);
START SLAVE;
```

Note:

The `DO_DOMAIN_IDS` option can only be specified if the replica is replicating in GTID mode. Therefore, the MASTER_USE_GTID option must also be set to some value other than `no` in order to use this option.

IGNORE_DOMAIN_IDS

The `IGNORE_DOMAIN_IDS` option for `CHANGE MASTER` can be used to configure a replica to ignore binary log events if the transaction's GTID is in a specific gtid_domain_id value. Filtered binary log events will not get logged to the replica's relay log, and they will not be applied by the replica.

The option's value can be specified by providing a comma-separated list of gtid_domain_id values. Duplicate values are automatically ignored. For example:

```
STOP SLAVE;
CHANGE MASTER TO
   IGNORE_DOMAIN_IDS = (1,2);
START SLAVE;
```

If you would like to clear a previously set list, then you can set the value to an empty list. For example:

```
STOP SLAVE;
CHANGE MASTER TO
   IGNORE_DOMAIN_IDS = ();
START SLAVE;
```

Note:

The DO_DOMAIN_IDS option and the IGNORE_DOMAIN_IDS option cannot both be set to non-empty values at the same time. If you want to set the IGNORE_DOMAIN_IDS option, and the DO_DOMAIN_IDS option was previously set, then you need to clear the value of the DO_DOMAIN_IDS option. For example:

```
STOP SLAVE;
CHANGE MASTER TO
   DO_DOMAIN_IDS = (),
   IGNORE_DOMAIN_IDS = (1,2);
START SLAVE;
```

Note:

The `IGNORE_DOMAIN_IDS` option can only be specified if the replica is replicating in GTID mode. Therefore, the MASTER_USE_GTID option must also be set to some value other than `no` in order to use this option.

Delayed Replication Options

MASTER_DELAY

The `MASTER_DELAY` option for `CHANGE MASTER` can be used to enable delayed replication. This option specifies the time in seconds (at least) that a replica should lag behind the primary up to a maximum value of 2147483647, or about 68 years. Before executing an event, the replica will first wait, if necessary, until the given time has passed since the event was created on the primary. The result is that the replica will reflect the state of the primary some time back in the past. The default is zero, no delay.

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_DELAY=3600;
START SLAVE;
```

Changing Option Values

If you don't specify a given option when executing the `CHANGE MASTER` statement, then the option keeps its old value in most cases. Most of the time, there is no need to specify the options that do not need to change. For example, if the password for the user account that the replica uses to connect to its primary has changed, but no other options need to change, then you can just change the MASTER_PASSWORD option by executing the following commands:

```
STOP SLAVE;
CHANGE MASTER TO
   MASTER_PASSWORD='new3cret';
START SLAVE;
```

There are some cases where options are implicitly reset, such as when the MASTER_HOST and MASTER_PORT options are changed.

Option Persistence

The values of the MASTER_LOG_FILE and MASTER_LOG_POS options (i.e. the binary log position on the primary) and most other options are written to either the default `master.info` file or the file that is configured by the

master_info_file option. The replica's I/O thread keeps this binary log position updated as it downloads events only when MASTER_USE_GTID optionis set to `NO`. Otherwise the file is not updated on a per event basis.

The master_info_file option can be set either on the command-line or in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
master_info_file=/mariadb/myserver1-master.info
```

The values of the RELAY_LOG_FILE and RELAY_LOG_POS options (i.e. the relay log position) are written to either the default `relay-log.info` file or the file that is configured by the relay_log_info_file system variable. The replica's SQL thread keeps this relay log position updated as it applies events.

The relay_log_info_file system variable can be set either on the command-line or in a server option group in an option file prior to starting up the server. For example:

```
[mariadb]
...
relay_log_info_file=/mariadb/myserver1-relay-log.info
```

GTID Persistence

If the replica is replicating binary log events that contain GTIDs, then the replica's SQL thread will write every GTID that it applies to the mysql.gtid_slave_pos table. This GTID can be inspected and modified through the gtid_slave_pos system variable.

If the replica has the log_slave_updates system variable enabled and if the replica has the binary log enabled, then every write by the replica's SQL thread will also go into the replica's binary log. This means that GTIDs of replicated transactions would be reflected in the value of the gtid_binlog_pos system variable.

Creating a replica from a backup

The `CHANGE MASTER` statement is useful for setting up a replica when you have a backup of the primary and you also have the binary log position or GTID position corresponding to the backup.

After restoring the backup on the replica, you could execute something like this to use the binary log position:

```
CHANGE MASTER TO
   MASTER_LOG_FILE='master2-bin.001',
   MASTER_LOG_POS=4;
START SLAVE;
```

Or you could execute something like this to use the GTID position:

```
SET GLOBAL gtid_slave_pos='0-1-153';
CHANGE MASTER TO
   MASTER_USE_GTID=slave_pos;
START SLAVE;
```

See Setting up a Replication Replica with mariadb-backup for more information on how to do this with mariadb-backup.

Example

The following example changes the primary and primary's binary log coordinates. This is used when you want to set up the replica to replicate the primary:

```
CHANGE MASTER TO
   MASTER_HOST='master2.mycompany.com',
   MASTER_USER='replication',
   MASTER_PASSWORD='bigs3cret',
   MASTER_PORT=3306,
   MASTER_LOG_FILE='master2-bin.001',
   MASTER_LOG_POS=4,
   MASTER_CONNECT_RETRY=10;
START SLAVE;
```

See Also

- Setting up replication
- START REPLICA
- Multi-source replication
- RESET REPLICA. Removes a connection created with `CHANGE MASTER TO`.
- Global Transaction ID

This page is licensed: GPLv2, originally from fill_help_tables.sql

RESET MASTER

```
RESET MASTER [TO #]
```

Deletes all binary log files listed in the index file, resets the binary log index file to be empty, and creates a new binary log file with a suffix of .000001.

If `TO #` is given, then the first new binary log file will start from number #.

This statement is for use only when the master is started for the first time and should never be used if any slaves are actively replicating from the binary log.

See Also

- The PURGE BINARY LOGS statement is intended for use in active replication.

This page is licensed: GPLv2, originally from fill_help_tables.sql

RESET REPLICA

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Syntax

```
RESET { SLAVE | REPLICA } ["connection_name"] [ALL]  [FOR CHANNEL "connection_name"].
```

Description

RESET REPLICA makes the replica forget its replication position in the master's binary log. This statement is meant to be used for a clean start. It deletes the master.info and relay-log.info files, all the relay log files, and starts a new relay log file. To use RESET REPLICA, the replica threads must be stopped (use STOP REPLICA if necessary).

Note: All relay log files are deleted, even if they have not been completely executed by the replica SQL thread. (This is a condition likely to exist on a replication replica if you have issued a STOP REPLICA statement or if the replica is

highly loaded.)

Note: `RESET REPLICA` does not reset the global `gtid_slave_pos` variable. This means that a replica server configured with `CHANGE MASTER TO MASTER_USE_GTID=slave_pos` will not receive events with GTIDs occurring before the state saved in `gtid_slave_pos`. If the intent is to reprocess these events, `gtid_slave_pos` must be manually reset, e.g., by executing `set global gtid_slave_pos=""`.

Connection information stored in the master.info file is immediately reset using any values specified in the corresponding startup options. This information includes values such as master host, master port, master user, and master password. If the replica SQL thread was in the middle of replicating temporary tables when it was stopped, and RESET REPLICA is issued, these replicated temporary tables are deleted on the replica.

The `ALL` also resets the `PORT`, `HOST`, `USER`, and `PASSWORD` parameters for the replica. If you are using a connection name, it will be permanently deleted it and it will not show up anymore in SHOW ALL REPLICAS STATUS.

connection_name

The `connection_name` option is used for multi-source replication.

If there is only one nameless primary, or the default primary (as specified by the default_master_connection system variable) is intended, `connection_name` can be omitted. If provided, the `RESET REPLICA` statement will apply to the specified primary. `connection_name` is case-insensitive.

MariaDB starting with 10.7.0

> **CURRENT**
>
> The `FOR CHANNEL` keyword was added for MySQL compatibility. This is identical to using the channel_name directly after `RESET REPLICA`.
>
> **< 10.7.0**
>
> `FOR CHANNEL` is not available.

The `FOR CHANNEL` keyword was added for MySQL compatibility. This is identical as using the channel_name directly after `RESET REPLICA`.

MariaDB starting with 11.6.0

> **CURRENT**
>
> `RESET REPLICA` resets the `Master` / `Slave_last_event_time` and `Connects_Tried` values (see SHOW REPLICA STATUS).
>
> **< 12.0**
>
> `RESET REPLICA` resets the `Master` / `Slave_last_event_time` values (see SHOW REPLICA STATUS).
>
> **< 11.6**
>
> `RESET REPLICA` does not reset the `Master` / `Slave_last_event_time` values (see SHOW REPLICA STATUS).

See Also

- STOP REPLICA stops the replica, but it can be restarted with START REPLICA or after next MariaDB server restart.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SET GLOBAL SQL_SLAVE_SKIP_COUNTER

Syntax

```
SET GLOBAL sql_slave_skip_counter = N
```

Description

This statement skips the next `N` events from the primary. This is useful for recovering from replication stops caused by a statement.

If multi-source replication is used, this statement applies to the default connection. It could be necessary to change the value of the default_master_connection system variable.

Note that, if the event is a transaction, the whole transaction will be skipped. With non-transactional engines, an event is always a single statement.

This statement is valid only when the replica threads are not running. Otherwise, it produces an error.

The statement does not automatically restart the replica threads.

Example

```
SHOW SLAVE STATUS \G
...
SET GLOBAL sql_slave_skip_counter = 1;
START SLAVE;
```

Multi-source replication:

```
SET @@default_master_connection = 'master_01';
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;
START SLAVE;
```

Multiple Replication Domains

`sql_slave_skip_counter` can't be used to skip transactions on a replica if GTID replication is in use and if gtid_slave_pos contains multiple gtid_domain_id values. In that case, you'll get an error like the following:

```
ERROR 1966 (HY000): When using parallel replication and GTID with multiple
 replication domains, @@sql_slave_skip_counter can not be used. Instead,
 setting @@gtid_slave_pos explicitly can be  used to skip to after a given GTID
 position.
```

In order to skip transactions in cases like this, you will have to manually change gtid_slave_pos.

See Also

- Selectively Skipping Replication of Binlog Events

This page is licensed: GPLv2, originally from fill_help_tables.sql

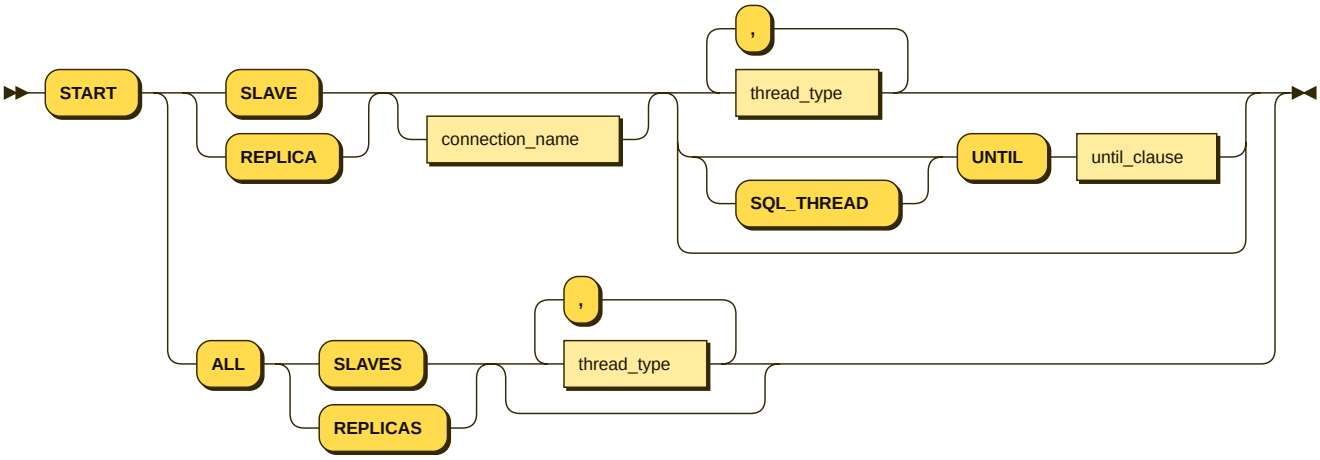
START REPLICA

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

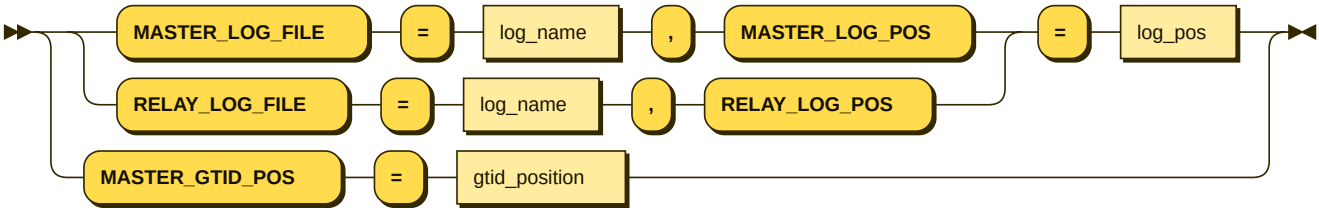
Syntax

```
START { SLAVE | REPLICA } ["connection_name"] [thread_type [, thread_type] ... ]
START { SLAVE | REPLICA } ["connection_name"] [SQL_THREAD] UNTIL
    MASTER_LOG_FILE = 'log_name', MASTER_LOG_POS = log_pos
START { SLAVE | REPLICA } ["connection_name"] [SQL_THREAD] UNTIL
    RELAY_LOG_FILE = 'log_name', RELAY_LOG_POS = log_pos
START { SLAVE | REPLICA } ["connection_name"] [SQL_THREAD] UNTIL
    MASTER_GTID_POS = <GTID position>
START ALL { SLAVES | REPLICAS } [thread_type [, thread_type]]

thread_type: IO_THREAD | SQL_THREAD
```



Railroad diagram of START REPLICA — equivalent to the BNF above



Railroad diagram of until_clause

Railroad diagram of thread_type

Description

> **CURRENT**
>
> `START REPLICA` is a synonym for `START SLAVE`.
>
> **< 10.5.1**
>
> Only `START SLAVE` can be used.

`START SLAVE` or `START REPLICA` with no thread_type options starts both of the replica threads (see replication) needed to connect with a primary setup with CHANGE MASTER TO. The I/O thread reads events from the primary server and stores them in the relay log. The SQL thread reads events from the relay log and executes them.

> **CURRENT**

`START SLAVE` requires the REPLICATION SLAVE ADMIN privilege.

< 10.5.2

`START SLAVE` requires the SUPER privilege.

If `START SLAVE` succeeds in starting the replica threads, it returns without any error. However, even in that case, it might be that the replica threads start and then later stop (for example, because they do not manage to connect to the primary or read its binary log, or some other problem). `START SLAVE` does not warn you about this. You must check the replica's error log for error messages generated by the replica threads or check that they are running satisfactorily with SHOW REPLICA STATUS (or SHOW REPLICA STATUS).

START REPLICA UNTIL

`START SLAVE UNTIL` refers to the `SQL_THREAD` replication position at which the `SQL_THREAD` replication will halt. If `SQL_THREAD` isn't specified, both threads are started.

`START SLAVE UNTIL master_gtid_pos=xxx` is also supported. See Global Transaction ID/START REPLICA UNTIL master_gtid_pos=xxx for more details.

MariaDB starting with 11.3.0

CURRENT

SQL_BEFORE_GTIDS|SQL_AFTER_GTIDS

The `START SLAVE UNTIL` statement contains the options `SQL_BEFORE_GTIDS` and `SQL_AFTER_GTIDS` to allow control of whether the replica stops before or after a provided GTID state. Its syntax is:

```
START SLAVE UNTIL (SQL_BEFORE_GTIDS|SQL_AFTER_GTIDS)="<gtid_list>"
```

See Global Transaction ID#SQL_BEFORE_GTIDS/SQL_AFTER_GTIDS for details.

< 11.3

`SQL_BEFORE_GTIDS` and `SQL_AFTER_GTIDS` are not available.

connection_name

If there is only one nameless primary, or the default primary (as specified by the default_master_connection system variable) is intended, `connection_name` can be omitted. If provided, the `START SLAVE` statement will apply to the specified primary. `connection_name` is case-insensitive.

CURRENT

The `FOR CHANNEL` keyword is available for MySQL compatibility. This is identical to using the channel_name directly after `START SLAVE`.

< 10.7.0

The `FOR CHANNEL` keyword is not available.

START ALL REPLICAS

`START ALL SLAVES` starts all configured replicas (replicas with master_host not empty) that were not started before. It will give a `note` for all started connections. You can check the notes with SHOW WARNINGS. `START ALL REPLICAS` is an alias for `START ALL SLAVES`.

See Also

- Setting up replication.
- CHANGE MASTER TO is used to create and change connections.
- STOP REPLICA is used to stop a running connection.
- RESET REPLICA is used to reset parameters for a connection and also to permanently delete a primary connection.

This page is licensed: GPLv2, originally from fill_help_tables.sql

STOP REPLICA

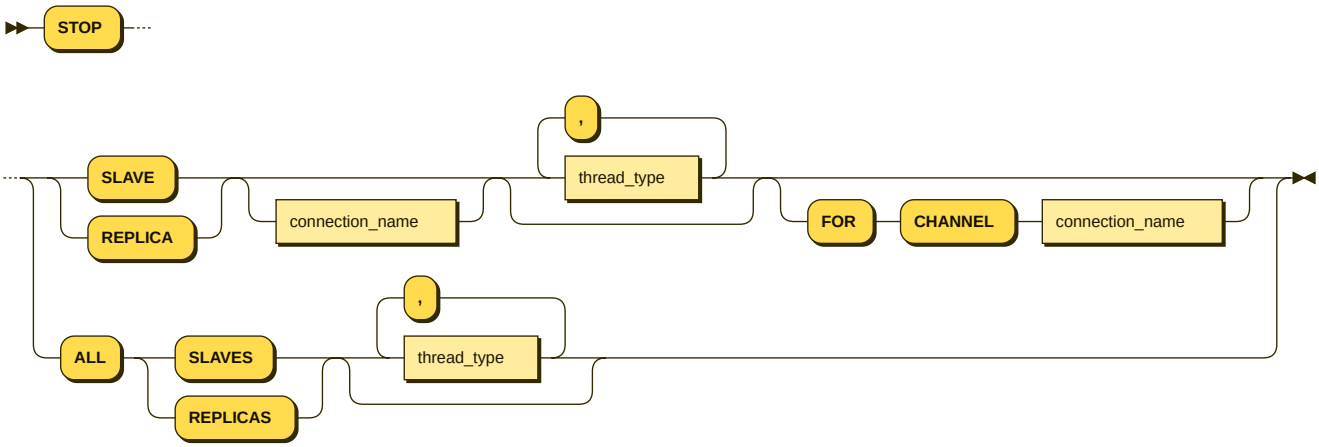
> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Syntax

```
STOP { SLAVE | REPLICA } ["connection_name"] [thread_type [, thread_type] ... ]
    [FOR CHANNEL "connection_name"]

STOP ALL { SLAVES | REPLICAS } [thread_type [, thread_type]]

thread_type: IO_THREAD | SQL_THREAD
```



Railroad diagram of STOP REPLICA — equivalent to the BNF above

Railroad diagram of thread_type

Description

Stops the replica threads. `STOP SLAVE` requires the SUPER privilege, or, from MariaDB 10.5.2, the REPLICATION SLAVE ADMIN privilege.

Like START REPLICA, this statement may be used with the `IO_THREAD` and `SQL_THREAD` options to name the thread or threads to be stopped. In almost all cases, one never need to use the `thread_type` options.

`STOP SLAVE` waits until any current replication event group affecting one or more non-transactional tables has finished executing (if there is any such replication group), or until the user issues a KILL QUERY or KILL CONNECTION statement.

Note that `STOP SLAVE` doesn't delete the connection permanently. Next time you execute START REPLICA or the MariaDB server restarts, the replica connection is restored with it's original arguments. If you want to delete a connection, you should execute RESET REPLICA.

STOP ALL REPLICAS

`STOP ALL SLAVES` stops all your running replicas. It will give you a `note` for every stopped connection. You can check the notes with SHOW WARNINGS. `STOP ALL REPLICAS` is an alias for `STOP ALL SLAVES`.

connection_name

The `connection_name` option is used for multi-source replication.

If there is only one nameless primary, or the default primary (as specified by the default_master_connection system variable) is intended, `connection_name` can be omitted. If provided, the `STOP SLAVE` statement will apply to the specified primary. `connection_name` is case-insensitive.

MariaDB starting with 10.7.0

> **CURRENT**
>
> The `FOR CHANNEL` keyword is available for MySQL compatibility. This is identical as using the channel_name directly after `STOP SLAVE`.
>
> **< 10.7.0**
>
> The `FOR CHANNEL` keyword is not available.

See Also

- CHANGE MASTER TO is used to create and change connections.
- START REPLICA is used to start a predefined connection.
- RESET REPLICA is used to reset parameters for a connection and also to permanently delete a primary connection.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SET Statements

See SET.

Assign values to different types of variables. Learn the syntax for setting user-defined variables, system variables, and stored program variables.

See SET CHARACTER SET.

Map strings to a specific character set. This command updates the character set for the client, results, and connection to ensure correct data encoding.

See SET GLOBAL SQL_SLAVE_SKIP_COUNTER.

SET GLOBAL sql_slave_skip_counter skips the next N events from the primary to recover from replication stops, and is valid only while the replica threads are stopped.

See SET NAMES.

Configure the character set and collation for the current connection. This ensures the server correctly interprets data sent by the client application.

See SET PASSWORD.

Complete reference for SET PASSWORD in MariaDB. Complete syntax guide with all options, clauses, and practical examples with comprehensive examples and best.

See SET PATH.

SET PATH sets the list of schemas MariaDB searches when invoking stored routines and packages, available from MariaDB 12.3 via the @@path system variable.

See SET SQL_LOG_BIN.

Enable or disable binary logging for the current session. This statement allows administrators to perform operations without replicating them to replicas.

See SET ROLE.

Sets the current role for the session. Learn how to enable none, or a specific role to change your current privileges dynamically.

See SET STATEMENT.

Set a system variable for the duration of a single query. This statement allows temporary configuration changes that apply only to the immediate statement.

See SET TRANSACTION.

Define isolation levels and access modes for transactions. Learn to configure the behavior of the next transaction or the entire session for data consistency.

See SET Variable.

Assign values to user-defined variables. This guide explains how to store data in session-specific variables for reuse in subsequent SQL statements.

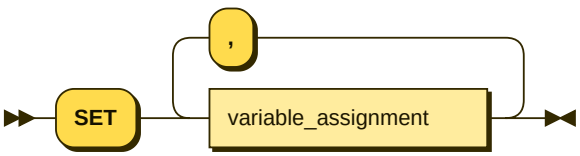
This page is licensed: CC BY-SA / Gnu FDL

SET

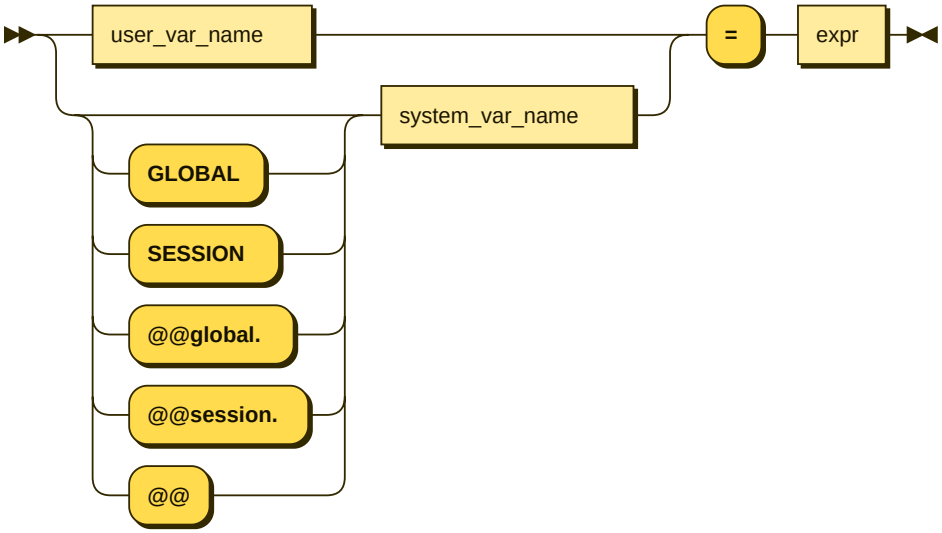
Syntax

```
SET variable_assignment [, variable_assignment] ...

variable_assignment:
      user_var_name = expr
    | [GLOBAL | SESSION] system_var_name = expr
    | [@@global. | @@session. | @@]system_var_name = expr
```



Railroad diagram of SET



Railroad diagram of variable_assignment

One can also set a user variable in any expression with this syntax:

```
user_var_name:= expr
```

Description

CURRENT

The `SET` statement assigns values to different types of variables that affect the operation of the server or your client.

< 10.0

The `SET` statement assigns values to different types of variables that affect the operation of the server or your client. Older versions of MySQL employed `SET OPTION`, but this syntax was deprecated in favor of `SET` without `OPTION`.

Changing a system variable by using the SET statement does not make the change permanently. To do so, the change must be made in a configuration file.

For setting variables on a per-query basis, see SET STATEMENT.

See SHOW VARIABLES for documentation on viewing server system variables.

See Server System Variables for a list of all the system variables.

GLOBAL / SESSION

When setting a system variable, the scope can be specified as either GLOBAL or SESSION.

A global variable change affects all new sessions. It does not affect any currently open sessions, including the one that made the change.

A session variable change affects the current session only.

If the variable has a session value, not specifying either GLOBAL or SESSION will be the same as specifying SESSION. If the variable only has a global value, not specifying GLOBAL or SESSION will apply to the change to the global value.

DEFAULT

Setting a global variable to DEFAULT will restore it to the server default, and setting a session variable to DEFAULT will restore it to the current global value.

Examples

- innodb_sync_spin_loops is a global variable.
- skip_parallel_replication is a session variable.
- max_error_count is both global and session.

```
SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
 INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
  VARIABLE_NAME IN ('max_error_count', 'skip_parallel_replication', 'innodb_sync_spin_loops');
+---------------------------+---------------+--------------+
| VARIABLE_NAME             | SESSION_VALUE | GLOBAL_VALUE |
+---------------------------+---------------+--------------+
| MAX_ERROR_COUNT           | 64            | 64           |
| SKIP_PARALLEL_REPLICATION | OFF           | NULL         |
| INNODB_SYNC_SPIN_LOOPS    | NULL          | 30           |
+---------------------------+---------------+--------------+
```

Setting the session values:

```
SET max_error_count=128;Query OK, 0 rows affected (0.000 sec)

SET skip_parallel_replication=ON;Query OK, 0 rows affected (0.000 sec)

SET innodb_sync_spin_loops=60;
ERROR 1229 (HY000): Variable 'innodb_sync_spin_loops' is a GLOBAL variable
  and should be set with SET GLOBAL

SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
 INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
  VARIABLE_NAME IN ('max_error_count', 'skip_parallel_replication', 'innodb_sync_spin_loops');
+---------------------------+---------------+--------------+
| VARIABLE_NAME             | SESSION_VALUE | GLOBAL_VALUE |
+---------------------------+---------------+--------------+
| MAX_ERROR_COUNT           | 128           | 64           |
| SKIP_PARALLEL_REPLICATION | ON            | NULL         |
| INNODB_SYNC_SPIN_LOOPS    | NULL          | 30           |
+---------------------------+---------------+--------------+
```

Setting the global values:

```
SET GLOBAL max_error_count=256;

SET GLOBAL skip_parallel_replication=ON;
ERROR 1228 (HY000): Variable 'skip_parallel_replication' is a SESSION variable
  and can't be used with SET GLOBAL

SET GLOBAL innodb_sync_spin_loops=120;

SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
 INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
  VARIABLE_NAME IN ('max_error_count', 'skip_parallel_replication', 'innodb_sync_spin_loops');
+---------------------------+---------------+--------------+
| VARIABLE_NAME             | SESSION_VALUE | GLOBAL_VALUE |
+---------------------------+---------------+--------------+
| MAX_ERROR_COUNT           | 128           | 256          |
| SKIP_PARALLEL_REPLICATION | ON            | NULL         |
| INNODB_SYNC_SPIN_LOOPS    | NULL          | 120          |
+---------------------------+---------------+--------------+
```

SHOW VARIABLES will by default return the session value unless the variable is global only.

```
SHOW VARIABLES LIKE 'max_error_count';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_error_count | 128   |
+-----------------+-------+

SHOW VARIABLES LIKE 'skip_parallel_replication';
+---------------------------+-------+
| Variable_name             | Value |
+---------------------------+-------+
| skip_parallel_replication | ON    |
+---------------------------+-------+

SHOW VARIABLES LIKE 'innodb_sync_spin_loops';
+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| innodb_sync_spin_loops | 120   |
+------------------------+-------+
```

Using the inplace syntax:

```
SELECT (@a:=1);
+---------+
| (@a:=1) |
+---------+
|       1 |
+---------+

SELECT @a;
+------+
| @a   |
+------+
|    1 |
+------+
```

See Also

- Using last_value() to return data of used rows
- SET STATEMENT
- SET Variable
- SET Data Type
- DECLARE Variable

This page is licensed: GPLv2, originally from fill_help_tables.sql

SET CHARACTER SET

Syntax

```
SET {CHARACTER SET | CHARSET}
    {charset_name | DEFAULT}
```

Description

Sets the character_set_client and character_set_results session system variables to the specified character set and collation_connection to the value of collation_database, which implicitly sets character_set_connection to the value of character_set_database.

This maps all strings sent between the current client and the server with the given mapping.

Example

```
SHOW VARIABLES LIKE 'character_set\_%';
+--------------------------+--------+
| Variable_name            | Value  |
+--------------------------+--------+
| character_set_client     | utf8   |
| character_set_connection | utf8   |
| character_set_database   | latin1 |
| character_set_filesystem | binary |
| character_set_results    | utf8   |
| character_set_server     | latin1 |
| character_set_system     | utf8   |
+--------------------------+--------+

SHOW VARIABLES LIKE 'collation%';
+----------------------+-------------------+
| Variable_name        | Value             |
+----------------------+-------------------+
| collation_connection | utf8_general_ci   |
| collation_database   | latin1_swedish_ci |
| collation_server     | latin1_swedish_ci |
+----------------------+-------------------+

SET CHARACTER SET utf8mb4;

SHOW VARIABLES LIKE 'character_set\_%';
+--------------------------+---------+
| Variable_name            | Value   |
+--------------------------+---------+
| character_set_client     | utf8mb4 |
| character_set_connection | latin1  |
| character_set_database   | latin1  |
| character_set_filesystem | binary  |
| character_set_results    | utf8mb4 |
| character_set_server     | latin1  |
| character_set_system     | utf8    |
+--------------------------+---------+

SHOW VARIABLES LIKE 'collation%';
+----------------------+-------------------+
| Variable_name        | Value             |
+----------------------+-------------------+
| collation_connection | latin1_swedish_ci |
| collation_database   | latin1_swedish_ci |
| collation_server     | latin1_swedish_ci |
+----------------------+-------------------+
```

See Also

- Setting Character Sets and Collations
- SET NAMES

This page is licensed: CC BY-SA / Gnu FDL

SET NAMES

Syntax

```
SET NAMES {'charset_name'
    [COLLATE 'collation_name'] | DEFAULT}
```

Description

Sets the character_set_client, character_set_connection, character_set_results and, implicitly, the collation_connection session system variables to the specified character set and collation.

This determines which character set the client will use to send statements to the server, and the server will use for sending results back to the client.

`ucs2`, `utf16`, `utf16le` and `utf32` are not valid character sets for `SET NAMES`, as they cannot be used as client character sets.

The collation clause is optional. If not defined (or if `DEFAULT` is specified), the default collation for the character set will be used.

Quotes are optional for the character set or collation clauses.

Examples

CURRENT

`utf8mb4` is the default for the affected variables:

```
SET NAMES DEFAULT;

SELECT VARIABLE_NAME, SESSION_VALUE
    FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
    VARIABLE_NAME LIKE 'character_set_con%' OR
    VARIABLE_NAME LIKE 'character_set_cl%' OR
    VARIABLE_NAME LIKE 'character_set_re%' OR
    VARIABLE_NAME LIKE 'collation_c%';
+--------------------------+----------------------+
| VARIABLE_NAME            | SESSION_VALUE        |
+--------------------------+----------------------+
| CHARACTER_SET_RESULTS    | utf8mb4              |
| CHARACTER_SET_CONNECTION | utf8mb4              |
| CHARACTER_SET_CLIENT     | utf8mb4              |
| COLLATION_CONNECTION     | utf8mb4_uca1400_ai_ci |
+--------------------------+----------------------+
```

< 11.8

The `utf8` character set (and related collations) is an alias for `utf8mb3`, rather than the other way around. MariaDB 11.2 added the character_set_collations variable, so the `SELECT` query is more specific in this example:

```
SELECT VARIABLE_NAME, SESSION_VALUE
    FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
    VARIABLE_NAME LIKE 'character_set_con%' OR
    VARIABLE_NAME LIKE 'character_set_cl%' OR
    VARIABLE_NAME LIKE 'character_set_re%' OR
    VARIABLE_NAME LIKE 'collation_c%';
+--------------------------+--------------------+
| VARIABLE_NAME            | SESSION_VALUE      |
+--------------------------+--------------------+
| CHARACTER_SET_RESULTS    | utf8mb3            |
| CHARACTER_SET_CONNECTION | utf8mb3            |
| CHARACTER_SET_CLIENT     | utf8mb3            |
| COLLATION_CONNECTION     | utf8mb3_general_ci |
+--------------------------+--------------------+

SET NAMES utf8mb4;

SELECT VARIABLE_NAME, SESSION_VALUE
    FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
    VARIABLE_NAME LIKE 'character_set_con%' OR
    VARIABLE_NAME LIKE 'character_set_cl%' OR
    VARIABLE_NAME LIKE 'character_set_re%' OR
    VARIABLE_NAME LIKE 'collation_c%';
+--------------------------+--------------------+
| VARIABLE_NAME            | SESSION_VALUE      |
+--------------------------+--------------------+
| CHARACTER_SET_RESULTS    | utf8mb4            |
| CHARACTER_SET_CONNECTION | utf8mb4            |
| CHARACTER_SET_CLIENT     | utf8mb4            |
| COLLATION_CONNECTION     | utf8mb4_general_ci |
+--------------------------+--------------------+
```

< 10.6

The utf8 character set (and related collation) is the default for the given variables:

```
SELECT VARIABLE_NAME, SESSION_VALUE
  FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
  VARIABLE_NAME LIKE 'character_set_c%' OR
  VARIABLE_NAME LIKE 'character_set_re%' OR
  VARIABLE_NAME LIKE 'collation_c%';
+--------------------------+-----------------+
| VARIABLE_NAME            | SESSION_VALUE   |
+--------------------------+-----------------+
| CHARACTER_SET_RESULTS    | utf8            |
| CHARACTER_SET_CONNECTION | utf8            |
| CHARACTER_SET_CLIENT     | utf8            |
| COLLATION_CONNECTION     | utf8_general_ci |
+--------------------------+-----------------+

SET NAMES big5;

SELECT VARIABLE_NAME, SESSION_VALUE
  FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
  VARIABLE_NAME LIKE 'character_set_c%' OR
  VARIABLE_NAME LIKE 'character_set_re%' OR
  VARIABLE_NAME LIKE 'collation_c%';
+--------------------------+-----------------+
| VARIABLE_NAME            | SESSION_VALUE   |
+--------------------------+-----------------+
| CHARACTER_SET_RESULTS    | big5            |
| CHARACTER_SET_CONNECTION | big5            |
| CHARACTER_SET_CLIENT     | big5            |
| COLLATION_CONNECTION     | big5_chinese_ci |
+--------------------------+-----------------+

SET NAMES 'latin1' COLLATE 'latin1_bin';

SELECT VARIABLE_NAME, SESSION_VALUE
  FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
  VARIABLE_NAME LIKE 'character_set_c%' OR
  VARIABLE_NAME LIKE 'character_set_re%' OR
  VARIABLE_NAME LIKE 'collation_c%';
+--------------------------+---------------+
| VARIABLE_NAME            | SESSION_VALUE |
+--------------------------+---------------+
| CHARACTER_SET_RESULTS    | latin1        |
| CHARACTER_SET_CONNECTION | latin1        |
| CHARACTER_SET_CLIENT     | latin1        |
| COLLATION_CONNECTION     | latin1_bin    |
+--------------------------+---------------+

SET NAMES DEFAULT;

SELECT VARIABLE_NAME, SESSION_VALUE
  FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
  VARIABLE_NAME LIKE 'character_set_c%' OR
  VARIABLE_NAME LIKE 'character_set_re%' OR
  VARIABLE_NAME LIKE 'collation_c%';
+--------------------------+-------------------+
| VARIABLE_NAME            | SESSION_VALUE     |
+--------------------------+-------------------+
| CHARACTER_SET_RESULTS    | latin1            |
| CHARACTER_SET_CONNECTION | latin1            |
| CHARACTER_SET_CLIENT     | latin1            |
| COLLATION_CONNECTION     | latin1_swedish_ci |
+--------------------------+-------------------+
```

See Also

- SET CHARACTER SET
- Setting Character Sets and Collations

- Character Sets and Collations

This page is licensed: CC BY-SA / Gnu FDL

SET PATH

> **Note:**
>
> This functionality is available from MariaDB 12.3.

Syntax

```
SET PATH 'db1 [, db2 [, ...]]'
```

Description

Sets the list of schemas (databases) that MariaDB uses to search for stored routines to invoke. Such a search happens in the `CALL` statement and when stored function are used in expressions.

The value of the path is a **string** which contains comma-separated schema names. A schema name can be quoted with backticks (or double quotes, as appropriate). A special name `CURRENT_SCHEMA` means to look in the current schema. Spaces are ignored, unless they are part of a quoted schema name. A path cannot contain two equivalent schema names.

The following two commands invoke the same stored procedure:

```
CALL sys.optimizer_switch_on();
```

and

```
SET PATH 'sys';
CALL optimizer_switch_on();
```

A default value of the path is `'CURRENT_SCHEMA'` which provides the traditional behavior — stored routines must be in the current database.

Also the path is used to look up packages in package routine invocations. Assuming there is a package `UTL_ENCODE` in the `sys` schema, and it contains the function `BASE64_DECODE()`, one can write

```
SELECT sys.UTL_ENCODE.BASE64_DECODE('data');
```

or

```
SET PATH='sys';
SELECT UTL_ENCODE.BASE64_DECODE('data');
```

Notes

- This functionality introduces the `@@path` system variable (session and global).
- The `SET PATH` statement sets the `@@session.path` variable. It is also possible to assign a new value to the `@@path` variable directory, using a statement like this: `SET @@path='sys'`.
- Prepared statements, stored routines, and views remember the value of the path when they are created. The value of the path at the time of the execution does not apply.

See Also

- Preparable statements

This page is licensed: CC BY-SA / Gnu FDL

SET SQL_LOG_BIN

Syntax

```
SET [SESSION] sql_log_bin = {0|1}
```

Description

Sets the sql_log_bin system variable, which disables or enables binary logging for the current connection, if the client has the `SUPER` privilege. The statement is refused with an error if the client does not have that privilege.

Note that setting `sql_log_bin=1` has no effect if log_bin variable, which enables global binary logging, is not set.

> **CURRENT**
>
> You cannot set `sql_log_bin` as a global variable.
>
> **< 5.6 / 5.5**
>
> You can set `sql_log_bin` as a global variable. This is considered dangerous, though, as it can damage replication.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SET STATEMENT

Syntax

```
SET STATEMENT var1=value1 [, var2=value2, ...]
  FOR <statement>
```

`var` is a system variable (list of allowed variables is provided below), and `value` is a constant literal.

Description

`SET STATEMENT var1=value1 FOR stmt`

is roughly equivalent to

```
SET @save_value=@@var1;
SET SESSION var1=value1;
stmt;
SET SESSION var1=@save_value;
```

The server parses the whole statement before executing it, so any variables set in this fashion that affect the parser may not have the expected effect. Examples include the charset variables, sql_mode=ansi_quotes, etc.

Examples

One can limit statement execution time max_statement_time:

```
SET STATEMENT max_statement_time=1000 FOR SELECT ... ;
```

One can switch on/off individual optimizations:

```
SET STATEMENT optimizer_switch='materialization=off' FOR SELECT ....;
```

It is possible to enable MRR/BKA for a query:

```
SET STATEMENT  join_cache_level=6, optimizer_switch='mrr=on'  FOR SELECT ...
```

Note that it makes no sense to try to set a session variable inside a `SET STATEMENT`:

```
#USELESS STATEMENT
SET STATEMENT sort_buffer_size = 100000 FOR SET SESSION sort_buffer_size = 200000;
```

For the above, after setting sort_buffer_size to 200000 it will be reset to its original state (the state before the `SET STATEMENT` started) after the statement execution.

Limitations

There are a number of variables that cannot be set on per-query basis. These include:

- `autocommit`
- `character_set_client`
- `character_set_connection`
- `character_set_filesystem`
- `collation_connection`
- `debug_sync`
- `interactive_timeout`
- `profiling`
- `profiling_history_size`
- `query_cache_type`
- `skip_replication`
- `sql_log_off`
- `wait_timeout`

Source

- The feature was originally implemented as a Google Summer of Code 2009 project by Joseph Lukas.
- Percona Server 5.6 included it as Per-query variable statement
- MariaDB ported the patch and fixed many bugs. The task in MariaDB Jira is MDEV-5231.

This page is licensed: CC BY-SA / Gnu FDL

SET TRANSACTION

Syntax

```
SET [GLOBAL | SESSION] TRANSACTION
    transaction_property [, transaction_property] ...

transaction_property:
    ISOLATION LEVEL level
  | READ WRITE
  | READ ONLY

level:
     REPEATABLE READ
   | READ COMMITTED
   | READ UNCOMMITTED
   | SERIALIZABLE
```

Overview

This statement sets the transaction isolation level or the transaction access mode globally, for the current session, or for the next transaction:

- With the `GLOBAL` keyword, the statement sets the default transaction level globally for all subsequent sessions. Existing sessions are unaffected.
- With the `SESSION` keyword, the statement sets the default transaction level for all subsequent transactions performed within the current session.
- Without any `SESSION` or `GLOBAL` keyword, the statement sets the isolation level for only the next (not started) transaction performed within the current session. After that it reverts to using the session value.

A change to the global default isolation level requires the SUPER privilege. Any session is free to change its session isolation level (even in the middle of a transaction), or the isolation level for its next transaction.

Isolation Level

To set the global default isolation level at server startup, use the –transaction-isolation=level option on the command line or in an option file. Values of level for this option use dashes rather than spaces, so the allowable values are READ_UNCOMMITTED,READ-COMMITTED, REPEATABLE-READ, or SERIALIZABLE. For example, to set the default isolation level to `REPEATABLE READ`, use these lines in the `[mariadb]` section of an option file:

```
[mariadb]
transaction-isolation = REPEATABLE-READ
```

CURRENT

To determine the global and session transaction isolation levels at runtime, check the value of the transaction_isolation variable.

< 11.1.1

To determine the global and session transaction isolation levels at runtime, check the value of the tx_isolation system variable.

```
SELECT @@GLOBAL.transaction_isolation, @@tx_isolation;
```

InnoDB supports each of the translation isolation levels described here using different locking strategies. The default level is `REPEATABLE READ`. For additional information about InnoDB record-level locks and how it uses them to execute various types of statements, see InnoDB Lock Modes, and innodb-locks-set.html.

Isolation Levels

The following sections describe how MariaDB supports the different transaction levels.

> **Warning:**
>
> With both `READ UNCOMMITTED` and `READ COMMITTED` isolation levels, you can't expect results to be deterministic between successive statements of the same transaction.

READ UNCOMMITTED

`SELECT` statements are performed in a non-locking fashion, but a possible earlier version of a row might be used. Thus, using this isolation level, such reads are not consistent. This is also called a "dirty read". Otherwise, this isolation level works like `READ COMMITTED`.

Warning:

With both `READ UNCOMMITTED` and `READ COMMITTED` isolation levels, you can't expect results to be deterministic between successive statements of the same transaction.

READ COMMITTED

A somewhat Oracle-like isolation level with respect to consistent (non-locking) reads: Each consistent read, even within the same transaction, sets and reads its own fresh snapshot. See innodb-consistent-read.html.

For locking reads (`SELECT` with `FOR UPDATE` or `LOCK IN SHARE MODE`), InnoDB locks only index records, not the gaps before them, and thus allows the free insertion of new records next to locked records. For `UPDATE` and `DELETE` statements, locking depends on whether the statement uses a unique index with a unique search condition (such as `WHERE id = 100`), or a range-type search condition (such as `WHERE id > 100`). For a unique index with a unique search condition, InnoDB locks only the index record found, not the gap before it. For range-type searches, InnoDB locks the index range scanned, using gap locks or next-key (gap plus index-record) locks to block insertions by other sessions into the gaps covered by the range. This is necessary because "phantom rows" must be blocked for MariaDB replication and recovery to work.

Note:

If the `READ COMMITTED` isolation level is used or the innodb_locks_unsafe_for_binlog system variable is enabled, there is no InnoDB gap locking except for foreign-key constraint checking and duplicate-key checking. Also, record locks for non-matching rows are released after MariaDB has evaluated the `WHERE` condition. If you use `READ COMMITTED` or enable `innodb_locks_unsafe_for_binlog` , you must use row-based binary logging.

Note:

Rows that don't match are not being locked in a so called semiconsistent read. This means you might see only a partially consistent read when the transaction isolation level is `READ COMMITTED` or `READ UNCOMMITTED` .

(A semiconsistent read applies to `UPDATE` and `DELETE` statements. Those statements skip locked rows, provided the version in the current read does not match the `WHERE` condition. Also, if the latest version of a record was successfully locked, but found not to match the condition, the lock is released.)

REPEATABLE READ

This is the default isolation level for InnoDB. For consistent reads, there is an important difference from the `READ COMMITTED` isolation level: All consistent reads within the same transaction read the snapshot established by the first read. This convention means that if you issue several plain (non-locking) `SELECT` statements within the same transaction, these `SELECT` statements are consistent also with respect to each other. See innodb-consistent-read.html.

CURRENT

For locking reads (`SELECT` with `FOR UPDATE` or `LOCK IN SHARE MODE`), `UPDATE` , and `DELETE` statements, locking depends on whether the statement uses a unique index with a unique search condition, or a range-type search condition. MariaDB does not relax the gap locking for unique indexes.

< 11.0 / 10.11.2 / 10.10.3 / 10.9.5 / 10.8.7 / 10.7.8 / 10.6.12 / 10.5.19 / 10.4.28 / 10.3.38

For locking reads (`SELECT` with `FOR UPDATE` or `LOCK IN SHARE MODE`), `UPDATE` , and `DELETE` statements, locking depends on whether the statement uses a unique index with a unique search condition, or a range-type search condition. For a unique index with a unique search condition, InnoDB locks only the index record found, not the gap before it.

For other search conditions, InnoDB locks the index range scanned, using gap locks or next-key (gap plus index-record) locks to block insertions by other sessions into the gaps covered by the range.

This is the minimum isolation level for non-distributed XA transactions.

Snapshot Isolation and DML Operations

> **Note:**
>
> This behavior is available from MariaDB 11.6.2 and 12.3.

The `innodb_snapshot_isolation` system variable is enabled by default. This introduces a stricter enforcement of `REPEATABLE READ` for `UPDATE` and `DELETE` statements:

- **Conflict Detection:** If an `UPDATE` or `DELETE` attempts to modify a row that has been changed by a concurrent transaction since your snapshot was established, the operation is rejected.
- **ER_CHECKREAD (1020):** This rejection triggers error `ER_CHECKREAD`. The revised error message suggests that the user should try restarting the transaction.
- **Automatic Rollback:** Unlike a simple statement error, `ER_CHECKREAD` is treated similarly to a deadlock: **the entire transaction is rolled back**.
- **Purpose:** This prevents the transaction from switching to "current-read" mode for that row, which would otherwise allow the transaction to observe concurrent changes it did not make, violating the pure repeatable read invariant.

Traditional Locking Behavior

If `innodb_snapshot_isolation` is disabled (set to `OFF`), InnoDB follows traditional behavior where locking reads (`SELECT ... FOR UPDATE`), `UPDATE`, and `DELETE` statements read the latest committed version of rows. In this mode, subsequent non-locking `SELECT` statements for those same rows also return the current version rather than the snapshot version, which can lead to non-repeatable read anomalies.

SERIALIZABLE

This level is like `REPEATABLE READ`, but InnoDB implicitly converts all plain `SELECT` statements to SELECT … LOCK IN SHARE MODE if autocommit is disabled. If autocommit is enabled, the `SELECT` is its own transaction. It therefore is known to be read only and can be serialized if performed as a consistent (non-locking) read and need not block for other transactions. (This means that to force a plain `SELECT` to block if other transactions have modified the selected rows, you should disable autocommit.)

Distributed XA transactions should always use this isolation level.

innodb_snapshop_isolation

> **CURRENT**
>
> If the innodb_snapshot_isolation system variable is not set to `ON`, strictly-speaking anything other than `READ UNCOMMITTED` is not clearly defined. innodb_snapshot_isolation defaults to `OFF` for backwards compatibility. Setting to `ON` will result in attempts to acquire a lock on a record that does not exist in the current read view raising an error, and the transaction being rolled back.
>
> **< 11.4.2 / 11.2.4 / 11.1.15 / 11.0.6 / 10.11.8 / 10.6.18**
>
> If the innodb_snapshot_isolation system variable is not set to `ON`, strictly-speaking anything other than `READ UNCOMMITTED` is not clearly defined.

Access Mode

> **CURRENT**
>
> The access mode specifies whether the transaction is allowed to write data or not. By default, transactions are in `READ WRITE` mode (see the tx_read_only system variable). `READ ONLY` mode allows the storage engine to apply optimizations that cannot be used for transactions which write data. Note that, unlike the global read_only mode, the READ_ONLY ADMIN privilege doesn't allow writes, and DDL statements on temporary tables are not allowed either.
>
> **< 10.11.0**
>
> The access mode specifies whether the transaction is allowed to write data or not. By default, transactions are in `READ WRITE` mode (see the tx_read_only system variable). `READ ONLY` mode allows the storage engine to apply optimizations that cannot be used for transactions which write data. Note that, unlike the global read_only mode, the SUPER privilege doesn't allow writes, and DDL statements on temporary tables are not allowed either.

It is not permitted to specify both `READ WRITE` and `READ ONLY` in the same statement.

`READ WRITE` and `READ ONLY` can also be specified in the START TRANSACTION statement, in which case the specified mode is only valid for one transaction.

Examples

```
SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Attempting to set the isolation level within an existing transaction without specifying `GLOBAL` or `SESSION`.

```
START TRANSACTION;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
ERROR 1568 (25001): Transaction characteristics can't be changed while a transaction is in progress
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SET Variable

Syntax

```
SET var_name = expr [, var_name = expr] ...
```

Description

The `SET` statement in stored programs is an extended version of the general SET statement. Referenced variables may be ones declared inside a stored program, global system variables, or user-defined variables.

The `SET` statement in stored programs is implemented as part of the pre-existing SET syntax. This allows an extended syntax of `SET a=x, b=y, ...` where different variable types (locally declared variables, global and session server variables, user-defined variables) can be mixed. This also allows combinations of local variables and some options that make sense only for system variables; in that case, the options are recognized but ignored.

`SET` can be used with both local variables and user-defined variables.

When setting several variables using the columns returned by a query, SELECT INTO should be preferred.

To set many variables to the same value, the LAST_VALUE( ) function can be used.

Below is an example of how a user-defined variable may be set:

```
SET @x = 1;
```

See Also

- SET
- SET STATEMENT
- DECLARE Variable

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW Statements

- About SHOW
- Extended SHOW
- SHOW ANALYZE
- SHOW AUTHORS
- SHOW BINARY LOGS
- SHOW BINLOG EVENTS
- SHOW MASTER STATUS
- SHOW CHARACTER SET
- SHOW CLIENT_STATISTICS
- SHOW COLLATION
- SHOW COLUMNS
- SHOW CONTRIBUTORS
- SHOW CREATE DATABASE
- SHOW CREATE EVENT
- SHOW CREATE FUNCTION
- SHOW CREATE PACKAGE BODY
- SHOW CREATE PACKAGE
- SHOW CREATE PROCEDURE
- SHOW CREATE SEQUENCE
- SHOW CREATE SERVER
- SHOW CREATE TABLE
- SHOW CREATE TRIGGER
- SHOW CREATE USER
- SHOW CREATE VIEW
- SHOW DATABASES
- SHOW ENGINE INNODB STATUS
- SHOW ENGINE
- SHOW ENGINES
- SHOW ERRORS
- SHOW EVENTS
- SHOW EXPLAIN
- SHOW FUNCTION CODE
- SHOW FUNCTION STATUS
- SHOW GRANTS
- SHOW INDEX_STATISTICS

- SHOW INDEX
- SHOW LOCALES
- SHOW OPEN TABLES
- SHOW PACKAGE BODY STATUS
- SHOW PACKAGE STATUS
- SHOW PLUGINS SONAME
- SHOW PLUGINS
- SHOW PRIVILEGES
- SHOW PROCEDURE CODE
- SHOW PROCEDURE STATUS
- SHOW PROCESSLIST
- SHOW PROFILE
- SHOW PROFILES
- SHOW QUERY_RESPONSE_TIME
- SHOW RELAYLOG EVENTS
- SHOW REPLICA HOSTS
- SHOW REPLICA STATUS
- SHOW STATUS
- SHOW TABLE_STATISTICS
- SHOW TABLE STATUS
- SHOW TABLES
- SHOW TRIGGERS
- SHOW USER_STATISTICS
- SHOW USER_VARIABLES
- SHOW VARIABLES
- SHOW WARNINGS
- SHOW WSREP_MEMBERSHIP
- SHOW WSREP_STATUS

This page is licensed: CC BY-SA / Gnu FDL

About SHOW

`SHOW` provides information about various aspects of MariaDB Server. A list of the various `SHOW` statements is here.

The general syntax is:

```
SHOW LIKE 'pattern' | WHERE expr
```

The `LIKE` and `WHERE` clauses make sense only for particular `SHOW` statements, though. See Extended SHOW for what `SHOW` statements benefit from using those clauses.

If the syntax for a given `SHOW` statement includes `LIKE 'pattern'` , `'pattern'` is a string that can contain the SQL `%` and `_` wildcard characters. The pattern is useful for restricting statement output to matching values.

This page is licensed: GPLv2, originally from fill_help_tables.sql

Extended SHOW

The following SHOW statements can be extended using a `WHERE` clause and a `LIKE` clause to refine the results:

- SHOW CHARACTER SET
- SHOW COLLATION
- SHOW COLUMNS
- SHOW DATABASES
- SHOW FUNCTION STATUS
- SHOW INDEX``
- SHOW OPEN TABLES
- SHOW PACKAGE STATUS
- SHOW PACKAGE BODY STATUS
- SHOW INDEX
- SHOW PROCEDURE STATUS
- SHOW STATUS
- SHOW TABLE STATUS
- SHOW TABLES
- SHOW TRIGGERS
- SHOW VARIABLES

As with a regular SELECT, the `WHERE` clause can be used for the specific columns returned, and the LIKE clause with the regular wildcards.

Examples

This statement shows all tables:

```
SHOW TABLES;
+----------------------+
| Tables_in_test       |
+----------------------+
| animal_count         |
| animals              |
| are_the_mooses_loose |
| aria_test2           |
| t1                   |
| view1                |
+----------------------+
```

This statement only shows tables starting with the letter ‘a’:

```
SHOW TABLES WHERE Tables_in_test LIKE 'a%';
+----------------------+
| Tables_in_test       |
+----------------------+
| animal_count         |
| animals              |
| are_the_mooses_loose |
| aria_test2           |
+----------------------+
```

This statement shows variables whose names start with *aria* and have a value greater than 8192:

```
SHOW VARIABLES WHERE Variable_name LIKE 'aria%' AND Value >8192;
+------------------------------+---------------------+
| Variable_name                | Value               |
+------------------------------+---------------------+
| aria_checkpoint_log_activity | 1048576             |
| aria_log_file_size           | 1073741824          |
| aria_max_sort_file_size      | 9223372036853727232 |
| aria_pagecache_buffer_size   | 134217728           |
| aria_sort_buffer_size        | 134217728           |
+------------------------------+---------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

SHOW ANALYZE

> **Note:**
>
> `SHOW ANALYZE` was added in MariaDB 10.9.

Syntax

```
SHOW ANALYZE [FORMAT=JSON] FOR <connection_id>;
```

Description

`SHOW ANALYZE` allows one to retrieve ANALYZE-like output from a currently running statement. The statement

```
SHOW ANALYZE [FORMAT=JSON] FOR <connection_id>;
```

connects to the query running in connection `connection_id`, gets information about the query plan it is executing, *also gets information about the runtime statistics of the execution so far* and returns it in a format similar to `ANALYZE [FORMAT=JSON]` output.

This is similar to the SHOW EXPLAIN command, the difference being that `SHOW ANALYZE` also produces runtime statistics information.

Use Case

You're trying to troubleshoot a query that never finishes. Since it doesn't finish, it is not possible to get ANALYZE output for it. With `SHOW ANALYZE`, you can get the runtime statistics without waiting for the query to finish.

Examples

Example 1: Row Counts

Consider the tables `orders` and `customer` and a join query finding the total amount of orders from customers with Gold status:

```
EXPLAIN format=json
SELECT sum(orders.amount)
FROM
  customer JOIN orders ON customer.cust_id=orders.cust_id
WHERE
  customer.status='GOLD';
```

The output of this query looks like this:

```
+------+-------------+----------+------+---------------+---------+---------+------------------+--------+-
------------+
| id   | select_type | table    | type | possible_keys | key     | key_len | ref              | rows   |
Extra       |
+------+-------------+----------+------+---------------+---------+---------+------------------+--------+-
------------+
|    1 | SIMPLE      | customer | ALL  | PRIMARY       | NULL    | NULL    | NULL             | 199786 |
Using where |
|    1 | SIMPLE      | orders   | ref  | cust_id       | cust_id | 5       | customer.cust_id | 1      |
|
+------+-------------+----------+------+---------------+---------+---------+------------------+--------+-
------------+
```

We run the SELECT, and it has been running for 30 seconds. Let's try `SHOW ANALYZE`:

```
SHOW ANALYZE format=json FOR 3;
| {
  "r_query_time_in_progress_ms": 32138,
```

The statement shows how long the query has been running.

```
"query_block": {
    "select_id": 1,
    "r_loops": 1,
    "nested_loop": [
      {
        "table": {
          "table_name": "customer",
          "access_type": "ALL",
          "possible_keys": ["PRIMARY"],
          "r_loops": 1,
          "rows": 199786,
          "r_rows": 110544,
```

`rows` shows the number of rows expected. `r_rows` in shows how many rows were processed so far (110K out of expected 200K). `r_loops` shows we're doing the first table scan (which is obvious for this query plan).

```
"filtered": 100,
          "r_filtered": 9.538283398,
          "attached_condition": "customer.`status` = 'GOLD'"
        }
      },
      {
        "table": {
          "table_name": "orders",
          "access_type": "ref",
          "possible_keys": ["cust_id"],
          "key": "cust_id",
          "key_length": "5",
          "used_key_parts": ["cust_id"],
          "ref": ["test.customer.cust_id"],
          "r_loops": 10544,
          "rows": 1,
          "r_rows": 99.99222307,
```

`rows: 1` shows the optimizer was expecting 1 order per customer. But `r_rows: 99.9` shows that it has found on average 100 orders per customer. This may be the reason the query is slower than expected.

The final chunk of the output doesn't have anything interesting but here it is:

```
"filtered": 100,
          "r_filtered": 100
        }
      }
    ]
  }
}
```

Example 2: Timing Information

Regular SELECT queries collect row count information, so `SHOW ANALYZE` can display it. However, detailed timing information is not collected, as collecting it may have CPU overhead. But if the target query is collecting timing information, `SHOW ANALYZE` will display it. How does one get the target query to collect timing information? Currently there is one way: if the target is running `ANALYZE`, it IS collecting timing information.
Re-running the previous example:

```
Connection 1> ANALYZE SELECT ... ;
```

```
Connection 2> SHOW ANALYZE FORMAT=JSON FOR <connection_id>;
ANALYZE
{
  "r_query_time_in_progress_ms": 30727,
  "query_block": {
    "select_id": 1,
    "r_loops": 1,
    "nested_loop": [
      {
        "table": {
          "table_name": "customer",
          "access_type": "ALL",
          "possible_keys": ["PRIMARY"],
          "r_loops": 1,
          "rows": 199786,
          "r_rows": 109994,
          "r_table_time_ms": 232.699,
          "r_other_time_ms": 46.355,
```

`ANALYZE` prints timing information in members named `r_..._time_ms`.
You can see that, so far, out of 30 seconds, only 232 millisecond were spent in reading the customer table. The bottleneck is elsewhere…

```
"filtered": 100,
          "r_filtered": 9.085950143,
          "attached_condition": "customer.`status` = 'GOLD'"
        }
      },
      {
        "table": {
          "table_name": "orders",
          "access_type": "ref",
          "possible_keys": ["cust_id"],
          "key": "cust_id",
          "key_length": "5",
          "used_key_parts": ["cust_id"],
          "ref": ["test.customer.cust_id"],
          "r_loops": 9994,
          "rows": 1,
          "r_rows": 99.99779868,
          "r_table_time_ms": 29460.609,
          "r_other_time_ms": 986.842,
```

29.4 seconds were spent reading the orders table (and 0.986 seconds in processing the obtained rows).
Now we can see where the query is spending time.

```
"filtered": 100,
          "r_filtered": 100
        }
      }
    ]
  }
}
```

See Also

- SHOW EXPLAIN command
- ANALYZE command
- MDEV-27021: Extend SHOW EXPLAIN to support SHOW ANALYZE [FORMAT=JSON]

This page is licensed: CC BY-SA / Gnu FDL

SHOW AUTHORS

Syntax

```
SHOW AUTHORS
```

Description

The `SHOW AUTHORS` statement displays information about the people who work on MariaDB. For each author, it displays Name, Location, and Comment values. All columns are encoded as latin1.

These include:

- First the active people in MariaDB are listed.
- Then the active people in MySQL.
- Last the people that have contributed to MariaDB/MySQL in the past.

The order is somewhat related to importance of the contribution given to the MariaDB project, but this is not 100% accurate. There is still room for improvement and debate…

Example

```
SHOW AUTHORS\G
*************************** 1. row ***************************
    Name: Michael (Monty) Widenius
Location: Tusby, Finland
 Comment: Lead developer and main author
*************************** 2. row ***************************
    Name: Sergei Golubchik
Location: Kerpen, Germany
 Comment: Architect, Full-text search, precision math, plugin framework, merges etc
*************************** 3. row ***************************
    Name: Igor Babaev
Location: Bellevue, USA
 Comment: Optimizer, keycache, core work
*************************** 4. row ***************************
    Name: Sergey Petrunia
Location: St. Petersburg, Russia
 Comment: Optimizer
*************************** 5. row ***************************
    Name: Oleksandr Byelkin
Location: Lugansk, Ukraine
 Comment: Query Cache (4.0), Subqueries (4.1), Views (5.0)
*************************** 6. row ***************************
    Name: Timour Katchaounov
Location: Sofia , Bulgaria
 Comment: Optimizer
*************************** 7. row ***************************
    Name: Kristian Nielsen
Location: Copenhagen, Denmark
 Comment: Replication, Async client prototocol, General buildbot stuff
*************************** 8. row ***************************
    Name: Alexander (Bar) Barkov
Location: Izhevsk, Russia
 Comment: Unicode and character sets
*************************** 9. row ***************************
    Name: Alexey Botchkov (Holyfoot)
Location: Izhevsk, Russia
 Comment: GIS extensions, embedded server, precision math
*************************** 10. row ***************************
    Name: Daniel Bartholomew
Location: Raleigh, USA
 Comment: MariaDB documentation, Buildbot, releases
*************************** 11. row ***************************
    Name: Colin Charles
Location: Selangor, Malesia
 Comment: MariaDB documentation, talks at a LOT of conferences
*************************** 12. row ***************************
    Name: Sergey Vojtovich
Location: Izhevsk, Russia
 Comment: initial implementation of plugin architecture, maintained native storage engines (MyISAM,
MEMORY, ARCHIVE, etc), rewrite of table cache
*************************** 13. row ***************************
    Name: Vladislav Vaintroub
Location: Mannheim, Germany
 Comment: MariaDB Java connector, new thread pool, Windows optimizations
*************************** 14. row ***************************
    Name: Elena Stepanova
Location: Sankt Petersburg, Russia
 Comment: QA, test cases
*************************** 15. row ***************************
    Name: Georg Richter
Location: Heidelberg, Germany
 Comment: New LGPL C connector, PHP connector
*************************** 16. row ***************************
    Name: Jan Lindström
Location: Ylämylly, Finland
 Comment: Working on InnoDB
*************************** 17. row ***************************
```

```
    Name: Lixun Peng
Location: Hangzhou, China
 Comment: Multi Source replication
*************************** 18. row ***************************
    Name: Olivier Bertrand
Location: Paris, France
 Comment: CONNECT storage engine
*************************** 19. row ***************************
    Name: Kentoku Shiba
Location: Tokyo, Japan
 Comment: Spider storage engine, metadata_lock_info Information schema
*************************** 20. row ***************************
    Name: Percona
Location: CA, USA
 Comment: XtraDB, microslow patches, extensions to slow log
*************************** 21. row ***************************
    Name: Vicentiu Ciorbaru
Location: Bucharest, Romania
 Comment: Roles
*************************** 22. row ***************************
    Name: Sudheera Palihakkara
Location:
 Comment: PCRE Regular Expressions
*************************** 23. row ***************************
    Name: Pavel Ivanov
Location: USA
 Comment: Some patches and bug fixes
*************************** 24. row ***************************
    Name: Konstantin Osipov
Location: Moscow, Russia
 Comment: Prepared statements (4.1), Cursors (5.0), GET_LOCK (10.0)
*************************** 25. row ***************************
    Name: Ian Gilfillan
Location: South Africa
 Comment: MariaDB documentation
*************************** 26. row ***************************
    Name: Federico Razolli
Location: Italy
 Comment: MariaDB documentation Italian translation
*************************** 27. row ***************************
    Name: Guilhem Bichot
Location: Bordeaux, France
 Comment: Replication (since 4.0)
*************************** 28. row ***************************
    Name: Andrei Elkin
Location: Espoo, Finland
 Comment: Replication
*************************** 29. row ***************************
    Name: Dmitri Lenev
Location: Moscow, Russia
 Comment: Time zones support (4.1), Triggers (5.0)
*************************** 30. row ***************************
    Name: Marc Alff
Location: Denver, CO, USA
 Comment: Signal, Resignal, Performance schema
*************************** 31. row ***************************
    Name: Mikael Ronström
Location: Stockholm, Sweden
 Comment: NDB Cluster, Partitioning, online alter table
*************************** 32. row ***************************
    Name: Ingo Strüwing
Location: Berlin, Germany
 Comment: Bug fixing in MyISAM, Merge tables etc
*************************** 33. row ***************************
    Name: Marko Mäkelä
Location: Helsinki, Finland
 Comment: InnoDB core developer
...
```

See Also

- SHOW CONTRIBUTORS. This list all members and sponsors of the MariaDB Foundation and other sponsors.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW BINARY LOGS

Syntax

```
SHOW BINARY LOGS
SHOW MASTER LOGS
```

Description

Lists the binary log files on the server. This statement is used as part of the procedure described in PURGE BINARY LOGS, that shows how to determine which logs can be purged.

> **CURRENT**
>
> This statement requires the BINLOG MONITOR privilege.
>
> **< 10.5.2**
>
> This statement requires the SUPER privilege and the REPLICATION_CLIENT privilege.

Examples

```
SHOW BINARY LOGS;
+--------------------+-----------+
| Log_name           | File_size |
+--------------------+-----------+
| mariadb-bin.000001 |     19039 |
| mariadb-bin.000002 |    717389 |
| mariadb-bin.000003 |       300 |
| mariadb-bin.000004 |       333 |
| mariadb-bin.000005 |       899 |
| mariadb-bin.000006 |       125 |
| mariadb-bin.000007 |     18907 |
| mariadb-bin.000008 |     19530 |
| mariadb-bin.000009 |       151 |
| mariadb-bin.000010 |       151 |
| mariadb-bin.000011 |       125 |
| mariadb-bin.000012 |       151 |
| mariadb-bin.000013 |       151 |
| mariadb-bin.000014 |       125 |
| mariadb-bin.000015 |       151 |
| mariadb-bin.000016 |       314 |
+--------------------+-----------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW BINLOG EVENTS

Syntax

```
SHOW BINLOG EVENTS
   [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]
```

Description

Shows the events in the binary log. If you do not specify `log_name`, the first binary log is displayed.

> **CURRENT**
>
> This statement requires the BINLOG MONITOR privilege.
>
> **< 10.5.2**
>
> This statement requires the REPLICATION SLAVE privilege.

Example

```
SHOW BINLOG EVENTS IN 'mysql_sandbox10019-bin.000002';
+-----------------------------+-----+------------------+-----------+-------------+--------------------
---------------------------+
| Log_name                    | Pos | Event_type       | Server_id | End_log_pos | Info
|
+-----------------------------+-----+------------------+-----------+-------------+--------------------
---------------------------+
| mysql_sandbox10019-bin.000002 |   4 | Format_desc      |         1 |         248 | Server ver:
10.0.19-MariaDB-log, Binlog ver: 4 |
| mysql_sandbox10019-bin.000002 | 248 | Gtid_list        |         1 |         273 | []
|
| mysql_sandbox10019-bin.000002 | 273 | Binlog_checkpoint |         1 |         325 | mysql_sandbox10019-
bin.000002                  |
| mysql_sandbox10019-bin.000002 | 325 | Gtid             |         1 |         363 | GTID 0-1-1
|
| mysql_sandbox10019-bin.000002 | 363 | Query            |         1 |         446 | CREATE DATABASE
blog                          |
| mysql_sandbox10019-bin.000002 | 446 | Gtid             |         1 |         484 | GTID 0-1-2
|
| mysql_sandbox10019-bin.000002 | 484 | Query            |         1 |         571 | use `blog`; CREATE
TABLE bb (id INT)           |
+-----------------------------+-----+------------------+-----------+-------------+--------------------
---------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW MASTER STATUS

Syntax

> **CURRENT**
>
> ```
> SHOW [MASTER | BINLOG] STATUS
> ```
>
> **< 10.5.2**
>
> ```
> SHOW MASTER STATUS
> ```

Description

Provides status information about the binary log files of the primary.

> **CURRENT**
>
> This statement requires the BINLOG MONITOR privilege.
>
> **< 10.5.2**

This statement requires the SUPER privilege and the REPLICATION_CLIENT privilege.

Note:

The following improved functionality is available from MariaDB 12.3.

To see information about the current GTIDs in the binary log, use the gtid_binlog_pos variable. It is enabled by default, and helps find the current state of the master server. (Previously, this required two statements, `SHOW MASTER STATUS` and `SELECT @@global.gtid_binlog_pos`.)

Example

CURRENT

From MariaDB 12.3, `SHOW BINLOG STATUS` includes the `Gtid_Binlog_Pos` column, so a separate `SELECT @@global.gtid_binlog_pos` statement is no longer required to see the current GTID position:

```
SHOW BINLOG STATUS;
+--------------------+----------+--------------+------------------+-----------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB | Gtid_Binlog_Pos |
+--------------------+----------+--------------+------------------+-----------------+
| mariadb-bin.000016 |      475 |              |                  | 0-1-2           |
+--------------------+----------+--------------+------------------+-----------------+
```

< 12.3

Before MariaDB 12.3, `SHOW BINLOG STATUS` (or `SHOW MASTER STATUS`) does not include the `Gtid_Binlog_Pos` column. To see the current GTID position, run an additional `SELECT @@global.gtid_binlog_pos`:

```
SHOW BINLOG STATUS;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| mariadb-bin.000016 |      475 |              |                  |
+--------------------+----------+--------------+------------------+
SELECT @@global.gtid_binlog_pos;
+---------------------------+
| @@global.gtid_binlog_pos |
+---------------------------+
| 0-1-2                     |
+---------------------------+
```

See Also

- MariaDB replication
- Using and Maintaining the Binary Log
- The gtid_binlog_pos variable

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CHARACTER SET

Syntax

```
SHOW CHARACTER SET
    [LIKE 'pattern' | WHERE expr]
```

Description

The `SHOW CHARACTER SET` statement shows all available character sets. The `LIKE` clause, if present on its own, indicates which character set names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

The same information can be queried from the Information Schema CHARACTER_SETS table.

See Setting Character Sets and Collations for details on specifying the character set at the server, database, table and column levels.

Examples

```
SHOW CHARACTER SET LIKE 'latin%';
+---------+-----------------------------+-------------------+--------+
| Charset | Description                 | Default collation | Maxlen |
+---------+-----------------------------+-------------------+--------+
| latin1  | cp1252 West European        | latin1_swedish_ci |      1 |
| latin2  | ISO 8859-2 Central European | latin2_general_ci |      1 |
| latin5  | ISO 8859-9 Turkish          | latin5_turkish_ci |      1 |
| latin7  | ISO 8859-13 Baltic          | latin7_general_ci |      1 |
+---------+-----------------------------+-------------------+--------+
```

```
SHOW CHARACTER SET WHERE Maxlen LIKE '2';
+---------+---------------------------+-------------------+--------+
| Charset | Description               | Default collation | Maxlen |
+---------+---------------------------+-------------------+--------+
| big5    | Big5 Traditional Chinese  | big5_chinese_ci   |      2 |
| sjis    | Shift-JIS Japanese        | sjis_japanese_ci  |      2 |
| euckr   | EUC-KR Korean             | euckr_korean_ci   |      2 |
| gb2312  | GB2312 Simplified Chinese | gb2312_chinese_ci |      2 |
| gbk     | GBK Simplified Chinese    | gbk_chinese_ci    |      2 |
| ucs2    | UCS-2 Unicode             | ucs2_general_ci   |      2 |
| cp932   | SJIS for Windows Japanese | cp932_japanese_ci |      2 |
+---------+---------------------------+-------------------+--------+
```

See Also

- Supported Character Sets and Collations
- Setting Character Sets and Collations
- Information Schema CHARACTER_SETS

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CLIENT_STATISTICS

Syntax

```
SHOW CLIENT_STATISTICS
```

Description

> **CURRENT**
>
> The `SHOW CLIENT_STATISTICS` statement has effectively been replaced by the generic SHOW TABLE STATISTICS statement. The information_schema.CLIENT_STATISTICS table holds statistics about client connections.
>
> **< 10.1.1**

> The `SHOW CLIENT_STATISTICS` statement is part of the User Statistics feature. The information_schema.CLIENT_STATISTICS table holds statistics about client connections.

The userstat system variable must be set to 1 to activate this feature. See the User Statistics and information_schema.CLIENT_STATISTICS articles for more information.

Example

```
SHOW CLIENT_STATISTICS\G
*************************** 1. row ***************************
               Client: localhost
     Total_connections: 35
Concurrent_connections: 0
        Connected_time: 708
             Busy_time: 2.5557979999999985
              Cpu_time: 0.04123740000000002
        Bytes_received: 3883
            Bytes_sent: 21595
  Binlog_bytes_written: 0
             Rows_read: 18
             Rows_sent: 115
          Rows_deleted: 0
         Rows_inserted: 0
          Rows_updated: 0
       Select_commands: 70
       Update_commands: 0
        Other_commands: 0
   Commit_transactions: 1
 Rollback_transactions: 0
    Denied_connections: 0
      Lost_connections: 0
         Access_denied: 0
         Empty_queries: 35
```

This page is licensed: CC BY-SA / Gnu FDL

SHOW COLLATION

Syntax

```
SHOW COLLATION
    [LIKE 'pattern' | WHERE expr]
```

Description

The output from `SHOW COLLATION` includes all available collations. The `LIKE` clause, if present on its own, indicates which collation names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

> **CURRENT**
>
> Similar information, including some extra information, can be queried from the Information Schema COLLATIONS table.
>
> **< 11.4.5**
>
> No similar information or extra information can be queried from the Information Schema COLLATIONS table.

See Setting Character Sets and Collations for details on specifying the collation at the server, database, table and column levels.

CURRENT

The `pad_attribute` column (not shown in the examples below) has a value of `NO PAD` or `PAD SPACE`. This attribute affects whether trailing spaces are significant in string comparisons. See the INFORMATION_SCHEMA.COLLATIONS table description for more information.

```
SHOW COLLATION LIKE 'utf8mb4_bin';
+-------------+---------+------+---------+----------+---------+---------------+
| Collation   | Charset | Id   | Default | Compiled | Sortlen | Pad_attribute |
+-------------+---------+------+---------+----------+---------+---------------+
| utf8mb4_bin | utf8mb4 | 46   |         | Yes      | 1       | PAD SPACE     |
+-------------+---------+------+---------+----------+---------+---------------+
```

< 12.1

The `pad_attribute` column is not available.

Examples

```
SHOW COLLATION LIKE 'latin1%';
+-------------------------+---------+------+---------+----------+---------+
| Collation               | Charset | Id   | Default | Compiled | Sortlen |
+-------------------------+---------+------+---------+----------+---------+
| latin1_german1_ci       | latin1  |    5 |         | Yes      |       1 |
| latin1_swedish_ci       | latin1  |    8 | Yes     | Yes      |       1 |
| latin1_danish_ci        | latin1  |   15 |         | Yes      |       1 |
| latin1_german2_ci       | latin1  |   31 |         | Yes      |       2 |
| latin1_bin              | latin1  |   47 |         | Yes      |       1 |
| latin1_general_ci       | latin1  |   48 |         | Yes      |       1 |
| latin1_general_cs       | latin1  |   49 |         | Yes      |       1 |
| latin1_spanish_ci       | latin1  |   94 |         | Yes      |       1 |
| latin1_swedish_nopad_ci | latin1  | 1032 |         | Yes      |       1 |
| latin1_nopad_bin        | latin1  | 1071 |         | Yes      |       1 |
+-------------------------+---------+------+---------+----------+---------+
```

```
SHOW COLLATION WHERE Sortlen LIKE '8' AND Charset LIKE 'utf8mb4';
+-----------------------------+---------+------+---------+----------+---------+
| Collation                   | Charset | Id   | Default | Compiled | Sortlen |
+-----------------------------+---------+------+---------+----------+---------+
| utf8mb4_unicode_ci          | utf8mb4 |  224 |         | Yes      |       8 |
| utf8mb4_icelandic_ci        | utf8mb4 |  225 |         | Yes      |       8 |
| utf8mb4_latvian_ci          | utf8mb4 |  226 |         | Yes      |       8 |
| utf8mb4_romanian_ci         | utf8mb4 |  227 |         | Yes      |       8 |
| utf8mb4_slovenian_ci        | utf8mb4 |  228 |         | Yes      |       8 |
| utf8mb4_polish_ci           | utf8mb4 |  229 |         | Yes      |       8 |
| utf8mb4_estonian_ci         | utf8mb4 |  230 |         | Yes      |       8 |
| utf8mb4_spanish_ci          | utf8mb4 |  231 |         | Yes      |       8 |
| utf8mb4_swedish_ci          | utf8mb4 |  232 |         | Yes      |       8 |
| utf8mb4_turkish_ci          | utf8mb4 |  233 |         | Yes      |       8 |
| utf8mb4_czech_ci            | utf8mb4 |  234 |         | Yes      |       8 |
| utf8mb4_danish_ci           | utf8mb4 |  235 |         | Yes      |       8 |
| utf8mb4_lithuanian_ci       | utf8mb4 |  236 |         | Yes      |       8 |
| utf8mb4_slovak_ci           | utf8mb4 |  237 |         | Yes      |       8 |
| utf8mb4_spanish2_ci         | utf8mb4 |  238 |         | Yes      |       8 |
| utf8mb4_roman_ci            | utf8mb4 |  239 |         | Yes      |       8 |
| utf8mb4_persian_ci          | utf8mb4 |  240 |         | Yes      |       8 |
| utf8mb4_esperanto_ci        | utf8mb4 |  241 |         | Yes      |       8 |
| utf8mb4_hungarian_ci        | utf8mb4 |  242 |         | Yes      |       8 |
| utf8mb4_sinhala_ci          | utf8mb4 |  243 |         | Yes      |       8 |
| utf8mb4_german2_ci          | utf8mb4 |  244 |         | Yes      |       8 |
| utf8mb4_croatian_mysql561_ci | utf8mb4 |  245 |         | Yes      |       8 |
| utf8mb4_unicode_520_ci      | utf8mb4 |  246 |         | Yes      |       8 |
| utf8mb4_vietnamese_ci       | utf8mb4 |  247 |         | Yes      |       8 |
| utf8mb4_croatian_ci         | utf8mb4 |  608 |         | Yes      |       8 |
| utf8mb4_myanmar_ci          | utf8mb4 |  609 |         | Yes      |       8 |
| utf8mb4_unicode_nopad_ci    | utf8mb4 | 1248 |         | Yes      |       8 |
| utf8mb4_unicode_520_nopad_ci | utf8mb4 | 1270 |         | Yes      |       8 |
+-----------------------------+---------+------+---------+----------+---------+
```

See Also

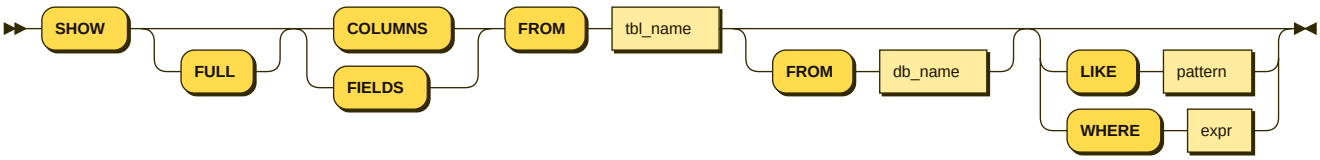
- Supported Character Sets and Collations
- Setting Character Sets and Collations
- Information Schema COLLATIONS

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW COLUMNS

Syntax

```
SHOW [FULL] {COLUMNS | FIELDS} FROM tbl_name [FROM db_name]
    [LIKE 'pattern' | WHERE expr]
```



Railroad diagram of SHOW COLUMNS — equivalent to the BNF above

Description

`SHOW COLUMNS` displays information about the columns in a given table. It also works for views. The `LIKE` clause, if present on its own, indicates which column names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

If the data types differ from what you expect them to be based on a `CREATE TABLE` statement, note that MariaDB sometimes changes data types when you create or alter a table. The conditions under which this occurs are described in the Silent Column Changes article.

The `FULL` keyword causes the output to include the column collation and comments, as well as the privileges you have for each column.

You can use `db_name.tbl_name` as an alternative to the `tbl_name FROM db_name` syntax. In other words, these two statements are equivalent:

```
SHOW COLUMNS FROM mytable FROM mydb;
SHOW COLUMNS FROM mydb.mytable;
```

`SHOW COLUMNS` displays the following values for each table column:

Field indicates the column name.

Type indicates the column data type.

Collation indicates the collation for non-binary string columns, or NULL for other columns. This value is displayed only if you use the FULL keyword.

The **Null** field contains YES if NULL values can be stored in the column, NO if not.

The **Key** field indicates whether the column is indexed:

- If Key is empty, the column either is not indexed or is indexed only as a secondary column in a multiple-column, non-unique index.
- If Key is PRI, the column is a `PRIMARY KEY` or is one of the columns in a multiple-column `PRIMARY KEY`.
- If Key is UNI, the column is the first column of a unique-valued index that cannot contain `NULL` values.
- If Key is MUL, multiple occurrences of a given value are allowed within the column. The column is the first column of a non-unique index or a unique-valued index that can contain `NULL` values.

If more than one of the **Key** values applies to a given column of a table, **Key** displays the one with the highest priority, in the order PRI, UNI, MUL.

A `UNIQUE` index may be displayed as `PRI` if it cannot contain `NULL` values and there is no `PRIMARY KEY` in the table. A `UNIQUE` index may display as `MUL` if several columns form a composite `UNIQUE` index; although the combination of the columns is unique, each column can still hold multiple occurrences of a given value.

The **Default** field indicates the default value that is assigned to the column.

The **Extra** field contains any additional information that is available about a given column.

Value	Description
AUTO_INCREMENT	The column was created with the AUTO_INCREMENT keyword.
PERSISTENT	The column was created with the PERSISTENT keyword.
VIRTUAL	The column was created with the VIRTUAL keyword.
on update CURRENT_TIMESTAMP	The column is a TIMESTAMP column that is automatically updated on INSERT and UPDATE.

Privileges indicates the privileges you have for the column. This value is displayed only if you use the `FULL` keyword.

Comment indicates any comment the column has. This value is displayed only if you use the `FULL` keyword.

`SHOW FIELDS` is a synonym for `SHOW COLUMNS`. Also DESCRIBE and EXPLAIN can be used as shortcuts.

You can also list a table's columns with:

```
mariadb-show db_name tbl_name
```

See the mariadb-show command for more details.

The DESCRIBE statement provides information similar to `SHOW COLUMNS`. The information_schema.COLUMNS table provides similar, but more complete, information.

The SHOW CREATE TABLE, SHOW TABLE STATUS, and SHOW INDEX statements also provide information about tables.

Examples

```
SHOW COLUMNS FROM city;
+------------+----------+------+-----+---------+----------------+
| Field      | Type     | Null | Key | Default | Extra          |
+------------+----------+------+-----+---------+----------------+
| Id         | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name       | char(35) | NO   |     |         |                |
| Country    | char(3)  | NO   | UNI |         |                |
| District   | char(20) | YES  | MUL |         |                |
| Population | int(11)  | NO   |     | 0       |                |
+------------+----------+------+-----+---------+----------------+
```

```
SHOW COLUMNS FROM employees WHERE Type LIKE 'Varchar%';
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| first_name    | varchar(30) | NO   | MUL | NULL    |       |
| last_name     | varchar(40) | NO   |     | NULL    |       |
| position      | varchar(25) | NO   |     | NULL    |       |
| home_address  | varchar(50) | NO   |     | NULL    |       |
| home_phone    | varchar(12) | NO   |     | NULL    |       |
| employee_code | varchar(25) | NO   | UNI | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

See Also

- DESCRIBE
- mariadb-show
- SHOW CREATE TABLE
- SHOW TABLE STATUS
- SHOW INDEX
- Extended SHOW
- Silent Column Changes

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CONTRIBUTORS

Syntax

```
SHOW CONTRIBUTORS
```

Description

The `SHOW CONTRIBUTORS` statement displays information about the companies and people who financially contribute to MariaDB. For each contributor, it displays `Name`, `Location`, and `Comment` values. All columns are encoded as `latin1`.

It displays all members and sponsors of the MariaDB Foundation as well as other financial contributors.

Example

```
SHOW CONTRIBUTORS;
+--------------------+------------------------------+-------------------------------------------------
-----------+
| Name               | Location                     | Comment
|
+--------------------+------------------------------+-------------------------------------------------
-----------+
| Alibaba Cloud      | https://www.alibabacloud.com/ | Platinum Sponsor of the MariaDB Foundation
|
| Tencent Cloud      | https://cloud.tencent.com    | Platinum Sponsor of the MariaDB Foundation
|
| Microsoft          | https://microsoft.com/       | Platinum Sponsor of the MariaDB Foundation
|
| MariaDB Corporation | https://mariadb.com          | Founding member, Platinum Sponsor of the MariaDB
Foundation |
| ServiceNow         | https://servicenow.com       | Platinum Sponsor of the MariaDB Foundation
|
| Intel              | https://www.intel.com        | Platinum Sponsor of the MariaDB Foundation
|
| SIT                | https://sit.org              | Platinum Sponsor of the MariaDB Foundation
|
| Visma              | https://visma.com            | Gold Sponsor of the MariaDB Foundation
|
| DBS                | https://dbs.com              | Gold Sponsor of the MariaDB Foundation
|
| IBM                | https://www.ibm.com          | Gold Sponsor of the MariaDB Foundation
|
| Automattic         | https://automattic.com       | Silver Sponsor of the MariaDB Foundation
|
| Percona            | https://www.percona.com/     | Sponsor of the MariaDB Foundation
|
| Galera Cluster     | https://galeracluster.com    | Sponsor of the MariaDB Foundation
|
| Google             | USA                          | Sponsoring encryption, parallel replication and
GTID       |
| Facebook           | USA                          | Sponsoring non-blocking API, LIMIT ROWS EXAMINED
etc        |
| Ronald Bradford    | Brisbane, Australia          | EFF contribution for UC2006 Auction
|
| Sheeri Kritzer     | Boston, Mass. USA            | EFF contribution for UC2006 Auction
|
| Mark Shuttleworth  | London, UK.                  | EFF contribution for UC2006 Auction
|
+--------------------+------------------------------+-------------------------------------------------
-----------+
```

See Also

- Log of MariaDB contributors.
- SHOW AUTHORS list the authors of MariaDB (including documentation, QA etc).
- MariaDB Foundation page on contributing financially.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CREATE DATABASE

Syntax

```
SHOW CREATE {DATABASE | SCHEMA} db_name
```

Description

Shows the CREATE DATABASE statement that creates the given database. `SHOW CREATE SCHEMA` is a synonym for `SHOW CREATE DATABASE`. `SHOW CREATE DATABASE` quotes database names according to the value of the sql_quote_show_create server system variable.

Examples

```
SHOW CREATE DATABASE test;
+----------+----------------------------------------------------------------+
| Database | Create Database                                                |
+----------+----------------------------------------------------------------+
| test     | CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+----------+----------------------------------------------------------------+

SHOW CREATE SCHEMA test;
+----------+----------------------------------------------------------------+
| Database | Create Database                                                |
+----------+----------------------------------------------------------------+
| test     | CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+----------+----------------------------------------------------------------+
```

With sql_quote_show_create turned off, the output looks slightly differently:

```
SHOW CREATE DATABASE test;
+----------+-----------------------------------------------------------+
| Database | Create Database                                           |
+----------+-----------------------------------------------------------+
| test     | CREATE DATABASE test /*!40100 DEFAULT CHARACTER SET latin1 */ |
+----------+-----------------------------------------------------------+
```

> **CURRENT**
>
> **< 10.6.5 / 10.5.13 / 10.4.22**
>
> In the `DEFAULT` clause of a `SHOW CREATE` statement, numbers are quoted.

See Also

- CREATE DATABASE
- ALTER DATABASE
- Character Sets and Collations

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CREATE EVENT

Syntax

```
SHOW CREATE EVENT event_name
```

Description

This statement displays the CREATE EVENT statement that creates a given event, as well as the SQL_MODE that was used when the trigger was created, and the character set used by the connection. To find out which events are present, use SHOW EVENTS.

> **CURRENT**

`SHOW CREATE EVENT` quotes identifiers according to the value of the sql_quote_show_create system variable.

< 10.6.5 / 10.5.13 / 10.4.22

`SHOW CREATE EVENT` quotes identifiers according to the value of the sql_quote_show_create system variable. Note, however, that the output of this statement is unreliably affected by the sql_quote_show_create system variable.

The information_schema.EVENTS table provides similar, but more complete, information.

Examples

```
SHOW CREATE EVENT test.e_daily\G
*************************** 1. row ***************************
               Event: e_daily
            sql_mode:
          time_zone: SYSTEM
        Create Event: CREATE EVENT `e_daily`
                        ON SCHEDULE EVERY 1 DAY
                        STARTS CURRENT_TIMESTAMP + INTERVAL 6 HOUR
                        ON COMPLETION NOT PRESERVE
                        ENABLE
                        COMMENT 'Saves total number of sessions then
                                clears the table each day'
                        DO BEGIN
                          INSERT INTO site_activity.totals (time, total)
                            SELECT CURRENT_TIMESTAMP, COUNT(*)
                            FROM site_activity.sessions;
                          DELETE FROM site_activity.sessions;
                        END
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
```

See also

- Events Overview
- CREATE EVENT
- ALTER EVENT
- DROP EVENT

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CREATE FUNCTION

Syntax

```
SHOW CREATE FUNCTION func_name
```

Description

This statement is similar to SHOW CREATE PROCEDURE but for stored functions.

CURRENT

`SHOW CREATE FUNCTION` quotes identifiers, according to the value of the sql_quote_show_create system variable.

< 10.6.5 / 10.5.13 / 10.4.22

> `SHOW CREATE FUNCTION` quotes identifiers, according to the value of the sql_quote_show_create system variable. However, the output of this statement is unreliably affected by the sql_quote_show_create system variable.

Example

```
SHOW CREATE FUNCTION VatCents\G
*************************** 1. row ***************************
            Function: VatCents
            sql_mode:
     Create Function: CREATE DEFINER=`root`@`localhost` FUNCTION `VatCents`(price DECIMAL(10,2)) RETURNS
int(11)
    DETERMINISTIC
BEGIN
 DECLARE x INT;
 SET x = price * 114;
 RETURN x;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

See Also

- Stored Functions
- CREATE FUNCTION

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CREATE PACKAGE BODY

Syntax

```
SHOW CREATE PACKAGE BODY  [ db_name . ] package_name
```

Description

> **CURRENT**
>
> The `SHOW CREATE PACKAGE BODY` statement shows the `CREATE PACKAGE BODY` statement that creates the given package body (that is, the implementation of the package).
>
> **< 11.4**
>
> The `SHOW CREATE PACKAGE BODY` statement shows the `CREATE PACKAGE BODY` statement that creates the given package body (that is, the implementation of the package). `CREATE PACKAGE BODY` can be used when Oracle SQL_MODE is set.

`SHOW CREATE PACKAGE BODY` quotes identifiers according to the value of the sql_quote_show_create system variable.

Examples

```
SHOW CREATE PACKAGE BODY employee_tools\G
*************************** 1. row ***************************
        Package body: employee_tools
            sql_mode:
PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ORACLE,NO_KEY_OPTIONS,NO_TABLE_OPTIONS,NO_FIELD_OPTIONS,NO_AUTO_
CREATE_USER
 Create Package Body: CREATE DEFINER="root"@"localhost" PACKAGE BODY "employee_tools" AS

  stdRaiseAmount DECIMAL(10,2):=500;

  PROCEDURE log (eid INT, ecmnt TEXT) AS
  BEGIN
    INSERT INTO employee_log (id, cmnt) VALUES (eid, ecmnt);
  END;

  PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2)) AS
    eid INT;
  BEGIN
    INSERT INTO employee (name, salary) VALUES (ename, esalary);
    eid:= last_insert_id();
    log(eid, 'hire ' || ename);
  END;

  FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2) AS
    nSalary DECIMAL(10,2);
  BEGIN
    SELECT salary INTO nSalary FROM employee WHERE id=eid;
    log(eid, 'getSalary id=' || eid || ' salary=' || nSalary);
    RETURN nSalary;
  END;

  PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2)) AS
  BEGIN
    UPDATE employee SET salary=salary+amount WHERE id=eid;
    log(eid, 'raiseSalary id=' || eid || ' amount=' || amount);
  END;

  PROCEDURE raiseSalaryStd(eid INT) AS
  BEGIN
    raiseSalary(eid, stdRaiseAmount);
    log(eid, 'raiseSalaryStd id=' || eid);
  END;

BEGIN
  log(0, 'Session ' || connection_id() || ' ' || current_user || ' started');
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

See Also

- CREATE PACKAGE
- SHOW CREATE PACKAGE
- DROP PACKAGE
- CREATE PACKAGE BODY
- DROP PACKAGE BODY
- Oracle SQL_MODE

This page is licensed: CC BY-SA / Gnu FDL

SHOW CREATE PACKAGE

Syntax

```
SHOW CREATE PACKAGE  [ db_name . ] package_name
```

Description

The `SHOW CREATE PACKAGE` statement can be used when Oracle SQL_MODE is set. It shows the `CREATE` statement that creates the given package specification.

`SHOW CREATE PACKAGE` quotes identifiers according to the value of the sql_quote_show_create system variable.

Examples

```
SHOW CREATE PACKAGE employee_tools\G
*************************** 1. row ***************************
             Package: employee_tools
            sql_mode:
PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ORACLE,NO_KEY_OPTIONS,NO_TABLE_OPTIONS,NO_FIELD_OPTIONS,NO_AUTO_
CREATE_USER
      Create Package: CREATE DEFINER="root"@"localhost" PACKAGE "employee_tools" AS
  FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2);
  PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2));
  PROCEDURE raiseSalaryStd(eid INT);
  PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2));
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

See Also

- CREATE PACKAGE
- DROP PACKAGE
- CREATE PACKAGE BODY
- SHOW CREATE PACKAGE BODY
- DROP PACKAGE BODY
- Oracle SQL_MODE

This page is licensed: CC BY-SA / Gnu FDL

SHOW CREATE PROCEDURE

Syntax

```
SHOW CREATE PROCEDURE proc_name
```

Description

This statement is a MariaDB extension. It returns the exact string that can be used to re-create the named stored procedure, as well as the SQL_MODE that was used when the trigger has been created and the character set used by the connection.. A similar statement, SHOW CREATE FUNCTION, displays information about stored functions.

Both statements require that:

- you are the owner of the routine;
- you have the SHOW CREATE ROUTINE privilege (from MariaDB 11.3.0); or

- you have the SELECT privilege on the mysql.proc table.

When none of the above statements are true, the statements display `NULL` for the `Create Procedure` or `Create Function` field.

> **Important:**
>
> Users with `SELECT` privileges on mysql.proc or `USAGE` privileges on `*.*` can view the text of routines, even when they do not have privileges for the function or procedure itself.

> **CURRENT**
>
> `SHOW CREATE PROCEDURE` quotes identifiers, according to the value of the sql_quote_show_create system variable.
>
> **< 10.6.5 / 10.5.13 / 10.4.22**
>
> `SHOW CREATE PROCEDURE` quotes identifiers, according to the value of the sql_quote_show_create system variable. The output of this statement is unreliably affected by the sql_quote_show_create system variable.

Examples

Here's a comparison of the `SHOW CREATE PROCEDURE` and SHOW CREATE FUNCTION statements.

```
SHOW CREATE PROCEDURE test.simpleproc\G
*************************** 1. row ***************************
           Procedure: simpleproc
            sql_mode:
    Create Procedure: CREATE PROCEDURE `simpleproc`(OUT param1 INT)
                      BEGIN
                      SELECT COUNT(*) INTO param1 FROM t;
                      END
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci

SHOW CREATE FUNCTION test.hello\G
*************************** 1. row ***************************
            Function: hello
            sql_mode:
     Create Function: CREATE FUNCTION `hello`(s CHAR(20))
                      RETURNS CHAR(50)
                      RETURN CONCAT('Hello, ',s,'!')
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
```

When the user issuing the statement does not have privileges on the routine, attempting to CALL the procedure raises Error 1370.

```
CALL test.prc1();
Error 1370 (42000): execute command denied to
  user 'test_user'@'localhost' for routine 'test'.'prc1'
```

If the user neither has privilege to the routine nor the SELECT privilege on mysql.proc table, it raises Error 1305, informing them that the procedure does not exist.

```
SHOW CREATE TABLES test.prc1\G
Error 1305 (42000): PROCEDURE prc1 does not exist
```

See Also

- Stored Procedure Overview
- CREATE PROCEDURE
- ALTER PROCEDURE
- DROP PROCEDURE
- SHOW PROCEDURE STATUS
- Stored Routine Privileges
- Information Schema ROUTINES Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CREATE SEQUENCE

Syntax

```
SHOW CREATE SEQUENCE sequence_name;
```

Description

Shows the CREATE SEQUENCE statement that creates the given sequence. The statement requires the `SELECT` privilege for the table.

`SHOW CREATE SEQUENCE` quotes identifiers according to the value of the sql_quote_show_create system variable.

Example

```
CREATE SEQUENCE s1 START WITH 50;
SHOW CREATE SEQUENCE s1\G;
*************************** 1. row ***************************
       Table: s1
Create Table: CREATE SEQUENCE `s1` start with 50 minvalue 1 maxvalue 9223372036854775806
  increment by 1 cache 1000 nocycle ENGINE=InnoDB
```

Notes

If you want to see the underlying table structure used for the `SEQUENCE` you can use SHOW CREATE TABLE on the `SEQUENCE`. You can also use `SELECT` to read the current recorded state of the `SEQUENCE`:

```
SHOW CREATE TABLE s1\G
*************************** 1. row ***************************
       Table: s1
Create Table: CREATE TABLE `s1` (
  `next_not_cached_value` bigint(21) NOT NULL,
  `minimum_value` bigint(21) NOT NULL,
  `maximum_value` bigint(21) NOT NULL,
  `start_value` bigint(21) NOT NULL COMMENT 'start value when sequences is created
    or value if RESTART is used',
  `increment` bigint(21) NOT NULL COMMENT 'increment value',
  `cache_size` bigint(21) unsigned NOT NULL,
  `cycle_option` tinyint(1) unsigned NOT NULL COMMENT '0 if no cycles are allowed,
    1 if the sequence should begin a new cycle when maximum_value is passed',
  `cycle_count` bigint(21) NOT NULL COMMENT 'How many cycles have been done'
) ENGINE=InnoDB SEQUENCE=1

SELECT * FROM s1\G
*************************** 1. row ***************************
next_not_cached_value: 50
        minimum_value: 1
        maximum_value: 9223372036854775806
          start_value: 50
            increment: 1
           cache_size: 1000
         cycle_option: 0
          cycle_count: 0
```

The Information Schema SEQUENCES Table also provides information about available sequences:

```
SELECT * FROM INFORMATION_SCHEMA.SEQUENCES\G
*************************** 1. row ***************************
       SEQUENCE_CATALOG: def
        SEQUENCE_SCHEMA: test
          SEQUENCE_NAME: s1
              DATA_TYPE: bigint
      NUMERIC_PRECISION: 64
NUMERIC_PRECISION_RADIX: 2
          NUMERIC_SCALE: 0
            START_VALUE: 50
          MINIMUM_VALUE: 1
          MAXIMUM_VALUE: 9223372036854775806
              INCREMENT: 1
           CYCLE_OPTION: 0
```

See Also

- CREATE SEQUENCE
- ALTER SEQUENCE
- Information Schema SEQUENCES Table

This page is licensed: CC BY-SA / Gnu FDL

SHOW CREATE SERVER

CURRENT

Syntax

```
SHOW CREATE SERVER server_name
```

Description

Shows the CREATE SERVER statement that created the given server definition.

Example

```
SHOW CREATE SERVER srv1\G
*************************** 1. row ***************************
       Server: srv1
Create Server: CREATE SERVER `srv1` FOREIGN DATA WRAPPER mysql
  OPTIONS (HOST '172.30.0.58', DATABASE 'db1', USER 'maxscale', PASSWORD 'password');
```

< 11.7

The `SHOW CREATE SERVER` statement is not available.

See Also

- CREATE SERVER

This page is licensed: CC BY-SA / Gnu FDL

SHOW CREATE TABLE

Syntax

```
SHOW CREATE TABLE tbl_name
```

Description

Shows the CREATE TABLE statement that creates the given table. The statement requires the SELECT privilege for the table. This statement also works with views and SEQUENCE.

`SHOW CREATE TABLE` quotes table and column names according to the value of the sql_quote_show_create server system variable.

Certain SQL_MODE values can result in parts of the original CREATE statement not being included in the output. MariaDB-specific table options, column options, and index options are not included in the output of this statement if the NO_TABLE_OPTIONS, NO_FIELD_OPTIONS and NO_KEY_OPTIONS SQL_MODE flags are used. All MariaDB-specific table attributes are also not shown when a non-MariaDB/MySQL emulation mode is used, which includes ANSI, DB2, POSTGRESQL, MSSQL, MAXDB or ORACLE.

Invalid table options, column options and index options are normally commented out (note, that it is possible to create a table with invalid options, by altering a table of a different engine, where these options were valid). To have them uncommented, enable the IGNORE_BAD_TABLE_OPTIONS SQL_MODE. Remember that replaying a CREATE TABLE statement with uncommented invalid options will fail with an error, unless the IGNORE_BAD_TABLE_OPTIONS SQL_MODE is in effect.

Note that `SHOW CREATE TABLE` is not meant to provide metadata about a table. It provides information about how the table was declared, but the real table structure could differ a bit. For example, if an index has been declared as `HASH`, the `CREATE TABLE` statement returned by `SHOW CREATE TABLE` will declare that index as `HASH`; however, it is possible that the index is in fact a `BTREE`, because the storage engine does not support `HASH`.

MariaDB permits TEXT and BLOB data types to be assigned a DEFAULT value. As a result, `SHOW CREATE TABLE` will append a `DEFAULT NULL` to nullable TEXT or BLOB fields if no specific default is provided.

CURRENT

Numbers are quoted in the `DEFAULT` clause in `SHOW CREATE` statement.

< 10.2.2

> Numbers are not quoted in the `DEFAULT` clause in `SHOW CREATE` statement.

Index Order

Indexes are sorted and displayed in the following order, which may differ from the order of the CREATE TABLE statement.

- PRIMARY KEY
- UNIQUE keys where all column are NOT NULL
- UNIQUE keys that don't contain partial segments
- Other UNIQUE keys
- LONG UNIQUE keys
- Normal keys
- Fulltext keys

See sql/sql_table.cc for details.

Examples

```
SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE `t` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `s` char(60) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

With sql_quote_show_create off:

```
SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE t (
  id int(11) NOT NULL AUTO_INCREMENT,
  s char(60) DEFAULT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

SQL_MODE impacting the output:

```
SELECT @@sql_mode;
+--------------------------------------------------------------------------------------------+
| @@sql_mode                                                                                 |
+--------------------------------------------------------------------------------------------+
| STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+--------------------------------------------------------------------------------------------+

CREATE TABLE `t1` (
       `id` int(11) NOT NULL AUTO_INCREMENT,
       `msg` varchar(100) DEFAULT NULL,
       PRIMARY KEY (`id`)
     ) ENGINE=InnoDB DEFAULT CHARSET=latin1
;

SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `msg` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1

SET SQL_MODE=ORACLE;

SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE "t1" (
  "id" int(11) NOT NULL,
  "msg" varchar(100) DEFAULT NULL,
  PRIMARY KEY ("id")
```

See Also

- SHOW CREATE SEQUENCE
- SHOW CREATE VIEW

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CREATE TRIGGER

Syntax

```
SHOW CREATE TRIGGER trigger_name
```

Description

This statement shows a CREATE TRIGGER statement that creates the given trigger, as well as the SQL_MODE that was used when the trigger has been created and the character set used by the connection.

The TRIGGER privilege is required on the table the trigger is defined for to execute this statement.

> **CURRENT**
>
> `SHOW CREATE TRIGGER` quotes identifiers, according to the value of the sql_quote_show_create system variable.
>
> **< 10.6.5 / 10.5.13 / 10.4.22**
>
> `SHOW CREATE TRIGGER` quotes identifiers, according to the value of the sql_quote_show_create system variable. However, the output of this statement is unreliably affected by the sql_quote_show_create system variable.

Examples

```
SHOW CREATE TRIGGER example\G
*************************** 1. row ***************************
               Trigger: example
              sql_mode: ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,STRICT_ALL_TABLES
,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_
ENGINE_SUBSTITUTION
SQL Original Statement: CREATE DEFINER=`root`@`localhost` TRIGGER example BEFORE
 INSERT ON t FOR EACH ROW
BEGIN
        SET NEW.c = NEW.c * 2;
END
  character_set_client: cp850
  collation_connection: cp850_general_ci
  Database Collation: utf8_general_ci
  Created: 2016-09-29 13:53:34.35
```

CURRENT

The `Created` column serves to better view multiple trigger events.

< 10.2.3

The `Created` column is unavailable.

See Also

- Trigger Overview
- CREATE TRIGGER
- DROP TRIGGER
- information_schema.TRIGGERS Table
- SHOW TRIGGERS
- Trigger Limitations

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW CREATE USER

Syntax

```
SHOW CREATE USER [user-name]
```

Description

Shows the CREATE USER statement that creates the given user. The statement requires the SELECT privilege for the mysql database, except for the current user. The CREATE USER statement for the current user is shown where no user is specified.

`SHOW CREATE USER` quotes identifiers according to the value of the sql_quote_show_create system variable.

Examples

```
CREATE USER foo4@test require cipher 'text'
  issuer 'foo_issuer' subject 'foo_subject';

SHOW CREATE USER foo4@test\G
*************************** 1. row ***************************
CREATE USER 'foo4'@'test'
  REQUIRE ISSUER 'foo_issuer'
  SUBJECT 'foo_subject'
  CIPHER 'text'
```

User Password Expiry:

```
CREATE USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY;

SHOW CREATE USER 'monty'@'localhost';
+-----------------------------------------------------------------+
| CREATE USER for monty@localhost                                 |
+-----------------------------------------------------------------+
| CREATE USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY |
+-----------------------------------------------------------------+
```

See Also

- CREATE USER
- ALTER USER
- SHOW GRANTS shows the `GRANTS/PRIVILEGES` for a user.
- SHOW PRIVILEGES shows the privileges supported by MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

SHOW CREATE VIEW

Syntax

```
SHOW CREATE VIEW [view-name]
```

Description

This statement shows a CREATE VIEW statement that creates the given view, as well as the character set used by the connection when the view was created. This statement also works with views.

`SHOW CREATE VIEW` quotes table, column and stored function names according to the value of the sql_quote_show_create server system variable.

Examples

```
SHOW CREATE VIEW example\G
*************************** 1. row ***************************
                View: example
         Create View: CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL
SECURITY DEFINER VIEW `example` AS (select `t`.`id` AS `id`,`t`.`s` AS `s` from
`t`)
character_set_client: cp850
collation_connection: cp850_general_ci
```

With sql_quote_show_create off:

```
SHOW CREATE VIEW example\G
*************************** 1. row ***************************
                View: example
         Create View: CREATE ALGORITHM=UNDEFINED DEFINER=root@localhost SQL SECU
RITY DEFINER VIEW example AS (select t.id AS id,t.s AS s from t)
character_set_client: cp850
collation_connection: cp850_general_ci
```

Grants

To be able to see a view, you need to have the SHOW VIEW and the SELECT privilege on the view:

```
GRANT SHOW VIEW,SELECT ON test_database.test_view TO 'test'@'localhost';
```

See Also

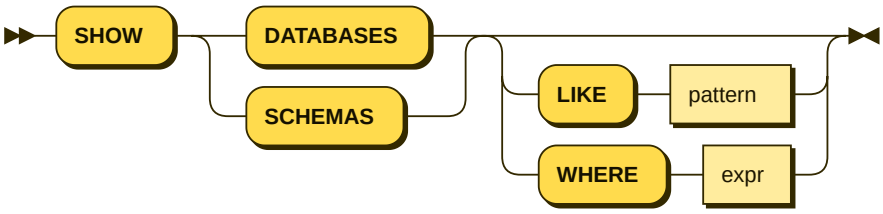
- Grant privileges to tables, views etc

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW DATABASES

Syntax

```
SHOW {DATABASES | SCHEMAS}
    [LIKE 'pattern' | WHERE expr]
```



Railroad diagram of SHOW DATABASES — equivalent to the BNF above

Description

`SHOW DATABASES` lists the databases on the MariaDB server host. `SHOW SCHEMAS` is a synonym for `SHOW DATABASES`. The `LIKE` clause, if present on its own, indicates which database names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

You see only those databases for which you have some kind of privilege, unless you have the global SHOW DATABASES privilege. You can also get this list using the mariadb-show command.

If the server was started with the `--skip-show-database` option, you cannot use this statement at all unless you have the SHOW DATABASES privilege.

The list of results returned by `SHOW DATABASES` is based on directories in the data directory, which is how MariaDB implements databases. It's possible that output includes directories that do not correspond to actual databases.

The Information Schema SCHEMATA table also contains database information.

Examples

```
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
```

```
SHOW DATABASES LIKE 'm%';
+---------------+
| Database (m%) |
+---------------+
| mysql         |
+---------------+
```

See Also

- CREATE DATABASE
- ALTER DATABASE
- DROP DATABASE
- SHOW CREATE DATABASE
- Character Sets and Collations
- Information Schema SCHEMATA Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW ENGINE INNODB STATUS

Syntax

```
SHOW ENGINE INNODB STATUS
```

`SHOW ENGINE INNODB STATUS` is a specific form of the SHOW ENGINE statement that displays the InnoDB Monitor output, which is extensive InnoDB information which can be useful in diagnosing problems.

The following sections are displayed

- Status: Shows the timestamp, monitor name and the number of seconds, or the elapsed time between the current time and the time the InnoDB Monitor output was last displayed. The per-second averages are based upon this time.
- BACKGROUND THREAD: srv_master_thread lines show work performed by the main background thread.
- SEMAPHORES: Threads waiting for a semaphore and stats on how the number of times threads have needed a spin or a wait on a mutex or rw-lock semaphore. If this number of threads is large, there may be I/O or contention issues. Reducing the size of the innodb_thread_concurrency system variable may help if contention is related to thread scheduling. `Spin rounds per wait` shows the number of spinlock rounds per OS wait for a mutex.
- LATEST FOREIGN KEY ERROR: Only shown if there has been a foreign key constraint error, it displays the failed statement and information about the constraint and the related tables.
- LATEST DETECTED DEADLOCK: Only shown if there has been a deadlock, it displays the transactions involved in the deadlock and the statements being executed, held and required locked and the transaction rolled back to.

- TRANSACTIONS: The output of this section can help identify lock contention, as well as reasons for the deadlocks.
- FILE I/O: InnoDB thread information as well as pending I/O operations and I/O performance statistics.
- INSERT BUFFER AND ADAPTIVE HASH INDEX: InnoDB insert buffer (old name for the change buffer) and adaptive hash index status information, including the number of each type of operation performed, and adaptive hash index performance.
- LOG: InnoDB log information, including current log sequence number, how far the log has been flushed to disk, the position at which InnoDB last took a checkpoint, pending writes and write performance statistics.
- BUFFER POOL AND MEMORY: Information on buffer pool pages read and written, which allows you to see the number of data file I/O operations performed by your queries. See InnoDB Buffer Pool for more. Similar information is also available from the INFORMATION_SCHEMA.INNODB_BUFFER_POOL_STATS table.
- ROW OPERATIONS:Information about the main thread, including the number and performance rate for each type of row operation.

If the innodb_status_output_locks system variable is set to `1`, extended lock information will be displayed.

Example output:

```
=====================================
2019-09-06 12:44:13 0x7f93cc236700 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 4 seconds
-----------------
BACKGROUND THREAD
-----------------
srv_master_thread loops: 2 srv_active, 0 srv_shutdown, 83698 srv_idle
srv_master_thread log flush and writes: 83682
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 15
OS WAIT ARRAY INFO: signal count 8
RW-shared spins 0, rounds 20, OS waits 7
RW-excl spins 0, rounds 0, OS waits 0
RW-sx spins 0, rounds 0, OS waits 0
Spin rounds per wait: 20.00 RW-shared, 0.00 RW-excl, 0.00 RW-sx
------------
TRANSACTIONS
------------
Trx id counter 236
Purge done for trx's n:o < 236 undo n:o < 0 state: running
History list length 22
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 421747401994584, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 421747401990328, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
--------
FILE I/O
--------
I/O thread 0 state: waiting for completed aio requests (insert buffer thread)
I/O thread 1 state: waiting for completed aio requests (log thread)
I/O thread 2 state: waiting for completed aio requests (read thread)
I/O thread 3 state: waiting for completed aio requests (read thread)
I/O thread 4 state: waiting for completed aio requests (read thread)
I/O thread 5 state: waiting for completed aio requests (read thread)
I/O thread 6 state: waiting for completed aio requests (write thread)
I/O thread 7 state: waiting for completed aio requests (write thread)
I/O thread 8 state: waiting for completed aio requests (write thread)
I/O thread 9 state: waiting for completed aio requests (write thread)
Pending normal aio reads: [0, 0, 0, 0] , aio writes: [0, 0, 0, 0] ,
 ibuf aio reads:, log i/o's:, sync i/o's:
Pending flushes (fsync) log: 0; buffer pool: 0
286 OS file reads, 171 OS file writes, 22 OS fsyncs
0.00 reads/s, 0 avg bytes/read, 0.00 writes/s, 0.00 fsyncs/s
-------------------------------------
INSERT BUFFER AND ADAPTIVE HASH INDEX
-------------------------------------
Ibuf: size 1, free list len 0, seg size 2, 0 merges
merged operations:
 insert 0, delete mark 0, delete 0
discarded operations:
 insert 0, delete mark 0, delete 0
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
0.00 hash searches/s, 0.00 non-hash searches/s
---
LOG
---
Log sequence number 445926
```

```
Log flushed up to   445926
Pages flushed up to 445926
Last checkpoint at  445917
0 pending log flushes, 0 pending chkp writes
18 log i/o's done, 0.00 log i/o's/second
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 167772160
Dictionary memory allocated 50768
Buffer pool size   8012
Free buffers       7611
Database pages     401
Old database pages 0
Modified db pages  0
Percent of dirty pages(LRU & free pages): 0.000
Max dirty pages percent: 75.000
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 0, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 264, created 137, written 156
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
No buffer pool page gets since the last printout
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 401, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
--------------
ROW OPERATIONS
--------------
0 queries inside InnoDB, 0 queries in queue
0 read views open inside InnoDB
Process ID=4267, Main thread ID=140272021272320, state: sleeping
Number of rows inserted 1, updated 0, deleted 0, read 1
0.00 inserts/s, 0.00 updates/s, 0.00 deletes/s, 0.00 reads/s
Number of system rows inserted 0, updated 0, deleted 0, read 0
0.00 inserts/s, 0.00 updates/s, 0.00 deletes/s, 0.00 reads/s
----------------------------
END OF INNODB MONITOR OUTPUT
============================
```

This page is licensed: CC BY-SA / Gnu FDL

SHOW ENGINE

Syntax

```
SHOW ENGINE [engine-name] {STATUS | MUTEX}
```

Description

`SHOW ENGINE` displays operational information about a storage engine. The following statements currently are supported:

```
SHOW ENGINE INNODB STATUS
SHOW ENGINE INNODB MUTEX
SHOW ENGINE PERFORMANCE_SCHEMA STATUS
SHOW ENGINE ROCKSDB STATUS
```

If the Sphinx Storage Engine is installed, the following is also supported:

```
SHOW ENGINE SPHINX STATUS
```

SHOW ENGINE INNODB STATUS

`SHOW ENGINE INNODB STATUS` displays extensive information from the standard InnoDB Monitor about the state of the InnoDB storage engine. See SHOW ENGINE INNODB STATUS for more.

SHOW ENGINE INNODB MUTEX

`SHOW ENGINE INNODB MUTEX` displays InnoDB mutex statistics.

The statement displays the following output fields:

- Type: Always InnoDB.
- Name: The source file where the mutex is implemented, and the line number in the file where the mutex is created. The line number is dependent on the MariaDB version.
- Status: This field displays the following values if `UNIV_DEBUG` was defined at compilation time (for example, in include/univ.h in the InnoDB part of the source tree). Only the `os_waits` value is displayed if `UNIV_DEBUG` was not defined. Without `UNIV_DEBUG`, the information on which the output is based is insufficient to distinguish regular mutexes and mutexes that protect rw-locks (which allow multiple readers or a single writer). Consequently, the output may appear to contain multiple rows for the same mutex.
  - count indicates how many times the mutex was requested.
  - spin_waits indicates how many times the spinlock had to run.
  - spin_rounds indicates the number of spinlock rounds. (spin_rounds divided by spin_waits provides the average round count.)
  - os_waits indicates the number of operating system waits. This occurs when the spinlock did not work (the mutex was not locked during the spinlock and it was necessary to yield to the operating system and wait).
  - os_yields indicates the number of times a the thread trying to lock a mutex gave up its timeslice and yielded to the operating system (on the presumption that allowing other threads to run will free the mutex so that it can be locked).
  - os_wait_times indicates the amount of time (in ms) spent in operating system waits, if the timed_mutexes system variable is 1 (ON). If timed_mutexes is 0 (OFF), timing is disabled, so os_wait_times is 0. timed_mutexes is off by default.

Information from this statement can be used to diagnose system problems. For example, large values of spin_waits and spin_rounds may indicate scalability problems.

The information_schema.INNODB_MUTEXES table provides similar information.

SHOW ENGINE PERFORMANCE_SCHEMA STATUS

This statement shows how much memory is used for performance_schema tables and internal buffers.

The output contains the following fields:

- Type: Always `performance_schema`.
- Name: The name of a table, the name of an internal buffer, or the `performance_schema` word, followed by a dot and an attribute. Internal buffers names are enclosed by parenthesis. `performance_schema` means that the attribute refers to the whole database (it is a total).
- Status: The value for the attribute.

The following attributes are shown, in this order, for all tables:

- row_size: The memory used for an individual record. This value will never change.
- row_count: The number of rows in the table or buffer. For some tables, this value depends on a server system variable.
- memory: For tables and `performance_schema`, this is the result of `row_size` * `row_count`.

For internal buffers, the attributes are:

- count
- size

SHOW ENGINE ROCKSDB STATUS

See also MyRocks Performance Troubleshooting

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW ENGINES

Syntax

```
SHOW [STORAGE] ENGINES
```

Description

`SHOW ENGINES` displays status information about the server's storage engines. This is particularly useful for checking whether a storage engine is supported, or to see what the default engine is. `SHOW TABLE TYPES` is a deprecated synonym.

The information_schema.ENGINES table provides the same information.

Since storage engines are plugins, different information about them is also shown in the information_schema.PLUGINS table and by the SHOW PLUGINS statement.

Note that both MySQL's InnoDB and Percona's XtraDB replacement are labeled as `InnoDB`. However, if XtraDB is in use, it will be specified in the `COMMENT` field. See XtraDB and InnoDB. The same applies to FederatedX.

The output consists of the following columns:

- `Engine` indicates the engine's name.
- `Support` indicates whether the engine is installed, and whether it is the default engine for the current session.
- `Comment` is a brief description.
- `Transactions`, `XA` and `Savepoints` indicate whether transactions, XA transactions and transaction savepoints are supported by the engine.

Examples

```
SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 2. row ***************************
      Engine: CSV
     Support: YES
     Comment: CSV storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 3. row ***************************
      Engine: MyISAM
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 4. row ***************************
      Engine: BLACKHOLE
     Support: YES
     Comment: /dev/null storage engine (anything you write to it disappears)
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 5. row ***************************
      Engine: FEDERATED
     Support: YES
     Comment: FederatedX pluggable storage engine
Transactions: YES
          XA: NO
  Savepoints: YES
*************************** 6. row ***************************
      Engine: MRG_MyISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 7. row ***************************
      Engine: ARCHIVE
     Support: YES
     Comment: Archive storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 8. row ***************************
      Engine: MEMORY
     Support: YES
     Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 9. row ***************************
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 10. row ***************************
      Engine: Aria
     Support: YES
```

```
     Comment: Crash-safe tables with MyISAM heritage
Transactions: NO
          XA: NO
  Savepoints: NO
10 rows in set (0.00 sec)
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW ERRORS

Syntax

```
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW ERRORS [LIMIT row_count OFFSET offset]
SHOW COUNT(*) ERRORS
```

Description

This statement is similar to SHOW WARNINGS, except that instead of displaying errors, warnings, and notes, it displays only errors.

The `LIMIT` clause has the same syntax as for the SELECT statement.

The `SHOW COUNT(*) ERRORS` statement displays the number of errors. You can also retrieve this number from the error_count variable.

```
SHOW COUNT(*) ERRORS;
SELECT @@error_count;
```

The value of error_count might be greater than the number of messages displayed by SHOW WARNINGS if the max_error_count system variable is set so low that not all messages are stored.

For a list of MariaDB error codes, see MariaDB Error Codes.

Examples

```
SELECT f();
ERROR 1305 (42000): FUNCTION f does not exist

SHOW COUNT(*) ERRORS;
+-----------------------+
| @@session.error_count |
+-----------------------+
|                     1 |
+-----------------------+

SHOW ERRORS;
+-------+------+--------------------------+
| Level | Code | Message                  |
+-------+------+--------------------------+
| Error | 1305 | FUNCTION f does not exist |
+-------+------+--------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW EVENTS

Syntax

```
SHOW EVENTS [{FROM | IN} schema_name]
    [LIKE 'pattern' | WHERE expr]
```

Description

Shows information about Event Manager events (created with CREATE EVENT). Requires the EVENT privilege. Without any arguments, `SHOW EVENTS` lists all of the events in the current schema:

```
SELECT CURRENT_USER(), SCHEMA();
+----------------+----------+
| CURRENT_USER() | SCHEMA() |
+----------------+----------+
| jon@ghidora    | myschema |
+----------------+----------+

SHOW EVENTS\G
*************************** 1. row ***************************
                  Db: myschema
                Name: e_daily
             Definer: jon@ghidora
           Time zone: SYSTEM
                Type: RECURRING
          Execute at: NULL
      Interval value: 10
      Interval field: SECOND
              Starts: 2006-02-09 10:41:23
                Ends: NULL
              Status: ENABLED
          Originator: 0
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
```

To see the event action, use SHOW CREATE EVENT instead, or look at the information_schema.EVENTS table.

To see events for a specific schema, use the `FROM` clause. For example, to see events for the test schema, use the following statement:

```
SHOW EVENTS FROM test;
```

The `LIKE` clause, if present, indicates which event names to match. The `WHERE` clause can be given to select rows using more general conditions, as discussed in Extended Show.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW EXPLAIN

Syntax

```
SHOW EXPLAIN [FORMAT=JSON] FOR <connection_id>;
EXPLAIN [FORMAT=JSON] FOR CONNECTION <connection_id>;
```

Description

The `SHOW EXPLAIN` command allows one to get an EXPLAIN (that is, a description of a query plan) of a query running in a certain connection.

```
SHOW EXPLAIN FOR <connection_id>;
```

will produce an `EXPLAIN` output for the query that connection number `connection_id` is running. The connection id can be obtained with SHOW PROCESSLIST.

```
SHOW EXPLAIN FOR 1;
+------+-------------+-------+-------+---------------+------+---------+------+---------+-------------+
| id   | select_type | table | type  | possible_keys | key  | key_len | ref  | rows    | Extra       |
+------+-------------+-------+-------+---------------+------+---------+------+---------+-------------+
|    1 | SIMPLE      | tbl   | index | NULL          | a    | 5       | NULL | 1000107 | Using index |
+------+-------------+-------+-------+---------------+------+---------+------+---------+-------------+
1 row in set, 1 warning (0.00 sec)
```

The output is always accompanied with a warning which shows the query the target connection is running (this shows what the `EXPLAIN` is for):

```
SHOW WARNINGS;
+-------+------+-----------------------+
| Level | Code | Message               |
+-------+------+-----------------------+
| Note  | 1003 | select sum(a) from tbl |
+-------+------+-----------------------+
1 row in set (0.00 sec)
```

> **CURRENT**
>
> **EXPLAIN FOR CONNECTION**
>
> The `EXPLAIN FOR CONNECTION` syntax was added for MySQL compatibility.
>
> **FORMAT=JSON**
>
> `SHOW EXPLAIN [FORMAT=JSON] FOR <connection_id>` extends `SHOW EXPLAIN` to return more detailed JSON output.
>
> **< 10.9**
>
> `EXPLAIN FOR CONNECTION` and `FORMAT=JSON` are not available.

Possible Errors

The output can be only produced if the target connection is *currently* running a query, which has a ready query plan. If this is not the case, the output will be:

```
SHOW EXPLAIN FOR 2;
ERROR 1932 (HY000): Target is not running an EXPLAINable command
```

You will get this error when:

- The target connection is not running a command for which one can run `EXPLAIN`;
- The target connection is running a command for which one can run `EXPLAIN`, but there is no query plan yet (for example, tables are open and locks are acquired before the query plan is produced).

Differences Between SHOW EXPLAIN and EXPLAIN Outputs

Background

In MySQL, `EXPLAIN` execution takes a slightly different route from the way the real query (typically the `SELECT`) is optimized. This is unfortunate, and has caused a number of bugs in `EXPLAIN`. (For example, see MDEV-326, MDEV-410, and lp:1013343.lp:992942 is not directly about `EXPLAIN`, but it also would not have existed if MySQL didn't try to delete parts of a query plan in the middle of the query)

`SHOW EXPLAIN` examines a running `SELECT`, and hence its output may be slightly different from what `EXPLAIN SELECT` would produce. We did our best to make sure that either the difference is negligible, or `SHOW EXPLAIN`'s

output is closer to reality than `EXPLAIN`'s output.

List of Recorded Differences

- `SHOW EXPLAIN` may have Extra=‘`no matching row in const table`’, where `EXPLAIN` would produce Extra=‘`Impossible WHERE ...`’
- For queries with subqueries, `SHOW EXPLAIN` may print `select_type==PRIMARY` where regular `EXPLAIN` used to print `select_type==SIMPLE`, or vice versa.

Required Permissions

Running `SHOW EXPLAIN` requires the same permissions as running SHOW PROCESSLIST would.

See Also

- EXPLAIN
- EXPLAIN ANALYZE, which will perform a query and outputs enhanced `EXPLAIN` results.
- SHOW ANALYZE
- It is also possible to save EXPLAIN into the slow query log.

This page is licensed: CC BY-SA / Gnu FDL

SHOW FUNCTION CODE

Syntax

```
SHOW FUNCTION CODE func_name
```

Description

`SHOW FUNCTION CODE` shows a representation of the internal implementation of the stored function.

It is similar to SHOW PROCEDURE CODE but for stored functions.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW FUNCTION STATUS

Syntax

```
SHOW FUNCTION STATUS
    [LIKE 'pattern' | WHERE expr]
```

Description

This statement is similar to SHOW PROCEDURE STATUS but for stored functions.

The LIKE clause, if present on its own, indicates which function names to match.

The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

The information_schema.ROUTINES table contains more detailed information.

Examples

Showing all stored functions:

```
SHOW FUNCTION STATUS\G
*************************** 1. row ***************************
                  Db: test
                Name: VatCents
                Type: FUNCTION
             Definer: root@localhost
            Modified: 2013-06-01 12:40:31
             Created: 2013-06-01 12:40:31
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

Stored functions whose name starts with 'V':

```
SHOW FUNCTION STATUS LIKE 'V%' \G
*************************** 1. row ***************************
                  Db: test
                Name: VatCents
                Type: FUNCTION
             Definer: root@localhost
            Modified: 2013-06-01 12:40:31
             Created: 2013-06-01 12:40:31
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

Stored functions with a security type of 'DEFINER':

```
SHOW FUNCTION STATUS WHERE Security_type LIKE 'DEFINER' \G
*************************** 1. row ***************************
                  Db: test
                Name: VatCents
                Type: FUNCTION
             Definer: root@localhost
            Modified: 2013-06-01 12:40:31
             Created: 2013-06-01 12:40:31
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW GRANTS

Syntax

```
SHOW GRANTS [FOR user|role]
```

Description

The `SHOW GRANTS` statement lists privileges granted to a particular user or role.

Users

The statement lists the GRANT statement or statements that must be issued to duplicate the privileges that are granted to a MariaDB user account. The account is named using the same format as for the `GRANT` statement; for

example, '`jeffrey'@'localhost`'. If you specify only the user name part of the account name, a host name part of '`%`' is used. For additional information about specifying account names, see GRANT.

```
SHOW GRANTS FOR 'root'@'localhost';
+---------------------------------------------------------------------+
| Grants for root@localhost                                           |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+---------------------------------------------------------------------+
```

To list the privileges granted to the account that you are using to connect to the server, you can use any of the following statements:

```
SHOW GRANTS;
SHOW GRANTS FOR CURRENT_USER;
SHOW GRANTS FOR CURRENT_USER();
```

If `SHOW GRANTS FOR CURRENT_USER` (or any of the equivalent syntaxes) is used in `DEFINER` context (such as within a stored procedure that is defined with `SQL SECURITY DEFINER`), the grants displayed are those of the definer and not the invoker.

Roles

`SHOW GRANTS` can also be used to view the privileges granted to a role.

Example

```
SHOW GRANTS FOR journalist;
+-----------------------------------------+
| Grants for journalist                   |
+-----------------------------------------+
| GRANT USAGE ON *.* TO 'journalist'      |
| GRANT DELETE ON `test`.* TO 'journalist' |
+-----------------------------------------+
```

CURRENT

FOR PUBLIC

GRANT … TO PUBLIC grants privileges to all users. `SHOW GRANTS FOR PUBLIC` shows all these grants.

```
SHOW GRANTS FOR public;
+-----------------------------------------------+
| Grants for PUBLIC                             |
+-----------------------------------------------+
| GRANT ALL PRIVILEGES ON `dev_db`.* TO `PUBLIC` |
+-----------------------------------------------+
```

< 10.11

`FOR PUBLIC` is not available.

Denies

From MariaDB 13.1, privileges blocked with DENY are listed as separate `DENY` lines beside the `GRANT` lines:

```
SHOW GRANTS FOR 'alice'@'localhost';
+-------------------------------------------------------------+
| Grants for alice@localhost                                  |
+-------------------------------------------------------------+
| GRANT USAGE ON *.* TO `alice`@`localhost`                   |
| DENY SELECT ON *.* TO `alice`@`localhost`                   |
| GRANT SELECT ON `hr`.`staff` TO `alice`@`localhost`         |
| DENY UPDATE (`salary`) ON `hr`.`staff` TO `alice`@`localhost` |
+-------------------------------------------------------------+
```

`DENY` lines are only visible to a connection that holds the `SELECT` privilege on the `mysql` database. An account without it sees its own `GRANT` lines but none of its denies, so denies are not self-advertising. This applies to `SHOW GRANTS`, `SHOW GRANTS FOR CURRENT_ROLE`, `SHOW GRANTS FOR role`, and `SHOW GRANTS FOR PUBLIC` alike.

The visibility check is itself subject to denies: an account with `SELECT` on `mysql.*` but a deny on `mysql.global_priv` sees no denies, and cannot run `SHOW GRANTS` for another account.

See Also

- Authentication from MariaDB 10.4
- SHOW CREATE USER shows how the user was created.
- SHOW PRIVILEGES shows the privileges supported by MariaDB.
- DENY blocks a privilege so that no `GRANT` can restore it.
- Roles

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW INDEX_STATISTICS

Syntax

```
SHOW INDEX_STATISTICS
```

Description

The information_schema.INDEX_STATISTICS table shows statistics on index usage and makes it possible to do such things as locating unused indexes and generating the commands to remove them.

> **CURRENT**
>
> `SHOW INDEX_STATISTICS` is replaced by the generic SHOW TABLE STATISTICS statement.
>
> **< 10.1.1 / 5.3**
>
> The `SHOW INDEX_STATISTICS` statement was introduced in MariaDB 5.2 as part of the User Statistics feature. It was removed as a separate statement in MariaDB 10.1.1, but effectively replaced by the generic SHOW TABLE STATISTICS statement.

The userstat system variable must be set to 1 to activate this feature. See the User Statistics and information_schema.INDEX_STATISTICS table for more information.

Example

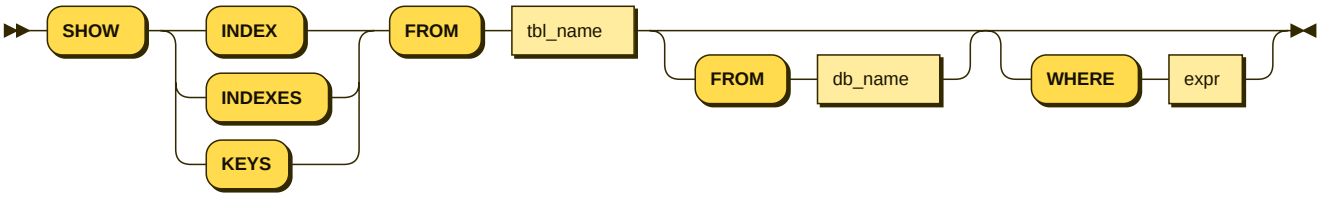
```
SHOW INDEX_STATISTICS;
+--------------+------------------+------------+-----------+
| Table_schema | Table_name       | Index_name | Rows_read |
+--------------+------------------+------------+-----------+
| test         | employees_example | PRIMARY    |         1 |
+--------------+------------------+------------+-----------+
```

This page is licensed: CC BY-SA / Gnu FDL

SHOW INDEX

Syntax

```
SHOW {INDEX | INDEXES | KEYS}
 FROM tbl_name [FROM db_name]
 [WHERE expr]
```



Railroad diagram of SHOW INDEX — equivalent to the BNF above

Description

`SHOW INDEX` returns table index information. The format resembles that of the SQLStatistics call in ODBC.

You can use `db_name.tbl_name` as an alternative to the `tbl_name FROM db_name` syntax. These two statements are equivalent:

```
SHOW INDEX FROM mytable FROM mydb;
SHOW INDEX FROM mydb.mytable;
```

`SHOW KEYS` and `SHOW INDEXES` are synonyms for `SHOW INDEX`.

You can also list a table's indexes with the mariadb-show command:

```
mariadb-show -k db_name tbl_name
```

The information_schema.STATISTICS table stores similar information.

The following fields are returned by `SHOW INDEX`.

Field	Description
Table	Table name
Non_unique	1 if the index permits duplicate values, 0 if values must be unique.
Key_name	Index name. The primary key is always named PRIMARY.
Seq_in_index	The column's sequence in the index, beginning with 1.
Column_name	Column name.
Collation	Either A, if the column is sorted in ascending order in the index, or NULL if it's not sorted.

Field	Description
Cardinality	Estimated number of unique values in the index. The cardinality statistics are calculated at various times, and can help the optimizer make improved decisions.
Sub_part	NULL if the entire column is included in the index, or the number of included characters if not.
Packed	NULL if the index is not packed, otherwise how the index is packed.
Null	NULL if NULL values are permitted in the column, an empty string if NULLs are not permitted.
Index_type	The index type, which can be BTREE, FULLTEXT, HASH or RTREE. See Storage Engine Index Types.
Comment	Other information, such as whether the index is disabled.
Index_comment	Contents of the COMMENT attribute when the index was created.
Ignored	Whether or not an index will be ignored by the optimizer. See Ignored Indexes. From MariaDB 10.6.0.

The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

Examples

```
CREATE TABLE IF NOT EXISTS `employees_example` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(30) NOT NULL,
  `last_name` VARCHAR(40) NOT NULL,
  `position` VARCHAR(25) NOT NULL,
  `home_address` VARCHAR(50) NOT NULL,
  `home_phone` VARCHAR(12) NOT NULL,
  `employee_code` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `employee_code` (`employee_code`),
  KEY `first_name` (`first_name`,`last_name`)
) ENGINE=Aria;

INSERT INTO `employees_example` (`first_name`, `last_name`, `position`, `home_address`, `home_phone`,
`employee_code`)
  VALUES
  ('Mustapha', 'Mond', 'Chief Executive Officer', '692 Promiscuous Plaza', '326-555-3492', 'MM1'),
  ('Henry', 'Foster', 'Store Manager', '314 Savage Circle', '326-555-3847', 'HF1'),
  ('Bernard', 'Marx', 'Cashier', '1240 Ambient Avenue', '326-555-8456', 'BM1'),
  ('Lenina', 'Crowne', 'Cashier', '281 Bumblepuppy Boulevard', '328-555-2349', 'LC1'),
  ('Fanny', 'Crowne', 'Restocker', '1023 Bokanovsky Lane', '326-555-6329', 'FC1'),
  ('Helmholtz', 'Watson', 'Janitor', '944 Soma Court', '329-555-2478', 'HW1');
```

```
SHOW INDEXES FROM employees_example\G
*************************** 1. row ***************************
        Table: employees_example
   Non_unique: 0
     Key_name: PRIMARY
 Seq_in_index: 1
  Column_name: id
    Collation: A
  Cardinality: 6
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
      Ignored: NO
*************************** 2. row ***************************
        Table: employees_example
   Non_unique: 0
     Key_name: employee_code
 Seq_in_index: 1
  Column_name: employee_code
    Collation: A
  Cardinality: 6
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
      Ignored: NO
*************************** 3. row ***************************
        Table: employees_example
   Non_unique: 1
     Key_name: first_name
 Seq_in_index: 1
  Column_name: first_name
    Collation: A
  Cardinality: NULL
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
      Ignored: NO
*************************** 4. row ***************************
        Table: employees_example
   Non_unique: 1
     Key_name: first_name
 Seq_in_index: 2
  Column_name: last_name
    Collation: A
  Cardinality: NULL
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
      Ignored: NO
```

See Also

- Ignored Indexes
- SHOW INDEX_STATISTICS

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW LOCALES

Syntax

```
SHOW LOCALES
```

Description

`SHOW LOCALES` was introduced as part of the Information Schema plugin extension.

`SHOW LOCALES` is used to return locales information as part of the Locales plugin. It requires this plugin to be installed to work.

While the information_schema.LOCALES table has 8 columns, the `SHOW LOCALES` statement will only display 4 of them:

Example

```
SHOW LOCALES;
+-----+-------+-------------------------------------+------------------------+
| Id  | Name  | Description                         | Error_Message_Language |
+-----+-------+-------------------------------------+------------------------+
|   0 | en_US | English - United States             | english                |
|   1 | en_GB | English - United Kingdom            | english                |
|   2 | ja_JP | Japanese - Japan                    | japanese               |
|   3 | sv_SE | Swedish - Sweden                    | swedish                |
...
```

This page is licensed: CC BY-SA / Gnu FDL

SHOW OPEN TABLES

Syntax

```
SHOW OPEN TABLES [FROM db_name]
    [LIKE 'pattern' | WHERE expr]
```

Description

`SHOW OPEN TABLES` lists the non-`TEMPORARY` tables that are currently open in the table cache. See table-cache.html.

The `FROM` and `LIKE` clauses may be used.

The `FROM` clause, if present, restricts the tables shown to those present in the `db_name` database.

The `LIKE` clause, if present on its own, indicates which table names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

The following information is returned:

Column	Description
Database	Database name.
Name	Table name.
In_use	Number of table instances being used.
Name_locked	1 if the table is name-locked, e.g. if it is being dropped or renamed, otherwise 0.

CURRENT

`LOCK TABLE... WRITE` acquires a strong MDL lock, and concurrent connections will wait on this MDL lock, so any subsequent `LOCK TABLE... WRITE` will not increment `In_use`.

< 5.5

Before MariaDB 5.5, each use of, for example, LOCK TABLE … WRITE would increment `In_use` for that table. With the implementation of the metadata locking improvements in MariaDB 5.5, `LOCK TABLE... WRITE` acquires a strong MDL lock, and concurrent connections will wait on this MDL lock, so any subsequent `LOCK TABLE... WRITE` will not increment `In_use`.

Example

```
SHOW OPEN TABLES;
+----------+--------------------------+--------+-------------+
| Database | Table                    | In_use | Name_locked |
+----------+--------------------------+--------+-------------+
...
| test     | xjson                    |      0 |           0 |
| test     | jauthor                  |      0 |           0 |
| test     | locks                    |      1 |           0 |
...
+----------+--------------------------+--------+-------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW PACKAGE BODY STATUS

Syntax

```
SHOW PACKAGE BODY STATUS
    [LIKE 'pattern' | WHERE expr]
```

Description

The `SHOW PACKAGE BODY STATUS` statement returns characteristics of stored package bodies (implementations), such as the database, name, type, creator, creation and modification dates, and character set information. A similar statement, SHOW PACKAGE STATUS, displays information about stored package specifications.

The `LIKE` clause, if present, indicates which package names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

The ROUTINES table in the INFORMATION_SCHEMA database contains more detailed information.

Examples

```
SHOW PACKAGE BODY STATUS LIKE 'pkg1'\G
*************************** 1. row ***************************
                  Db: test
                Name: pkg1
                Type: PACKAGE BODY
             Definer: root@localhost
            Modified: 2018-02-27 14:44:14
             Created: 2018-02-27 14:44:14
       Security_type: DEFINER
             Comment: This is my first package body
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

See Also

- SHOW PACKAGE STATUS
- SHOW CREATE PACKAGE BODY
- CREATE PACKAGE BODY
- DROP PACKAGE BODY
- Oracle SQL_MODE

This page is licensed: CC BY-SA / Gnu FDL

SHOW PACKAGE STATUS

Syntax

```
SHOW PACKAGE STATUS
    [LIKE 'pattern' | WHERE expr]
```

Description

The `SHOW PACKAGE STATUS` statement returns characteristics of stored package specifications, such as the database, name, type, creator, creation and modification dates, and character set information. A similar statement, SHOW PACKAGE BODY STATUS, displays information about stored package bodies (i.e. implementations).

The `LIKE` clause, if present, indicates which package names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

The ROUTINES table in the INFORMATION_SCHEMA database contains more detailed information.

Examples

```
SHOW PACKAGE STATUS LIKE 'pkg1'\G
*************************** 1. row ***************************
                  Db: test
                Name: pkg1
                Type: PACKAGE
             Definer: root@localhost
            Modified: 2018-02-27 14:38:15
             Created: 2018-02-27 14:38:15
       Security_type: DEFINER
             Comment: This is my first package
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

See Also

- SHOW PACKAGE BODY
- SHOW CREATE PACKAGE
- CREATE PACKAGE
- DROP PACKAGE
- Oracle SQL_MODE

This page is licensed: CC BY-SA / Gnu FDL

SHOW PLUGINS SONAME

Syntax

```
SHOW PLUGINS SONAME { library | LIKE 'pattern' | WHERE expr };
```

Description

`SHOW PLUGINS SONAME` displays information about compiled-in and all server plugins in the plugin_dir directory, including plugins that haven't been installed.

Examples

```
SHOW PLUGINS SONAME 'ha_example.so';
+----------+--------------+----------------+--------------+---------+
| Name     | Status       | Type           | Library      | License |
+----------+--------------+----------------+--------------+---------+
| EXAMPLE  | NOT INSTALLED | STORAGE ENGINE | ha_example.so | GPL     |
| UNUSABLE | NOT INSTALLED | DAEMON         | ha_example.so | GPL     |
+----------+--------------+----------------+--------------+---------+
```

There is also a corresponding information_schema table, called ALL_PLUGINS, which contains more complete information.

This page is licensed: CC BY-SA / Gnu FDL

SHOW PLUGINS

Syntax

```
SHOW PLUGINS;
```

Description

`SHOW PLUGINS` displays information about installed plugins. The `Library` column indicates the plugin library - if it is `NULL`, the plugin is built-in and cannot be uninstalled.

The PLUGINS table in the `information_schema` database contains more detailed information.

For specific information about storage engines (a particular type of plugin), see the information_schema.ENGINES table and the SHOW ENGINES statement.

Examples

```
SHOW PLUGINS;
+----------------------------+----------+--------------------+-------------+---------+
| Name                       | Status   | Type               | Library     | License |
+----------------------------+----------+--------------------+-------------+---------+
| binlog                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| mysql_native_password      | ACTIVE   | AUTHENTICATION     | NULL        | GPL     |
| mysql_old_password         | ACTIVE   | AUTHENTICATION     | NULL        | GPL     |
| MRG_MyISAM                 | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| MyISAM                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| CSV                        | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| MEMORY                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| FEDERATED                  | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| PERFORMANCE_SCHEMA         | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| Aria                       | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| InnoDB                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| INNODB_TRX                 | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
...
| INNODB_SYS_FOREIGN         | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_SYS_FOREIGN_COLS    | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| SPHINX                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| ARCHIVE                    | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| BLACKHOLE                  | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| FEEDBACK                   | DISABLED | INFORMATION SCHEMA | NULL        | GPL     |
| partition                  | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| pam                        | ACTIVE   | AUTHENTICATION     | auth_pam.so | GPL     |
+----------------------------+----------+--------------------+-------------+---------+
```

See Also

- List of Plugins
- Plugin Overview
- INFORMATION_SCHEMA.PLUGINS Table
- INSTALL PLUGIN
- INFORMATION_SCHEMA.ALL_PLUGINS Table (all plugins, installed or not)
- INSTALL SONAME
- UNINSTALL PLUGIN
- UNINSTALL SONAME

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW PRIVILEGES

Syntax

```
SHOW PRIVILEGES
```

Description

CURRENT

`SHOW PRIVILEGES` shows the list of system privileges that the MariaDB server supports. The exact list of privileges depends on the version of your server.

< 10.5.2 / 10.4.13 / 10.3.23

`SHOW PRIVILEGES` shows the list of system privileges that the MariaDB server supports. The exact list of privileges depends on the version of your server.

> Note that before MariaDB 10.3.23, MariaDB 10.4.13 and MariaDB 10.5.2 , the Delete history privilege displays as `Delete versioning rows` (MDEV-20382).

Example

> **Note:**
>
> The output is for MariaDB version from 10.5.9. In previous versions, it might look differently.

```
SHOW PRIVILEGES;
+--------------------------+---------------------------------------+-----------------------------------
--------------------------------+
| Privilege                | Context                               | Comment
|
+--------------------------+---------------------------------------+-----------------------------------
--------------------------------+
| Alter                    | Tables                                | To alter the table
|
| Alter routine            | Functions,Procedures                  | To alter or drop stored
functions/procedures                        |
| Create                   | Databases,Tables,Indexes              | To create new databases and tables
|
| Create routine           | Databases                             | To use CREATE FUNCTION/PROCEDURE
|
| Create temporary tables  | Databases                             | To use CREATE TEMPORARY TABLE
|
| Create view              | Tables                                | To create new views
|
| Create user              | Server Admin                          | To create new users
|
| Delete                   | Tables                                | To delete existing rows
|
| Delete history           | Tables                                | To delete versioning table
historical rows                           |
| Drop                     | Databases,Tables                      | To drop databases, tables, and views
|
| Event                    | Server Admin                          | To create, alter, drop and execute
events                          |
| Execute                  | Functions,Procedures                  | To execute stored routines
|
| File                     | File access on server                 | To read and write files on the
server                                  |
| Grant option             | Databases,Tables,Functions,Procedures | To give to other users those
privileges you possess                    |
| Index                    | Tables                                | To create or drop indexes
|
| Insert                   | Tables                                | To insert data into tables
|
| Lock tables              | Databases                             | To use LOCK TABLES (together with
SELECT privilege)                |
| Process                  | Server Admin                          | To view the plain text of currently
executing queries               |
| Proxy                    | Server Admin                          | To make proxy user possible
|
| References               | Databases,Tables                      | To have references on tables
|
| Reload                   | Server Admin                          | To reload or refresh tables, logs
and privileges                  |
| Binlog admin             | Server                                | To purge binary logs
|
| Binlog monitor           | Server                                | To use SHOW BINLOG STATUS and SHOW
BINARY LOG                      |
| Binlog replay            | Server                                | To use BINLOG (generated by mariadb-
binlog)                           |
| Replication master admin | Server                                | To monitor connected slaves
|
| Replication slave admin  | Server                                | To start/stop slave and apply binlog
events                           |
| Slave monitor            | Server                                | To use SHOW SLAVE STATUS and SHOW
RELAYLOG EVENTS                 |
| Replication slave        | Server Admin                          | To read binary log events from the
master                          |
| Select                   | Tables                                | To retrieve rows from table
|
| Show databases           | Server Admin                          | To see all databases with SHOW
DATABASES                               |
| Show view                | Tables                                | To see views with SHOW CREATE VIEW
```

```
|
| Shutdown                | Server Admin                           | To shut down the server
|
| Super                   | Server Admin                           | To use KILL thread, SET GLOBAL,
CHANGE MASTER, etc.               |
| Trigger                 | Tables                                 | To use triggers
|
| Create tablespace       | Server Admin                           | To create/alter/drop tablespaces
|
| Update                  | Tables                                 | To update existing rows
|
| Set user                | Server                                 | To create views and stored routines
with a different definer       |
| Federated admin         | Server                                 | To execute the CREATE SERVER, ALTER
SERVER, DROP SERVER statements |
| Connection admin        | Server                                 | To bypass connection limits and kill
other users' connections      |
| Read_only admin         | Server                                 | To perform write operations even if
@@read_only=ON                 |
| Usage                   | Server Admin                           | No privileges - allow connect only
|
+-------------------------+----------------------------------------+-----------------------------------
--------------------------------+
41 rows in set (0.000 sec)
```

See Also

- SHOW CREATE USER shows how the user was created.
- SHOW GRANTS shows the `GRANTS/PRIVILEGES` for a user.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW PROCEDURE CODE

Syntax

```
SHOW PROCEDURE CODE proc_name
```

Description

This statement is a MariaDB extension that is available only for servers that have been built with debugging support. It displays a representation of the internal implementation of the named stored procedure. A similar statement, SHOW FUNCTION CODE, displays information about stored functions.

Both statements require that you be the owner of the routine or have SELECT access to the mysql.proc table.

If the named routine is available, each statement produces a result set. Each row in the result set corresponds to one “instruction” in the routine. The first column is Pos, which is an ordinal number beginning with 0. The second column is Instruction, which contains an SQL statement (usually changed from the original source), or a directive which has meaning only to the stored-routine handler.

Examples

```
DELIMITER //

CREATE PROCEDURE p1 ()
  BEGIN
    DECLARE fanta INT DEFAULT 55;
    DROP TABLE t2;
    LOOP
      INSERT INTO t3 VALUES (fanta);
      END LOOP;
  END//
Query OK, 0 rows affected (0.00 sec)

SHOW PROCEDURE CODE p1//
+-----+----------------------------------------+
| Pos | Instruction                            |
+-----+----------------------------------------+
|   0 | set fanta@0 55                         |
|   1 | stmt 9 "DROP TABLE t2"                 |
|   2 | stmt 5 "INSERT INTO t3 VALUES (fanta)" |
|   3 | jump 2                                 |
+-----+----------------------------------------+
```

See Also

- Stored Procedure Overview
- CREATE PROCEDURE
- ALTER PROCEDURE
- DROP PROCEDURE
- SHOW CREATE PROCEDURE
- SHOW PROCEDURE STATUS
- Stored Routine Privileges
- Information Schema ROUTINES Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW PROCEDURE STATUS

Syntax

```
SHOW PROCEDURE STATUS
    [LIKE 'pattern' | WHERE expr]
```

Description

This statement is a MariaDB extension. It returns characteristics of a stored procedure, such as the database, name, type, creator, creation and modification dates, and character set information. A similar statement, SHOW FUNCTION STATUS, displays information about stored functions.

The `LIKE` clause, if present, indicates which procedure or function names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

The ROUTINES table in the INFORMATION_SCHEMA database contains more detailed information.

Examples

```
SHOW PROCEDURE STATUS LIKE 'p1'\G
*************************** 1. row ***************************
                  Db: test
                Name: p1
                Type: PROCEDURE
             Definer: root@localhost
            Modified: 2010-08-23 13:23:03
             Created: 2010-08-23 13:23:03
       Security_type: DEFINER
             Comment:
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
```

See Also

- Stored Procedure Overview
- CREATE PROCEDURE
- ALTER PROCEDURE
- DROP PROCEDURE
- SHOW CREATE PROCEDURE
- Stored Routine Privileges
- Information Schema ROUTINES Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW PROCESSLIST

Syntax

```
SHOW [FULL] PROCESSLIST
```

Description

`SHOW PROCESSLIST` shows which threads are running. You can also get this information from the information_schema.PROCESSLIST table or the mariadb-admin processlist command. If you have the PROCESS privilege, you can see all threads. Otherwise, you can see only your own threads (that is, threads associated with the MariaDB account that you are using). If you do not use the `FULL` keyword, only the first 100 characters of each statement are shown in the Info field.

The columns shown in `SHOW PROCESSLIST` are:

Name	Description
Id	The client's thread ID.
User	The username associated with the process.
Host	The host the client is connected to.
db	The default database of the process (`NULL` if no default).
Command	The command type. See Thread Command Values.
Time	The amount of time, in seconds, the process has been in its current state.
State	See Thread States.
Info	The statement being executed.
Progress	The total progress of the process (0-100%) (see Progress Reporting).

The information_schema.PROCESSLIST table contains a number of additional columns. See `TIME_MS` column in information_schema.PROCESSLIST for differences in the `TIME` column between MariaDB and MySQL.

Note that the `PROGRESS` field from the information schema, and the `PROGRESS` field from `SHOW PROCESSLIST` display different results. `SHOW PROCESSLIST` shows the total progress, while the information schema shows the progress for the current stage only.

Threads can be killed using their `thread_id` or their `query_id`, with the KILL statement.

Since queries on this table are locking, if the performance_schema is enabled, you may want to query the THREADS table instead.

Examples

```
SHOW PROCESSLIST;
+----+-----------------+-----------+------+---------+------+-----------------------+------------------+-
---------+
| Id | User            | Host      | db   | Command | Time | State                 | Info             |
Progress |
+----+-----------------+-----------+------+---------+------+-----------------------+------------------+-
---------+
|  2 | event_scheduler | localhost | NULL | Daemon  | 2693 | Waiting on empty queue | NULL             |
0.000 |
|  4 | root            | localhost | NULL | Query   |    0 | Table lock             | SHOW PROCESSLIST |
0.000 |
+----+-----------------+-----------+------+---------+------+-----------------------+------------------+-
---------+
```

See Also

CONNECTION_ID()

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW PROFILE

Syntax

```
SHOW PROFILE [type [, type] ... ]
    [FOR QUERY n]
    [LIMIT row_count [OFFSET offset]]

type:
    ALL
  | BLOCK IO
  | CONTEXT SWITCHES
  | CPU
  | IPC
  | MEMORY
  | PAGE FAULTS
  | SOURCE
  | SWAPS
```

Description

The `SHOW PROFILE` and SHOW PROFILES statements display profiling information that indicates resource usage for statements executed during the course of the current session.

Profiling is controlled by the profiling session variable, which has a default value of `0` (`OFF`). Profiling is enabled by setting profiling to `1` or `ON`:

```
SET profiling = 1;
```

`SHOW PROFILES` displays a list of the most recent statements sent to the master. The size of the list is controlled by the profiling_history_size session variable, which has a default value of `15`. The maximum value is `100`. Setting the value to `0` has the practical effect of disabling profiling.

All statements are profiled except `SHOW PROFILES` and `SHOW PROFILE`, so you will find neither of those statements in the profile list. Malformed statements are profiled. For example, `SHOW PROFILING` is an illegal statement, and a syntax error occurs if you try to execute it, but it will show up in the profiling list.

`SHOW PROFILE` displays detailed information about a single statement. Without the `FOR QUERY n` clause, the output pertains to the most recently executed statement. If `FOR QUERY n` is included, `SHOW PROFILE` displays information for statement *n*. The values of *n* correspond to the `Query_ID` values displayed by `SHOW PROFILES`.

The `LIMIT row_count` clause may be given to limit the output to *row_count* rows. If `LIMIT` is given, `OFFSET offset` may be added to begin the output offset rows into the full set of rows.

By default, `SHOW PROFILE` displays Status and Duration columns. The Status values are like the State values displayed by SHOW PROCESSLIST (see General Thread States), although there might be some minor differences in interpretation for the two statements for some status values.

Optional type values may be specified to display specific additional types of information:

- `ALL` displays all information
- `BLOCK IO` displays counts for block input and output operations
- `CONTEXT SWITCHES` displays counts for voluntary and involuntary context switches
- `CPU` displays user and system CPU usage times
- `IPC` displays counts for messages sent and received
- `MEMORY` is not currently implemented
- `PAGE FAULTS` displays counts for major and minor page faults
- `SOURCE` displays the names of functions from the source code, together with the name and line number of the file in which the function occurs
- `SWAPS` displays swap counts

Profiling is enabled per session. When a session ends, its profiling information is lost.

The information_schema.PROFILING table contains similar information.

Examples

```
SELECT @@profiling;
+-------------+
| @@profiling |
+-------------+
|           0 |
+-------------+

SET profiling = 1;

USE test;

DROP TABLE IF EXISTS t1;

CREATE TABLE T1 (id INT);

SHOW PROFILES;
+----------+------------+--------------------------+
| Query_ID | Duration   | Query                    |
+----------+------------+--------------------------+
|        1 | 0.00009200 | SELECT DATABASE()        |
|        2 | 0.00023800 | show databases           |
|        3 | 0.00018900 | show tables              |
|        4 | 0.00014700 | DROP TABLE IF EXISTS t1  |
|        5 | 0.24476900 | CREATE TABLE T1 (id INT) |
+----------+------------+--------------------------+

SHOW PROFILE;
+----------------------+----------+
| Status               | Duration |
+----------------------+----------+
| starting             | 0.000042 |
| checking permissions | 0.000044 |
| creating table       | 0.244645 |
| After create         | 0.000013 |
| query end            | 0.000003 |
| freeing items        | 0.000016 |
| logging slow query   | 0.000003 |
| cleaning up          | 0.000003 |
+----------------------+----------+

SHOW PROFILE FOR QUERY 4;
+--------------------+----------+
| Status             | Duration |
+--------------------+----------+
| starting           | 0.000126 |
| query end          | 0.000004 |
| freeing items      | 0.000012 |
| logging slow query | 0.000003 |
| cleaning up        | 0.000002 |
+--------------------+----------+

SHOW PROFILE CPU FOR QUERY 5;
+----------------------+----------+----------+------------+
| Status               | Duration | CPU_user | CPU_system |
+----------------------+----------+----------+------------+
| starting             | 0.000042 | 0.000000 |   0.000000 |
| checking permissions | 0.000044 | 0.000000 |   0.000000 |
| creating table       | 0.244645 | 0.000000 |   0.000000 |
| After create         | 0.000013 | 0.000000 |   0.000000 |
| query end            | 0.000003 | 0.000000 |   0.000000 |
| freeing items        | 0.000016 | 0.000000 |   0.000000 |
| logging slow query   | 0.000003 | 0.000000 |   0.000000 |
| cleaning up          | 0.000003 | 0.000000 |   0.000000 |
+----------------------+----------+----------+------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW PROFILES

Syntax

```
SHOW PROFILES
```

Description

The `SHOW PROFILES` statement displays profiling information that indicates resource usage for statements executed during the course of the current session. It is used together with SHOW PROFILE.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW QUERY_RESPONSE_TIME

Syntax

```
SHOW QUERY_RESPONSE_TIME
```

Description

It is possible to use `SHOW QUERY_RESPONSE_TIME` as an alternative for retrieving information from the QUERY_RESPONSE_TIME plugin.

This was introduced as part of the Information Schema plugin extension.

Example

```
SHOW QUERY_RESPONSE_TIME;
+----------------+-------+----------------+
| Time           | Count | Total          |
+----------------+-------+----------------+
|       0.000001 |     0 |       0.000000 |
|       0.000010 |    17 |       0.000094 |
|       0.000100 |  4301 |       0.236555 |
|       0.001000 |  1499 |       0.824450 |
|       0.010000 | 14851 |      81.680502 |
|       0.100000 |  8066 |     443.635693 |
|       1.000000 |     0 |       0.000000 |
|      10.000000 |     0 |       0.000000 |
|     100.000000 |     1 |      55.937094 |
|    1000.000000 |     0 |       0.000000 |
|   10000.000000 |     0 |       0.000000 |
|  100000.000000 |     0 |       0.000000 |
| 1000000.000000 |     0 |       0.000000 |
| TOO LONG       |     0 | TOO LONG       |
+----------------+-------+----------------+
```

This page is licensed: CC BY-SA / Gnu FDL

SHOW RELAYLOG EVENTS

> **Note:**
>
> The terms *master* and *slave* have historically been used in replication, and MariaDB has begun the process of adding *primary* and *replica* synonyms. The old terms will continue to be used to maintain backward compatibility - see MDEV-18777 to follow progress on this effort.

Syntax

```
SHOW RELAYLOG ['connection_name'] EVENTS
    [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]
    [ FOR CHANNEL 'channel_name']
```

Description

On replicas, this command shows the events in the relay log. If `'log_name'` is not specified, the first relay log is shown.

Syntax for the `LIMIT` clause is the same as for SELECT … LIMIT.

Using the `LIMIT` clause is highly recommended because the `SHOW RELAYLOG EVENTS` command returns the complete contents of the relay log, which can be quite large.

This command does not return events related to setting user and system variables. If you need those, use mariadb-binlog.

On the primary, this command does nothing.

> **CURRENT**
>
> Requires the REPLICA MONITOR privilege.
>
> **< 10.5.9 / 10.5.2 / 10.5.1**
>
> Requires the REPLICA MONITOR privilege (>= MariaDB 10.5.9), the REPLICATION SLAVE ADMIN privilege (>= MariaDB 10.5.2) or the REPLICATION SLAVE privilege (<= MariaDB 10.5.1).

connection_name

If there is only one nameless primary, or the default primary (as specified by the default_master_connection system variable) is intended, `connection_name` can be omitted. If provided, the `SHOW RELAYLOG` statement will apply to the specified primary. `connection_name` is case-insensitive.

> **CURRENT**
>
> The `FOR CHANNEL` keyword was added for MySQL compatibility. This is identical as using the `channel_name` directly after `SHOW RELAYLOG`.
>
> **< 10.7.0**
>
> `FOR CHANNEL` is not available.

This page is licensed: CC BY-SA / Gnu FDL

SHOW REPLICA HOSTS

Syntax

```
SHOW { SLAVE | REPLICA } HOSTS
```

Description

This command is run on the primary and displays a list of replicas that are currently registered with it. The output looks like this:

```
SHOW SLAVE HOSTS;
+------------+-----------+------+-----------+
| Server_id  | Host      | Port | Master_id |
+------------+-----------+------+-----------+
|  192168010 | iconnect2 | 3306 | 192168011 |
| 1921680101 | athena    | 3306 | 192168011 |
+------------+-----------+------+-----------+
```

`Server_id`: The unique server ID of the replica server, as configured in the server's option file, or on the command line with –server-id=value.

CURRENT

`Host`: The host name of the replica server, as configured in the server's option file, or on the command line with `--report-host=host_name` (note that this can differ from the machine name as configured in the operating system). If a replica doesn't configure `--report-host` explicitly, the value for the `Host` column is automatically extracted using the network connection's host name or IP address.

< 10.5

`Host`: The host name of the replica server, as configured in the server's option file, or on the command line with `--report-host=host_name` (note that this can differ from the machine name as configured in the operating system). If a replica doesn't configure `--report-host` explicitly, the value for the `Host` column is automatically extracted using the network connection's host name or IP address. The Host value is left blank if a replica's `--report-host` parameter is not configured.

`Port`: The port the replica server is listening on.

`Master_id`: The unique server ID of the primary server that the replica server is replicating from.

CURRENT

Requires the REPLICATION MASTER ADMIN privilege.

< 10.5.1

Requires the REPLICATION SLAVE privilege.

SHOW REPLICA HOSTS

CURRENT

`SHOW REPLICA HOSTS` is an alias for `SHOW SLAVE HOSTS`.

< 10.5.1

`SHOW REPLICA HOSTS` is not available, use `SHOW SLAVE HOSTS` instead.

See Also

- MariaDB replication
- Replication threads
- SHOW PROCESSLIST. In SHOW PROCESSLIST output on the primary, connected replicas are identified by `Binlog Dump`

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW REPLICA STATUS

Syntax

```
SHOW { SLAVE | REPLICA} ["connection_name"] STATUS [FOR CHANNEL "connection_name"]
```

or

```
SHOW ALL { SLAVES | REPLICAS } STATUS
```

Description

This statement is to be run on a replica and provides status information on essential parameters of the replica threads.

> **CURRENT**
>
> This statement requires the REPLICA MONITOR privilege.
>
> **< 10.5.9**
>
> This statement requires the REPLICA MONITOR privilege.
>
> **< 10.5.2**
>
> This statement requires the REPLICATION SLAVE ADMIN privilege.

Multi-Source

The `ALL` and `"connection_name"` options allow you to connect to many primaries at the same time.

> **CURRENT**
>
> `ALL SLAVES` or `ALL REPLICAS` gives you a list of all connections to the primary nodes.
>
> **< 10.5.1**
>
> `ALL SLAVES` gives you a list of all connections to the primary nodes.

The rows are sorted according to `Connection_name`.

If you specify a `connection_name`, you only get the information about that connection. If `connection_name` is not used, then the name set by `default_master_connection` is used. If the connection name doesn't exist you will get an error: `There is no master connection for 'xxx'`.

MariaDB starting with 10.7.0

The `FOR CHANNEL` keyword was added for MySQL compatibility. This is identical to using the channel_name directly after `SHOW SLAVE`.

Column Descriptions

The order in which the columns appear depends on the MariaDB version. This means that extracting a column value is best done by comparing the field name instead of using a fixed offset into the row.

> **CURRENT**
>
> These columns can also be viewed/extracted from the INFORMATION_SCHEMA.SLAVE_STATUS table.
>
> **< 11.6.0**

These columns cannot be viewed/extracted from the INFORMATION_SCHEMA.SLAVE_STATUS table.

CURRENT

Connection_name: Name of the primary connection. Returned with SHOW ALL SLAVES/REPLICAS STATUS only.

Slave_SQL_State: State of SQL thread. Returned with SHOW ALL SLAVES/REPLICAS STATUS only. See Replica SQL Thread States. Slave_IO_State: State of I/O thread. See Replica I/O Thread States.

< 10.5.1

Connection_name: Name of the primary connection. Returned with SHOW ALL REPLICAS STATUS only.

Slave_SQL_State: State of SQL thread. Returned with SHOW ALL REPLICAS STATUS only. See Replica SQL Thread States. Slave_IO_State: State of I/O thread. See Replica I/O Thread States.

Master_host: Primary host that the replica is connected to.

Master_user: Account user name being used to connect to the primary.

Master_port: The port being used to connect to the primary.

Connect_Retry: Time in seconds between retries to connect. The default is 60. The CHANGE MASTER TO statement can set this. The master-retry-count option determines the maximum number of reconnection attempts.

Master_Log_File: Name of the primary binary log file that the I/O thread is currently reading from.

Read_Master_Log_Pos: Position up to which the I/O thread has read in the current primary binary log file.

Relay_Log_File: Name of the relay log file that the SQL thread is currently processing.

Relay_Log_Pos: Position up to which the SQL thread has finished processing in the current relay log file.

Relay_Master_Log_File: Name of the primary binary log file that contains the most recent event executed by the SQL thread.

Slave_IO_Running: Whether the replica I/O thread is running and connected (Yes), running but not connected to a primary (Connecting) or not running (No).

Slave_SQL_Running: Whether or not the SQL thread is running.

Replicate_Do_DB: Databases specified for replicating with the replicate_do_db option.

Replicate_Ignore_DB: Databases specified for ignoring with the replicate_ignore_db option.

Replicate_Do_Table: Tables specified for replicating with the replicate_do_table option.

Replicate_Ignore_Table: Tables specified for ignoring with the replicate_ignore_table option.

Replicate_Wild_Do_Table: Tables specified for replicating with the replicate_wild_do_table option.

Replicate_Wild_Ignore_Table: Tables specified for ignoring with the replicate_wild_ignore_table option.

Last_Errno: Alias for Last_SQL_Errno (see below)

Last_Error: Alias for Last_SQL_Error (see below)

Skip_Counter: Number of events that a replica skips from the primary, as recorded in the sql_slave_skip_counter system variable.

Exec_Master_Log_Pos: Position up to which the SQL thread has processed in the current primary binary log file. Can be used to start a new replica from a current replica with the CHANGE MASTER TO … MASTER_LOG_POS

option.

Relay_Log_Space: Total size of all relay log files combined.

Until_Condition: One of four possible values: None, Primary, Relay, or Gtid, depending on the respective START REPLICA UNTIL condition.

Until_Log_File: The MASTER_LOG_FILE value of the START REPLICA UNTIL condition.

Until_Log_Pos: The MASTER_LOG_POS value of the START REPLICA UNTIL condition.

Master_SSL_Allowed: Whether an SSL connection to the primary is permitted (Yes) or not permitted (No). The value reflects the `MASTER_SSL` option configured via CHANGE MASTER TO and is independent of the runtime `have_ssl` state. A third value, Ignored, is emitted only on MariaDB builds compiled without SSL/TLS support (`HAVE_OPENSSL` undefined at build time); this does not apply to standard packaged builds, which always include OpenSSL, WolfSSL, or GnuTLS support.

Master_SSL_CA_File: The MASTER_SSL_CA option of the CHANGE MASTER TO statement.

Master_SSL_CA_Path: The MASTER_SSL_CAPATH option of the CHANGE MASTER TO statement.

Master_SSL_Cert: The MASTER_SSL_CERT option of the CHANGE MASTER TO statement.

Master_SSL_Cipher: The MASTER_SSL_CIPHER option of the CHANGE MASTER TO statement.

Master_SSL_Key: The MASTER_SSL_KEY option of the CHANGE MASTER TO statement.

CURRENT

Seconds_Behind_Master: Difference between the timestamp logged on the primary for the event that the replica is currently processing, and the current timestamp on the replica. Zero if the replica is not currently processing an event. With serial replication, seconds_behind_master is updated when the SQL thread begins executing a transaction. With parallel replication, seconds_behind_master is updated only after transactions commit. As a special case, the parallel replica additionally updates `seconds_behind_master` when the first transaction received after idling is queued to a worker for execution, to provide a reliable initial value for the duration until a transaction commits.

Additional behavior to be aware of:

1. `Seconds_Behind_Master` will update for ignored events, e.g. those skipped due to sql_slave_skip_counter.
2. On the serial replica, transactions with prior timestamps can update `Seconds_Behind_Master` such that it can go backwards, though this is not true for the parallel replica.
3. When configured with MASTER_DELAY, as a replicated transaction begins executing (i.e. on a serial or post-idle parallel replica), `Seconds_Behind_Master` will update before delaying, and while delaying occurs will grow to encompass the configured value.
4. There is a known issue, tracked by MDEV-17516, such that `Seconds_Behind_Master` will initially present as 0 on replica restart until a replicated transaction begins executing, even if the last replica session was lagging behind when stopped.

< 10.11.12 / 10.10.3 / 10.9.5 / 10.8.7 / 10.6.12 / 10.5.19

Seconds_Behind_Master: Difference between the timestamp logged on the primary for the event that the replica is currently processing, and the current timestamp on the replica. Zero if the replica is not currently processing an event. With serial replication, seconds_behind_master is updated when the SQL thread begins executing a transaction. With parallel replication, seconds_behind_master is updated only after transactions commit.

Additional behavior to be aware of:

1. `Seconds_Behind_Master` will update for ignored events, e.g. those skipped due to sql_slave_skip_counter.

2. On the serial replica, transactions with prior timestamps can update `Seconds_Behind_Master` such that it can go backwards, though this is not true for the parallel replica.
3. When configured with MASTER_DELAY, as a replicated transaction begins executing (i.e. on a serial or post-idle parallel replica), `Seconds_Behind_Master` will update before delaying, and while delaying occurs will grow to encompass the configured value.
4. There is a known issue, tracked by MDEV-17516, such that `Seconds_Behind_Master` will initially present as 0 on replica restart until a replicated transaction begins executing, even if the last replica session was lagging behind when stopped.

Master_SSL_Verify_Server_Cert: The MASTER_SSL_VERIFY_SERVER_CERT option of the CHANGE MASTER TO statement.

Last_IO_Errno: Error code of the most recent error that caused the I/O thread to stop (also recorded in the replica's error log). 0 means no error. RESET REPLICA or RESET MASTER will reset this value.

Last_IO_Error: Error message of the most recent error that caused the I/O thread to stop (also recorded in the replica's error log). An empty string means no error. RESET REPLICA or RESET MASTER will reset this value.

Last_SQL_Errno: Error code of the most recent error that caused the SQL thread to stop (also recorded in the replica's error log). 0 means no error. RESET REPLICA or RESET MASTER will reset this value.

Last_SQL_Error: Error message of the most recent error that caused the SQL thread to stop (also recorded in the replica's error log). An empty string means no error. RESET REPLICA or RESET MASTER will reset this value.

Replicate_Ignore_Server_Ids: List of server_ids that are currently being ignored for replication purposes, or an empty string for none, as specified in the IGNORE_SERVER_IDS option of the CHANGE MASTER TO statement.

Master_Server_Id: The primary's server_id value.

Master_SSL_Crl: The MASTER_SSL_CRL option of the CHANGE MASTER TO statement.

Master_SSL_Crlpath: The MASTER_SSL_CRLPATH option of the CHANGE MASTER TO statement.

Using_Gtid: Whether or not global transaction ID's are being used for replication (can be `No`, `Slave_Pos`, or `Current_Pos`).

Gtid_IO_Pos: Current global transaction ID value.

Replicate_Do_Domain_Ids: List of domain_ids that are currently being recognized for replication purposes, or an empty string for none, as specified in the `DO_DOMAIN_IDS` option of the CHANGE MASTER TO statement.

Replicate_Ignore_Domain_Ids: List of domain_ids that are currently being ignored for replication purposes, or an empty string for none, as specified in the `IGNORE_DOMAIN_IDS` option of the CHANGE MASTER TO statement.

Parallel_Mode: The in-order parallel replication mode as configured by the `slave_parallel_mode` system variable.

SQL_Delay: Value specified by MASTER_DELAY in CHANGE MASTER (or 0 if none).

SQL_Remaining_Delay: When the replica is delaying the execution of an event due to `MASTER_DELAY`, this is the number of seconds of delay remaining before the event will be applied. Otherwise, the value is NULL.

Slave_SQL_Running_State: The state of the SQL driver threads, same as in SHOW PROCESSLIST. When the replica is delaying the execution of an event due to MASTER_DELAY, this field displays: "Waiting until `MASTER_DELAY` seconds after primary executed event".

Slave_DDL_Groups: This status variable counts the occurrence of DDL statements. This is a slave-side counter for optimistic parallel replication.

Slave_Non_Transactional_Groups: This status variable counts the occurrence of non-transactional event groups. This is a slave-side counter for optimistic parallel replication.

Slave_Transactional_Groups: This status variable counts the occurrence of transactional event groups. This is a slave-side counter for optimistic parallel replication.

> **CURRENT**
>
> **Replicate_Rewrite_DB:** Databases specified for replicating and rewriting with the `replicate_rewrite_db` system variable/option.
>
> **< 10.11**
>
> **Replicate_Rewrite_DB** is not available.

Retried_transactions: The number of times that replicating transactions have been retried due to temporary errors (e.g. deadlocks in parallel replication). Returned with `SHOW ALL SLAVES STATUS` only.

Max_relay_log_size: Max relay log size for this connection. Returned with `SHOW ALL SLAVES STATUS` only.

Executed_log_entries: Number of binary log events that have been executed, irrespective of error outcome (i.e. if the event execution results in an error, this number will still increase). Returned with `SHOW ALL SLAVES STATUS` only.

Slave_received_heartbeats: Number of Heartbeat Log Events that the replica has received. Note this counter does not reset when the replica is restarted; only when a new CHANGE MASTER command has executed. Returned with `SHOW ALL SLAVES STATUS` only.

Slave_heartbeat_period: Configured (by CHANGE MASTER TO MASTER_HEARTBEAT_PERIOD) interval in seconds between replication heartbeats. Returned with `SHOW ALL SLAVES STATUS` only.

Gtid_Slave_Pos: The value of the global variable gtid_slave_pos, i.e. the GTID of the last event group replicated on a replica server, for each replication domain, as stored in the gtid_slave_pos system variable. Returned with `SHOW ALL SLAVES STATUS` only.

> **CURRENT**
>
> **Master_last_event_time:** Timestamp of the last event read from the primary by the IO thread. NULL until the replica has started and has read one query event from the primary that changes data.
>
> **Slave_last_event_time:** Timestamp, from the primary, of the last event committed on the replica. NULL until the replica has started and has read one query event from the primary that changes data.
>
> **Master_Slave_time_diff:** The difference of the above two timestamps. NULL until the replica has started and has read one query event from the primary that changes data.
>
> **< 11.6**
>
> **Master_last_event_time**, **Slave_last_event_time**, and **Master_Slave_time_diff** are not available.

> **CURRENT**
>
> **Connects_Tried:** The number of attempts done to connect to the primary. It starts from 0 with START REPLICA (but not STOP REPLICA), RESET REPLICA or `CHANGE MASTER TO MASTER_RETRY_COUNT`, and increments after each connection attempt until one succeeds or, after this reaches `Master_Retry_Count`, aborts the connection.
>
> **Master_Retry_Count:** The limit to `Connects_Tried` as configured by `CHANGE MASTER TO MASTER_RETRY_COUNT`.
>
> **< 12.0**
>
> **Connects_Tried:** and **Master_Retry_Count:** are not available. If the Performance Schema is enabled, `replication\_connection\_configuration` has `CONNECTION_RETRY_COUNT` available as an older alternative to

> `Master_Retry_Count`.

Examples

If you issue this statement using the mariadb client, you can use a `\G` statement terminator rather than a semicolon to obtain a more readable vertical layout.

```
SHOW SLAVE STATUS\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: db01.example.com
                  Master_User: replicant
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mariadb-bin.000021
          Read_Master_Log_Pos: 17315
               Relay_Log_File: relay-bin.000042
                Relay_Log_Pos: 17446
        Relay_Master_Log_File: mariadb-bin.000021
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 17146
              Relay_Log_Space: 17972
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: Yes
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 6
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
               Master_SSL_Crl:
           Master_SSL_Crlpath:
                   Using_Gtid: Slave_Pos
                  Gtid_IO_Pos: 0-1-2042
      Replicate_Do_Domain_Ids:
  Replicate_Ignore_Domain_Ids:
                Parallel_Mode: optimistic
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Updating
             Slave_DDL_Groups: 1
Slave_Non_Transactional_Groups: 0
    Slave_Transactional_Groups: 2041
          Replicate_Rewrite_DB:
```

```
SHOW ALL SLAVES STATUS\G
*************************** 1. row ***************************
               Connection_name:
               Slave_SQL_State: Updating
                Slave_IO_State: Waiting for master to send event
                   Master_Host: db01.example.com
                   Master_User: replicant
                   Master_Port: 3306
                 Connect_Retry: 60
               Master_Log_File: mariadb-bin.000021
           Read_Master_Log_Pos: 17315
                Relay_Log_File: relay-bin.000042
                 Relay_Log_Pos: 17446
         Relay_Master_Log_File: mariadb-bin.000021
              Slave_IO_Running: Yes
             Slave_SQL_Running: Yes
               Replicate_Do_DB:
           Replicate_Ignore_DB:
            Replicate_Do_Table:
        Replicate_Ignore_Table:
       Replicate_Wild_Do_Table:
   Replicate_Wild_Ignore_Table:
                    Last_Errno: 0
                    Last_Error:
                  Skip_Counter: 0
           Exec_Master_Log_Pos: 17146
               Relay_Log_Space: 17972
               Until_Condition: None
                Until_Log_File:
                 Until_Log_Pos: 0
            Master_SSL_Allowed: Yes
            Master_SSL_CA_File:
            Master_SSL_CA_Path:
               Master_SSL_Cert:
             Master_SSL_Cipher:
                Master_SSL_Key:
         Seconds_Behind_Master: 6
 Master_SSL_Verify_Server_Cert: No
                 Last_IO_Errno: 0
                 Last_IO_Error:
                Last_SQL_Errno: 0
                Last_SQL_Error:
   Replicate_Ignore_Server_Ids:
              Master_Server_Id: 1
                Master_SSL_Crl:
            Master_SSL_Crlpath:
                    Using_Gtid: Slave_Pos
                   Gtid_IO_Pos: 0-1-2042
       Replicate_Do_Domain_Ids:
   Replicate_Ignore_Domain_Ids:
                 Parallel_Mode: optimistic
                     SQL_Delay: 0
           SQL_Remaining_Delay: NULL
       Slave_SQL_Running_State: Updating
              Slave_DDL_Groups: 1
Slave_Non_Transactional_Groups: 0
    Slave_Transactional_Groups: 2041
          Replicate_Rewrite_DB:
          Retried_transactions: 0
            Max_relay_log_size: 1073741824
          Executed_log_entries: 6330
     Slave_received_heartbeats: 0
        Slave_heartbeat_period: 60.000
                Gtid_Slave_Pos: 0-1-2041
        Master_last_event_time: 2024-12-12 06:53:52
         Slave_last_event_time: 2024-12-12 06:53:50
        Master_Slave_time_diff: 2
```

You can also access some of the variables directly from status variables:

```
SET @@default_master_connection="test" ;
SHOW status like "%slave%"

Variable_name   Value
Com_show_slave_hosts    0
Com_show_slave_status   0
Com_start_all_slaves    0
Com_start_slave 0
Com_stop_all_slaves     0
Com_stop_slave  0
Rpl_semi_sync_slave_status      OFF
Slave_connections       0
Slave_heartbeat_period  1800.000
Slave_open_temp_tables  0
Slave_received_heartbeats       0
Slave_retried_transactions      0
Slave_running   OFF
Slaves_connected        0
Slaves_running  1
```

See Also

- MariaDB replication

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW STATUS

Syntax

```
SHOW [GLOBAL | SESSION] STATUS
    [LIKE 'pattern' | WHERE expr]
```

Description

`SHOW STATUS` provides server status information. This information also can be obtained using the mariadb-admin extended-status command, or by querying the Information Schema GLOBAL_STATUS and SESSION_STATUS tables. The `LIKE` clause, if present, indicates which variable names to match. The `WHERE` clause can be given to select rows using more general conditions.

With the `GLOBAL` modifier, `SHOW STATUS` displays the status values for all connections to MariaDB. With `SESSION`, it displays the status values for the current connection. If no modifier is present, the default is `SESSION`. `LOCAL` is a synonym for `SESSION`. If you see a lot of 0 values, the reason is probably that you have used `SHOW STATUS` with a new connection instead of `SHOW GLOBAL STATUS`.

Some status variables have only a global value. For these, you get the same value for both `GLOBAL` and `SESSION`.

See Server Status Variables for a full list, scope and description of the variables that can be viewed with `SHOW STATUS`.

The `LIKE` clause, if present on its own, indicates which variable name to match.

The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

Examples

```
SHOW GLOBAL STATUS;
+-----------------------------------------------------------+-----------------------------------------+
| Variable_name                                             | Value                                   |
+-----------------------------------------------------------+-----------------------------------------+
| Aborted_clients                                           | 0                                       |
| Aborted_connects                                          | 0                                       |
| Access_denied_errors                                      | 0                                       |
| Acl_column_grants                                         | 0                                       |
| Acl_database_grants                                       | 2                                       |
| Acl_function_grants                                       | 0                                       |
| Acl_procedure_grants                                      | 0                                       |
| Acl_proxy_users                                           | 2                                       |
| Acl_role_grants                                           | 0                                       |
| Acl_roles                                                 | 0                                       |
| Acl_table_grants                                          | 0                                       |
| Acl_users                                                 | 6                                       |
| Aria_pagecache_blocks_not_flushed                         | 0                                       |
| Aria_pagecache_blocks_unused                              | 15706                                   |
...
| wsrep_local_index                                         | 18446744073709551615                    |
| wsrep_provider_name                                       |                                         |
| wsrep_provider_vendor                                     |                                         |
| wsrep_provider_version                                    |                                         |
| wsrep_ready                                               | OFF                                     |
| wsrep_thread_count                                        | 0                                       |
+-----------------------------------------------------------+-----------------------------------------+
516 rows in set (0.00 sec)
```

Example of filtered output:

```
SHOW STATUS LIKE 'Key%';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| Key_blocks_not_flushed | 0      |
| Key_blocks_unused      | 107163 |
| Key_blocks_used        | 0      |
| Key_blocks_warm        | 0      |
| Key_read_requests      | 0      |
| Key_reads              | 0      |
| Key_write_requests     | 0      |
| Key_writes             | 0      |
+------------------------+--------+
8 rows in set (0.00 sec)
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW TABLE_STATISTICS

Syntax

```
SHOW TABLE_STATISTICS
```

Description

> **CURRENT**
>
> The `SHOW TABLE_STATISTICS` statement is part of the User Statistics feature. It was effectively replaced by the generic `SHOW TABLE STATISTICS` statement. The information_schema.TABLE_STATISTICS table shows statistics on table usage.
>
> **< 10.1.1**

> The `SHOW TABLE_STATISTICS` statement is part of the User Statistics feature. The information_schema.TABLE_STATISTICS table shows statistics on table usage.

The userstat system variable must be set to 1 to activate this feature. See the User Statistics and information_schema.TABLE_STATISTICS articles for more information.

Example

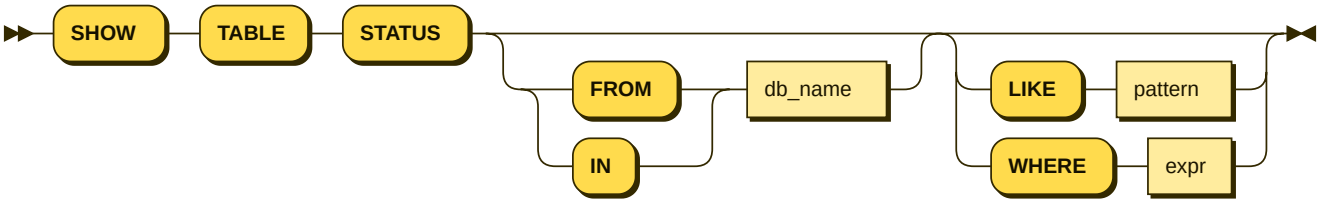
```
SHOW TABLE_STATISTICS\G
*************************** 1. row ***************************
           Table_schema: mysql
             Table_name: proxies_priv
              Rows_read: 2
           Rows_changed: 0
Rows_changed_x_#indexes: 0
*************************** 2. row ***************************
           Table_schema: test
             Table_name: employees_example
              Rows_read: 7
           Rows_changed: 0
Rows_changed_x_#indexes: 0
*************************** 3. row ***************************
           Table_schema: mysql
             Table_name: user
              Rows_read: 16
           Rows_changed: 0
Rows_changed_x_#indexes: 0
*************************** 4. row ***************************
           Table_schema: mysql
             Table_name: db
              Rows_read: 2
           Rows_changed: 0
Rows_changed_x_#indexes: 0
```

This page is licensed: CC BY-SA / Gnu FDL

SHOW TABLE STATUS

Syntax

```
SHOW TABLE STATUS [{FROM | IN} db_name]
    [LIKE 'pattern' | WHERE expr]
```



Railroad diagram of SHOW TABLE STATUS — equivalent to the BNF above

Description

CURRENT

`SHOW TABLE STATUS` works like SHOW TABLES, but provides more extensive information about each table.

< 11.2.0

> `SHOW TABLE STATUS` works like SHOW TABLES, but provides more extensive information about each table. Only non-TEMPORARY tables are shown.

The `LIKE` clause, if present on its own, indicates which table names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

The following information is returned:

Column	Description
Name	Table name.
Engine	Table storage engine.
Version	Version number from the table's .frm file.
Row_format	Row format (see InnoDB, Aria and MyISAM row formats).
Rows	Number of rows in the table. Some engines, such as InnoDB may store an estimate.
Avg_row_length	Average row length in the table.
Data_length	For InnoDB, the index size, in pages, multiplied by the page size. For Aria and MyISAM, length of the data file, in bytes. For MEMORY, the approximate allocated memory.
Max_data_length	Maximum length of the data file, ie the total number of bytes that could be stored in the table. Not used in InnoDB.
Index_length	Length of the index file.
Data_free	Bytes allocated but unused. For InnoDB tables in a shared tablespace, the free space of the shared tablespace with small safety margin. An estimate in the case of partitioned tables - see the PARTITIONS table.
Auto_increment	Next AUTO_INCREMENT value.
Create_time	Time the table was created. Some engines just return the ctime information from the file system layer here, in that case the value is not necessarily the table creation time but rather the time the file system metadata for it had last changed.
Update_time	Time the table was last updated. On Windows, the timestamp is not updated on update, so MyISAM values will be inaccurate. In InnoDB, if shared tablespaces are used, will be NULL, while buffering can also delay the update, so the value will differ from the actual time of the last UPDATE, INSERT or DELETE.
Check_time	Time the table was last checked. Not kept by all storage engines, in which case will be NULL.
Collation	Character set and collation.
Checksum	Live checksum value, if any.
Create_options	Extra CREATE TABLE options.
Comment	Table comment provided when MariaDB created the table.
Max_index_length	Maximum index length (supported by MyISAM and Aria tables).
Temporary	Until MariaDB 11.2.0, placeholder to signal that a table is a temporary table and always "N", except "Y" for generated information_schema tables and NULL for views. From MariaDB 11.2.0, will also be set to "Y" for local temporary tables.

Similar information can be found in the information_schema.TABLES table as well as by using mariadb-show:

```
mariadb-show --status db_name
```

Views

For views, all columns in `SHOW TABLE STATUS` are `NULL` except 'Name' and 'Comment'

Example

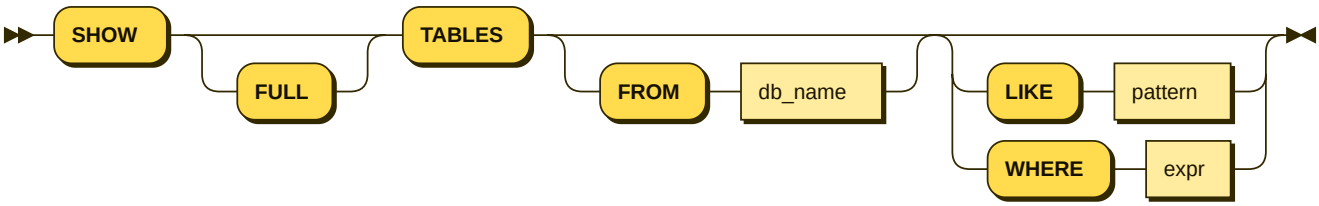
```
SHOW TABLE STATUS\G
*************************** 1. row ***************************
           Name: bus_routes
         Engine: InnoDB
        Version: 10
     Row_format: Dynamic
           Rows: 5
 Avg_row_length: 3276
    Data_length: 16384
Max_data_length: 0
   Index_length: 0
      Data_free: 0
 Auto_increment: NULL
    Create_time: 2017-05-24 11:17:46
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW TABLES

Syntax

```
SHOW [FULL] TABLES [FROM db_name]
    [LIKE 'pattern' | WHERE expr]
```



Railroad diagram of SHOW TABLES — equivalent to the BNF above

Description

CURRENT

`SHOW TABLES` lists the tables, sequences and views in a given database.

< 11.2.0

`SHOW TABLES` lists the tables (only non-`TEMPORARY` tables are shown), sequences and views in a given database.

The `LIKE` clause, if present on its own, indicates which table names to match. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW. For example, when

searching for tables in the `test` database, the column name for use in the `WHERE` and `LIKE` clauses will be `Tables_in_test`

The `FULL` modifier is supported such that `SHOW FULL TABLES` displays a second output column. Values for the second column, `Table_type`, are `BASE TABLE` for a table, `VIEW` for a view and `SEQUENCE` for a sequence.

You can also get this information using:

```
mariadb-show db_name
```

See mariadb-show for more details.

If you have no privileges for a base table or view, it does not show up in the output from `SHOW TABLES` or `mariadb-show db_name`.

The information_schema.TABLES table, as well as the SHOW TABLE STATUS statement, provide extended information about tables.

Examples

```
SHOW TABLES;
+----------------------+
| Tables_in_test       |
+----------------------+
| animal_count         |
| animals              |
| are_the_mooses_loose |
| aria_test2           |
| t1                   |
| view1                |
+----------------------+
```

Showing the tables beginning with *a* only.

```
SHOW TABLES WHERE Tables_in_test LIKE 'a%';
+----------------------+
| Tables_in_test       |
+----------------------+
| animal_count         |
| animals              |
| are_the_mooses_loose |
| aria_test2           |
+----------------------+
```

Showing tables and table types:

```
SHOW FULL TABLES;
+----------------+------------+
| Tables_in_test | Table_type |
+----------------+------------+
| s1             | SEQUENCE   |
| student        | BASE TABLE |
| v1             | VIEW       |
+----------------+------------+
```

Showing temporary tables: <= MariaDB 11.1

```
CREATE TABLE t (t INT(11));
CREATE TEMPORARY TABLE t (t INT(11));
CREATE TEMPORARY TABLE te (t INT(11));

SHOW TABLES;
+----------------+
| Tables_in_test |
+----------------+
| t              |
+----------------+
```

From MariaDB 11.2.0:

```
CREATE TABLE t (t INT(11));
CREATE TEMPORARY TABLE t (t INT(11));
CREATE TEMPORARY TABLE te (t INT(11));

SHOW TABLES;
+----------------+
| Tables_in_test |
+----------------+
| te             |
| t              |
| t              |
+----------------+
```

See Also

- SHOW TABLE STATUS
- The information_schema.TABLES table

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW TRIGGERS

Syntax

```
SHOW TRIGGERS [FROM db_name]
    [LIKE 'pattern' | WHERE expr]
```

Description

CURRENT

`SHOW TRIGGERS` lists the triggers currently defined for tables in a database (the default database unless a `FROM` clause is given). This statement requires the TRIGGER privilege.

< 5.1.22

`SHOW TRIGGERS` lists the triggers currently defined for tables in a database (the default database unless a `FROM` clause is given). This statement requires the `SUPER` privilege.

The `LIKE` clause, if present on its own, indicates which table names to match and causes the statement to display triggers for those tables. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

Similar information is stored in the information_schema.TRIGGERS table.

If there are multiple triggers for the same action, then the triggers are shown in action order.

Examples

For the trigger defined at Trigger Overview:

```
SHOW triggers Like 'animals' \G
*************************** 1. row ***************************
             Trigger: the_mooses_are_loose
               Event: INSERT
               Table: animals
           Statement: BEGIN
 IF NEW.name = 'Moose' THEN
  UPDATE animal_count SET animal_count.animals = animal_count.animals+100;
 ELSE
  UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
 END IF;
END
              Timing: AFTER
             Created: 2016-09-29 13:53:34.35
            sql_mode:
             Definer: root@localhost
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

Listing all triggers associated with a certain table:

```
SHOW TRIGGERS FROM test WHERE `Table` = 'user' \G
*************************** 1. row ***************************
             Trigger: user_ai
               Event: INSERT
               Table: user
           Statement: BEGIN END
              Timing: AFTER
             Created:  2016-09-29 13:53:34.35
            sql_mode:
             Definer: root@%
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

```
SHOW triggers WHERE Event Like 'Insert' \G
*************************** 1. row ***************************
             Trigger: the_mooses_are_loose
               Event: INSERT
               Table: animals
           Statement: BEGIN
 IF NEW.name = 'Moose' THEN
  UPDATE animal_count SET animal_count.animals = animal_count.animals+100;
 ELSE
  UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
 END IF;
END
              Timing: AFTER
             Created: 2016-09-29 13:53:34.35
            sql_mode:
             Definer: root@localhost
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

CURRENT

`character_set_client` is the session value of the character_set_client system variable when the trigger was created.

`collation_connection` is the session value of the collation_connection system variable when the trigger was created.

`Database Collation` is the collation of the database with which the trigger is associated.

< 5.1.21

`character_set_client`, `collation_connection`, and `Database Collation` are not available.

Note:

Old triggers created before MySQL 5.7 and MariaDB 10.2.3 have NULL in the `Created` column.

See also

- Trigger Overview
- CREATE TRIGGER
- DROP TRIGGER
- information_schema.TRIGGERS table
- SHOW CREATE TRIGGER
- Trigger Limitations

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW USER_STATISTICS

Syntax

```
SHOW USER_STATISTICS
```

Description

CURRENT

The `SHOW USER_STATISTICS` statement is part of the User Statistics feature. It was effectively replaced by the generic SHOW information_schema_table statement. The information_schema.USER_STATISTICS table holds statistics about user activity. You can use this table to find out such things as which user is causing the most load and which users are being abusive. You can also use this table to measure how close to capacity the server may be.

< 10.1.1

The `SHOW USER_STATISTICS` statement is part of the User Statistics feature. The information_schema.USER_STATISTICS table holds statistics about user activity. You can use this table to find out such things as which user is causing the most load and which users are being abusive. You can also use this table to measure how close to capacity the server may be.

The userstat system variable must be set to 1 to activate this feature. See the User Statistics and information_schema.USER_STATISTICS table for more information.

Example

```
SHOW USER_STATISTICS\G
*************************** 1. row ***************************
                  User: root
     Total_connections: 1
Concurrent_connections: 0
        Connected_time: 3297
             Busy_time: 0.14113400000000006
              Cpu_time: 0.017637000000000003
        Bytes_received: 969
            Bytes_sent: 22355
  Binlog_bytes_written: 0
             Rows_read: 10
             Rows_sent: 67
          Rows_deleted: 0
         Rows_inserted: 0
          Rows_updated: 0
       Select_commands: 7
       Update_commands: 0
        Other_commands: 0
   Commit_transactions: 1
 Rollback_transactions: 0
    Denied_connections: 0
      Lost_connections: 0
         Access_denied: 0
         Empty_queries: 7
```

This page is licensed: CC BY-SA / Gnu FDL

SHOW USER_VARIABLES

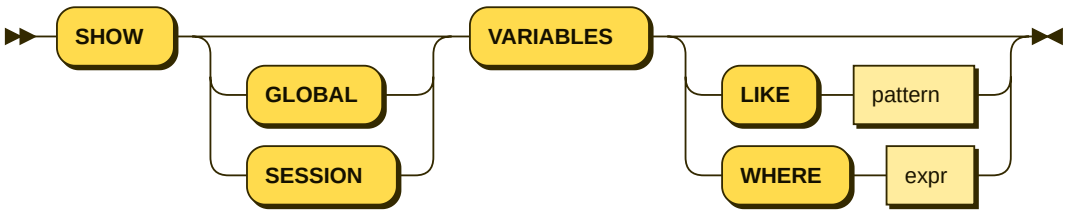
See the User Variables Plugin for details.

This page is licensed: CC BY-SA / Gnu FDL

SHOW VARIABLES

Syntax

```
SHOW [GLOBAL | SESSION] VARIABLES
    [LIKE 'pattern' | WHERE expr]
```



Railroad diagram of SHOW VARIABLES — equivalent to the BNF above

Description

`SHOW VARIABLES` shows the values of MariaDB system variables. This does not include user-defined variables - see here for details on viewing those.

System variable information can also be obtained using the mariadb-admin variables command. The `LIKE` clause, if present, indicates which variable names to match. The `WHERE` clause can be given to select rows using more general conditions.

With the `GLOBAL` modifier, `SHOW VARIABLES` displays the values that are used for new connections to MariaDB. With `SESSION`, it displays the values that are in effect for the current connection. If no modifier is present, the default

is `SESSION`. `LOCAL` is a synonym for `SESSION`. With a `LIKE` clause, the statement displays only rows for those variables with names that match the pattern. To obtain the row for a specific variable, use a `LIKE` clause as shown:

```
SHOW VARIABLES LIKE 'maria_group_commit';
SHOW SESSION VARIABLES LIKE 'maria_group_commit';
```

To get a list of variables whose name match a pattern, use the "`%`" wildcard character in a `LIKE` clause:

```
SHOW VARIABLES LIKE '%maria%';
SHOW GLOBAL VARIABLES LIKE '%maria%';
```

Wildcard characters can be used in any position within the pattern to be matched. Strictly speaking, because "`_`" is a wildcard that matches any single character, you should escape it as "`\_`" to match it literally. In practice, this is rarely necessary.

The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.

See SET for information on setting server system variables.

See Server System Variables for a list of all the variables that can be set.

You can also see the server variables by querying the Information Schema GLOBAL_VARIABLES and SESSION_VARIABLES tables.

Examples

```
SHOW VARIABLES LIKE 'aria%';
+----------------------------------------+---------------------+
| Variable_name                          | Value               |
+----------------------------------------+---------------------+
| aria_block_size                        | 8192                |
| aria_checkpoint_interval               | 30                  |
| aria_checkpoint_log_activity           | 1048576             |
| aria_force_start_after_recovery_failures | 0                   |
| aria_group_commit                      | none                |
| aria_group_commit_interval             | 0                   |
| aria_log_file_size                     | 1073741824          |
| aria_log_purge_type                    | immediate           |
| aria_max_sort_file_size                | 9223372036853727232 |
| aria_page_checksum                     | ON                  |
| aria_pagecache_age_threshold           | 300                 |
| aria_pagecache_buffer_size             | 134217728           |
| aria_pagecache_division_limit          | 100                 |
| aria_recover                           | NORMAL              |
| aria_repair_threads                    | 1                   |
| aria_sort_buffer_size                  | 134217728           |
| aria_stats_method                      | nulls_unequal       |
| aria_sync_log_dir                      | NEWFILE             |
| aria_used_for_temp_tables              | ON                  |
+----------------------------------------+---------------------+
```

```
SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
  INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
  VARIABLE_NAME LIKE 'max_error_count' OR
  VARIABLE_NAME LIKE 'innodb_sync_spin_loops';
+---------------------------+---------------+--------------+
| VARIABLE_NAME             | SESSION_VALUE | GLOBAL_VALUE |
+---------------------------+---------------+--------------+
| MAX_ERROR_COUNT           | 64            | 64           |
| INNODB_SYNC_SPIN_LOOPS    | NULL          | 30           |
+---------------------------+---------------+--------------+

SET GLOBAL max_error_count=128;

SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
  INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
  VARIABLE_NAME LIKE 'max_error_count' OR
  VARIABLE_NAME LIKE 'innodb_sync_spin_loops';
+---------------------------+---------------+--------------+
| VARIABLE_NAME             | SESSION_VALUE | GLOBAL_VALUE |
+---------------------------+---------------+--------------+
| MAX_ERROR_COUNT           | 64            | 128          |
| INNODB_SYNC_SPIN_LOOPS    | NULL          | 30           |
+---------------------------+---------------+--------------+

SET GLOBAL max_error_count=128;

SHOW VARIABLES LIKE 'max_error_count';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_error_count | 64    |
+-----------------+-------+

SHOW GLOBAL VARIABLES LIKE 'max_error_count';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_error_count | 128   |
+-----------------+-------+
```

Because the following variable only has a global scope, the global value is returned even when specifying SESSION (in this case by default):

```
SHOW VARIABLES LIKE 'innodb_sync_spin_loops';
+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| innodb_sync_spin_loops | 30    |
+------------------------+-------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW WARNINGS

Syntax

```
SHOW WARNINGS [LIMIT [offset,] row_count]
SHOW ERRORS [LIMIT row_count OFFSET offset]
SHOW COUNT(*) WARNINGS
```

Description

`SHOW WARNINGS` shows the error, warning, and note messages that resulted from the last statement that generated messages in the current session. It shows nothing if the last statement used a table and generated no messages.

(That is, a statement that uses a table but generates no messages clears the message list.) Statements that do not use tables and do not generate messages have no effect on the message list.

A note is different to a warning in that it only appears if the sql_notes variable is set to 1 (the default), and is not converted to an error if strict mode is enabled.

A related statement, SHOW ERRORS, shows only the errors.

The `SHOW COUNT(*) WARNINGS` statement displays the total number of errors, warnings, and notes. You can also retrieve this number from the warning_count variable:

```
SHOW COUNT(*) WARNINGS;
SELECT @@warning_count;
```

The value of warning_count might be greater than the number of messages displayed by `SHOW WARNINGS` if the max_error_count system variable is set so low that not all messages are stored.

The `LIMIT` clause has the same syntax as for the SELECT statement.

`SHOW WARNINGS` can be used after EXPLAIN EXTENDED to see how a query is internally rewritten by MariaDB.

If the sql_notes server variable is set to 1, Notes are included in the output of `SHOW WARNINGS`; if it is set to 0, this statement will not show (or count) Notes.

The results of `SHOW WARNINGS` and `SHOW COUNT(*) WARNINGS` are directly sent to the client. If you need to access those information in a stored program, you can use the GET DIAGNOSTICS statement instead.

The mariadb client also has a number of options related to warnings. The `\W` command will show warnings after every statement, while `\w` will disable this. Starting the client with the `--show-warnings` option will show warnings after every statement.

MariaDB implements a stored routine error stack trace. `SHOW WARNINGS` can also be used to show more information. See the example below.

Examples

```
SELECT 1/0;
+------+
| 1/0  |
+------+
| NULL |
+------+

SHOW COUNT(*) WARNINGS;
+--------------------------+
| @@session.warning_count |
+--------------------------+
|                        1 |
+--------------------------+

SHOW WARNINGS;
+---------+------+---------------+
| Level   | Code | Message       |
+---------+------+---------------+
| Warning | 1365 | Division by 0 |
+---------+------+---------------+
```

Stack Trace

Displaying a stack trace:

```
DELIMITER $$
CREATE OR REPLACE PROCEDURE p1()
  BEGIN
    DECLARE c CURSOR FOR SELECT * FROM not_existing;
    OPEN c;
    CLOSE c;
  END;
$$
CREATE OR REPLACE PROCEDURE p2()
  BEGIN
    CALL p1;
  END;
$$
DELIMITER ;
CALL p2;
ERROR 1146 (42S02): Table 'test.not_existing' doesn't exist

SHOW WARNINGS;
+-------+------+----------------------------------------+
| Level | Code | Message                                |
+-------+------+----------------------------------------+
| Error | 1146 | Table 'test.not_existing' doesn't exist |
| Note  | 4091 | At line 6 in test.p1                   |
| Note  | 4091 | At line 4 in test.p2                   |
+-------+------+----------------------------------------+
```

`SHOW WARNINGS` displays a stack trace, showing where the error actually happened:

- Line 4 in test.p1 is the OPEN command which actually raised the error.
- Line 3 in test.p2 is the CALL statement, calling p1 from p2.

See Also

- SHOW ERRORS

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHOW WSREP_MEMBERSHIP

`SHOW WSREP_MEMBERSHIP` is part of the WSREP_INFO plugin.

Syntax

```
SHOW WSREP_MEMBERSHIP
```

Description

The `SHOW WSREP_MEMBERSHIP` statement returns Galera node cluster membership information. It returns the same information as found in the information_schema.WSREP_MEMBERSHIP table. Only users with the SUPER privilege can access this information.

Examples

```
SHOW WSREP_MEMBERSHIP;
+-------+--------------------------------------+----------+-----------------+
| Index | Uuid                                 | Name     | Address         |
+-------+--------------------------------------+----------+-----------------+
|     0 | 19058073-8940-11e4-8570-16af7bf8fced | my_node1 | 10.0.2.15:16001 |
|     1 | 19f2b0e0-8942-11e4-9cb8-b39e8ee0b5dd | my_node3 | 10.0.2.15:16003 |
|     2 | d85e62db-8941-11e4-b1ef-4bc9980e476d | my_node2 | 10.0.2.15:16002 |
+-------+--------------------------------------+----------+-----------------+
```

This page is licensed: CC BY-SA / Gnu FDL

SHOW WSREP_STATUS

`SHOW WSREP_STATUS` is part of the WSREP_INFO plugin.

Syntax

```
SHOW WSREP_STATUS
```

Description

The `SHOW WSREP_STATUS` statement returns Galera node and cluster status information. It returns the same information as found in the information_schema.WSREP_STATUS table. Only users with the SUPER privilege can access this information.

Examples

```
SHOW WSREP_STATUS;
+------------+-------------+----------------+--------------+
| Node_Index | Node_Status | Cluster_Status | Cluster_Size |
+------------+-------------+----------------+--------------+
|          0 | Synced      | Primary        |            3 |
+------------+-------------+----------------+--------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Data Definition (DDL)

See Atomic DDL.

Learn about crash-safe DDL operations in MariaDB. This feature ensures data definition statements are either fully committed or completely rolled back, preventing metadata inconsistency.

See CONSTRAINT.

Complete constraints reference: PRIMARY KEY, FOREIGN KEY, UNIQUE, and CHECK syntax in CREATE/ALTER TABLE, ON DELETE/UPDATE actions, TABLE_CONSTRAINTS table.

See RENAME TABLE.

Change the name of one or more tables atomically. This command moves tables within or between databases while preserving their data and structure.

See Renaming Databases.

Learn the supported methods for renaming a database. Since RENAME DATABASE is not available, this guide outlines safe workarounds like dumping and reloading or moving tables.

See ALTER.

Access the reference for ALTER statements. This section lists commands to modify existing database objects, including tables, databases, users, and servers.

See CREATE.

Explore the CREATE statements used to define new database objects. This guide covers syntax for creating databases, tables, indexes, views, and stored routines.

See DROP.

Find statements to remove database objects. This section details the syntax for deleting databases, tables, users, and other entities when they are no longer needed.

This page is licensed: CC BY-SA / Gnu FDL

Atomic DDL

> **CURRENT**
>
> We improved readability for DDL (Data Definition Language) operations to make most of them atomic, and the rest crash-safe, even if the server crashes in the middle of an operation.
>
> **BACKGROUND**
>
> These improvements were made in MariaDB 10.6.1.

The design of Atomic/Crash-safe DDL (MDEV-17567) allows it to work with all storage engines.

Definitions

- Atomic means that either the operation succeeds (and is logged to the binary log or is completely reversed.
- Crash-safe means that in case of a crash, after the server has restarted, all tables are consistent, there are no temporary files or tables on disk and the binary log matches the status of the server.
- DDL Data definition language.
- DML Data manipulation language.
- 'DDL recovery log' or 'DDL log' for short, is the new log file, `ddl_recovery.log` by default, that stores all DDL operations in progress. This is used to recover the state of the server in case of sudden crash.

Background

Before 10.6, in case of a crash, there was a small possibility that one of the following things could happen:

- There could be temporary tables starting with `#sql-alter` or `#sql-shadow` or temporary files ending with '' left.
- The table in the storage engine and the table's .frm file could be out of sync.
- During a multi-table rename, only some of the tables were renamed.

Which DDL Operations are Now Atomic

- CREATE TABLE, except when used with CREATE OR REPLACE, which is only crash safe.
- RENAME TABLE and RENAME TABLES.
- CREATE VIEW
- CREATE SEQUENCE
- CREATE TRIGGER
- DROP TRIGGER
- DROP TABLE and DROP VIEW. Dropping multiple tables is only crash safe.
- ALTER TABLE
- ALTER SEQUENCE is not listed above as it is internally implemented as a DML.

Which DDL Operations are Now Crash Safe

DROP TABLE of Multiple Tables.

DROP TABLE over multiple tables is treated as if every DROP is a separate, atomic operation. This means that after a crash, all fully, or partly, dropped tables will be dropped and logged to the binary log. The undropped tables will be

left untouched.

CREATE OR REPLACE TABLE

CREATE OR REPLACE TABLE foo is implemented as:

```
DROP TABLE IF EXISTS foo;
CREATE TABLE foo ...
```

This means that if there is a crash during `CREATE TABLE` then the original table 'foo' will be dropped even if the new table was not created.
If the table was not re-created, the binary log will contain the `DROP TABLE`.

DROP DATABASE

DROP DATABASE is implemented as:

```
  DROP TABLE table
LOOP OVER ALL tables
```

Each DROP TABLE is atomic, but in case of a crash, things will work the same way as DROP TABLE with multiple tables.

Atomic with Different Storage Engines

Atomic/Crash-safe DDL works with all storage engines that either have atomic DDLs internally or are able to re-execute `DROP` or `RENAME` in case of failure.

This should be true for most storage engines. The ones that still need some work are:

- The S3 storage engine.
- The partitioning engine. Partitioning should be atomic for most cases, but there are still some known issues that need to be tested and fixed.

The DDL Log Recovery File

The new startup option –log-ddl-recovery=path (`ddl_recovery.log` by default) can be used to specify the place for the DDL log file. This is mainly useful in the case when one has a filesystem on persistent memory, as there is a lot of sync on this file during DDL operations.

This file contains all DDL operations that are in progress.

At MariaDB server startup, the DDL log file is copied to a file with the same base name but with a `backup.log` suffix. This is mainly done to be able to find out what went wrong if recovery fails.

If the server crashes during recovery (unlikely but possible), the recovery will continue where it was before. The recovery will retry each entry up to 3 times before giving up and proceeding with the next entry.

Conclusions

- We believe that a clean separation of layers leads to an easier-to-maintain solution. The Atomic DDL implementation in MariaDB 10.6 introduced minimal changes to the storage engine API, mainly for native ALTER TABLE.
- In our InnoDB implementation, no file format changes were needed on top of the RENAME undo log that was introduced in MariaDB 10.2.19 for a backup-safe TRUNCATE re-implementation. Correct use of sound design principles (write-ahead logging and transactions; also file creation now follows the ARIES protocol) is sufficient. We removed the hacks (at most one CREATE or DROP per transaction) and correctly implemented `rollback` and `purge` triggers for the InnoDB SYS_INDEXES table.

- Numerous DDL recovery bugs in InnoDB were found and fixed quickly thanks to rr-project.org. We are still working on one: data files must not be deleted before the DDL transaction is committed.

Thanks to Atomic/Crash-safe DDL, the MariaDB server is now much more stable and reliable in unstable environments. There is still ongoing work to fix the few remaining issues mentioned above to make all DDL operations Atomic.

See Also

- MDEV-17567 Atomic DDL. This MDEV entry links to all other entries related to Atomic operations that contains a lot of information how things are implemented.

This page is licensed: CC BY-SA / Gnu FDL

CONSTRAINT

MariaDB supports constraints at table level, using CREATE TABLE or ALTER TABLE statements.

Syntax

```
[CONSTRAINT [symbol]] constraint_expression

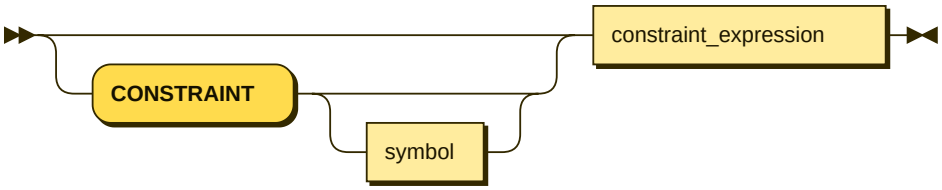
constraint_expression:
  | PRIMARY KEY [index_type] (index_col_name, ...) [index_option] ...
  | FOREIGN KEY [index_name] (index_col_name, ...)
       REFERENCES tbl_name (index_col_name, ...)
       [ON DELETE reference_option]
       [ON UPDATE reference_option]
  | UNIQUE [INDEX|KEY] [index_name]
       [index_type] (index_col_name, ...) [index_option] ...
  | CHECK (check_constraints)

index_type:
  USING {BTREE | HASH | RTREE}

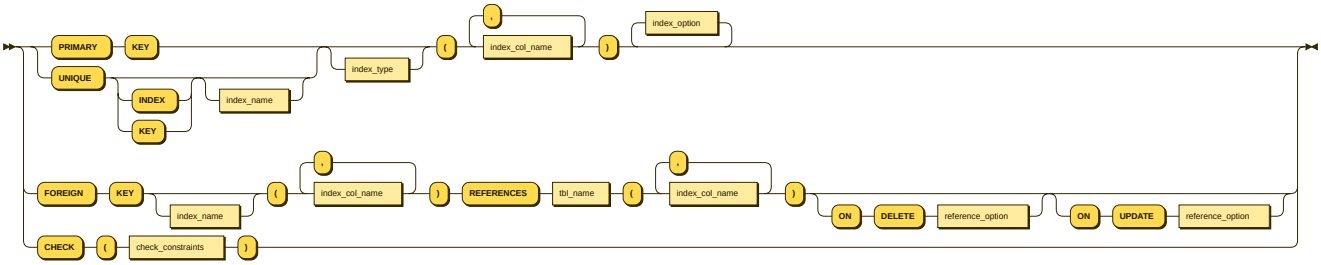
index_col_name:
  col_name [(length)] [ASC | DESC]

index_option:
  | KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
  | CLUSTERING={YES|NO}

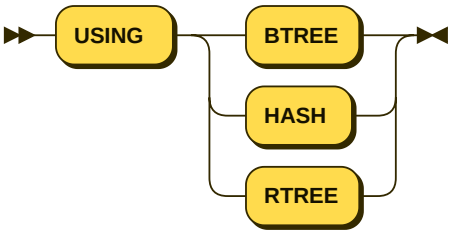
reference_option:
  RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```



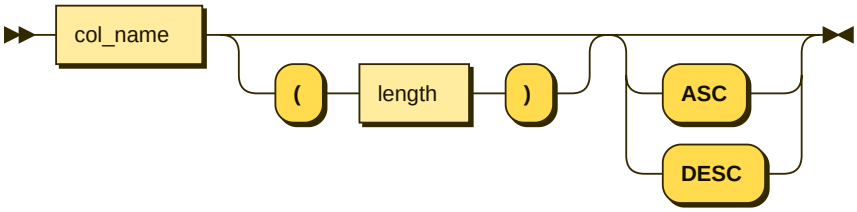
Railroad diagram of CONSTRAINT — equivalent to the BNF above



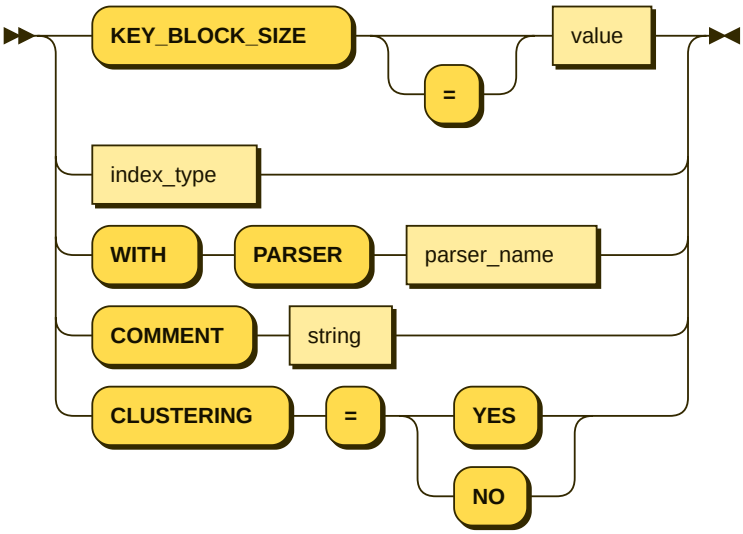
Railroad diagram of constraint_expression



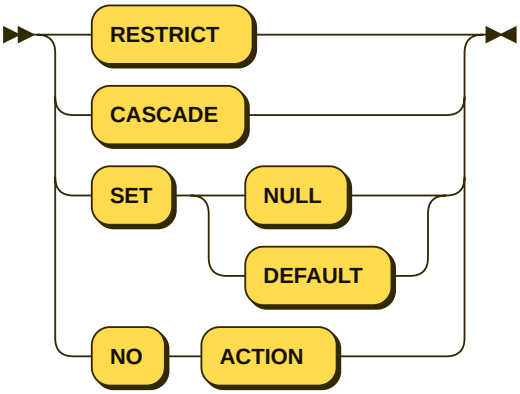
Railroad diagram of index_type



Railroad diagram of index_col_name



Railroad diagram of index_option



Railroad diagram of reference_option

Description

Constraints provide restrictions on the data you can add to a table. This allows to enforce data integrity in MariaDB, rather than through application logic. When a statement violates a constraint, MariaDB throws an error.

There are four types of table constraints:

- PRIMARY KEY – Sets the column for referencing rows. Values must be `UNIQUE` and `NOT NULL`.
  - This constraint is documented here.
- FOREIGN KEY – Sets the column to reference the primary key on another table.
  - This constraint is documented here and, in more detail, here and here.
- UNIQUE – Requires values in column or columns only occur once in the table.
  - This constraint is documented here.
- CHECK – Checks whether the data meets the given condition.
  - This constraint is documented here and, in more detail, here.

The Information Schema TABLE_CONSTRAINTS Table contains information about tables that have constraints:

```
SELECT CONSTRAINT_SCHEMA, CONSTRAINT_NAME, TABLE_NAME, CONSTRAINT_TYPE  FROM
information_schema.TABLE_CONSTRAINTS LIMIT 5;
+-------------------+----------------------+---------------+-----------------+
| CONSTRAINT_SCHEMA | CONSTRAINT_NAME      | TABLE_NAME    | CONSTRAINT_TYPE |
+-------------------+----------------------+---------------+-----------------+
| nation            | PRIMARY              | countries     | PRIMARY KEY     |
| nation            | country_code2        | countries     | UNIQUE          |
| nation            | country_code3        | countries     | UNIQUE          |
| nation            | countries_ibfk_1     | countries     | FOREIGN KEY     |
| nation            | PRIMARY              | regions       | PRIMARY KEY     |
+-------------------+----------------------+---------------+-----------------+
```

FOREIGN KEY Constraints

InnoDB supports foreign key constraints. The syntax for a foreign key constraint definition in InnoDB looks like this:

```
[CONSTRAINT [symbol]] FOREIGN KEY
    [index_name] (index_col_name, ...)
    REFERENCES tbl_name (index_col_name,...)
    [ON DELETE reference_option]
    [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION
```

> **Note:**
>
> The symbol clause is optional. If you omit it, MariaDB automatically sets one.

The Information Schema REFERENTIAL_CONSTRAINTS table has more information about foreign keys:

```
SELECT * FROM information_schema.REFERENTIAL_CONSTRAINTS LIMIT 2 \G
*************************** 1. row ***************************
       CONSTRAINT_CATALOG: def
        CONSTRAINT_SCHEMA: nation
          CONSTRAINT_NAME: countries_ibfk_1
UNIQUE_CONSTRAINT_CATALOG: def
 UNIQUE_CONSTRAINT_SCHEMA: nation
   UNIQUE_CONSTRAINT_NAME: PRIMARY
            MATCH_OPTION: NONE
             UPDATE_RULE: RESTRICT
             DELETE_RULE: RESTRICT
              TABLE_NAME: countries
    REFERENCED_TABLE_NAME: regions
*************************** 2. row ***************************
       CONSTRAINT_CATALOG: def
        CONSTRAINT_SCHEMA: nation
          CONSTRAINT_NAME: regions_ibfk_1
UNIQUE_CONSTRAINT_CATALOG: def
 UNIQUE_CONSTRAINT_SCHEMA: nation
   UNIQUE_CONSTRAINT_NAME: PRIMARY
            MATCH_OPTION: NONE
             UPDATE_RULE: RESTRICT
             DELETE_RULE: RESTRICT
              TABLE_NAME: regions
    REFERENCED_TABLE_NAME: continents
```

Note:

From MariaDB 12.1, multiple tables in the same database are allowed to use the same constraint name.

Consider this example, which creates two tables, then adds constraints to them:

```
CREATE TABLE t1(a int, b int, key(a),key(b))engine=innodb;
CREATE TABLE t2(a int, b int, key(a),key(b))engine=innodb;
ALTER TABLE t2 ADD CONSTRAINT c FOREIGN KEY (b) REFERENCES t1(a);
ALTER TABLE t1 ADD CONSTRAINT c FOREIGN KEY (b) REFERENCES t2(a);
```

Previously, the last statement failed because of duplicate constraint names:

```
mysqltest: At line 5: query 'alter table t1 add constraint c foreign key (b) references t2(a)' failed:
ER_CANT_CREATE_TABLE (1005): Can't create table `test`.`t1` (errno: 121 "Duplicate key on write or
update")
```

Now, the statement works, because internally, names are assigned that aren't duplicates. (User-visible names can be duplicates, but internally they're prefixed with the schema and table names.)

CHECK Constraints

Constraints are enforced. You can define constraints in 2 different ways:

- `CHECK(expression)` given as part of a column definition.
- `CONSTRAINT [constraint_name] CHECK (expression)`

Before a row is inserted or updated, all constraints are evaluated in the order they are defined. If any constraint expression returns false, then the row is not inserted or updated. You can use most deterministic functions in a constraint, including UDFs.

```
CREATE TABLE t1 (a INT CHECK (a>2), b INT CHECK (b>2), CONSTRAINT a_greater CHECK (a>b));
```

If you use the `CONSTRAINT [constraint_name] CHECK (expression)` format and don't give a name to the constraint, the constraint gets an automatically generated name. This is done so that you can later delete it with ALTER TABLE DROP *constraint*.

You can disable all constraint expression checks by setting the check_constraint_checks variable to `OFF`. This is useful for example when loading a table that violates some constraints that you want to later find and fix in SQL.

Replication

In row-based replication, only the master checks constraints, and failed statements are not replicated. In statement-based replication, the slaves also check constraints. Constraints should therefore be identical, as well as deterministic, in a replication environment.

Auto_increment

auto_increment columns are not permitted in check constraints.

Examples

Foreign Key Constraint

```
CREATE TABLE product (category INT NOT NULL, id INT NOT NULL,
                      price DECIMAL,
                      PRIMARY KEY(category, id)) ENGINE=INNODB;
CREATE TABLE customer (id INT NOT NULL,
                       PRIMARY KEY (id)) ENGINE=INNODB;
CREATE TABLE product_order (NO INT NOT NULL AUTO_INCREMENT,
                            product_category INT NOT NULL,
                            product_id INT NOT NULL,
                            customer_id INT NOT NULL,
                            PRIMARY KEY(NO),
                            INDEX (product_category, product_id),
                            FOREIGN KEY (product_category, product_id)
                              REFERENCES product(category, id)
                              ON UPDATE CASCADE ON DELETE RESTRICT,
                            INDEX (customer_id),
                            FOREIGN KEY (customer_id)
                              REFERENCES customer(id)) ENGINE=INNODB;
```

Numeric Constraints and Comparisons

```
CREATE TABLE t1 (a INT CHECK (a>2), b INT CHECK (b>2), CONSTRAINT a_greater CHECK (a>b));

INSERT INTO t1(a) VALUES (1);
ERROR 4022 (23000): CONSTRAINT `a` failed for `test`.`t1`

INSERT INTO t1(a,b) VALUES (3,4);
ERROR 4022 (23000): CONSTRAINT `a_greater` failed for `test`.`t1`

INSERT INTO t1(a,b) VALUES (4,3);
Query OK, 1 row affected (0.04 sec)
```

Adding Constraints

```
ALTER TABLE t1 ADD CONSTRAINT a_greater CHECK (a>b);
```

Dropping Constraints

```
ALTER TABLE t1 DROP CONSTRAINT a_greater;
```

Date Comparisons and Character Length

```
CREATE TABLE t2 (name VARCHAR(30) CHECK (CHAR_LENGTH(name)>2), start_date DATE,
  end_date DATE CHECK (start_date IS NULL OR end_date IS NULL OR start_date<end_date));

INSERT INTO t2(name, start_date, end_date) VALUES('Ione', '2003-12-15', '2014-11-09');
Query OK, 1 row affected (0.04 sec)

INSERT INTO t2(name, start_date, end_date) VALUES('Io', '2003-12-15', '2014-11-09');
ERROR 4022 (23000): CONSTRAINT `name` failed for `test`.`t2`

INSERT INTO t2(name, start_date, end_date) VALUES('Ione', NULL, '2014-11-09');
Query OK, 1 row affected (0.04 sec)

INSERT INTO t2(name, start_date, end_date) VALUES('Ione', '2015-12-15', '2014-11-09');
ERROR 4022 (23000): CONSTRAINT `end_date` failed for `test`.`t2`
```

Misplaced Parenthesis

```
CREATE TABLE t3 (name VARCHAR(30) CHECK (CHAR_LENGTH(name>2)), start_date DATE,
  end_date DATE CHECK (start_date IS NULL OR end_date IS NULL OR start_date<end_date));
Query OK, 0 rows affected (0.32 sec)

INSERT INTO t3(name, start_date, end_date) VALUES('Io', '2003-12-15', '2014-11-09');
Query OK, 1 row affected, 1 warning (0.04 sec)

SHOW WARNINGS;
+---------+------+-----------------------------------------+
| Level   | Code | Message                                 |
+---------+------+-----------------------------------------+
| Warning | 1292 | Truncated incorrect DOUBLE value: 'Io' |
+---------+------+-----------------------------------------+
```

In the last two examples, compare the definition of table *t2* to that of table *t3*. `CHAR_LENGTH(name)>2` is different from `CHAR_LENGTH(name>2)`, as the latter mistakenly performs a numeric comparison on the *name* field, leading to unexpected results.

See Also

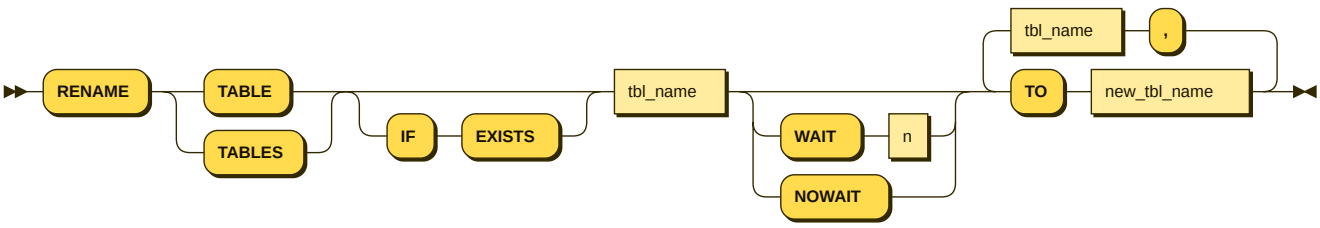
- Foreign Keys

This page is licensed: GPLv2, originally from fill_help_tables.sql

RENAME TABLE

Syntax

```
RENAME TABLE[S] [IF EXISTS] tbl_name
  [WAIT n | NOWAIT]
  TO new_tbl_name
    [, tbl_name2 TO new_tbl_name2] ...
```



Railroad diagram of RENAME TABLE — equivalent to the BNF above

Description

This statement renames one or more tables or views, but not the privileges associated with them. For InnoDB tables, it also triggers a reload of InnoDB statistics.

IF EXISTS

> **Note:**
>
> If this clause is used, you don't get an error if the table to be renamed doesn't exist.

The rename operation is done atomically, which means that no other session can access any of the tables while the rename is running. For example, if you have an existing table `old_table`, you can create another table `new_table` that has the same structure but is empty, and then replace the existing table with the empty one as follows (assuming that `backup_table` does not already exist):

```
CREATE TABLE new_table (...);
RENAME TABLE old_table TO backup_table, new_table TO old_table;
```

Moving Tables to Another Database

`tbl_name` can optionally be specified as `db_name.tbl_name`. See Identifier Qualifiers. This allows to use `RENAME` to move a table from a database to another (as long as they are on the same filesystem):

```
RENAME TABLE db1.t TO db2.t;
```

Limitations of Moving Tables to Another Database

> **Warning:**
>
> Note that moving a table to another database is not possible if it has triggers.

Trying to do so produces the following error:

```
ERROR 1435 (HY000): Trigger in wrong schema
```

> **Warning:**
>
> Views cannot be moved to another database.

Trying to do so produces the following error:

```
ERROR 1450 (HY000): Changing schema from 'old_db' to 'new_db' is not allowed.
```

Renaming Multiple Tables at Once

Multiple tables can be renamed in a single statement. The presence or absence of the optional `S` (`RENAME TABLE` or `RENAME TABLES`) has no impact, whether a single or multiple tables are being renamed.

> **Warning:**
>
> If a `RENAME TABLE` renames more than one table and one renaming fails, all renames executed by the same statement are rolled back.

Renames are always executed in the specified order. Knowing this, it is also possible to swap two tables' names:

```
RENAME TABLE t1 TO tmp_table,
    t2 TO t1,
    tmp_table TO t2;
```

WAIT/NOWAIT

Set the lock wait timeout. See WAIT and NOWAIT.

Privileges

Executing the `RENAME TABLE` statement requires the ALTER and DROP privileges on the original table, and the CREATE and INSERT privileges on the new table.

Atomic RENAME TABLE

> **CURRENT**
>
> `RENAME TABLE` is atomic for most storage engines, including InnoDB, MyRocks, MyISAM and Aria (MDEV-23842).
>
> This means that if there is a crash (server down or power outage) during `RENAME TABLE`, all tables revert to their original names and any changes to trigger files are reverted.
>
> See Atomic DDL for more information.
>
> **< 10.6.1**
>
> `RENAME TABLE` is **not** atomic.
>
> There is a small chance that, during a server crash happening in the middle of `RENAME TABLE`, some tables could have been renamed (in the worst case partly) while others would not be renamed.

This page is licensed: GPLv2, originally from fill_help_tables.sql

Renaming Databases

There is no `RENAME DATABASE` statement. To rename a database, use one of the following procedures.

Using RENAME TABLE

> **Important:**
>
> Use this procedure only if your tables have neither triggers nor views or events.

To move your tables from a database named db1 to one named db2, run these statements:

```
CREATE DATABASE db2;
# Do this for every table in database db1
RENAME TABLE db1.t TO db2.t;
# When no table is left in database db1, optionally drop it
DROP DATABASE db1;
```

> **Warning:**
>
> Privileges are neither dropped for `db1`, nor are they "copied" to `db2`. Use SHOW PRIVILEGES to see which privileges are related to `db1`, then apply those privileges to `db2`.

Renaming Databases Manually

To overcome the limitations of the previous procedure, you can do the following.

In the following steps, the source database is named `PROD` , and the destination database `TEST` .

- **Full Backup and Restore**: If restoring to a different server, use mariadb-backup to create a full backup, then restore it to the intended server. (This step is optional. It is not necessary if your renamed database is to reside on the same computer.)

> **Step 1**
>
> **Dump Logical Objects.**
>
> `RENAME TABLE` does not work for triggers, events, and routines. You need to dump these logical objects separately.
>
> ```
> mariadb-dump PROD --no-data --routines --triggers --events \
> > PROD_routines_triggers_events.sql
> ```

> **Step 2**
>
> **Generate RENAME TABLE Commands.**
>
> Run the following query to generate a script with the necessary `RENAME TABLE` statements. This is much faster than a full logical dump.
>
> ```
> mysql -ss -e"SELECT CONCAT('RENAME TABLE PROD.', TABLE_NAME, ' TO TEST.', \
> TABLE_NAME, ';') FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'PROD'" \
> > PROD_rename_table.sql
> ```

> **Step 3**
>
> **List all Existing Objects.**
>
> ```
> SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
>        WHERE TABLE_SCHEMA = 'PROD';
> SELECT EVENT_SCHEMA, EVENT_NAME FROM INFORMATION_SCHEMA.EVENTS
>        WHERE EVENT_SCHEMA = 'PROD';
> SELECT ROUTINE_SCHEMA, ROUTINE_NAME FROM INFORMATION_SCHEMA.ROUTINES
>        WHERE ROUTINE_SCHEMA = 'PROD';
> SELECT TRIGGER_SCHEMA, TRIGGER_NAME FROM INFORMATION_SCHEMA.TRIGGERS
>  WHERE TRIGGER_SCHEMA = 'PROD';
> ```

> **Step 4**
>
> **Create the new Database.**
>
> ```
> CREATE DATABASE TEST;
> ```

> **Step 5**
>
> **Run the rename_table Script.**
>
> ```
> mysql TEST < PROD_rename_table.sql
> ```

> **Step 6**
>
> **Restore Logical Objects.**
>
> After the rename script completes, restore the triggers, routines, and events into the new database.
>
> ```
> mysql TEST < PROD_routines_triggers_events.sql
> ```

Step 7

Verify all Objects are Restored.

Verify that all objects have been correctly moved to the new `TEST` database.

```
SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
       WHERE TABLE_SCHEMA = 'TEST';
SELECT EVENT_SCHEMA, EVENT_NAME FROM INFORMATION_SCHEMA.EVENTS
       WHERE EVENT_SCHEMA = 'TEST';
SELECT ROUTINE_SCHEMA, ROUTINE_NAME FROM INFORMATION_SCHEMA.ROUTINES
       WHERE ROUTINE_SCHEMA = 'TEST';
SELECT TRIGGER_SCHEMA, TRIGGER_NAME FROM INFORMATION_SCHEMA.TRIGGERS
       WHERE TRIGGER_SCHEMA = 'TEST';
```

Step 8

Cleanup.

Once you have confirmed everything looks good, you can drop the old `PROD` database.

> **Warning:**
>
> Privileges are neither dropped for `PROD`, nor are they "copied" to `TEST`. Use SHOW PRIVILEGES to see which privileges are related to `PROD`, then apply those privileges to `TEST`.

This page is licensed: CC BY-SA / Gnu FDL

ALTER

See ALTER DATABASE.

Modify database characteristics. Learn how to change global properties like the default character set and collation for a specific database.

See ALTER FUNCTION.

Change stored function characteristics. This statement allows modifying the security context or comments of a stored function without dropping it.

See ALTER LOGFILE GROUP.

Understand the support status of this statement. Originally designed for NDB Cluster, it is not currently supported in MariaDB Server.

See ALTER SERVER.

Modify server definitions. Update connection information for external servers defined with CREATE SERVER, primarily used by the Federated engine.

See ALTER TABLE.

Complete ALTER TABLE guide for MariaDB. Complete syntax for modifying columns, indexes, constraints, and table properties with comprehensive examples and.

See ALTER TABLESPACE.

Understand the status of tablespace management. This statement, originally for NDB, is not supported in MariaDB for InnoDB tablespaces.

See ALTER VIEW.

Documentation for the ALTER VIEW statement, which is used to modify an existing view's definition without dropping and recreating it.

This page is licensed: CC BY-SA / Gnu FDL

ALTER DATABASE

Syntax

```
ALTER {DATABASE | SCHEMA} [db_name]
    alter_specification ...
ALTER {DATABASE | SCHEMA} db_name
    UPGRADE DATA DIRECTORY NAME

alter_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | COMMENT [=] 'comment'
```

Description

`ALTER DATABASE` enables you to change the overall characteristics of a database. These characteristics are stored in the `db.opt` file in the database directory. To use `ALTER DATABASE`, you need the `ALTER` privilege on the database. `ALTER SCHEMA` is a synonym for `ALTER DATABASE`.

The `CHARACTER SET` clause changes the default database character set. The `COLLATE` clause changes the default database collation. See Character Sets and Collations for more.

You can see what character sets and collations are available using, respectively, the SHOW CHARACTER SET and SHOW COLLATION statements.

Changing the default character set/collation of a database does not change the character set/collation of any stored procedures or stored functions that were previously created, and relied on the defaults. These need to be dropped and recreated in order to apply the character set/collation changes.

The database name can be omitted from the first syntax, in which case the statement applies to the default database.

The syntax that includes the `UPGRADE DATA DIRECTORY NAME` clause. It updates the name of the directory associated with the database to use the encoding implemented in MySQL for mapping database names to database directory names (see Identifier to File Name Mapping). This clause is for use under these conditions:

- It is intended when upgrading MySQL from older versions.
- It is intended to update a database directory name to the current encoding format if the name contains special characters that need encoding.
- The statement is used by mariadb-check (as invoked by mariadb-upgrade).

For example, if a database in MySQL 5.0 has a name of `a-b-c`, the name contains instance of the `-' character. In 5.0, the database directory is also named `a-b-c`, which is not necessarily safe for all file systems. In MySQL, the same database name is encoded as `a@002db@002dc` to produce a file-system-neutral directory name.

When a MySQL installation is upgraded from an older version, the server displays a name such as `a-b-c` (which is in the old format) as `#mysql50#a-b-c`, and you must refer to the name using the `#mysql50#` prefix. Use `UPGRADE DATA DIRECTORY NAME` in this case to explicitly tell the server to re-encode the database directory name to the current encoding format:

```
ALTER DATABASE `#mysql50#a-b-c` UPGRADE DATA DIRECTORY NAME;
```

After executing this statement, you can refer to the database as a-b-c without the special `#mysql50` prefix.

Note:

MariaDB starting with 10.5.0

From MariaDB 10.5.0, it is possible to add a comment of a maximum of 1024 bytes. If the comment length exceeds this length, a error/warning code 4144 is thrown. The database comment is also added to the db.opt file, as well as to the information_schema.schemata table.

Examples

CURRENT

```
ALTER DATABASE p COMMENT='Presentations';
```

< 10.5

```
ALTER DATABASE test CHARACTER SET='utf8'  COLLATE='utf8_bin';
```

See Also

- CREATE DATABASE
- DROP DATABASE
- SHOW CREATE DATABASE
- SHOW DATABASES
- Character Sets and Collations
- Information Schema SCHEMATA Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

ALTER FUNCTION

Syntax

```
ALTER FUNCTION func_name [characteristic ...]

characteristic:
    { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'
```

Description

This statement can be used to change the characteristics of a stored function. More than one change may be specified in an `ALTER FUNCTION` statement. However, you cannot change the parameters or body of a stored function using this statement; to make such changes, you must drop and re-create the function using DROP FUNCTION and CREATE FUNCTION.

You must have the `ALTER ROUTINE` privilege for the function. (That privilege is granted automatically to the function creator.) If binary logging is enabled, the `ALTER FUNCTION` statement might also require the `SUPER` privilege, as described in Binary Logging of Stored Routines.

Example

```
ALTER FUNCTION hello SQL SECURITY INVOKER;
```

See Also

- CREATE FUNCTION
- SHOW CREATE FUNCTION
- DROP FUNCTION
- SHOW FUNCTION STATUS
- Information Schema ROUTINES Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

ALTER LOGFILE GROUP

> **Note:**
>
> The `ALTER LOGFILE GROUP` statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. See MDEV-19295 for more information.

Syntax

```
ALTER LOGFILE GROUP logfile_group
    ADD UNDOFILE 'file_name'
    [INITIAL_SIZE [=] size]
    [WAIT]
    ENGINE [=] engine_name
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ALTER SERVER

Syntax

```
ALTER SERVER server_name
    OPTIONS (option [, option] ...)
```

Description

Alters the server information for *server_name*, adjusting the specified options as per the CREATE SERVER command. The corresponding fields in the mysql.servers table are updated accordingly. This statement requires the SUPER privilege or, from MariaDB 10.5.2, the FEDERATED ADMIN privilege.

ALTER SERVER is not written to the binary log, irrespective of the binary log format being used. From MariaDB 10.1.13, Galera replicates the CREATE SERVER, ALTER SERVER and DROP SERVER statements.

Examples

```
ALTER SERVER s OPTIONS (USER 'sally');
```

See Also

- CREATE SERVER
- DROP SERVER
- Spider Storage Engine
- mysql.servers table

This page is licensed: GPLv2, originally from fill_help_tables.sql

Online Schema Change

> **Note:**
>
> This functionality is available from MariaDB 11.4.

The term *Online Schema Change* refers to the ability to update the table schema without blocking concurrent DML[39] statements for the duration of the schema change.

Example

Starting with a table named `item` which has 2 columns (`id SERIAL` , `name TEXT`), we add a new `TIMESTAMP` column. While the change is performed, a concurrent connection inserts a new row.

Connection 1:

```
ALTER TABLE items ADD ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP;
```

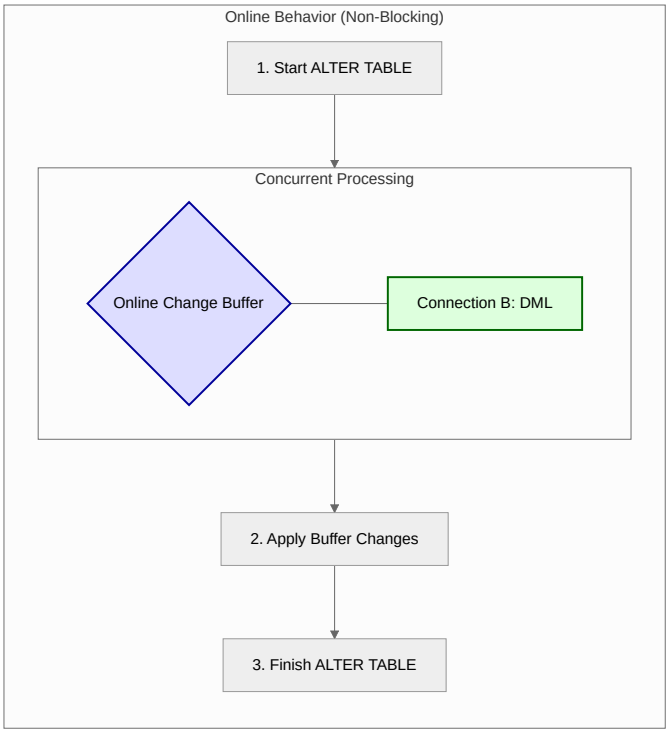
Connection 2:

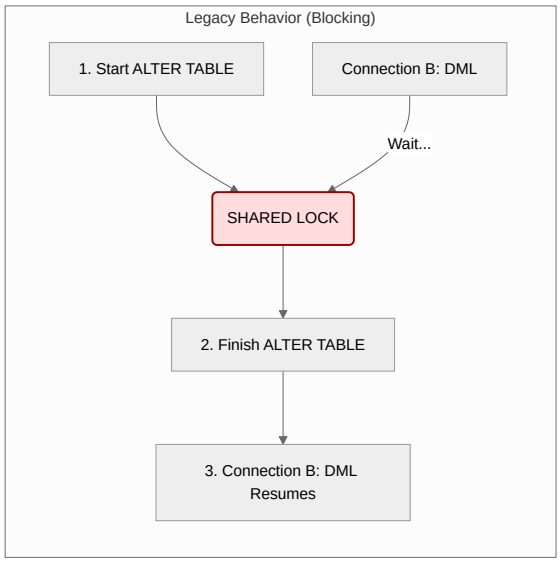
```
INSERT INTO items(name) VALUES (“New item”);
```

Previous Behavior

If the `INSERT` statement starts after the `ALTER TABLE` statement, it is not blocked, but proceeds normally, and may finish before the `ALTER TABLE` statement.

`ALTER TABLE` always allows concurrent SELECT statements. If the `LOCK=NONE` locking strategy is chosen, it allows concurrent modifications via DML[40] statements like `INSERT` , `DELETE` , or `UPDATE` . `LOCK=NONE` is supported by the InnoDB and the Partition engine when `ALGORITHM=NOCOPY` is chosen, and is a default locking strategy when available.



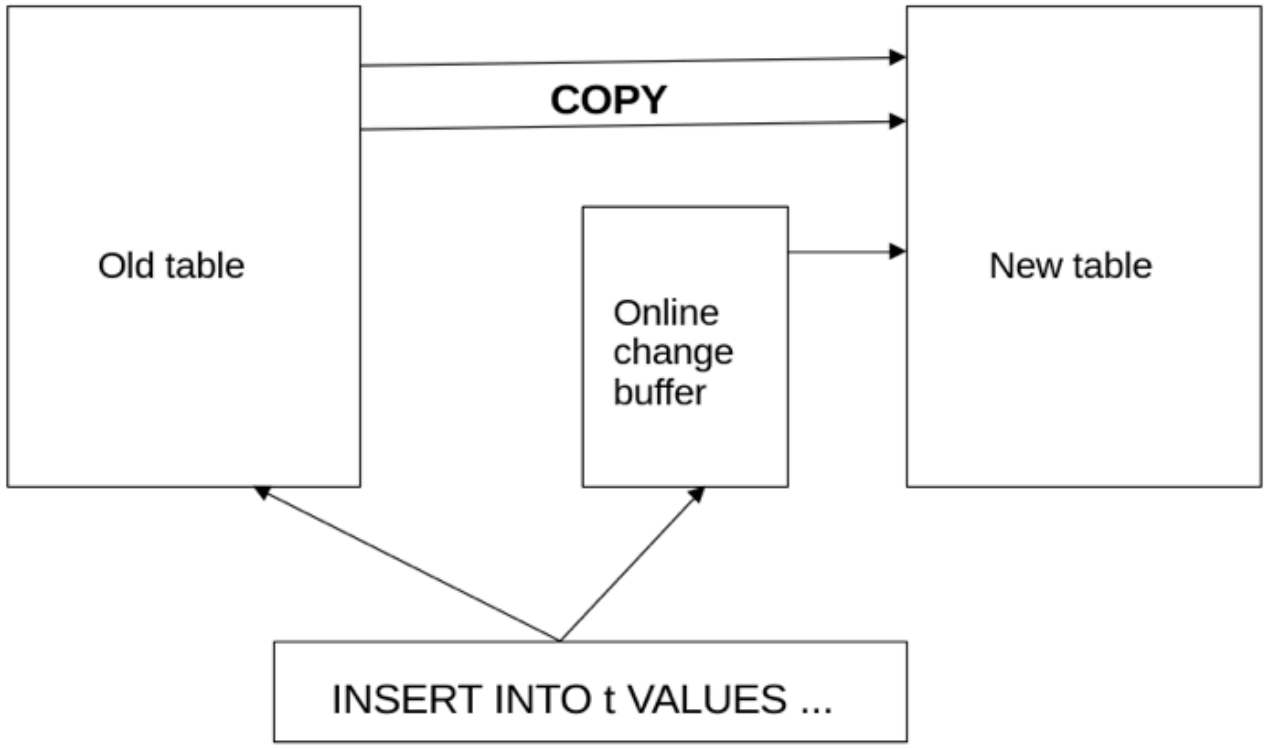


New Behavior

- From MariaDB 11.2, ALTER TABLE is universally online.
- In most cases, unless explicitly specified otherwise, `ALTER TABLE` implicitly operates as if `ALGORITHM=COPY, LOCK=NONE` is set.
- This means it uses the `COPY` algorithm while simultaneously allowing concurrent DML statements on the altered table.
- A few exceptions to this rule are described under Limitations.

Mechanism

`LOCK=NONE` adds one extra step to the copy algorithm. It introduces a new internal entity, the **online change buffer**.



The online change buffer works like this:

1. A new table is created using the old table content as it was like at the beginning of the `ALTER TABLE` statement. For InnoDB and other transactional engines, this means copying in `REPEATABLE READ` isolation mode.
2. In the meantime, every concurrent change is written in the old table and is duplicated to the online change buffer.
3. `ALTER TABLE` applies any accumulated changes to the online change buffer.

> **Note:**
>
> While all copying and online changes application happens without blocking concurrent `DML`, `ALTER TABLE` acquires an `EXCLUSIVE` lock on the table for a short amount of time, which is necessary to synchronize with all parallel operations that have not yet finished.

Online Change Buffer Details

- Storage and Scope: Despite the name, the buffer is not purely in-memory; it is implemented as a temporary file on the filesystem using MariaDB's `create_temp_file` function. It is created on a per-table basis, rather than globally or per-session. The file is typically stored in the directory defined by the `TMPDIR` environment variable, or the OS default temporary path (like `GetTempPath` on Windows).

- Size Limits: The online schema change reuses the temporary buffer mechanism used in the binlog for transaction/statement caches. There is currently no way to explicitly limit its maximum size via a system variable; it is only bounded by available disk space and OS-specific limits, such as a 4GB limit on a 32-bit machine.
- Disk Full Risks: Because there is no explicit size limit, heavy concurrent DML during a long schema change can fill up the disk hosting the temporary directory. This behaves similarly to a sort buffer for unindexed `SELECT` statements causing a disk full event, which could hang operations.
- Crash Behavior: If the server crashes during the `ALTER` statement, no manual cleanup is required. The temporary files are created with OS-level ephemeral flags (like `O_TMPFILE` on supported Linux systems or `O_TEMPORARY | O_SHORT_LIVED` on Windows) or are unlinked immediately upon creation. This ensures the operating system automatically reclaims the file space.

Monitoring and Troubleshooting

- Currently, MariaDB does not provide a native mechanism to monitor or troubleshoot the online change buffer.
- There are no specific `STATUS` variables available to track the size or usage of this buffer.
- Furthermore, there are no corresponding tables in the `information_schema` or `performance_schema` to collect this data from a running server.
- Administrators must proactively monitor the available disk space of the filesystem hosting the temporary directory (`tmpdir`) during large `ALTER TABLE` operations to prevent disk exhaustion.

Supported Storage Engines

For the new behavior to work, engines must allow concurrent writes, while other connections can read and have a repeatable-read behavior.

MyISAM and Aria

MyISAM is capable of concurrent INSERTsand also supports concurrent inserts while reading. It doesn't support a REPEATABLE READ transaction isolation layer, but newly inserted data isn't visible to other (reading) clients until the end of the statement. Therefore, MyISAM and Aria satisfy the criterion, even though those engines aren't transactional.

Statements that update or delete rows are blocked until the end of `ALTER TABLE` (or evaluate before `ALTER TABLE` acquires a lock, if it started earlier).

InnoDB

InnoDB supports `REPEATABLE READ` isolation layer and allows concurrent writes, which means online schema change works, greatly improving its online ability represented by `INPLACE` and `INSTANT` algorithms. By default, an engine-native algorithm is chosen whenever possible. If InnoDB isn't capable of holding the schema change natively, a `COPY` algorithm using online schema change is applied. This is described in detail here.

Other Storage Engines

Many other storage engines do not support online schema change, for a variety of reasons:

- ColumnStore does not support `REPEATABLE READ` isolation layer.
- Blackhole and Sequence don't support `LOCK=NONE` , or are read-only, like S3.
- Connect cannot perform reads during write operations.
- Spider does not perform table-copy operations, but rather reassigns a table with a new set of metadata.

Online Schema Change vs. InnoDB INPLACE

Everything that can be done by InnoDB's `INPLACE` can also be done by online `COPY`. However, tables with foreign keys with cascade operations are not supported (see Limitations).

- Changing the column's data type is possible.
- A column can be modified to `NOT NULL`.
- A column with non-constant `DEFAULT` value can be added.
- A `STORED` generated column can be added.
- A `CHECK` constraint can be added.
- A `PRIMARY KEY` can be dropped.
- No key is required in the resulting table.
- A table can be changed to be partitioned, and partitioning can be removed.
- `SYSTEM VERSIONING` can be added.
- `AUTO_INCREMENT` can be added to the existing column (however, see Limitations).

Limitations

The following limitations apply:

- AUTO_INCREMENT columns cannot be *added*, neither can `DEFAULT (…NEXTVAL(…))`. Changing an existing column to `AUTO_INCREMENT` is allowed, though. The two following conditions should be met:
  - The column must be `NOT NULL` in the old schema.
  - The `NO_AUTO_VALUE_ON_ZERO` mode must be unset.
  - At least one `UNIQUE NOT NULL` key must be in the old schema, and remain unchanged.
- `ALTER IGNORE TABLE` is not implemented. This operation can make the altered table have fewer rows than the original one (because duplicates are skipped), and the online copy algorithm doesn't support operations that change the number of rows in the table.
- `ALTER TABLE DROP SYSTEM VERSIONING` is not supported either, for the same reason.
- Tables with foreign keys with `CASCADE/SET NULL/SET DEFAULT` operations can't go online under `ALGORITHM=COPY` — similar to disallowing `CHECK` constraints and stored generated columns, cascade operations are done internally by the storage engine, so they bypass the online changes buffer. However, most operations are allowed by the InnoDB `INPLACE` algorithm. For each DDL[41] request, the most optimal algorithm is chosen automatically.
- Transaction-versioned tables can only be changed under `INPLACE/INSTANT ALTER TABLE` algorithms.
- All the constraints (`CHECK, UNIQUE, FOREIGN KEY`) are evaluated for each row change that is applied from the online change buffer. This means that changes that are made during `ALTER TABLE's` main phase must not violate the final table schema at any point in time. Checks can be disabled by setting `check_constraint_checks` and `FOREIGN_KEY_CHECKS` to `OFF`.
- `ALTER TABLE … ORDER BY` isn't supported because changes from the buffer are applied at the end, which might break the strict ordering of rows.
- Online schema change isn't available in embedded MariaDB Server due to replication libraries that are not present in that variant.

Switching Back

> **CURRENT**
>
> Online copy is the default mode whenever `NOCOPY` does not apply. In case of any problem with it, it can be disabled by specifying `LOCK=SHARED` to force the usual `COPY` algorithm.
>
> Additionally, to better support existing workflows, there is a new `old_mode` flag. The following statement disables online copy by default:
>
> ```
> SET old_mode= LOCK_ALTER_TABLE_COPY;
> ```

> With that mode enabled, you can still use the new behavior, by explicitly specifying `LOCK=NONE` in the `ALTER TABLE` statement.
>
> Server-wide online schema change expands MariaDB Server's capability for the `LOCK=NONE` approach broadly.
>
> **< 11.2**
>
> Online copy is **not** the default mode.

See Also

For more information, read this blog post: Reduced operational downtime with new ALTER TABLE

This page is licensed: CC BY-SA / Gnu FDL

ALTER TABLESPACE

> **Note:**
>
> The `ALTER TABLESPACE` statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. In MySQL 5.7 and later, the statement is also supported for InnoDB. However, MariaDB has chosen not to include that specific feature. See MDEV-19294 for more information.

This page is licensed: GPLv2, originally from fill_help_tables.sql

CREATE

- CREATE CATALOG
- CREATE DATABASE
- CREATE EVENT
- CREATE FUNCTION
- CREATE FUNCTION UDF
- CREATE INDEX
- CREATE LOGFILE GROUP
- CREATE PACKAGE
- CREATE PACKAGE BODY
- CREATE PROCEDURE
- CREATE ROLE
- CREATE SEQUENCE
- CREATE SERVER
- CREATE TABLE
- CREATE TABLE with Vectors
- CREATE TABLESPACE
- CREATE TRIGGER
- CREATE USER
- CREATE VIEW
- Generated Columns
- Invisible Columns
- Silent Column Changes

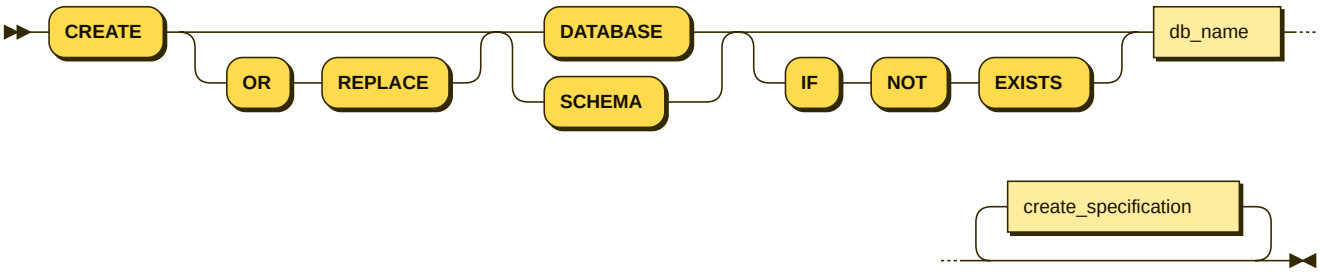
This page is licensed: CC BY-SA / Gnu FDL

CREATE DATABASE

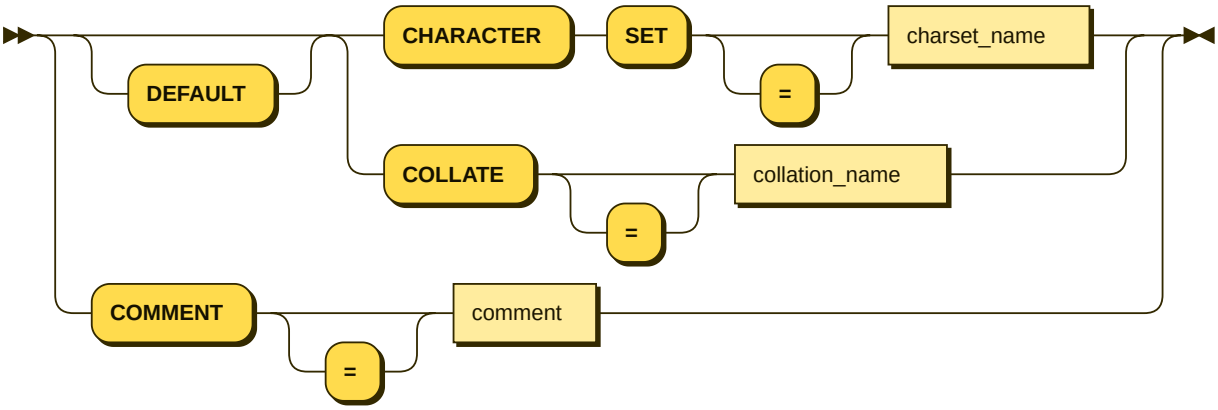
Syntax

```
CREATE [OR REPLACE] {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | COMMENT [=] 'comment'
```



Railroad diagram of CREATE DATABASE — equivalent to the BNF above



Railroad diagram of create_specification

Description

`CREATE DATABASE` creates a database with the given name. To use this statement, you need the CREATE privilege for the database. `CREATE SCHEMA` is a synonym for `CREATE DATABASE`.

For valid identifiers to use as database names, see Identifier Names.

OR REPLACE

If the optional `OR REPLACE` clause is used, it acts as a shortcut for:

```
DROP DATABASE IF EXISTS db_name;
CREATE DATABASE db_name ...;
```

IF NOT EXISTS

When the `IF NOT EXISTS` clause is used, MariaDB will return a warning instead of an error if the specified database already exists.

COMMENT

> **CURRENT**

> The maximum length of a comment is 1024 bytes. If the comment length exceeds this length, an error/warning code 4144 is thrown. The database comment is also added to the `db.opt` file, as well as to the information_schema.schemata table.
>
> **< 10.5.0**
>
> Comments added for databases do not exist.

Examples

```
CREATE DATABASE db1;
Query OK, 1 row affected (0.18 sec)

CREATE DATABASE db1;
ERROR 1007 (HY000): Can't create database 'db1'; database exists

CREATE OR REPLACE DATABASE db1;
Query OK, 2 rows affected (0.00 sec)

CREATE DATABASE IF NOT EXISTS db1;
Query OK, 1 row affected, 1 warning (0.01 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------------+
| Level | Code | Message                                       |
+-------+------+-----------------------------------------------+
| Note  | 1007 | Can't create database 'db1'; database exists |
+-------+------+-----------------------------------------------+
```

Setting the character sets and collation. See Setting Character Sets and Collations for more details.

```
CREATE DATABASE czech_slovak_names
  CHARACTER SET = 'keybcs2'
  COLLATE = 'keybcs2_bin';
```

```
CREATE DATABASE presentations COMMENT 'Presentations for conferences';
```

See Also

- Identifier Names
- DROP DATABASE
- SHOW CREATE DATABASE
- ALTER DATABASE
- SHOW DATABASES
- Character Sets and Collations
- Information Schema SCHEMATA Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

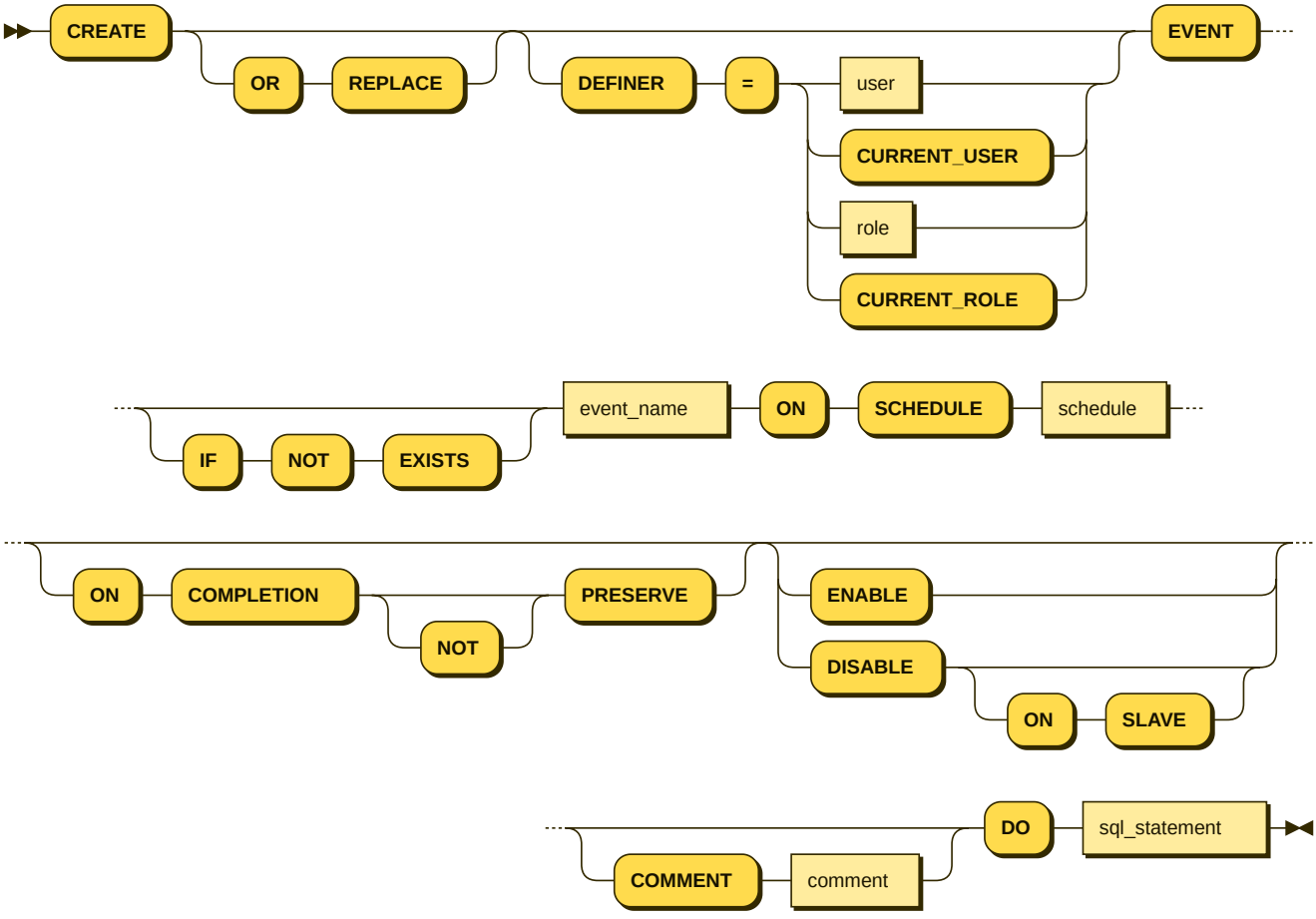
CREATE EVENT

Syntax

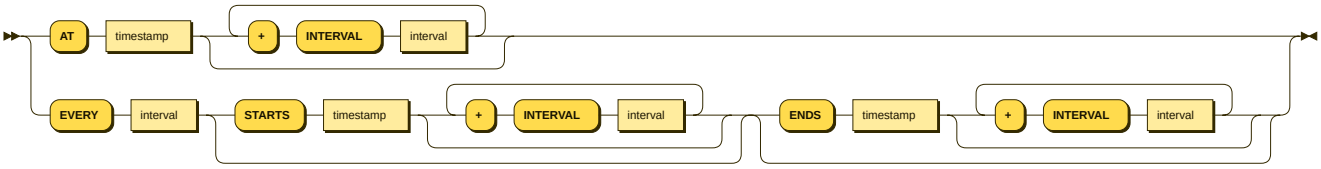
```
CREATE [OR REPLACE]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    EVENT
    [IF NOT EXISTS]
    event_name
    ON SCHEDULE schedule
    [ON COMPLETION [NOT] PRESERVE]
    [ENABLE | DISABLE | DISABLE ON SLAVE]
    [COMMENT 'comment']
    DO sql_statement

schedule:
    AT timestamp [+ INTERVAL interval] ...
  | EVERY interval
    [STARTS timestamp [+ INTERVAL interval] ...]
    [ENDS timestamp [+ INTERVAL interval] ...]

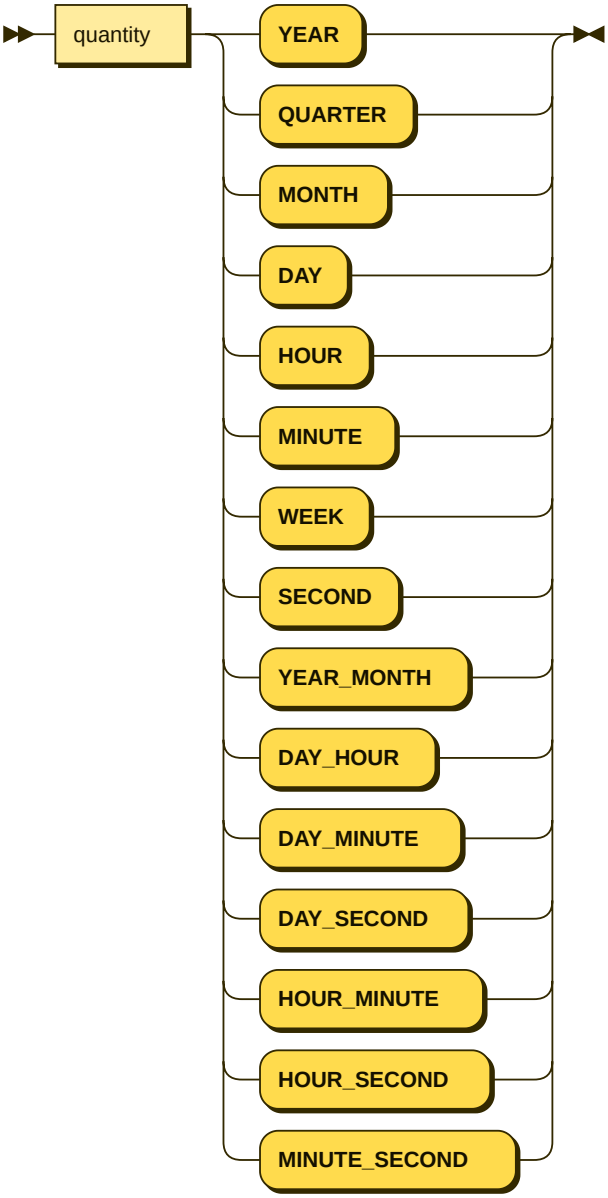
interval:
    quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
              WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
              DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```



Railroad diagram of CREATE EVENT — equivalent to the BNF above



Railroad diagram of schedule



Railroad diagram of interval

Description

This statement creates and schedules a new event. It requires the EVENT privilege for the schema in which the event is to be created.

The minimum requirements for a valid CREATE EVENT statement are as follows:

- The keywords `CREATE EVENT` plus an event name, which uniquely identifies the event in the current schema.
- An `ON SCHEDULE` clause, which determines when and how often the event executes.
- A `DO` clause, which contains the SQL statement to be executed by an event.

Here is an example of a minimal `CREATE EVENT` statement:

```
CREATE EVENT myevent
    ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
    DO
      UPDATE myschema.mytable SET mycol = mycol + 1;
```

The previous statement creates an event named myevent. This event executes once
— one hour following its creation
— by running an SQL statement that increments the value of the `myschema.mytable` table's `mycol` column by 1.

The event_name must be a valid MariaDB identifier with a maximum length of 64 characters. It may be delimited using back ticks, and may be qualified with the name of a database schema. An event is associated with both a MariaDB user (the definer) and a schema, and its name must be unique among names of events within that schema. In general, the rules governing event names are the same as those for names of stored routines. See Identifier Names.

If no schema is indicated as part of event_name, the default (current) schema is assumed.

For valid identifiers to use as event names, see Identifier Names.

OR REPLACE

CURRENT

The `OR REPLACE` clause works like this: If the event already exists, instead of an error being returned, the existing event will be dropped and replaced by the newly defined event.

< 10.1.4

The `OR REPLACE` clause is not available.

IF NOT EXISTS

If the `IF NOT EXISTS` clause is used, MariaDB will return a warning instead of an error if the event already exists. Cannot be used together with OR REPLACE.

ON SCHEDULE

The `ON SCHEDULE` clause can be used to specify when the event must be triggered.

AT

If you want to execute the event only once (one time event), you can use the `AT` keyword, followed by a timestamp. If you use CURRENT_TIMESTAMP, the event acts as soon as it is created. As a convenience, you can add one or more intervals to that timestamp. You can also specify a timestamp in the past, so that the event is stored but not triggered, until you modify it via ALTER EVENT.

The following example shows how to create an event that will be triggered tomorrow at a certain time:

```
CREATE EVENT example
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 DAY + INTERVAL 3 HOUR
DO something;
```

You can also specify that an event must be triggered at a regular interval (recurring event). In such cases, use the `EVERY` clause followed by the interval.

If an event is recurring, you can specify when the first execution must happen via the `STARTS` clause and a maximum time for the last execution via the `ENDS` clause. `STARTS` and `ENDS` clauses are followed by a timestamp

and, optionally, one or more intervals. The `ENDS` clause can specify a timestamp in the past, so that the event is stored but not executed until you modify it via ALTER EVENT.

In the following example, next month a recurring event will be triggered hourly for a week:

```
CREATE EVENT example
ON SCHEDULE EVERY 1 HOUR
STARTS CURRENT_TIMESTAMP + INTERVAL 1 MONTH
ENDS CURRENT_TIMESTAMP + INTERVAL 1 MONTH + INTERVAL 1 WEEK
DO some_task;
```

Intervals consist of a quantity and a time unit. The time units are the same used for other statements and time functions, except that you can't use microseconds for events. For simple time units, like `HOUR` or `MINUTE`, the quantity is an integer number, for example '10 MINUTE'. For composite time units, like `HOUR_MINUTE` or `HOUR_SECOND`, the quantity must be a string with all involved simple values and their separators, for example '2:30' or '2:30:30'.

ON COMPLETION [NOT] PRESERVE

The `ON COMPLETION` clause can be used to specify if the event must be deleted after its last execution (that is, after its `AT` or `ENDS` timestamp is past). By default, events are dropped when they are expired. To explicitly state that this is the desired behaviour, you can use `ON COMPLETION NOT PRESERVE`. Instead, if you want the event to be preserved, you can use `ON COMPLETION PRESERVE`.

In you specify `ON COMPLETION NOT PRESERVE`, and you specify a timestamp in the past for `AT` or `ENDS` clause, the event will be immediately dropped. In such cases, you will get a Note 1558: "Event execution time is in the past and ON COMPLETION NOT PRESERVE is set. The event was dropped immediately after creation".

ENABLE/DISABLE/DISABLE ON SLAVE

Events are `ENABLE`d by default. If you want to stop MariaDB from executing an event, you may specify `DISABLE`. When it is ready to be activated, you may enable it using ALTER EVENT. Another option is `DISABLE ON SLAVE`, which indicates that an event was created on a master and has been replicated to the slave, which is prevented from executing the event. If `DISABLE ON SLAVE` is specifically set, the event will be disabled everywhere. It will not be executed on the master or the replicas.

COMMENT

The `COMMENT` clause may be used to set a comment for the event. Maximum length for comments is 64 characters. The comment is a string, so it must be quoted. To see events comments, you can query the INFORMATION_SCHEMA.EVENTS table (the column is named `EVENT_COMMENT`).

Examples

Minimal `CREATE EVENT` statement:

```
CREATE EVENT myevent
    ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
    DO
      UPDATE myschema.mytable SET mycol = mycol + 1;
```

An event that will be triggered tomorrow at a certain time:

```
CREATE EVENT example
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 DAY + INTERVAL 3 HOUR
DO something;
```

Next month a recurring event will be triggered hourly for a week:

```
CREATE EVENT example
ON SCHEDULE EVERY 1 HOUR
STARTS CURRENT_TIMESTAMP + INTERVAL 1 MONTH
ENDS CURRENT_TIMESTAMP + INTERVAL 1 MONTH + INTERVAL 1 WEEK
DO some_task;
```

OR REPLACE and IF NOT EXISTS:

```
CREATE EVENT myevent
    ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
    DO
      UPDATE myschema.mytable SET mycol = mycol + 1;
ERROR 1537 (HY000): Event 'myevent' already exists

CREATE OR REPLACE EVENT myevent
    ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
    DO
      UPDATE myschema.mytable SET mycol = mycol + 1;;
Query OK, 0 rows affected (0.00 sec)

CREATE EVENT IF NOT EXISTS myevent
    ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
    DO
      UPDATE myschema.mytable SET mycol = mycol + 1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

 SHOW WARNINGS;
+-------+------+--------------------------------+
| Level | Code | Message                        |
+-------+------+--------------------------------+
| Note  | 1537 | Event 'myevent' already exists |
+-------+------+--------------------------------+
```

See Also

- Event Limitations
- Identifier Names
- Events Overview
- SHOW CREATE EVENT
- ALTER EVENT
- DROP EVENT

This page is licensed: GPLv2, originally from fill_help_tables.sql

CREATE FUNCTION

Syntax

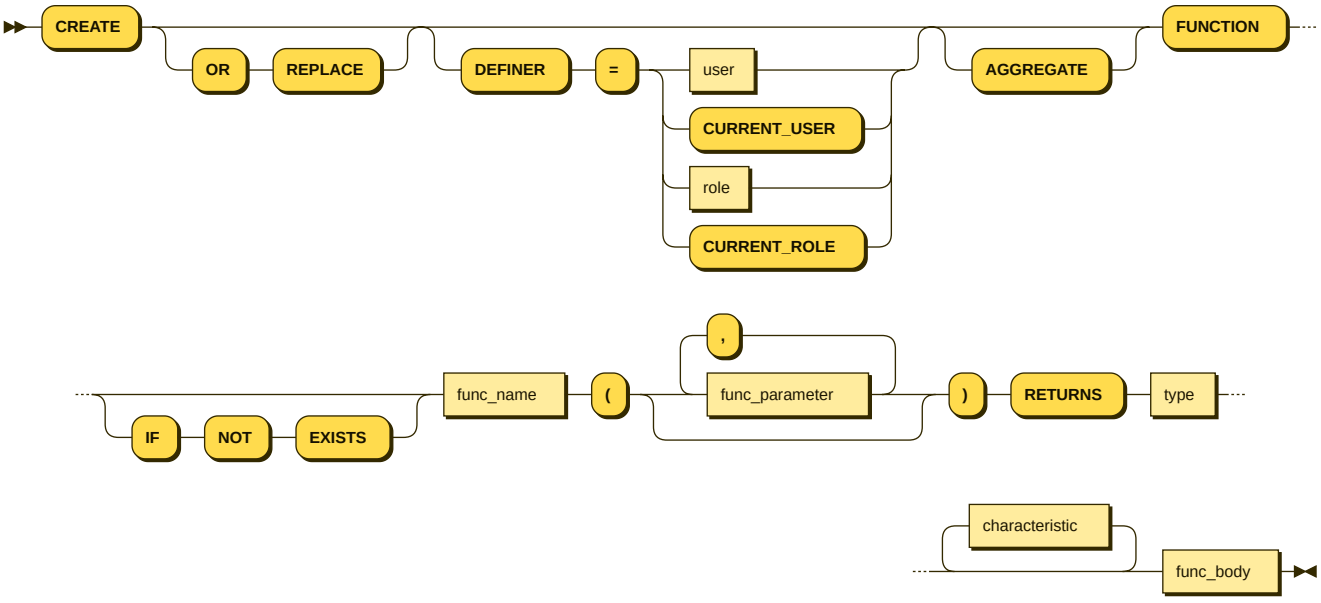
```
CREATE [OR REPLACE]
    [DEFINER = {user | CURRENT_USER | role | CURRENT_ROLE }]
    [AGGREGATE] FUNCTION [IF NOT EXISTS] func_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...]
    func_body

func_parameter:
    [ OUT | INOUT | IN OUT ] param_name type |
    [ IN ] param_name type [DEFAULT value or expression]

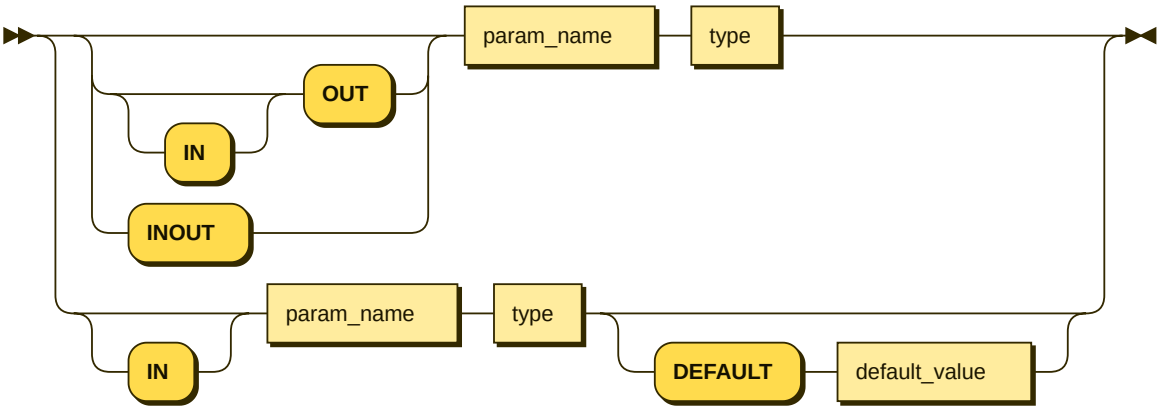
type:
    Any valid MariaDB data type

characteristic:
    LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'

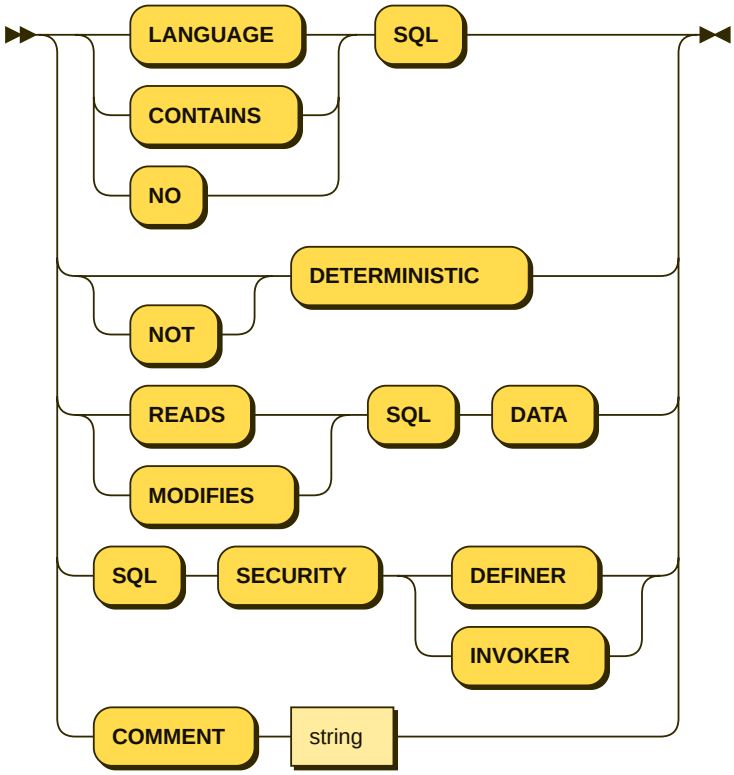
func_body:
    Valid SQL procedure statement
```



Railroad diagram of CREATE FUNCTION — equivalent to the BNF above



Railroad diagram of func_parameter



Railroad diagram of characteristic

Description

Use the `CREATE FUNCTION` statement to create a new stored function. You must have the CREATE ROUTINE database privilege to use `CREATE FUNCTION`. A function takes any number of arguments and returns a value from the function body. The function body can be any valid SQL expression as you would use, for example, in any select expression. If you have the appropriate privileges, you can call the function exactly as you would any built-in function. See Security below for details on privileges.

You can also use a variant of the `CREATE FUNCTION` statement to install a user-defined function (UDF) defined by a plugin. See CREATE FUNCTION (UDF) for details.

You can use a SELECT statement for the function body by enclosing it in parentheses, exactly as you would to use a subselect for any other expression. The `SELECT` statement must return a single value. If more than one column is returned when the function is called, error 1241 results. If more than one row is returned when the function is called, error 1242 results. Use a `LIMIT` clause to ensure only one row is returned.

You can also replace the `RETURN` clause with a BEGIN…END compound statement. The compound statement must contain a `RETURN` statement. When the function is called, the `RETURN` statement immediately returns its result, and any statements after `RETURN` are effectively ignored.

By default, a function is associated with the current database. To associate the function explicitly with a given database, specify the fully-qualified name as `db_name.func_name` when you create it. If the function name is the same as the name of a built-in function, you must use the fully qualified name when you call it.

The parameter list enclosed within parentheses must always be present. If there are no parameters, an empty parameter list of `()` should be used. Parameter names are not case-sensitive.

For valid identifiers to use as function names, see Identifier Names.

RETURN

The `RETURN` clause can return a function body. In newer versions of MariaDB, it can also return a cursor (using `SYS_REFCURSOR`).

CURRENT

RETURN `func_body`

The `RETURN` clause can return a function body (a valid SQL `PROCEDURE` statement).

RETURN `SYS_REFCURSOR`

It can also return a cursor, if the function was declared with a `SYS_REFCURSOR` variable, like this:

```
CREATE FUNCTION f1 RETURN SYS_REFCURSOR AS
  c SYS_REFCURSOR;
BEGIN
  OPEN c FOR SELECT a FROM t1 ORDER BY a;
  RETURN c;
END;
/
```

Alternatively, a cursor can be returned in an `OUT` parameter, see this section.

< COMMUNITY SERVER 12.0 / ENTERPRISE SERVER 11.8

RETURN `func_body`

The `RETURN` clause can return a function body (a valid SQL `PROCEDURE` statement).

AGGREGATE

It is possible to create stored aggregate functions as well. See Stored Aggregate Functions for details.

IN | OUT | INOUT | IN OUT

CURRENT

`OUT`, `INOUT` and its equivalent `IN OUT`, are only valid if called from `SET` and not `SELECT`. These quantifiers are especially useful for creating functions with more than one return value. This allows functions to be more complex and nested.

```
DELIMITER $$
CREATE FUNCTION add_func3(IN a INT, IN b INT, OUT c INT) RETURNS INT
BEGIN
  SET c = 100;
  RETURN a + b;
END;
$$
DELIMITER ;

SET @a = 2;
SET @b = 3;
SET @c = 0;
SET @res= add_func3(@a, @b, @c);

SELECT add_func3(@a, @b, @c);
ERROR 4186 (HY000): OUT or INOUT argument 3 for function add_func3 is not allowed here

DELIMITER $$
CREATE FUNCTION add_func4(IN a INT, IN b INT, d INT) RETURNS INT
BEGIN
  DECLARE c, res INT;
  SET res = add_func3(a, b, c) + d;
  if (c > 99) then
    return  3;
  else
    return res;
  end if;
END;
$$

DELIMITER ;

SELECT add_func4(1,2,3);
+------------------+
| add_func4(1,2,3) |
+------------------+
|                3 |
+------------------+
```

< 10.8.0

Quantifiers are not available.

CURRENT

A cursor can be returned, like this:

```
CREATE TABLE t1 (a INT);
INSERT INTO t1 VALUES (10),(20);

CREATE FUNCTION f1 RETURN SYS_REFCURSOR AS
  c SYS_REFCURSOR;
BEGIN
  OPEN c FOR SELECT a FROM t1 ORDER BY a;
  RETURN c;
END;
/

CREATE PROCEDURE p1 AS
  a INT;
  c SYS_REFCURSOR DEFAULT f1();
BEGIN
  LOOP
    FETCH c INTO a;
    EXIT WHEN c%NOTFOUND;
    dbms_output.put_line(a);
  END LOOP;
  CLOSE c;
  END;
/

CALL p1();
```

< COMMUNITY SERVER 12.0 / ENTERPRISE SERVER 11.8

This feature is not available.

RETURNS

The `RETURNS` clause specifies the return type of the function. `NULL` values are permitted with all return types.

What happens if the `RETURN` clause returns a value of a different type? It depends on the SQL_MODE in effect at the moment of the function creation.

If the `SQL_MODE` is strict (`STRICT_ALL_TABLES` or `STRICT_TRANS_TABLES` flags are specified), a 1366 error will be produced.

Otherwise, the value is coerced to the proper type. For example, if a function specifies an `ENUM` or `SET` value in the `RETURNS` clause, but the `RETURN` clause returns an integer, the value returned from the function is the string for the corresponding `ENUM` member of set of `SET` members.

MariaDB stores the `SQL_MODE` system variable setting that is in effect at the time a routine is created, and always executes the routine with this setting in force, regardless of the server SQL mode in effect when the routine is invoked.

LANGUAGE SQL

`LANGUAGE SQL` is a standard SQL clause which can be used in MariaDB for portability. However that clause has no meaning, because SQL is the only supported language for stored functions.

A function is deterministic if it can produce only one result for a given list of parameters. If the result may be affected by stored data, server variables, random numbers or any value that is not explicitly passed, then the function is not deterministic. Also, a function is non-deterministic if it uses nondeterministic functions like NOW() or CURRENT_TIMESTAMP(). The optimizer may choose a faster execution plan if it known that the function is deterministic. In such cases, you should declare the routine using the `DETERMINISTIC` keyword. If you want to explicitly state that the function is not deterministic (which is the default) you can use the `NOT DETERMINISTIC` keywords.

If you declare a non-deterministic function as `DETERMINISTIC`, you may get incorrect results. If you declare a deterministic function as `NOT DETERMINISTIC`, in some cases the queries will be slower.

OR REPLACE

If the optional `OR REPLACE` clause is used, it acts as a shortcut for:

```
DROP FUNCTION IF EXISTS function_name;
CREATE FUNCTION function_name ...;
```

with the exception that any existing privileges for the function are not dropped.

IF NOT EXISTS

If the `IF NOT EXISTS` clause is used, MariaDB will return a warning instead of an error if the function already exists. Cannot be used together with OR REPLACE.

[NOT] DETERMINISTIC

The `[NOT] DETERMINISTIC` clause also affects binary logging, because the `STATEMENT` format can not be used to store or replicate non-deterministic statements.

`CONTAINS SQL`, `NO SQL`, `READS SQL DATA`, and `MODIFIES SQL DATA` are informative clauses that tell the server what the function does. MariaDB does not check in any way whether the specified clause is correct. If none of these clauses are specified, `CONTAINS SQL` is used by default.

MODIFIES SQL DATA

`MODIFIES SQL DATA` means that the function contains statements that may modify data stored in databases. This happens if the function contains statements like DELETE, UPDATE, INSERT, REPLACE or DDL.

READS SQL DATA

`READS SQL DATA` means that the function reads data stored in databases, but does not modify any data. This happens if SELECT statements are used, but there no write operations are executed.

CONTAINS SQL

`CONTAINS SQL` means that the function contains at least one SQL statement, but it does not read or write any data stored in a database. Examples include SET or DO.

NO SQL

`NO SQL` means nothing, because MariaDB does not currently support any language other than SQL.

Oracle Mode

A subset of Oracle's PL/SQL language is supported in addition to the traditional SQL/PSM-based MariaDB syntax. See Oracle mode for details on changes when running Oracle mode.

Security

You must have the EXECUTE privilege on a function to call it. MariaDB automatically grants the `EXECUTE` and `ALTER ROUTINE` privileges to the account that called `CREATE FUNCTION`, even if the `DEFINER` clause was used.

Each function has an account associated as the definer. By default, the definer is the account that created the function. Use the `DEFINER` clause to specify a different account as the definer. You must have the SET USER privilege to use the `DEFINER` clause. See Account Names for details on specifying accounts.

The `SQL SECURITY` clause specifies what privileges are used when a function is called. If `SQL SECURITY` is `INVOKER`, the function body will be evaluated using the privileges of the user calling the function. If `SQL SECURITY` is `DEFINER`, the function body is always evaluated using the privileges of the definer account. `DEFINER` is the default.

This allows you to create functions that grant limited access to certain data. For example, say you have a table that stores some employee information, and that you've granted `SELECT` privileges only on certain columns to the user account `roger`.

```
CREATE TABLE employees (name TINYTEXT, dept TINYTEXT, salary INT);
GRANT SELECT (name, dept) ON employees TO roger;
```

To allow the user the get the maximum salary for a department, define a function and grant the `EXECUTE` privilege:

```
CREATE FUNCTION max_salary (dept TINYTEXT) RETURNS INT RETURN
  (SELECT MAX(salary) FROM employees WHERE employees.dept = dept);
GRANT EXECUTE ON FUNCTION max_salary TO roger;
```

Since `SQL SECURITY` defaults to `DEFINER`, whenever the user `roger` calls this function, the subselect will execute with your privileges. As long as you have privileges to select the salary of each employee, the caller of the function will be able to get the maximum salary for each department without being able to see individual salaries.

Character Sets and Collations

Function return types can be declared to use any valid character set and collation. If used, the `COLLATE` attribute needs to be preceded by a `CHARACTER SET` attribute.

If the character set and collation are not specifically set in the statement, the database defaults at the time of creation will be used. If the database defaults change at a later stage, the stored function character set/collation will not be changed at the same time; the stored function needs to be dropped and recreated to ensure the same character set/collation as the database is used.

Examples

The following example function takes a parameter, performs an operation using an SQL function, and returns the result.

```
CREATE FUNCTION hello (s CHAR(20))
    RETURNS CHAR(50) DETERMINISTIC
    RETURN CONCAT('Hello, ',s,'!');

SELECT hello('world');
+----------------+
| hello('world') |
+----------------+
| Hello, world!  |
+----------------+
```

You can use a compound statement in a function to manipulate data with statements like `INSERT` and `UPDATE`. The following example creates a counter function that uses a temporary table to store the current value. Because the compound statement contains statements terminated with semicolons, you have to first change the statement delimiter with the `DELIMITER` statement to allow the semicolon to be used in the function body. See Delimiters in the mariadb client for more.

```
CREATE TEMPORARY TABLE counter (c INT);
INSERT INTO counter VALUES (0);
DELIMITER //
CREATE FUNCTION counter () RETURNS INT
  BEGIN
    UPDATE counter SET c = c + 1;
    RETURN (SELECT c FROM counter LIMIT 1);
  END //
DELIMITER ;
```

Character set and collation:

```
CREATE FUNCTION hello2 (s CHAR(20))
  RETURNS CHAR(50) CHARACTER SET 'utf8' COLLATE 'utf8_bin' DETERMINISTIC
  RETURN CONCAT('Hello, ',s,'!');
```

See Also

- Identifier Names
- Stored Aggregate Functions
- CREATE FUNCTION (UDF)
- SHOW CREATE FUNCTION
- ALTER FUNCTION
- DROP FUNCTION
- SHOW FUNCTION STATUS
- Stored Routine Privileges
- Information Schema ROUTINES Table

This page is licensed: GPLv2, originally from fill_help_tables.sql

CREATE INDEX

Syntax

```
CREATE [OR REPLACE] [UNIQUE|FULLTEXT|SPATIAL|VECTOR] INDEX
    [IF NOT EXISTS] index_name
    [index_type]
    ON tbl_name (index_col_name,...)
    [WAIT n | NOWAIT]
    [index_option]
    [algorithm_option | lock_option] ...

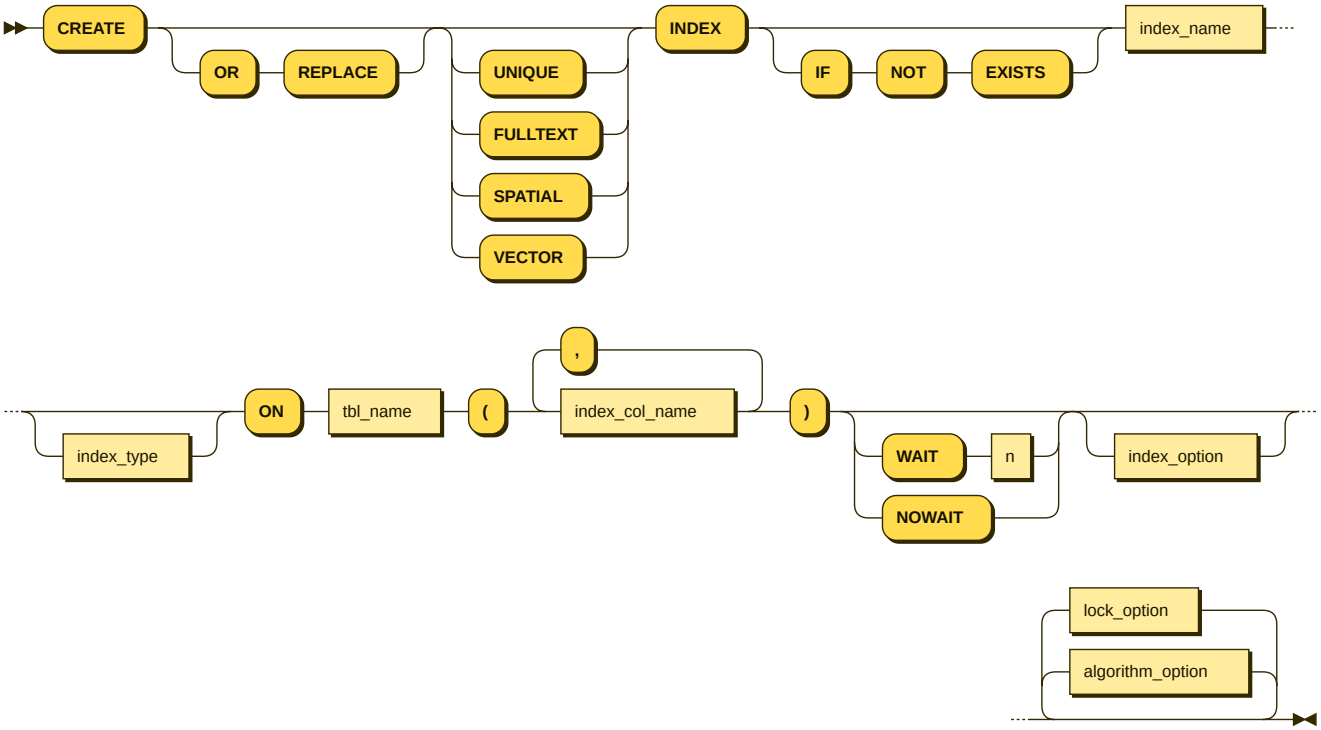
index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH | RTREE}

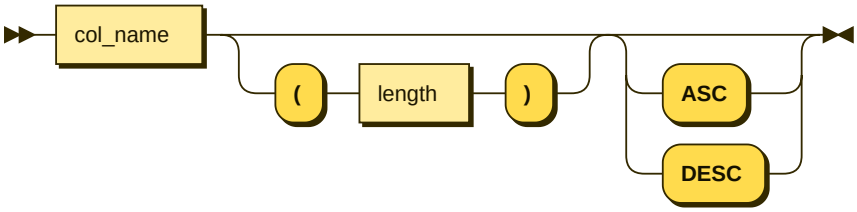
index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
  | CLUSTERING = {YES | NO}
  | IGNORED | NOT IGNORED
  | DISTANCE = {EUCLIDEAN | COSINE}
  | M = number

algorithm_option:
    ALGORITHM [=] {DEFAULT|INPLACE|COPY|NOCOPY|INSTANT}

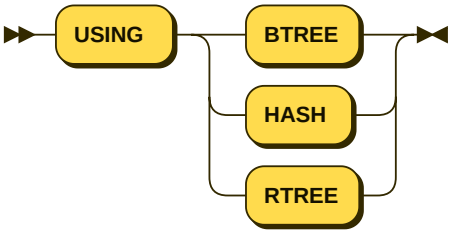
lock_option:
    LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
```



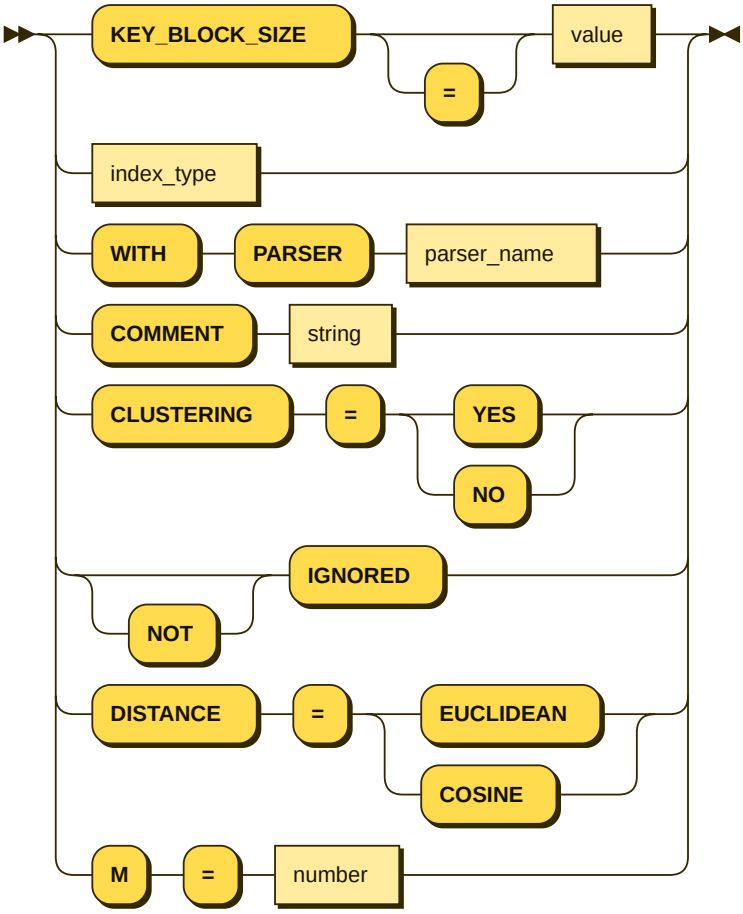
Railroad diagram of CREATE INDEX — equivalent to the BNF above



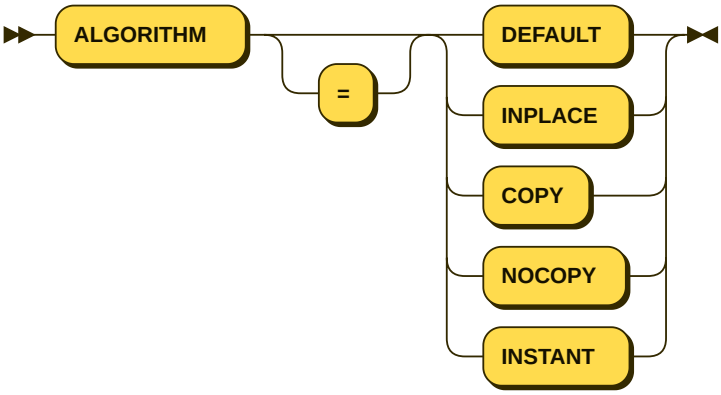
Railroad diagram of index_col_name



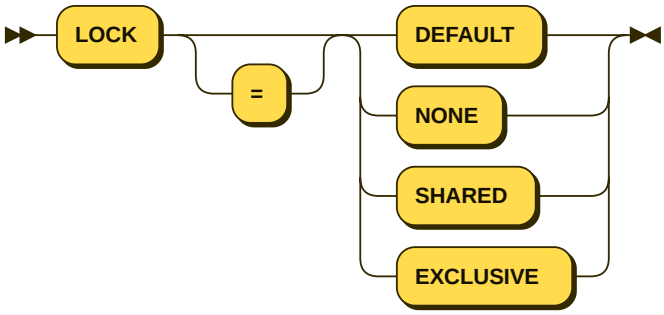
Railroad diagram of index_type



Railroad diagram of index_option



Railroad diagram of algorithm_option



Railroad diagram of lock_option

Description

The `CREATE INDEX` statement is used to add indexes to a table. Indexes can be created at the same as the table, with the CREATE TABLE statement. In some cases, such as for InnoDB primary keys, doing so during creation is preferable, as adding a primary key will involve rebuilding the table.

The statement is mapped to an `ALTER TABLE` statement to create indexes. See ALTER TABLE. `CREATE INDEX` cannot be used to create a PRIMARY KEY; use `ALTER TABLE` instead.

If another connection is using the table, a metadata lock is active, and this statement will wait until the lock is released. This is also true for non-transactional tables.

Another shortcut, DROP INDEX, allows the removal of an index.

For valid identifiers to use as index names, see Identifier Names.

For limits on InnoDB indexes, see InnoDB Limitations.

Note that `KEY_BLOCK_SIZE` is currently ignored in `CREATE INDEX`, although it is included in the output of SHOW CREATE TABLE.

Privileges

Executing the `CREATE INDEX` statement requires the INDEX privilege for the table or the database.

Online DDL

Online DDL is supported with the ALGORITHM and LOCK clauses.

See InnoDB Online DDL Overview for more information on online DDL with InnoDB.

CREATE OR REPLACE INDEX

If the `OR REPLACE` clause is used and if the index already exists, then instead of returning an error, the server will drop the existing index and replace it with the newly defined index.

CREATE INDEX IF NOT EXISTS

If the `IF NOT EXISTS` clause is used, then the index will only be created if an index with the same name does not already exist. If the index already exists, then a warning will be triggered by default.

Index Definitions

See CREATE TABLE: Index Definitions for information about index definitions.

WAIT/NOWAIT

Set the lock wait timeout. See WAIT and NOWAIT.

ALGORITHM

See ALTER TABLE: ALGORITHM for more information.

LOCK

See ALTER TABLE: LOCK for more information.

Progress Reporting

MariaDB provides progress reporting for `CREATE INDEX` statement for clients that support the new progress reporting protocol. For example, if you were using the mariadb client, then the progress report might look like this::

```
CREATE INDEX i ON tab (num);
Stage: 1 of 2 'copy to tmp table'    46% of stage
```

The progress report is also shown in the output of the SHOW PROCESSLIST statement and in the contents of the information_schema.PROCESSLIST table.

See Progress Reporting for more information.

WITHOUT OVERLAPS

CURRENT

The `WITHOUT OVERLAPS` clause allows you to constrain a primary or unique index such that application-time periods cannot overlap. It can be used like this:

```
CREATE UNIQUE INDEX u ON rooms (room_number, p WITHOUT OVERLAPS);
```

< 10.5.3

`WITHOUT OVERLAPS` is not available.

Examples

Creating a unique index:

```
CREATE UNIQUE INDEX HomePhone ON Employees(Home_Phone);
```

OR REPLACE and IF NOT EXISTS:

```
CREATE INDEX xi ON xx5 (x);
Query OK, 0 rows affected (0.03 sec)

CREATE INDEX xi ON xx5 (x);
ERROR 1061 (42000): Duplicate key name 'xi'

CREATE OR REPLACE INDEX xi ON xx5 (x);
Query OK, 0 rows affected (0.03 sec)

CREATE INDEX IF NOT EXISTS xi ON xx5 (x);
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-------+------+-------------------------+
| Level | Code | Message                 |
+-------+------+-------------------------+
| Note  | 1061 | Duplicate key name 'xi' |
+-------+------+-------------------------+
```

See Also

- Identifier Names
- Getting Started with Indexes
- ALTER TABLE
- DROP INDEX
- SHOW INDEX
- SPATIAL INDEX
- Full-text Indexes
- WITHOUT OVERLAPS
- Ignored Indexes
- InnoDB Limitations

This page is licensed: GPLv2, originally from fill_help_tables.sql

CREATE LOGFILE GROUP

> **Note:**
>
> The `CREATE LOGFILE GROUP` statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. See MDEV-19295 for more information.

This page is licensed: CC BY-SA / Gnu FDL

CREATE PACKAGE

> **CURRENT**
>
> The `CREATE PACKAGE` statement can be used in any mode.
>
> **< 11.4**
>
> The `CREATE PACKAGE` statement can be used when Oracle SQL_MODE is set.

In Oracle SQL mode, the PL/SQL dialect is used, while if Oracle mode is not set (the default), SQL/PSM is used.

Syntax (Oracle mode)

```
CREATE
    [ OR REPLACE]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    PACKAGE [ IF NOT EXISTS ]
    [ db_name . ] package_name
    [ package_characteristic ... ]
{ AS | IS }
    [ package_specification_element ... ]
END [ package_name ]

package_characteristic:
    COMMENT 'string'
  | SQL SECURITY { DEFINER | INVOKER }

package_specification_element:
    FUNCTION_SYM package_specification_function ;
  | PROCEDURE_SYM package_specification_procedure ;

package_specification_function:
    func_name [ ( func_param [, func_param]... ) ]
    RETURN type
    [ package_routine_characteristic... ]

package_specification_procedure:
    proc_name [ ( proc_param [, proc_param]... ) ]
    [ package_routine_characteristic... ]

func_param:
    param_name [ IN | OUT | INOUT | IN OUT ] type

proc_param:
    param_name [ IN | OUT | INOUT | IN OUT ] type

type:
    Any valid MariaDB explicit or anchored data type

package_routine_characteristic:
      COMMENT  'string'
    | LANGUAGE SQL
    | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
    | SQL SECURITY { DEFINER | INVOKER }
```

Syntax (non-Oracle mode)

```
CREATE
    [ OR REPLACE]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    PACKAGE [ IF NOT EXISTS ]
    [ db_name . ] package_name
    [ package_characteristic ... ]
    [ package_specification_element ... ]
END

package_characteristic:
    COMMENT 'string'
  | SQL SECURITY { DEFINER | INVOKER }

package_specification_element:
    FUNCTION_SYM package_specification_function ;
  | PROCEDURE_SYM package_specification_procedure ;

package_specification_function:
    func_name [ ( func_param [, func_param]... ) ]
    RETURNS type
    [ package_routine_characteristic... ]

package_specification_procedure:
    proc_name [ ( proc_param [, proc_param]... ) ]
    [ package_routine_characteristic... ]

func_param:
    param_name [ IN | OUT | INOUT | IN OUT ] type

proc_param:
    param_name [ IN | OUT | INOUT | IN OUT ] type

type:
    Any valid MariaDB explicit or anchored data type

package_routine_characteristic:
      COMMENT  'string'
    | LANGUAGE SQL
    | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
    | SQL SECURITY { DEFINER | INVOKER }
```

Description

`CREATE PACKAGE` creates the specification for a stored package (a collection of logically related stored objects). A stored package specification declares public routines (procedures and functions) of the package, but does not implement these routines.

A package whose specification was created by the `CREATE PACKAGE` statement, should later be implemented using the CREATE PACKAGE BODY statement.

Function parameter quantifiers IN | OUT | INOUT | IN OUT

MariaDB starting with 10.8.0

> **CURRENT**
>
> The function parameter quantifiers for `IN`, `OUT`, `INOUT`, and `IN OUT` are supported anywhere.
>
> **< 10.8.0**
>
> The function parameter quantifiers for `IN`, `OUT`, `INOUT`, and `IN OUT` are supported only in procedures.

`OUT`, `INOUT` and its equivalent `IN OUT`, are only valid if called from `SET` and not `SELECT`. These quantifiers are especially useful for creating functions and procedures with more than one return value. This allows functions and procedures to be more complex and nested.

Examples

```
SET sql_mode=ORACLE; # unnecessary from MariaDB 11.4
DELIMITER $$
CREATE OR REPLACE PACKAGE employee_tools AS
  FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2);
  PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2));
  PROCEDURE raiseSalaryStd(eid INT);
  PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2));
END;
$$
DELIMITER ;
```

See Also

- CREATE PACKAGE BODY
- SHOW CREATE PACKAGE
- DROP PACKAGE
- Oracle SQL_MODE

This page is licensed: CC BY-SA / Gnu FDL

CREATE PACKAGE BODY

> **CURRENT**
>
> The `CREATE PACKAGE BODY` statement can be used in any mode.
>
> **< 11.4**
>
> The `CREATE PACKAGE BODY` statement can be used in Oracle SQL_MODE.

In Oracle mode, the PL/SQL dialect is used, while in non-Oracle mode, SQL/PSM is used.

Syntax (Oracle mode)

```
CREATE [ OR REPLACE ]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    PACKAGE BODY
    [ IF NOT EXISTS ]
    [ db_name . ] package_name
    [ package_characteristic... ]
{ AS | IS }
    package_implementation_declare_section
    package_implementation_executable_section
END [ package_name]

package_implementation_declare_section:
    package_implementation_item_declaration
      [ package_implementation_item_declaration... ]
      [ package_implementation_routine_definition... ]
  | package_implementation_routine_definition
      [ package_implementation_routine_definition...]

package_implementation_item_declaration:
    variable_declaration ;

variable_declaration:
    variable_name[,...] type [:= expr ]

package_implementation_routine_definition:
    FUNCTION package_specification_function
       [ package_implementation_function_body ] ;
  | PROCEDURE package_specification_procedure
       [ package_implementation_procedure_body ] ;

package_implementation_function_body:
    { AS | IS } package_routine_body [func_name]

package_implementation_procedure_body:
    { AS | IS } package_routine_body [proc_name]

package_routine_body:
    [ package_routine_declarations ]
    BEGIN
      statements [ EXCEPTION exception_handlers ]
    END

package_routine_declarations:
    package_routine_declaration ';' [package_routine_declaration ';']...

package_routine_declaration:
          variable_declaration
        | condition_name CONDITION FOR condition_value
        | user_exception_name EXCEPTION
        | CURSOR_SYM cursor_name
          [ ( cursor_formal_parameters ) ]
          IS select_statement
        ;

package_implementation_executable_section:
          END
        | BEGIN
            statement ; [statement ; ]...
          [EXCEPTION exception_handlers]
          END

exception_handlers:
           exception_handler [exception_handler...]

exception_handler:
          WHEN_SYM condition_value [, condition_value]...
            THEN_SYM statement ; [statement ;]...
```

```
condition_value:
          condition_name
        | user_exception_name
        | SQLWARNING
        | SQLEXCEPTION
        | NOT FOUND
        | OTHERS_SYM
        | SQLSTATE [VALUE] sqlstate_value
        | mariadb_error_code
```

Description

The `CREATE PACKAGE BODY` statement creates the package body for a stored package. The package specification must be previously created using the CREATE PACKAGE statement.

A package body provides implementations of the package public routines and can optionally have:

- package-wide private variables
- package private routines
- forward declarations for private routines
- an executable initialization section

> Starting with MariaDB 13.0.1, package bodies can define `RECORD` and `REF CURSOR` types that can be used as:
>
> - Parameters (IN, OUT, INOUT) for procedures and functions defined in the package
> - RETURN clause data types for package functions
>
> For detailed syntax and examples, see DECLARE TYPE.

Examples

```
SET sql_mode=ORACLE;
DELIMITER $$
CREATE OR REPLACE PACKAGE employee_tools AS
  FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2);
  PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2));
  PROCEDURE raiseSalaryStd(eid INT);
  PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2));
END;
$$
CREATE PACKAGE BODY employee_tools AS
  -- package body variables
  stdRaiseAmount DECIMAL(10,2):=500;

  -- private routines
  PROCEDURE log (eid INT, ecmnt TEXT) AS
  BEGIN
    INSERT INTO employee_log (id, cmnt) VALUES (eid, ecmnt);
  END;

  -- public routines
  PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2)) AS
    eid INT;
  BEGIN
    INSERT INTO employee (name, salary) VALUES (ename, esalary);
    eid:= last_insert_id();
    log(eid, 'hire ' || ename);
  END;

  FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2) AS
    nSalary DECIMAL(10,2);
  BEGIN
    SELECT salary INTO nSalary FROM employee WHERE id=eid;
    log(eid, 'getSalary id=' || eid || ' salary=' || nSalary);
    RETURN nSalary;
  END;

  PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2)) AS
  BEGIN
    UPDATE employee SET salary=salary+amount WHERE id=eid;
    log(eid, 'raiseSalary id=' || eid || ' amount=' || amount);
  END;

  PROCEDURE raiseSalaryStd(eid INT) AS
  BEGIN
    raiseSalary(eid, stdRaiseAmount);
    log(eid, 'raiseSalaryStd id=' || eid);
  END;

BEGIN
  -- This code is executed when the current session
  -- accesses any of the package routines for the first time
  log(0, 'Session ' || connection_id() || ' ' || current_user || ' started');
END;
$$

DELIMITER ;
```

See Also

- CREATE PACKAGE
- SHOW CREATE PACKAGE BODY
- DROP PACKAGE BODY
- Oracle SQL_MODE

This page is licensed: CC BY-SA / Gnu FDL

CREATE SERVER

Syntax

```
CREATE [OR REPLACE] SERVER [IF NOT EXISTS] server_name
    FOREIGN DATA WRAPPER wrapper_name
    OPTIONS (option [, option] ...)

option: <= MariaDB 11.6
  { HOST character-literal
  | DATABASE character-literal
  | USER character-literal
  | PASSWORD character-literal
  | SOCKET character-literal
  | OWNER character-literal
  | PORT numeric-literal }

option: >= MariaDB Enterprise Server 11.4 / Community Server 11.7
  { HOST character-literal
  | DATABASE character-literal
  | USER character-literal
  | PASSWORD character-literal
  | SOCKET character-literal
  | OWNER character-literal
  | PORT numeric-literal
  | PORT quoted-numerical-literal
  | identifier character-literal}
```

Description

> **CURRENT**
>
> This statement creates the definition of a server for use with the Spider, Connect, FEDERATED, or FederatedX storage engine. The `CREATE SERVER` statement creates a new row in the servers table within the mysql database. This statement requires the FEDERATED ADMIN privilege.
>
> **< 10.5.2**
>
> This statement creates the definition of a server for use with the Spider, Connect, FEDERATED, or FederatedX storage engine. The `CREATE SERVER` statement creates a new row in the servers table within the mysql database. This statement requires the SUPER privilege.

The server_name should be a unique reference to the server. Server definitions are global within the scope of the server, it is not possible to qualify the server definition to a specific database. server_name has a maximum length of 64 characters (names longer than 64 characters are silently truncated), and is case-insensitive. You may specify the name as a quoted string.

The wrapper_name may be quoted with single quotes. Supported values are:

- `mysql`
- `mariadb`

For each option you must specify either a character literal or numeric literal. Character literals are UTF-8, support a maximum length of 64 characters and default to a blank (empty) string. String literals are silently truncated to 64 characters. Numeric literals must be a number between `0` and `9999`, default value is `0`.

Note: The `OWNER` option is currently not applied, and has no effect on the ownership or operation of the server connection that is created.

The `CREATE SERVER` statement creates an entry in the mysql.servers table that can later be used with the `CREATE TABLE` statement when creating a Spider, Connect, FederatedX or FEDERATED table. The options that you specify will be used to populate the columns in the mysql.servers table. The table columns are `Server_name`, `Host`, `Db`, `Username`, `Password`, `Port`, and `Socket`.

Note: When used with the Spider storage engine, connection information provided via `CREATE SERVER` may be overridden by table-level or engine-defined parameters. For information on connection configuration and precedence rules, see the Spider Storage Engine Core Concepts page.

DROP SERVER removes a previously created server definition.

`CREATE SERVER` is not written to the binary log, irrespective of the binary log format being used and therefore will not replicate.

> **CURRENT**
>
> Galera replicates the `CREATE SERVER`, ALTER SERVER and DROP SERVER statements.
>
> **< 10.1.13**
>
> Galera does not replicate the `CREATE SERVER`, ALTER SERVER and DROP SERVER statements.

For valid identifiers to use as server names, see Identifier Names.

> **CURRENT**
>
> The SHOW CREATE SERVER statement can be used to show the `CREATE SERVER` statement that created a given server definition.
>
> **< ENTERPRISE SERVER 11.4 / COMMUNITY SERVER 11.7**
>
> The SHOW CREATE SERVER statement cannot be used to show the `CREATE SERVER` statement that created a given server definition.

OR REPLACE

If the optional `OR REPLACE` clause is used, it acts as a shortcut for:

```
DROP SERVER IF EXISTS name;
CREATE SERVER server_name ...;
```

IF NOT EXISTS

If the `IF NOT EXISTS` clause is used, MariaDB will return a warning instead of an error if the server already exists. Cannot be used together with `OR REPLACE`.

Examples

```
CREATE SERVER s
FOREIGN DATA WRAPPER mariadb
OPTIONS (USER 'Remote', HOST '192.168.1.106', DATABASE 'test');
```

`OR REPLACE` and `IF NOT EXISTS`:

```
CREATE SERVER s
FOREIGN DATA WRAPPER mariadb
OPTIONS (USER 'Remote', HOST '192.168.1.106', DATABASE 'test');
ERROR 1476 (HY000): The foreign server, s, you are trying to create already exists

CREATE OR REPLACE SERVER s
FOREIGN DATA WRAPPER mariadb
OPTIONS (USER 'Remote', HOST '192.168.1.106', DATABASE 'test');
Query OK, 0 rows affected (0.00 sec)

CREATE SERVER IF NOT EXISTS s
FOREIGN DATA WRAPPER mariadb
OPTIONS (USER 'Remote', HOST '192.168.1.106', DATABASE 'test');
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------------------------------+
| Level | Code | Message                                                         |
+-------+------+-----------------------------------------------------------------+
| Note  | 1476 | The foreign server, s, you are trying to create already exists |
+-------+------+-----------------------------------------------------------------+
```

See Also

- Identifier Names
- ALTER SERVER
- DROP SERVER
- Spider Storage Engine
- Connect Storage Engine
- mysql.servers table
- SHOW CREATE SERVER

This page is licensed: GPLv2, originally from fill_help_tables.sql

CREATE TABLESPACE

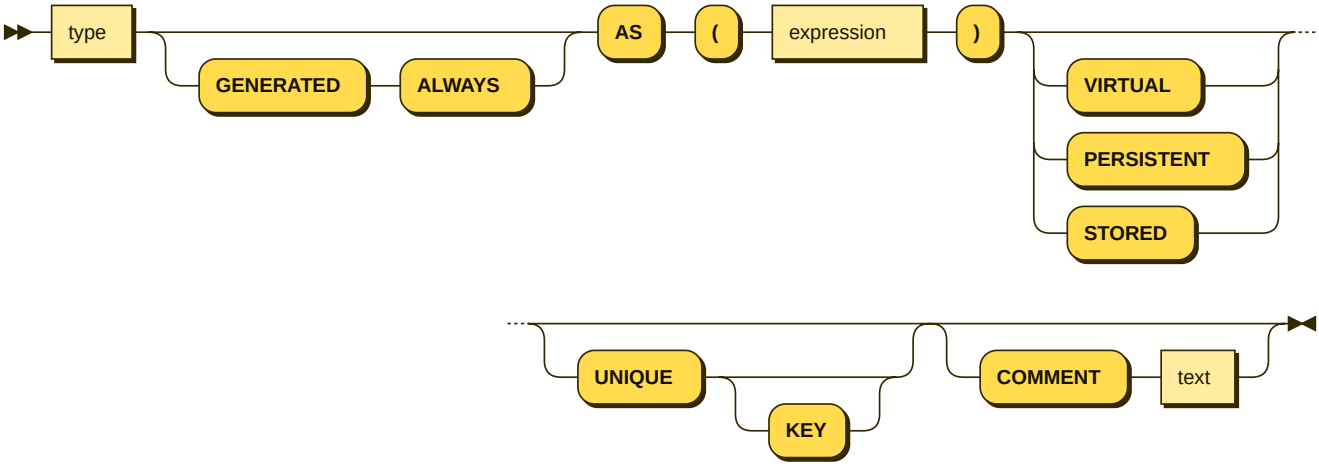
> **Note:**
>
> The `CREATE TABLESPACE` statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. In MySQL 5.7 and later, the statement is also supported for InnoDB. However, MariaDB has chosen not to include that specific feature. See MDEV-19294 for more information.

This page is licensed: GPLv2, originally from fill_help_tables.sql

Generated Columns

Syntax

```
<type> [GENERATED ALWAYS] AS (<expression>)
[VIRTUAL | PERSISTENT | STORED] [UNIQUE [KEY]] [COMMENT <text>]
```



Railroad diagram of a generated-column definition — equivalent to the BNF above

> **CURRENT**
>
> MariaDB's generated columns syntax is designed to be similar to the syntax for Microsoft SQL Server's computed columns and Oracle Database's virtual columns. The syntax is also compatible with the syntax for MySQL's generated columns.
>
> **< 10.2**
>
> MariaDB's generated columns syntax is designed to be similar to the syntax for Microsoft SQL Server's computed columns and Oracle Database's virtual columns. The syntax is **not** compatible with the syntax for MySQL's generated columns.

Description

A generated column is a column in a table that cannot explicitly be set to a specific value in a DML query. Instead, its value is automatically generated based on an expression. This expression might generate the value based on the values of other columns in the table, or it might generate the value by calling built-in functions or user-defined functions (UDFs).

There are two types of generated columns:

- `PERSISTENT` (a.k.a. `STORED`): This type's value is actually stored in the table.
- `VIRTUAL`: This type's value is not stored at all. Instead, the value is generated dynamically when the table is queried. This type is the default.

Generated columns are also sometimes called computed columns or virtual columns.

Supported Features

Storage Engine Support

- Generated columns can only be used with storage engines which support them. If you try to use a storage engine that does not support them, then you will see an error similar to the following:

```
ERROR 1910 (HY000): <storage engine> storage engine does not support generated columns
```

- InnoDB, Aria, MyISAM and CONNECT support generated columns.
- A column in a MERGE table can be built on a `PERSISTENT` generated column.
  - However, a column in a MERGE table can not be defined as a `VIRTUAL` and `PERSISTENT` generated column.

Data Type Support

All data types are supported when defining generated columns.

Using the ZEROFILL column option is supported when defining generated columns.

> **TAB 1**
>
> Using the AUTO_INCREMENT column option is not supported when defining generated columns.
>
> **< 10.2.25**
>
> Using the AUTO_INCREMENT column option is supported when defining generated columns.
>
> > **Warning:**
> >
> > It does not work correctly, though. See MDEV-11117.

Index Support

Using a generated column as a table's primary key is not supported. See MDEV-5590 for more information. If you try to use one as a primary key, then you will see an error similar to the following:

```
ERROR 1903 (HY000): Primary key cannot be defined upon a computed column
```

Using `PERSISTENT` generated columns as part of a foreign key is supported.

Referencing `PERSISTENT` generated columns as part of a foreign key is also supported.

However, using the `ON UPDATE CASCADE`, `ON UPDATE SET NULL`, or `ON DELETE SET NULL` clauses is not supported. If you try to use an unsupported clause, then you will see an error similar to the following:

```
ERROR 1905 (HY000): Cannot define foreign key with ON UPDATE SET NULL clause on a computed column
```

Defining indexes on both `VIRTUAL` and `PERSISTENT` generated columns is supported.

If an index is defined on a generated column, then the optimizer considers using it in the same way as indexes based on "real" columns.

> **CURRENT**
>
> The optimizer can recognize use of indexed virtual column expressions in the `WHERE` clause and use them to construct range and `ref(const)` accesses. See Virtual Column Support in the Optimizer.
>
> **< 11.8**
>
> The optimizer **cannot** recognize use of indexed virtual column expressions in the `WHERE` clause and use them to construct range and `ref(const)` accesses. See Virtual Column Support in the Optimizer.

Statement Support

Generated columns are used in DML queries just as if they were "real" columns.

- However, `VIRTUAL` and `PERSISTENT` generated columns differ in how their data is stored.
  - Values for `PERSISTENT` generated columns are generated whenever a DML queries inserts or updates the row with the special `DEFAULT` value. This generates the columns value, and it is stored in the table like the other "real" columns. This value can be read by other DML queries just like the other "real" columns.
  - Values for `VIRTUAL` generated columns are not stored in the table. Instead, the value is generated dynamically whenever the column is queried. If other columns in a row are queried, but the `VIRTUAL`

generated column is not one of the queried columns, then the column's value is not generated.

The SELECT statement supports generated columns.

Generated columns can be referenced in the INSERT, UPDATE, and DELETE statements.

- However, `VIRTUAL` or `PERSISTENT` generated columns cannot be explicitly set to any other values than `NULL` or DEFAULT. If a generated column is explicitly set to any other value, then the outcome depends on whether strict mode is enabled in sql_mode. If it is not enabled, then a warning will be raised and the default generated value will be used instead. If it is enabled, then an error will be raised instead.

The CREATE TABLE statement has limited support for generated columns.

- It supports defining generated columns in a new table.
- It supports using generated columns to partition tables.
- It does not support using the versioning clauses with generated columns.

The ALTER TABLE statement has limited support for generated columns.

- It supports the `MODIFY` and `CHANGE` clauses for `PERSISTENT` generated columns.
- It does not support the `MODIFY` clause for `VIRTUAL` generated columns if ALGORITHM is not set to `COPY`. See MDEV-15476 for more information.
- It does not support the `CHANGE` clause for `VIRTUAL` generated columns if ALGORITHM is not set to `COPY`. See MDEV-17035 for more information.
- It does not support altering a table if ALGORITHM is not set to `COPY` if the table has a `VIRTUAL` generated column that is indexed. See MDEV-14046 for more information.
- It does not support adding a `VIRTUAL` generated column with the `ADD` clause if the same statement is also adding other columns if ALGORITHM is not set to `COPY`. See MDEV-17468 for more information.
- It also does not support altering an existing column into a `VIRTUAL` generated column.
- It supports using generated columns to partition tables.
- It does not support using the versioning clauses with generated columns.

The SHOW CREATE TABLE statement supports generated columns.

The DESCRIBE statement can be used to check whether a table has generated columns.

- You can tell which columns are generated by looking for the ones where the `Extra` column is set to either `VIRTUAL` or `PERSISTENT`. For example:

```
DESCRIBE table1;
+-------+-------------+------+-----+---------+------------+
| Field | Type        | Null | Key | Default | Extra      |
+-------+-------------+------+-----+---------+------------+
| a     | int(11)     | NO   |     | NULL    |            |
| b     | varchar(32) | YES  |     | NULL    |            |
| c     | int(11)     | YES  |     | NULL    | VIRTUAL    |
| d     | varchar(5)  | YES  |     | NULL    | PERSISTENT |
+-------+-------------+------+-----+---------+------------+
```

Generated columns can be properly referenced in the `NEW` and `OLD` rows in triggers.

Stored procedures support generated columns.

The HANDLER statement supports generated columns.

Expression Support

Most legal, deterministic expressions which can be calculated are supported in expressions for generated columns.

Most built-in functions are supported in expressions for generated columns.

- However, some built-in functions can't be supported for technical reasons. For example, If you try to use an unsupported function in an expression, an error is generated similar to the following:

```
ERROR 1901 (HY000): Function or expression 'dayname()' cannot be used in the GENERATED ALWAYS AS clause
of `v`
```

Subqueries are not supported in expressions for generated columns because the underlying data can change.

Using anything that depends on data outside the row is not supported in expressions for generated columns.

Stored functions are not supported in expressions for generated columns. See MDEV-17587 for more information.

Non-deterministic built-in functions are supported in expressions for not indexed `VIRTUAL` generated columns.

Non-deterministic built-in functions are not supported in expressions for `PERSISTENT` or indexed `VIRTUAL` generated columns.

User-defined functions (UDFs) are supported in expressions for generated columns.

- However, MariaDB can't check whether a UDF is deterministic, so it is up to the user to be sure that they do not use non-deterministic UDFs with `VIRTUAL` generated columns.

Defining a generated column based on other generated columns defined before it in the table definition is supported. For example:

```
CREATE TABLE t1 (a INT AS (1), b INT AS (a));
```

However, defining a generated column based on other generated columns defined after in the table definition is not supported in expressions for generation columns because generated columns are calculated in the order they are defined.

Using an expression that exceeds 255 characters in length is supported in expressions for generated columns. The new limit for the entire table definition, including all expressions for generated columns, is 65,535 bytes.

Using constant expressions is supported in expressions for generated columns. For example:

```
CREATE TABLE t1 (a int as (1));
```

Making Stored Values Consistent

When a generated column is `PERSISTENT` or indexed, the value of the expression needs to be consistent regardless of the SQL Mode flags in the current session. If it is not, then the table will be seen as corrupted when the value that should actually be returned by the computed expression and the value that was previously stored and/or indexed using a different sql_mode setting disagree.

There are currently two affected classes of inconsistencies: character padding and unsigned subtraction:

- For a `VARCHAR` or `TEXT` generated column the length of the value returned can vary depending on the PAD_CHAR_TO_FULL_LENGTH sql_mode flag. To make the value consistent, create the generated column using an RTRIM() or RPAD() function. Alternately, create the generated column as a `CHAR` column so that its data is always fully padded.
- If a `SIGNED` generated column is based on the subtraction of an `UNSIGNED` value, the resulting value can vary depending on how large the value is and the NO_UNSIGNED_SUBTRACTION sql_mode flag. To make the value consistent, use CAST() to ensure that each `UNSIGNED` operand is `SIGNED` before the subtraction.

> **CURRENT**
>
> A fatal error is generated when trying to create a generated column whose value can change depending on the SQL Mode when its data is `PERSISTENT` or indexed. For an existing generated column that has a potentially

inconsistent value, a warning about a bad expression is generated the first time it is used (if warnings are enabled).

< 10.5

For an existing generated column that has a potentially inconsistent value, a warning about a bad expression is generated the first time it is used (if warnings are enabled).

CURRENT

A potentially inconsistent generated column outputs a warning when created or first used (without restricting the creation).

< 10.4.8 / 10.3.18 / 10.2.27

A potentially inconsistent generated column does not output a warning when created or first used.

CURRENT

Here is an example of two tables that are warned about:

```
CREATE TABLE bad_pad (
  txt CHAR(5),
  -- CHAR -> VARCHAR or CHAR -> TEXT can't be persistent or indexed:
  vtxt VARCHAR(5) AS (txt) PERSISTENT
);

CREATE TABLE bad_sub (
  num1 BIGINT UNSIGNED,
  num2 BIGINT UNSIGNED,
  -- The resulting value can vary for some large values
  vnum BIGINT AS (num1 - num2) VIRTUAL,
  KEY(vnum)
);
```

The warnings look like this:

```
Warning (Code 1901): Function or expression '`txt`' cannot be used in the GENERATED ALWAYS AS clause of
`vtxt`
Warning (Code 1105): Expression depends on the @@sql_mode value PAD_CHAR_TO_FULL_LENGTH

Warning (Code 1901): Function or expression '`num1` - `num2`' cannot be used in the GENERATED ALWAYS AS
clause of `vnum`
Warning (Code 1105): Expression depends on the @@sql_mode value NO_UNSIGNED_SUBTRACTION
```

To work around the issue, force the padding or type to make the generated column's expression return a consistent value. For example:

```
CREATE TABLE good_pad (
  txt CHAR(5),
  -- Using RTRIM() or RPAD() makes the value consistent:
  vtxt VARCHAR(5) AS (RTRIM(txt)) PERSISTENT,
  -- When not persistent or indexed, it is OK for the value to vary by mode:
  vtxt2 VARCHAR(5) AS (txt) VIRTUAL,
  -- CHAR -> CHAR is always OK:
  txt2 CHAR(5) AS (txt) PERSISTENT
);

CREATE TABLE good_sub (
  num1 BIGINT UNSIGNED,
  num2 BIGINT UNSIGNED,
  -- The indexed value will always be consistent in this expression:
  vnum BIGINT AS (CAST(num1 AS SIGNED) - CAST(num2 AS SIGNED)) VIRTUAL,
  KEY(vnum)
);
```

< 10.6

Here is an example of two tables whose creation is rejected:

```
CREATE TABLE bad_pad (
  txt CHAR(5),
  -- CHAR -> VARCHAR or CHAR -> TEXT can't be persistent or indexed:
  vtxt VARCHAR(5) AS (txt) PERSISTENT
);

CREATE TABLE bad_sub (
  num1 BIGINT UNSIGNED,
  num2 BIGINT UNSIGNED,
  -- The resulting value can vary for some large values
  vnum BIGINT AS (num1 - num2) VIRTUAL,
  KEY(vnum)
);
```

MySQL Compatibility Support

- The `STORED` keyword is supported as an alias for the `PERSISTENT` keyword.
- Tables created with MySQL 5.7 or later that contain MySQL's generated columns can be imported into MariaDB without a dump and restore.

Implementation Differences

Generated columns are subject to various constraints in other DBMSs that are not present in MariaDB's implementation. Generated columns may also be called computed columns or virtual columns in different implementations. The various details for a specific implementation can be found in the documentation for each specific DBMS.

Implementation Differences Compared to Microsoft SQL Server

MariaDB's generated columns implementation does not enforce the following restrictions that are present in Microsoft SQL Server's computed columns implementation:

- MariaDB allows server variables in generated column expressions, including those that change dynamically, such as warning_count.
- MariaDB allows the CONVERT_TZ() function to be called with a named time zone as an argument, even though time zone names and time offsets are configurable.
- MariaDB allows the CAST() function to be used with non-unicode character sets, even though character sets are configurable and differ between binaries/versions.

- MariaDB allows FLOAT expressions to be used in generated columns. Microsoft SQL Server considers these expressions to be “imprecise” due to potential cross-platform differences in floating-point implementations and precision.
- Microsoft SQL Server requires the ARITHABORT mode to be set, so that division by zero returns an error, and not a NULL.
- Microsoft SQL Server requires `QUOTED_IDENTIFIER` to be set in sql_mode. In MariaDB, if data is inserted without `ANSI_QUOTES` set in sql_mode, then it will be processed and stored differently in a generated column that contains quoted identifiers.

Microsoft SQL Server enforces the above restrictions by doing one of the following things:

- Refusing to create computed columns.
- Refusing to allow updates to a table containing them.
- Refusing to use an index over such a column if it can not be guaranteed that the expression is fully deterministic.

In MariaDB, as long as the sql_mode, language, and other settings that were in effect during the CREATE TABLE remain unchanged, the generated column expression will always be evaluated the same. If any of these things change, then please be aware that the generated column expression might not be evaluated the same way as it previously was.

If you try to update a virtual column, you will get an error if the default strict mode is enabled in sql_mode, or a warning otherwise.

Development History

Generated columns was originally developed by Andrey Zhakov. It was then modified by Sanja Byelkin and Igor Babaev at Monty Program for inclusion in MariaDB. Monty did the work on MariaDB 10.2 to lift some of the limitations.

Examples

Here is an example table that uses both `VIRTUAL` and `PERSISTENT` virtual columns:

```
USE TEST;

CREATE TABLE table1 (
     a INT NOT NULL,
     b VARCHAR(32),
     c INT AS (a MOD 10) VIRTUAL,
     d VARCHAR(5) AS (LEFT(b,5)) PERSISTENT);
```

If you describe the table, you can easily see which columns are virtual by looking in the “Extra” column:

```
DESCRIBE table1;
+-------+-------------+------+-----+---------+------------+
| Field | Type        | Null | Key | Default | Extra      |
+-------+-------------+------+-----+---------+------------+
| a     | int(11)     | NO   |     | NULL    |            |
| b     | varchar(32) | YES  |     | NULL    |            |
| c     | int(11)     | YES  |     | NULL    | VIRTUAL    |
| d     | varchar(5)  | YES  |     | NULL    | PERSISTENT |
+-------+-------------+------+-----+---------+------------+
```

To find out what function(s) generate the value of the virtual column you can use `SHOW CREATE TABLE`:

```
SHOW CREATE TABLE table1;

| table1 | CREATE TABLE `table1` (
  `a` int(11) NOT NULL,
  `b` varchar(32) DEFAULT NULL,
  `c` int(11) AS (a mod 10) VIRTUAL,
  `d` varchar(5) AS (left(b,5)) PERSISTENT
) ENGINE=MyISAM DEFAULT CHARSET=latin1 |
```

If you try to insert non-default values into a virtual column, you will receive a warning and what you tried to insert will be ignored and the derived value inserted instead:

```
WARNINGS;
Show warnings enabled.

INSERT INTO table1 VALUES (1, 'some text',default,default);
Query OK, 1 row affected (0.00 sec)

INSERT INTO table1 VALUES (2, 'more text',5,default);
Query OK, 1 row affected, 1 warning (0.00 sec)

Warning (Code 1645): The value specified for computed column 'c' in table 'table1' has been ignored.

INSERT INTO table1 VALUES (123, 'even more text',default,'something');
Query OK, 1 row affected, 2 warnings (0.00 sec)

Warning (Code 1645): The value specified for computed column 'd' in table 'table1' has been ignored.
Warning (Code 1265): Data truncated for column 'd' at row 1

SELECT * FROM table1;
+-----+----------------+------+-------+
| a   | b              | c    | d     |
+-----+----------------+------+-------+
|   1 | some text      |    1 | some  |
|   2 | more text      |    2 | more  |
| 123 | even more text |    3 | even  |
+-----+----------------+------+-------+
3 rows in set (0.00 sec)
```

If the `ZEROFILL` clause is specified, it should be placed directly after the type definition, before the `AS (<expression>)`:

```
CREATE TABLE table2 (a INT, b INT ZEROFILL AS (a*2) VIRTUAL);
INSERT INTO table2 (a) VALUES (1);

SELECT * FROM table2;
+------+------------+
| a    | b          |
+------+------------+
|    1 | 0000000002 |
+------+------------+
1 row in set (0.00 sec)
```

You can also use virtual columns to implement a “poor man’s partial index”. See example at the end of Unique Index.

See Also

- Generated Columns in `mariadb-dump`, for how generated columns appear in dump output.
- Putting Virtual Columns to good use on the mariadb.com blog.

This page is licensed: CC BY-SA / Gnu FDL

Invisible Columns

Invisible columns (sometimes also called hidden columns) are hidden in certain contexts.

Columns can be given an `INVISIBLE` attribute in a CREATE TABLE or ALTER TABLE statement. These columns will then not be listed in the results of a SELECT * statement, nor do they need to be assigned a value in an INSERT statement, unless INSERT explicitly mentions them by name.

Since `SELECT *` does not return the invisible columns, new tables or views created in this manner will have no trace of the invisible columns. If specifically referenced in the SELECT statement, the columns will be brought into the view/new table, but the INVISIBLE attribute will not.

Invisible columns can be declared as `NOT NULL`, but then require a `DEFAULT` value.

> **Note:**
>
> It is not possible for all columns in a table to be invisible.

Examples

```
CREATE TABLE t (x INT INVISIBLE);
ERROR 1113 (42000): A table must have at least 1 column

CREATE TABLE t (x INT, y INT INVISIBLE, z INT INVISIBLE NOT NULL);
ERROR 4106 (HY000): Invisible column `z` must have a default value

CREATE TABLE t (x INT, y INT INVISIBLE, z INT INVISIBLE NOT NULL DEFAULT 4);

INSERT INTO t VALUES (1),(2);

INSERT INTO t (x,y) VALUES (3,33);

SELECT * FROM t;
+------+
| x    |
+------+
|    1 |
|    2 |
|    3 |
+------+

SELECT x,y,z FROM t;
+------+------+---+
| x    | y    | z |
+------+------+---+
|    1 | NULL | 4 |
|    2 | NULL | 4 |
|    3 |   33 | 4 |
+------+------+---+

DESC t;
+-------+---------+------+-----+---------+-----------+
| Field | Type    | Null | Key | Default | Extra     |
+-------+---------+------+-----+---------+-----------+
| x     | int(11) | YES  |     | NULL    |           |
| y     | int(11) | YES  |     | NULL    | INVISIBLE |
| z     | int(11) | NO   |     | 4       | INVISIBLE |
+-------+---------+------+-----+---------+-----------+

ALTER TABLE t MODIFY x INT INVISIBLE, MODIFY y INT, MODIFY z INT NOT NULL DEFAULT 4;

DESC t;
+-------+---------+------+-----+---------+-----------+
| Field | Type    | Null | Key | Default | Extra     |
+-------+---------+------+-----+---------+-----------+
| x     | int(11) | YES  |     | NULL    | INVISIBLE |
| y     | int(11) | YES  |     | NULL    |           |
| z     | int(11) | NO   |     | 4       |           |
+-------+---------+------+-----+---------+-----------+
```

Creating a view from a table with hidden columns:

```
CREATE VIEW v1 AS SELECT * FROM t;

DESC v1;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| y     | int(11) | YES  |     | NULL    |       |
| z     | int(11) | NO   |     | 4       |       |
+-------+---------+------+-----+---------+-------+

CREATE VIEW v2 AS SELECT x,y,z FROM t;

DESC v2;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| x     | int(11) | YES  |     | NULL    |       |
| y     | int(11) | YES  |     | NULL    |       |
| z     | int(11) | NO   |     | 4       |       |
+-------+---------+------+-----+---------+-------+
```

Adding a Surrogate Primary Key:

```
CREATE TABLE t1 (x BIGINT UNSIGNED NOT NULL, y VARCHAR(16), z TEXT);

INSERT INTO t1 VALUES (123, 'qq11', 'ipsum');

INSERT INTO t1 VALUES (123, 'qq22', 'lorem');

ALTER TABLE t1 ADD pkid SERIAL PRIMARY KEY invisible FIRST;

INSERT INTO t1 VALUES (123, 'qq33', 'amet');

SELECT * FROM t1;
+-----+------+-------+
| x   | y    | z     |
+-----+------+-------+
| 123 | qq11 | ipsum |
| 123 | qq22 | lorem |
| 123 | qq33 | amet  |
+-----+------+-------+

SELECT pkid, z FROM t1;
+------+-------+
| pkid | z     |
+------+-------+
|    1 | ipsum |
|    2 | lorem |
|    3 | amet  |
+------+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

Silent Column Changes

When a CREATE TABLE or ALTER TABLE command is issued, MariaDB silently changes a column specification in the following cases:

- PRIMARY KEY columns are always NOT NULL.
- Any trailing spaces from SET and ENUM values are discarded.
- TIMESTAMP columns are always NOT NULL, and display sizes are discarded.
- A row-size limit of 65535 bytes applies.

- If strict SQL mode is not enabled (by default, it is), a VARCHAR column longer than 65535 become TEXT, and a VARBINARY columns longer than 65535 becomes a BLOB. If strict mode is enabled the silent changes will not be made, and an error will occur.
- If a USING clause specifies an index that's not permitted by the storage engine, the engine will instead use another available index type that can be applied without affecting results.
- If the CHARACTER SET binary attribute is specified, the column is created as the matching binary data type. A TEXT becomes a BLOB, CHAR a BINARY and VARCHAR a VARBINARY. ENUMs and SETs are created as defined.

To ease imports from other RDBMSs, MariaDB also silently maps the following data types:

Other Vendor Type	MariaDB Type
BOOL	TINYINT
BOOLEAN	TINYINT
CHARACTER VARYING(M)	VARCHAR(M)
FIXED	DECIMAL
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT
LONG VARBINARY	MEDIUMBLOB
LONG VARCHAR	MEDIUMTEXT
LONG	MEDIUMTEXT
MIDDLEINT	MEDIUMINT
NUMERIC	DECIMAL

Currently, all MySQL types are supported in MariaDB.

Example

Silent changes in action:

```
CREATE TABLE SilenceIsGolden
   (
    f1 TEXT CHARACTER SET BINARY,
    f2 VARCHAR(15) CHARACTER SET BINARY,
    f3 CHAR CHARACTER SET BINARY,
    f4 ENUM('x','y','z') CHARACTER SET BINARY,
    f5 VARCHAR (65536),
    f6 VARBINARY (65536),
    f7 INT1
   );
Query OK, 0 rows affected, 2 warnings (0.31 sec)

SHOW WARNINGS;
+-------+------+-----------------------------------------------+
| Level | Code | Message                                       |
+-------+------+-----------------------------------------------+
| Note  | 1246 | Converting column 'f5' from VARCHAR to TEXT   |
| Note  | 1246 | Converting column 'f6' from VARBINARY to BLOB |
+-------+------+-----------------------------------------------+

DESCRIBE SilenceIsGolden;
+-------+------------------+------+-----+---------+-------+
| Field | Type             | Null | Key | Default | Extra |
+-------+------------------+------+-----+---------+-------+
| f1    | blob             | YES  |     | NULL    |       |
| f2    | varbinary(15)    | YES  |     | NULL    |       |
| f3    | binary(1)        | YES  |     | NULL    |       |
| f4    | enum('x','y','z') | YES  |     | NULL    |       |
| f5    | mediumtext       | YES  |     | NULL    |       |
| f6    | mediumblob       | YES  |     | NULL    |       |
| f7    | tinyint(4)       | YES  |     | NULL    |       |
+-------+------------------+------+-----+---------+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

DROP

See DROP DATABASE.

Complete guide to removing databases in MariaDB. Complete DROP DATABASE syntax with IF EXISTS, permissions, and recovery options for production use.

See DROP EVENT.

Remove a scheduled event from the server. This command stops the event from executing and deletes its definition from the system tables.

See DROP INDEX.

Remove an existing index from a table. This command deletes the index structure, potentially impacting query performance but freeing storage.

See DROP LOGFILE GROUP.

Remove a log file group. This statement, primarily for NDB Cluster, deletes the undo log files associated with the specified log file group.

See DROP PACKAGE BODY.

Delete the body of a stored package. This command removes the implementation logic while preserving the package specification and interface.

See DROP PACKAGE.

Remove a stored package completely. This command deletes both the package specification (interface) and its body (implementation) from the database.

See DROP SERVER.

Remove a server definition. This command deletes the connection details for a remote server used by the FEDERATED or SPIDER storage engines.

See DROP TABLE.

Complete DROP TABLE syntax: TEMPORARY, IF EXISTS, WAIT/NOWAIT, RESTRICT/CASCADE options, metadata locks, atomic DROP, and replication behavior.

See DROP TABLESPACE.

Delete a tablespace. This command removes the physical file container used for storing table data, applicable to engines like InnoDB or NDB.

See DROP TRIGGER.

Remove a trigger from a table. This command deletes the trigger definition, preventing it from firing on future INSERT, UPDATE, or DELETE events.

See DROP VIEW.

Explains how to use the DROP VIEW statement to remove one or more views from the database, including required privileges.

This page is licensed: CC BY-SA / Gnu FDL

DROP DATABASE

Syntax

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

Description

`DROP DATABASE` drops all tables in the database and deletes the database. Be very careful with this statement! To use DROP DATABASE, you need the DROP privilege on the database. `DROP SCHEMA` is a synonym for `DROP DATABASE`.

Important: When a database is dropped, user privileges on the database are not automatically dropped. See GRANT.

IF EXISTS

Use `IF EXISTS` to prevent an error from occurring for databases that do not exist. A `NOTE` is generated for each non-existent database when using `IF EXISTS`. See SHOW WARNINGS.

Atomic DDL

> **CURRENT**
>
> MariaDB 10.6.1 supports Atomic DDL. `DROP DATABASE` is implemented as
>
> ```
> LOOP OVER ALL tables
>   DROP TABLE tbl
> ```
>
> Each individual DROP TABLE is atomic while `DROP DATABASE` as a whole is crash-safe.

> **< 10.6.1**
>
> Atomic `DROP` is not available.

Examples

```
DROP DATABASE bufg;
Query OK, 0 rows affected (0.39 sec)

DROP DATABASE bufg;
ERROR 1008 (HY000): Can't drop database 'bufg'; database doesn't exist

 \W
SHOW warnings enabled.

DROP DATABASE IF EXISTS bufg;
Query OK, 0 rows affected, 1 warning (0.00 sec)
Note (Code 1008): Can't DROP DATABASE 'bufg'; DATABASE doesn't exist
```

See Also

- CREATE DATABASE
- ALTER DATABASE
- SHOW DATABASES
- Information Schema SCHEMATA Table
- SHOW CREATE DATABASE

This page is licensed: GPLv2, originally from fill_help_tables.sql

DROP EVENT

Syntax

```
DROP EVENT [IF EXISTS] event_name
```

Description

This statement drops the event named `event_name`. The event immediately ceases being active, and is deleted completely from the server.

If the event does not exist, the error `ERROR 1517 (HY000): Unknown event 'event_name'` results. You can override this and cause the statement to generate a `NOTE` for non-existent events instead by using `IF EXISTS`. See SHOW WARNINGS.

This statement requires the EVENT privilege.

Examples

```
DROP EVENT myevent3;
```

Using the IF EXISTS clause:

```
DROP EVENT IF EXISTS myevent3;
Query OK, 0 rows affected, 1 warning (0.01 sec)

SHOW WARNINGS;
+-------+------+-------------------------------+
| Level | Code | Message                       |
+-------+------+-------------------------------+
| Note  | 1305 | Event myevent3 does not exist |
+-------+------+-------------------------------+
```

See also

- Events Overview
- CREATE EVENT
- SHOW CREATE EVENT
- ALTER EVENT

This page is licensed: GPLv2, originally from fill_help_tables.sql

DROP INDEX

Syntax

```
DROP INDEX [IF EXISTS] index_name ON tbl_name
    [WAIT n |NOWAIT]
```

Description

`DROP INDEX` drops the index named `index_name` from the table `tbl_name`. This statement is mapped to an `ALTER TABLE` statement to drop the index.

If another connection is using the table, a metadata lock is active, and this statement will wait until the lock is released. This is also true for non-transactional tables.

See ALTER TABLE.

Another shortcut, CREATE INDEX, allows the creation of an index.

To remove the primary key, `PRIMARY` must be specified as index_name. Note that the quotes are necessary, because `PRIMARY` is a keyword.

Privileges

Executing the `DROP INDEX` statement requires the INDEX privilege for the table or the database.

Online DDL

Online DDL is used by default with InnoDB, when the drop index operation supports it.

See InnoDB Online DDL Overview for more information on online DDL with InnoDB.

DROP INDEX IF EXISTS ...

If the `IF EXISTS` clause is used, then MariaDB will return a warning instead of an error if the index does not exist.

WAIT/NOWAIT

Sets the lock wait timeout. See WAIT and NOWAIT.

Progress Reporting

MariaDB provides progress reporting for `DROP INDEX` statement for clients that support the new progress reporting protocol. For example, if you were using the mariadb client, then the progress report might look like this::

See Also

- Getting Started with Indexes
- CREATE INDEX
- ALTER TABLE

This page is licensed: GPLv2, originally from fill_help_tables.sql

DROP LOGFILE GROUP

> **Note:**
>
> The `DROP LOGFILE GROUP` statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. See MDEV-19295 for more information.

This page is licensed: CC BY-SA / Gnu FDL

DROP PACKAGE BODY

Syntax

```
DROP PACKAGE BODY [IF EXISTS]  [ db_name . ] package_name
```

Description

The `DROP PACKAGE BODY` statement can be used when Oracle SQL_MODE is set.

The `DROP PACKAGE BODY` statement drops the package body (that is, the implementation) which was created using the CREATE PACKAGE BODY statement.

Note, `DROP PACKAGE BODY` drops only the package implementation, but does not drop the package specification. Use DROP PACKAGE to drop the package entirely (i.e. both implementation and specification).

See also

- CREATE PACKAGE
- SHOW CREATE PACKAGE
- DROP PACKAGE
- CREATE PACKAGE BODY
- SHOW CREATE PACKAGE BODY
- Oracle SQL_MODE

This page is licensed: CC BY-SA / Gnu FDL

DROP PACKAGE

Syntax

```
DROP PACKAGE [IF EXISTS]  [ db_name . ] package_name
```

Description

The `DROP PACKAGE` statement can be used when Oracle SQL_MODE is set.

The `DROP PACKAGE` statement drops a stored package entirely:

- Drops the package specification (earlier created using the CREATE PACKAGE statement).
- Drops the package implementation, if the implementation was already created using the CREATE PACKAGE BODY statement.

See Also

- SHOW CREATE PACKAGE
- CREATE PACKAGE
- CREATE PACKAGE BODY
- DROP PACKAGE BODY
- Oracle SQL_MODE

This page is licensed: CC BY-SA / Gnu FDL

DROP SERVER

Syntax

```
DROP SERVER [ IF EXISTS ] server_name
```

Description

Drops the server definition for the server named *server_name*. The corresponding row within the mysql.servers table will be deleted. This statement requires the SUPER privilege or, from MariaDB 10.5.2, the FEDERATED ADMIN privilege.

Dropping a server for a table does not affect any FederatedX, FEDERATED, Connect or Spider tables that used this connection information when they were created.

DROP SERVER is not written to the binary log, irrespective of the binary log format being used. From MariaDB 10.1.13, Galera replicates the CREATE SERVER, ALTER SERVER and DROP SERVER statements.

IF EXISTS

If the IF EXISTS clause is used, MariaDB will not return an error if the server does not exist. Unlike all other statements, DROP SERVER IF EXISTS does not issue a note if the server does not exist. See MDEV-9400.

Examples

```
DROP SERVER s;
```

IF EXISTS:

```
DROP SERVER s;
ERROR 1477 (HY000): The foreign server name you are trying to reference
  does not exist. Data source error:  s

DROP SERVER IF EXISTS s;
Query OK, 0 rows affected (0.00 sec)
```

See Also

- CREATE SERVER
- ALTER SERVER
- Spider Storage Engine
- FederatedX Storage Engine
- Connect Storage Engine
- mysql.servers table

This page is licensed: GPLv2, originally from fill_help_tables.sql

DROP TABLESPACE

> **Note:**
>
> The `DROP TABLESPACE` statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. In MySQL 5.7 and later, the statement is also supported for InnoDB. However, MariaDB has chosen not to include that specific feature. See MDEV-19294 for more information.

This page is licensed: GPLv2, originally from fill_help_tables.sql

DROP TRIGGER

Syntax

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name
```

Description

This statement drops a trigger. The schema (database) name is optional. If the schema is omitted, the trigger is dropped from the default schema. Its use requires the `TRIGGER` privilege for the table associated with the trigger.

Use `IF EXISTS` to prevent an error from occurring for a trigger that does not exist. A `NOTE` is generated for a non-existent trigger when using `IF EXISTS`. See SHOW WARNINGS.

Note: Triggers for a table are also dropped if you drop the table.

Atomic DDL

CURRENT

`DROP TRIGGER` is atomic.

< 10.6.1

`DROP TRIGGER` is **not** atomic.

Examples

```
DROP TRIGGER test.example_trigger;
```

Using the IF EXISTS clause:

```
DROP TRIGGER IF EXISTS test.example_trigger;
Query OK, 0 rows affected, 1 warning (0.01 sec)

SHOW WARNINGS;
+-------+------+------------------------+
| Level | Code | Message                |
+-------+------+------------------------+
| Note  | 1360 | Trigger does not exist |
+-------+------+------------------------+
```

See Also

- Trigger Overview
- CREATE TRIGGER
- Information Schema TRIGGERS Table
- SHOW TRIGGERS
- SHOW CREATE TRIGGER
- Trigger Limitations

This page is licensed: GPLv2, originally from fill_help_tables.sql

Data Manipulation (DML)

See Changing & Deleting Data.

Learn to change and delete data in MariaDB Server. This section covers UPDATE and DELETE SQL statements, enabling you to modify existing records and remove unwanted information efficiently.

See Inserting & Loading Data.

Learn to insert and load data into MariaDB Server. This section covers INSERT and LOAD DATA SQL statements, enabling you to efficiently add new records to your databases.

See Selecting Data.

Learn to select data in MariaDB Server using SQL. This section covers various SELECT statement clauses, including WHERE, GROUP BY, and ORDER BY, to retrieve and filter your data effectively.

This page is licensed: CC BY-SA / Gnu FDL

Changing & Deleting Data

See DELETE.

Complete guide to deleting data in MariaDB. Complete DELETE syntax with WHERE filtering, JOIN operations, CTEs, and safety considerations for production use.

See HIGH_PRIORITY and LOW_PRIORITY.

Control locking priority for table access. These modifiers determine whether read or write operations take precedence when multiple threads access a table.

See REPLACE.

Insert or replace rows based on unique keys. This statement acts like INSERT, but if a duplicate key exists, it deletes the old row and inserts the new one.

See REPLACE…RETURNING.

Replace rows and retrieve the results immediately. This extension returns the values of the replaced or inserted rows in the same operation.

See UPDATE.

Complete UPDATE statement guide for MariaDB. Complete syntax reference with WHERE conditions, JOIN operations, CTEs, and multi-table updates for production use.

This page is licensed: CC BY-SA / Gnu FDL

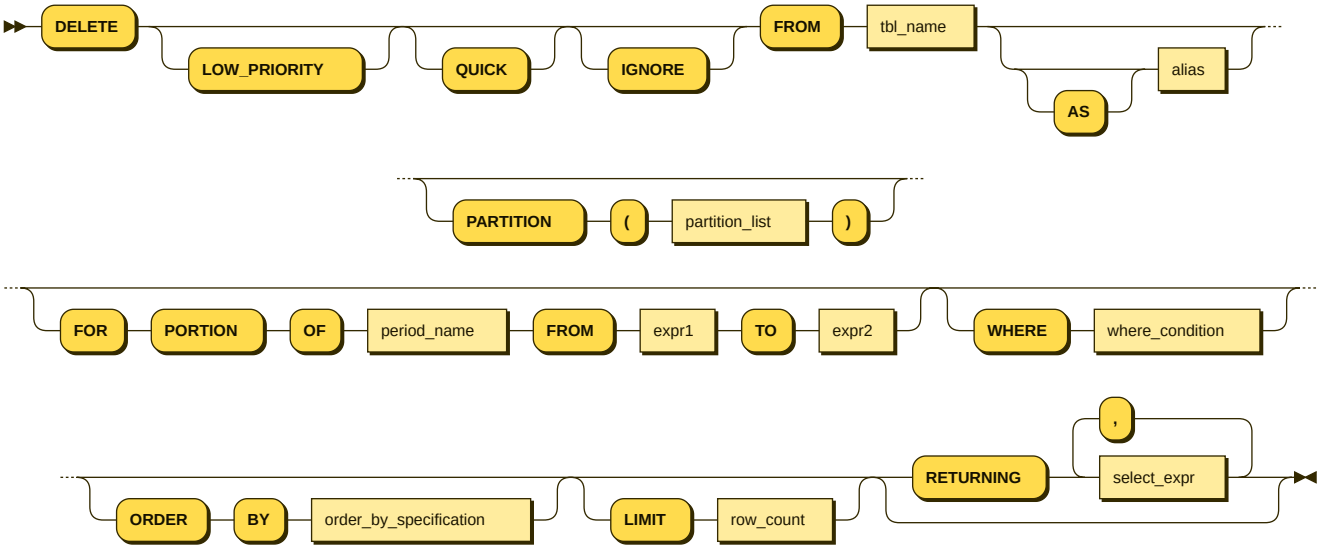
DELETE

Syntax

> **Note:**
>
> For the CTE[42] syntax, available from MariaDB 12.3, see here.

Single-table syntax:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
  FROM tbl_name [[AS] alias]    -- alias from MariaDB 11.6
  [PARTITION (partition_list)]
  [FOR PORTION OF period_name FROM expr1 TO expr2]
  [{USE|FORCE|IGNORE} {INDEX|KEY}
    [FOR {JOIN|ORDER BY|GROUP BY}]
    (index_list)]               -- from MariaDB 11.8.1
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
  [RETURNING select_expr
    [, select_expr ...]]
```



Railroad diagram of single-table DELETE — equivalent to the BNF above

The `AS alias` clause is available from MariaDB 11.6. `order_by_specification` stands in for the abbreviated `ORDER BY ...` in the source BNF; see ORDER BY for its full form.

Multiple-table syntax:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    tbl_name[.*] [, tbl_name[.*]] ...
    FROM table_references
    [WHERE where_condition]
    [ORDER BY ...]              -- from MariaDB 11.8.1
    [LIMIT row_count]          -- from MariaDB 11.8.1
```

Or:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    FROM tbl_name[.*] [, tbl_name[.*]] ...
    USING table_references
    [WHERE where_condition]
    [ORDER BY ...]              -- from MariaDB 11.8.1
    [LIMIT row_count]          -- from MariaDB 11.8.1
```

Trimming history:

```
DELETE HISTORY
  FROM tbl_name [PARTITION (partition_list)]
  [BEFORE SYSTEM_TIME [TIMESTAMP|TRANSACTION] expression]
```

CTE Syntax

> **Note:**
>
> This syntax is available from MariaDB 12.3.

```
WITH [RECURSIVE] table_reference [(columns_list)] AS  (
  SELECT ...
)
DELETE FROM non_cte_table expression
```

- `non_cte_table` is a table not defined by a CTE (common table expression).
- `expression` is a `WHERE` clause or a `USING` / `WHERE` clause.
- Supporting CTEs with `DELETE` is an extension of the SQL standard, similar to how MySQL does it.
- With `DELETE`, CTEs are read-only, like other derived tables – you cannot delete rows from tables in the CTE expression.
- For use cases, see the CTE examples.

Description

Option	Description
LOW_PRIORITY	Wait until all `SELECT` statement are done before starting the statement. Used with storage engines that uses table locking (MyISAM, Aria etc). See HIGH_PRIORITY and LOW_PRIORITY clauses for details.
QUICK	Signal the storage engine that it should expect that a lot of rows are deleted. The storage engine can do things to speed up the `DELETE` like ignoring merging of data blocks until all rows are deleted from the block (instead of when a block is half full). This speeds up things at the expanse of lost space in data blocks. At least MyISAM and Aria support this feature.
IGNORE	Don't stop the query even if a not-critical error occurs (like data overflow). See How IGNORE works for a full description.

For the single-table syntax, the `DELETE` statement deletes rows from tbl_name and returns a count of the number of deleted rows. This count can be obtained by calling the ROW_COUNT() function. The `WHERE` clause, if given, specifies the conditions that identify which rows to delete. With no `WHERE` clause, all rows are deleted. If the ORDER BY clause is specified, the rows are deleted in the order that is specified. The LIMIT clause places a limit on the number of rows that can be deleted.

For the multiple-table syntax, `DELETE` deletes from each `tbl_name` the rows that satisfy the conditions. From MariaDB 11.8.1, the multiple-table syntax also accepts ORDER BY and LIMIT; in earlier releases, these clauses could not be used with multiple-table `DELETE`. A `DELETE` can also reference tables which are located in different databases; see Identifier Qualifiers for the syntax.

From MariaDB 11.8.1, single-table `DELETE` accepts index hints (`USE INDEX`, `FORCE INDEX`, and `IGNORE INDEX`) to influence which index the optimizer uses, with the same syntax as in SELECT. See Index Hints: How to Force Query Plans.

`where_condition` is an expression that evaluates to true for each row to be deleted. It is specified as described in SELECT.

You need the `DELETE` privilege on a table to delete rows from it. You need only the `SELECT` privilege for any columns that are only read, such as those named in the `WHERE` clause. See GRANT.

As stated, a `DELETE` statement with no `WHERE` clause deletes all rows. A faster way to do this, when you do not need to know the number of deleted rows, is to use `TRUNCATE TABLE`. However, within a transaction or if you have a lock on the table, `TRUNCATE TABLE` cannot be used whereas `DELETE` can. See TRUNCATE TABLE, and LOCK.

AS

> **CURRENT**
>
> Single-table `DELETE` statements support aliases. For example:
>
> ```
> CREATE TABLE t1 (c1 INT);
> INSERT INTO t1 VALUES (1), (2);
>
> DELETE FROM t1 AS a1 WHERE a1.c1 = 2;
> ```
>
> **< 11.6**
>
> Single-table `DELETE` statements do **not** support aliases.

PARTITION

See Partition Pruning and Selection for details.

FOR PORTION OF

See Application Time Periods - Deletion by Portion.

RETURNING

It is possible to return a result set of the deleted rows for a single table to the client by using the syntax `DELETE ... RETURNING select_expr [, select_expr2 ...]]`

Any of SQL expression that can be calculated from a single row fields is allowed. Subqueries are allowed. The AS keyword is allowed, so it is possible to use aliases.

The use of aggregate functions is not allowed. `RETURNING` cannot be used in multi-table `DELETE` statements.

Same Source and Target Table

It is possible to delete from a table with the same source and target. For example:

```
DELETE FROM t1 WHERE c1 IN (SELECT b.c1 FROM t1 b WHERE b.c2=0);
```

DELETE HISTORY

You can use `DELETE HISTORY` to delete historical information from System-versioned tables.

Examples

ORDER BY and LIMIT

How to use the ORDER BY and LIMIT clauses:

```
DELETE FROM page_hit ORDER BY TIMESTAMP LIMIT 1000000;
```

From MariaDB 11.8.1, `ORDER BY` and `LIMIT` can also be used with the multiple-table syntax:

```
DELETE t1.*, t2.* FROM t1, t2 ORDER BY t1.id DESC LIMIT 3;
```

How to use the `RETURNING` clause:

```
DELETE FROM t RETURNING f1;
+------+
| f1   |
+------+
|    5 |
|   50 |
|  500 |
+------+
```

The following statement joins two tables: one is only used to satisfy a `WHERE` condition, but no row is deleted from it; rows from the other table are deleted, instead.

```
DELETE post FROM blog INNER JOIN post WHERE blog.id = post.blog_id;
```

Deleting from the Same Source and Target

```
CREATE TABLE t1 (c1 INT, c2 INT);
DELETE FROM t1 WHERE c1 IN (SELECT b.c1 FROM t1 b WHERE b.c2=0);
```

The statement returns:

```
Query OK, 0 rows affected (0.00 sec)
```

CTE Single-Table

```
WITH cte1 AS (SELECT * FROM t1 WHERE c < 5),
     cte2 AS (SELECT * FROM t2 WHERE b < 5)
     DELETE FROM t3 WHERE t3.a = (SELECT a FROM cte1 WHERE b IN (SELECT b FROM cte2));
```

CTE Multi-Table

```
WITH cte1 AS (SELECT * FROM t1 WHERE c < 5),
     cte2 AS (SELECT * FROM t2 WHERE b < 5)
     DELETE FROM t3 USING t3, cte1 WHERE t3.a = cte1.a AND cte1.b IN (SELECT b FROM cte2);
```

```
WITH cte1 AS (SELECT * FROM t1 WHERE c < 5),
     cte2 AS (SELECT * FROM t2 WHERE b < 5)
     DELETE FROM t3 USING t3, cte1, cte2 WHERE cte1.a AND cte1.b = cte2.b;
```

See Also

- How IGNORE works
- SELECT
- ORDER BY

- LIMIT
- REPLACE ... RETURNING
- INSERT ... RETURNING
- Returning clause (video)

This page is licensed: GPLv2, originally from fill_help_tables.sql

HIGH_PRIORITY and LOW_PRIORITY

The InnoDB storage engine uses row-level locking to ensure data integrity. However some storage engines (such as MEMORY, MyISAM, Aria and MERGE) lock the whole table to prevent conflicts. These storage engines use two separate queues to remember pending statements; one is for SELECTs and the other one is for write statements (INSERT, DELETE, UPDATE). By default, the latter has a higher priority.

To give write operations a lower priority, the low_priority_updates server system variable can be set to `ON`. The option is available on both the global and session levels, and it can be set at startup or via the SET statement.

When too many table locks have been set by write statements, some pending SELECTs are executed. The maximum number of write locks that can be acquired before this happens is determined by the max_write_lock_count server system variable, which is dynamic.

If write statements have a higher priority (default), the priority of individual write statements (INSERT, REPLACE, UPDATE, DELETE) can be changed via the `LOW_PRIORITY` attribute, and the priority of a `SELECT` statement can be raised via the `HIGH_PRIORITY` attribute. Also, LOCK TABLES supports a `LOW_PRIORITY` attribute for `WRITE` locks.

If read statements have a higher priority, the priority of an `INSERT` can be changed via the `HIGH_PRIORITY` attribute. However, the priority of other write statements cannot be raised individually.

The use of `LOW_PRIORITY` or `HIGH_PRIORITY` for an `INSERT` prevents Concurrent Inserts from being used.

See Also

- INSERT
- INSERT DELAYED
- INSERT SELECT
- Concurrent Inserts
- INSERT - Default & Duplicate Values
- INSERT IGNORE
- INSERT ON DUPLICATE KEY UPDATE

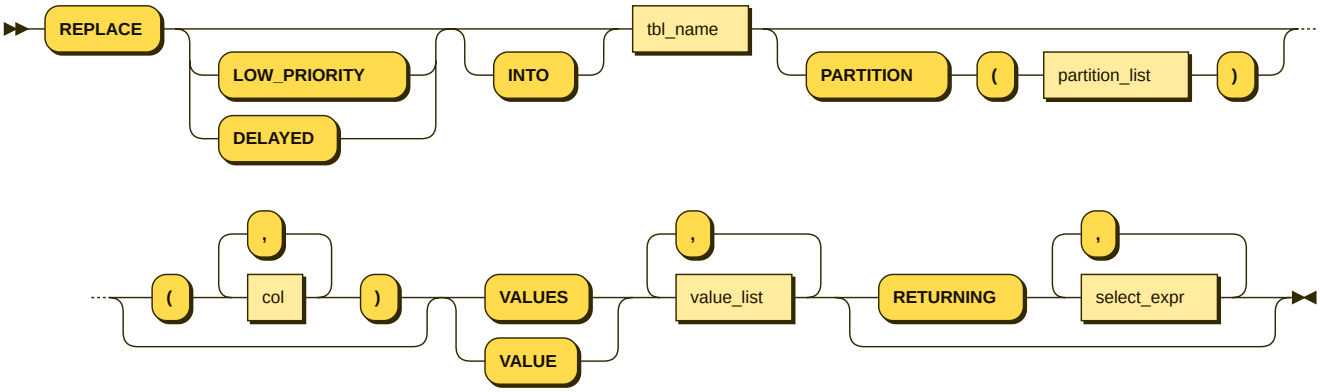
This page is licensed: CC BY-SA / Gnu FDL

REPLACE

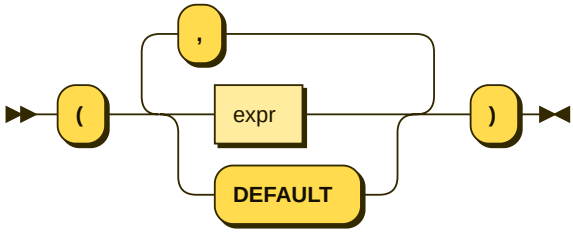
> **Note:**
>
> The RETURNING clause was introduced in MariaDB 10.5.0.

Syntax

```
REPLACE [LOW_PRIORITY | DELAYED]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[RETURNING select_expr
      [, select_expr ...]]
```



Railroad diagram of REPLACE — equivalent to the BNF above



Railroad diagram of value_list

Or:

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [PARTITION (partition_list)]
    SET col={expr | DEFAULT}, ...
[RETURNING select_expr
      [, select_expr ...]]
```

Or:

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
    SELECT ...
[RETURNING select_expr
      [, select_expr ...]]
```

Description

`REPLACE` works exactly like INSERT, except that if an old row in the table has the same value as a new row for a `PRIMARY KEY` or a `UNIQUE` index, the old row is deleted before the new row is inserted. If the table has more than one `UNIQUE` keys, it is possible that the new row conflicts with more than one row. In this case, all conflicting rows will be deleted.

The table name can be specified in the form `db_name`.`tbl_name` or, if a default database is selected, in the form `tbl_name` (see Identifier Qualifiers). This allows to use REPLACE … SELECT to copy rows between different databases.

Basically it works like this:

```
BEGIN;
SELECT 1 FROM t1 WHERE KEY=# FOR UPDATE;
IF FOUND-ROW
  DELETE FROM t1 WHERE KEY=# ;
ENDIF
INSERT INTO t1 VALUES (...);
END;
```

The above can be replaced with:

```
REPLACE INTO t1 VALUES (...)
```

`REPLACE` is a MariaDB/MySQL extension to the SQL standard. It either inserts, or deletes and inserts. For other MariaDB/MySQL extensions to standard SQL — that also handle duplicate values — see IGNORE and INSERT ON DUPLICATE KEY UPDATE.

Note that unless the table has a `PRIMARY KEY` or `UNIQUE` index, using a `REPLACE` statement makes no sense. It becomes equivalent to `INSERT`, because there is no index to be used to determine whether a new row duplicates another.

Values for all columns are taken from the values. See Partition Pruning and Selection for details. Specified in the `REPLACE` statement. Any missing columns are set to their default values, just as happens for `INSERT`. You cannot refer to values from the current row and use them in the new row. If you use an assignment such as `'SET col = col + 1'`, the reference to the column name on the right hand side is treated as `DEFAULT(col)`, so the assignment is equivalent to `'SET col = DEFAULT(col) + 1'`.

To use `REPLACE`, you must have both the `INSERT` and `DELETE` privileges for the table.

There are some issues you should be aware of, before using `REPLACE`:

- If there is an AUTO_INCREMENT field, a new value will be generated.
- If there are foreign keys, `ON DELETE` action will be activated by `REPLACE`.
- Triggers on `DELETE` and `INSERT` will be activated by `REPLACE`.

To avoid some of these behaviors, you can use `INSERT ... ON DUPLICATE KEY UPDATE`.

This statement activates `INSERT` and `DELETE` triggers. See Trigger Overview for details.

PARTITION

See Partition Pruning and Selection for details.

REPLACE RETURNING

`REPLACE ... RETURNING` returns a resultset of the replaced rows. This returns the listed columns for all the rows that are replaced, or alternatively, the specified `SELECT` expression. Any SQL expressions which can be calculated can be used in the select expression for the `RETURNING` clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

Examples

Simple `REPLACE` statement:

```
REPLACE INTO t2 VALUES (1,'Leopard'),(2,'Dog') RETURNING id2, id2+id2
as Total ,id2|id2, id2&&id2;
+-----+-------+---------+----------+
| id2 | Total | id2|id2 | id2&&id2 |
+-----+-------+---------+----------+
|   1 |     2 |       1 |        1 |
|   2 |     4 |       2 |        1 |
+-----+-------+---------+----------+
```

Using stored functions in `RETURNING`:

```
DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
    BEGIN
      RETURN (SELECT arg+arg);
    END|

DELIMITER ;
PREPARE stmt FROM "REPLACE INTO t2 SET id2=3, animal2='Fox' RETURNING f2(id2),
UPPER(animal2)";

EXECUTE stmt;
+---------+----------------+
| f2(id2) | UPPER(animal2) |
+---------+----------------+
|       6 | FOX            |
+---------+----------------+
```

Subqueries in the statement:

```
REPLACE INTO t1 SELECT * FROM t2 RETURNING (SELECT id2 FROM t2 WHERE
id2 IN (SELECT id2 FROM t2 WHERE id2=1)) AS new_id;
+--------+
| new_id |
+--------+
|      1 |
|      1 |
|      1 |
|      1 |
+--------+
```

Subqueries in the `RETURNING` clause that return more than one row or column cannot be used..

Aggregate functions cannot be used in the RETURNING clause. Since aggregate functions work on a set of values and if the purpose is to get the row count, `ROW_COUNT()` with `SELECT` can be used, or it can be used in `REPLACE...SELECT`.

`REPLACE ... RETURNING` returns a result set of the replaced rows. This returns the listed columns for all the rows that are replaced, or alternatively, the specified `SELECT` expression. Any SQL expressions which can be calculated can be used in the select expression for the `RETURNING` clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

Examples

Simple `REPLACE` statement:

```
REPLACE INTO t2 VALUES (1,'Leopard'),(2,'Dog') RETURNING id2, id2+id2
as Total ,id2|id2, id2&&id2;
+-----+-------+---------+----------+
| id2 | Total | id2|id2 | id2&&id2 |
+-----+-------+---------+----------+
|   1 |     2 |       1 |        1 |
|   2 |     4 |       2 |        1 |
+-----+-------+---------+----------+
```

Using stored functions in `RETURNING`:

```
DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
    BEGIN
      RETURN (SELECT arg+arg);
    END|

DELIMITER ;
PREPARE stmt FROM "REPLACE INTO t2 SET id2=3, animal2='Fox' RETURNING f2(id2),
UPPER(animal2)";

EXECUTE stmt;
+---------+----------------+
| f2(id2) | UPPER(animal2) |
+---------+----------------+
|       6 | FOX            |
+---------+----------------+
```

Subqueries in the statement:

```
REPLACE INTO t1 SELECT * FROM t2 RETURNING (SELECT id2 FROM t2 WHERE
id2 IN (SELECT id2 FROM t2 WHERE id2=1)) AS new_id;
+--------+
| new_id |
+--------+
|      1 |
|      1 |
|      1 |
|      1 |
+--------+
```

Subqueries in the `RETURNING` clause that return more than one row or column cannot be used..

Aggregate functions cannot be used in the `RETURNING` clause. Since aggregate functions work on a set of values and if the purpose is to get the row count, `ROW_COUNT()` with `SELECT` can be used, or it can be used in `REPLACE...SELECT...RETURNING` if the table in the `RETURNING` clause is not the same as the `REPLACE` table. `SELECT...RETURNING` if the table in the `RETURNING` clause is not the same as the `REPLACE` table.

See Also

- INSERT
- HIGH_PRIORITY and LOW_PRIORITY clauses
- INSERT DELAYED for details on the `DELAYED` clause

This page is licensed: GPLv2, originally from fill_help_tables.sql

REPLACE...RETURNING

> **Note:**
>
> `REPLACE ... RETURNING` was added in MariaDB 10.5.0, and returns a result set of the replaced rows.

Syntax

```
REPLACE [LOW_PRIORITY | DELAYED]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[RETURNING select_expr
      [, select_expr ...]]
```

Or:

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [PARTITION (partition_list)]
    SET col={expr | DEFAULT}, ...
[RETURNING select_expr
      [, select_expr ...]]
```

Or:

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
    SELECT ...
[RETURNING select_expr
      [, select_expr ...]]
```

Description

`REPLACE ... RETURNING` returns a result set of the replaced rows. The statement returns the listed columns for all the rows that are replaced, or alternatively, the specified `SELECT` expression. Any SQL expressions which can be calculated can be used in the select expression for the RETURNING clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

Examples

Simple `REPLACE` statement:

```
REPLACE INTO t2 VALUES (1,'Leopard'),(2,'Dog') RETURNING id2, id2+id2
AS Total ,id2|id2, id2&&id2;
+-----+-------+---------+----------+
| id2 | Total | id2|id2 | id2&&id2 |
+-----+-------+---------+----------+
|   1 |     2 |       1 |        1 |
|   2 |     4 |       2 |        1 |
+-----+-------+---------+----------+
```

Using stored functions in `RETURNING`:

```
DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
    BEGIN
      RETURN (SELECT arg+arg);
    END|

DELIMITER ;
PREPARE stmt FROM "REPLACE INTO t2 SET id2=3, animal2='Fox' RETURNING f2(id2),
UPPER(animal2)";

EXECUTE stmt;
+---------+----------------+
| f2(id2) | UPPER(animal2) |
+---------+----------------+
|       6 | FOX            |
+---------+----------------+
```

Subqueries in the statement:

```
REPLACE INTO t1 SELECT * FROM t2 RETURNING (SELECT id2 FROM t2 WHERE
id2 IN (SELECT id2 FROM t2 WHERE id2=1)) AS new_id;
+--------+
| new_id |
+--------+
|      1 |
|      1 |
|      1 |
|      1 |
+--------+
```

Subqueries in the `RETURNING` clause that return more than one row or column cannot be used..

Aggregate functions cannot be used in the `RETURNING` clause. Since aggregate functions work on a set of values and if the purpose is to get the row count, `ROW_COUNT()` with `SELECT` can be used, or it can be used in `REPLACE...SELECT...RETURNING` if the table in the `RETURNING` clause is not the same as the `REPLACE` table.

See Also

- INSERT ... RETURNING
- DELETE ... RETURNING
- Returning clause (video)

This page is licensed: CC BY-SA / Gnu FDL

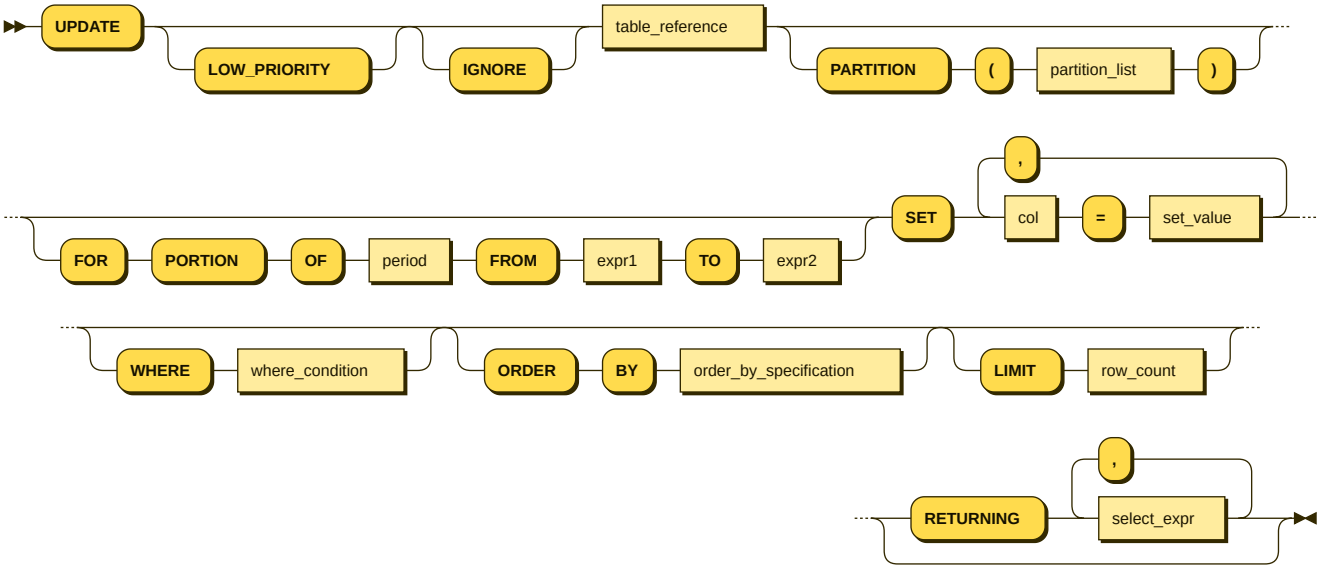
UPDATE

Syntax

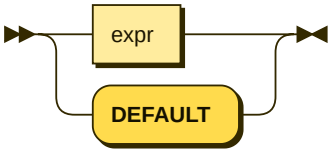
> **Note:**
>
> For the CTE[43] syntax, available from MariaDB 12.3, see here.

Single-table syntax:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
  [PARTITION (partition_list)]
  [FOR PORTION OF period FROM expr1 TO expr2]
  SET col1={expr1|DEFAULT} [,col2={expr2|DEFAULT}] ...
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
  [RETURNING select_expr
    [, select_expr ...]]
```



Railroad diagram of single-table UPDATE — equivalent to the BNF above



Railroad diagram of set_value

The abbreviated `[ORDER BY ...]` shown above is the standard `ORDER BY` clause; see ORDER BY for its full form.

> **Note:**
>
> The `RETURNING` clause is available from MariaDB 13.0.
>
> `RETURNING` works only for single tables.
>
> It uses the `OLD_VALUE()` function to return a value *val* as the old value (before the `UPDATE`), and optionally `val` as the new value (after the `UPDATE`).
>
> See this example for how it is used.

Multiple-table syntax:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
    SET col1={expr1|DEFAULT} [, col2={expr2|DEFAULT}] ...
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

CTE Syntax

> **Note:**
>
> This syntax is available from MariaDB 12.3.

```
WITH [RECURSIVE] table_reference [(columns_list)] AS  (
  SELECT ...
)
UPDATE non_cte_table expression
```

- `non_cte_table` is a table not defined by a CTE (common table expression).

- `expression` is the rest of the `UPDATE` statement body (the `SET` clause, and optionally `WHERE`, `ORDER BY`, `LIMIT`, and `RETURNING`).
- Supporting CTEs with `UPDATE` is an extension of the SQL standard, similar to how MySQL does it.
- With `UPDATE`, CTEs are read-only, like other derived tables – you cannot update columns from tables in the CTE expression.
- For use cases, see the CTE examples.

Description

For the single-table syntax, the `UPDATE` statement updates columns of existing rows in the named table with new values. The `SET` clause indicates which columns to modify and the values they should be given. Each value can be given as an expression, or the keyword `DEFAULT` to set a column explicitly to its default value. The `WHERE` clause, if given, specifies the conditions that identify which rows to update. With no `WHERE` clause, all rows are updated. If the ORDER BY clause is specified, the rows are updated in the order that is specified. The LIMIT clause places a limit on the number of rows that can be updated.

Both clauses can be used with multiple-table updates. For the multiple-table syntax, `UPDATE` updates rows in each table named in `table_references` that satisfy the conditions.

An `UPDATE` can also reference tables which are located in different databases; see Identifier Qualifiers for the syntax.

`where_condition` is an expression that evaluates to true for each row to be updated.

`table_references` and `where_condition` are as specified as described in SELECT.

For single-table updates, assignments are evaluated in left-to-right order, while for multi-table updates, there is no guarantee of a particular order. If the `SIMULTANEOUS_ASSIGNMENT` sql_mode is set, `UPDATE` statements evaluate all assignments simultaneously.

You need the `UPDATE` privilege only for columns referenced in an `UPDATE` that are actually updated. You need only the SELECT privilege for any columns that are read but not modified. See GRANT.

The `UPDATE` statement supports the following modifiers:

- If you use the `LOW_PRIORITY` keyword, execution of the `UPDATE` is delayed until no other clients are reading from the table. This affects only storage engines that use only table-level locking (MyISAM, MEMORY, MERGE). See HIGH_PRIORITY and LOW_PRIORITY clauses for details.
- If you use the `IGNORE` keyword, the update statement does not abort even if errors occur during the update. Rows for which duplicate-key conflicts occur are not updated. Rows for which columns are updated to values that would cause data conversion errors are updated to the closest valid values instead.

PARTITION

See Partition Pruning and Selection for details.

FOR PORTION OF

See Application Time Periods - Updating by Portion.

UPDATE Statements With the Same Source and Target

`UPDATE` statements may have the same source and target. For example, given the following table:

```
DROP TABLE t1;
CREATE TABLE t1 (c1 INT, c2 INT);
INSERT INTO t1 VALUES (10,10), (20,20);

UPDATE t1 SET c1=c1+1 WHERE c2=(SELECT MAX(c2) FROM t1);

SELECT * FROM t1;
+------+------+
| c1   | c2   |
+------+------+
|   10 |   10 |
|   21 |   20 |
+------+------+
```

Examples

Single-Table

```
UPDATE table_name SET column1 = value1, column2 = value2 WHERE id=100;
```

Single-Table With RETURNING Clause

> **Note:**
>
> The RETURNING clause is available from MariaDB 13.0.

```
UPDATE t SET a=a+1 RETURNING OLD_VALUE(a) AS old, a as new;
+------+------+
| old  | new  |
+------+------+
|    1 |    2 |
+------+------+
```

Multi-Table

```
UPDATE tab1, tab2 SET tab1.column1 = value1, tab1.column2 = value2 WHERE tab1.id = tab2.id;
```

CTE Single-Table

```
WITH cte1 AS (SELECT * FROM t1 WHERE c < 5),
     cte2 AS (SELECT * FROM t2 WHERE b < 5)
     UPDATE t3 SET t3.a = (SELECT a FROM cte1 WHERE b IN (SELECT b FROM cte2));
```

CTE Multi-Table

```
WITH cte1 AS (SELECT * FROM t1 WHERE c < 5),
     cte2 AS (SELECT * FROM t2 WHERE b < 5)
     UPDATE t3, cte1 SET t3.a = cte1.a WHERE cte1.b IN (SELECT b FROM cte2);
```

```
WITH cte1 AS (SELECT * FROM t1 WHERE c < 5),
     cte2 AS (SELECT * FROM t2 WHERE b < 5)
     UPDATE t3, cte1, cte2 SET t3.a = cte1.a WHERE cte1.b = cte2.b;
```

See Also

- How IGNORE works
- SELECT
- ORDER BY

- LIMIT
- Identifier Qualifiers

This page is licensed: GPLv2, originally from fill_help_tables.sql

Inserting & Loading Data

See Concurrent Inserts.

Understand concurrent inserts in MyISAM. This feature allows SELECT statements to run simultaneously with INSERT operations, reducing lock contention and improving performance.

See IGNORE.

Learn about the IGNORE keyword. This modifier suppresses certain errors during statement execution, downgrading them to warnings to allow the operation to proceed.

See INSERT.

Complete guide to inserting data in MariaDB. Complete INSERT syntax for single rows, bulk operations, and ON DUPLICATE KEY handling for production use.

See INSERT - Default & Duplicate Values.

Handle default and duplicate values during insertion. Learn how MariaDB manages missing columns and how to resolve duplicate key conflicts using various strategies.

See INSERT DELAYED.

Queue inserts for later execution. This MyISAM-specific extension returns control immediately to the client while the server inserts rows when the table is free.

See INSERT IGNORE.

Complete INSERT IGNORE reference: ignore insert errors and warnings, duplicate key error handling, invalid value coercion, and SHOW WARNINGS compatibility.

See INSERT ON DUPLICATE KEY UPDATE.

Complete guide to inserting data in MariaDB. Complete INSERT syntax for single rows, bulk operations, and ON DUPLICATE KEY handling for production use.

See INSERT SELECT.

Copy data between tables. This statement inserts the result set of a SELECT query directly into a target table, enabling efficient bulk data transfer.

See INSERT…RETURNING.

Insert rows and immediately retrieve the results. This extension returns the inserted values, including auto-increments and defaults, in the same round trip.

See LOAD Data into Tables or Index.

Bulk load data efficiently. This section covers commands like LOAD DATA INFILE and LOAD XML for high-speed data import from text or XML files.

This page is licensed: CC BY-SA / Gnu FDL

Concurrent Inserts

The MyISAM storage engine supports concurrent inserts. This feature allows SELECT statements to be executed during INSERT operations, reducing contention.

Whether concurrent inserts can be used or not depends on the value of the concurrent_insert server system variable:

- `NEVER` (0) disables concurrent inserts.
- `AUTO` (1) allows concurrent inserts only when the target table has no free blocks (no data in the middle of the table has been deleted after the last OPTIMIZE TABLE). This is the default.
- `ALWAYS` (2) always enables concurrent inserts, in which case new rows are added at the end of a table if the table is being used by another thread.

If the binary log is used, CREATE TABLE … SELECT and INSERT … SELECT statements cannot use concurrent inserts. These statements acquire a read lock on the table, so concurrent inserts will need to wait. This way, the log can be safely used to restore data.

Concurrent inserts are not used by replicas with the row-based replication (see binary log formats).

If an INSERT statement contains the HIGH_PRIORITY clause, concurrent inserts cannot be used. INSERT … DELAYED is usually unneeded if concurrent inserts are enabled.

LOAD DATA INFILE uses concurrent inserts if the `CONCURRENT` keyword is specified and concurrent_insert is not `NEVER`. This makes the statement slower (even if no other sessions access the table) but reduces contention.

LOCK TABLES allows non-conflicting concurrent inserts if a `READ LOCAL` lock is used. Concurrent inserts are not allowed if the `LOCAL` keyword is omitted.

Notes

The decision to enable concurrent insert for a table is done when the table is opened. If you change the value of concurrent_insert, it will only affect newly opened tables. If you want it to work for also for tables in use or cached, you should do FLUSH TABLES after setting the variable.

See Also

- INSERT
- INSERT DELAYED
- INSERT SELECT
- HIGH_PRIORITY and LOW_PRIORITY
- INSERT - Default & Duplicate Values
- INSERT IGNORE
- INSERT ON DUPLICATE KEY UPDATE

This page is licensed: CC BY-SA / Gnu FDL

IGNORE

The `IGNORE` option tells the server to ignore some common errors.

`IGNORE` can be used with the following statements:

- DELETE
- INSERT (see also INSERT IGNORE)
- LOAD DATA INFILE
- UPDATE
- ALTER TABLE
- CREATE TABLE … SELECT
- INSERT … SELECT

The logic used:

- Variables out of ranges are replaced with the maximum/minimum value.
- SQL_MODEs `STRICT_TRANS_TABLES`, `STRICT_ALL_TABLES`, `NO_ZERO_IN_DATE`, `NO_ZERO_DATE` are ignored.
- Inserting `NULL` in a `NOT NULL` field will insert 0 (in a numerical field), 0000-00-00 (in a date field) or an empty string (in a character field).
- Rows that cause a duplicate key error or break a foreign key constraint are not inserted, updated, or deleted.

The following errors are ignored:

Error number	Symbolic error name	Description
1022	ER_DUP_KEY	Can't write; duplicate key in table '%s'
1048	ER_BAD_NULL_ERROR	Column '%s' cannot be null
1062	ER_DUP_ENTRY	Duplicate entry '%s' for key %d
1242	ER_SUBQUERY_NO_1_ROW	Subquery returns more than 1 row
1264	ER_WARN_DATA_OUT_OF_RANGE	Out of range value for column '%s' at row %ld
1265	WARN_DATA_TRUNCATED	Data truncated for column '%s' at row %ld
1292	ER_TRUNCATED_WRONG_VALUE	Truncated incorrect %s value: '%s'
1366	ER_TRUNCATED_WRONG_VALUE_FOR_FIELD	Incorrect integer value
1369	ER_VIEW_CHECK_FAILED	CHECK OPTION failed '%s.%s'
1451	ER_ROW_IS_REFERENCED_2	Cannot delete or update a parent row
1452	ER_NO_REFERENCED_ROW_2	Cannot add or update a child row: a foreign key constraint fails (%s)
1526	ER_NO_PARTITION_FOR_GIVEN_VALUE	Table has no partition for value %s
1586	ER_DUP_ENTRY_WITH_KEY_NAME	Duplicate entry '%s' for key '%s'
1591	ER_NO_PARTITION_FOR_GIVEN_VALUE_SILENT	Table has no partition for some existing values
1748	ER_ROW_DOES_NOT_MATCH_GIVEN_PARTITION_SET	Found a row not matching the given partition set

Ignored errors normally generate a warning.

A property of the `IGNORE` clause consists in causing transactional engines and non-transactional engines (like InnoDB and Aria) to behave the same way. For example, normally a multi-row insert which tries to violate a `UNIQUE` constraint is completely rolled back on InnoDB, but might be partially executed on Aria. With the `IGNORE` clause, the statement will be partially executed in both engines.

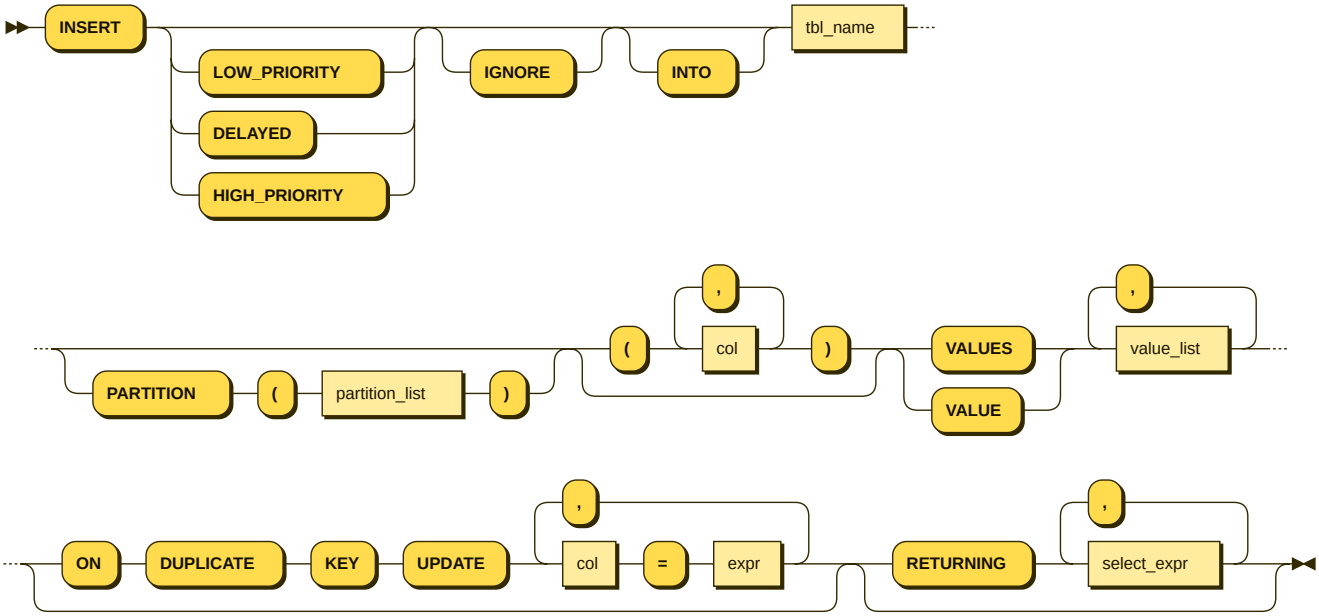
Duplicate key errors also generate warnings. The OLD_MODE server variable can be used to prevent this.

This page is licensed: CC BY-SA / Gnu FDL

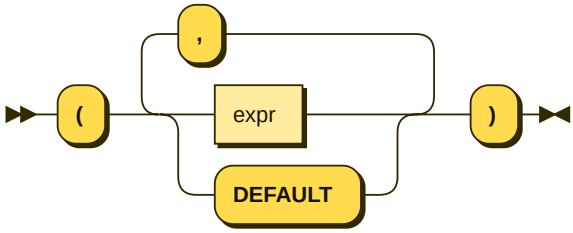
INSERT

Syntax

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
 [ ON DUPLICATE KEY UPDATE
   col=expr
     [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```



Railroad diagram of INSERT ... VALUES — equivalent to the BNF above



Railroad diagram of value_list

Or:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)]
    SET col={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Or:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

The `INSERT` statement is used to insert new rows into an existing table. The `INSERT ... VALUES` and `INSERT ... SET` forms of the statement insert rows based on explicitly specified values. The `INSERT ... SELECT` form inserts rows selected from another table or tables. `INSERT ... SELECT` is discussed further in the INSERT ... SELECT article.

The table name can be specified in the form `db_name.tbl_name` or, if a default database is selected, in the form `tbl_name` (see Identifier Qualifiers). This allows to use INSERT ... SELECT to copy rows between different databases.

The PARTITION clause can be used in both the INSERT and the SELECT part. See Partition Pruning and Selection for details.

> **CURRENT**
>
> The `RETURNING` clause can be used.
>
> **< 10.5**
>
> The `RETURNING` clause is not available.

The columns list is optional. It specifies which values are explicitly inserted, and in which order. If this clause is not specified, all values must be explicitly specified, in the same order they are listed in the table definition.

The list of value follow the `VALUES` or `VALUE` keyword (which are interchangeable, regardless how much values you want to insert), and is wrapped by parenthesis. The values must be listed in the same order as the columns list. It is possible to specify more than one list to insert more than one rows with a single statement. If many rows are inserted, this is a speed optimization.

For one-row statements, the `SET` clause may be more simple, because you don't need to remember the columns order. All values are specified in the form `col = expr`.

Values can also be specified in the form of a SQL expression or subquery. A subquery in the `VALUES` list may read the same table named in the `INTO` clause — its result is computed before the insert. A subquery in a `RETURNING` clause, however, cannot read the table being inserted into.

If you use the `LOW_PRIORITY` keyword, execution of the `INSERT` is delayed until no other clients are reading from the table. If you use the `HIGH_PRIORITY` keyword, the statement has the same priority as `SELECT`s. This affects only storage engines that use only table-level locking (MyISAM, MEMORY, MERGE). However, if one of these keywords is specified, concurrent inserts cannot be used. See HIGH_PRIORITY and LOW_PRIORITY clauses for details.

INSERT DELAYED

For more details on the `DELAYED` option, see INSERT DELAYED.

HIGH PRIORITY and LOW PRIORITY

See HIGH_PRIORITY and LOW_PRIORITY.

Defaults and Duplicate Values

See INSERT - Default & Duplicate Values for details..

INSERT IGNORE

See INSERT IGNORE.

INSERT ON DUPLICATE KEY UPDATE

See INSERT ON DUPLICATE KEY UPDATE.

Examples

Specifying the column names:

```
INSERT INTO person (first_name, last_name) VALUES ('John', 'Doe');
```

Inserting more than 1 row at a time:

```
INSERT INTO tbl_name VALUES (1, "row 1"), (2, "row 2");
```

Using the `SET` clause:

```
INSERT INTO person SET first_name = 'John', last_name = 'Doe';
```

SELECTing from another table:

```
INSERT INTO contractor SELECT * FROM person WHERE status = 'c';
```

See INSERT ON DUPLICATE KEY UPDATE and INSERT IGNORE for further examples.

INSERT ... RETURNING

`INSERT ... RETURNING` returns a result set of the inserted rows.

It returns the listed columns for all the rows that are inserted, or alternatively, the specified `SELECT` expression. Any SQL expressions which can be calculated can be used in the select expression for the `RETURNING` clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

Examples

Simple INSERT statement:

```
INSERT INTO t2 VALUES (1,'Dog'),(2,'Lion'),(3,'Tiger'),(4,'Leopard')
RETURNING id2,id2+id2,id2&id2,id2||id2;
+-----+---------+---------+----------+
| id2 | id2+id2 | id2&id2 | id2||id2 |
+-----+---------+---------+----------+
|   1 |       2 |       1 |        1 |
|   2 |       4 |       2 |        1 |
|   3 |       6 |       3 |        1 |
|   4 |       8 |       4 |        1 |
+-----+---------+---------+----------+
```

Using stored functions in RETURNING

```
DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
    BEGIN
       RETURN (SELECT arg+arg);
    END|

DELIMITER ;

PREPARE stmt FROM "INSERT INTO t1 SET id1=1, animal1='Bear' RETURNING f(id1), UPPER(animal1)";

EXECUTE stmt;
+---------+----------------+
| f(id1)  | UPPER(animal1) |
+---------+----------------+
|       2 | BEAR           |
+---------+----------------+
```

Subqueries in the `RETURNING` clause that return more than one row or column cannot be used.

Aggregate functions cannot be used in the `RETURNING` clause. Since aggregate functions work on a set of values, and if the purpose is to get the row count, `ROW_COUNT()` with `SELECT` can be used or it can be used in `INSERT...SELECT...RETURNING` if the table in the RETURNING clause is not the same as the `INSERT` table.

See Also

- INSERT DELAYED
- INSERT SELECT
- REPLACE Equivalent to `DELETE` + `INSERT` of conflicting row.
- HIGH_PRIORITY and LOW_PRIORITY
- Concurrent Inserts
- INSERT - Default & Duplicate Values
- INSERT IGNORE
- INSERT ON DUPLICATE KEY UPDATE
- How to quickly insert data into MariaDB

This page is licensed: GPLv2, originally from fill_help_tables.sql

INSERT - Default & Duplicate Values

Default Values

If the SQL_MODE contains `STRICT_TRANS_TABLES` and you are inserting into a transactional table (like InnoDB), or if the SQL_MODE contains `STRICT_ALL_TABLES`, all `NOT NULL` columns which do not have a `DEFAULT` value (and are not AUTO_INCREMENT) must be explicitly referenced in `INSERT` statements. If not, an error like this is produced:

```
ERROR 1364 (HY000): Field 'col' doesn't have a default value
```

In all other cases, if a `NOT NULL` column without a `DEFAULT` value is not referenced, an empty value will be inserted (for example, 0 for `INTEGER` columns and '' for `CHAR` columns). See NULL Values in MariaDB:Inserting for examples.

If a `NOT NULL` column that has a `DEFAULT` value is not referenced, its declared `DEFAULT` value is inserted.

If a `NULL` column having a `DEFAULT` value is not referenced, its default value will be inserted. It is also possible to explicitly assign the default value using the `DEFAULT` keyword or the DEFAULT() function.

If the `DEFAULT` keyword is used but the column does not have a `DEFAULT` value, an error like this is produced:

```
ERROR 1364 (HY000): Field 'col' doesn't have a default value
```

Duplicate Values

By default, if you try to insert a duplicate row and there is a `UNIQUE` index, `INSERT` stops and an error like this is produced:

```
ERROR 1062 (23000): Duplicate entry 'dup_value' for key 'col'
```

To handle duplicates you can use the IGNORE clause, INSERT ON DUPLICATE KEY UPDATE or the REPLACE statement. Note that the DELAYED option is ignored when you use ON DUPLICATE KEY UPDATE.

See Also

- INSERT
- INSERT DELAYED
- INSERT SELECT
- HIGH_PRIORITY and LOW_PRIORITY
- Concurrent Inserts

- INSERT IGNORE
- INSERT ON DUPLICATE KEY UPDATE

This page is licensed: CC BY-SA / Gnu FDL

INSERT DELAYED

Syntax

```
INSERT DELAYED ...
```

Description

The `DELAYED` option for the INSERT statement is a MariaDB/MySQL extension to standard SQL that is very useful if you have clients that cannot or need not wait for the `INSERT` to complete. This is a common situation when you use MariaDB for logging and you also periodically run `SELECT` and `UPDATE` statements that take a long time to complete.

When a client uses `INSERT DELAYED`, it gets an okay from the server at once, and the row is queued to be inserted when the table is not in use by any other thread.

Another major benefit of using `INSERT DELAYED` is that inserts from many clients are bundled together and written in one block. This is much faster than performing many separate inserts.

Note that `INSERT DELAYED` is slower than a normal `INSERT` if the table is not otherwise in use. There is also the additional overhead for the server to handle a separate thread for each table for which there are delayed rows. This means that you should use `INSERT DELAYED` only when you are really sure that you need it.

The queued rows are held only in memory until they are inserted into the table. This means that if you terminate mariadbd forcibly (for example, with kill -9) or if mariadbd dies unexpectedly, any queued rows that have not been written to disk are lost.

The number of concurrent `INSERT DELAYED` threads is limited by the max_delayed_threads server system variables. If it is set to 0, `INSERT DELAYED` is disabled. The session value can be equal to the global value, or 0 to disable this statement for the current session. If this limit has been reached, the `DELAYED` clause will be silently ignore for subsequent statements (no error will be produced).

Limitations

There are some limitations on the use of `DELAYED`:

- `INSERT DELAYED` works only with MyISAM, MEMORY, ARCHIVE, BLACKHOLE, non-transactional Aria (Aria tables are transactional by default), and OQGRAPH tables. If you execute INSERT DELAYED with another storage engine, you will get an error like this: `ERROR 1616 (HY000): DELAYED option not supported for table 'tab_name'`
- For MyISAM tables, if there are no free blocks in the middle of the data file, concurrent SELECT and INSERT statements are supported. Under these circumstances, you very seldom need to use `INSERT DELAYED` with MyISAM.
- `INSERT DELAYED` should be used only for `INSERT` statements that specify value lists. The server ignores `DELAYED` for `INSERT ... SELECT` or `INSERT ... ON DUPLICATE KEY UPDATE` statements.
- Because the `INSERT DELAYED` statement returns immediately, before the rows are inserted, you cannot use `LAST_INSERT_ID()` to get the `AUTO_INCREMENT` value that the statement might generate.
- `DELAYED` rows are not visible to `SELECT` statements until they actually have been inserted.
- After `INSERT DELAYED`, ROW_COUNT() returns the number of the rows you tried to insert, not the number of the successful writes.

- `DELAYED` is ignored on slave replication servers, so that `INSERT DELAYED` is treated as a normal `INSERT` on slaves. This is because `DELAYED` could cause the slave to have different data than the master. `INSERT DELAYED` statements are not safe for replication.
- Pending `INSERT DELAYED` statements are lost if a table is write locked and ALTER TABLE is used to modify the table structure.
- `INSERT DELAYED` is not supported for views. If you try, you will get an error like this: `ERROR 1347 (HY000): 'view_name' is not BASE TABLE`
- `INSERT DELAYED` is not supported for partitioned tables.
- `INSERT DELAYED` is not supported within stored programs.
- `INSERT DELAYED` does not work with triggers.
- `INSERT DELAYED` does not work if there is a check constraint in place.
- `INSERT DELAYED` does not work if skip-new mode is active.

See Also

- INSERT
- INSERT SELECT
- HIGH_PRIORITY and LOW_PRIORITY
- Concurrent Inserts
- INSERT - Default & Duplicate Values
- INSERT IGNORE
- INSERT ON DUPLICATE KEY UPDATE

This page is licensed: GPLv2, originally from fill_help_tables.sql

INSERT IGNORE

Ignoring Errors

Normally INSERT stops and rolls back when it encounters an error.

By using the IGNORE keyword all errors are converted to warnings, which will not stop inserts of additional rows.

Invalid values are changed to the closest valid value and inserted, with a warning produced.

The IGNORE and DELAYED options are ignored when you use ON DUPLICATE KEY UPDATE.

> **CURRENT**
>
> Warnings are issued for duplicate key errors when using `IGNORE`. You can get the old behavior if you set OLD_MODE to `NO_DUP_KEY_WARNINGS_WITH_IGNORE`.
>
> **< 5.5.28**
>
> No warnings are issued for duplicate key errors when using `IGNORE`.

See IGNORE for a full description of effects.

Examples

```
CREATE TABLE t1 (x INT UNIQUE);

INSERT INTO t1 VALUES(1),(2);

INSERT INTO t1 VALUES(2),(3);
ERROR 1062 (23000): Duplicate entry '2' for key 'x'
SELECT * FROM t1;
+------+
| x    |
+------+
|    1 |
|    2 |
+------+

INSERT IGNORE INTO t1 VALUES(2),(3);
Query OK, 1 row affected, 1 warning (0.04 sec)

SHOW WARNINGS;
+---------+------+---------------------------------+
| Level   | Code | Message                         |
+---------+------+---------------------------------+
| Warning | 1062 | Duplicate entry '2' for key 'x' |
+---------+------+---------------------------------+

SELECT * FROM t1;
+------+
| x    |
+------+
|    1 |
|    2 |
|    3 |
+------+
```

Converting values:

```
CREATE OR REPLACE TABLE t2(id INT, t VARCHAR(2) NOT NULL, n INT NOT NULL);

INSERT INTO t2(id) VALUES (1),(2);
ERROR 1364 (HY000): Field 't' doesn't have a default value

INSERT IGNORE INTO t2(id) VALUES (1),(2);
Query OK, 2 rows affected, 2 warnings (0.026 sec)
Records: 2  Duplicates: 0  Warnings: 2

SHOW WARNINGS;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1364 | Field 't' doesn't have a default value |
| Warning | 1364 | Field 'n' doesn't have a default value |
+---------+------+----------------------------------------+

SELECT * FROM t2;
+------+---+---+
| id   | t | n |
+------+---+---+
|    1 |   | 0 |
|    2 |   | 0 |
+------+---+---+
```

See INSERT ON DUPLICATE KEY UPDATE for further examples using that syntax.

See Also

- INSERT

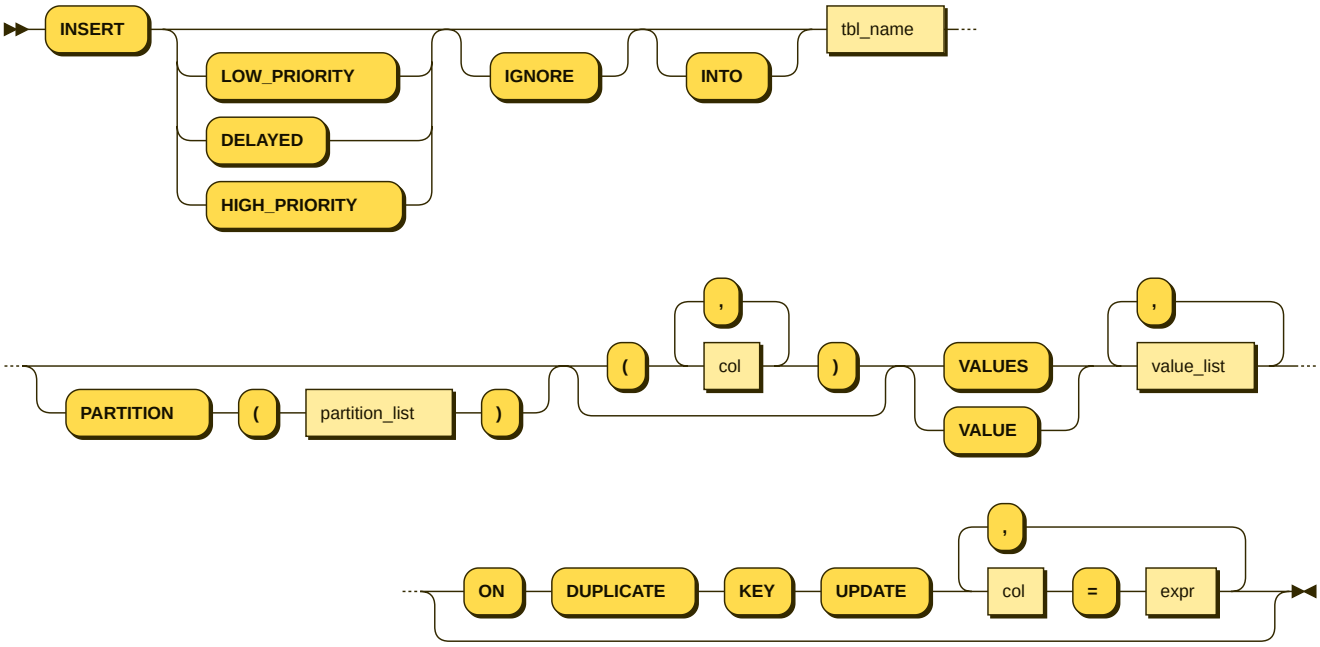
- INSERT DELAYED
- INSERT SELECT
- HIGH_PRIORITY and LOW_PRIORITY
- Concurrent Inserts
- INSERT - Default & Duplicate Values
- INSERT ON DUPLICATE KEY UPDATE

This page is licensed: CC BY-SA / Gnu FDL

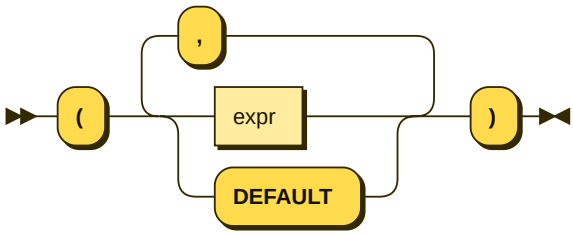
INSERT ON DUPLICATE KEY UPDATE

Syntax

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
  [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
  {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
  [ ON DUPLICATE KEY UPDATE
    col=expr
      [, col=expr] ... ]
```



Railroad diagram of INSERT ... ON DUPLICATE KEY UPDATE — equivalent to the BNF above



Railroad diagram of value_list

Or:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)]
    SET col={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ]
```

Or:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ]
```

Description

`INSERT ... ON DUPLICATE KEY UPDATE` (often called "upsert") is a MariaDB/MySQL extension to the INSERT statement that, if it finds a duplicate unique or primary key, will instead perform an UPDATE.

The row/s affected value is reported as 1 if a row is inserted, and 2 if a row is updated, unless the API's `CLIENT_FOUND_ROWS` flag is set.

If more than one unique index is matched, only the first is updated. It is not recommended to use this statement on tables with more than one unique index.

If the table has an AUTO_INCREMENT primary key and the statement inserts a new row, the LAST_INSERT_ID() function returns the AUTO_INCREMENT value of that newly inserted row. It is not affected when the statement updates an existing row instead.

The VALUES() function can only be used in a `ON DUPLICATE KEY UPDATE` clause and has no meaning in any other context. It returns the column values from the `INSERT` portion of the statement. This function is particularly useful for multi-rows inserts.

The DELAYED option is ignored when you use `ON DUPLICATE KEY UPDATE`.

See Partition Pruning and Selection for details on the PARTITION clause.

This statement activates INSERT and UPDATE triggers. See Trigger Overview for details.

See also a similar statement, REPLACE.

Examples

```
CREATE TABLE ins_duplicate (id INT PRIMARY KEY, animal VARCHAR(30));
INSERT INTO ins_duplicate VALUES (1,'Aardvark'), (2,'Cheetah'), (3,'Zebra');
```

If there is no existing key, the statement runs as a regular INSERT:

```
INSERT INTO ins_duplicate VALUES (4,'Gorilla')
  ON DUPLICATE KEY UPDATE animal='Gorilla';
Query OK, 1 row affected (0.07 sec)
```

```
SELECT * FROM ins_duplicate;
+----+----------+
| id | animal   |
+----+----------+
|  1 | Aardvark |
|  2 | Cheetah  |
|  3 | Zebra    |
|  4 | Gorilla  |
+----+----------+
```

A regular INSERT with a primary key value of 1 will fail, due to the existing key:

```
INSERT INTO ins_duplicate VALUES (1,'Antelope');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

However, we can use an INSERT ON DUPLICATE KEY UPDATE instead:

```
INSERT INTO ins_duplicate VALUES (1,'Antelope')
  ON DUPLICATE KEY UPDATE animal='Antelope';
Query OK, 2 rows affected (0.09 sec)
```

Note that there are two rows reported as affected, but this refers only to the UPDATE.

```
SELECT * FROM ins_duplicate;
+----+----------+
| id | animal   |
+----+----------+
|  1 | Antelope |
|  2 | Cheetah  |
|  3 | Zebra    |
|  4 | Gorilla  |
+----+----------+
```

Adding a second unique column:

```
ALTER TABLE ins_duplicate ADD id2 INT;
UPDATE ins_duplicate SET id2=id+10;
ALTER TABLE ins_duplicate ADD UNIQUE KEY(id2);
```

Where two rows match the unique keys match, only the first is updated. This can be unsafe and is not recommended unless you are certain what you are doing.

```
INSERT INTO ins_duplicate VALUES (2,'Lion',13)
  ON DUPLICATE KEY UPDATE animal='Lion';
Query OK, 2 rows affected (0.004 sec)

SELECT * FROM ins_duplicate;
+----+----------+------+
| id | animal   | id2  |
+----+----------+------+
|  1 | Antelope |   11 |
|  2 | Lion     |   12 |
|  3 | Zebra    |   13 |
|  4 | Gorilla  |   14 |
+----+----------+------+
```

Although the third row with an id of 3 has an id2 of 13, which also matched, it was not updated.

Changing id to an auto_increment field. If a new row is added, the auto_increment is moved forward. If the row is updated, it remains the same.

```
ALTER TABLE `ins_duplicate` CHANGE `id` `id` INT( 11 ) NOT NULL AUTO_INCREMENT;
ALTER TABLE ins_duplicate DROP id2;
SELECT Auto_increment FROM INFORMATION_SCHEMA.TABLES
  WHERE TABLE_NAME='ins_duplicate';
+----------------+
| Auto_increment |
+----------------+
|              5 |
+----------------+

INSERT INTO ins_duplicate VALUES (2,'Leopard')
  ON DUPLICATE KEY UPDATE animal='Leopard';
Query OK, 2 rows affected (0.00 sec)

SELECT Auto_increment FROM INFORMATION_SCHEMA.TABLES
  WHERE TABLE_NAME='ins_duplicate';
+----------------+
| Auto_increment |
+----------------+
|              5 |
+----------------+

INSERT INTO ins_duplicate VALUES (5,'Wild Dog')
  ON DUPLICATE KEY UPDATE animal='Wild Dog';
Query OK, 1 row affected (0.09 sec)

SELECT * FROM ins_duplicate;
+----+----------+
| id | animal   |
+----+----------+
|  1 | Antelope |
|  2 | Leopard  |
|  3 | Zebra    |
|  4 | Gorilla  |
|  5 | Wild Dog |
+----+----------+

SELECT Auto_increment FROM INFORMATION_SCHEMA.TABLES
  WHERE TABLE_NAME='ins_duplicate';
+----------------+
| Auto_increment |
+----------------+
|              6 |
+----------------+
```

Referring to column values from the INSERT portion of the statement:

```
INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6)
    ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);
```

See the VALUES() function for more.

See Also

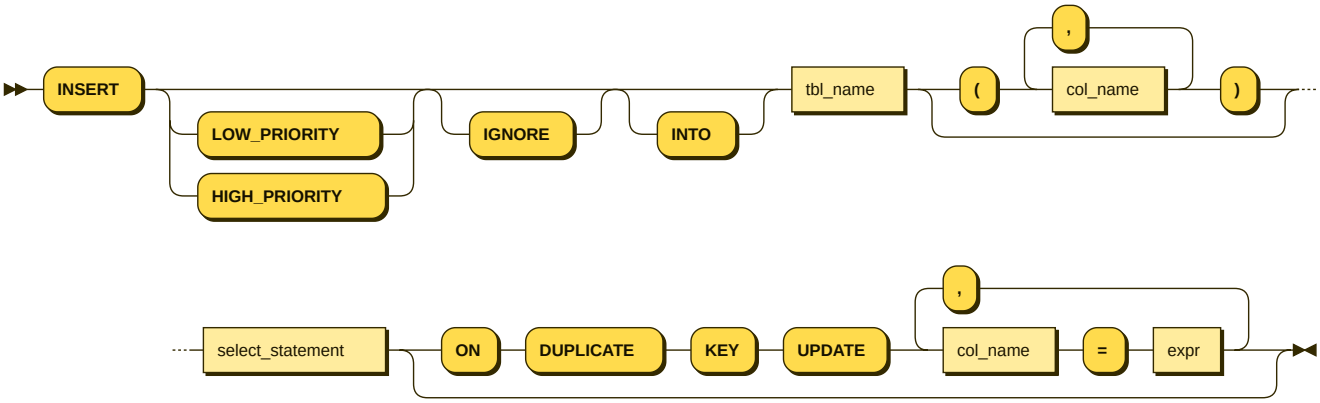
- INSERT
- INSERT DELAYED
- INSERT SELECT
- HIGH_PRIORITY and LOW_PRIORITY
- Concurrent Inserts
- INSERT - Default & Duplicate Values
- INSERT IGNORE
- VALUES()

This page is licensed: CC BY-SA / Gnu FDL

INSERT SELECT

Syntax

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```



Railroad diagram of INSERT ... SELECT — equivalent to the BNF above

Description

With `INSERT ... SELECT`, you can quickly insert many rows into a table from one or more other tables. For example:

```
INSERT INTO tbl_temp2 (fld_id)
  SELECT tbl_temp1.fld_order_id
  FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;
```

`tbl_name` can also be specified in the form `db_name.tbl_name` (see Identifier Qualifiers). This allows to copy rows between different databases.

If the new table has a primary key or UNIQUE indexes, you can use IGNORE to handle duplicate key errors during the query. The newer values will not be inserted if an identical value already exists.

REPLACE can be used instead of `INSERT` to prevent duplicates on `UNIQUE` indexes by deleting old values. In that case, `ON DUPLICATE KEY UPDATE` cannot be used.

`INSERT ... SELECT` works for tables which already exist. To create a table for a given resultset, you can use CREATE TABLE ... SELECT.

See Also

- INSERT
- INSERT DELAYED
- HIGH_PRIORITY and LOW_PRIORITY
- Concurrent Inserts
- INSERT - Default & Duplicate Values
- INSERT IGNORE
- INSERT ON DUPLICATE KEY UPDATE

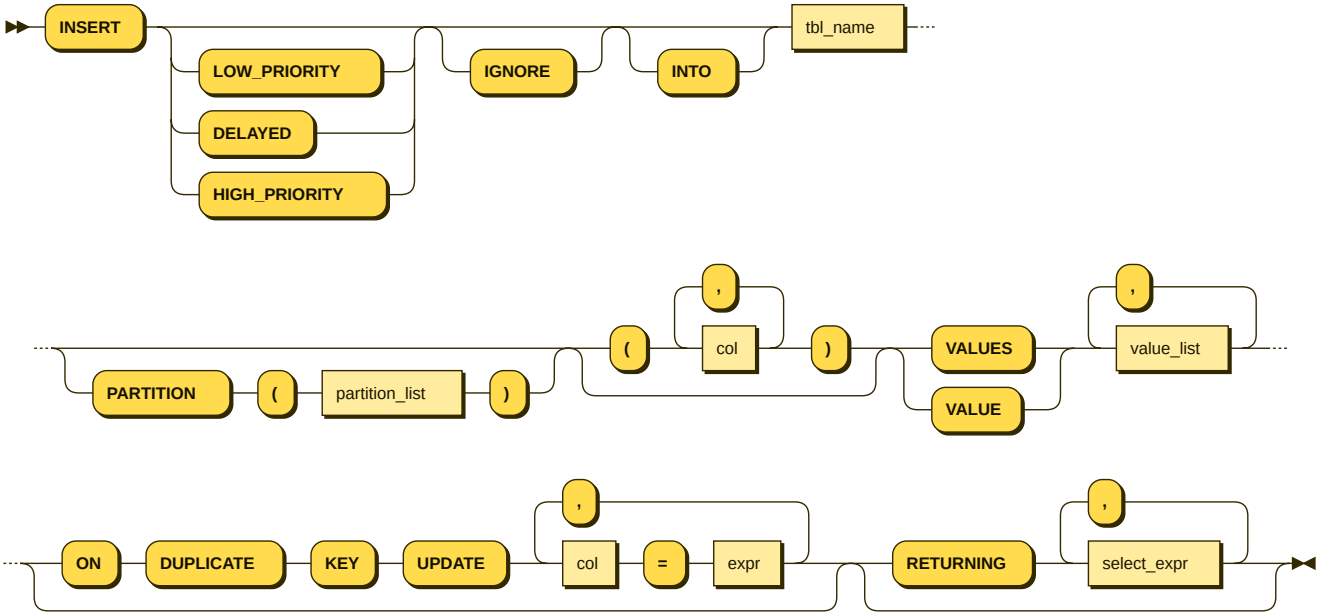
This page is licensed: GPLv2, originally from fill_help_tables.sql

INSERT…RETURNING

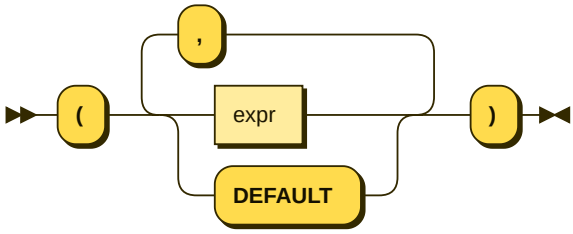
> **Note:**
>
> INSERT … RETURNING was added in MariaDB 10.5.0, and returns a result set of the inserted rows.

Syntax

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
 [ ON DUPLICATE KEY UPDATE
   col=expr
     [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```



Railroad diagram of INSERT (with RETURNING and ON DUPLICATE KEY UPDATE branches) — equivalent to the BNF above



Railroad diagram of value_list

Or:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)]
    SET col={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Or:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Description

`INSERT ... RETURNING` returns a resultset of the inserted rows.

It returns the listed columns for all the rows that are inserted, or alternatively, the specified `SELECT` expression. Any SQL expressions which can be calculated can be used in the select expression for the `RETURNING` clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

Examples

Simple INSERT statements:

```
CREATE OR REPLACE TABLE t2 (id INT, animal VARCHAR(20), t TIMESTAMP);

INSERT INTO t2 (id) VALUES (2),(3) RETURNING id,t;
+------+---------------------+
| id   | t                   |
+------+---------------------+
|    2 | 2021-04-28 00:59:32 |
|    3 | 2021-04-28 00:59:32 |
+------+---------------------+
```

```
INSERT INTO t2(id,animal) VALUES (1,'Dog'),(2,'Lion'),(3,'Tiger'),(4,'Leopard')
  RETURNING id,id+id,id&id,id||id;
+------+-------+-------+--------+
| id   | id+id | id&id | id||id |
+------+-------+-------+--------+
|    1 |     2 |     1 |      1 |
|    2 |     4 |     2 |      1 |
|    3 |     6 |     3 |      1 |
|    4 |     8 |     4 |      1 |
+------+-------+-------+--------+
```

Using stored functions in `RETURNING`:

```
DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
    BEGIN
       RETURN (SELECT arg+arg);
    END|

DELIMITER ;

PREPARE stmt FROM "INSERT INTO t1 SET id1=1, animal1='Bear' RETURNING f(id1), UPPER(animal1)";

EXECUTE stmt;
+---------+----------------+
| f(id1)  | UPPER(animal1) |
+---------+----------------+
|       2 | BEAR           |
+---------+----------------+
```

Subqueries in the `RETURNING` clause that return more than one row or column cannot be used.

Aggregate functions cannot be used in the `RETURNING` clause. Since aggregate functions work on a set of values, and if the purpose is to get the row count, `ROW_COUNT()` with `SELECT` can be used or it can be used in `INSERT...SELECT...RETURNING` if the table in the `RETURNING` clause is not the same as the `INSERT` table.

See Also

- INSERT
- REPLACE … RETURNING
- DELETE … RETURNING
- Returning clause (video)

This page is licensed: CC BY-SA / Gnu FDL

LOAD Data into Tables or Index

See LOAD DATA INFILE.

Import rows into a table at high speed from a text file, with control over field and line formatting, column mapping, and duplicate handling.

See LOAD INDEX.

Preload table indexes into the key cache. This command, used for MyISAM tables, loads index blocks into memory to warm up the cache and improve subsequent query performance.

See LOAD XML.

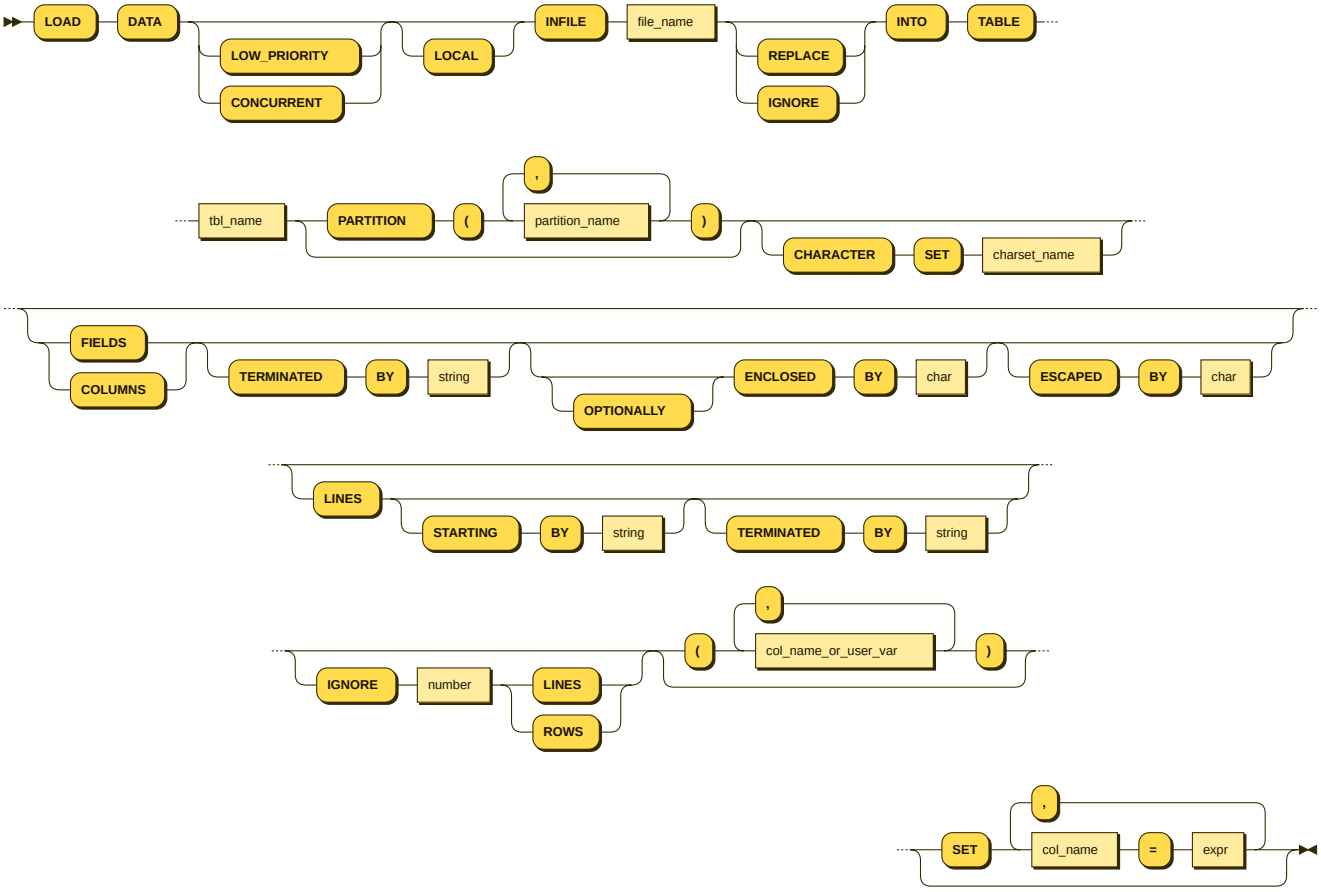
Read data from an XML file into a table. This command parses XML content, mapping elements and attributes to table columns for direct data import.

This page is licensed: CC BY-SA / Gnu FDL

LOAD DATA INFILE

Syntax

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name, ...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'CHAR']
        [ESCAPED BY 'CHAR']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES|ROWS}]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```



Railroad diagram of LOAD DATA INFILE — equivalent to the BNF above

Description

`LOAD DATA INFILE` is unsafe for statement-based replication.

Reads rows from a text file into the designated table on the database at a very high speed. The file name must be given as a literal string.

Files are written to disk using the SELECT INTO OUTFILE statement. You can then read the files back into a table using the `LOAD DATA INFILE` statement. The `FIELDS` and `LINES` clauses are the same in both statements and by default fields are expected to be terminated with tabs () and lines with newlines (). These clauses are optional, but if both are specified then the `FIELDS` clause must precede `LINES`.

Executing this statement activates `INSERT` triggers.

One must have the FILE privilege to be able to execute LOAD DATA INFILE. This is to ensure normal users cannot read system files. LOAD DATA LOCAL INFILE does not have this requirement.

If the secure_file_priv system variable is set (by default it is not), the loaded file must be present in the specified directory.

Note that MariaDB's systemd unit file restricts access to `/home`, `/root`, and `/run/user` by default. See Configuring access to home directories.

LOAD DATA LOCAL INFILE

When you execute the `LOAD DATA INFILE` statement, MariaDB Server attempts to read the input file from its own file system. By contrast, when you execute the `LOAD DATA LOCAL INFILE` statement, the client attempts to read the input file from its file system, and it sends the contents of the input file to the MariaDB Server. This allows you to load files from the client's local file system into the database.

If you don't want to permit this operation (perhaps for security reasons), you can disable the `LOAD DATA LOCAL INFILE` statement on either the server or the client.

- The `LOAD DATA LOCAL INFILE` statement can be disabled on the server by setting the local_infile system variable to `0`.
- The `LOAD DATA LOCAL INFILE` statement can be disabled on the client. If you are using MariaDB Connector/C, this can be done by unsetting the `CLIENT_LOCAL_FILES` capability flag with the mysql_real_connect function or by unsetting the `MYSQL_OPT_LOCAL_INFILE` option with mysql_optionsv function. If you are using a different client or client library, then see the documentation for your specific client or client library to determine how it handles the `LOAD DATA LOCAL INFILE` statement.
- The `LOAD DATA LOCAL INFILE` strict modes like `STRICT_TRANS_TABLES` are disabled with keyword "local". (MDEV-11235)

If the `LOAD DATA LOCAL INFILE` statement is disabled by either the server or the client and if the user attempts to execute it, then the server will cause the statement to fail with the following error message:

```
The used command is not allowed with this MariaDB version
```

Note that it is not entirely accurate to say that the MariaDB version does not support the command. It would be more accurate to say that the MariaDB configuration does not support the command. See MDEV-20500 for more information.

From MariaDB 10.5.2, the error message is more accurate:

```
The used command is not allowed because the MariaDB server or client
  has disabled the local infile capability
```

REPLACE and IGNORE

If you load data from a file into a table that already contains data and has a primary key, you may encounter issues where the statement attempts to insert a row with a primary key that already exists. When this happens, the statement fails with Error 1062, protecting the data already on the table. If you want MariaDB to overwrite duplicates, use the `REPLACE` keyword.

The `REPLACE` keyword works like the REPLACE statement. Here, the statement attempts to load the data from the file. If the row does not exist, it adds it to the table. If the row contains an existing primary key, it replaces the table data. That is, in the event of a conflict, it assumes the file contains the desired row.

This operation can cause a degradation in load speed by a factor of 20 or more if the part that has already been loaded is larger than the capacity of the InnoDB Buffer Pool. This happens because it causes a lot of turnaround in the buffer pool.

Use the `IGNORE` keyword when you want to skip any rows that contain a conflicting primary key. Here, the statement attempts to load the data from the file. If the row does not exist, it adds it to the table. If the row contains an existing primary key, it ignores the addition request and moves on to the next. That is, in the event of a conflict, it assumes the table contains the desired row.

`IGNORE number {LINES|ROWS}`

The `IGNORE number LINES` syntax can be used to ignore a number of rows from the beginning of the file. Most often this is needed when the file starts with one row that includes the column headings.

Character Sets

When the statement opens the file, it attempts to read the contents using the default character-set, as defined by the character_set_database system variable.

In the cases where the file was written using a character-set other than the default, you can specify the character-set to use with the `CHARACTER SET` clause in the statement. It ignores character-sets specified by the SET NAMES statement and by the character_set_client system variable. Setting the `CHARACTER SET` clause to a value of `binary` indicates "no conversion."

The statement interprets all fields in the file as having the same character-set, regardless of the column data type. To properly interpret file contents, you must ensure that it was written with the correct character-set. If you write a data file with mariadb-dump -T or with the SELECT INTO OUTFILE statement with the mariadb client, be sure to use the `--default-character-set` option, so that the output is written with the desired character-set.

When using mixed character sets, use the `CHARACTER SET` clause in both SELECT INTO OUTFILE and `LOAD DATA INFILE` to ensure that MariaDB correctly interprets the escape sequences.

The character_set_filesystem system variable controls the interpretation of the filename.
It is currently not possible to load data files that use the `ucs2` character set.

Preprocessing Inputs

col_name_or_user_var can be a column name, or a user variable. In the case of a variable, the SET statement can be used to preprocess the value before loading into the table.

Priority and Concurrency

In storage engines that perform table-level locking (MyISAM, MEMORY and MERGE), using the LOW_PRIORITY keyword, MariaDB delays insertions until no other clients are reading from the table. Alternatively, when using the MyISAM storage engine, you can use the CONCURRENT keyword to perform concurrent insertion.

The `LOW_PRIORITY` and `CONCURRENT` keywords are mutually exclusive. They cannot be used in the same statement.

Progress Reporting

The `LOAD DATA INFILE` statement supports progress reporting. You may find this useful when dealing with long-running operations. Using another client you can issue a SHOW PROCESSLIST query to check the progress of the data load.

Using mariadb-import

> **CURRENT**
>
> MariaDB ships with a separate utility for loading data from files: mariadb-import. It operates by sending `LOAD DATA INFILE` statements to the server.
>
> **< 10.5**
>
> MariaDB ships with a separate utility for loading data from files: `mysqlimport` . It operates by sending `LOAD DATA INFILE` statements to the server.

Using mariadb-import you can compress the file using the `--compress` option, to get better performance over slow networks, providing both the client and server support the compressed protocol. Use the `--local` option to load from the local file system.

Indexing

In cases where the storage engine supports ALTER TABLE… DISABLE KEYS statements (MyISAM and Aria), the `LOAD DATA INFILE` statement automatically disables indexes during the execution.

Examples

You have a file with this content (note the separator is ‘,’, not tab, which is the default):

```
2,2
3,3
4,4
5,5
6,8
```

```
CREATE TABLE t1 (a INT, b INT, c INT, PRIMARY KEY (a));
LOAD DATA LOCAL INFILE
 '/tmp/loaddata7.dat' INTO TABLE t1 FIELDS TERMINATED BY ',' (a,b) SET c=a+b;
SELECT * FROM t1;
+------+------+------+
| a    | b    | c    |
+------+------+------+
|    2 |    2 |    4 |
|    3 |    3 |    6 |
|    4 |    4 |    8 |
|    5 |    5 |   10 |
|    6 |    8 |   14 |
+------+------+------+
```

Another example, given the following data (the separator is a tab):

```
1	a
2	b
```

The value of the first column is doubled before loading:

```
LOAD DATA INFILE 'ld.txt' INTO TABLE ld (@i,v) SET i=@i*2;

SELECT * FROM ld;
+------+------+
| i    | v    |
+------+------+
|    2 | a    |
|    4 | b    |
+------+------+
```

See Also

- How to quickly insert data into MariaDB
- Character Sets and Collations
- SELECT … INTO OUTFILE
- mariadb-import

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOAD INDEX

Syntax

```
LOAD INDEX INTO CACHE
  tbl_index_list [, tbl_index_list] ...

tbl_index_list:
  tbl_name
    [[INDEX|KEY] (index_name[, index_name] ...)]
    [IGNORE LEAVES]
```

Description

The `LOAD INDEX INTO CACHE` statement preloads a table index into the key cache to which it has been assigned by an explicit CACHE INDEX statement, or into the default key cache otherwise. `LOAD INDEX INTO CACHE` is used only for MyISAM or Aria tables.

The `IGNORE LEAVES` modifier causes only blocks for the nonleaf nodes of the index to be preloaded.

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOAD XML

Syntax

```
LOAD XML [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE [db_name.]tbl_name
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<tagname>']
    [IGNORE number {LINES | ROWS}]
    [(column_or_user_var,...)]
    [SET col_name = expr,...]
```

Description

The LOAD XML statement reads data from an XML file into a table. The `file_name` must be given as a literal string. The `tagname` in the optional `ROWS IDENTIFIED BY` clause must also be given as a literal string, and must be surrounded by angle brackets (< and >).

LOAD XML acts as the complement of running the mariadb client in XML output mode (that is, starting the client with the –xml option). To write data from a table to an XML file, use a command such as the following one from the system shell:

```
shell> mariadb --xml -e 'SELECT * FROM mytable' > file.xml
```

To read the file back into a table, use LOAD XML INFILE. By default, the element is considered to be the equivalent of a database table row; this can be changed using the ROWS IDENTIFIED BY clause.

This statement supports three different XML formats:

- Column names as attributes and column values as attribute values:

```
<row column1="value1" column2="value2" .../>
```

- Column names as tags and column values as the content of these tags:

```
<row>
  <column1>value1</column1>
  <column2>value2</column2>
</row>
```

- Column names are the name attributes of tags, and values are the contents of these tags:

```
<row>
  <field name='column1'>value1</field>
  <field name='column2'>value2</field>
</row>
```

This is the format used by other tools, such as mariadb-dump.

All 3 formats can be used in the same XML file; the import routine automatically detects the format for each row and interprets it correctly. Tags are matched based on the tag or attribute name and the column name.

The following clauses work essentially the same way for LOAD XML as they do for LOAD DATA:

- `LOW_PRIORITY` or `CONCURRENT`
- `LOCAL`
- `REPLACE` or `IGNORE`
- `CHARACTER SET`
- (column_or_user_var,…)
- `SET`

See LOAD DATA for more information about these clauses.

The `IGNORE` number `LINES` or `IGNORE` number `ROWS` clause causes the first number rows in the XML file to be skipped. It is analogous to the `LOAD DATA` statement's `IGNORE ... LINES` clause.

If the LOW_PRIORITY keyword is used, insertions are delayed until no other clients are reading from the table. The `CONCURRENT` keyword allows the use of concurrent inserts. These clauses cannot be specified together.

This statement activates `INSERT` triggers.

See Also

- The CONNECT storage engine has an XML table type.

This page is licensed: CC BY-SA / Gnu FDL

Selecting Data

- DUAL
- FOR UPDATE
- GROUP BY
- LIMIT
- LOCK IN SHARE MODE
- Optimizer Hints
- ORDER BY
- PROCEDURE
- SELECT
- SELECT INTO DUMPFILE
- SELECT INTO OUTFILE
- SELECT … OFFSET … FETCH
- SELECT WITH ROLLUP
- Common Table Expressions (CTE)
- Joins
- Set Operations
- Subqueries

This page is licensed: CC BY-SA / Gnu FDL

DUAL

Description

You can use `DUAL` instead of a table name in situations where no tables are referenced, such as the following SELECT statement:

```
SELECT 1 + 1 FROM DUAL;
+-------+
| 1 + 1 |
+-------+
|     2 |
+-------+
```

`DUAL` is purely for the convenience of people who require that all `SELECT` statements should have `FROM` and possibly other clauses. MariaDB ignores the clauses. MariaDB does not require `FROM DUAL` if no tables are referenced.

FROM DUAL could be used when you only SELECT computed values, but require a `WHERE` clause, perhaps to test that a script correctly handles empty resultsets:

```
SELECT 1 FROM DUAL WHERE FALSE;
Empty set (0.00 sec)
```

See Also

- SELECT

This page is licensed: GPLv2, originally from fill_help_tables.sql

FOR UPDATE

InnoDB supports row-level locking. Selected rows can be locked using LOCK IN SHARE MODE or FOR UPDATE. In both cases, a lock is acquired on the rows read by the query, and it will be released when the current transaction is committed.

The `FOR UPDATE` clause of SELECT applies only when autocommit is set to 0 or the `SELECT` is enclosed in a transaction. A lock is acquired on the rows, and other transactions are prevented from writing the rows, acquire locks, and from reading them (unless their isolation level is READ UNCOMMITTED).

If `autocommit` is set to 1, the LOCK IN SHARE MODE and `FOR UPDATE` clauses have no effect in InnoDB. For non-transactional storage engines like MyISAM and ARIA, a table level lock will be taken even if autocommit is set to 1.

If the isolation level is set to SERIALIZABLE, all plain `SELECT` statements are converted to `SELECT ... LOCK IN SHARE MODE`, provided `AUTOCOMMIT` is disabled (or the statement runs inside an explicit transaction). With `AUTOCOMMIT` enabled, plain `SELECT` statements run as consistent reads.

Example

```
SELECT * FROM trans WHERE period=2001 FOR UPDATE;
```

See Also

- SELECT
- LOCK IN SHARE MODE
- InnoDB Lock Modes

This page is licensed: CC BY-SA / Gnu FDL

GROUP BY

Use the `GROUP BY` clause in a SELECT statement to group rows together that have the same value in one or more column, or the same computed value using expressions with any functions and operators except grouping functions. When you use a `GROUP BY` clause, you will get a single result row for each group of rows that have the same value for the expression given in `GROUP BY`.

When grouping rows, grouping values are compared as if by the = operator. For string values, the `=` operator ignores trailing whitespace and may normalize characters and ignore case, depending on the collation in use.

You can use any of the grouping functions in your select expression. Their values will be calculated based on all the rows that have been grouped together for each result row. If you select a non-grouped column or a value computed from a non-grouped column, it is undefined which row the returned value is taken from. This is not permitted if the `ONLY_FULL_GROUP_BY` SQL_MODE is used.

You can use multiple expressions in the `GROUP BY` clause, separated by commas.
Rows are grouped together if they match on each of the expressions.

You can also use a single integer as the grouping expression. If you use an integer *n*, the results will be grouped by the _n_th column in the select expression.

The `WHERE` clause is applied before the `GROUP BY` clause. It filters non-aggregated rows before the rows are grouped together. To filter grouped rows based on aggregate values, use the `HAVING` clause. The `HAVING` clause takes any expression and evaluates it as a boolean, just like the `WHERE` clause. You can use grouping functions in the `HAVING` clause. As with the select expression, if you reference non-grouped columns in the `HAVING` clause, the behavior is undefined.

By default, if a `GROUP BY` clause is present, the rows in the output will be sorted by the expressions used in the `GROUP BY`. You can also specify `ASC` or `DESC` (ascending, descending) after those expressions, like in ORDER BY. The default is `ASC`.

If you want the rows to be sorted by another field, you can add an explicit ORDER BY. If you don't want the result to be ordered, you can add ORDER BY NULL.

WITH ROLLUP

The `WITH ROLLUP` modifier adds extra rows to the result set that represent super-aggregate summaries. For a full description with examples, see SELECT WITH ROLLUP.

GROUP BY Examples

Consider the following table that records how many times each user has played and won a game:

```
CREATE TABLE plays (name VARCHAR(16), plays INT, wins INT);
INSERT INTO plays VALUES
  ("John", 20, 5),
  ("Robert", 22, 8),
  ("Wanda", 32, 8),
  ("Susan", 17, 3);
```

Get a list of win counts along with a count:

```
SELECT wins, COUNT(*) FROM plays GROUP BY wins;
+------+----------+
| wins | COUNT(*) |
+------+----------+
|    3 |        1 |
|    5 |        1 |
|    8 |        2 |
+------+----------+
3 rows in set (0.00 sec)
```

The `GROUP BY` expression can be a computed value, and can refer back to an identifier specified with `AS`. Get a list of win averages along with a count:

```
SELECT (wins / plays) AS winavg, COUNT(*) FROM plays GROUP BY winavg;
+--------+----------+
| winavg | COUNT(*) |
+--------+----------+
| 0.1765 |        1 |
| 0.2500 |        2 |
| 0.3636 |        1 |
+--------+----------+
3 rows in set (0.00 sec)
```

You can use any grouping function in the select expression. For each win average as above, get a list of the average play count taken to get that average:

```
SELECT (wins / plays) AS winavg, AVG(plays) FROM plays
  GROUP BY winavg;
+--------+------------+
| winavg | AVG(plays) |
+--------+------------+
| 0.1765 |    17.0000 |
| 0.2500 |    26.0000 |
| 0.3636 |    22.0000 |
+--------+------------+
3 rows in set (0.00 sec)
```

You can filter on aggregate information using the `HAVING` clause. The `HAVING` clause is applied after `GROUP BY` and allows you to filter on aggregate data that is not available to the `WHERE` clause. Restrict the above example to results that involve an average number of plays over 20:

```
SELECT (wins / plays) AS winavg, AVG(plays) FROM plays
  GROUP BY winavg HAVING AVG(plays) > 20;
+--------+------------+
| winavg | AVG(plays) |
+--------+------------+
| 0.2500 |    26.0000 |
| 0.3636 |    22.0000 |
+--------+------------+
2 rows in set (0.00 sec)
```

See Also

- SELECT
- Joins and Subqueries
- LIMIT
- ORDER BY
- Common Table Expressions
- SELECT WITH ROLLUP
- SELECT INTO OUTFILE

- SELECT INTO DUMPFILE
- FOR UPDATE
- LOCK IN SHARE MODE
- Optimizer Hints

This page is licensed: CC BY-SA / Gnu FDL

LIMIT

Description

Use the `LIMIT` clause to restrict the number of returned rows. When you use a single integer *n* with `LIMIT`, the first *n* rows will be returned. Use the ORDER BY clause to control which rows come first. You can also select a number of rows after an offset using either of the following:

```
LIMIT offset, row_count
LIMIT row_count OFFSET offset
```

When you provide an offset *m* with a limit *n*, the first *m* rows will be ignored, and the following *n* rows will be returned.

Executing an UPDATE with the `LIMIT` clause is not safe for replication. `LIMIT 0` is an exception to this rule.

There is a LIMIT ROWS EXAMINED optimization which provides the means to terminate the execution of SELECT statements which examine too many rows, and thus use too many resources. See LIMIT ROWS EXAMINED.

Multi-Table Updates

It is possible to use `LIMIT` (or ORDER BY) in a multi-table UPDATE statement.

GROUP_CONCAT

> **CURRENT**
>
> It is possible to use `LIMIT` with GROUP_CONCAT().
>
> **< 10.3.3**
>
> It is **not** possible to use `LIMIT` with GROUP_CONCAT().

Examples

```
CREATE TABLE members (name VARCHAR(20));
INSERT INTO members VALUES('Jagdish'),('Kenny'),('Rokurou'),('Immaculada');

SELECT * FROM members;
+------------+
| name       |
+------------+
| Jagdish    |
| Kenny      |
| Rokurou    |
| Immaculada |
+------------+
```

Select the first two names (no ordering specified):

```
SELECT * FROM members LIMIT 2;
+---------+
| name    |
+---------+
| Jagdish |
| Kenny   |
+---------+
```

All the names in alphabetical order:

```
SELECT * FROM members ORDER BY name;
+------------+
| name       |
+------------+
| Immaculada |
| Jagdish    |
| Kenny      |
| Rokurou    |
+------------+
```

The first two names, ordered alphabetically:

```
SELECT * FROM members ORDER BY name LIMIT 2;
+------------+
| name       |
+------------+
| Immaculada |
| Jagdish    |
+------------+
```

The third name, ordered alphabetically (the first name would be offset zero, so the third is offset two):

```
SELECT * FROM members ORDER BY name LIMIT 2,1;
+-------+
| name  |
+-------+
| Kenny |
+-------+
```

`LIMIT` can be used in a multi-table update:

```
CREATE TABLE warehouse (product_id INT, qty INT);
INSERT INTO warehouse VALUES (1,100),(2,100),(3,100),(4,100);

CREATE TABLE store (product_id INT, qty INT);
INSERT INTO store VALUES (1,5),(2,5),(3,5),(4,5);

UPDATE warehouse,store SET warehouse.qty = warehouse.qty-2, store.qty = store.qty+2
  WHERE (warehouse.product_id = store.product_id AND store.product_id  >= 1)
    ORDER BY store.product_id DESC LIMIT 2;

SELECT * FROM warehouse;
+------------+------+
| product_id | qty  |
+------------+------+
|          1 |  100 |
|          2 |  100 |
|          3 |   98 |
|          4 |   98 |
+------------+------+

SELECT * FROM store;
+------------+------+
| product_id | qty  |
+------------+------+
|          1 |    5 |
|          2 |    5 |
|          3 |    7 |
|          4 |    7 |
+------------+------+
```

When using `LIMIT` with GROUP_CONCAT, you can simplify certain queries. Consider this table:

```
CREATE TABLE d (dd DATE, cc INT);

INSERT INTO d VALUES ('2017-01-01',1);
INSERT INTO d VALUES ('2017-01-02',2);
INSERT INTO d VALUES ('2017-01-04',3);
```

The following query works fine, but is rather complex:

```
SELECT SUBSTRING_INDEX(GROUP_CONCAT(CONCAT_WS(":",dd,cc)
ORDER BY cc DESC),",",1) FROM d;
+-----------------------------------------------------------------------------+
| SUBSTRING_INDEX(GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC),",",1) |
+-----------------------------------------------------------------------------+
| 2017-01-04:3                                                                |
+-----------------------------------------------------------------------------+
```

It can be simplified to this:

```
SELECT GROUP_CONCAT(CONCAT_WS(":",dd,cc)
ORDER BY cc DESC LIMIT 1) FROM d;
+-------------------------------------------------------------+
| GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC LIMIT 1) |
+-------------------------------------------------------------+
| 2017-01-04:3                                                |
+-------------------------------------------------------------+
```

See Also

- OFFSET … FETCH Like limit, but also support `WITH TIES`
- ROWNUM() function
- SELECT

- UPDATE
- DELETE
- Joins and Subqueries
- ORDER BY
- GROUP BY
- Common Table Expressions
- SELECT WITH ROLLUP
- SELECT INTO OUTFILE
- SELECT INTO DUMPFILE
- FOR UPDATE
- LOCK IN SHARE MODE
- Optimizer Hints
- SELECT … OFFSET … FETCH

This page is licensed: CC BY-SA / Gnu FDL

LOCK IN SHARE MODE

InnoDB supports row-level locking. Selected rows can be locked using `LOCK IN SHARE MODE` or FOR UPDATE. In both cases, a lock is acquired on the rows read by the query, and it will be released when the current transaction is committed.

When `LOCK IN SHARE MODE` is specified in a SELECT statement, MariaDB will wait until all transactions that have modified the rows are committed. Then, a read lock is acquired. All transactions can read the rows, but if they want to modify them, they have to wait until your transaction is committed.

InnoDB locks index records rather than logical rows. If a `SELECT` is satisfied by a secondary index (a covering index read), it just locks that index. Instead, the primary key records will be locked if a concurrent `UPDATE` modifies columns outside of that index. These locks do not overlap under `READ COMMITTED` isolation level, so the update can proceed. To ensure a block is used, the `SELECT` query must reference columns from the primary index.

For further information and an example, see InnoDB Lock Modes.

If autocommit is set to 1 (the default), the LOCK IN SHARE MODE and FOR UPDATE clauses have no effect in InnoDB. For non-transactional storage engines like MyISAM and ARIA, a table level lock will be taken even if autocommit is set to 1.

See Also

- SELECT
- FOR UPDATE
- InnoDB Lock Modes

This page is licensed: CC BY-SA / Gnu FDL

ORDER BY

Description

Use the `ORDER BY` clause to order a resultset, such as that are returned from a SELECT statement. You can specify just a column or use any expression with functions. If you are using the `GROUP BY` clause, you can use grouping functions in `ORDER BY`. Ordering is done after grouping.

You can use multiple ordering expressions, separated by commas. Rows are sorted by the first expression, then by the second expression if they have the same value for the first, and so on.

You can use the keywords `ASC` and `DESC` after each ordering expression to force that ordering to be ascending or descending, respectively. Ordering is ascending by default.

You can also use a single integer as the ordering expression. If you use an integer *n*, the results is ordered by the n^{th} column in the select expression.

When string values are compared, they are compared as if by the STRCMP function. `STRCMP` ignores trailing whitespace and may normalize characters and ignore case, depending on the collation in use.

Duplicated entries in the `ORDER BY` clause are removed.

`ORDER BY` can also be used to order the activities of a DELETE or UPDATE statement (usually with the LIMIT clause).

> **CURRENT**
>
> It is possible to use `ORDER BY` (or LIMIT) in a multi-table UPDATE statement.
>
> **< 10.3.2**
>
> It is **not** possible to use `ORDER BY` (or LIMIT) in a multi-table UPDATE statement.

> **CURRENT**
>
> MariaDB allows packed sort keys and values of non-sorted fields in the sort buffer. This can make filesort temporary files much smaller when `VARCHAR`, `CHAR` or `BLOB` columns are used, notably speeding up some `ORDER BY` sorts.
>
> **< 10.5**
>
> MariaDB does not allow packed sort keys and values of non-sorted fields in the sort buffer.

Examples

```
CREATE TABLE seq (i INT, x VARCHAR(1));
INSERT INTO seq VALUES (1,'a'), (2,'b'), (3,'b'), (4,'f'), (5,'e');

SELECT * FROM seq ORDER BY i;
+------+------+
| i    | x    |
+------+------+
|    1 | a    |
|    2 | b    |
|    3 | b    |
|    4 | f    |
|    5 | e    |
+------+------+

SELECT * FROM seq ORDER BY i DESC;
+------+------+
| i    | x    |
+------+------+
|    5 | e    |
|    4 | f    |
|    3 | b    |
|    2 | b    |
|    1 | a    |
+------+------+

SELECT * FROM seq ORDER BY x,i;
+------+------+
| i    | x    |
+------+------+
|    1 | a    |
|    2 | b    |
|    3 | b    |
|    5 | e    |
|    4 | f    |
+------+------+
```

`ORDER BY` in an UPDATE statement, in conjunction with LIMIT:

```
UPDATE seq SET x='z' WHERE x='b' ORDER BY i DESC LIMIT 1;

SELECT * FROM seq;
+------+------+
| i    | x    |
+------+------+
|    1 | a    |
|    2 | b    |
|    3 | z    |
|    4 | f    |
|    5 | e    |
+------+------+
```

`ORDER BY` can be used in a multi-table update:

```
CREATE TABLE warehouse (product_id INT, qty INT);
INSERT INTO warehouse VALUES (1,100),(2,100),(3,100),(4,100);

CREATE TABLE store (product_id INT, qty INT);
INSERT INTO store VALUES (1,5),(2,5),(3,5),(4,5);

UPDATE warehouse,store SET warehouse.qty = warehouse.qty-2, store.qty = store.qty+2
  WHERE (warehouse.product_id = store.product_id AND store.product_id  >= 1)
    ORDER BY store.product_id DESC LIMIT 2;

SELECT * FROM warehouse;
+------------+------+
| product_id | qty  |
+------------+------+
|          1 |  100 |
|          2 |  100 |
|          3 |   98 |
|          4 |   98 |
+------------+------+

SELECT * FROM store;
+------------+------+
| product_id | qty  |
+------------+------+
|          1 |    5 |
|          2 |    5 |
|          3 |    7 |
|          4 |    7 |
+------------+------+
```

See Also

- Why is ORDER BY in a FROM subquery ignored?
- SELECT
- UPDATE
- DELETE
- Improvements to ORDER BY Optimization
- Joins and Subqueries
- LIMIT
- GROUP BY
- Common Table Expressions
- SELECT WITH ROLLUP
- SELECT INTO OUTFILE
- SELECT INTO DUMPFILE
- FOR UPDATE
- LOCK IN SHARE MODE
- Optimizer Hints

This page is licensed: CC BY-SA / Gnu FDL

PROCEDURE

The `PROCEDURE` clause of SELECT passes the whole result set to a Procedure which will process it. These Procedures are not Stored Procedures, and can only be written in the C language, so it is necessary to recompile the server.

Currently, the only available procedure is ANALYSE, which examines the resultset and suggests the optimal datatypes for each column. It is defined in the `sql/sql_analyse.cc` file, and can be used as an example to create

more Procedures.

This clause cannot be used in a view definition.

See Also

- SELECT
- Stored Procedures

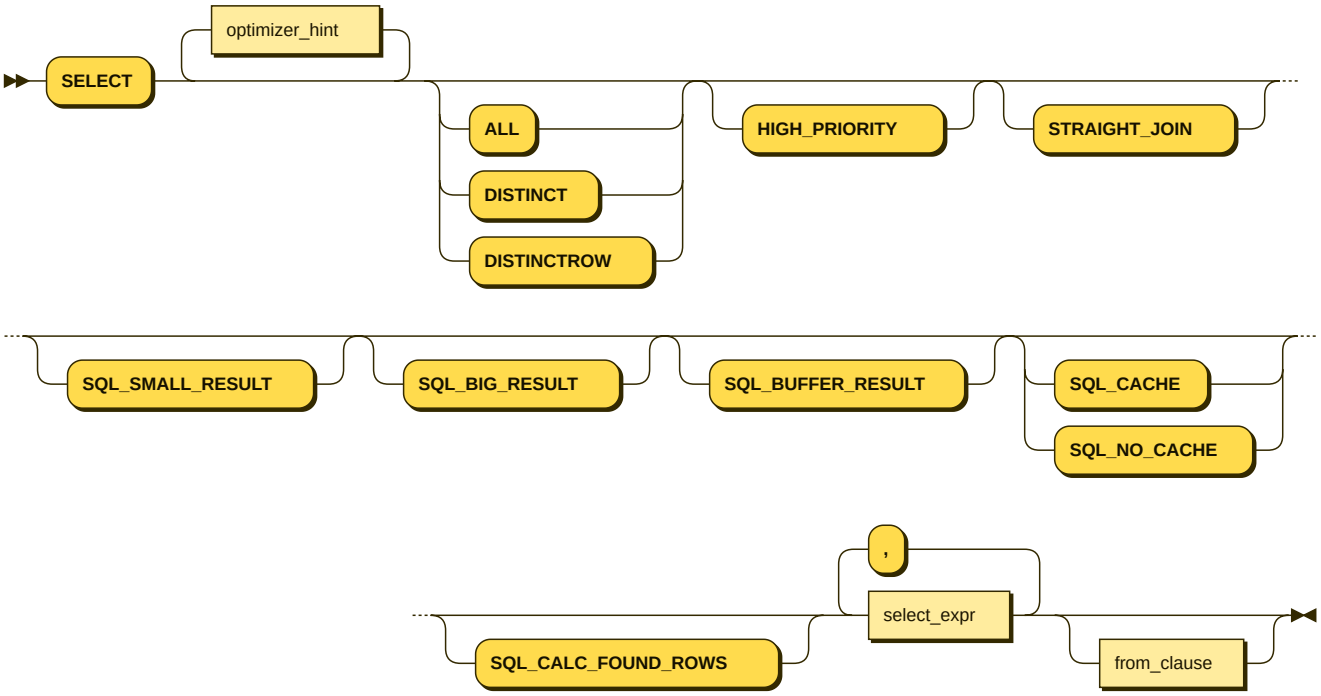
This page is licensed: CC BY-SA / Gnu FDL

SELECT

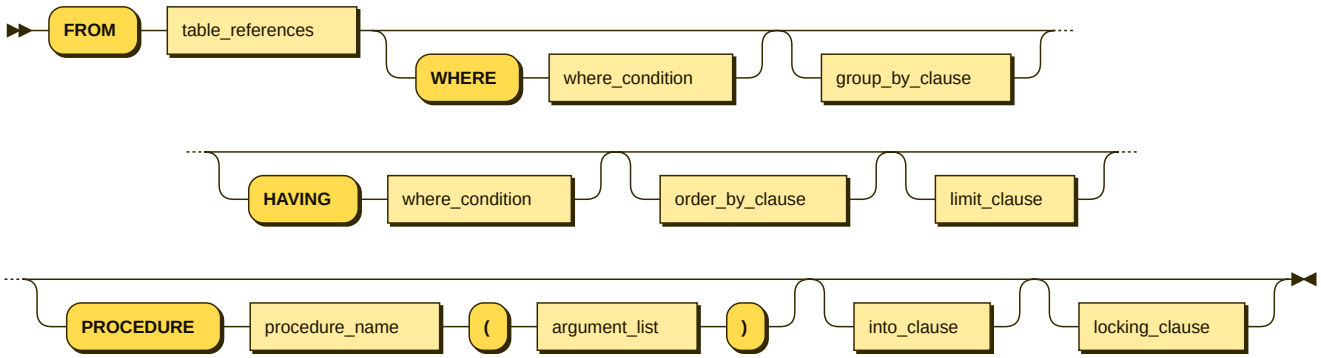
Syntax

```
SELECT
    [/*+ hints */]
    [/*+ JOIN_PREFIX(argument_list) */]
    [/*+ JOIN_ORDER(argument_list) */]
    [/*+ JOIN_FIXED_ORDER(argument_list) */]
    [/*+ JOIN_SUFFIX(argument_list) */]
    [/*+ MAX_EXECUTION_TIME(milliseconds) */]
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset
      [ROWS EXAMINED rows_limit] } |
        [OFFSET start { ROW | ROWS }]
        [FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } { ONLY | WITH TIES }] ]
      [PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options] |
        INTO DUMPFILE 'file_name' | INTO var_name [, var_name] ]
      [FOR UPDATE lock_option | LOCK IN SHARE MODE lock_option]
export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
lock_option:
    [WAIT n | NOWAIT | SKIP LOCKED]
```

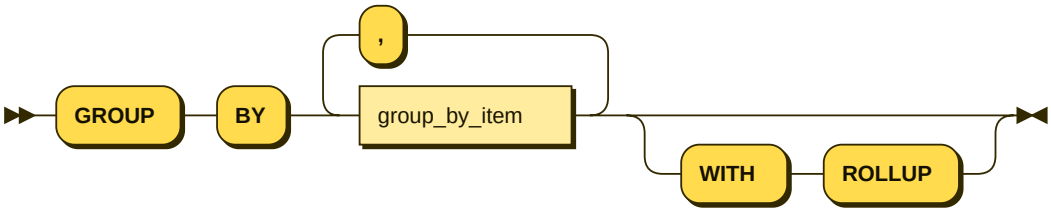
The SELECT grammar is broken out into named sub-clauses for readability. Each clause has its own diagram below.



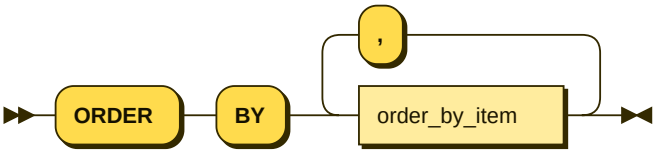
Railroad diagram of SELECT — top-level



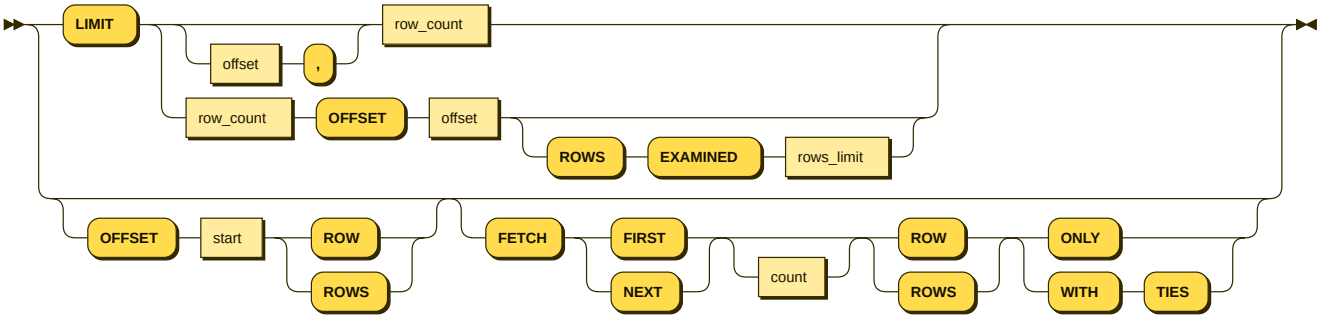
Railroad diagram of from_clause



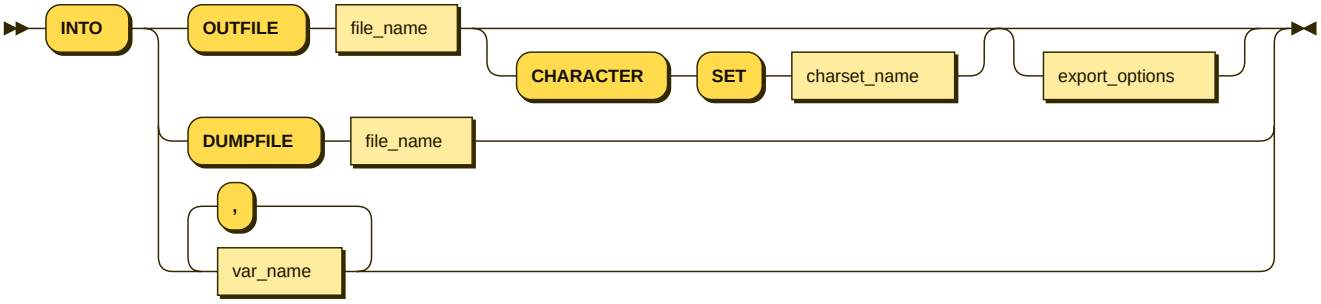
Railroad diagram of group_by_clause



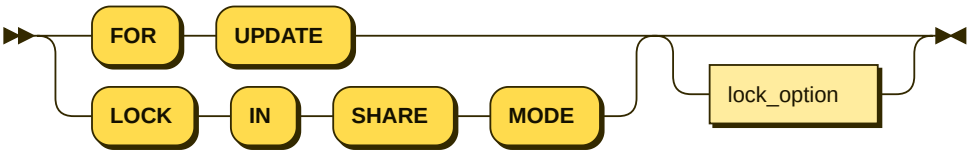
Railroad diagram of order_by_clause



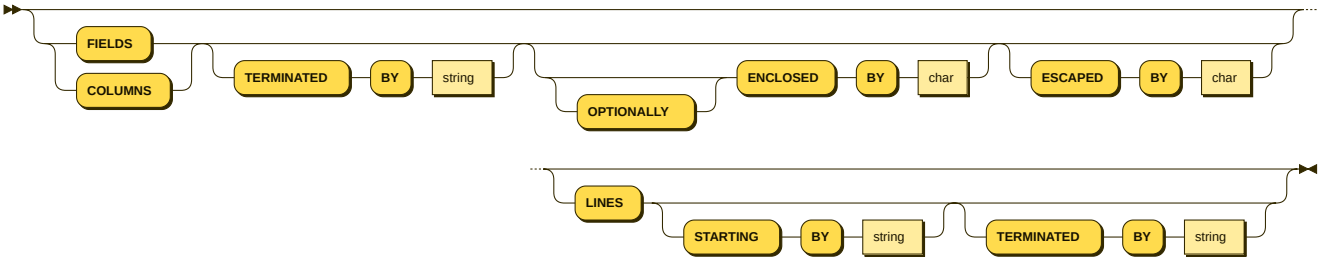
Railroad diagram of limit_clause



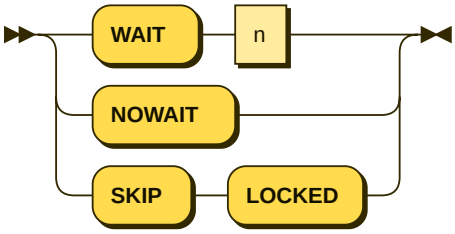
Railroad diagram of into_clause



Railroad diagram of locking_clause



Railroad diagram of export_options



Railroad diagram of lock_option

CURRENT

`[/*+ hints */]` syntax is available.

< 11.8

`[/*+ hints */]` syntax is **not** available.

CURRENT

Available join order hints can be found here.

< 12.0

Join order hints are **not** available.

CURRENT

`[/*+ MAX_EXECUTION_TIME( milliseconds ) */]` syntax is available.

The hint limits the time of statement execution to the number of milliseconds given in the hint argument.

< 12.0

`[/*+ MAX_EXECUTION_TIME( milliseconds ) */]` syntax is **not** available.

Description

`SELECT` is used to retrieve rows selected from one or more tables, and can include UNION statements and subqueries.

- Each `select_expr` expression indicates a column or data that you want to retrieve. You must have at least one select expression. See Select Expressions below.
- The `FROM` clause indicates the table or tables from which to retrieve rows. Use either a single table name or a `JOIN` expression. See JOIN for details. If no table is involved, FROM DUAL can be specified.
- Each table can also be specified as `db_name.tabl_name`. Each column can also be specified as `tbl_name.col_name` or even `db_name.tbl_name.col_name`. This allows one to write queries which involve multiple databases. See Identifier Qualifiers for syntax details.
- The `WHERE` clause, if given, indicates the condition or conditions that rows must satisfy to be selected. The `where_condition` is an expression that evaluates to true for each row to be selected. The statement selects all rows if there is no WHERE clause.
  - In the `WHERE` clause, you can use any of the functions and operators that MariaDB supports, except for aggregate (summary) functions. See Functions and Operators and Functions and Modifiers for use with GROUP BY (aggregate).
- Use the ORDER BY clause to order the results.
- Use the LIMIT clause to restrict the results to only a certain number of rows, optionally with an offset.
- Use the GROUP BY and `HAVING` clauses to group rows together when they have columns or computed values in common.

`SELECT` can also be used to retrieve rows computed without reference to any table.

Select Expressions

A `SELECT` statement must contain one or more select expressions, separated by commas. Each select expression can be one of the following:

- The name of a column.
- Any expression using functions and operators.
- `*` to select all columns from all tables in the `FROM` clause.
- `tbl_name.*` to select all columns from just the table `tbl_name`.

When specifying a column, you can either use just the column name or qualify the column name with the name of the table using `tbl_name.col_name`. The qualified form is useful if you are joining multiple tables in the `FROM` clause. If you do not qualify the column names when selecting from multiple tables, MariaDB will try to find the column in each table. It is an error if that column name exists in multiple tables.

You can quote column names using backticks. If you are qualifying column names with table names, quote each part separately as `tbl_name`.`col_name`.

If you use any grouping functions in any of the select expressions, all rows in your results will be implicitly grouped, as if you had used `GROUP BY NULL`. `GROUP BY NULL` being an expression behaves specially such that the entire result set is treated as a group.

DISTINCT

A query may produce some identical rows. By default, all rows are retrieved, even when their values are the same. To explicitly specify that you want to retrieve identical rows, use the `ALL` option. If you want duplicates to be removed from the result set, use the `DISTINCT` option. `DISTINCTROW` is a synonym for `DISTINCT`. See also COUNT DISTINCT and SELECT UNIQUE in Oracle mode.

INTO

The `INTO` clause is used to specify that the query results should be written to a file or variable.

- SELECT INTO OUTFILE - formatting and writing the result to an external file.
- SELECT INTO DUMPFILE - binary-safe writing of the unformatted results to an external file.
- SELECT INTO Variable - selecting and setting variables.

The reverse of `SELECT INTO OUTFILE` is LOAD DATA.

LIMIT

Restricts the number of returned rows. See LIMIT and LIMIT ROWS EXAMINED for details.

LOCK IN SHARE MODE/FOR UPDATE

See LOCK IN SHARE MODE and FOR UPDATE for details on the respective locking clauses.

OFFSET ... FETCH

> **CURRENT**
>
> See SELECT … OFFSET … FETCH.
>
> **< 10.6**
>
> The clause doesn't exist.

ORDER BY

Order a result set. See ORDER BY for details.

PARTITION

Specifies to the optimizer which partitions are relevant for the query. Other partitions will not be read. See Partition Pruning and Selection for details.

PROCEDURE

Passes the whole result set to a C Procedure. See PROCEDURE and PROCEDURE ANALYSE (the only built-in procedure not requiring the server to be recompiled).

SKIP LOCKED

> **CURRENT**
>
> This causes rows that couldn't be locked (LOCK IN SHARE MODE or FOR UPDATE) to be excluded from the result set. An explicit `NOWAIT` is implied here. This is only implemented on InnoDB tables and ignored otherwise.
>
> **< 10.6**
>
> The clause doesn't exist.

Optimizer Hints

These include HIGH_PRIORITY, STRAIGHT_JOIN, SQL_SMALL_RESULT | SQL_BIG_RESULT, SQL_BUFFER_RESULT, SQL_CACHE | SQL_NO_CACHE, and SQL_CALC_FOUND_ROWS.

See Optimizer Hints for details.

max_statement_time clause

By using max_statement_time in conjunction with SET STATEMENT, it is possible to limit the execution time of individual queries. For example:

```
SET STATEMENT max_statement_time=100 FOR
  SELECT field1 FROM table_name ORDER BY field1;
```

WAIT/NOWAIT

Set the lock wait timeout. See WAIT and NOWAIT.

Examples

```
SELECT f1,f2 FROM t1 WHERE (f3<=10) AND (f4='y');
```

See Getting Data from MariaDB (Beginner tutorial), or the various sub-articles, for more examples.

See Also

- Getting Data from MariaDB (Beginner tutorial)
- Joins and Subqueries
- LIMIT
- ORDER BY
- GROUP BY
- Common Table Expressions
- SELECT WITH ROLLUP
- SELECT INTO OUTFILE
- SELECT INTO DUMPFILE
- FOR UPDATE
- LOCK IN SHARE MODE
- Optimizer Hints
- Oracle mode from MariaDB 10.3

This page is licensed: GPLv2, originally from fill_help_tables.sql

SELECT INTO DUMPFILE

Syntax

```
SELECT ... INTO DUMPFILE 'file_path'
```

Description

`SELECT ... INTO DUMPFILE` is a SELECT clause which writes the resultset into a single unformatted row, without any separators, in a file. The results will not be returned to the client.

file_path can be an absolute path, or a relative path starting from the data directory. It can only be specified as a string literal, not as a variable. However, the statement can be dynamically composed and executed as a prepared statement to work around this limitation.

This statement is binary-safe and so is particularly useful for writing BLOB values to file. It can be used, for example, to copy an image or an audio document from the database to a file.

The file must not exist. It cannot be overwritten. A user needs the FILE privilege to run this statement. Also, MariaDB needs permission to write files in the specified location. If the secure_file_priv system variable is set to a non-empty directory name, the file can only be written to that directory.

> **CURRENT**
>
> The character_set_filesystem system variable has controlled interpretation of file names that are given as literal strings.
>
> **< 5.1**
>
> The character_set_filesystem system variable does not have controlled interpretation of file names that are given as literal strings.

Example

```
SELECT _utf8'Hello world!' INTO DUMPFILE '/tmp/world';

SELECT LOAD_FILE('/tmp/world') AS world;
+--------------+
| world        |
+--------------+
| Hello world! |
+--------------+
```

See Also

- SELECT
- LOAD_FILE()
- SELECT INTO Variable
- SELECT INTO OUTFILE

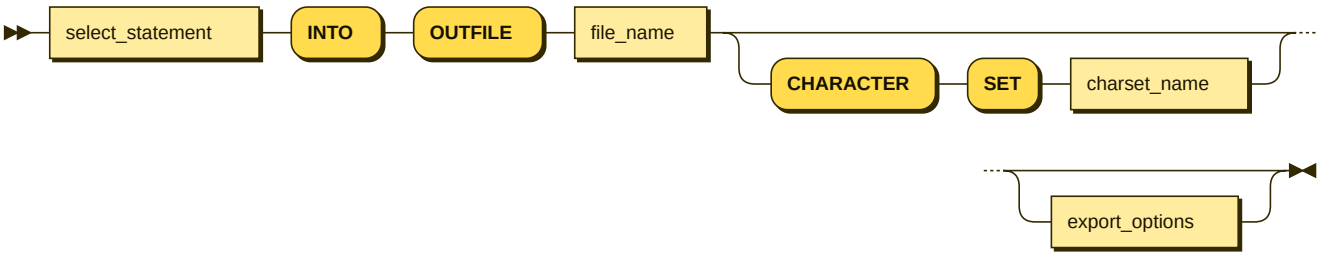
This page is licensed: CC BY-SA / Gnu FDL

SELECT INTO OUTFILE

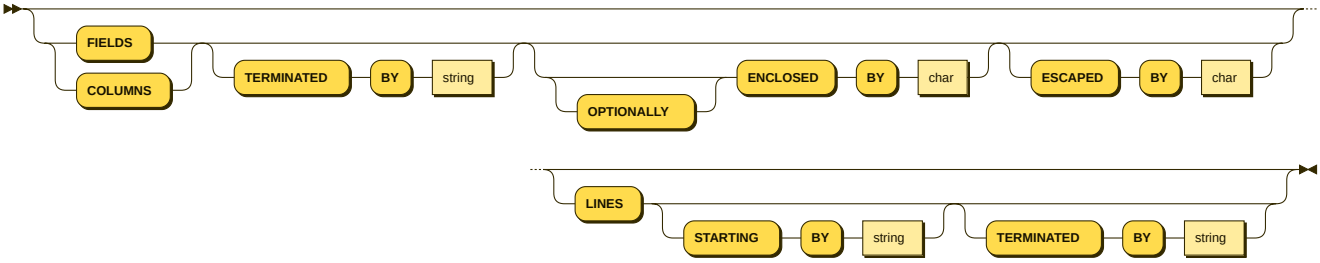
Syntax

```
SELECT ... INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        [export_options]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```



Railroad diagram of SELECT INTO OUTFILE — equivalent to the BNF above



Railroad diagram of export_options

Description

`SELECT INTO OUTFILE` writes the resulting rows to a file, and allows the use of column and row terminators to specify a particular output format. The default is to terminate fields with tabs () and lines with newlines ().

The file must not exist. It cannot be overwritten. A user needs the FILE privilege to run this statement. Also, MariaDB needs permission to write files in the specified location. If the secure_file_priv system variable is set to a non-empty directory name, the file can only be written to that directory or one of its subdirectories.

The LOAD DATA INFILE statement complements `SELECT INTO OUTFILE`.

Character Sets

The `CHARACTER SET` clause specifies the character set in which the results are to be written. Without the clause, no conversion takes place (the binary character set). In this case, if there are multiple character sets, the output will contain these too, and may not easily be able to be reloaded.

In cases where you have two servers using different character-sets, using `SELECT INTO OUTFILE` to transfer data from one to the other can have unexpected results. To ensure that MariaDB correctly interprets the escape sequences, use the `CHARACTER SET` clause on both the `SELECT INTO OUTFILE` statement and the subsequent LOAD DATA INFILE statement.

Example

The following example produces a file in the CSV format:

```
SELECT customer_id, firstname, surname FROM customer
  INTO OUTFILE '/exportdata/customers.txt'
  FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
  LINES TERMINATED BY '\n';
```

The following ANSI syntax is also supported for simple `SELECT` without `UNION` :

```
SELECT customer_id, firstname, surname INTO OUTFILE '/exportdata/customers.txt'
  FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
  LINES TERMINATED BY '\n'
  FROM customers;
```

If you want to use the ANSI syntax with `UNION` or similar construct you have to use the syntax:

```
SELECT  * INTO OUTFILE "/tmp/skr3"
FROM (SELECT * FROM t1 UNION SELECT * FROM t1);
```

See Also

- SELECT
- LOAD_FILE() function
- LOAD DATA INFILE
- SELECT INTO Variable
- SELECT INTO DUMPFILE

This page is licensed: CC BY-SA / Gnu FDL

SELECT ... OFFSET ... FETCH

> **Note:**
>
> `SELECT ... OFFSET ... FETCH` is available from MariaDB 10.6.

Syntax

```
OFFSET start { ROW | ROWS }
FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } { ONLY | WITH TIES }
```

Description

The `OFFSET` clause allows one to return only those elements of a resultset that come after a specified offset. The `FETCH` clause specifies the number of rows to return, while `ONLY` or `WITH TIES` specifies whether or not to also return any further results that tie for last place according to the ordered resultset.

Either the singular `ROW` or the plural `ROWS` can be used after the `OFFSET` and `FETCH` clauses; the choice has no impact on the results.

`FIRST` and `NEXT` give the same result.

In the case of `WITH TIES`, an ORDER BY clause is required, otherwise an error will be returned.

```
SELECT i FROM t1 FETCH FIRST 2 ROWS WITH TIES;
ERROR 4180 (HY000): FETCH ... WITH TIES requires ORDER BY clause to be present
```

Examples

Given a table with 6 rows:

```
CREATE OR REPLACE TABLE t1 (i INT);
INSERT INTO t1 VALUES (1),(2),(3),(4), (4), (5);
SELECT i FROM t1 ORDER BY i ASC;
+------+
| i    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    4 |
|    5 |
+------+
```

`OFFSET 2` allows one to skip the first two results:

```
SELECT i FROM t1 ORDER BY i ASC OFFSET 2 ROWS;
+------+
| i    |
+------+
|    3 |
|    4 |
|    4 |
|    5 |
+------+
```

`FETCH FIRST 3 ROWS ONLY` limits the results to three rows only:

```
SELECT i FROM t1 ORDER BY i ASC OFFSET 1 ROWS FETCH FIRST 3 ROWS ONLY;
+------+
| i    |
+------+
|    2 |
|    3 |
|    4 |
+------+
```

The same outcome can also be achieved with the LIMIT clause:

```
SELECT i FROM t1 ORDER BY i ASC LIMIT 3 OFFSET 1;
+------+
| i    |
+------+
|    2 |
|    3 |
|    4 |
+------+
```

`WITH TIES` ensures the tied result `4` is also returned:

```
SELECT i FROM t1 ORDER BY i ASC OFFSET 1 ROWS FETCH FIRST 3 ROWS WITH TIES;
+------+
| i    |
+------+
|    2 |
|    3 |
|    4 |
|    4 |
+------+
```

See Also

- LIMIT
- ORDER BY
- SELECT

This page is licensed: CC BY-SA / Gnu FDL

SELECT WITH ROLLUP

Syntax

See SELECT.

Description

The `WITH ROLLUP` modifier adds extra rows to the result set that represent super-aggregate summaries. The super-aggregated column is represented by a `NULL` value. Multiple aggregates over different columns will be added if there are multiple `GROUP BY` columns.

The LIMIT clause can be used at the same time, and is applied after the `WITH ROLLUP` rows have been added.

`WITH ROLLUP` cannot be used with ORDER BY. Some sorting is still possible by using `ASC` or `DESC` clauses with the `GROUP BY` column, although the super-aggregate rows will always be added last.

Examples

These examples use the following sample table

```
CREATE TABLE booksales (
  country VARCHAR(35), genre ENUM('fiction','non-fiction'), year YEAR, sales INT);

INSERT INTO booksales VALUES
  ('Senegal','fiction',2014,12234), ('Senegal','fiction',2015,15647),
  ('Senegal','non-fiction',2014,64980), ('Senegal','non-fiction',2015,78901),
  ('Paraguay','fiction',2014,87970), ('Paraguay','fiction',2015,76940),
  ('Paraguay','non-fiction',2014,8760), ('Paraguay','non-fiction',2015,9030);
```

The `WITH ROLLUP` modifier in this example adds an extra row that aggregates both years:

```
SELECT year, SUM(sales) FROM booksales GROUP BY year;
+------+------------+
| year | SUM(sales) |
+------+------------+
| 2014 |     173944 |
| 2015 |     180518 |
+------+------------+
2 rows in set (0.08 sec)

SELECT year, SUM(sales) FROM booksales GROUP BY year WITH ROLLUP;
+------+------------+
| year | SUM(sales) |
+------+------------+
| 2014 |     173944 |
| 2015 |     180518 |
| NULL |     354462 |
+------+------------+
```

Each time the genre, the year, or the country change, another super-aggregate row is added:

```
SELECT country, year, genre, SUM(sales)
  FROM booksales GROUP BY country, year, genre;
+----------+------+-------------+------------+
| country  | year | genre       | SUM(sales) |
+----------+------+-------------+------------+
| Paraguay | 2014 | fiction     |      87970 |
| Paraguay | 2014 | non-fiction |       8760 |
| Paraguay | 2015 | fiction     |      76940 |
| Paraguay | 2015 | non-fiction |       9030 |
| Senegal  | 2014 | fiction     |      12234 |
| Senegal  | 2014 | non-fiction |      64980 |
| Senegal  | 2015 | fiction     |      15647 |
| Senegal  | 2015 | non-fiction |      78901 |
+----------+------+-------------+------------+

SELECT country, year, genre, SUM(sales)
  FROM booksales GROUP BY country, year, genre WITH ROLLUP;
+----------+------+-------------+------------+
| country  | year | genre       | SUM(sales) |
+----------+------+-------------+------------+
| Paraguay | 2014 | fiction     |      87970 |
| Paraguay | 2014 | non-fiction |       8760 |
| Paraguay | 2014 | NULL        |      96730 |
| Paraguay | 2015 | fiction     |      76940 |
| Paraguay | 2015 | non-fiction |       9030 |
| Paraguay | 2015 | NULL        |      85970 |
| Paraguay | NULL | NULL        |     182700 |
| Senegal  | 2014 | fiction     |      12234 |
| Senegal  | 2014 | non-fiction |      64980 |
| Senegal  | 2014 | NULL        |      77214 |
| Senegal  | 2015 | fiction     |      15647 |
| Senegal  | 2015 | non-fiction |      78901 |
| Senegal  | 2015 | NULL        |      94548 |
| Senegal  | NULL | NULL        |     171762 |
| NULL     | NULL | NULL        |     354462 |
+----------+------+-------------+------------+
```

The `LIMIT` clause, applied after `WITH ROLLUP`:

```
SELECT country, year, genre, SUM(sales)
  FROM booksales GROUP BY country, year, genre WITH ROLLUP LIMIT 4;
+----------+------+-------------+------------+
| country  | year | genre       | SUM(sales) |
+----------+------+-------------+------------+
| Paraguay | 2014 | fiction     |      87970 |
| Paraguay | 2014 | non-fiction |       8760 |
| Paraguay | 2014 | NULL        |      96730 |
| Paraguay | 2015 | fiction     |      76940 |
+----------+------+-------------+------------+
```

Sorting by year descending:

```
SELECT country, year, genre, SUM(sales)
  FROM booksales GROUP BY country, year DESC, genre WITH ROLLUP;
+----------+------+-------------+------------+
| country  | year | genre       | SUM(sales) |
+----------+------+-------------+------------+
| Paraguay | 2015 | fiction     |      76940 |
| Paraguay | 2015 | non-fiction |       9030 |
| Paraguay | 2015 | NULL        |      85970 |
| Paraguay | 2014 | fiction     |      87970 |
| Paraguay | 2014 | non-fiction |       8760 |
| Paraguay | 2014 | NULL        |      96730 |
| Paraguay | NULL | NULL        |     182700 |
| Senegal  | 2015 | fiction     |      15647 |
| Senegal  | 2015 | non-fiction |      78901 |
| Senegal  | 2015 | NULL        |      94548 |
| Senegal  | 2014 | fiction     |      12234 |
| Senegal  | 2014 | non-fiction |      64980 |
| Senegal  | 2014 | NULL        |      77214 |
| Senegal  | NULL | NULL        |     171762 |
| NULL     | NULL | NULL        |     354462 |
+----------+------+-------------+------------+
```

See Also

- SELECT
- Joins and Subqueries
- LIMIT
- ORDER BY
- GROUP BY
- Common Table Expressions
- SELECT INTO OUTFILE
- SELECT INTO DUMPFILE
- FOR UPDATE
- LOCK IN SHARE MODE
- Optimizer Hints

This page is licensed: CC BY-SA / Gnu FDL

Common Table Expressions (CTE)

See Recursive Common Table Expressions Overview.

Process hierarchical data using recursive CTEs. These expressions reference themselves to repeatedly execute a subquery, perfect for traversing tree structures or generating sequences.

See Non-Recursive Common Table Expressions Overview.

Define simple temporary result sets. Non-recursive CTEs act like query-local views, improving readability by allowing you to define and reuse subqueries within a single statement.

See WITH.

Complete WITH clause reference: WITH [RECURSIVE] AS (SELECT…) syntax, recursive CTE support, CYCLE… RESTRICT cycle detection, and max_recursive_iterations.

This page is licensed: CC BY-SA / Gnu FDL

Recursive Common Table Expressions Overview

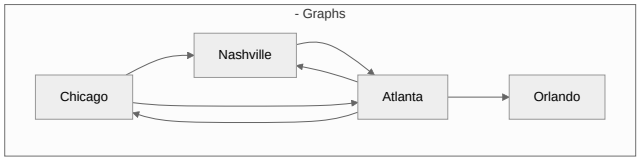
Common Table Expressions (CTEs) are a standard SQL feature, and are essentially temporary named result sets. They are defined using the `WITH` clause and can be referenced like tables within the main query. CTEs first appeared in the SQL standard in 1999, and the first implementations began appearing in 2007.

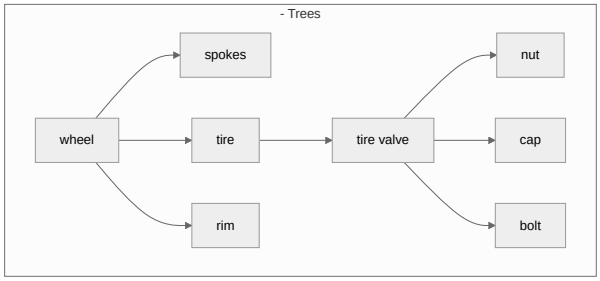
MariaDB supports two types of CTEs:

- Non-recursive;
- Recursive, which this article covers.

Recursive CTE

A recursive CTE will repeatedly execute subsets of the data until it obtains the complete result set. This makes it particularly useful for handling hierarchical or tree-structured data, such as organizational charts, category trees, or parent-child relationships. max_recursive_iterations avoids infinite loops. CTEs permit a query to reference itself. SQL is generally poor at recursive structures.



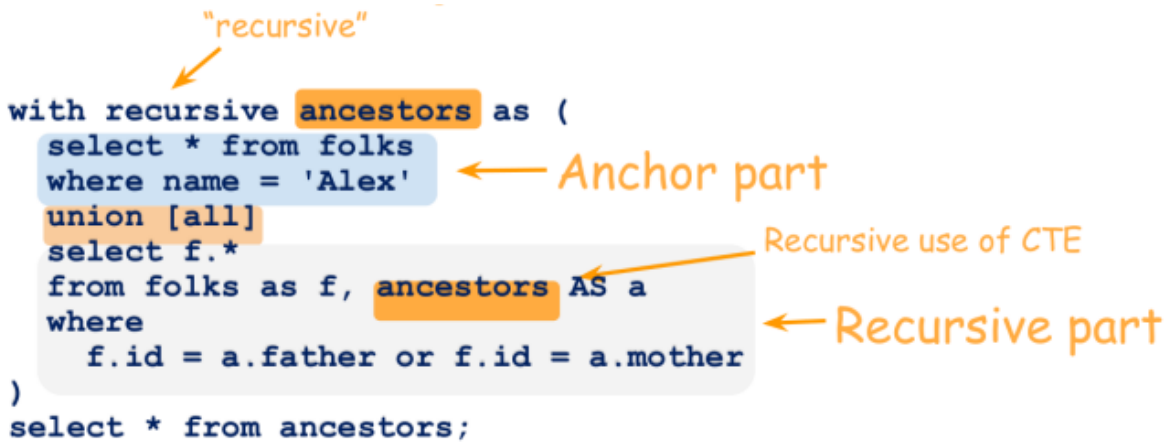


A tree has exactly one path from root to any node, while a graph can have multiple paths and cycles, such as the bidirectional Chicago-Atlanta and Nashville-Atlanta connections.

Recursive CTE Syntax

WITH RECURSIVE signifies a recursive CTE. It is given a name, followed by a body (the main query). The body consists of two parts:

- An anchor member (the initial query)
- A recursive member (which references the CTE itself).



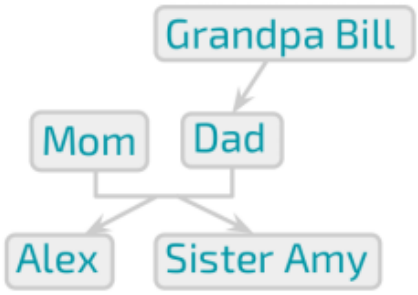
rcte_syntax

Computation

Given the following structure:

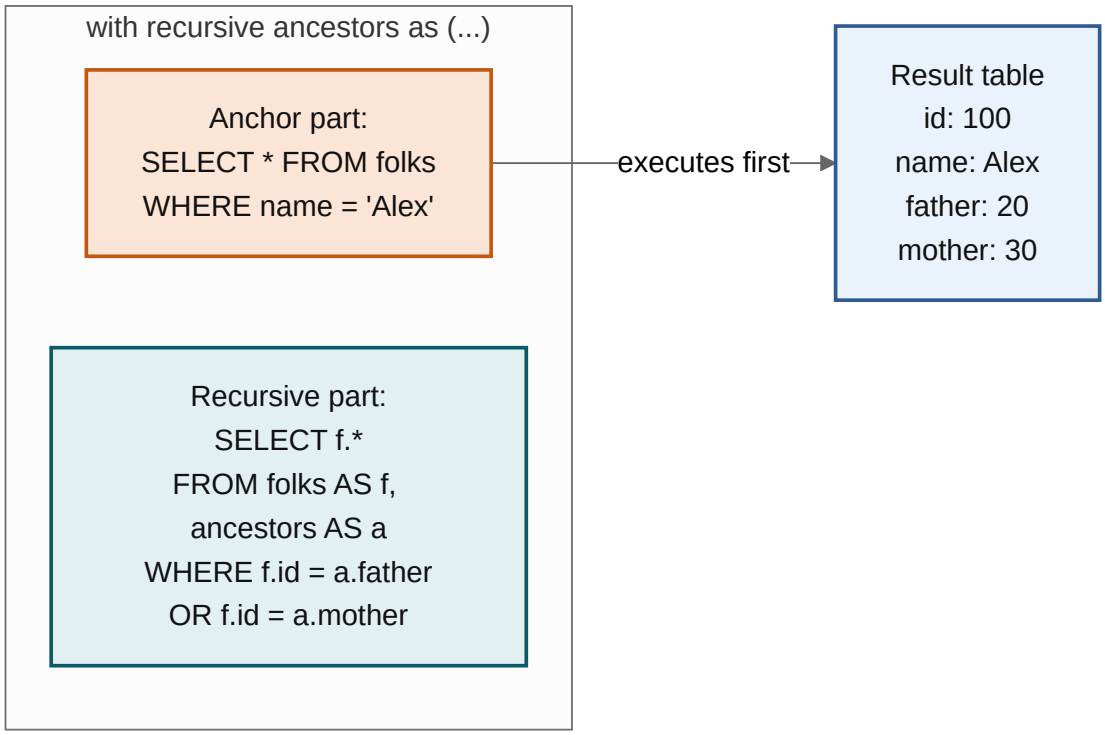
Recursive CTE computation

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL
10	Grandpa Bill	NULL	NULL
98	Sister Amy	20	30



rcte_computation

1. First execute the anchor part of the query:



Step #1: executing only the anchor part of the query returns the result table's single row — id 100, name Alex, father 20, mother 30.

2. Next, execute the recursive part of the query:

Computation

Result table

id	name	father	mother
100	Alex	20	30

```
with recursive ancestors as (
  select * from folks
  where name = 'Alex'
  union
  select f.*
  from folks as f, ancestors AS a
  where
    f.id = a.father or f.id = a.mother
)
select * from ancestors;
```

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL
10	Grandpa Bill	NULL	NULL
98	Sister Amy	20	30

Step #2:
execution of the recursive part

rcte_computation_2

Computation

Result table

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL

```
with recursive ancestors as (
  select * from folks
  where name = 'Alex'
  union
  select f.*
  from folks as f, ancestors AS a
  where
    f.id = a.father or f.id = a.mother
)
select * from ancestors;
```

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL
10	Grandpa Bill	NULL	NULL
98	Sister Amy	20	30

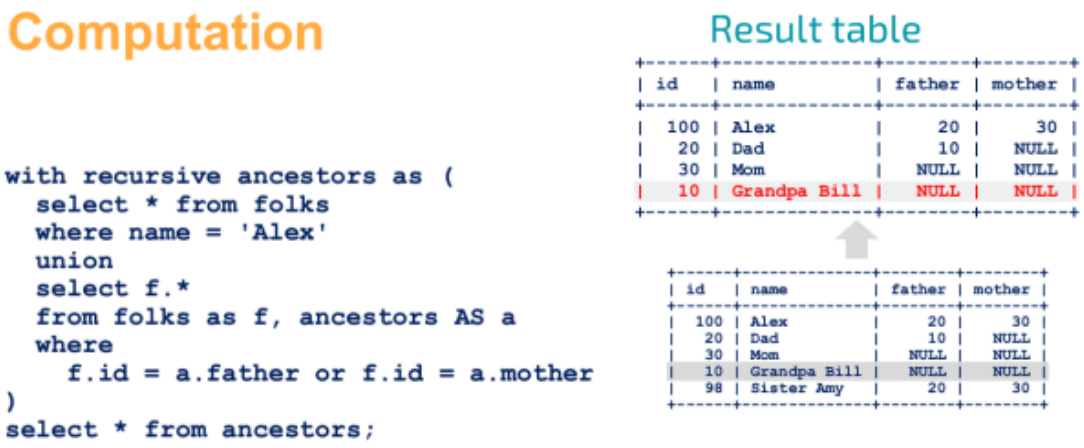
Computation

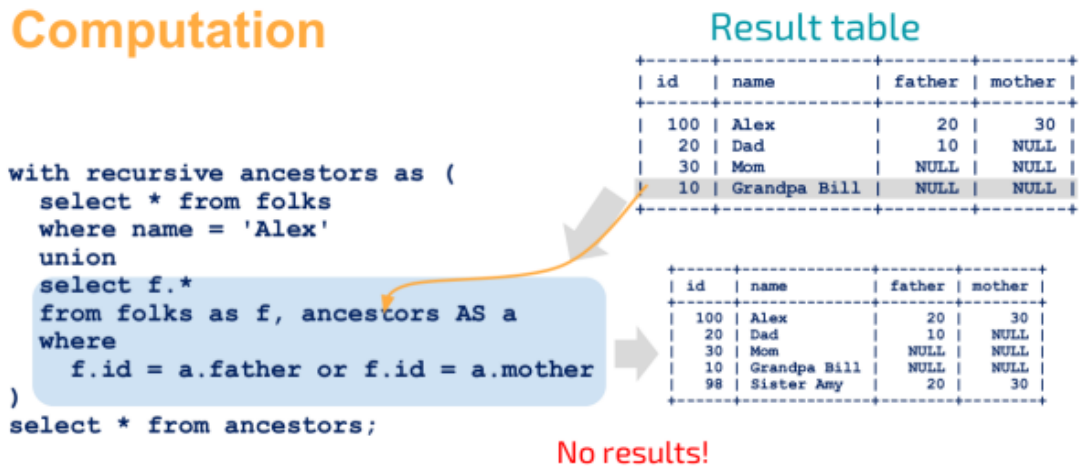
Result table

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL

```
with recursive ancestors as (
  select * from folks
  where name = 'Alex'
  union
  select f.*
  from folks as f, ancestors AS a
  where
    f.id = a.father or f.id = a.mother
)
select * from ancestors;
```

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL
10	Grandpa Bill	NULL	NULL
98	Sister Amy	20	30





Summary

```
WITH RECURSIVE R AS (
  SELECT anchor_data
  UNION [all]
  SELECT recursive_part
  FROM R, ...
)
SELECT ...
```

1. Compute anchor_data.
2. Compute recursive_part to get the new data.
3. If (new data is non-empty) go to 2.

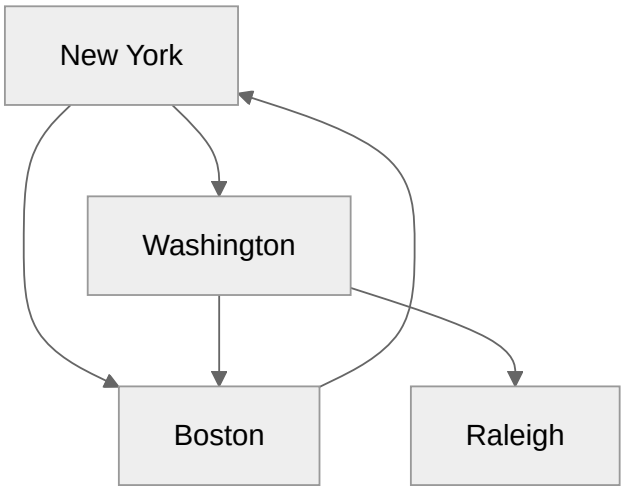
Recursive CTE Examples

Transitive closure - determining bus destinations

Sample data:

origin	dst
New York	Boston
Boston	New York
New York	Washington
Washington	Boston

origin	dst
Washington	Raleigh



The `bus_routes` table as a directed graph: New York and Boston connect both ways, while New York → Washington → Boston/Raleigh are one-way routes.

```
CREATE TABLE bus_routes (origin VARCHAR(50), dst VARCHAR(50));
INSERT INTO bus_routes VALUES
  ('New York', 'Boston'),
  ('Boston', 'New York'),
  ('New York', 'Washington'),
  ('Washington', 'Boston'),
  ('Washington', 'Raleigh');
```

Now, we want to return the bus destinations with New York as the origin:

```
WITH RECURSIVE bus_dst as (
    SELECT origin as dst FROM bus_routes WHERE origin='New York'
  UNION
    SELECT bus_routes.dst FROM bus_routes JOIN bus_dst ON bus_dst.dst= bus_routes.origin
)
SELECT * FROM bus_dst;
+------------+
| dst        |
+------------+
| New York   |
| Boston     |
| Washington |
| Raleigh    |
+------------+
```

The above example is computed as follows:

First, the anchor data is calculated:

- Starting from New York.
- Boston and Washington are added.

Next, the recursive part:

- Starting from Boston and then Washington.
- Raleigh is added.
- UNION excludes nodes that are already present.

Computing paths - determining bus routes

This time, we are trying to get bus routes such as “New York -> Washington -> Raleigh”.

Using the same sample data as the previous example:

```
WITH RECURSIVE paths (cur_path, cur_dest) AS (
    SELECT origin, origin FROM bus_routes WHERE origin='New York'
  UNION
    SELECT CONCAT(paths.cur_path, ',', bus_routes.dst), bus_routes.dst
     FROM paths
     JOIN bus_routes
       ON paths.cur_dest = bus_routes.origin AND
         NOT FIND_IN_SET(bus_routes.dst, paths.cur_path)
)
SELECT * FROM paths;
+-----------------------------+------------+
| cur_path                    | cur_dest   |
+-----------------------------+------------+
| New York                    | New York   |
| New York,Boston             | Boston     |
| New York,Washington         | Washington |
| New York,Washington,Boston  | Boston     |
| New York,Washington,Raleigh | Raleigh    |
+-----------------------------+------------+
```

Type Resolution and Data Truncation

In previous versions of MariaDB, the data type of a Recursive CTE column was entirely determined by the Anchor (non-recursive) part. If the Recursive part carried out calculations that generated a greater value, the data was silently truncated, leading to incorrect results.

Starting with the versions listed below, MariaDB prevents silent data loss. The server now validates that the data generated by the recursive part matches the column types given by the anchor part. If a mismatch that leads to truncation is detected, the query will now fail with an error. See MDEV-12325 for more information.

MariaDB Series	Fixed Version
10.3	10.3.36
10.4	10.4.26
10.5	10.5.17
10.6	10.6.9
10.7	10.7.5

Example: Managing Out of Range Errors in Recursive Part

MariaDB now detects an error when a recursive part executes a calculation (such as adding a large offset) that exceeds the anchor part’s data type capacity. To ensure that the query succeeds, CAST can be used in the anchor part to declare a column size that will allow for the expected growth of your recursive calculations.

Consider a table representing a company employee hierarchy:

```
CREATE TABLE t1
(
  id INT,     -- employee ID
  mid INT,    -- manager ID, or NULL if top level employee
  name TEXT   -- employee name
);
INSERT INTO t1 VALUES (0,NULL, 'Name');
INSERT INTO t1 VALUES (1,0,    'Name1');
INSERT INTO t1 VALUES (2,0,    'Name2');
INSERT INTO t1 VALUES (11,1,   'Name11');
INSERT INTO t1 VALUES (12,1,   'Name12');
```

The following recursive CTE attempts to generate hierarchy levels by changing the mid value in the recursive part:

```
WITH RECURSIVE
cteReports (level, id, mid, name) AS
(
  SELECT 1, id, mid, name FROM t1 WHERE mid IS NULL
  UNION ALL
  SELECT r.level + 1, e.id, e.mid + 1000000000000, e.name FROM t1 e
  INNER JOIN cteReports r ON e.mid = r.id
)
SELECT
  level, id, mid, name,
  (SELECT name FROM t1 WHERE id= cteReports.mid) AS mname
FROM cteReports
ORDER BY level, mid;
```

This query fails with an error with MariaDB versions that incorporate the fix for MDEV-12325:

```
Error 1264 (22003): Out of range value for column 'mid' in 'cteReports'
```

This happens because the anchor member selects `mid` as the standard integer, but the recursive part attempts to add `1000000000000`, which exceeds the `INT` limit.

To avoid this error, the data type must be widened (for example, to `DOUBLE` or `BIGINT`) in the anchor part, since recursive CTE column types are defined there.

```
WITH RECURSIVE
cteReports (level, id, mid, name) AS
(
  SELECT 1, id, cast(mid as double), name FROM t1 WHERE mid IS NULL
  UNION ALL
  SELECT r.level + 1, e.id, e.mid + 1000000000000, e.name FROM t1 e
   INNER JOIN cteReports r ON e.mid = r.id
)
SELECT
  level, id, mid, name,
  (SELECT name FROM t1 WHERE id= cteReports.mid) AS mname
FROM cteReports
ORDER BY level, mid;
```

By explicitly widening the data type in the anchor part, all values generated by the recursive part fit within the defined column type, and the query executes successfully.

The expected query result would be:

```
+-------+------+---------------+--------+--------+
| level | id   | mid           | name   | mname  |
+-------+------+---------------+--------+--------+
|     1 |    0 | NULL          | Name   | NULL   |
|     2 |    1 | 1000000000000 | Name1  | NAME   |
|     2 |    2 | 1000000000000 | Name2  | NAME   |
|     3 |   11 | 1000000000001 | Name11 | NAME1  |
|     3 |   12 | 1000000000001 | Name12 | NAME1  |
+-------+------+------------+--------+-----------+
```

This page is licensed: CC BY-SA / Gnu FDL

Non-Recursive Common Table Expressions Overview

Common Table Expressions (CTEs) are a standard SQL feature, and are essentially temporary named result sets. There are two kinds of CTEs: Non-Recursive, which this article covers; and Recursive.

Non-Recursive CTEs

The WITH keyword signifies a CTE. It is given a name, followed by a body (the main query).

CTEs are similar to derived tables:

```
WITH engineers AS
   ( SELECT * FROM employees
     WHERE dept = 'Engineering' )

SELECT * FROM engineers
WHERE ...
```

```
SELECT * FROM
   ( SELECT * FROM employees
     WHERE dept = 'Engineering' ) AS engineers
WHERE
...
```

A non-recursive CTE is basically a query-local VIEW. There are several advantages and caveats to them. The syntax is more readable than a nested `FROM (SELECT ...)`.
A CTE can refer to another and it can be referenced from multiple places.

A CTE referencing Another CTE

Using this format makes for a more readable SQL than a nested `FROM(SELECT ...)` clause:

```
WITH engineers AS (
SELECT * FROM employees
WHERE dept IN('Development','Support') ),
eu_engineers AS ( SELECT * FROM engineers WHERE country IN('NL',...) )
SELECT
...
FROM eu_engineers;
```

Multiple Uses of a CTE

This can be an ‘anti-self join’, for example:

```
WITH engineers AS (
SELECT * FROM employees
WHERE dept IN('Development','Support') )

SELECT * FROM engineers E1
WHERE NOT EXISTS
   (SELECT 1 FROM engineers E2
    WHERE E2.country=E1.country
    AND E2.name <> E1.name );
```

Or, for year-over-year comparisons, for example:

```
WITH sales_product_year AS (
SELECT product, YEAR(ship_date) AS year,
SUM(price) AS total_amt
FROM item_sales
GROUP BY product, year )

SELECT *
FROM sales_product_year CUR,
sales_product_year PREV,
WHERE CUR.product=PREV.product
AND  CUR.year=PREV.year + 1
AND CUR.total_amt > PREV.total_amt
```

Another use is to compare individuals against their group. Below is an example of how this might be executed:

```
WITH sales_product_year AS (
SELECT product,
YEAR(ship_date) AS year,
SUM(price) AS total_amt
FROM item_sales
GROUP BY product, year
)

SELECT *
FROM sales_product_year S1
WHERE
total_amt >
    (SELECT 0.1 * SUM(total_amt)
     FROM sales_product_year S2
     WHERE S2.year = S1.year)
```

This page is licensed: CC BY-SA / Gnu FDL

WITH

Syntax

```
WITH [RECURSIVE] table_reference [(columns_list)] AS  (
  SELECT ...
)
[CYCLE cycle_column_list RESTRICT]
SELECT ...
```

Description

The `WITH` keyword signifies a Common Table Expression (CTE). It allows you to refer to a subquery expression many times in a query, as if having a temporary table that only exists for the duration of a query.

There are two kinds of CTEs:

- Non-Recursive.

- Recursive (signified by the `RECURSIVE` keyword).

You can use `table_reference` as any normal table in the external `SELECT` part. You can also use `WITH` in subqueries, as well as with EXPLAIN and SELECT.

Poorly-formed recursive CTEs can in theory cause infinite loops. The max_recursive_iterations system variable limits the number of recursions.

CYCLE ... RESTRICT

CURRENT

The `CYCLE` clause enables CTE cycle detection, avoiding excessive or infinite loops,
MariaDB supports a relaxed, non-standard grammar.
The SQL Standard permits a `CYCLE` clause, as follows:

```
WITH RECURSIVE ... (
  ...
)
CYCLE <cycle column list>
SET <cycle mark column> TO <cycle mark value> DEFAULT <non-cycle mark value>
USING <path column>
```

where all clauses are mandatory.
MariaDB does not support this, but permits a non-standard relaxed grammar, as follows:

```
WITH RECURSIVE ... (
  ...
)
CYCLE <cycle column list> RESTRICT
```

With the use of `CYCLE ... RESTRICT` it makes no difference whether the CTE uses `UNION ALL` or `UNION DISTINCT` anymore. `UNION ALL` means “all rows, but without cycles”, which is exactly what the `CYCLE` clause enables. And `UNION DISTINCT` means all rows should be different, which, again, is what will happen — as uniqueness is enforced over a subset of columns, complete rows will automatically all be different.

< 10.5.2

`CYCLE ... RESTRICT` is not available.

Examples

Below is an example with the `WITH` at the top level:

```
WITH t AS (SELECT a FROM t1 WHERE b >= 'c')
  SELECT * FROM t2, t WHERE t2.c = t.a;
```

The example below uses `WITH` in a subquery:

```
SELECT t1.a, t1.b FROM t1, t2
  WHERE t1.a > t2.c
     AND t2.c IN(WITH t AS (SELECT * FROM t1 WHERE t1.a < 5)
                 SELECT t2.c FROM t2, t WHERE t2.c = t.a);
```

Below is an example of a Recursive CTE:

```
WITH RECURSIVE ancestors AS
 ( SELECT * FROM folks
   WHERE name="Alex"
   UNION
   SELECT f.*
   FROM folks AS f, ancestors AS a
   WHERE f.id = a.father OR f.id = a.mother )
SELECT * FROM ancestors;
```

Consider the following structure and data:

```
CREATE TABLE t1 (from_ int, to_ int);
INSERT INTO t1 VALUES (1,2), (1,100), (2,3), (3,4), (4,1);
SELECT * FROM t1;
+-------+------+
| from_ | to_  |
+-------+------+
|     1 |    2 |
|     1 |  100 |
|     2 |    3 |
|     3 |    4 |
|     4 |    1 |
+-------+------+
```

Given the above, the following query would theoretically result in an infinite loop due to the last record in t1 (note that max_recursive_iterations is set to 10 for the purposes of this example, to avoid the excessive number of cycles):

```
SET max_recursive_iterations=10;

WITH RECURSIVE cte (depth, from_, to_) AS (
  SELECT 0,1,1 UNION DISTINCT SELECT depth+1, t1.from_, t1.to_
    FROM t1, cte  WHERE t1.from_ = cte.to_
)
SELECT * FROM cte;
+-------+-------+------+
| depth | from_ | to_  |
+-------+-------+------+
|     0 |     1 |    1 |
|     1 |     1 |    2 |
|     1 |     1 |  100 |
|     2 |     2 |    3 |
|     3 |     3 |    4 |
|     4 |     4 |    1 |
|     5 |     1 |    2 |
|     5 |     1 |  100 |
|     6 |     2 |    3 |
|     7 |     3 |    4 |
|     8 |     4 |    1 |
|     9 |     1 |    2 |
|     9 |     1 |  100 |
|    10 |     2 |    3 |
+-------+-------+------+
```

However, the `CYCLE ... RESTRICT` clause can overcome this:

```
WITH RECURSIVE cte (depth, from_, to_) AS (
  SELECT 0,1,1 UNION SELECT depth+1, t1.from_, t1.to_
    FROM t1, cte WHERE t1.from_ = cte.to_
)
CYCLE from_, to_ RESTRICT
SELECT * FROM cte;
+-------+-------+------+
| depth | from_ | to_  |
+-------+-------+------+
|     0 |     1 |    1 |
|     1 |     1 |    2 |
|     1 |     1 |  100 |
|     2 |     2 |    3 |
|     3 |     3 |    4 |
|     4 |     4 |    1 |
+-------+-------+------+
```

See Also

- Non-Recursive Common Table Expressions Overview
- Recursive Common Table Expressions Overview

This page is licensed: CC BY-SA / Gnu FDL

Joins

See JOIN Syntax.

Main `JOIN` page, documenting all join types available in MariaDB.

See Joining Tables with JOIN Clauses.

Introduction to joining tables in MariaDB.

See Advanced Joins.

Introduction to advanced joins in MariaDB.

See Comma vs JOIN.

How to use a comma instead of a `JOIN` clause, and why you should always prefer the latter.

This page is licensed: CC BY-SA / Gnu FDL

JOIN Syntax

> **Note:**
>
> For an introduction to joins, see Joining Tables with JOIN Clauses Guide.

Description

MariaDB supports the following `JOIN` syntaxes for the `table_references` part of SELECT statements and multiple-table DELETE and UPDATE statements:

```
table_references:
    table_reference [, table_reference] ...

table_reference:
    table_factor
  | join_table

table_factor (<= MariaDB 11.6):
    tbl_name [PARTITION (partition_list)]
        [query_system_time_period_specification] [[AS] alias] [index_hint_list]
  | table_subquery [query_system_time_period_specification] [AS] alias
  | ( table_references )
  | { OJ table_reference LEFT OUTER JOIN table_reference
        ON conditional_expr }

table_factor (>= MariaDB 11.7):
    tbl_name [PARTITION (partition_list)]
        [query_system_time_period_specification] [[AS] alias] [index_hint_list]
  | table_subquery [query_system_time_period_specification] [AS] alias [(column_name_list)]
  | ( table_references )
  | { OJ table_reference LEFT OUTER JOIN table_reference
        ON conditional_expr }

join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor

join_condition:
    ON conditional_expr
  | USING (column_list)

query_system_time_period_specification:
    FOR SYSTEM_TIME AS OF point_in_time
  | FOR SYSTEM_TIME BETWEEN point_in_time AND point_in_time
  | FOR SYSTEM_TIME FROM point_in_time TO point_in_time
  | FOR SYSTEM_TIME ALL

point_in_time:
    [TIMESTAMP] expression
  | TRANSACTION expression

index_hint_list:
    index_hint [, index_hint] ...

index_hint:
    USE {INDEX|KEY}
      [{FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
  | IGNORE {INDEX|KEY}
      [{FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
  | FORCE {INDEX|KEY}
      [{FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

index_list:
    index_name [, index_name] ...
```

A *table reference* is also known as a *join expression*.

Each table can also be specified as `db_name`.`tabl_name`. This allows to write queries which involve multiple databases. See Identifier Qualifiers for syntax details.

The syntax of `table_factor` is an extension to the SQL Standard. The latter accepts only `table_reference`, not a list of them inside a pair of parentheses.

This is a conservative extension if we consider each comma in a list of table_reference items as equivalent to an inner join. Consider this query:

```
SELECT * FROM t1 LEFT JOIN (t2, t3, t4)
                ON (t2.a=t1.a AND t3.b=t1.b AND t4.c=t1.c)
```

It is equivalent to this query:

```
SELECT * FROM t1 LEFT JOIN (t2 CROSS JOIN t3 CROSS JOIN t4)
                ON (t2.a=t1.a AND t3.b=t1.b AND t4.c=t1.c)
```

> **Note:**
>
> In MariaDB, `CROSS JOIN` is a syntactic equivalent to `INNER JOIN` (they can replace each other). In standard SQL, they are not equivalent. `INNER JOIN` is used with an `ON` clause, `CROSS JOIN` is used otherwise.

In general, parentheses can be ignored in join expressions containing only inner join operations. MariaDB also supports nested joins (see Nested Join Optimization).

Subqueries

A table subquery is specified as a parenthesized query and must contain a following derived table name (specified as *alias* in the above syntax specification).

CURRENT

You can optionally specify a list of column names in parenthesis.

```
SELECT ic1, ic2, ic3 FROM
      (
        SELECT c1, c2, c3 FROM t1 GROUP BY c1
      ) dt2 (ic1, ic2, ic3)
    JOIN t2 ON t2.c1 = dt2.ic1
    WHERE c2 > 0
    GROUP BY ic1
```

Here, the table subquery for t1 will be materialized and named dt2, with column names ic1, ic2, ic3. These column names are used outside the subquery.

< 11.7

You **cannot** optionally specify a list of column names in parenthesis.

See also Correlation Column List.

System-Versioned Tabled

See System-versioned tables for more information about the `FOR SYSTEM_TIME` syntax.

Index Hints

Index hints can be specified to affect how the MariaDB optimizer makes use of indexes. For more information, see How to force query plans.

Oracle Mode

> **Note:**
>
> This feature is available from MariaDB 12.1.

Overview

When Oracle mode is active, the Oracle-style `(+)` syntax can be used. For example, the following two queries are identical:

```
SELECT * FROM t1 LEFT JOIN t2 ON t1.a = t2.b;
```

```
SELECT * FROM t1, t2 WHERE t1.a = t2.b(+);
```

Similarly, the following two queries are identical:

```
SELECT * FROM t1 RIGHT JOIN t2 ON t1.a = t2.b;
```

```
SELECT * FROM t1, t2 WHERE t1.a(+) = t2.b;
```

With more than two tables, these two queries are identical:

```
SELECT * FROM t1 LEFT JOIN t2 ON t1.a = t2.a LEFT JOIN t3 ON t2.a = t3.a
```

```
SELECT * FROM t1, t2, t3 WHERE t1.a = t2.a(+) AND t2.a = t3.a(+)
```

To "rewrite" a join query using the `(+)` syntax, use EXPLAIN EXTENDED (the last line is an approximation at using the "regular" `LEFT JOIN` syntax):

```
EXPLAIN EXTENDED
    SELECT * FROM t1, t2, t3 WHERE t1.a = t2.a(+) AND t2.a = t3.a(+);
id  select_type table   type    possible_keys   key key_len ref rows    filtered    Extra
1   SIMPLE  t1  ALL NULL    NULL    NULL    NULL    3   100.00
1   SIMPLE  t2  ALL NULL    NULL    NULL    NULL    3   100.00  Using where; Using join buffer (flat, BNL
join)
1   SIMPLE  t3  ALL NULL    NULL    NULL    NULL    3   100.00  Using where; Using join buffer
(incremental, BNL join)
Warnings:
Note    1003    select "test"."t1"."a" AS "a","test"."t2"."a" AS "a","test"."t3"."a" AS "a" from
"test"."t1" left join "test"."t2" on("test"."t2"."a" = "test"."t1"."a") left join "test"."t3"
on("test"."t3"."a" = "test"."t2"."a") where 1
```

Limitations

The table whose columns are marked with the `(+)` operator in a subexpression (a part of the `WHERE` clause divided by `AND`) are the **inner part** of the expression. A table whose columns are not marked with the operator belong to the **outer part**.

Example of a single subexpression within a `WHERE` clause:

```
... WHERE t1.a = t2.a(+)
```

Example of two subexpressions within a `WHERE` clause – here, both `t1.a = t2.a(+)` and `t2.a = t3.a(+)` are inner parts, because both contain a `(+)` operator:

```
... WHERE t1.a = t2.a(+) AND t2.a = t3.a(+)
```

Example of two subexpressions within a `WHERE` clause – here, `t1.a = t2.a(+)` is the inner part (because of the `(+)` operator), and `t2.a = 42` is the outer part (it doesn't have a `(+)` operator:

```
... WHERE t1.a = t2.a(+) AND t2.a = 42
```

"Rewritten" as a "regular" join, that clause looks like this:

```
... FROM t1 LEFT JOIN t2 ON (t1.a = t2.a) WHERE t2.a = 42
```

The following limitations apply:

- The `(+)` operator can only be used in a `WHERE` clause.
- The `(+)` operator can only be applied to a table column, and the column should be from the local `SELECT`, not from an outer `SELECT`.
- The `(+)` operator cannot be used with other `JOIN` methods – it must be a comma-separated list in the `FROM` clause.
- When the `WHERE` clause combines subexpressions with `OR`, the `(+)` operator cannot be used.
- The `(+)` operator cannot be used on the right side of an `IN` function.
- The `(+)` operator cannot be used in row operations.
- The `(+)` operator cannot be used when two or more tables are on one side of and marked with the `(+)` operator and some are not.
- The `(+)` operator cannot create loops (or cycles) of dependence, where the same table appears on both sides of the operator in one expression, or through a chain of expressions.

Error Codes

The following errors may occur when not adhering to the `(+)` operator limitations or for other reasons:

- `ER_INVALID_USE_OF_ORA_JOIN`
- `ER_INVALID_USE_OF_ORA_JOIN_OUTER_REF`
- `ER_INVALID_USE_OF_ORA_JOIN_WRONG_FUNC`
- `ER_INVALID_USE_OF_ORA_JOIN_ONE_TABLE`
- `ER_INVALID_USE_OF_ORA_JOIN_CYCLE`
- `ER_INVALID_USE_OF_ORA_JOIN_MIX`
- `WARN_ORA_JOIN_IGNORED`

You can find examples of the errors by error code in these files (which are available on GitHub):

- `mysql-test/suite/compat/oracle/t/ora_outer_join.test`
- `mysql-test/suite/compat/oracle/t/ora_outer_join_err.test`

Examples

```
SELECT left_tbl.*
  FROM left_tbl LEFT JOIN right_tbl ON left_tbl.id = right_tbl.id
  WHERE right_tbl.id IS NULL;
```

See Also

- Joining Tables with JOIN Clauses Guide
- More Advanced Joins
- Comma vs JOIN
- Joins, Subqueries and SET

This page is licensed: GPLv2, originally from fill_help_tables.sql

Comma vs JOIN

A query to retrieve the list of phone numbers for clients who ordered in the last two weeks might be written in a couple of ways. Here are two:

```
SELECT *
FROM
  clients,
  orders,
  phoneNumbers
WHERE
  clients.id = orders.clientId
  AND clients.id = phoneNumbers.clientId
  AND orderPlaced >= NOW() - INTERVAL 2 WEEK;
```

```
SELECT *
FROM
  clients
  INNER JOIN orders ON clients.id = orders.clientId
  INNER JOIN phoneNumbers ON clients.id = phoneNumbers.clientId
WHERE
  orderPlaced >= NOW() - INTERVAL 2 WEEK;
```

Does it make a difference? Not much as written. But you should use the second form, for these reasons:

- Readability. Once the `WHERE` clause contains more than two conditions, it becomes tedious to pick out the difference between business logic (only dates in the last two weeks) and relational logic (which fields relate clients to orders). Using the `JOIN` syntax with an `ON` clause makes the `WHERE` list shorter, and makes it very easy to see how tables relate to each other.
- Flexibility. Let's say we need to see all clients even if they don't have a phone number in the system. With the second version, it's easy; just change `INNER JOIN phoneNumbers` to `LEFT JOIN phoneNumbers`. Try that with the first version, and MariaDB will raise an `ER_BAD_FIELD_ERROR` ("Unknown column") error when the query executes, because the change in precedence between the comma operator and the `JOIN` keyword means the `ON` clause can no longer resolve columns from the comma-joined table. The solution is to rearrange the `FROM` clause or add parentheses to override the precedence, and that quickly becomes frustrating.
- Portability. If your queries use standard SQL syntax, you will have an easier time switching to a different database should the need ever arise.

See Also

- "MySQL joins: ON vs. USING vs. Theta-style" — A blog entry about this topic.
- Joining Tables with JOIN Clauses Guide
- More Advanced Joins
- JOIN Syntax
- Joins, Subqueries and SET

The initial version of this article was copied, with permission, from Comma_vs_JOIN on 2012-10-05.

This page is licensed: CC BY-SA / Gnu FDL

Set Operations

See Precedence Control in Table Operations.

Control the execution order of UNION, EXCEPT, and INTERSECT operations. Learn how to use parentheses to define explicit operation priority.

See EXCEPT.

Return rows from the first result set that do not appear in the second. This set operator performs a subtraction of two datasets.

See INTERSECT.

Return only the rows that appear in both result sets. INTERSECT implicitly applies DISTINCT and follows the same column-naming, ORDER BY, and LIMIT rules as UNION.

See MINUS.

Oracle-compatible synonym for the EXCEPT operator. It returns rows from the first query that are not present in the second query.

See UNION.

Combine results from multiple SELECT statements into a single result set. This operator can optionally remove duplicates or include all rows.

This page is licensed: CC BY-SA / Gnu FDL

Precedence Control in Table Operations

You can control the ordering of execution on table operations using parentheses.

Syntax

```
(  expression )
[ORDER BY [column[, column...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Description

Using parentheses in SQL allows to control the order of execution for SELECT statements and Table Value Constructor, including UNION, EXCEPT, and INTERSECT operations. MariaDB executes the parenthetical expression before the rest of the statement. You can then use ORDER BY and LIMIT clauses the further organize the result set.

> **Note:**
>
> The Optimizer may rearrange the exact order in which MariaDB executes different parts of the statement. When it calculates the result set, however, it returns values as though the parenthetical expression were executed first.

Example

```
CREATE TABLE test.t1 (num INT);

INSERT INTO test.t1 VALUES (1),(2),(3);

(SELECT * FROM test.t1
 UNION
 VALUES (10))
INTERSECT
VALUES (1),(3),(10),(11);
+------+
| num  |
+------+
|    1 |
|    3 |
|   10 |
+------+

((SELECT * FROM test.t1
  UNION
  VALUES (10))
 INTERSECT
 VALUES (1),(3),(10),(11))
ORDER BY 1 DESC;
+------+
| num  |
+------+
|   10 |
|    3 |
|    1 |
+------+
```

This page is licensed: CC BY-SA / Gnu FDL

EXCEPT

EXCEPT

The result of `EXCEPT` contains all records of the left `SELECT` result set except records which are in right `SELECT` result set. In other words, it is the subtraction of two result sets.

> **CURRENT**
>
> `MINUS` is a synonym when SQL_MODE=ORACLE is set.
>
> **< 10.6.1**
>
> `MINUS` is a synonym is not available.

Syntax

```
SELECT ...
(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION [ALL | DISTINCT])
  SELECT ...
[(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION [ALL | DISTINCT])
  SELECT ...]
[ORDER BY [{col_name | expr | position} [ASC | DESC]
  [, {col_name | expr | position} [ASC | DESC] ...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}
| OFFSET start { ROW | ROWS }
| FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } { ONLY | WITH TIES } ]
```

Description

MariaDB supports `EXCEPT` and INTERSECT in addition to UNION.

The queries before and after `EXCEPT` must be SELECT or VALUES statements.

All behavior for naming columns, `ORDER BY` and `LIMIT` is the same as for UNION. Note that the alternative SELECT … OFFSET … FETCH syntax is only supported. This allows us to use the `WITH TIES` clause.

`EXCEPT` implicitly supposes a `DISTINCT` operation.

The result of `EXCEPT` is all records of the left `SELECT` result except records which are in right `SELECT` result set, i.e. it is subtraction of two result sets.

`EXCEPT` and `UNION` have the same operation precedence and `INTERSECT` has a higher precedence, unless running in Oracle mode, in which case all three have the same precedence.

Parentheses

Parentheses can be used to specify precedence.

ALL/DISTINCT

> **CURRENT**
>
> `EXCEPT ALL` and `EXCEPT DISTINCT` . The `ALL` operator leaves duplicates intact, while the `DISTINCT` operator removes duplicates. `DISTINCT` is the default behavior if neither operator is supplied.
>
> **< 10.5**
>
> Only `EXCEPT DISTINCT` is available.

Examples

Show customers which are not employees:

```
(SELECT e_name AS name, email FROM customers)
EXCEPT
(SELECT c_name AS name, email FROM employees);
```

Difference between UNION, EXCEPT and INTERSECT:

```
CREATE TABLE seqs (i INT);
INSERT INTO seqs VALUES (1),(2),(2),(3),(3),(4),(5),(6);

SELECT i FROM seqs WHERE i <= 3 UNION SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
+------+

SELECT i FROM seqs WHERE i <= 3 UNION ALL SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
|    2 |
|    3 |
|    3 |
|    3 |
|    3 |
|    4 |
|    5 |
|    6 |
+------+

SELECT i FROM seqs WHERE i <= 3 EXCEPT SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
+------+

SELECT i FROM seqs WHERE i <= 3 EXCEPT ALL SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
|    2 |
+------+

SELECT i FROM seqs WHERE i <= 3 INTERSECT SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    3 |
+------+

SELECT i FROM seqs WHERE i <= 3 INTERSECT ALL SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    3 |
|    3 |
+------+
```

Parentheses for specifying precedence:

```
CREATE OR REPLACE TABLE t1 (a INT);
CREATE OR REPLACE TABLE t2 (b INT);
CREATE OR REPLACE TABLE t3 (c INT);

INSERT INTO t1 VALUES (1),(2),(3),(4);
INSERT INTO t2 VALUES (5),(6);
INSERT INTO t3 VALUES (1),(6);

((SELECT a FROM t1) UNION (SELECT b FROM t2)) EXCEPT (SELECT c FROM t3);
+------+
| a    |
+------+
|    2 |
|    3 |
|    4 |
|    5 |
+------+

(SELECT a FROM t1) UNION ((SELECT b FROM t2) EXCEPT (SELECT c FROM t3));
+------+
| a    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
+------+
```

Here is an example that makes use of the SEQUENCE storage engine and the VALUES statement, to generate a numeric sequence and remove some arbitrary numbers from it:

```
(SELECT seq FROM seq_1_to_10) EXCEPT VALUES (2), (3), (4);
+-----+
| seq |
+-----+
|   1 |
|   5 |
|   6 |
|   7 |
|   8 |
|   9 |
|  10 |
+-----+
```

See Also

- UNION
- INTERSECT
- Get Set for Set Theory: UNION, INTERSECT and EXCEPT in SQL (video tutorial)

This page is licensed: CC BY-SA / Gnu FDL

INTERSECT

Return only rows that appear in both result sets. This set operator identifies the common records shared between two queries.

Syntax

```
SELECT ...
(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION [ALL | DISTINCT]) SELECT ...
[(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION [ALL | DISTINCT]) SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Description

MariaDB has supported `INTERSECT` (as well as EXCEPT) in addition to UNION since MariaDB 10.3.

All behavior for naming columns, `ORDER BY` and `LIMIT` is the same as for UNION.

`INTERSECT` implicitly supposes a `DISTINCT` operation.

The result of an intersect is the intersection of right and left `SELECT` results, i.e. only records that are present in both result sets will be included in the result of the operation.

`INTERSECT` has higher precedence than `UNION` and `EXCEPT` (unless running running in Oracle mode, in which case all three have the same precedence). If possible it will be executed linearly, but if not, it will be translated to a subquery in the `FROM` clause:

```
(SELECT a,b FROM t1)
UNION
(SELECT c,d FROM t2)
INTERSECT
(SELECT e,f FROM t3)
UNION
(SELECT 4,4);
```

will be translated to:

```
(SELECT a,b FROM t1)
UNION
SELECT c,d FROM
  ((SELECT c,d FROM t2)
   INTERSECT
   (SELECT e,f FROM t3)) dummy_subselect
UNION
(SELECT 4,4)
```

Parentheses

CURRENT

Parentheses can be used to specify precedence.

< 10.4

Parentheses **cannot** be used to specify precedence.

MariaDB starting with 10.5.0

ALL/DISTINCT

CURRENT

`INTERSECT ALL` and `INTERSECT DISTINCT` . The `ALL` operator leaves duplicates intact, while the `DISTINCT` operator removes duplicates. `DISTINCT` is the default behavior if neither operator is supplied.

< 10.5

> `DISTINCT` is the only behavior available.

Examples

Show customers which are employees:

```
(SELECT e_name AS name, email FROM employees)
INTERSECT
(SELECT c_name AS name, email FROM customers);
```

Difference between UNION, EXCEPT and `INTERSECT`:

```
CREATE TABLE seqs (i INT);
INSERT INTO seqs VALUES (1),(2),(2),(3),(3),(4),(5),(6);

SELECT i FROM seqs WHERE i <= 3 UNION SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
+------+

SELECT i FROM seqs WHERE i <= 3 UNION ALL SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
|    2 |
|    3 |
|    3 |
|    3 |
|    3 |
|    4 |
|    5 |
|    6 |
+------+

SELECT i FROM seqs WHERE i <= 3 EXCEPT SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
+------+

SELECT i FROM seqs WHERE i <= 3 EXCEPT ALL SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
|    2 |
+------+

SELECT i FROM seqs WHERE i <= 3 INTERSECT SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    3 |
+------+

SELECT i FROM seqs WHERE i <= 3 INTERSECT ALL SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    3 |
|    3 |
+------+
```

Parentheses for specifying precedence:

```
CREATE OR REPLACE TABLE t1 (a INT);
CREATE OR REPLACE TABLE t2 (b INT);
CREATE OR REPLACE TABLE t3 (c INT);

INSERT INTO t1 VALUES (1),(2),(3),(4);
INSERT INTO t2 VALUES (5),(6);
INSERT INTO t3 VALUES (1),(6);

((SELECT a FROM t1) UNION (SELECT b FROM t2)) INTERSECT (SELECT c FROM t3);
+------+
| a    |
+------+
|    1 |
|    6 |
+------+

(SELECT a FROM t1) UNION ((SELECT b FROM t2) INTERSECT (SELECT c FROM t3));
+------+
| a    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    6 |
+------+
```

See Also

- UNION
- EXCEPT
- Get Set for Set Theory: UNION, INTERSECT and EXCEPT in SQL (video tutorial)

This page is licensed: CC BY-SA / Gnu FDL

MINUS

> **Note:**
>
> `MINUS` is available starting from MariaDB 10.6.1.

`MINUS` is a synonym for EXCEPT when SQL_MODE=ORACLE is set.

```
CREATE TABLE seqs (i INT);
INSERT INTO seqs VALUES (1),(2),(2),(3),(3),(4),(5),(6);

SET SQL_MODE='ORACLE';

SELECT i FROM seqs WHERE i <= 3 MINUS SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
+------+
```

This page is licensed: CC BY-SA / Gnu FDL

UNION

`UNION` is used to combine the results from multiple SELECT statements into a single result set.

Syntax

```
SELECT ...
UNION [ALL | DISTINCT] SELECT ...
[UNION [ALL | DISTINCT] SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Description

`UNION` is used to combine the results from multiple SELECT statements into a single result set.

The column names from the first `SELECT` statement are used as the column names for the results returned. Selected columns listed in corresponding positions of each SELECT statement should have the same data type. (For example, the first column selected by the first statement should have the same type as the first column selected by the other statements.)

If they don't, the type and length of the columns in the result take into account the values returned by all of the SELECTs, so there is no need for explicit casting. Note that currently this is not the case for recursive CTEs - see MDEV-12325.

Table names can be specified as `db_name`.`tbl_name`. This permits writing `UNION`s which involve multiple databases. See Identifier Qualifiers for syntax details.

A UNION's global `ORDER BY` cannot reference an aggregate function. (Aggregate functions are allowed within the individual `SELECT` statements of a UNION.)

`EXCEPT` and `UNION` have the same operation precedence and `INTERSECT` has a higher precedence, unless running in Oracle mode, in which case all three have the same precedence.

ALL/DISTINCT

The `ALL` keyword causes duplicate rows to be preserved. The `DISTINCT` keyword (the default if the keyword is omitted) causes duplicate rows to be removed by the results.

`UNION ALL` and `UNION DISTINCT` can both be present in a query. In this case, `UNION` DISTINCT will override any `UNION ALL` to its left.

The server can in most cases execute `UNION ALL` without creating a temporary table (see MDEV-334).

ORDER BY and LIMIT

Individual `SELECT` statements can contain their own ORDER BY and LIMIT clauses. In this case, the individual queries need to be wrapped between parentheses. However, this does not affect the order of the UNION, so they only are useful to limit the record read by one `SELECT`.

The `UNION` can have global ORDER BY and LIMIT clauses, which affect the whole result set. If the columns retrieved by individual SELECT statements have an alias (AS), the `ORDER BY` must use that alias, not the real column names.

HIGH_PRIORITY

Specifying a query as HIGH_PRIORITY does not work inside a `UNION`. If applied to the first `SELECT`, it is ignored. Applying to a later `SELECT` results in a syntax error:

```
ERROR 1234 (42000): Incorrect usage/placement of 'HIGH_PRIORITY'
```

SELECT … INTO …

Individual `SELECT` statements cannot be written INTO DUMPFILE or INTO OUTFILE. If the last `SELECT` statement specifies `INTO DUMPFILE` or `INTO OUTFILE`, the entire result of the `UNION` will be written. Placing the clause after any other `SELECT` will result in a syntax error.

If the result is a single row, SELECT … INTO @var_name can also be used.

Parentheses

> **CURRENT**
>
> Parentheses can be used to specify precedence.
>
> **< 10.4**
>
> Parentheses **cannot** be used to specify precedence.

Examples

`UNION` between tables having different column names:

```
(SELECT e_name AS name, email FROM employees)
UNION
(SELECT c_name AS name, email FROM customers);
```

Specifying the `UNION` global order and limiting total rows:

```
(SELECT name, email FROM employees)
UNION
(SELECT name, email FROM customers)
ORDER BY name LIMIT 10;
```

Adding a constant row:

```
(SELECT 'John Doe' AS name, 'john.doe@example.net' AS email)
UNION
(SELECT name, email FROM customers);
```

Differing types:

```
SELECT CAST('x' AS CHAR(1)) UNION SELECT REPEAT('y',4);
+----------------------+
| CAST('x' AS CHAR(1)) |
+----------------------+
| x                    |
| yyyy                 |
+----------------------+
```

Returning the results in order of each individual `SELECT` by use of a sort column:

```
(SELECT 1 AS sort_column, e_name AS name, email FROM employees)
UNION
(SELECT 2, c_name AS name, email FROM customers) ORDER BY sort_column;
```

Difference between UNION, EXCEPT and INTERSECT:

```
CREATE TABLE seqs (i INT);
INSERT INTO seqs VALUES (1),(2),(2),(3),(3),(4),(5),(6);

SELECT i FROM seqs WHERE i <= 3 UNION SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
+------+

SELECT i FROM seqs WHERE i <= 3 UNION ALL SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
|    2 |
|    3 |
|    3 |
|    3 |
|    3 |
|    4 |
|    5 |
|    6 |
+------+

SELECT i FROM seqs WHERE i <= 3 EXCEPT SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
+------+

SELECT i FROM seqs WHERE i <= 3 EXCEPT ALL SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    1 |
|    2 |
|    2 |
+------+

SELECT i FROM seqs WHERE i <= 3 INTERSECT SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    3 |
+------+

SELECT i FROM seqs WHERE i <= 3 INTERSECT ALL SELECT i FROM seqs WHERE i>=3;
+------+
| i    |
+------+
|    3 |
|    3 |
+------+
```

```
CREATE OR REPLACE TABLE t1 (a INT);
CREATE OR REPLACE TABLE t2 (b INT);
CREATE OR REPLACE TABLE t3 (c INT);

INSERT INTO t1 VALUES (1),(2),(3),(4);
INSERT INTO t2 VALUES (5),(6);
INSERT INTO t3 VALUES (1),(6);

((SELECT a FROM t1) UNION (SELECT b FROM t2)) INTERSECT (SELECT c FROM t3);
+------+
| a    |
+------+
|    1 |
|    6 |
+------+

(SELECT a FROM t1) UNION ((SELECT b FROM t2) INTERSECT (SELECT c FROM t3));
+------+
| a    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    6 |
+------+
```

See Also

- SELECT
- EXCEPT
- INTERSECT
- Recursive Common Table Expressions Overview
- Get Set for Set Theory: UNION, INTERSECT and EXCEPT in SQL (video tutorial)

This page is licensed: GPLv2, originally from fill_help_tables.sql

Subqueries

See Subqueries With ALL.

Compare a value against all results from a subquery. The `ALL` operator returns `TRUE` if the comparison holds for every row returned by the subquery.

See Subqueries With ANY and SOME.

Compare a value against any result from a subquery. The `ANY` (or `SOME`) operator returns `TRUE` if the comparison holds for at least one row. `IN` can be used for `=ANY`.

See Subqueries With EXISTS.

Test for the existence of rows. The `EXISTS` operator returns `TRUE` if the subquery returns at least one row, often used for correlated subqueries.

See Subqueries With JOINs.

Understand when to use subqueries versus joins. This guide explains performance implications and how to rewrite subqueries as joins for efficiency.

See Subqueries in a FROM Clause (Derived Tables).

Use a subquery as a temporary table. Derived tables allow you to select from the result set of another query within the `FROM` clause.

See Row Subqueries.

Retrieve a single row of multiple values. A row subquery returns a tuple that can be compared against a row constructor in the outer query.

See Scalar Subqueries.

Return a single value. A scalar subquery produces a one-row, one-column result that can be used anywhere a constant or expression is valid.

See Subquery Limitations.

Review restrictions on subquery usage. This page details unsupported operations, such as modifying a table while selecting from it in a subquery.

This page is licensed: CC BY-SA / Gnu FDL

Subqueries With ALL

Subqueries and ALL

Subqueries using the ALL keyword will return `true` if the comparison returns `true` for each row returned by the subquery, or the subquery returns no rows.

Syntax

```
scalar_expression comparison_operator ALL <Table subquery>
```

- `scalar_expression` may be any expression that evaluates to a single value.
- `comparison_operator` may be any one of: `=`, `>`, `<`, `>=`, `<=`, `<>` or `!=`

`ALL` returns:

- `NULL` if the comparison operator returns `NULL` for at least one row returned by the Table subquery or scalar_expression returns `NULL`.
- `FALSE` if the comparison operator returns `FALSE` for at least one row returned by the Table subquery.
- `TRUE` if the comparison operator returns `TRUE` for all rows returned by the Table subquery, or if Table subquery returns no rows.

`NOT IN` is an alias for `<> ALL`.

Examples

```
CREATE TABLE sq1 (num TINYINT);

CREATE TABLE sq2 (num2 TINYINT);

INSERT INTO sq1 VALUES(100);

INSERT INTO sq2 VALUES(40),(50),(60);

SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2);
+------+
| num  |
+------+
|  100 |
+------+
```

Since `100` > all of `40`, `50` and `60`, the evaluation is true and the row is returned.

Adding a second row to sq1, where the evaluation for that record is false:

```
INSERT INTO sq1 VALUES(30);

SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2);
+------+
| num  |
+------+
|  100 |
+------+
```

Adding a new row to sq2, causing all evaluations to be false:

```
INSERT INTO sq2 VALUES(120);

SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2);
Empty set (0.00 sec)
```

When the subquery returns no results, the evaluation is still true:

```
SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2 WHERE num2 > 300);
+------+
| num  |
+------+
|  100 |
|   30 |
+------+
```

Evaluating against a NULL will cause the result to be unknown, or not true, and therefore return no rows:

```
INSERT INTO sq2 VALUES (NULL);

SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2);
```

This page is licensed: CC BY-SA / Gnu FDL

Subqueries With ANY and SOME

Syntax

`SOME` is a synonym for `ANY`.

```
scalar_expression comparison_operator ANY <Table subquery>
scalar_expression comparison_operator SOME <Table subquery>
```

- `scalar_expression` may be any expression that evaluates to a single value.
- `comparison_operator` may be any one of `=`, `>`, `<`, `>=`, `<=`, `<>` or `!=`.

> **Note:**
>
> **About IN**
>
> Although `IN` functions similarly to `= ANY`, it is documented separately due to unique optimization characteristics. Refer to Subqueries with IN and NOT IN for more details.

`ANY` or `SOME` returns:

- `TRUE` if the comparison operator returns `TRUE` for at least one row returned by the table subquery.
- `FALSE` if the comparison operator returns `FALSE` for all rows returned by the table subquery, or if the table subquery has no rows.
- `NULL` if the comparison operator returns `NULL` for at least one row returned by the table subquery and doesn't returns `TRUE` for any of them, or if `scalar_expression` returns `NULL`.

Examples

The subsequent examples use these tables:

```
CREATE TABLE sq1 (num1 TINYINT);
CREATE TABLE sq2 (num2 TINYINT);
INSERT INTO sq1 VALUES(100);
INSERT INTO sq2 VALUES(40),(50),(120);
```

Subquery With `ANY`

```
SELECT * FROM sq1 WHERE num1 > ANY (SELECT * FROM sq2);
+------+
| num1 |
+------+
|  100 |
+------+
```

`100` is greater than two of the three values, and so the expression evaluates to `true`.

Subquery With `SOME`

`SOME` is a synonym for `ANY`.

```
SELECT * FROM sq1 WHERE num1 > SOME (SELECT * FROM sq2);
+------+
| num1 |
+------+
|  100 |
+------+
```

`100` is greater than two of the three values, and so the expression evaluates to `true`.

See Also

- Subqueries with IN
- Subqueries with ALL

This page is licensed: CC BY-SA / Gnu FDL

Subqueries with IN and NOT IN

The `IN` operator tests whether a value is a member of a set of values returned by a subquery. It is the most commonly used construct for equality-based subquery comparisons in MariaDB.

While `IN` is logically related to the `ANY` and `SOME` operators, it has distinct semantics and receives specialized treatment from the query optimizer.

Syntax

```
scalar_expression [NOT] IN (subquery)
```

- `scalar_expression`: An expression that evaluates to a single value.
- `subquery`: A subquery that returns a single column.

Relationship to ANY and SOME

In MariaDB, `IN` is a synonym for `= ANY`.

- `IN` evaluates to `TRUE` if the comparison returns `TRUE` for at least one row produced by the subquery.

- `IN` evaluates to `FALSE` if the subquery returns no rows.

Despite this logical equivalence, the optimizer applies additional transformations to `IN` subqueries (such as semi-joins) that do not always apply to other quantitative `ANY` comparisons like `> ANY` or `<= SOME`.

Handling NULL Values

The behavior of `IN` and `NOT IN` follows SQL three-valued logic.

- Source Expression is NULL: If the `scalar_expression` is `NULL`, the result is `NULL`.
- Subquery Contains NULL (IN): If no match is found and the subquery contains at least one `NULL`, the result is `NULL`.
- Subquery Contains NULL (NOT IN): If the subquery returns any `NULL` value, `NOT IN` will always evaluate to `NULL` or `FALSE`, never `TRUE`, even if no matching non-null value exists.

> **Warning:**
>
> Avoid using `NOT IN` if the subquery columns can contain `NULL` values. In such cases, `NOT EXISTS` is generally safer and more predictable.

NOT IN vs. <> ANY

`NOT IN` and `<> ANY` are not synonyms and represent different logical tests:

OperatorText	Logical MeaningText	ResultText
`NOT IN`	“Is this value absent from the entire set?”	Returns `TRUE` only if the value matches no rows in the subquery.
`<> ANY`	“Is there at least one value in the set that is different?”	Returns `TRUE` if there is at least one row in the subquery that differs from the value.

Optimization and Performance

MariaDB treats `IN` subqueries as high-priority candidates for specific optimizations, including:

- Semi-join transformations: Converting the subquery into a join to improve execution speed.
- Subquery materialization: Storing the subquery result in a temporary internal table to avoid repeated execution.
- IN-to-EXISTS rewrites: Converting the predicate based on cost estimates.

Examples

The following examples use table `sq1` (values: `50`, `100`) and table `sq2` (values: `40`, `50`, `120`).

Using IN

```
SELECT * FROM sq1 WHERE num1 IN (SELECT num2 FROM sq2);
```

Result: `50` (Because `50` exists in both tables).

Using NOT IN

```
SELECT * FROM sq1 WHERE num1 NOT IN (SELECT num2 FROM sq2);
```

Result: `100` (Because `100` is not present in `sq2`).

NOT IN with NULL Pitfall

```
-- sq1 contains 50, 100
SELECT * FROM sq1 WHERE num1 NOT IN (SELECT NULL);
```

Result: Empty set (The condition evaluates to `NULL` for all rows, so no rows are returned).

This page is licensed: CC BY-SA / Gnu FDL

Subqueries With EXISTS

Subqueries and EXISTS

Syntax

```
SELECT ... WHERE EXISTS <Table subquery>
```

Description

Subqueries using the `EXISTS` keyword will return `true` if the subquery returns any rows. Conversely, subqueries using `NOT EXISTS` will return `true` only if the subquery returns no rows from the table.

EXISTS subqueries ignore the columns specified by the SELECT of the subquery, since they're not relevant. For example,

```
SELECT col1 FROM t1 WHERE EXISTS (SELECT * FROM t2);
```

and

```
SELECT col1 FROM t1 WHERE EXISTS (SELECT col2 FROM t2);
```

produce identical results.

Examples

```
CREATE TABLE sq1 (num TINYINT);

CREATE TABLE sq2 (num2 TINYINT);

INSERT INTO sq1 VALUES(100);

INSERT INTO sq2 VALUES(40),(50),(60);

SELECT * FROM sq1 WHERE EXISTS (SELECT * FROM sq2 WHERE num2>50);
+------+
| num  |
+------+
|  100 |
+------+

SELECT * FROM sq1 WHERE NOT EXISTS (SELECT * FROM sq2 GROUP BY num2 HAVING MIN(num2)=40);
Empty set (0.00 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Subqueries With JOINs

Subqueries and JOINs

A subquery can quite often, but not in all cases, be rewritten as a JOIN.

Rewriting Subqueries as JOINS

A subquery using `IN` can be rewritten with the `DISTINCT` keyword. Consider this query:

```
SELECT * FROM table1 WHERE col1 IN (SELECT col1 FROM table2);
```

It can be rewritten like this:

```
SELECT DISTINCT table1.* FROM table1, table2 WHERE table1.col1=table2.col1;
```

`NOT IN` or `NOT EXISTS` queries can also be rewritten. For example, these two queries return the same result when the compared columns contain no `NULL` values. They can differ when `NULL`s are present, because `NOT IN` evaluates to `UNKNOWN` against a `NULL`:

```
SELECT * FROM table1 WHERE col1 NOT IN (SELECT col1 FROM table2);
SELECT * FROM table1 WHERE NOT EXISTS (SELECT col1 FROM table2
         WHERE table1.col1=table2.col1);
```

They can both be rewritten like this:

```
SELECT table1.* FROM table1 LEFT JOIN table2 ON table1.col1=table2.col1
       WHERE table2.col1 IS NULL;
```

Subqueries that can be rewritten as a `LEFT JOIN` are sometimes more efficient.

Using Subqueries instead of JOINS

There are some scenarios, though, which call for subqueries rather than joins:

- When you want duplicates, but not false duplicates. Suppose `Table_1` has three rows — {`1`,`1`,`2`} — and `Table_2` has two rows — {`1`,`2`,`2`}. If you need to list the rows in `Table_1` which are also in `Table_2`, only this subquery-based `SELECT` statement will give the right answer (`1`,`1`,`2`):

```
SELECT Table_1.column_1
FROM   Table_1
WHERE  Table_1.column_1 IN
  (SELECT Table_2.column_1
    FROM   Table_2);
```

- This SQL statement won't work:

```
SELECT Table_1.column_1
FROM   Table_1,Table_2
WHERE  Table_1.column_1 = Table_2.column_1;
```

- because the result will be {`1`,`1`,`2`,`2`} — and the duplication of 2 is an error. This SQL statement won't work either:

```
SELECT DISTINCT Table_1.column_1
FROM   Table_1,Table_2
WHERE  Table_1.column_1 = Table_2.column_1;
```

- because the result will be {`1`,`2`} — and the removal of the duplicated 1 is an error too.
- When the outermost statement is not a query. The SQL statement:

```
UPDATE Table_1 SET column_1 = (SELECT column_1 FROM Table_2);
```

- can't be expressed using a join unless some rare SQL3 features are used.

- When the join is over an expression. The SQL statement:

```
SELECT * FROM Table_1
WHERE column_1 + 5 =
  (SELECT MAX(column_1) FROM Table_2);
```

- is hard to express with a join. In fact, the only way we can think of is this SQL statement:

```
SELECT Table_1.*
FROM   Table_1,
      (SELECT MAX(column_1) AS max_column_1 FROM Table_2) AS Table_2
WHERE  Table_1.column_1 + 5 = Table_2.max_column_1;
```

- which still involves a parenthesized query, so nothing is gained from the transformation.
- When you want to see the exception. For example, suppose the question is: Which books are longer than Das Kapital? These two queries are effectively almost the same:

```
SELECT DISTINCT Bookcolumn_1.*
FROM   Books AS Bookcolumn_1 JOIN Books AS Bookcolumn_2 USING(page_count)
WHERE  title = 'Das Kapital';

SELECT DISTINCT Bookcolumn_1.*
FROM   Books AS Bookcolumn_1
WHERE  Bookcolumn_1.page_count >
  (SELECT DISTINCT page_count
  FROM   Books AS Bookcolumn_2
  WHERE  title = 'Das Kapital');
```

- The difference is between these two SQL statements is, if there are two editions of Das Kapital (with different page counts), then the self-join example will return the books which are longer than the shortest edition of Das Kapital. That might be the wrong answer, since the original question didn't ask for "… longer than `ANY` book named Das Kapital" (it seems to contain a false assumption that there's only one edition).

This page is licensed: CC BY-SA / Gnu FDL

Subqueries in a FROM Clause (Derived Tables)

Although subqueries are more commonly placed in a WHERE clause, they can also form part of the FROM clause. Such subqueries are commonly called derived tables.

If a subquery is used in this way, you must also use an AS clause to name the result of the subquery.

ORACLE mode

MariaDB starting with 10.6.0

> **CURRENT**
>
> Anonymous subqueries in a FROM clause (no AS clause) are permitted in ORACLE mode.
>
> **< 10.6**
>
> Anonymous subqueries in a FROM clause (no `AS` clause) are **not** permitted in ORACLE mode.

Correlation Column List

MariaDB starting with 11.7.0

> **CURRENT**

> It is possible to assign column names in the derived table name syntax element.
>
> **< 11.7**
>
> It is **not** possible to assign column names in the derived table name syntax element.

Examples

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
  ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
  ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
  ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
  ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);
```

Assume that, given the data above, you want to return the average total for all students. In other words, the average of Chun's 148 (75+73), Esben's 74 (43+31), etc.

You cannot do the following:

```
SELECT AVG(SUM(score)) FROM student GROUP BY name;
ERROR 1111 (HY000): Invalid use of group function
```

A subquery in the FROM clause is however permitted:

```
SELECT AVG(sq_sum) FROM (SELECT SUM(score) AS sq_sum FROM student GROUP BY name) AS t;
+-------------+
| AVG(sq_sum) |
+-------------+
|    134.0000 |
+-------------+
```

The following is permitted only in ORACLE mode, where a derived table may omit its alias:

```
SELECT * FROM (SELECT 1 FROM DUAL), (SELECT 2 FROM DUAL);
```

In this example, the second column of the derived table `dt` is used both within (`WHERE` c2 > 0), and outside, (`WHERE` a2 > 10), the specification. Both conditions apply to t1.c2.

```
CREATE OR REPLACE TABLE t1(c1 INT, c2 INT, c3 INT);

SELECT a1, a2 FROM (SELECT c1, c2, c3 FROM t1 WHERE c2 > 0) AS dt (a1, a2, a3) WHERE a2 > 10;
```

This page is licensed: CC BY-SA / Gnu FDL

Row Subqueries

A row subquery is a subquery returning a single row, as opposed to a scalar subquery, which returns a single column from a row, or a literal.

Examples

```
CREATE TABLE staff (name VARCHAR(10), age TINYINT);

CREATE TABLE customer (name VARCHAR(10), age TINYINT);

INSERT INTO staff VALUES ('Bilhah',37), ('Valerius',61), ('Maia',25);

INSERT INTO customer VALUES ('Thanasis',48), ('Valerius',61), ('Brion',51);

SELECT * FROM staff WHERE (name,age) = (SELECT name,age FROM customer WHERE name='Valerius');
+----------+------+
| name     | age  |
+----------+------+
| Valerius |   61 |
+----------+------+
```

Finding all rows in one table and also in another:

```
SELECT name,age FROM staff WHERE (name,age) IN (SELECT name,age FROM customer);
+----------+------+
| name     | age  |
+----------+------+
| Valerius |   61 |
+----------+------+
```

This page is licensed: CC BY-SA / Gnu FDL

Scalar Subqueries

A scalar subquery is a subquery that returns a single value. This is the simplest form of a subquery, and can be used in most places a literal or single column value is valid.

The data type, length and character set and collation are all taken from the result returned by the subquery. The result of a subquery can always be NULL, that is, no result returned. Even if the original value is defined as NOT NULL, this is disregarded.

A subquery cannot be used where only a literal is expected, for example LOAD DATA INFILE expects a literal string containing the file name, and LIMIT requires a literal integer.

Examples

```
CREATE TABLE sq1 (num TINYINT);

CREATE TABLE sq2 (num TINYINT);

INSERT INTO sq1 VALUES (1);

INSERT INTO sq2 VALUES (10* (SELECT num FROM sq1));

SELECT * FROM sq2;
+------+
| num  |
+------+
|   10 |
+------+
```

Inserting a second row means the subquery is no longer a scalar, and this particular query is not valid:

```
INSERT INTO sq1 VALUES (2);

INSERT INTO sq2 VALUES (10* (SELECT num FROM sq1));
ERROR 1242 (21000): Subquery returns more than 1 row
```

No rows in the subquery, so the scalar is NULL:

```
INSERT INTO sq2 VALUES (10* (SELECT num FROM sq3 WHERE num='3'));

SELECT * FROM sq2;
+------+
| num  |
+------+
|   10 |
| NULL |
+------+
```

A more traditional scalar subquery, as part of a `WHERE` clause:

```
SELECT * FROM sq1 WHERE num = (SELECT MAX(num)/10 FROM sq2);
+------+
| num  |
+------+
|    1 |
+------+
```

This page is licensed: CC BY-SA / Gnu FDL

Subquery Limitations

There are a number of limitations regarding subqueries, which are discussed below.

The following tables and data will be used in the examples that follow:

```
CREATE TABLE staff(name VARCHAR(10),age TINYINT);

CREATE TABLE customer(name VARCHAR(10),age TINYINT);
```

```
INSERT INTO staff VALUES
('Bilhah',37), ('Valerius',61), ('Maia',25);

INSERT INTO customer VALUES
('Thanasis',48), ('Valerius',61), ('Brion',51);
```

ORDER BY and LIMIT

LIMIT is not permitted in a subquery used with `IN`, `ALL`, `ANY`, or `SOME`. ORDER BY on its own is allowed. For example:

```
SELECT * FROM staff WHERE name IN (SELECT name FROM customer ORDER BY name);
+----------+------+
| name     | age  |
+----------+------+
| Valerius |   61 |
+----------+------+
```

is valid, but

```
SELECT * FROM staff WHERE name IN (SELECT NAME FROM customer ORDER BY name LIMIT 1);
ERROR 1235 (42000): This version of MariaDB doesn't
  yet support 'LIMIT & IN/ALL/ANY/SOME subquery'
```

is not.

Modifying and Selecting from the Same Table

It's not possible to both modify and select from the same table in a subquery. For example:

```
DELETE FROM staff WHERE name = (SELECT name FROM staff WHERE age=61);
ERROR 1093 (HY000): Table 'staff' is specified twice, both
  as a target for 'DELETE' and as a separate source for data
```

Row Comparison Operations

There is only partial support for row comparison operations. The expression in

```
expr op {ALL|ANY|SOME} subquery,
```

must be scalar and the subquery can only return a single column.

However, because of the way `IN` is implemented (it is rewritten as a sequence of `=` comparisons and `AND`), the expression in

```
expression [NOT] IN subquery
```

is permitted to be an n-tuple and the subquery can return rows of n-tuples.

For example:

```
SELECT * FROM staff WHERE (name,age) NOT IN (
  SELECT name,age FROM customer WHERE age >=51
);
+--------+------+
| name   | age  |
+--------+------+
| Bilhah |   37 |
| Maia   |   25 |
+--------+------+
```

is permitted, but

```
SELECT * FROM staff WHERE (name,age) = ALL (
  SELECT name,age FROM customer WHERE age >=51
);
ERROR 1241 (21000): Operand should contain 1 column(s)
```

is not.

Correlated Subqueries

Subqueries in the `FROM` clause cannot be correlated subqueries. They cannot be evaluated for each row of the outer query since they are evaluated to produce a result set during when the query is executed.

Stored Functions

A subquery can refer to a stored function which modifies data. This is an extension to the SQL standard, but can result in indeterminate outcomes. For example, take:

```
SELECT ... WHERE x IN (SELECT f() ...);
```

where *f()* inserts rows. The function *f()* could be executed a different number of times depending on how the optimizer chooses to handle the query.

This sort of construct is therefore not safe to use in replication that is not row-based, as there could be different results on the master and the slave.

This page is licensed: CC BY-SA / Gnu FDL

Geometry Statements

See Geometry Constructors.

Learn about geometry constructors in MariaDB Server. This section details SQL functions for creating spatial data types like points, lines, and polygons, enabling geospatial data management.

See Geometry Properties.

Learn about geometry properties. This section details SQL functions for retrieving attributes of spatial objects, such as area, length, and bounding box, essential for geospatial analysis.

See Geometry Relations.

Learn about geometry relations in MariaDB Server. This section details SQL functions for determining spatial relationships between geometric objects, such as ST_Intersects and ST_Contains.

See LineString Properties.

Learn about LINESTRING properties in MariaDB Server. This section details SQL functions for retrieving attributes of linear spatial objects, such as length, number of points, and start/end points.

See MBR (Minimum Bounding Rectangle).

Learn about Minimum Bounding Rectangles (MBR) in MariaDB Server. This section details how to calculate and use MBRs for spatial indexing and efficient querying of geometric data.

See Miscellaneous GIS functions.

Explore miscellaneous GIS functions in MariaDB Server. This section details various SQL functions that support geographic information system operations and spatial data analysis.

See Point Properties.

Learn about POINT properties in MariaDB Server. This section details SQL functions for retrieving attributes of point spatial objects, such as their X and Y coordinates.

See Polygon Properties.

Learn about POLYGON properties in MariaDB Server. This section details SQL functions for retrieving attributes of polygonal spatial objects, such as area, perimeter, and the number of rings.

See WKB.

Learn about Well-Known Binary (WKB) in MariaDB Server. This section details how to represent and store geometric data in a binary format, essential for efficient spatial data exchange and storage.

See WKT.

Learn about Well-Known Text (WKT) in MariaDB Server. This section details how to represent and store geometric data in a text format, essential for human-readable spatial data exchange.

This page is licensed: CC BY-SA / Gnu FDL

Geometry Constructors

- BUFFER
- CONVEXHULL
- GEOMETRYCOLLECTION
- LINESTRING
- MULTILINESTRING
- MULTIPOINT

- MULTIPOLYGON
- POINT
- PointOnSurface
- POLYGON
- ST_BUFFER
- ST_CONVEXHULL
- ST_INTERSECTION
- ST_POINTONSURFACE
- ST_SYMDIFFERENCE
- ST_UNION
- ST_AsGeoJSON
- ST_GeomFromGeoJSON

This page is licensed: CC BY-SA / Gnu FDL

BUFFER

A synonym for ST_BUFFER.

This page is licensed: CC BY-SA / Gnu FDL

CONVEXHULL

A synonym for ST_CONVEXHULL.

This page is licensed: CC BY-SA / Gnu FDL

GEOMETRYCOLLECTION

Syntax

```
GeometryCollection(g1,g2,...)
```

Description

Constructs a WKB GeometryCollection. If any argument is not a well-formed WKB representation of a geometry, the return value is `NULL`.

Examples

```
CREATE TABLE gis_geometrycollection  (g GEOMETRYCOLLECTION);
SHOW FIELDS FROM gis_geometrycollection;
INSERT INTO gis_geometrycollection VALUES
    (GeomCollFromText('GEOMETRYCOLLECTION(POINT(0 0), LINESTRING(0 0,10 10))')),
    (GeometryFromWKB(AsWKB(GeometryCollection(Point(44, 6), LineString(Point(3, 6), Point(7, 9))))),
    (GeomFromText('GeometryCollection()')),
    (GeomFromText('GeometryCollection EMPTY'));
```

```
CREATE TABLE geomcoll_example (
  g GEOMETRYCOLLECTION
);
```

```
INSERT INTO geomcoll_example VALUES
  (ST_GeomCollFromText(
    'GEOMETRYCOLLECTION(POINT(0 0), LINESTRING(0 1, 0 2),
    POINT(1 0))')),
  (GeometryCollection(
    Point(0, 0),
    LineString(Point(0, 1), Point(0, 2))));
```

```
SELECT ST_AsWKT(g) FROM geomcoll_example;

+-------------------------------------------------------------+
| ST_AsWKT(g)                                                 |
+-------------------------------------------------------------+
| GEOMETRYCOLLECTION(POINT(0 0),LINESTRING(0 1,0 2),POINT(1 0)) |
| GEOMETRYCOLLECTION(POINT(0 0),LINESTRING(0 1,0 2))          |
+-------------------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LINESTRING

Syntax

```
LineString(pt1,pt2,...)
```

Description

WKB LineString from WKB Point coordinate data.
Constructs a WKB LineString value from a number of WKB Point arguments. If any argument is not a WKB Point, the return value is `NULL`. If the number of Point arguments is less than two, the return value is `NULL`.

Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT AsText(EndPoint(GeomFromText(@ls)));
+-------------------------------------+
| AsText(EndPoint(GeomFromText(@ls))) |
+-------------------------------------+
| POINT(3 3)                          |
+-------------------------------------+

CREATE TABLE gis_line  (g LINESTRING);
INSERT INTO gis_line VALUES
    (LineFromText('LINESTRING(0 0,0 10,10 0)')),
    (LineStringFromText('LINESTRING(10 10,20 10,20 20,10 20,10 10)')),
    (LineStringFromWKB(AsWKB(LineString(Point(10, 10), Point(40, 10)))));
```

Linestring_Example:

```
CREATE TABLE linestring_example (
  g LINESTRING
);
```

```
INSERT INTO linestring_example VALUES
  (ST_LineFromText('LINESTRING(0 0, 1 1, 2 2)')),
  (ST_LineStringFromText('LINESTRING(10 10, 20 10, 20 20, 10 20, 10 10)')),
  (LineString(Point(10, 10), Point(40, 10)));
```

```
SELECT ST_AsWKT(g) FROM linestring_example;
<</code>>

<<sql>>
+-----------------------------------------+
| ST_AsWKT(g)                             |
+-----------------------------------------+
| LINESTRING(0 0,1 1,2 2)                 |
| LINESTRING(10 10,20 10,20 20,10 20,10 10) |
| LINESTRING(10 10,40 10)                 |
+-----------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MULTILINESTRING

Syntax

```
MultiLineString(ls1,ls2,...)
```

Description

Constructs a WKB MultiLineString value using WKB LineString arguments. If any argument is not a WKB LineString, the return value is `NULL`.

Example

```
CREATE TABLE gis_multi_line (g MULTILINESTRING);
INSERT INTO gis_multi_line VALUES
 (MultiLineStringFromText('MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48))')),
 (MLineFromText('MULTILINESTRING((10 48,10 21,10 0))')),
 (MLineFromWKB(AsWKB(MultiLineString(LineString(Point(1, 2),
    Point(3, 5)), LineString(Point(2, 5),Point(5, 8),Point(21, 7))))));
```

MultiLineString_Example:

```
CREATE TABLE mlstr_example (
  m MULTILINESTRING
);
```

```
INSERT INTO mlstr_example VALUES
  (ST_MultiLineStringFromText(
    'MULTILINESTRING((0 40, 0 20, 6 30, 12 20, 12 40),
    (15 40, 15 20, 25 20, 30 25, 30 35, 25 40, 15 40))')),
  (ST_MLineFromText('MULTILINESTRING((0 0, 1 1, 2 2))')),
  (MultiLineString(
    LineString(Point(0, 40), Point(0, 20)),
    LineString(Point(6, 30), Point(12, 20), Point(12, 40))));
```

```
INSERT INTO mlstr_example VALUES
  (MultiLineStringFromText(
    'MULTILINESTRING((0 40, 0 20, 6 30, 12 20, 12 40),
    (15 40, 15 20, 25 20, 30 25, 30 35, 25 40, 15 40))')),
  (MLineFromText('MULTILINESTRING((0 0, 1 1, 2 2))')),
  (MultiLineString(
    LineString(Point(0, 40), Point(0, 20)),
    LineString(Point(6, 30), Point(12, 20), Point(12, 40))));
```

```
SELECT ST_AsWKT(m) FROM mlstr_example;
```

```
+-----------------------------------------------------------------------------------------------+
| ST_AsWKT(m)                                                                                   |
+-----------------------------------------------------------------------------------------------+
| MULTILINESTRING((0 40,0 20,6 30,12 20,12 40),(15 40,15 20,25 20,30 25,30 35,25 40,15 40)) |
| MULTILINESTRING((0 0,1 1,2 2))                                                                |
| MULTILINESTRING((0 40,0 20),(6 30,12 20,12 40))                                               |
+-----------------------------------------------------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MULTIPOINT

Syntax

```
MultiPoint(pt1,pt2,...)
```

Description

Constructs a WKB MultiPoint value using WKB Point arguments. If any argument is not a WKB Point, the return value is `NULL`.

Examples

```
SET @g = ST_GEOMFROMTEXT('MultiPoint( 1 1, 2 2, 5 3, 7 2, 9 3, 8 4, 6 6, 6 9, 4 9, 1 5 )');

CREATE TABLE gis_multi_point (g MULTIPOINT);
INSERT INTO gis_multi_point VALUES
    (MultiPointFromText('MULTIPOINT(0 0,10 10,10 20,20 20)')),
    (MPointFromText('MULTIPOINT(1 1,11 11,11 21,21 21)')),
    (MPointFromWKB(AsWKB(MultiPoint(Point(3, 6), Point(4, 10)))));
```

MultiPoint_Example:

```
CREATE TABLE multipoint_example (
  m MULTIPOINT
);
```

```
INSERT INTO multipoint_example VALUES
  (ST_MultiPointFromText('MULTIPOINT(0 0, 1 0, 1 1, 0 1)')),
  (ST_MPointFromText('MULTIPOINT(1 1, 2 2, 3 3, 4 4)')),
  (MultiPoint(Point(0, 0), Point(1, 1)));
```

```
INSERT INTO multipoint_example VALUES
  (MultiPointFromText('MULTIPOINT(0 0, 1 0, 1 1, 0 1)')),
  (MPointFromText('MULTIPOINT(1 1, 2 2, 3 3, 4 4)')),
  (MultiPoint(Point(0, 0), Point(1, 1)));
```

```
SELECT ST_AsWKT(m) FROM multipoint_example;
```

```
+----------------------------+
| ST_AsWKT(m)                |
+----------------------------+
| MULTIPOINT(0 0,1 0,1 1,0 1) |
| MULTIPOINT(1 1,2 2,3 3,4 4) |
| MULTIPOINT(0 0,1 1)         |
+----------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MULTIPOLYGON

Syntax

```
MultiPolygon(poly1,poly2,...)
```

Description

Constructs a WKB MultiPolygon value from a set of WKB Polygon arguments. If any argument is not a WKB Polygon, the return value is `NULL`.

Example

```
CREATE TABLE gis_multi_polygon  (g MULTIPOLYGON);
INSERT INTO gis_multi_polygon VALUES
    (MultiPolygonFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),
     ((59 18,67 18,67 13,59 13,59 18)))')),
    (MPolyFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),
        ((59 18,67 18,67 13,59 13,59 18)))')),
    (MPolyFromWKB(AsWKB(MultiPolygon(Polygon(LineString(
       Point(0, 3), Point(3, 3), Point(3, 0), Point(0, 3)))))));
```

MultiPolygon_Example:

```
CREATE TABLE multipolygon_example (
  m MULTIPOLYGON
);
```

```
INSERT INTO multipolygon_example VALUES
  (ST_MultiPolygonFromText(
    'MULTIPOLYGON(((0 40, 0 20, 6 30, 12 20, 12 40, 0 40),
    (15 40, 15 20, 25 20, 30 25, 30 35, 25 40, 15 40)))')),
  (ST_MPolyFromText(
    'MULTIPOLYGON(((-5 45, 35 45, 35 15, -5 15, -5 45),
    (0 40, 0 20, 6 30, 12 20, 12 40, 0 40),
    (15 40, 15 20, 25 20, 30 25, 30 35, 25 40, 15 40)))')),
  (MultiPolygon(Polygon(LineString(Point(0, 0), Point(0, 1),
     Point(1, 1), Point(1, 0), Point(0, 0)))));
```

```
SELECT ST_AsWKT(m) FROM multipolygon_example;
```

```
+-----------------------------------------------------------------------------------------------------------
----------------------+
| ST_AsWKT(m)
|
+-----------------------------------------------------------------------------------------------------------
----------------------+
| MULTIPOLYGON(((0 40,0 20,6 30,12 20,12 40,0 40),(15 40,15 20,25 20,30 25,30 35,25 40,15 40)))
|
| MULTIPOLYGON(((-5 45,35 45,35 15,-5 15,-5 45),(0 40,0 20,6 30,12 20,12 40,0 40),(15 40,15 20,25 20,30
25,30 35,25 40,15 40))) |
| MULTIPOLYGON(((0 0,0 1,1 1,1 0,0 0)))
|
+-----------------------------------------------------------------------------------------------------------
----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

POINT

Syntax

```
Point(x,y)
```

Description

Constructs a WKB Point using the given coordinates.

Examples

```
SET @g = ST_GEOMFROMTEXT('Point(1 1)');

CREATE TABLE gis_point  (g POINT);
INSERT INTO gis_point VALUES
    (PointFromText('POINT(10 10)')),
    (PointFromText('POINT(20 10)')),
    (PointFromText('POINT(20 20)')),
    (PointFromWKB(AsWKB(PointFromText('POINT(10 20)'))));
```

Point_Example:

```
CREATE TABLE point_example (
  p POINT
);
```

```
INSERT INTO point_example VALUES
  (ST_PointFromText('POINT(1 1)')),
  (ST_PointFromText('POINT(2 2)')),
  (Point(3, 3)),
  (Point(4, 4));
```

```
SELECT ST_AsWKT(p) FROM point_example;
```

```
+-------------+
| ST_AsWKT(p) |
+-------------+
| POINT(1 1)  |
| POINT(2 2)  |
| POINT(3 3)  |
| POINT(4 4)  |
+-------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

PointOnSurface

A synonym for ST_PointOnSurface.

This page is licensed: CC BY-SA / Gnu FDL

POLYGON

Syntax

```
Polygon(ls1,ls2,...)
```

Description

Constructs a WKB Polygon value from a number of WKB LineString arguments. If any argument does not represent the WKB of a LinearRing (that is, not a closed and simple LineString) the return value is `NULL`.

Note that according to the OpenGIS standard, a POLYGON should have exactly one ExteriorRing and all other rings should lie within that ExteriorRing and thus be the InteriorRings. Practically, however, some systems, including MariaDB's, permit polygons to have several 'ExteriorRings'. In the case of there being multiple, non-overlapping exterior rings ST_NUMINTERIORRINGS() will return 1.

Examples

```
SET @g = ST_GEOMFROMTEXT('POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1))');

CREATE TABLE gis_polygon   (g POLYGON);
INSERT INTO gis_polygon VALUES
    (PolygonFromText('POLYGON((10 10,20 10,20 20,10 20,10 10))')),
    (PolyFromText('POLYGON((0 0,50 0,50 50,0 50,0 0), (10 10,20 10,20 20,10 20,10 10))')),
    (PolyFromWKB(AsWKB(Polygon(LineString(Point(0, 0), Point(30, 0), Point(30, 30), Point(0, 0))))));
```

Non-overlapping 'polygon':

```
SELECT ST_NumInteriorRings(ST_PolyFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),
  (-1 -1,-5 -1,-5 -5,-1 -5,-1 -1))')) AS NumInteriorRings;
+------------------+
| NumInteriorRings |
+------------------+
|                1 |
+------------------+
```

Polygon_Example:

```
CREATE TABLE polygon_example (
  p POLYGON
);
```

```
INSERT INTO polygon_example VALUES
  (ST_PolygonFromText('POLYGON((0 40, 0 20, 6 30, 12 20, 12 40, 0 40))')),
  (ST_PolyFromText('POLYGON((15 40, 15 20, 25 20, 30 25, 30 35, 25 40, 15 40))')),
  (Polygon(LineString(Point(0, 0), Point(0, 1), Point(1, 1),
  Point(1, 0), Point(0, 0))));
```

```
SELECT ST_AsWKT(p) FROM polygon_example;
```

```
+-----------------------------------------------------+
| ST_AsWKT(p)                                         |
+-----------------------------------------------------+
| POLYGON((0 40,0 20,6 30,12 20,12 40,0 40))          |
| POLYGON((15 40,15 20,25 20,30 25,30 35,25 40,15 40)) |
| POLYGON((0 0,0 1,1 1,1 0,0 0))                      |
+-----------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_BUFFER

Syntax

```
ST_BUFFER(g1,r)
BUFFER(g1,r)
```

Description

Returns a geometry that represents all points whose distance from geometry `g1` is less than or equal to distance, or radius, `r`.

Uses for this function could include creating for example a new geometry representing a buffer zone around an island.

BUFFER() is a synonym.

Examples

Determining whether a point is within a buffer zone:

```
SET @g1 = ST_GEOMFROMTEXT('POLYGON((10 10, 10 20, 20 20, 20 10, 10 10))');

SET @g2 = ST_GEOMFROMTEXT('POINT(8 8)');

SELECT ST_WITHIN(@g2,ST_BUFFER(@g1,5));
+--------------------------------+
| ST_WITHIN(@g2,ST_BUFFER(@g1,5)) |
+--------------------------------+
|                              1 |
+--------------------------------+

SELECT ST_WITHIN(@g2,ST_BUFFER(@g1,1));
+--------------------------------+
| ST_WITHIN(@g2,ST_BUFFER(@g1,1)) |
+--------------------------------+
|                              0 |
+--------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_CONVEXHULL

Syntax

```
ST_ConvexHull(g)
ConvexHull(g)
```

Description

Given a geometry, returns a geometry that is the minimum convex geometry enclosing all geometries within the set. Returns NULL if the geometry value is NULL or an empty value.

ST_ConvexHull() and ConvexHull() are synonyms.

Examples

The ConvexHull of a single point is simply the single point:

```
SET @g = ST_GEOMFROMTEXT('Point(0 0)');

SELECT ST_ASTEXT(ST_CONVEXHULL(@g));
+-----------------------------+
| ST_ASTEXT(ST_CONVEXHULL(@g)) |
+-----------------------------+
| POINT(0 0)                  |
+-----------------------------+
```

```
SET @g = ST_GEOMFROMTEXT('MultiPoint(0 0, 1 2, 2 3)');

SELECT ST_ASTEXT(ST_CONVEXHULL(@g));
+-----------------------------+
| ST_ASTEXT(ST_CONVEXHULL(@g)) |
+-----------------------------+
| POLYGON((0 0,1 2,2 3,0 0))   |
+-----------------------------+
```

```
SET @g = ST_GEOMFROMTEXT('MultiPoint( 1 1, 2 2, 5 3, 7 2, 9 3, 8 4, 6 6, 6 9, 4 9, 1 5 )');

SELECT ST_ASTEXT(ST_CONVEXHULL(@g));
+---------------------------------------+
| ST_ASTEXT(ST_CONVEXHULL(@g))          |
+---------------------------------------+
| POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1)) |
+---------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_INTERSECTION

Syntax

```
ST_INTERSECTION(g1,g2)
```

Description

Returns a geometry that is the intersection, or shared portion, of geometry `g1` and geometry `g2`.

Examples

```
SET @g1 = ST_GEOMFROMTEXT('POINT(2 1)');

SET @g2 = ST_GEOMFROMTEXT('LINESTRING(2 1, 0 2)');

SELECT ASTEXT(ST_INTERSECTION(@g1,@g2));
+---------------------------------+
| ASTEXT(ST_INTERSECTION(@g1,@g2)) |
+---------------------------------+
| POINT(2 1)                      |
+---------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_POINTONSURFACE

Syntax

```
ST_PointOnSurface(g)
PointOnSurface(g)
```

Description

Given a geometry, returns a POINT guaranteed to intersect a surface. However, see MDEV-7514.

ST_PointOnSurface() and PointOnSurface() are synonyms.

This page is licensed: CC BY-SA / Gnu FDL

ST_SYMDIFFERENCE

Syntax

```
ST_SYMDIFFERENCE(g1,g2)
```

Description

Returns a geometry that represents the portions of geometry `g1` and geometry `g2` that don't intersect.

Examples

```
SET @g1 = ST_GEOMFROMTEXT('LINESTRING(10 20, 10 40)');

SET @g2 = ST_GEOMFROMTEXT('LINESTRING(10 15, 10 25)');

SELECT ASTEXT(ST_SYMDIFFERENCE(@g1,@g2));
+--------------------------------------------+
| ASTEXT(ST_SYMDIFFERENCE(@g1,@g2))          |
+--------------------------------------------+
| MULTILINESTRING((10 15,10 20),(10 25,10 40)) |
+--------------------------------------------+

SET @g2 = ST_GeomFromText('LINESTRING(10 20, 10 41)');

SELECT ASTEXT(ST_SYMDIFFERENCE(@g1,@g2));
+-----------------------------------+
| ASTEXT(ST_SYMDIFFERENCE(@g1,@g2)) |
+-----------------------------------+
| LINESTRING(10 40,10 41)           |
+-----------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_UNION

Syntax

```
ST_UNION(g1,g2)
```

Description

Returns a geometry that is the union of the geometry `g1` and geometry `g2`.

Examples

```
SET @g1 = GEOMFROMTEXT('POINT (0 2)');

SET @g2 = GEOMFROMTEXT('POINT (2 0)');

SELECT ASTEXT(ST_UNION(@g1,@g2));
+---------------------------+
| ASTEXT(ST_UNION(@g1,@g2)) |
+---------------------------+
| MULTIPOINT(2 0,0 2)       |
+---------------------------+
```

```
SET @g1 = GEOMFROMTEXT('POLYGON((0 0,0 3,3 3,3 0,0 0))');

SET @g2 = GEOMFROMTEXT('POLYGON((2 2,4 2,4 4,2 4,2 2))');

SELECT ASTEXT(ST_UNION(@g1,@g2));
+---------------------------------------------+
| ASTEXT(ST_UNION(@g1,@g2))                   |
+---------------------------------------------+
| POLYGON((0 0,0 3,2 3,2 4,4 4,4 2,3 2,3 0,0 0)) |
+---------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_AsGeoJSON

Syntax

```
ST_AsGeoJSON(g[, max_decimals[, options]])
```

Description

Returns the given geometry *g* as a GeoJSON element. The optional *max_decimals* limits the maximum number of decimals displayed.

The optional *options* flag can be set to `1` to add a bounding box to the output.

Examples

```
SELECT ST_AsGeoJSON(ST_GeomFromText('POINT(5.3 7.2)'));
+-------------------------------------------------+
| ST_AsGeoJSON(ST_GeomFromText('POINT(5.3 7.2)')) |
+-------------------------------------------------+
| {"type": "Point", "coordinates": [5.3, 7.2]}    |
+-------------------------------------------------+
```

See Also

- ST_GeomFromGeoJSON

This page is licensed: CC BY-SA / Gnu FDL

ST_GeomFromGeoJSON

Syntax

```
ST_GeomFromGeoJSON(g[, option])
```

Description

Given a GeoJSON input *g*, returns a geometry object. The *option* specifies what to do if *g* contains geometries with coordinate dimensions higher than 2.

Option	Description
1	Return an error (the default)
2 - 4	The document is accepted, but the coordinates for higher coordinate dimensions are stripped off.

Examples

```
SET @j = '{ "type": "Point", "coordinates": [5.3, 15.0]}';

SELECT ST_AsText(ST_GeomFromGeoJSON(@j));
+-----------------------------------+
| ST_AsText(ST_GeomFromGeoJSON(@j)) |
+-----------------------------------+
| POINT(5.3 15)                     |
+-----------------------------------+
```

See Also

- ST_AsGeoJSON

This page is licensed: CC BY-SA / Gnu FDL

Geometry Properties

- BOUNDARY
- DIMENSION
- ENVELOPE
- GeometryN
- GeometryType
- IsClosed
- IsEmpty
- IsRing
- IsSimple
- NumGeometries
- SRID
- ST_BOUNDARY
- ST_DIMENSION
- ST_ENVELOPE
- ST_GEOMETRYN
- ST_GEOMETRYTYPE
- ST_ISCLOSED
- ST_ISEMPTY
- ST_IsRing
- ST_IsSimple
- ST_NUMGEOMETRIES
- ST_RELATE
- ST_SRID

This page is licensed: CC BY-SA / Gnu FDL

BOUNDARY

A synonym for ST_BOUNDARY.

This page is licensed: CC BY-SA / Gnu FDL

DIMENSION

A synonym for ST_DIMENSION.

This page is licensed: CC BY-SA / Gnu FDL

ENVELOPE

A synonym for ST_ENVELOPE.

This page is licensed: CC BY-SA / Gnu FDL

GeometryN

A synonym for ST_GeometryN.

This page is licensed: CC BY-SA / Gnu FDL

GeometryType

A synonym for ST_GeometryType.

This page is licensed: CC BY-SA / Gnu FDL

IsClosed

A synonym for ST_IsClosed.

This page is licensed: CC BY-SA / Gnu FDL

IsEmpty

A synonym for ST_IsEmpty.

This page is licensed: CC BY-SA / Gnu FDL

IsRing

A synonym for ST_IsRing.

This page is licensed: CC BY-SA / Gnu FDL

IsSimple

A synonym for ST_IsSImple.

This page is licensed: CC BY-SA / Gnu FDL

NumGeometries

A synonym for ST_NumGeometries.

This page is licensed: CC BY-SA / Gnu FDL

SRID

A synonym for ST_SRID.

This page is licensed: CC BY-SA / Gnu FDL

ST_BOUNDARY

Syntax

```
ST_BOUNDARY(g)
BOUNDARY(g)
```

Description

Returns a geometry that is the closure of the combinatorial boundary of the geometry value `g`.

BOUNDARY() is a synonym.

Examples

```
SELECT ST_AsText(ST_Boundary(ST_GeomFromText('LINESTRING(3 3,0 0, -3 3)')));
+---------------------------------------------------------------------+
| ST_AsText(ST_Boundary(ST_GeomFromText('LINESTRING(3 3,0 0, -3 3)'))) |
+---------------------------------------------------------------------+
| MULTIPOINT(3 3,-3 3)                                                |
+---------------------------------------------------------------------+

SELECT ST_AsText(ST_Boundary(ST_GeomFromText('POLYGON((3 3,0 0, -3 3, 3 3))')));
+------------------------------------------------------------------------+
| ST_AsText(ST_Boundary(ST_GeomFromText('POLYGON((3 3,0 0, -3 3, 3 3))'))) |
+------------------------------------------------------------------------+
| LINESTRING(3 3,0 0,-3 3,3 3)                                           |
+------------------------------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_DIMENSION

Syntax

```
ST_Dimension(g)
Dimension(g)
```

Description

Returns the inherent dimension of the geometry value `g`. The result can be

Dimension	Definition
-1	empty geometry
0	geometry with no length or area
1	geometry with no area but nonzero length
2	geometry with nonzero area

`ST_Dimension()` and `Dimension()` are synonyms.

Examples

```
SELECT Dimension(GeomFromText('LineString(1 1,2 2)'));
+-----------------------------------------------+
| Dimension(GeomFromText('LineString(1 1,2 2)')) |
+-----------------------------------------------+
|                                             1 |
+-----------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_ENVELOPE

Syntax

```
ST_ENVELOPE(g)
ENVELOPE(g)
```

Description

Returns the Minimum Bounding Rectangle (MBR) for the geometry value `g`. The result is returned as a Polygon value.

The polygon is defined by the corner points of the bounding box:

```
POLYGON((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
```

`ST_ENVELOPE()` and `ENVELOPE()` are synonyms.

Examples

```
SELECT AsText(ST_ENVELOPE(GeomFromText('LineString(1 1,4 4)')));
+-----------------------------------------------------------+
| AsText(ST_ENVELOPE(GeomFromText('LineString(1 1,4 4)'))) |
+-----------------------------------------------------------+
| POLYGON((1 1,4 1,4 4,1 4,1 1))                            |
+-----------------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_GEOMETRYN

Syntax

```
ST_GeometryN(gc,N)
GeometryN(gc,N)
```

Description

Returns the N-th geometry in the GeometryCollection *gc*. Geometries are numbered beginning with 1.

`ST_GeometryN()` and `GeometryN()` are synonyms.

Example

```
SET @gc = 'GeometryCollection(Point(1 1),LineString(12 14, 9 11))';

SELECT AsText(GeometryN(GeomFromText(@gc),1));
+--------------------------------------+
| AsText(GeometryN(GeomFromText(@gc),1)) |
+--------------------------------------+
| POINT(1 1)                           |
+--------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_GEOMETRYTYPE

Syntax

```
ST_GeometryType(g)
GeometryType(g)
```

Description

Returns as a string the name of the geometry type of which the geometry instance `g` is a member. The name corresponds to one of the instantiable Geometry subclasses.

`ST_GeometryType()` and `GeometryType()` are synonyms.

Examples

```
SELECT GeometryType(GeomFromText('POINT(1 1)'));
+---------------------------------------+
| GeometryType(GeomFromText('POINT(1 1)')) |
+---------------------------------------+
| POINT                                 |
+---------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_ISCLOSED

Syntax

```
ST_IsClosed(g)
IsClosed(g)
```

Description

Returns 1 if a given LINESTRING's start and end points are the same, or 0 if they are not the same.

`ST_IsClosed()` and `IsClosed()` are synonyms.

Examples

```
SET @ls = 'LineString(0 0, 0 4, 4 4, 0 0)';
SELECT ST_ISCLOSED(GEOMFROMTEXT(@ls));
+--------------------------------+
| ST_ISCLOSED(GEOMFROMTEXT(@ls)) |
+--------------------------------+
|                              1 |
+--------------------------------+

SET @ls = 'LineString(0 0, 0 4, 4 4, 0 1)';
SELECT ST_ISCLOSED(GEOMFROMTEXT(@ls));
+--------------------------------+
| ST_ISCLOSED(GEOMFROMTEXT(@ls)) |
+--------------------------------+
|                              0 |
+--------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_ISEMPTY

Syntax

```
ST_IsEmpty(g)
IsEmpty(g)
```

Description

IsEmpty is a function defined by the OpenGIS specification, but is not fully implemented by MariaDB or MySQL.

Since MariaDB and MySQL do not support GIS EMPTY values such as POINT EMPTY, as implemented it simply returns `1` if the geometry value *`g`* is invalid, `0` if it is valid, and `NULL` if the argument is `NULL`.

`ST_IsEmpty()` and `IsEmpty()` are synonyms.

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_IsRing

Syntax

```
ST_IsRing(g)
IsRing(g)
```

Description

Returns true if a given LINESTRING is a ring, that is, both ST_IsClosed and ST_IsSimple. A simple curve does not pass through the same point more than once. However, see MDEV-7510.

`St_IsRing()` and `IsRing()` are synonyms.

This page is licensed: CC BY-SA / Gnu FDL

ST_IsSimple

Syntax

```
ST_IsSimple(g)
IsSimple(g)
```

Description

Returns true if the given Geometry has no anomalous geometric points, false if it does, or NULL if given a NULL value.

ST_IsSimple() and IsSimple() are synonyms.

Examples

A POINT is always simple.

```
SET @g = 'Point(1 2)';

SELECT ST_ISSIMPLE(GEOMFROMTEXT(@g));
+------------------------------+
| ST_ISSIMPLE(GEOMFROMTEXT(@g)) |
+------------------------------+
|                            1 |
+------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_NUMGEOMETRIES

Syntax

```
ST_NumGeometries(gc)
NumGeometries(gc)
```

Description

Returns the number of geometries in the GeometryCollection `gc`.

`ST_NumGeometries()` and `NumGeometries()` are synonyms.

Example

```
SET @gc = 'GeometryCollection(Point(1 1),LineString(2 2, 3 3))';

SELECT NUMGEOMETRIES(GeomFromText(@gc));
+----------------------------------+
| NUMGEOMETRIES(GeomFromText(@gc)) |
+----------------------------------+
|                                2 |
+----------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_RELATE

Syntax

```
ST_Relate(g1, g2, i)
```

Description

Returns true if Geometry `g1` is spatially related to Geometry `g2` by testing for intersections between the interior, boundary and exterior of the two geometries as specified by the values in intersection matrix pattern `i`.

This page is licensed: CC BY-SA / Gnu FDL

ST_SRID

Syntax

```
ST_SRID(g)
SRID(g)
```

Description

Returns an integer indicating the Spatial Reference System ID for the geometry value g.

In MariaDB, the SRID value is just an integer associated with the geometry value. All calculations are done assuming Euclidean (planar) geometry.

`ST_SRID()` and `SRID()` are synonyms.

Examples

```
SELECT SRID(GeomFromText('LineString(1 1,2 2)',101));
+---------------------------------------------+
| SRID(GeomFromText('LineString(1 1,2 2)',101)) |
+---------------------------------------------+
|                                         101 |
+---------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

Geometry Relations

- CONTAINS
- CROSSES
- DISJOINT
- EQUALS
- INTERSECTS
- OVERLAPS
- ST_CONTAINS
- ST_CROSSES
- ST_DIFFERENCE
- ST_DISJOINT
- ST_DISTANCE
- ST_DISTANCE_SPHERE
- ST_EQUALS
- ST_INTERSECTS
- ST_LENGTH
- ST_OVERLAPS
- ST_TOUCHES
- ST_WITHIN
- TOUCHES
- WITHIN

This page is licensed: CC BY-SA / Gnu FDL

CONTAINS

Syntax

```
Contains(g1,g2)
```

Description

Returns `1` or `0` to indicate whether a geometry `g1` completely contains geometry `g2`. CONTAINS() is based on the original MySQL implementation and uses object bounding rectangles, while ST_CONTAINS() uses object shapes.

This tests the opposite relationship to Within().

This page is licensed: GPLv2, originally from fill_help_tables.sql

CROSSES

Syntax

```
Crosses(g1,g2)
```

Description

Returns `1` if `g1` spatially crosses `g2`. Returns `NULL` if `g1` is a Polygon or a MultiPolygon, or if `g2` is a Point or a MultiPoint. Otherwise, returns `0`.

The term spatially crosses denotes a spatial relation between two given geometries that has the following properties:

- The two geometries intersect.
- Their intersection results in a geometry that has a dimension that is one less than the maximum dimension of the two given geometries.
- Their intersection is not equal to either of the two given geometries.

`CROSSES()` is based on the original MySQL implementation, and uses object bounding rectangles, while ST_CROSSES() uses object shapes.

This page is licensed: GPLv2, originally from fill_help_tables.sql

DISJOINT

Syntax

```
Disjoint(g1,g2)
```

Description

Returns `1` or `0` to indicate whether `g1` is spatially disjoint from (does not intersect) `g2`.

`DISJOINT()` tests the opposite relationship to INTERSECTS().

`DISJOINT()` is based on the original MySQL implementation and uses object bounding rectangles, while ST_DISJOINT() uses object shapes.

This page is licensed: GPLv2, originally from fill_help_tables.sql

EQUALS

Syntax

```
MBREQUALS(g1,g2)
```

Description

Returns `1` or `0` to indicate whether *`g1`* is spatially equal to *`g2`*.

EQUALS() is based on the original MySQL implementation and uses object bounding rectangles, while ST_EQUALS() uses object shapes.

`MBREQUALS` is a synonym for `Equals`.

This page is licensed: GPLv2, originally from fill_help_tables.sql

INTERSECTS

Syntax

```
INTERSECTS(g1,g2)
```

Description

Returns `1` or `0` to indicate whether geometry *`g1`* spatially intersects geometry *`g2`*.

INTERSECTS() is based on the original MySQL implementation and uses object bounding rectangles, while ST_INTERSECTS() uses object shapes.

`INTERSECTS()` tests the opposite relationship to DISJOINT().

This page is licensed: GPLv2, originally from fill_help_tables.sql

OVERLAPS

Syntax

```
OVERLAPS(g1,g2)
```

Description

Returns `1` or `0` to indicate whether `g1` spatially overlaps `g2`.
The term spatially overlaps is used if two geometries of equal dimensions intersect and their intersection results in a geometry of the same dimension but not equal to either of the given geometries.

`OVERLAPS()` is based on the original MySQL implementation and uses object bounding rectangles, while ST_OVERLAPS() uses object shapes.

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_CONTAINS

Syntax

```
ST_CONTAINS(g1,g2)
```

Description

Returns `1` or `0` to indicate whether a geometry `g1` completely contains geometry `g2`.

ST_CONTAINS() uses object shapes, while CONTAINS(), based on the original MySQL implementation, uses object bounding rectangles.

ST_CONTAINS tests the opposite relationship to ST_WITHIN().

Examples

```
SET @g1 = ST_GEOMFROMTEXT('POLYGON((175 150, 20 40, 50 60, 125 100, 175 150))');

SET @g2 = ST_GEOMFROMTEXT('POINT(174 149)');

SELECT ST_CONTAINS(@g1,@g2);
+----------------------+
| ST_CONTAINS(@g1,@g2) |
+----------------------+
|                    1 |
+----------------------+

SET @g2 = ST_GEOMFROMTEXT('POINT(175 151)');

SELECT ST_CONTAINS(@g1,@g2);
+----------------------+
| ST_CONTAINS(@g1,@g2) |
+----------------------+
|                    0 |
+----------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_CROSSES

Syntax

```
ST_CROSSES(g1,g2)
```

Description

Returns `1` if geometry *`g1`* spatially crosses geometry *`g2`*. Returns `NULL` if `g1` is a Polygon or a MultiPolygon, or if `g2` is a Point or a MultiPoint. Otherwise, returns `0`.

The term spatially crosses denotes a spatial relation between two given geometries that has the following properties:

- The two geometries intersect.
- Their intersection results in a geometry that has a dimension that is one less than the maximum dimension of the two given geometries.
- Their intersection is not equal to either of the two given geometries.

`ST_CROSSES()` uses object shapes, while CROSSES(), based on the original MySQL implementation, uses object bounding rectangles.

Examples

```
SET @g1 = ST_GEOMFROMTEXT('LINESTRING(174 149, 176 151)');

SET @g2 = ST_GEOMFROMTEXT('POLYGON((175 150, 20 40, 50 60, 125 100, 175 150))');

SELECT ST_CROSSES(@g1,@g2);
+---------------------+
| ST_CROSSES(@g1,@g2) |
+---------------------+
|                   1 |
+---------------------+

SET @g1 = ST_GEOMFROMTEXT('LINESTRING(176 149, 176 151)');

SELECT ST_CROSSES(@g1,@g2);
+---------------------+
| ST_CROSSES(@g1,@g2) |
+---------------------+
|                   0 |
+---------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_DIFFERENCE

Syntax

```
ST_DIFFERENCE(g1,g2)
```

Description

Returns a geometry representing the point set difference of the given geometry values.

Example

```
SET @g1 = POINT(10,10), @g2 = POINT(20,20);

SELECT ST_AsText(ST_Difference(@g1, @g2));
+-----------------------------------+
| ST_AsText(ST_Difference(@g1, @g2)) |
+-----------------------------------+
| POINT(10 10)                      |
+-----------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_DISJOINT

Syntax

```
ST_DISJOINT(g1,g2)
```

Description

Returns `1` or `0` to indicate whether geometry *`g1`* is spatially disjoint from (does not intersect with) geometry *`g2`*.

`ST_DISJOINT()` uses object shapes, while DISJOINT(), based on the original MySQL implementation, uses object bounding rectangles.

`ST_DISJOINT()` tests the opposite relationship to ST_INTERSECTS().

Examples

```
SET @g1 = ST_GEOMFROMTEXT('POINT(0 0)');

SET @g2 = ST_GEOMFROMTEXT('LINESTRING(2 0, 0 2)');

SELECT ST_DISJOINT(@g1,@g2);
+----------------------+
| ST_DISJOINT(@g1,@g2) |
+----------------------+
|                    1 |
+----------------------+

SET @g2 = ST_GEOMFROMTEXT('LINESTRING(0 0, 0 2)');

SELECT ST_DISJOINT(@g1,@g2);
+----------------------+
| ST_DISJOINT(@g1,@g2) |
+----------------------+
|                    0 |
+----------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_DISTANCE

Syntax

```
ST_DISTANCE(g1,g2)
```

Description

Returns the distance between two geometries, or null if not given valid inputs.

Example

```
SELECT ST_Distance(POINT(1,2),POINT(2,2));
+-----------------------------------+
| ST_Distance(POINT(1,2),POINT(2,2)) |
+-----------------------------------+
|                                 1 |
+-----------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_DISTANCE_SPHERE

Syntax

```
ST_DISTANCE_SPHERE(g1,g2,[r])
```

Description

Returns the spherical distance in meters between two geometries (point or multipoint) on a sphere. The optional radius *r* is in meters, must be positive, and defaults to the Earth's radius (6370986 meters) if not specified. If either of the two geometries are not valid, `NULL` is returned.

Example

```
SET @zenica   = ST_GeomFromText('POINT(17.907743 44.203438)');
SET @sarajevo = ST_GeomFromText('POINT(18.413076 43.856258)');
SELECT ST_Distance_Sphere(@zenica, @sarajevo);
+--------------------------------------+
| ST_Distance_Sphere(@zenica, @sarajevo) |
+--------------------------------------+
|                     55878.59337591705 |
+--------------------------------------+

SELECT ST_Distance_Sphere(@zenica, @sarajevo, 6370986);
+------------------------------------------------+
| ST_Distance_Sphere(@zenica, @sarajevo, 6370986) |
+------------------------------------------------+
|                              55878.59337591705 |
+------------------------------------------------+

SELECT ST_Distance_Sphere(@zenica, @sarajevo, 200);
+-------------------------------------------+
| ST_Distance_Sphere(@zenica, @sarajevo, 200) |
+-------------------------------------------+
|                          1.754158410516584 |
+-------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_EQUALS

Syntax

```
ST_EQUALS(g1,g2)
```

Description

Returns `1` or `0` to indicate whether geometry *`g1`* is spatially equal to geometry *`g2`*.

`ST_EQUALS()` uses object shapes, while EQUALS(), based on the original MySQL implementation, uses object bounding rectangles.

Examples

```
SET @g1 = ST_GEOMFROMTEXT('LINESTRING(174 149, 176 151)');

SET @g2 = ST_GEOMFROMTEXT('LINESTRING(176 151, 174 149)');

SELECT ST_EQUALS(@g1,@g2);
+--------------------+
| ST_EQUALS(@g1,@g2) |
+--------------------+
|                  1 |
+--------------------+
```

```
SET @g1 = ST_GEOMFROMTEXT('POINT(0 2)');

SET @g1 = ST_GEOMFROMTEXT('POINT(2 0)');

SELECT ST_EQUALS(@g1,@g2);
+--------------------+
| ST_EQUALS(@g1,@g2) |
+--------------------+
|                  0 |
+--------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_INTERSECTS

Syntax

```
ST_INTERSECTS(g1,g2)
```

Description

Returns `1` or `0` to indicate whether geometry *`g1`* spatially intersects geometry *`g2`*.

`ST_INTERSECTS()` uses object shapes, while INTERSECTS(), based on the original MySQL implementation, uses object bounding rectangles.

`ST_INTERSECTS()` tests the opposite relationship to ST_DISJOINT().

Examples

```
SET @g1 = ST_GEOMFROMTEXT('POINT(0 0)');

SET @g2 = ST_GEOMFROMTEXT('LINESTRING(0 0, 0 2)');

SELECT ST_INTERSECTS(@g1,@g2);
+------------------------+
| ST_INTERSECTS(@g1,@g2) |
+------------------------+
|                      1 |
+------------------------+
```

```
SET @g2 = ST_GEOMFROMTEXT('LINESTRING(2 0, 0 2)');

SELECT ST_INTERSECTS(@g1,@g2);
+------------------------+
| ST_INTERSECTS(@g1,@g2) |
+------------------------+
|                      0 |
+------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_LENGTH

Syntax

```
ST_LENGTH(ls)
```

Description

Returns as a double-precision number the length of the LineString value *`ls`* in its associated spatial reference.

Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT ST_LENGTH(ST_GeomFromText(@ls));
+---------------------------------+
| ST_LENGTH(ST_GeomFromText(@ls)) |
+---------------------------------+
|               2.82842712474619 |
+---------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_OVERLAPS

Syntax

```
ST_OVERLAPS(g1,g2)
```

Description

Returns `1` or `0` to indicate whether geometry *`g1`* spatially overlaps geometry *`g2`*.

The term spatially overlaps is used if two geometries of equal dimensions intersect and their intersection results in a geometry of the same dimension but not equal to either of the given geometries.

`ST_OVERLAPS()` uses object shapes, while OVERLAPS(), based on the original MySQL implementation, uses object bounding rectangles.

This page is licensed: CC BY-SA / Gnu FDL

ST_TOUCHES

Syntax

```
ST_TOUCHES(g1,g2)
```

Description

Returns `1` or `0` to indicate whether geometry *`g1`* spatially touches geometry *`g2`*. Two geometries spatially touch if the interiors of the geometries do not intersect, but the boundary of one of the geometries intersects either the boundary or the interior of the other.

ST_TOUCHES() uses object shapes, while TOUCHES(), based on the original MySQL implementation, uses object bounding rectangles.

Examples

```
SET @g1 = ST_GEOMFROMTEXT('POINT(2 0)');

SET @g2 = ST_GEOMFROMTEXT('LINESTRING(2 0, 0 2)');

SELECT ST_TOUCHES(@g1,@g2);
+---------------------+
| ST_TOUCHES(@g1,@g2) |
+---------------------+
|                   1 |
+---------------------+

SET @g1 = ST_GEOMFROMTEXT('POINT(2 1)');

SELECT ST_TOUCHES(@g1,@g2);
+---------------------+
| ST_TOUCHES(@g1,@g2) |
+---------------------+
|                   0 |
+---------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_WITHIN

Syntax

```
ST_WITHIN(g1,g2)
```

Description

Returns `1` or `0` to indicate whether geometry *`g1`* is spatially within geometry *`g2`*.

This tests the opposite relationship as ST_CONTAINS().

`ST_WITHIN()` uses object shapes, while WITHIN(), based on the original MySQL implementation, uses object bounding rectangles.

Examples

```
SET @g1 = ST_GEOMFROMTEXT('POINT(174 149)');

SET @g2 = ST_GEOMFROMTEXT('POLYGON((175 150, 20 40, 50 60, 125 100, 175 150))');

SELECT ST_WITHIN(@g1,@g2);
+--------------------+
| ST_WITHIN(@g1,@g2) |
+--------------------+
|                  1 |
+--------------------+

SET @g1 = ST_GEOMFROMTEXT('POINT(176 151)');

SELECT ST_WITHIN(@g1,@g2);
+--------------------+
| ST_WITHIN(@g1,@g2) |
+--------------------+
|                  0 |
+--------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

TOUCHES

Syntax

```
Touches(g1,g2)
```

Description

Returns `1` or `0` to indicate whether `g1` spatially touches `g2`. Two geometries spatially touch if the interiors of the geometries do not intersect, but the boundary of one of the geometries intersects either the boundary or the interior of the other.

`TOUCHES()` is based on the original MySQL implementation and uses object bounding rectangles, while ST_TOUCHES() uses object shapes.

This page is licensed: GPLv2, originally from fill_help_tables.sql

WITHIN

Syntax

```
Within(g1,g2)
```

Description

Returns `1` or `0` to indicate whether `g1` is spatially within `g2`. This tests the opposite relationship as Contains().

`WITHIN()` is based on the original MySQL implementation, and uses object bounding rectangles, while ST_WITHIN() uses object shapes.

Examples

```
SET @g1 = GEOMFROMTEXT('POINT(174 149)');
SET @g2 = GEOMFROMTEXT('POINT(176 151)');
SET @g3 = GEOMFROMTEXT('POLYGON((175 150, 20 40, 50 60, 125 100, 175 150))');

SELECT within(@g1,@g3);
+------------------+
| within(@g1,@g3) |
+------------------+
|               1 |
+------------------+

SELECT within(@g2,@g3);
+------------------+
| within(@g2,@g3) |
+------------------+
|               0 |
+------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LineString Properties

See ENDPOINT.

Synonym for ST_ENDPOINT. Returns the last point of a LineString geometry.

See GLENGTH.

Synonym for ST_LENGTH. Calculates the length of a LineString or MultiLineString in its associated spatial reference units.

See NumPoints.

Synonym for ST_NUMPOINTS. Returns the number of points in a LineString geometry.

See PointN.

Synonym for ST_POINTN. Returns the N-th point in a LineString geometry, where N is a 1-based index.

See STARTPOINT.

Synonym for ST_STARTPOINT. Returns the first point of a LineString geometry.

See ST_ENDPOINT.

Returns the end Point of a LineString. This function retrieves the final coordinate in the linear geometry sequence.

See ST_NUMPOINTS.

Returns the count of Points in a LineString. This function calculates the total number of vertices defining the line.

See ST_POINTN.

Returns the N-th Point in a LineString. This function retrieves a specific point from the sequence based on its 1-based index.

See ST_STARTPOINT.

Returns the start Point of a LineString. This function retrieves the initial coordinate in the linear geometry sequence.

This page is licensed: CC BY-SA / Gnu FDL

ENDPOINT

A synonym for ST_ENDPOINT.

This page is licensed: CC BY-SA / Gnu FDL

GLENGTH

Syntax

```
GLength(ls)
```

Description

Returns as a double-precision number the length of the LineString value `ls` in its associated spatial reference.

Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT GLength(GeomFromText(@ls));
+---------------------------+
| GLength(GeomFromText(@ls)) |
+---------------------------+
|           2.82842712474619 |
+---------------------------+
```

See Also

- ST_LENGTH() is the OpenGIS equivalent.

This page is licensed: GPLv2, originally from fill_help_tables.sql

NumPoints

A synonym for ST_NumPoints.

This page is licensed: CC BY-SA / Gnu FDL

PointN

A synonym for ST_PointN.

This page is licensed: CC BY-SA / Gnu FDL

STARTPOINT

A synonym for ST_STARTPOINT.

This page is licensed: CC BY-SA / Gnu FDL

ST_ENDPOINT

Syntax

```
ST_EndPoint(ls)
EndPoint(ls)
```

Description

Returns the Point that is the endpoint of the LineString value `ls`.

`ST_EndPoint()` and `EndPoint()` are synonyms.

Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT AsText(EndPoint(GeomFromText(@ls)));
+-------------------------------------+
| AsText(EndPoint(GeomFromText(@ls))) |
+-------------------------------------+
| POINT(3 3)                          |
+-------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_NUMPOINTS

Syntax

```
ST_NumPoints(ls)
NumPoints(ls)
```

Description

Returns the number of Point objects in the LineString value `ls`.

`ST_NumPoints()` and `NumPoints()` are synonyms.

Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT NumPoints(GeomFromText(@ls));
+------------------------------+
| NumPoints(GeomFromText(@ls)) |
+------------------------------+
|                            3 |
+------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_POINTN

Syntax

```
ST_PointN(ls,N)
PointN(ls,N)
```

Description

Returns the N-th Point in the LineString value `ls`. Points are numbered beginning with `1`.

`ST_PointN()` and `PointN()` are synonyms.

Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT AsText(PointN(GeomFromText(@ls),2));
+-------------------------------------+
| AsText(PointN(GeomFromText(@ls),2)) |
+-------------------------------------+
| POINT(2 2)                          |
+-------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_STARTPOINT

Syntax

```
ST_StartPoint(ls)
StartPoint(ls)
```

Description

Returns the Point that is the start point of the LineString value `ls`.

`ST_StartPoint()` and `StartPoint()` are synonyms.

Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT AsText(StartPoint(GeomFromText(@ls)));
+--------------------------------------+
| AsText(StartPoint(GeomFromText(@ls))) |
+--------------------------------------+
| POINT(1 1)                           |
+--------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MBR (Minimum Bounding Rectangle)

See MBR Definition.

Understand Minimum Bounding Rectangles. An MBR is the smallest rectangle that completely encloses a geometry, defined by its minimum and maximum X and Y coordinates.

See MBRContains.

Check if one MBR contains another. Returns 1 if the Minimum Bounding Rectangle of the first geometry completely encloses the MBR of the second geometry.

See MBRCoveredBy.

Check if one MBR is covered by another. Returns 1 if the MBR of the first geometry is entirely contained within the MBR of the second geometry.

See MBRDisjoint.

Check if two MBRs are disjoint. Returns 1 if the Minimum Bounding Rectangles of the two geometries do not intersect or touch at all.

See MBREqual.

Check if two MBRs are identical. Returns 1 if the Minimum Bounding Rectangles of both geometries share the exact same coordinates.

See MBRIntersects.

Check if two MBRs intersect. Returns 1 if the Minimum Bounding Rectangles of the geometries share any portion of space, including boundaries.

See MBROverlaps.

Check if two MBRs overlap. Returns 1 if the MBRs intersect but neither completely contains the other, and they have the same dimension.

See MBRTouches.

Check if two MBRs touch. Returns 1 if the MBRs intersect only at their boundaries and do not share any interior points.

See MBRWithin.

Check if one MBR is within another. Returns 1 if the MBR of the first geometry is completely enclosed by the MBR of the second geometry.

This page is licensed: CC BY-SA / Gnu FDL

MBR Definition

Description

The MBR (Minimum Bounding Rectangle), or Envelope is the bounding geometry, formed by the minimum and maximum (X,Y) coordinates:

Examples

```
((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MBRContains

Syntax

```
MBRContains(g1,g2)
```

Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangle of g1 contains the Minimum Bounding Rectangle of g2. This tests the opposite relationship as MBRWithin().

Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');

SET @g2 = GeomFromText('Point(1 1)');

SELECT MBRContains(@g1,@g2), MBRContains(@g2,@g1);
+----------------------+----------------------+
| MBRContains(@g1,@g2) | MBRContains(@g2,@g1) |
+----------------------+----------------------+
|                    1 |                    0 |
+----------------------+----------------------+
```

See Also

- MBRWithin

This page is licensed: GPLv2, originally from fill_help_tables.sql

MBRCoveredBy

> **Note:**
>
> MBRCoveredBy is available from MariaDB 12.0.

Syntax

```
MBRCoveredBy(g1, g2)
```

Description

Returns 1 if the minimum bounding rectangle of `g1` is covered by the minimum bounding rectangle of `g2`, otherwise 0.

Returns `NULL` If any argument is `NULL`, or an argument is an empty geometry.

Examples

```
SET @g1a = ST_GeomFromText('Point(5 6)');
SET @g1b = ST_GeomFromText('Point(5 11)');
SET @g2 = ST_GeomFromText('Polygon((0 0,0 10,10 10,10 0,0 0))');

SELECT MBRCoveredby(@g1a,@g2), MBRCoveredby(@g1b,@g2);
+------------------------+------------------------+
| MBRCoveredby(@g1a,@g2) | MBRCoveredby(@g1b,@g2) |
+------------------------+------------------------+
|                      1 |                      0 |
+------------------------+------------------------+
```

See Also

This page is licensed: CC BY-SA / Gnu FDL

MBRDisjoint

Syntax

```
MBRDisjoint(g1,g2)
```

Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries g1 and g2 are disjoint. Two geometries are disjoint if they do not intersect, that is touch or overlap.

Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((4 4,4 7,7 7,7 4,4 4))');
SELECTmbrdisjoint(@g1,@g2);
+----------------------+
| mbrdisjoint(@g1,@g2) |
+----------------------+
|                    1 |
+----------------------+

SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbrdisjoint(@g1,@g2);
+----------------------+
| mbrdisjoint(@g1,@g2) |
+----------------------+
|                    0 |
+----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MBREqual

Syntax

```
MBREqual(g1,g2)
MBREquals(g1,g2)
```

Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries g1 and g2 are the same.

`MBREquals` is a synonym.

Examples

```
SET @g1=GEOMFROMTEXT('LINESTRING(0 0, 1 2)');
SET @g2=GEOMFROMTEXT('POLYGON((0 0, 0 2, 1 2, 1 0, 0 0))');
SELECT MbrEqual(@g1,@g2);
+------------------+
| MbrEqual(@g1,@g2) |
+------------------+
|                1 |
+------------------+

SET @g1=GEOMFROMTEXT('LINESTRING(0 0, 1 3)');
SET @g2=GEOMFROMTEXT('POLYGON((0 0, 0 2, 1 4, 1 0, 0 0))');
SELECT MbrEqual(@g1,@g2);
+------------------+
| MbrEqual(@g1,@g2) |
+------------------+
|                0 |
+------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MBRIntersects

Syntax

```
MBRIntersects(g1,g2)
```

Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries g1 and g2 intersect.

Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbrintersects(@g1,@g2);
+------------------------+
| mbrintersects(@g1,@g2) |
+------------------------+
|                      1 |
+------------------------+

SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((4 4,4 7,7 7,7 4,4 4))');
SELECT mbrintersects(@g1,@g2);
+------------------------+
| mbrintersects(@g1,@g2) |
+------------------------+
|                      0 |
+------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MBROverlaps

Syntax

```
MBROverlaps(g1,g2)
```

Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries `g1` and `g2` overlap. The term spatially overlaps is used if two geometries intersect and their intersection results in a geometry of the same dimension but not equal to either of the given geometries.

Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((4 4,4 7,7 7,7 4,4 4))');
SELECT mbroverlaps(@g1,@g2);
+----------------------+
| mbroverlaps(@g1,@g2) |
+----------------------+
|                    0 |
+----------------------+

SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbroverlaps(@g1,@g2);
+----------------------+
| mbroverlaps(@g1,@g2) |
+----------------------+
|                    0 |
+----------------------+

SET @g1 = GeomFromText('Polygon((0 0,0 4,4 4,4 0,0 0))');
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbroverlaps(@g1,@g2);
+----------------------+
| mbroverlaps(@g1,@g2) |
+----------------------+
|                    1 |
+----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MBRTouches

Syntax

```
MBRTouches(g1,g2)
```

Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries `g1` and `g2` touch. Two geometries spatially touch if the interiors of the geometries do not intersect, but the boundary of one of the geometries intersects either the boundary or the interior of the other.

Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((4 4,4 7,7 7,7 4,4 4))');
SELECT mbrtouches(@g1,@g2);
+---------------------+
| mbrtouches(@g1,@g2) |
+---------------------+
|                   0 |
+---------------------+

SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbrtouches(@g1,@g2);
+---------------------+
| mbrtouches(@g1,@g2) |
+---------------------+
|                   1 |
+---------------------+

SET @g1 = GeomFromText('Polygon((0 0,0 4,4 4,4 0,0 0))');
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbrtouches(@g1,@g2);
+---------------------+
| mbrtouches(@g1,@g2) |
+---------------------+
|                   0 |
+---------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MBRWithin

Syntax

```
MBRWithin(g1,g2)
```

Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangle of g1 is within the Minimum Bounding Rectangle of g2. This tests the opposite relationship as MBRContains().

Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((0 0,0 5,5 5,5 0,0 0))');
SELECT MBRWithin(@g1,@g2), MBRWithin(@g2,@g1);
+--------------------+--------------------+
| MBRWithin(@g1,@g2) | MBRWithin(@g2,@g1) |
+--------------------+--------------------+
|                  1 |                  0 |
+--------------------+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

Miscellaneous GIS functions

See ST_Collect.

Aggregate multiple geometries into a collection. This function creates a MultiPoint, MultiLineString, MultiPolygon, or GeometryCollection from a set of geometry arguments.

See ST_GeoHash.

Generate a Geohash string from a point or coordinates. This function encodes spatial locations into short, alphanumeric strings for efficient indexing and proximity searches.

See ST_IsValid.

Check if a geometry is valid. This function returns 1 if the geometry complies with OGC specifications (e.g., no self-intersections), 0 otherwise.

See ST_LatFromGeoHash.

Decode a Geohash to retrieve the latitude. This function returns the latitude coordinate (Y-axis) from a given Geohash string.

See ST_LongFromGeoHash.

Decode a Geohash to retrieve the longitude. This function returns the longitude coordinate (X-axis) from a given Geohash string.

See ST_PointFromGeoHash.

Create a Point geometry from a Geohash. This function decodes a Geohash string into a Point object representing the location's center.

See ST_Simplify.

Simplify a geometry using the Douglas-Peucker algorithm. This function reduces the number of vertices in a geometry while preserving its general shape, useful for rendering maps.

See ST_Validate.

Validate and optionally return a geometry. This function checks if a geometry is valid according to OGC rules; it returns the geometry if valid, or NULL if not.

This page is licensed: CC BY-SA / Gnu FDL

ST_Collect

> **Note:**
>
> ST_Collect is available from MariaDB 12.0.

Syntax

```
ST_Collect(g)
```

Description

ST_Collect is an aggregate function that can also be used as a window function.

Given multiple geometries, returns the aggregation of the distinct geometry arguments. This function also supports the DISTINCT option. If DISTINCT is used, it returns the aggregation of the distinct geometry arguments.

The resulting value type is chosen using the following policy:

- If all arguments are Point values, the result is a MultiPoint value.
- If all arguments are LineString values, the result is a MultiLineString value.
- If all arguments are Polygon values, the result is a MultiPolygon value.
- Otherwise, the result is a GeometryCollection value.

If there are multiple geometry arguments and those arguments are in the same spatial reference system (SRS), the return value is in that SRS. If those arguments are not in the same SRS, an

`ER_GIS_DIFFERENT_SRIDS_AGGREGATION` error occurs.

Examples

Multiple Point geometries aggregated into a MultiPoint geometry:

```
CREATE OR REPLACE TABLE t1 ( running_number INTEGER NOT NULL
  AUTO_INCREMENT, grouping_condition INTEGER, location GEOMETRY , PRIMARY KEY (
  running_number));

INSERT INTO t1 ( grouping_condition, location ) VALUES
  ( 0,ST_GEOMFROMTEXT('POINT(0 0)',4326)),
  ( 1,ST_GEOMFROMTEXT('POINT(0 0)',4326)),
  ( 0,ST_GEOMFROMTEXT('POINT(1 0)',4326)),
  ( 1,ST_GEOMFROMTEXT('POINT(2 0)',4326)),
  ( 0,ST_GEOMFROMTEXT('POINT(3 0)',4326));

SELECT ST_EQUALS( (SELECT ST_COLLECT( location ) AS t FROM t1),
  ST_GEOMFROMTEXT('MULTIPOINT(0 0,0 0,1 0,2 0,3 0) ',4326)) AS equals;
+--------+
| equals |
+--------+
|      1 |
+--------+
```

See Also

- Geometry Constructors
- ST_AsText

This page is licensed: CC BY-SA / Gnu FDL

ST_GeoHash

> **Note:**
>
> ST_GeoHash is available from MariaDB 12.0.

Syntax

```
ST_GeoHash(longitude, latitude, max_length)
ST_GeoHash(point, max_length)
```

Description

Returns the geohash corresponding to the input values, or NULL if any argument is NULL. Geohashes encode latitude and longitude coordinates into a text string made up only of numeric and lowercase latin letter characters.

The `longitude` parameter is a numeric value in the interval [180, -180]. `latitude` is a numeric value in the interval [90, -90].

In the case of `point`, the x coordinate is treated as the longitude and the y coordinate is treated as the latitude. The same constraints apply.

The `max_length` parameter is the upper limit on the resulting string size and cannot exceed 100.

The ST_LatFromGeoHash function decodes a given geohash and returns the latitude.

Examples

```
SELECT ST_GeoHash(ST_GeomFromText('POINT(1 1)'),15), ST_GeoHash(0,30,15);
+-----------------------------------------------+---------------------+
| ST_GeoHash(ST_GeomFromText('POINT(1 1)'),15) | ST_GeoHash(0,30,15) |
+-----------------------------------------------+---------------------+
| s00twy01mtw037m                               | sj248j248j248j2     |
+-----------------------------------------------+---------------------+
```

See Also

- ST_LatFromGeoHash
- Geometry Constructors
- ST_GeomFromText

This page is licensed: CC BY-SA / Gnu FDL

ST_IsValid

Note:

ST_IsValid is available from MariaDB 12.0.

Syntax

```
ST_IsValid(g)
```

Description

Given a geometry input, returns 1 if the argument is geometrically valid according to the OGC specifications, 0 if the argument is not geometrically valid.

Unlike ST_Validate, requires valid GIS data, or `ERROR 3037 (22023): Invalid GIS data provided to function st_isvalid` is returned.

Examples

```
SELECT ST_IsValid(ST_GeomFromText('LINESTRING (0 0, 1 1)'));
+-----------------------------------------------------+
| ST_IsValid(ST_GeomFromText('LINESTRING (0 0, 1 1)')) |
+-----------------------------------------------------+
|                                                   1 |
+-----------------------------------------------------+

SELECT ST_IsValid(ST_GeomFromText('LINESTRING (0 0, 0 0)'));
+-----------------------------------------------------+
| ST_IsValid(ST_GeomFromText('LINESTRING (0 0, 0 0)')) |
+-----------------------------------------------------+
|                                                   0 |
+-----------------------------------------------------+
```

A POINT requires both x and y coordinates:

```
SELECT ST_IsValid(ST_GeomFromText('POINT (0)'));
ERROR 3037 (22023): Invalid GIS data provided to function st_isvalid.
```

See Also

- Geometry Constructors
- ST_AsText

This page is licensed: CC BY-SA / Gnu FDL

ST_LatFromGeoHash

> **Note:**
>
> ST_LatFromGeoHash is available from MariaDB 12.0.

Syntax

```
ST_LatFromGeoHash(geohash)
```

Description

Decodes a given `geohash` string and returns the latitude in the interval [90, -90].

If the argument is NULL, the return value is NULL. If the argument is invalid, an ER_INCORRECT_TYPE error is thrown.

The ST_GeoHash function can be used to generate geohashes.

Examples

```
SELECT ST_LatFromGeoHash('zzzzzzzzz'), ST_LatFromGeoHash('xvrfxvrfxvrfxvr');
+-------------------------------+-------------------------------------+
| ST_LatFromGeoHash('zzzzzzzzz') | ST_LatFromGeoHash('xvrfxvrfxvrfxvr') |
+-------------------------------+-------------------------------------+
|                            90 |                                  30 |
+-------------------------------+-------------------------------------+
```

See Also

- ST_GeoHash

This page is licensed: CC BY-SA / Gnu FDL

ST_LongFromGeoHash

> **Note:**
>
> ST_LongFromGeoHash is available from MariaDB 12.0.

Syntax

```
ST_LongFromGeoHash(geohash)
```

Description

Decodes a given `geohash` string and returns the longitude in the interval [180, -180].

If the argument is NULL, the return value is NULL. If the argument is invalid, an ER_INCORRECT_TYPE error is thrown.

The ST_GeoHash function can be used to generate geohashes.

Examples

```
SELECT ST_LongFromGeoHash('zzzzzzzzz'), ST_LongFromGeoHash('sj248j248j248j2');
+---------------------------------+--------------------------------------+
| ST_LongFromGeoHash('zzzzzzzzz') | ST_LongFromGeoHash('sj248j248j248j2') |
+---------------------------------+--------------------------------------+
|                             180 |                                    0 |
+---------------------------------+--------------------------------------+
```

See Also

- ST_GeoHash
- ST_LatFromGeoHash

This page is licensed: CC BY-SA / Gnu FDL

ST_PointFromGeoHash

> **Note:**
>
> ST_PointFromGeoHash is available from MariaDB 12.0.

Syntax

```
ST_PointFromGeoHash(geohash, srid)
```

Description

Takes a given `geohash` string and returns a point where the x is the longitude and the y is the latitude.

The latitude is returned as a numeric value in the interval [180, -180]. The longitude is returned as a numeric value in the interval [90, -90]. If the argument is NULL, the return value is NULL. If the argument is invalid, an ER_GIS_INVALID_DATA is thrown.

Examples

```
SELECT ST_ASTEXT(ST_POINTFROMGEOHASH("s00twy01mtw037m",0));
+----------------------------------------------------+
| ST_ASTEXT(ST_POINTFROMGEOHASH("s00twy01mtw037m",0)) |
+----------------------------------------------------+
| POINT(1 1)                                         |
+----------------------------------------------------+

SELECT ST_ASTEXT(ST_POINTFROMGEOHASH(ST_GEOHASH(180,90,20),0));
+-----------------------------------------------------------+
| ST_ASTEXT(ST_POINTFROMGEOHASH(ST_GEOHASH(180,90,20),0)) |
+-----------------------------------------------------------+
| POINT(180 90)                                             |
+-----------------------------------------------------------+
```

See Also

- ST_GeoHash

This page is licensed: CC BY-SA / Gnu FDL

ST_Simplify

> **Note:**
>
> ST_Simplify is available from MariaDB 12.0.

Syntax

```
ST_Simplify(g, max_distance)
```

Description

Takes as input a geometry (g) and a double (max_distance). It applies the Ramer–Douglas–Peucker algorithm on `g` and returns the resulting geometry.

The goal of the Douglas-Peucker algorithm is to provide generalized simplifications by returning a geometry that is similar to `g` but uses only a subset of points. To perform the simplification, all the vertices that are shorter than `max_distance` are removed.

The algorithm may produce self-intersections and therefore result in invalid geometries. ST_IsValid can be used to test validity of the result.

If the max_distance is not positive or is `NULL`, an `ER_WRONG_ARGUMENT` will occur.

Examples

```
SELECT ST_AsText(ST_Simplify(ST_GeomFromText('LINESTRING(0 0,0 2,2 2,2 4,4 4,4 6,6 6)'), 0.5));
+-----------------------------------------------------------------------------------------+
| ST_AsText(ST_Simplify(ST_GeomFromText('LINESTRING(0 0,0 2,2 2,2 4,4 4,4 6,6 6)'), 0.5)) |
+-----------------------------------------------------------------------------------------+
| LINESTRING(0 0,0 2,2 2,2 4,4 4,4 6,6 6)                                                 |
+-----------------------------------------------------------------------------------------+

SELECT ST_AsText(ST_Simplify(ST_GeomFromText('LINESTRING(0 0,0 2,2 2,2 4,4 4,4 6,6 6)'), 1));
+-----------------------------------------------------------------------------------------+
| ST_AsText(ST_Simplify(ST_GeomFromText('LINESTRING(0 0,0 2,2 2,2 4,4 4,4 6,6 6)'), 1)) |
+-----------------------------------------------------------------------------------------+
| LINESTRING(0 0,0 2,2 2,2 4,6 6)                                                         |
+-----------------------------------------------------------------------------------------+
```

See Also

- Geometry Constructors
- ST_AsText
- ST_GeomFromText

This page is licensed: CC BY-SA / Gnu FDL

ST_Validate

> **Note:**
>
> ST_Validate is available from MariaDB 12.0.

Syntax

```
ST_Validate(g)
```

Description

The function checks that a given geometry is compliant with the Well-Known Binary (WKB) format and Spatial Reference System Identifier (SRID) syntax, and is geometrically valid.

It returns the geometry if it's valid, or NULL if not.

The function is useful to filter out invalid geometry data.

Examples

A POINT requires both x and y coordinates:

```
SELECT ST_ASTEXT(ST_VALIDATE(ST_GeomFromText('POINT(1 0)')));
+-----------------------------------------------------+
| ST_ASTEXT(ST_VALIDATE(ST_GeomFromText('POINT(1 0)'))) |
+-----------------------------------------------------+
| POINT(1 0)                                          |
+-----------------------------------------------------+

SELECT ST_ASTEXT(ST_VALIDATE(ST_GeomFromText('POINT(1)')));
+-----------------------------------------------------+
| ST_ASTEXT(ST_VALIDATE(ST_GeomFromText('POINT(1)'))) |
+-----------------------------------------------------+
| NULL                                                |
+-----------------------------------------------------+
```

See Also

- Geometry Constructors
- ST_AsText

This page is licensed: CC BY-SA / Gnu FDL

Point Properties

See X.

Synonym for ST_X. Returns the X-coordinate value of a Point geometry as a double-precision number.

See Y.

Synonym for ST_Y. Returns the Y-coordinate value of a Point geometry as a double-precision number.

See ST_X.

Returns the X-coordinate of a Point geometry. This function extracts the horizontal coordinate value as a double-precision number.

See ST_Y.

Returns the Y-coordinate of a Point geometry. This function extracts the vertical coordinate value as a double-precision number.

This page is licensed: CC BY-SA / Gnu FDL

X

A synonym for ST_X.

This page is licensed: CC BY-SA / Gnu FDL

Y

A synonym for ST_Y.

This page is licensed: CC BY-SA / Gnu FDL

ST_X

Syntax

```
ST_X(p)
X(p)
```

Description

Returns the X-coordinate value for the point `p` as a double-precision number.

`ST_X()` and `X()` are synonyms.

Examples

```
SET @pt = 'Point(56.7 53.34)';

SELECT X(GeomFromText(@pt));
+----------------------+
| X(GeomFromText(@pt)) |
+----------------------+
|                 56.7 |
+----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_Y

Syntax

```
ST_Y(p)
Y(p)
```

Description

Returns the Y-coordinate value for the point p as a double-precision number.

`ST_Y()` and `Y()` are synonyms.

Examples

```
SET @pt = 'Point(56.7 53.34)';

SELECT Y(GeomFromText(@pt));
+----------------------+
| Y(GeomFromText(@pt)) |
+----------------------+
|                53.34 |
+----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

Polygon Properties

See CENTROID.

Synonym for ST_CENTROID. Returns the mathematical centroid of the Polygon or MultiPolygon as a Point geometry.

See AREA.

Synonym for ST_AREA. Returns the double-precision area of the Polygon or MultiPolygon, calculated in its spatial reference system.

See ExteriorRing.

Synonym for ST_ExteriorRing. Returns the exterior ring of a Polygon as a LineString.

See InteriorRingN.

Synonym for ST_InteriorRingN. Returns the N-th interior ring of a Polygon as a LineString.

See NumInteriorRings.

Synonym for ST_NumInteriorRings. Returns the number of interior rings in a Polygon geometry.

See ST_AREA.

Returns the area of a Polygon or MultiPolygon. The result is a double-precision number measured in the geometry's spatial reference units.

See ST_CENTROID.

Returns the centroid of a Polygon or MultiPolygon. The result is a Point geometry representing the mathematical center of mass.

See ST_ExteriorRing.

Returns the exterior ring of a Polygon. This function extracts the outer boundary of the polygon as a LineString geometry.

See ST_InteriorRingN.

Returns the N-th interior ring of a Polygon. This function retrieves a specific inner hole of the polygon as a LineString.

See ST_NumInteriorRings.

Returns the count of interior rings in a Polygon. This function calculates the total number of inner holes within the polygon.

This page is licensed: CC BY-SA / Gnu FDL

CENTROID

A synonym for ST_CENTROID.

This page is licensed: CC BY-SA / Gnu FDL

AREA

A synonym for ST_AREA.

This page is licensed: CC BY-SA / Gnu FDL

ExteriorRing

A synonym for ST_ExteriorRing.

This page is licensed: CC BY-SA / Gnu FDL

InteriorRingN

A synonym for ST_InteriorRingN.

This page is licensed: CC BY-SA / Gnu FDL

NumInteriorRings

A synonym for ST_NumInteriorRings.

This page is licensed: CC BY-SA / Gnu FDL

ST_AREA

Syntax

```
ST_Area(poly)
Area(poly)
```

Description

Returns as a double-precision number the area of the Polygon value `poly`, as measured in its spatial reference system.

`ST_Area()` and `Area()` are synonyms.

Examples

```
SET @poly = 'Polygon((0 0,0 3,3 0,0 0),(1 1,1 2,2 1,1 1))';

SELECT Area(GeomFromText(@poly));
+---------------------------+
| Area(GeomFromText(@poly)) |
+---------------------------+
|                         4 |
+---------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_CENTROID

Syntax

```
ST_Centroid(mpoly)
Centroid(mpoly)
```

Description

Returns a point reflecting the mathematical centroid (geometric center) for the MultiPolygon *mpoly*. The resulting point will not necessarily be on the MultiPolygon.

`ST_Centroid()` and `Centroid()` are synonyms.

Examples

```
SET @poly = ST_GeomFromText('POLYGON((0 0,20 0,20 20,0 20,0 0))');
SELECT ST_AsText(ST_Centroid(@poly)) AS center;
+--------------+
| center       |
+--------------+
| POINT(10 10) |
+--------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ST_ExteriorRing

Syntax

```
ST_ExteriorRing(poly)
ExteriorRing(poly)
```

Description

Returns the exterior ring of the Polygon value `poly` as a LineString.

`ST_ExteriorRing()` and `ExteriorRing()` are synonyms.

Examples

```
SET @poly = 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';

SELECT AsText(ExteriorRing(GeomFromText(@poly)));
+------------------------------------------+
| AsText(ExteriorRing(GeomFromText(@poly))) |
+------------------------------------------+
| LINESTRING(0 0,0 3,3 3,3 0,0 0)          |
+------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_InteriorRingN

Syntax

```
ST_InteriorRingN(poly,N)
InteriorRingN(poly,N)
```

Description

Returns the N-th interior ring for the Polygon value `poly` as a LineString. Rings are numbered beginning with 1.

`ST_InteriorRingN()` and `InteriorRingN()` are synonyms.

Examples

```
SET @poly = 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';

SELECT AsText(InteriorRingN(GeomFromText(@poly),1));
+---------------------------------------------+
| AsText(InteriorRingN(GeomFromText(@poly),1)) |
+---------------------------------------------+
| LINESTRING(1 1,1 2,2 2,2 1,1 1)             |
+---------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_NumInteriorRings

Syntax

```
ST_NumInteriorRings(poly)
NumInteriorRings(poly)
```

Description

Returns an integer containing the number of interior rings in the Polygon value `poly`.

Note that according the OpenGIS standard, a POLYGON should have exactly one ExteriorRing and all other rings should lie within that ExteriorRing and thus be the InteriorRings. Practically, however, some systems, including MariaDB's, permit polygons to have several 'ExteriorRings'. In the case of there being multiple, non-overlapping exterior rings ST_NumInteriorRings() will return `1`.

`ST_NumInteriorRings()` and `NumInteriorRings()` are synonyms.

Examples

```
SET @poly = 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';

SELECT NumInteriorRings(GeomFromText(@poly));
+--------------------------------------+
| NumInteriorRings(GeomFromText(@poly)) |
+--------------------------------------+
|                                    1 |
+--------------------------------------+
```

Non-overlapping 'polygon':

```
SELECT ST_NumInteriorRings(ST_PolyFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),
  (-1 -1,-5 -1,-5 -5,-1 -5,-1 -1))')) AS NumInteriorRings;
+------------------+
| NumInteriorRings |
+------------------+
|                1 |
+------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

WKB

- AsBinary
- AsWKB
- GeomCollFromWKB
- GeometryCollectionFromWKB
- GeometryFromWKB
- GeomFromWKB
- LineFromWKB
- LineStringFromWKB
- MLineFromWKB
- MPointFromWKB
- MPolyFromWKB

- MultiLineStringFromWKB
- MultiPointFromWKB
- MultiPolygonFromWKB
- PointFromWKB
- PolyFromWKB
- PolygonFromWKB
- ST_AsBinary
- ST_AsWKB
- ST_GeomCollFromWKB
- ST_GeometryCollectionFromWKB
- ST_GeometryFromWKB
- ST_GeomFromWKB
- ST_LineFromWKB
- ST_LineStringFromWKB
- ST_MPointFromWKB
- ST_MPolyFromWKB
- ST_MultiPointFromWKB
- ST_MultiPolygonFromWKB
- ST_PointFromWKB
- ST_PolyFromWKB
- ST_PolygonFromWKB
- Well-Known Binary (WKB) Format

This page is licensed: CC BY-SA / Gnu FDL

AsBinary

A synonym for ST_AsBinary().

This page is licensed: CC BY-SA / Gnu FDL

AsWKB

A synonym for ST_AsBinary().

This page is licensed: CC BY-SA / Gnu FDL

GeomCollFromWKB

A synonym for ST_GeomCollFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

GeometryCollectionFromWKB

A synonym for ST_GeomCollFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

GeometryFromWKB

A synonym for ST_GeomFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

GeomFromWKB

A synonym for ST_GeomFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

LineFromWKB

A synonym for ST_LineFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

LineStringFromWKB

A synonym for ST_LineFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

MLineFromWKB

Syntax

```
MLineFromWKB(wkb[,srid])
MultiLineStringFromWKB(wkb[,srid])
```

Description

Constructs a MULTILINESTRING value using its WKB representation and SRID.

`MLineFromWKB()` and `MultiLineStringFromWKB()` are synonyms.

Examples

```
SET @g = ST_AsBinary(MLineFromText('MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48))'));

SELECT ST_AsText(MLineFromWKB(@g));
+-------------------------------------------------------+
| ST_AsText(MLineFromWKB(@g))                           |
+-------------------------------------------------------+
| MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48)) |
+-------------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MPointFromWKB

Syntax

```
MPointFromWKB(wkb[,srid])
MultiPointFromWKB(wkb[,srid])
```

Description

Constructs a MULTIPOINT value using its WKB representation and SRID.

`MPointFromWKB()` and `MultiPointFromWKB()` are synonyms.

Examples

```
SET @g = ST_AsBinary(MPointFromText('MultiPoint( 1 1, 2 2, 5 3, 7 2, 9 3, 8 4, 6 6, 6 9, 4 9, 1 5 )'));

SELECT ST_AsText(MPointFromWKB(@g));
+-----------------------------------------------------+
| ST_AsText(MPointFromWKB(@g))                        |
+-----------------------------------------------------+
| MULTIPOINT(1 1,2 2,5 3,7 2,9 3,8 4,6 6,6 9,4 9,1 5) |
+-----------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MPolyFromWKB

Syntax

```
MPolyFromWKB(wkb[,srid])
MultiPolygonFromWKB(wkb[,srid])
```

Description

Constructs a MULTIPOLYGON value using its WKB representation and SRID.

`MPolyFromWKB()` and `MultiPolygonFromWKB()` are synonyms.

Examples

```
SET @g = ST_AsBinary(MPointFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48
6,52 18)),((59 18,67 18,67 13,59 13,59 18)))'));

SELECT ST_AsText(MPolyFromWKB(@g))\G
*************************** 1. row ***************************
ST_AsText(MPolyFromWKB(@g)): MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),
((59 18,67 18,67 13,59 13,59 18)))
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MultiLineStringFromWKB

A synonym for MLineFromWKB().

This page is licensed: CC BY-SA / Gnu FDL

MultiPointFromWKB

A synonym for MPointFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

MultiPolygonFromWKB

Synonym for MPolyFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

PointFromWKB

A synonym for ST_PointFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

PolyFromWKB

A synonym for ST_PolyFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

PolygonFromWKB

A synonym for ST_PolyFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

ST_AsBinary

Syntax

```
ST_AsBinary(g)
AsBinary(g)
ST_AsWKB(g)
AsWKB(g)
```

Description

Converts a value in internal geometry format to its WKB representation and returns the binary result.

`ST_AsBinary()`, `AsBinary()`, `ST_AsWKB()` and `AsWKB()` are synonyms,

Examples

```
SET @poly = ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))');
SELECT ST_AsBinary(@poly);

SELECT ST_AsText(ST_GeomFromWKB(ST_AsWKB(@poly)));
+-------------------------------------------+
| ST_AsText(ST_GeomFromWKB(ST_AsWKB(@poly))) |
+-------------------------------------------+
| POLYGON((0 0,0 1,1 1,1 0,0 0))             |
+-------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_AsWKB

A synonym for ST_AsBinary().

This page is licensed: CC BY-SA / Gnu FDL

ST_GeomCollFromWKB

Syntax

```
ST_GeomCollFromWKB(wkb[,srid])
ST_GeometryCollectionFromWKB(wkb[,srid])
GeomCollFromWKB(wkb[,srid])
GeometryCollectionFromWKB(wkb[,srid])
```

Description

Constructs a GEOMETRYCOLLECTION value using its WKB representation and SRID.

`ST_GeomCollFromWKB()`, `ST_GeometryCollectionFromWKB()`, `GeomCollFromWKB()` and `GeometryCollectionFromWKB()` are synonyms.

Examples

```
SET @g = ST_AsBinary(ST_GeomFromText('GEOMETRYCOLLECTION(
   POLYGON((5 5,10 5,10 10,5 5)),POINT(10 10))'));

SELECT ST_AsText(ST_GeomCollFromWKB(@g));
+-------------------------------------------------------------+
| ST_AsText(ST_GeomCollFromWKB(@g))                           |
+-------------------------------------------------------------+
| GEOMETRYCOLLECTION(POLYGON((5 5,10 5,10 10,5 5)),POINT(10 10)) |
+-------------------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_GeometryCollectionFromWKB

A synonym for ST_GeomCollFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

ST_GeometryFromWKB

A synonym for ST_GeomFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

ST_GeomFromWKB

Syntax

```
ST_GeomFromWKB(wkb[,srid])
ST_GeometryFromWKB(wkb[,srid])
GeomFromWKB(wkb[,srid])
GeometryFromWKB(wkb[,srid])
```

Description

Constructs a geometry value of any type using its WKB representation and SRID.

`ST_GeomFromWKB()`, `ST_GeometryFromWKB()`, `GeomFromWKB()` and `GeometryFromWKB()` are synonyms.

Examples

```
SET @g = ST_AsBinary(ST_LineFromText('LINESTRING(0 4, 4 6)'));

SELECT ST_AsText(ST_GeomFromWKB(@g));
+------------------------------+
| ST_AsText(ST_GeomFromWKB(@g)) |
+------------------------------+
| LINESTRING(0 4,4 6)          |
+------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_LineFromWKB

Syntax

```
ST_LineFromWKB(wkb[,srid])
LineFromWKB(wkb[,srid])
ST_LineStringFromWKB(wkb[,srid])
LineStringFromWKB(wkb[,srid])
```

Description

Constructs a LINESTRING value using its WKB representation and SRID.

`ST_LineFromWKB()`, `LineFromWKB()`, `ST_LineStringFromWKB()`, and `LineStringFromWKB()` are synonyms.

Examples

```
SET @g = ST_AsBinary(ST_LineFromText('LineString(0 4,4 6)'));

SELECT ST_AsText(ST_LineFromWKB(@g)) AS l;
+--------------------+
| l                  |
+--------------------+
| LINESTRING(0 4,4 6) |
+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_LineStringFromWKB

A synonym for ST_LineFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

ST_MPointFromWKB

A synonym for ST_GeomFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

ST_MPolyFromWKB

A synonym for ST_GeomFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

ST_MultiPointFromWKB

A synonym for ST_GeomFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

ST_MultiPolygonFromWKB

A synonym for ST_GeomFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

ST_PointFromWKB

Syntax

```
ST_PointFromWKB(wkb[,srid])
PointFromWKB(wkb[,srid])
```

Description

Constructs a POINT value using its WKB representation and SRID.

`ST_PointFromWKB()` and `PointFromWKB()` are synonyms.

Examples

```
SET @g = ST_AsBinary(ST_PointFromText('POINT(0 4)'));

SELECT ST_AsText(ST_PointFromWKB(@g)) AS p;
+------------+
| p          |
+------------+
| POINT(0 4) |
+------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_PolyFromWKB

Syntax

```
ST_PolyFromWKB(wkb[,srid])
ST_PolygonFromWKB(wkb[,srid])
PolyFromWKB(wkb[,srid])
PolygonFromWKB(wkb[,srid])
```

Description

Constructs a POLYGON value using its WKB representation and SRID.

`ST_PolyFromWKB()`, `ST_PolygonFromWKB()`, `PolyFromWKB()` and `PolygonFromWKB()` are synonyms.

Examples

```
SET @g = ST_AsBinary(ST_PolyFromText('POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1))'));

SELECT ST_AsText(ST_PolyFromWKB(@g)) AS p;
+---------------------------------------+
| p                                     |
+---------------------------------------+
| POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1)) |
+---------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_PolygonFromWKB

A synonym for ST_PolyFromWKB.

This page is licensed: CC BY-SA / Gnu FDL

Well-Known Binary (WKB) Format

WKB stands for Well-Known Binary, a format for representing geographical and geometrical data.

WKB uses 1-byte unsigned integers, 4-byte unsigned integers, and 8-byte double-precision numbers.

- The first byte indicates the byte order. 00 for big endian, or 01 for little endian.
- The next 4 bytes indicate the geometry type. Values from 1 to 7 indicate whether the type is Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, or GeometryCollection respectively.
- The 8-byte floats represent the coordinates.

Take the following example, a sequence of 21 bytes each represented by two hex digits:

```
000000000140000000000000004010000000000000
```

- It's big endian
  - 000000000140000000000000004010000000000000
- It's a POINT
  - 000000000140000000000000004010000000000000
- The X coordinate is 2.0
  - 000000000140000000000000004010000000000000
- The Y-coordinate is 4.0
  - 000000000140000000000000004010000000000000

This page is licensed: CC BY-SA / Gnu FDL

WKT

- WKT Definition
- AsText
- AsWKT
- GeomCollFromText
- GeometryCollectionFromText
- GeometryFromText
- GeomFromText
- LineFromText
- LineStringFromText
- MLineFromText
- MPointFromText
- MPolyFromText
- MultiLineStringFromText
- MultiPointFromText
- MultiPolygonFromText
- PointFromText
- PolyFromText
- PolygonFromText
- ST_AsText
- ST_ASWKT
- ST_GeomCollFromText
- ST_GeometryCollectionFromText

- ST_GeometryFromText
- ST_GeomFromText
- ST_LineFromText
- ST_LineStringFromText
- ST_MPointFromText
- ST_MPolyFromText
- ST_MultiLineStringFromText
- ST_MultiPointFromText
- ST_MultiPolygonFromText
- ST_PointFromText
- ST_PolyFromText
- ST_PolygonFromText
- ST_MLineFromText

This page is licensed: CC BY-SA / Gnu FDL

WKT Definition

Description

The Well-Known Text (WKT) representation of Geometry is designed to exchange geometry data in ASCII form. Examples of the basic geometry types include:

Geometry Types
POINT
LINESTRING
POLYGON
MULTIPOINT
MULTILINESTRING
MULTIPOLYGON
GEOMETRYCOLLECTION
GEOMETRY

See Also

- Geometry Types

This page is licensed: GPLv2, originally from fill_help_tables.sql

AsText

A synonym for ST_AsText().

This page is licensed: CC BY-SA / Gnu FDL

AsWKT

A synonym for ST_AsText().

This page is licensed: CC BY-SA / Gnu FDL

GeomCollFromText

A synonym for ST_GeomCollFromText.

This page is licensed: CC BY-SA / Gnu FDL

GeometryCollectionFromText

A synonym for ST_GeomCollFromText.

This page is licensed: CC BY-SA / Gnu FDL

GeometryFromText

A synonym for ST_GeomFromText.

This page is licensed: CC BY-SA / Gnu FDL

GeomFromText

A synonym for ST_GeomFromText.

This page is licensed: CC BY-SA / Gnu FDL

LineFromText

A synonym for ST_LineFromText.

This page is licensed: CC BY-SA / Gnu FDL

LineStringFromText

A synonym for ST_LineFromText.

This page is licensed: CC BY-SA / Gnu FDL

MLineFromText

Syntax

```
MLineFromText(wkt[,srid])
MultiLineStringFromText(wkt[,srid])
```

Description

Constructs a MULTILINESTRING value using its WKT representation and SRID.

`MLineFromText()` and `MultiLineStringFromText()` are synonyms.

Examples

```
CREATE TABLE gis_multi_line (g MULTILINESTRING);
SHOW FIELDS FROM gis_multi_line;
INSERT INTO gis_multi_line VALUES
    (MultiLineStringFromText('MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48))')),
    (MLineFromText('MULTILINESTRING((10 48,10 21,10 0))')),
    (MLineFromWKB(AsWKB(MultiLineString(
      LineString(Point(1, 2), Point(3, 5)),
      LineString(Point(2, 5), Point(5, 8), Point(21, 7))))));
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MPointFromText

Syntax

```
MPointFromText(wkt[,srid])
MultiPointFromText(wkt[,srid])
```

Description

Constructs a MULTIPOINT value using its WKT representation and SRID.

`MPointFromText()` and `MultiPointFromText()` are synonyms.

Examples

```
CREATE TABLE gis_multi_point (g MULTIPOINT);
SHOW FIELDS FROM gis_multi_point;
INSERT INTO gis_multi_point VALUES
    (MultiPointFromText('MULTIPOINT(0 0,10 10,10 20,20 20)')),
    (MPointFromText('MULTIPOINT(1 1,11 11,11 21,21 21)')),
    (MPointFromWKB(AsWKB(MultiPoint(Point(3, 6), Point(4, 10)))));
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MPolyFromText

Syntax

```
MPolyFromText(wkt[,srid])
MultiPolygonFromText(wkt[,srid])
```

Description

Constructs a MULTIPOLYGON value using its WKT representation and SRID.

`MPolyFromText()` and `MultiPolygonFromText()` are synonyms.

Examples

```
CREATE TABLE gis_multi_polygon  (g MULTIPOLYGON);
SHOW FIELDS FROM gis_multi_polygon;
INSERT INTO gis_multi_polygon VALUES
    (MultiPolygonFromText('MULTIPOLYGON(
       ((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),
       ((59 18,67 18,67 13,59 13,59 18)))')),
    (MPolyFromText('MULTIPOLYGON(
       ((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),
       ((59 18,67 18,67 13,59 13,59 18)))')),
    (MPolyFromWKB(AsWKB(MultiPolygon(Polygon(
       LineString(Point(0, 3), Point(3, 3), Point(3, 0), Point(0, 3)))))));
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MultiLineStringFromText

A synonym for MLineFromText.

This page is licensed: CC BY-SA / Gnu FDL

MultiPointFromText

A synonym for MPointFromText.

This page is licensed: CC BY-SA / Gnu FDL

MultiPolygonFromText

A synonym for MPolyFromText.

This page is licensed: CC BY-SA / Gnu FDL

PointFromText

A synonym for ST_PointFromText.

This page is licensed: CC BY-SA / Gnu FDL

PolyFromText

A synonym for ST_PolyFromText.

This page is licensed: CC BY-SA / Gnu FDL

PolygonFromText

A synonym for ST_PolyFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_AsText

Syntax

```
ST_AsText(g)
AsText(g)
ST_AsWKT(g)
AsWKT(g)
```

Description

Converts a value in internal geometry format to its WKT representation and returns the string result.

`ST_AsText()`, `AsText()`, `ST_AsWKT()` and `AsWKT()` are all synonyms.

Examples

```
SET @g = 'LineString(1 1,4 4,6 6)';

SELECT ST_AsText(ST_GeomFromText(@g));
+--------------------------------+
| ST_AsText(ST_GeomFromText(@g)) |
+--------------------------------+
| LINESTRING(1 1,4 4,6 6)        |
+--------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_ASWKT

A synonym for ST_ASTEXT().

This page is licensed: CC BY-SA / Gnu FDL

ST_GeomCollFromText

Syntax

```
ST_GeomCollFromText(wkt[,srid])
ST_GeometryCollectionFromText(wkt[,srid])
GeomCollFromText(wkt[,srid])
GeometryCollectionFromText(wkt[,srid])
```

Description

Constructs a GEOMETRYCOLLECTION value using its WKT representation and SRID.

`ST_GeomCollFromText()`, `ST_GeometryCollectionFromText()`, `GeomCollFromText()` and `GeometryCollectionFromText()` are all synonyms.

Example

```
CREATE TABLE gis_geometrycollection  (g GEOMETRYCOLLECTION);
SHOW FIELDS FROM gis_geometrycollection;
INSERT INTO gis_geometrycollection VALUES
    (GeomCollFromText('GEOMETRYCOLLECTION(POINT(0 0), LINESTRING(0 0,10 10))')),
    (GeometryFromWKB(AsWKB(GeometryCollection(Point(44, 6), LineString(Point(3, 6), Point(7, 9))))),
    (GeomFromText('GeometryCollection()')),
    (GeomFromText('GeometryCollection EMPTY'));
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_GeometryCollectionFromText

A synonym for ST_GeomCollFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_GeometryFromText

A synonym for ST_GeomFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_GeomFromText

Syntax

```
ST_GeomFromText(wkt[,srid])
ST_GeometryFromText(wkt[,srid])
GeomFromText(wkt[,srid])
GeometryFromText(wkt[,srid])

ST_MultiLineStringFromText(wkt[,srid])
ST_MLineFromText(wkt[,srid])
ST_MPointFromText(wkt[,srid])
ST_MultiPointFromText(wkt[,srid])
```

Description

Constructs a geometry value of any type using its WKT representation and SRID.

`GeomFromText()`, `GeometryFromText()`, `ST_GeomFromText()`, `ST_GeometryFromText()`, `ST_MultiLineStringFromText`, `ST_MLineFromText`, `ST_MultiPolyFromText` and `ST_MPolygonFromText` are all synonyms.

Example

```
SET @g = ST_GEOMFROMTEXT('POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1))');
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_LineFromText

Syntax

```
ST_LineFromText(wkt[,srid])
ST_LineStringFromText(wkt[,srid])
LineFromText(wkt[,srid])
LineStringFromText(wkt[,srid])
```

Description

Constructs a LINESTRING value using its WKT representation and SRID.

`ST_LineFromText()`, `ST_LineStringFromText()`, `ST_LineFromText()` and `ST_LineStringFromText()` are all synonyms.

Examples

```
CREATE TABLE gis_line  (g LINESTRING);
SHOW FIELDS FROM gis_line;
INSERT INTO gis_line VALUES
    (LineFromText('LINESTRING(0 0,0 10,10 0)')),
    (LineStringFromText('LINESTRING(10 10,20 10,20 20,10 20,10 10)')),
    (LineStringFromWKB(AsWKB(LineString(Point(10, 10), Point(40, 10)))));
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_LineStringFromText

A synonym for ST_LineFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_MPointFromText

A synonym for ST_GeomFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_MPolyFromText

A synonym for ST_GeomFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_MultiLineStringFromText

A synonym for ST_GeomFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_MultiPointFromText

A synonym for ST_GeomFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_MultiPolygonFromText

A synonym for ST_GeomFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_PointFromText

Syntax

```
ST_PointFromText(wkt[,srid])
PointFromText(wkt[,srid])
```

Description

Constructs a POINT value using its WKT representation and SRID.

`ST_PointFromText()` and `PointFromText()` are synonyms.

Examples

```
CREATE TABLE gis_point  (g POINT);
SHOW FIELDS FROM gis_point;
INSERT INTO gis_point VALUES
    (PointFromText('POINT(10 10)')),
    (PointFromText('POINT(20 10)')),
    (PointFromText('POINT(20 20)')),
    (PointFromWKB(AsWKB(PointFromText('POINT(10 20)'))));
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_PolyFromText

Syntax

```
ST_PolyFromText(wkt[,srid])
ST_PolygonFromText(wkt[,srid])
PolyFromText(wkt[,srid])
PolygonFromText(wkt[,srid])
```

Description

Constructs a POLYGON value using its WKT representation and SRID.

`ST_PolyFromText()`, `ST_PolygonFromText()`, `PolyFromText()` and `ST_PolygonFromText()` are all synonyms.

Examples

```
CREATE TABLE gis_polygon   (g POLYGON);
INSERT INTO gis_polygon VALUES
    (PolygonFromText('POLYGON((10 10,20 10,20 20,10 20,10 10))')),
    (PolyFromText('POLYGON((0 0,50 0,50 50,0 50,0 0), (10 10,20 10,20 20,10 20,10 10))'));
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ST_PolygonFromText

A synonym for ST_PolyFromText.

This page is licensed: CC BY-SA / Gnu FDL

ST_MLineFromText

A synonym for ST_GeomFromText.

This page is licensed: CC BY-SA / Gnu FDL

Prepared Statements

See DEALLOCATE / DROP PREPARE.

Release a prepared statement to free resources. This command removes the statement definition and its name from the current session.

See EXECUTE Statement.

Run a previously prepared statement. This command executes the statement using the specified name, optionally supplying input parameters.

See EXECUTE IMMEDIATE.

Prepare and run a dynamic SQL statement in one step. This command simplifies the process by combining the PREPARE and EXECUTE operations.

See PREPARE Statement.

Parse and optimize a SQL statement for later use. This command assigns a name to the statement, enabling efficient execution with parameters.

This page is licensed: CC BY-SA / Gnu FDL

DEALLOCATE / DROP PREPARE

Syntax

```
{DEALLOCATE | DROP} PREPARE stmt_name
```

Description

To deallocate a prepared statement produced with PREPARE, use a `DEALLOCATE PREPARE` statement that refers to the prepared statement name.

A prepared statement is implicitly deallocated when a new `PREPARE` command is issued. In that case, there is no need to use `DEALLOCATE`.

Attempting to execute a prepared statement after deallocating it results in an error, as if it was not prepared at all:

```
ERROR 1243 (HY000): Unknown prepared statement handler (stmt_name) given to EXECUTE
```

If the specified statement has not been PREPAREd, an error similar to the following will be produced:

```
ERROR 1243 (HY000): Unknown prepared statement handler (stmt_name) given to DEALLOCATE PREPARE
```

Example

See example in PREPARE.

See Also

- PREPARE Statement
- EXECUTE Statement
- EXECUTE IMMEDIATE

This page is licensed: GPLv2, originally from fill_help_tables.sql

EXECUTE Statement

Syntax

```
EXECUTE stmt_name
    [USING expression[, expression] ...]
```

Description

After preparing a statement with PREPARE, you execute it with an `EXECUTE` statement that refers to the prepared statement name. If the prepared statement contains any parameter markers, you must supply a `USING` clause that lists expressions containing the values to be bound to the parameters. The `USING` clause must name exactly as many expressions as the number of parameter markers in the statement.

You can execute a given prepared statement multiple times, passing different variables to it or setting the variables to different values before each execution.

If the specified statement has not been PREPAREd, an error similar to the following is produced:

```
ERROR 1243 (HY000): Unknown prepared statement handler (stmt_name) given to EXECUTE
```

> **CURRENT**
>
> `EXECUTE` with arbitrary expression as parameters can be used, not just user variables (@var_name).
>
> **< 10.2.3**
>
> You can only use user variables (@var_name) as parameters.

Example

See example in PREPARE.

See Also

- EXECUTE IMMEDIATE

This page is licensed: GPLv2, originally from fill_help_tables.sql

EXECUTE IMMEDIATE

Syntax

```
EXECUTE IMMEDIATE statement
    [USING param[, param] ...]

param:
    expression | IGNORE | DEFAULT
```

Description

`EXECUTE IMMEDIATE` executes a dynamic SQL statement created on the fly, which can reduce performance overhead. For example:

```
EXECUTE IMMEDIATE 'SELECT 1'
```

which is shorthand for:

```
PREPARE stmt FROM "select 1";
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

EXECUTE IMMEDIATE supports complex expressions as prepare source and parameters:

```
EXECUTE IMMEDIATE CONCAT('SELECT COUNT(*) FROM ', 't1', ' WHERE a=?') USING 5+5;
```

Limitations: subselects and stored function calls are not supported as a prepare source.

The following examples return an error:

```
CREATE OR REPLACE FUNCTION f1() RETURNS VARCHAR(64) RETURN 'SELECT * FROM t1';
EXECUTE IMMEDIATE f1();
ERROR 1970 (42000): EXECUTE IMMEDIATE does not support subqueries or stored functions

EXECUTE IMMEDIATE (SELECT 'SELECT * FROM t1');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
  corresponds to your MariaDB server version for the right syntax to use near
  'SELECT 'SELECT * FROM t1')' at line 1

CREATE OR REPLACE FUNCTION f1() RETURNS INT RETURN 10;
EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING f1();
ERROR 1970 (42000): EXECUTE..USING does not support subqueries or stored functions

EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING (SELECT 10);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
  corresponds to your MariaDB server version for the right syntax to use near
  'SELECT 10)' at line 1
```

One can use a user or an SP variable as a workaround:

```
CREATE OR REPLACE FUNCTION f1() RETURNS VARCHAR(64) RETURN 'SELECT * FROM t1';
SET @stmt=f1();
EXECUTE IMMEDIATE @stmt;

SET @stmt=(SELECT 'SELECT 1');
EXECUTE IMMEDIATE @stmt;

CREATE OR REPLACE FUNCTION f1() RETURNS INT RETURN 10;
SET @param=f1();
EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING @param;

SET @param=(SELECT 10);
EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING @param;
```

`EXECUTE IMMEDIATE` supports user variables and SP variables as OUT parameters:

```
DELIMITER $$
CREATE OR REPLACE PROCEDURE p1(OUT a INT)
BEGIN
  SET a:= 10;
END;
$$
DELIMITER ;
SET @a=2;
EXECUTE IMMEDIATE 'CALL p1(?)' USING @a;
SELECT @a;
+------+
| @a   |
+------+
|   10 |
+------+
```

Similar to `PREPARE`, `EXECUTE IMMEDIATE` is allowed in stored procedures but is not allowed in stored functions.

This example uses `EXECUTE IMMEDIATE` inside a stored procedure:

```
DELIMITER $$
CREATE OR REPLACE PROCEDURE p1()
BEGIN
  EXECUTE IMMEDIATE 'SELECT 1';
END;
$$
DELIMITER ;
CALL p1;
+---+
| 1 |
+---+
| 1 |
+---+
```

This script returns an error:

```
DELIMITER $$
CREATE FUNCTION f1() RETURNS INT
BEGIN
  EXECUTE IMMEDIATE 'DO 1';
  RETURN 1;
END;
$$
ERROR 1336 (0A000): Dynamic SQL is not allowed in stored function or trigger
```

`EXECUTE IMMEDIATE` can use `DEFAULT` and `IGNORE` indicators as bind parameters:

```
CREATE OR REPLACE TABLE t1 (a INT DEFAULT 10);
EXECUTE IMMEDIATE 'INSERT INTO t1 VALUES (?)' USING DEFAULT;
SELECT * FROM t1;
+------+
| a    |
+------+
|   10 |
+------+
```

EXECUTE IMMEDIATE increments the Com_execute_immediate status variable, as well as the Com_stmt_prepare, Com_stmt_execute and Com_stmt_close status variables.

Note, EXECUTE IMMEDIATE does not increment the Com_execute_sql status variable. *Com_execute_sql* is used only for PREPARE..EXECUTE.

This session screenshot demonstrates how EXECUTE IMMEDIATE affects status variables:

```
SELECT * FROM INFORMATION_SCHEMA.SESSION_STATUS WHERE VARIABLE_NAME RLIKE
  ('COM_(EXECUTE|STMT_PREPARE|STMT_EXECUTE|STMT_CLOSE)');

+-----------------------+----------------+
| VARIABLE_NAME         | VARIABLE_VALUE |
+-----------------------+----------------+
| COM_EXECUTE_IMMEDIATE | 0              |
| COM_EXECUTE_SQL       | 0              |
| COM_STMT_CLOSE        | 0              |
| COM_STMT_EXECUTE      | 0              |
| COM_STMT_PREPARE      | 0              |
+-----------------------+----------------+

EXECUTE IMMEDIATE 'SELECT 1';
+---+
| 1 |
+---+
| 1 |
+---+

SELECT * FROM INFORMATION_SCHEMA.SESSION_STATUS WHERE VARIABLE_NAME RLIKE
  ('COM_(EXECUTE|STMT_PREPARE|STMT_EXECUTE|STMT_CLOSE)');
+-----------------------+----------------+
| VARIABLE_NAME         | VARIABLE_VALUE |
+-----------------------+----------------+
| COM_EXECUTE_IMMEDIATE | 1              |
| COM_EXECUTE_SQL       | 0              |
| COM_STMT_CLOSE        | 1              |
| COM_STMT_EXECUTE      | 1              |
| COM_STMT_PREPARE      | 1              |
+-----------------------+----------------+
```

This page is licensed: CC BY-SA / Gnu FDL

PREPARE Statement

Syntax

```
PREPARE stmt_name FROM preparable_stmt
```

Description

The `PREPARE` statement prepares a statement and assigns it a name, `stmt_name`, by which to refer to the statement later. Statement names are not case sensitive. `preparable_stmt` is an expression with the text of the statement. It cannot contain stored function calls or subqueries. The text must represent a single SQL statement, not multiple statements. Within the statement, `?` characters can be used as parameter markers to indicate where data values are to be bound to the query later when you execute it. The `?` characters should not be enclosed within quotes, even if you intend to bind them to string values. Parameter markers can be used only where expressions should appear, not for SQL keywords, identifiers, and so forth.

The scope of a prepared statement is the session within which it is created. Other sessions cannot see it.

If a prepared statement with the given name already exists, it is deallocated implicitly before the new statement is prepared. This means that if the new statement contains an error and cannot be prepared, an error is returned and no statement with the given name exists.

Prepared statements can be `PREPARE` and EXECUTE in a stored procedure, but not in a stored function or trigger. Also, even if the statement is prepared with `PREPARE` in a procedure, it will not be deallocated when the procedure execution ends.

A prepared statement can access user-defined variables, but not local variables or procedure's parameters.

If the prepared statement contains a syntax error, PREPARE will fail. As a side effect, stored procedures can use it to check if a statement is valid. For example:

```
CREATE PROCEDURE `test_stmt`(IN sql_text TEXT)
BEGIN
        DECLARE EXIT HANDLER FOR SQLEXCEPTION
        BEGIN
                SELECT CONCAT(sql_text, ' is not valid');
        END;
        SET @SQL := sql_text;
        PREPARE stmt FROM @SQL;
        DEALLOCATE PREPARE stmt;
END;
```

The FOUND_ROWS() and ROW_COUNT() functions, if called immediately after `EXECUTE`, return the number of rows read or affected by the prepared statements; however, if they are called after `DEALLOCATE PREPARE`, they provide information about this statement. If the prepared statement produces errors or warnings, GET DIAGNOSTICS return information about them. `DEALLOCATE PREPARE` shouldn't clear the diagnostics area, unless it produces an error.

A prepared statement is executed with EXECUTE and released with DEALLOCATE PREPARE.

The max_prepared_stmt_count server system variable determines the number of allowed prepared statements that can be prepared on the server. If it is set to `0`, prepared statements are not allowed. If the limit is reached, an error similar to the following will be produced:

```
ERROR 1461 (42000): Can't create more than max_prepared_stmt_count statements
  (current value: 0)
```

Oracle Mode

In Oracle mode, `PREPARE stmt FROM 'SELECT :1, :2'` is used instead of `?`.

Permitted Statements

> **Note:**
>
> The following is valid from MariaDB **10.6.2.**

All statements can be prepared, except PREPARE, EXECUTE, and DEALLOCATE / DROP PREPARE.

Prior to this, not all statements can be prepared. Only the following SQL commands are permitted:

- ALTER TABLE
- ANALYZE TABLE
- BINLOG
- CACHE INDEX
- CALL
- CHANGE MASTER
- CHECKSUM {TABLE | TABLES}
- COMMIT
- {CREATE | DROP} DATABASE
- {CREATE | DROP} INDEX
- {CREATE | RENAME | DROP} TABLE
- {CREATE | RENAME | DROP} USER
- {CREATE | DROP} VIEW

- DELETE
- DESCRIBE
- DO
- EXPLAIN
- FLUSH {TABLE | TABLES | TABLES WITH READ LOCK | HOSTS | PRIVILEGES | LOGS | STATUS | MASTER | SLAVE | DES_KEY_FILE | USER_RESOURCES | QUERY CACHE | TABLE_STATISTICS | INDEX_STATISTICS | USER_STATISTICS | CLIENT_STATISTICS}
- GRANT
- INSERT
- INSTALL {PLUGIN | SONAME}
- HANDLER READ
- KILL
- LOAD INDEX INTO CACHE
- OPTIMIZE TABLE
- REPAIR TABLE
- REPLACE
- RESET {MASTER | SLAVE | QUERY CACHE}
- REVOKE
- ROLLBACK
- SELECT
- SET
- SET GLOBAL SQL_SLAVE_SKIP_COUNTER
- SET ROLE
- SET SQL_LOG_BIN
- SET TRANSACTION ISOLATION LEVEL
- SHOW EXPLAIN
- SHOW {DATABASES | TABLES | OPEN TABLES | TABLE STATUS | COLUMNS | INDEX | TRIGGERS |EVENTS | GRANTS | CHARACTER SET | COLLATION | ENGINES | [PLUGINS SONAME] | PRIVILEGES |PROCESSLIST | PROFILE | PROFILES | VARIABLES | STATUS | WARNINGS | ERRORS |TABLE_STATISTICS | INDEX_STATISTICS | USER_STATISTICS | CLIENT_STATISTICS | AUTHORS |CONTRIBUTORS}
- SHOW CREATE {DATABASE | TABLE | VIEW | PROCEDURE | FUNCTION | TRIGGER | EVENT}
- SHOW {FUNCTION | PROCEDURE} CODE
- SHOW BINLOG EVENTS
- SHOW SLAVE HOSTS
- SHOW {MASTER | BINARY} LOGS
- SHOW {MASTER | SLAVE | TABLES | INNODB | FUNCTION | PROCEDURE} STATUS
- SLAVE {START | STOP}
- TRUNCATE TABLE
- SHUTDOWN
- UNINSTALL {PLUGIN | SONAME}
- UPDATE

Synonyms are not listed here, but can be used. For example, `DESC` can be used instead of `DESCRIBE`.

Compound statements can be prepared too.

Note that if a statement can be run in a stored routine, it will work even if it is called by a prepared statement. For example, SIGNAL can't be directly prepared. However, it is allowed in stored routines. If the x() procedure contains `SIGNAL`, you can still prepare and execute the '`CALL` x();' prepared statement.

`PREPARE` supports most kinds of expressions as well, for example:

```
PREPARE stmt FROM CONCAT('SELECT * FROM ', table_name);
```

When `PREPARE` is used with a statement which is not supported, the following error is produced:

```
ERROR 1295 (HY000): This command is not supported in the prepared statement protocol yet
```

Example

```
CREATE TABLE t1 (a INT,b CHAR(10));
INSERT INTO t1 VALUES (1,"one"),(2, "two"),(3,"three");
PREPARE test FROM "select * from t1 where a=?";
SET @param=2;
EXECUTE test USING @param;
+------+------+
| a    | b    |
+------+------+
|    2 | two  |
+------+------+
SET @param=3;
EXECUTE test USING @param;
+------+-------+
| a    | b     |
+------+-------+
|    3 | three |
+------+-------+
DEALLOCATE PREPARE test;
```

Since identifiers are not permitted as prepared statements parameters, sometimes it is necessary to dynamically compose an SQL statement. This technique is called *dynamic SQL*). The following example shows how to use dynamic SQL:

```
CREATE PROCEDURE test.stmt_test(IN tab_name VARCHAR(64))
BEGIN
    SET @sql = CONCAT('SELECT COUNT(*) FROM ', tab_name);
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END;

CALL test.stmt_test('mysql.user');
+----------+
| COUNT(*) |
+----------+
|        4 |
+----------+
```

Use of variables in prepared statements:

```
PREPARE stmt FROM 'SELECT @x;';

SET @x = 1;

EXECUTE stmt;
+------+
| @x   |
+------+
|    1 |
+------+

SET @x = 0;

EXECUTE stmt;
+------+
| @x   |
+------+
|    0 |
+------+

DEALLOCATE PREPARE stmt;
```

See Also

- EXECUTE Statement
- DEALLOCATE / DROP Prepared Statement
- EXECUTE IMMEDIATE
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

Programmatic & Compound Statements

See BEGIN END.

Group multiple SQL statements into a logical block. This construct defines a compound statement, creating a new scope for variables and exception handling.

See CASE Statement.

Complete reference for CASE Statement in MariaDB. Complete syntax guide with all options, clauses, and practical examples with comprehensive examples and.

See DECLARE CONDITION.

Define named error conditions. This statement associates a name with a specific SQLSTATE or MariaDB error code for easier handling in stored programs.

See DECLARE HANDLER.

Specify actions for error conditions. This statement defines handler routines (CONTINUE or EXIT) to manage exceptions or warnings within a block.

See DECLARE TYPE.

Define data types for Oracle compatibility. This statement allows declaring PL/SQL-style record types and associative arrays, and REF CURSOR types within stored procedures.

See DECLARE Variable.

Initialize local variables within a stored program. This statement defines variables with a specific data type and optional default value.

See FOR.

Loop through a range or cursor result set. This control flow statement repeatedly executes a block of code for each item in a specified range or query.

See GOTO.

Jump to a labeled point in the code. This Oracle-compatible statement transfers execution control to a specific label within the stored program.

See IF.

Execute code based on conditions. This control flow statement runs different blocks of SQL statements depending on whether a specified condition is true.

See ITERATE.

Restart the current loop. This statement jumps back to the beginning of a LOOP, REPEAT, or WHILE block, skipping any remaining statements in the current iteration.

See Labels.

Identify blocks and loops for flow control. Labels provide names for BEGIN…END blocks or loops, allowing them to be targeted by LEAVE, ITERATE, or GOTO statements.

See LEAVE.

Exit a labeled block or loop immediately. This statement terminates the execution of the current loop or compound statement and continues after the block.

See LOOP.

Create a simple loop construct. This statement repeatedly executes a block of code until explicitly terminated by a LEAVE statement.

See REPEAT LOOP.

Execute a block until a condition is met. This loop construct runs at least once and continues repeating as long as the UNTIL condition remains false.

See RESIGNAL.

Propagate error conditions. This statement allows a handler to pass an error condition back to the caller or modify the error information before passing it on.

See RETURN.

Exit a stored function and return a value. This statement terminates function execution and sends the specified result back to the caller.

See SELECT INTO.

Assign query results to variables. This statement retrieves column values from a single row and stores them in local variables or user-defined variables.

See SIGNAL.

Raise a custom error condition. This statement allows stored programs to generate specific error messages and SQLSTATEs to handle application logic exceptions.

See WHILE.

Execute a block while a condition is true. This loop construct checks a condition before each iteration and repeats the block as long as the condition holds.

See Using Compound Statements Outside of Stored Programs.

Use compound statements such as IF and BEGIN…END outside of stored programs, with the delimiter set appropriately.

See Cursors.

Learn about cursors in MariaDB Server's programmatic compound statements. This section details how to iterate over result sets row-by-row within stored procedures and functions.

See Diagnostics.

Learn about diagnostics in programmatic compound statements. This section covers error handling and information retrieval within stored procedures and functions for effective debugging.

This page is licensed: CC BY-SA / Gnu FDL

BEGIN END

Syntax

```
[begin_label:] BEGIN [NOT ATOMIC]
    [statement_list]
END [end_label]
```

`NOT ATOMIC` is required when used outside of a stored procedure. Inside stored procedures or within an anonymous block, `BEGIN` alone starts a new anonymous block.

Description

`BEGIN ... END` syntax is used for writing compound statements. A compound statement can contain multiple statements, enclosed by the `BEGIN` and `END` keywords. statement_list represents a list of one or more statements, each terminated by a semicolon (i.e., `;`) statement delimiter. statement_list is optional, which means that the empty compound statement (`BEGIN END`) is legal.

Note that `END` will perform a commit. If you are running in autocommit mode, every statement will be committed separately. If you are not running in `autocommit` mode, you must execute a COMMIT or ROLLBACK after `END` to get the database up to date.

Use of multiple statements requires that a client is able to send statement strings containing the statement delimiter. This is handled in the mysql command-line client with the DELIMITERcommand.
Changing the `;` end-of-statement delimiter (for example, to `//`) allows `;` to be used in a program body.

A compound statement within a stored program can be labeled. `end_label` cannot be given unless `begin_label` also is present. If both are present, they must be the same.

`BEGIN ... END` constructs can be nested. Each block can define its own variables, a `CONDITION`, a `HANDLER` and a CURSOR, which don't exist in the outer blocks. The most local declarations override the outer objects which use the same name (see example below).

The declarations order is the following:

- DECLARE local variables
- DECLARE CONDITIONs
- DECLARE CURSORs
- DECLARE HANDLERs

Note that `DECLARE HANDLER` contains another `BEGIN ... END` construct.

Here is an example of a very simple, anonymous block:

```
BEGIN NOT ATOMIC
SET @a=1;
CREATE TABLE test.t1(a INT);
END|
```

Below is an example of nested blocks in a stored procedure:

```
CREATE PROCEDURE t( )
BEGIN
   DECLARE x TINYINT UNSIGNED DEFAULT 1;
   BEGIN
      DECLARE x CHAR(2) DEFAULT '02';
       DECLARE y TINYINT UNSIGNED DEFAULT 10;
       SELECT x, y;
   END;
   SELECT x;
END;
```

In this example, a TINYINT variable, `x` is declared in the outer block. But in the inner block `x` is redeclared as a CHAR and an `y` variable is declared. The inner SELECT shows the “new” value of `x`, and the value of `y`. But when x is selected in the outer block, the “old” value is returned. The final SELECT doesn’t try to read `y`, because it doesn’t exist in that context.

See Also

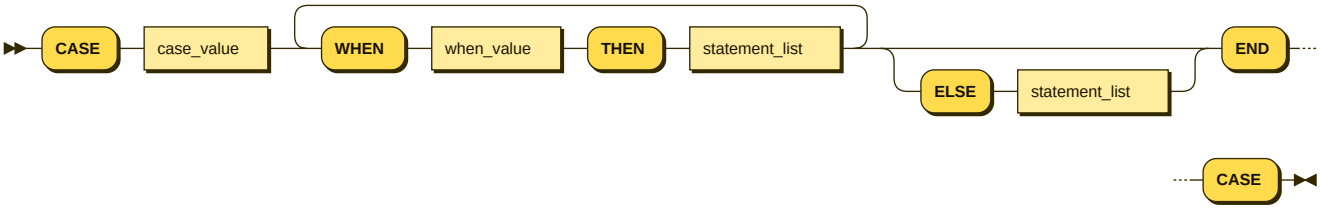
- Using compound statements outside of stored programs
- Changes in Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

CASE Statement

Syntax

```
CASE case_value
    WHEN when_value THEN statement_list
    [WHEN when_value THEN statement_list] ...
    [ELSE statement_list]
END CASE
```



Railroad diagram of the simple CASE form — equivalent to the BNF above

Or:

```
CASE
    WHEN search_condition THEN statement_list
    [WHEN search_condition THEN statement_list] ...
    [ELSE statement_list]
END CASE
```

Description

The text on this page describes the `CASE` statement for stored programs. See the CASE OPERATOR for details on the CASE operator outside of stored programs.

The `CASE` statement for stored programs implements a complex conditional construct. If a `search_condition` evaluates to true, the corresponding SQL statement list is executed. If no search condition matches, the statement list in the `ELSE` clause is executed. Each `statement_list` consists of one or more statements.

The `CASE` statement cannot have an `ELSE NULL` clause, and it is terminated with `END CASE` instead of `END`. implements a complex conditional construct. If a `search_condition` evaluates to true, the corresponding SQL statement list is executed. If no search condition matches, the statement list in the `ELSE` clause is executed. Each `statement_list` consists of one or more statements.

If no when_value or search_condition matches the value tested and the `CASE` statement contains no `ELSE` clause, a Case not found for `CASE` statement error results.

Each statement_list consists of one or more statements; an empty `statement_list` is not allowed. To handle situations where no value is matched by any `WHEN` clause, use an `ELSE` containing an empty BEGIN … END block, as shown in this example:

```
DELIMITER |
CREATE PROCEDURE p()
BEGIN
  DECLARE v INT DEFAULT 1;
  CASE v
    WHEN 2 THEN SELECT v;
    WHEN 3 THEN SELECT 0;
    ELSE BEGIN END;
  END CASE;
END;
|
```

The indentation used here in the `ELSE` clause is for purposes of clarity only and is not otherwise significant. See Delimiters in the mariadb client for more on the use of the delimiter command.

Note: The syntax of the `CASE` statement used inside stored programs differs slightly from that of the SQL CASE expression described in CASE OPERATOR. The `CASE` statement cannot have an `ELSE NULL` clause, and it is terminated with `END CASE` instead of `END`.

See Also

- The CASE operator, which differs from the CASE statement described above.
- The IF statement.

This page is licensed: GPLv2, originally from fill_help_tables.sql

DECLARE CONDITION

Syntax

```
DECLARE condition_name CONDITION FOR condition_value

condition_value:
    SQLSTATE [VALUE] sqlstate_value
  | mysql_error_code
```

Description

The `DECLARE ... CONDITION` statement defines a named error condition. It specifies a condition that needs specific handling and associates a name with that condition. Later, the name can be used in a DECLARE … HANDLER, SIGNAL or RESIGNAL statement (as long as the statement is located in the same BEGIN … END block).

Conditions must be declared after local variables, but before CURSORs and HANDLERs.

A condition_value for `DECLARE ... CONDITION` can be an SQLSTATE value (a 5-character string literal) or a MySQL error code (a number). You should not use `SQLSTATE` value '00000' or MySQL error code 0, because those indicate success rather than an error condition. If you try, or if you specify an invalid `SQLSTATE` value, an error like this is produced:

```
ERROR 1407 (42000): Bad SQLSTATE: '00000'
```

For a list of `SQLSTATE` values and MariaDB error codes, see MariaDB Error Codes.

This page is licensed: GPLv2, originally from fill_help_tables.sql

DECLARE HANDLER

Syntax

```
DECLARE handler_type HANDLER
    FOR condition_value [, condition_value] ...
    statement

handler_type:
    CONTINUE
  | EXIT
  | UNDO

condition_value:
    SQLSTATE [VALUE] sqlstate_value
  | condition_name
  | SQLWARNING
  | NOT FOUND
  | SQLEXCEPTION
  | mariadb_error_code
```

Description

The `DECLARE ... HANDLER` statement specifies handlers that each may deal with one or more conditions. If one of these conditions occurs, the specified statement is executed. statement can be a simple statement (for example, `SET var_name = value`), or it can be a compound statement written using BEGIN and END.

Handlers must be declared after local variables, a `CONDITION` and a CURSOR.

For a `CONTINUE` handler, execution of the current program continues after execution of the handler statement. For an EXIT handler, execution terminates for the BEGIN … END compound statement in which the handler is declared. (This is true even if the condition occurs in an inner block.) The `UNDO` handler type statement is not supported.

If a condition occurs for which no handler has been declared, the default action is `EXIT`.

A condition_value for `DECLARE ... HANDLER` can be any of the following values:

- An SQLSTATE value (a 5-character string literal) or a MariaDB error code (a number). You should not use `SQLSTATE` value '00000' or MariaDB error code 0, because those indicate success rather than an error condition. For a list of `SQLSTATE` values and MariaDB error codes, see MariaDB Error Codes.
- A condition name previously specified with `DECLARE ... CONDITION`. It must be in the same stored program. See DECLARE CONDITION.

- `SQLWARNING` is shorthand for the class of SQLSTATE values that begin with '01'.
- `NOT FOUND` is shorthand for the class of `SQLSTATE` values that begin with '02'. This is relevant only the context of cursors and is used to control what happens when a cursor reaches the end of a data set. If no more rows are available, a No Data condition occurs with `SQLSTATE` value 02000. To detect this condition, you can set up a handler for it (or for a `NOT FOUND` condition). An example is shown in Cursor Overview. This condition also occurs for SELECT … INTO var_list statements that retrieve no rows.
- `SQLEXCEPTION` is shorthand for the class of `SQLSTATE` values that do not begin with '00', '01', or '02'.

When an error raises, in some cases it could be handled by multiple `HANDLER`s. For example, there may be an handler for 1050 error, a separate handler for the 42S01 SQLSTATE, and another separate handler for the `SQLEXCEPTION` class: in theory all occurrences of `HANDLER` may catch the 1050 error, but MariaDB chooses the `HANDLER` with the highest precedence. Here are the precedence rules:

- Handlers which refer to an error code have the highest precedence.
- Handlers which refer to a SQLSTATE come next.
- Handlers which refer to an error class have the lowest precedence.

In some cases, a statement could produce multiple errors. If this happens, in some cases multiple handlers could have the highest precedence. In such cases, the choice of the handler is indeterminate.

Note that if an error occurs within a `CONTINUE HANDLER` block, it can be handled by another `HANDLER`. However, a `HANDLER` which is already in the stack (that is, it has been called to handle an error and its execution didn't finish yet) cannot handle new errors—this prevents endless loops. For example, suppose that a stored procedure contains a `CONTINUE HANDLER` for `SQLWARNING` and another `CONTINUE HANDLER` for `NOT FOUND`. At some point, a `NOT FOUND` error occurs, and the execution enters the `NOT FOUND HANDLER`. But within that handler, a warning occurs, and the execution enters the `SQLWARNING HANDLER`. If another `NOT FOUND` error occurs, it cannot be handled again by the `NOT FOUND HANDLER`, because its execution is not finished.

When a `DECLARE HANDLER` block can handle more than one error condition, it may be useful to know which errors occurred. To do so, you can use the GET DIAGNOSTICS statement.

An error that is handled by a `DECLARE HANDLER` construct can be issued again using the RESIGNAL statement.

Below is an example using `DECLARE HANDLER`:

```
CREATE TABLE test.t (s1 INT, PRIMARY KEY (s1));

DELIMITER //

CREATE PROCEDURE handlerdemo ( )
     BEGIN
       DECLARE CONTINUE HANDLER FOR SQLSTATE '23000' SET @x2 = 1;
       SET @x = 1;
       INSERT INTO test.t VALUES (1);
       SET @x = 2;
       INSERT INTO test.t VALUES (1);
       SET @x = 3;
     END;
     //

DELIMITER ;

CALL handlerdemo( );

SELECT @x;
+------+
| @x   |
+------+
|    3 |
+------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DECLARE TYPE

> **Note:**
>
> This feature is available from MariaDB 12.1.

Overview

The `DECLARE TYPE` declaration specifies user-defined data types that must be compatible with Oracle within stored procedures and anonymous blocks. It provides Oracle-compatible type declarations, such as record types, associative arrays, and `REF CURSOR` types, which allows for more flexible data handling and better compatibility with Oracle PL/SQL.

Associative arrays (`INDEX BY` tables) offer an in-memory key-value structure for quick data storage and effective lookups. REF CURSOR types allow for the creation of cursor variables that refer to query result sets and can be passed between program blocks.

The general syntax for defining associative array types is as follows:

```
DECLARE
   TYPE type_name TABLE OF rec_type_name INDEX BY idx_type_name
```

- `type_name` supports explicit and anchored data types (for instance, `t1.col1%TYPE`).
- `rec_type_name` supports scalar and record data types.
- The `INDEX BY` clause defines the key type using `idx_type_name` , which supports integer and string data types.

Associative Arrays

In Oracle, associative arrays (called index-by tables) are sparse collections of elements indexed by keys, which can be integers or strings.

Here's an example of how to declare an associative array in Oracle:

```
DECLARE
  TYPE array_t IS TABLE OF VARCHAR2(64) INDEX BY PLS_INTEGER;
  array array_t;
BEGIN
  array(1) := 'Hello';
  array(2) := 'World';
  DBMS_OUTPUT.PUT_LINE(array(1));
END;
```

Methods

Associative arrays support the following methods:

- `FIRST` — a function that returns the first key
- `LAST` — a function that returns the last key
- `NEXT` — a function that returns the key after the given one
- `PRIOR` — a function that returns the key before the given one
- `COUNT` — a function that returns the number of elements
- `EXISTS` — a function that returns `TRUE` if the key exists
- `DELETE` — a procedure that removes a specific key, or clears the array

While the MariaDB implementation is largely aligned with Oracle's implementation, there are a few differences:

- **Only literals as keys in the constructor**: When using constructors, keys must be literals — Oracle allows expressions.
- **Collation control**: Instead of `NLS_SORT` or `NLS_COMP`, MariaDB uses the SQL-standard `COLLATE` clause.
- **No nested associative arrays**: Arrays of arrays are not supported.

These differences are largely rooted in architectural constraints — MariaDB is aiming at staying as close to Oracle semantics as possible while maintaining performance and predictability.

Examples

Associative Array of Scalar Elements

Explicit type_name

```
DECLARE
  TYPE salary IS TABLE OF NUMBER INDEX BY VARCHAR2(20);
  salary_list salary;
  name VARCHAR2(20);
BEGIN
  salary_list('Rajnisj') := 62000;
  salary_list('James') := 78000;
  name:= salary_list.FIRST;
  WHILE name IS NOT NULL
  LOOP
    dbms_output.put_line(name || ' ' || TO_CHAR(salary_list(name)));
    name:= salary_list.NEXT(name);
  END LOOP;
END;
/
```

Anchored type_name

```
CREATE TABLE t1 (a INT);
DECLARE
  TYPE salary IS TABLE OF t1.a%TYPE INDEX BY VARCHAR2(20);
  salary_list salary;
  name VARCHAR2(20);
BEGIN
  salary_list('Rajnisj') := 62000;
  salary_list('James') := 78000;
  name:= salary_list.FIRST;
  WHILE name IS NOT NULL
  LOOP
    dbms_output.put_line(name || ' ' || TO_CHAR(salary_list(name)));
    name:= salary_list.NEXT(name);
  END LOOP;
END;
/
```

Associative Array of Records

Using Explicit Data Types

```
DECLARE
  TYPE person_t IS RECORD
  (
    first_name VARCHAR(64),
    last_name VARCHAR(64)
  );
  person person_t;
  TYPE table_of_peson_t IS TABLE OF person_t INDEX BY VARCHAR(20);
  person_by_nickname table_of_peson_t;
  nick VARCHAR(20);
BEGIN
  person_by_nickname('Monty') := person_t('Michael', 'Widenius');
  person_by_nickname('Serg') := person_t('Sergei ', 'Golubchik');
  nick:= person_by_nickname.FIRST;
  WHILE nick IS NOT NULL
  LOOP
    person:= person_by_nickname(nick);
    dbms_output.put_line(nick || ' ' || person.first_name || ' '|| person.last_name);
    nick:= person_by_nickname.NEXT(nick);
  END LOOP;
  /
```

Using Anchored Data Types

```
DROP TABLE persons;
CREATE TABLE persons (nickname VARCHAR(64), first_name VARCHAR(64), last_name VARCHAR(64));
INSERT INTO persons VALUES ('Serg','Sergei ', 'Golubchik');
INSERT INTO persons VALUES ('Monty','Michael', 'Widenius');
DECLARE
  TYPE table_of_person_t IS TABLE OF persons%ROWTYPE INDEX BY persons.nickname%TYPE;
  person_by_nickname table_of_person_t;
  nickname persons.nickname%TYPE;
  person persons%ROWTYPE;
BEGIN
  FOR rec IN (SELECT * FROM persons)
  LOOP
    person_by_nickname(rec.nickname):= rec;
  END LOOP;

  nickname:= person_by_nickname.FIRST;
  WHILE nickname IS NOT NULL
  LOOP
    person:= person_by_nickname(nickname);
    dbms_output.put_line(person.nickname || ' ' || person.first_name || ' '|| person.last_name);
    nickname:= person_by_nickname.NEXT(nickname);
  END LOOP;
END;
/
```

RECORD Types

In `sql_mode=ORACLE`, the `TYPE ... IS RECORD` statement allows you to define a user-defined data structure consisting of one or more fields.

Previously, TYPE-defined `RECORD` types could only be used in local program blocks and not in routine parameters or function `RETURN` clauses.

Syntax

```
TYPE record_type_name IS RECORD (
    field_name data_type [NOT NULL] [DEFAULT expr],
    ...
);
```

- `record_type_name` is the name of the new record type.
- Each `field_name` is a named attribute of the record.
- Each `field_type` is a valid MariaDB data type or anchored type (`%TYPE`, `%ROWTYPE`).

RECORD Types in Routine Parameters and Function RETURN

Starting with MariaDB 13.0.1, custom types defined with `DECLARE TYPE` (including `RECORD` and `REF CURSOR`) can be used as:

- Parameters of stored procedures and functions
- `RETURN` types of stored functions

Earlier, the use of a `RECORD` type in these contexts was prohibited by the MariaDB grammar and resulted in:

```
ERROR 4161 (HY000): Unknown data type
```

This change improves compatibility with Oracle PL/SQL, especially for private helper routines inside packages and for `REF CURSOR` usage.

Using RECORDs in Routine Parameters

Starting with MariaDB 13.0.1, stored procedures within the same package can use a `RECORD` type declared inside the package body as a parameter.

```
SET sql_mode=ORACLE;
DELIMITER $$
CREATE OR REPLACE PACKAGE pkg1 AS
  PROCEDURE p1();
END;
CREATE OR REPLACE PACKAGE BODY pkg1 AS
  TYPE rec0_t IS RECORD (a INT, b VARCHAR(2), c INT);
  PROCEDURE private_p1(p0 rec0_t) AS
  BEGIN
    NULL;
  END;
  PROCEDURE p1 AS
  BEGIN
    CALL private_p1(a0);
  END;
END;
$$
DELIMITER ;
```

Using RECORDs as a Function Return Type

A `RECORD` type declared inside a package body can also be used as the return type for a stored function inside the same package.

```
SET sql_mode=ORACLE;
DELIMITER $$
CREATE OR REPLACE PACKAGE pkg1 AS
  PROCEDURE p1();
END;
CREATE OR REPLACE PACKAGE BODY pkg1 AS
  TYPE rec0_t IS RECORD (a INT, b VARCHAR(2), c INT);
  FUNCTION private_f1() RETURN rec0_t AS
  BEGIN
    RETURN NULL;
  END;
  PROCEDURE p1() AS
  BEGIN
    DO private_f1();
  END;
END;
$$
DELIMITER ;
```

Requirement

This feature requires Oracle SQL mode at package creation time. The SQL mode is stored when the package is created, so `SET sql_mode=ORACLE` must be executed before running `CREATE PACKAGE` and `CREATE PACKAGE BODY`. It does not need to be set when calling the package's stored procedures or functions.

REF CURSOR Types

MariaDB supports Oracle-compatible `REF CURSOR` type declarations as a part of the `DECLARE TYPE` statement. A `REF CURSOR` is a cursor variable that can refer to a query result set and be passed across program blocks, such as stored procedures or functions.

`REF CURSOR` types must be specified with a `TYPE` declaration before any variables of that type can be declared. It can be defined as weak or strong based on whether a return type is specified.

Syntax

```
ref_cursor_type_definition:
   TYPE type_name IS REF CURSOR [ RETURN return_type ];
```

- If `RETURN` is omitted, the cursor type is weak.
- If `RETURN` is specified, the cursor type is strong and limited to a single row structure.

Weak REF CURSOR

A weak `REF CURSOR` type is defined without a `RETURN` clause. It can be used with any query result.

```
TYPE weak_cursor IS REF CURSOR; -- weak type: no RETURN clause
```

Strong REF CURSOR

A strong `REF CURSOR` type is defined by a `RETURN` clause that defines the row structure the cursor must return.

```
TYPE storong_cursor IS REF CURSOR RETURN employees%ROWTYPE;  -- strong type
```

Supported RETURN Types

MariaDB supports the following `RETURN` clause formats:

RETURN `record_type`

The return type can be a user-defined record type. Columns that correspond to the field names and types of the record must be returned by the query run against the cursor.

```
DROP TABLE t1;
CREATE TABLE t1 (a INT,b VARCHAR(10));
INSERT INTO t1 VALUES (10,'b10');
DECLARE
  TYPE rec_t IS RECORD (a INT, b VARCHAR(19));
  TYPE cur_rec_t IS REF CURSOR RETURN rec_t;
  c0 cur_rec_t;
  r0 rec_t;
BEGIN
  OPEN c0 FOR SELECT * FROM t1;
  LOOP
    FETCH c0 INTO r0;
    EXIT WHEN c0%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(r0.a || ' ' || r0.b);
  END LOOP;
END;
```

RETURN `record_type%ROWTYPE`

The declared `RECORD` type can include anchored field types, such as `t1.b%TYPE`. This enables the record and therefore the cursor to automatically modify in the event that the underlying table column type changes.

```
DROP TABLE t1;
CREATE TABLE t1 (a INT,b VARCHAR(10));
INSERT INTO t1 VALUES (10,'b10');
DECLARE
  TYPE rec_t IS RECORD (a t1.a%TYPE, b t1.b%TYPE);
  TYPE cur_rec_t IS REF CURSOR RETURN rec_t;
  r0 rec_t;
  c0 cur_rec_t;
BEGIN
  OPEN c0 FOR SELECT * FROM t1;
  LOOP
    FETCH c0 INTO r0;
    EXIT WHEN c0%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(r0.a || ' ' || r0.b);
  END LOOP;
END;
```

RETURN `record_variable%TYPE`

The cursor's return type is determined from a defined variable using the `%TYPE` attribute. At the time of declaration, the cursor inherits the type of the variable's row structure.

```
CREATE OR REPLACE PROCEDURE p1 IS
  TYPE rec0_t IS RECORD (a INT, b VARCHAR(10));
  v0 rec0_t;
  TYPE cur0_t IS REF CURSOR RETURN v0%TYPE;
  c0 cur0_t;
BEGIN
  OPEN c0 FOR SELECT 1,2 FROM DUAL;
  FETCH c0 INTO v0;
  DBMS_OUTPUT.PUT_LINE(v0.a || v0.b);
  CLOSE c0;
END;
```

RETURN `cursor%ROWTYPE`

The row structure of an existing static cursor declared in the same block serves as the anchor for the cursor's return type. The column types and names from that cursor's `SELECT` statement are inherited by the `REF CURSOR`.

```
CREATE OR REPLACE PROCEDURE p1 IS
  CURSOR cs IS SELECT 1 AS a,2 AS b FROM DUAL;
  TYPE curs_t IS REF CURSOR RETURN cs%ROWTYPE;
  c0 curs_t;
  v0 cs%ROWTYPE;
BEGIN
  OPEN c0 FOR SELECT 1,2 FROM DUAL;
  FETCH c0 INTO v0;
  DBMS_OUTPUT.PUT_LINE(v0.a || v0.b);
  CLOSE c0;
END;
```

RETURN `table_or_view%ROWTYPE`

Using `%ROWTYPE`, the return type of the cursor is directly linked to a table or view row structure. This cursor variable can only be used with queries that return all columns from the referenced table or view in the same sequence.

```
DROP TABLE t1;
CREATE TABLE t1 (a INT,b VARCHAR(10));
INSERT INTO t1 VALUES (10,'b10');
DECLARE
  TYPE cur_rec_t IS REF CURSOR RETURN t1%ROWTYPE;
  c0 cur_rec_t;
  r0 t1%ROWTYPE;
BEGIN
  OPEN c0 FOR SELECT * FROM t1;
  LOOP
    FETCH c0 INTO r0;
    EXIT WHEN c0%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(r0.a || ' ' || r0.b);
  END LOOP;
END;
```

RETURN `cursor_variable%ROWTYPE`

The cursor's return type is derived from another `REF CURSOR` variable using `%ROWTYPE`. This allows for the connection of cursor type declarations, so that a second cursor inherits its structure from a previously specified cursor variable.

```
DROP TABLE t1;
CREATE TABLE t1 (a INT,b VARCHAR(10));
INSERT INTO t1 VALUES (10,'b10');
DECLARE
  TYPE cur1_t IS REF CURSOR RETURN t1%ROWTYPE;
  c1 cur1_t;
  TYPE cur0_t IS REF CURSOR RETURN c1%ROWTYPE;
  c0 cur0_t;
  r0 c0%ROWTYPE;
BEGIN
  OPEN c0 FOR SELECT * FROM t1;
  LOOP
    FETCH c0 INTO r0;
    EXIT WHEN c0%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(r0.a || ' ' || r0.b);
  END LOOP;
END;
```

See Also

- DECLARE CURSOR
- DECLARE Variable
- Oracle Mode
- CREATE FUNCTION
- CREATE PROCEDURE

This page is licensed: CC BY-SA / Gnu FDL

DECLARE Variable

Syntax

```
DECLARE var_name [, var_name] ... [[ROW] TYPE OF]] type [DEFAULT value]
```

Description

This statement is used to declare local variables within stored programs. To provide a default value for the variable, include a `DEFAULT` clause. The value can be specified as an expression (even subqueries are permitted); it need not be a constant. If the `DEFAULT` clause is missing, the initial value is `NULL`.

Local variables are treated like stored routine parameters with respect to data type and overflow checking. See CREATE PROCEDURE.

Local variables must be declared before `CONDITION`s, CURSORs and `HANDLER`s.

Local variable names are not case sensitive.

The scope of a local variable is within the `BEGIN ... END` block where it is declared. The variable can be referred to in blocks nested within the declaring block, except those blocks that declare a variable with the same name.

TYPE OF / ROW TYPE OF

Anchored data types allow a data type to be defined based on another object, such as a table row, rather than specifically set in the declaration. If the anchor object changes, so will the anchored data type. This can lead to routines being easier to maintain, so that if the data type in the table is changed, it will automatically be changed in the routine as well.

Variables declared with `ROW TYPE OF` will have the same features as implicit ROW variables. It is not possible to use `ROW TYPE OF` variables in a LIMIT clause.

The real data type of `TYPE OF` and `ROW TYPE OF table_name` will become known at the very beginning of the stored routine call. ALTER TABLE or DROP TABLE statements performed inside the current routine on the tables that appear in anchors won't affect the data type of the anchored variables, even if the variable is declared after an ALTER TABLE or DROP TABLE statement.

The real data type of a `ROW TYPE OF cursor_name` variable will become known when execution enters into the block where the variable is declared. Data type instantiation will happen only once. In a cursor `ROW TYPE OF` variable that is declared inside a loop, its data type will become known on the very first iteration and won't change on further loop iterations.

The tables referenced in `TYPE OF` and `ROW TYPE OF` declarations will be checked for existence at the beginning of the stored routine call. CREATE PROCEDURE or CREATE FUNCTION will not check the referenced tables for existence.

> **Note:**
>
> **Dynamic Cursors Limitation**
>
> `ROW TYPE OF` is not supported for dynamic cursors (cursors declared for a prepared statement). This is because the cursor's structure is dependent on the prepared statement, which is not known at the time the stored procedure context is created.

Examples

```
DECLARE name VARCHAR(5) DEFAULT 'monty';
  DECLARE x INT DEFAULT 10;
  DECLARE Y SMALLINT;
```

`TYPE OF` and `ROW TYPE OF` :

```
DECLARE tmp TYPE OF t1.a; -- Get the data type from the column {{a}} in the table {{t1}}

DECLARE rec1 ROW TYPE OF t1; -- Get the row data type from the table {{t1}}

DECLARE rec2 ROW TYPE OF cur1; -- Get the row data type from the cursor {{cur1}}
```

See Also

- User-Defined variables

This page is licensed: GPLv2, originally from fill_help_tables.sql

FOR

Syntax

Integer range `FOR` loop:

```
[begin_label:]
FOR var_name IN [ REVERSE ] lower_bound .. upper_bound
DO statement_list
END FOR [ end_label ]
```

Explicit cursor `FOR` loop:

```
[begin_label:]
FOR record_name IN cursor_name [ ( cursor_actual_parameter_list)]
DO statement_list
END FOR [ end_label ]
```

Explicit cursor `FOR` loop (Oracle mode):

```
[begin_label:]
FOR record_name IN cursor_name [ ( cursor_actual_parameter_list)]
LOOP
  statement_list
END LOOP [ end_label ]
```

Implicit cursor `FOR` loop:

```
[begin_label:]
FOR record_name IN ( select_statement )
DO statement_list
END FOR [ end_label ]
```

Description

`FOR` loops allow code to be executed a fixed number of times.

In an integer range `FOR` loop, MariaDB will compare the lower bound and upper bound values, and assign the lower bound value to a counter. If `REVERSE` is not specified, and the upper bound value is greater than or equal to the counter, the counter will be incremented and the statement will continue, after which the loop is entered again. If the upper bound value is greater than the counter, the loop will be exited.

> **Note:**
>
> **Dynamic Cursors Limitation**
>
> `FOR` loops cannot be used with dynamic cursors (cursors declared for prepared statement names). To handle dynamic cursors, employ explicit `OPEN`, `FETCH`, and `CLOSE` statements inside a `LOOP` or similar construct.

If REVERSE is specified, the counter is decremented, and the upper bound value needs to be less than or equal for the loop to continue.

Examples

Integer range `FOR` loop:

```
CREATE TABLE t1 (a INT);

DELIMITER //

FOR i IN 1..3
DO
  INSERT INTO t1 VALUES (i);
END FOR;
//

DELIMITER ;

SELECT * FROM t1;
+------+
| a    |
+------+
|    1 |
|    2 |
|    3 |
+------+
```

`REVERSE` integer range `FOR` loop:

```
CREATE OR REPLACE TABLE t1 (a INT);

DELIMITER //
FOR i IN REVERSE 4..12
    DO
    INSERT INTO t1 VALUES (i);
END FOR;
//
Query OK, 9 rows affected (0.422 sec)

DELIMITER ;

SELECT * FROM t1;
+------+
| a    |
+------+
|   12 |
|   11 |
|   10 |
|    9 |
|    8 |
|    7 |
|    6 |
|    5 |
|    4 |
+------+
```

Explicit cursor in Oracle mode:

```
SET sql_mode=ORACLE;

CREATE OR REPLACE TABLE t1 (a INT, b VARCHAR(32));

INSERT INTO t1 VALUES (10,'b0');
INSERT INTO t1 VALUES (11,'b1');
INSERT INTO t1 VALUES (12,'b2');

DELIMITER //

CREATE OR REPLACE PROCEDURE p1(pa INT) AS
  CURSOR cur(va INT) IS
    SELECT a, b FROM t1 WHERE a=va;
BEGIN
  FOR rec IN cur(pa)
  LOOP
    SELECT rec.a, rec.b;
  END LOOP;
END;
//

DELIMITER ;

CALL p1(10);
+-------+-------+
| rec.a | rec.b |
+-------+-------+
|    10 | b0    |
+-------+-------+

CALL p1(11);
+-------+-------+
| rec.a | rec.b |
+-------+-------+
|    11 | b1    |
+-------+-------+

CALL p1(12);
+-------+-------+
| rec.a | rec.b |
+-------+-------+
|    12 | b2    |
+-------+-------+

CALL p1(13);
Query OK, 0 rows affected (0.000 sec)
```

See Also

- LOOP

This page is licensed: CC BY-SA / Gnu FDL

GOTO

Syntax

```
GOTO label
```

Description

The `GOTO` statement causes the code to jump to the specified label, and continue operating from there. It is only accepted when in Oracle mode.

Example

```
SET sql_mode=ORACLE;

DELIMITER //

CREATE OR REPLACE PROCEDURE p1 AS

BEGIN

  SELECT 1;
  GOTO label;
  SELECT 2;
  <<label>>
  SELECT 3;

END;

//

DELIMITER

CALL p1();
+---+
| 1 |
+---+
| 1 |
+---+
1 row in set (0.000 sec)

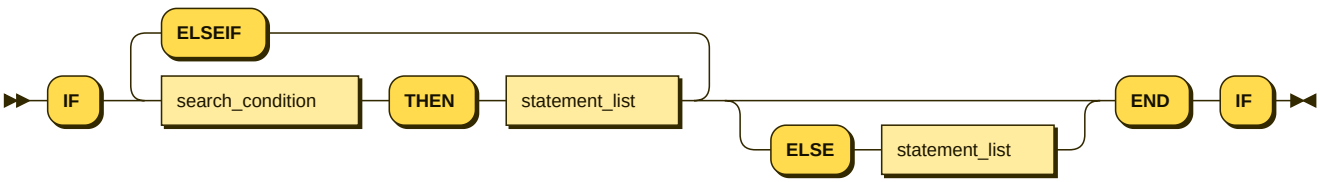
+---+
| 3 |
+---+
| 3 |
+---+
1 row in set (0.000 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

IF

Syntax

```
IF search_condition THEN statement_list
    [ELSEIF search_condition THEN statement_list] ...
    [ELSE statement_list]
END IF
```



Railroad diagram of the IF statement — equivalent to the BNF above

Description

`IF` implements a basic conditional construct. If the `search_condition` evaluates to true, the corresponding SQL statement list is executed. If no `search_condition` matches, the statement list in the `ELSE` clause is executed. Each statement_list consists of one or more statements.

See Also

- The IF() function, which differs from the `IF` statement described above.
- Changes in Oracle mode.
- The CASE statement.

This page is licensed: GPLv2, originally from fill_help_tables.sql

ITERATE

Syntax

```
ITERATE label
```

`ITERATE` can appear only within LOOP, REPEAT, and WHILE statements. `ITERATE` means "do the loop again", and uses the statement's label to determine which statements to repeat. The label must be in the same stored program, not in a caller procedure.

If you try to use `ITERATE` with a non-existing label, or if the label is associated to a construct which is not a loop, the following error will be produced:

```
ERROR 1308 (42000): ITERATE with no matching label: <label_name>
```

Below is an example of how `ITERATE` might be used:

```
CREATE PROCEDURE doiterate(p1 INT)
BEGIN
  label1: LOOP
    SET p1 = p1 + 1;
    IF p1 < 10 THEN ITERATE label1; END IF;
    LEAVE label1;
  END LOOP label1;
  SET @x = p1;
END
```

See Also

- LEAVE - Exits a loop (or any labeled code block)

This page is licensed: GPLv2, originally from fill_help_tables.sql

Labels

Syntax

```
label: <construct>
[label]
```

Labels are MariaDB identifiers which can be used to identify a BEGIN … END construct or a loop. They have a maximum length of 16 characters and can be quoted with backticks (i.e.., ```).

Labels have a start part and an end part. The start part must precede the portion of code it refers to, must be followed by a colon (`:`) and can be on the same or different line. The end part is optional and adds nothing, but can make the code more readable. If used, the end part must precede the construct's delimiter (`;`). Constructs identified by a label can be nested. Each construct can be identified by only one label.

Labels need not be unique in the stored program they belong to. However, a label for an inner loop cannot be identical to a label for an outer loop. In this case, the following error would be produced:

```
ERROR 1309 (42000): Redefining label <label_name>
```

LEAVE and ITERATE statements can be used to exit or repeat a portion of code identified by a label. They must be in the same Stored Routine, Trigger or Event which contains the target label.

Below is an example using a simple label that is used to exit a LOOP:

```
CREATE PROCEDURE `test_sp`()
BEGIN
   `my_label`:
   LOOP
      SELECT 'looping';
      LEAVE `my_label`;
   END LOOP;
   SELECT 'out of loop';
END;
```

The following label is used to exit a procedure, and has an end part:

```
CREATE PROCEDURE `test_sp`()
`my_label`:
BEGIN
   IF @var = 1 THEN
      LEAVE `my_label`;
   END IF;
   DO something();
END `my_label`;
```

This page is licensed: CC BY-SA / Gnu FDL

LEAVE

Syntax

```
LEAVE label
```

This statement is used to exit the flow control construct that has the given label. The label must be in the same stored program, not in a caller procedure. `LEAVE` can be used within BEGIN … END or loop constructs (LOOP, REPEAT, WHILE). In Stored Procedures, Triggers and Events, LEAVE can refer to the outmost `BEGIN ... END` construct; in that case, the program exits the procedure. In Stored Functions, RETURN can be used instead.

> **Note:**
>
> `LEAVE` cannot be used to exit a DECLARE HANDLER block.

If you try to `LEAVE` a non-existing label, or if you try to `LEAVE` a `HANDLER` block, the following error will be produced:

```
ERROR 1308 (42000): LEAVE with no matching label: <label_name>
```

The following example uses `LEAVE` to exit the procedure if a condition is true:

```
CREATE PROCEDURE proc(IN p TINYINT)
CONTAINS SQL
`whole_proc`:
BEGIN
   SELECT 1;
   IF p < 1 THEN
      LEAVE `whole_proc`;
   END IF;
   SELECT 2;
END;

CALL proc(0);
+---+
| 1 |
+---+
| 1 |
+---+
```

See Also

- ITERATE - Repeats a loop execution

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOOP

Syntax

```
[begin_label:] LOOP
    statement_list
END LOOP [end_label]
```

Description

`LOOP` implements a simple loop construct, enabling repeated execution of the statement list, which consists of one or more statements, each terminated by a semicolon (i.e., `;`) statement delimiter. The statements within the loop are repeated until the loop is exited; usually this is accomplished with a LEAVE statement.

A `LOOP` statement can be labeled. `end_label` cannot be given unless `begin_label` also is present. If both are present, they must be the same.

See Delimiters in the mariadb client for more on delimiter usage in the client.

See Also

- LOOP in Oracle mode
- ITERATE
- LEAVE
- FOR Loops

This page is licensed: GPLv2, originally from fill_help_tables.sql

REPEAT LOOP

Syntax

```
[begin_label:] REPEAT
    statement_list
UNTIL search_condition
END REPEAT [end_label]
```

The statement list within a `REPEAT` statement is repeated until the search_condition is true. Thus, a `REPEAT` always enters the loop at least once. statement_list consists of one or more statements, each terminated by a semicolon (i.e., `;`) statement delimiter.

A `REPEAT` statement can be labeled. end_label cannot be given unless begin_label is also present. If both are present, they must be the same.

See Delimiters in the mariadb client for more on client delimiter usage.

```
DELIMITER //

CREATE PROCEDURE dorepeat(p1 INT)
  BEGIN
    SET @x = 0;
    REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
  END
//

CALL dorepeat(1000)//

SELECT @x//
+------+
| @x   |
+------+
| 1001 |
+------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

RESIGNAL

Syntax

```
RESIGNAL [error_condition]
    [SET error_property
    [, error_property] ...]

error_condition:
    SQLSTATE [VALUE] 'sqlstate_value'
  | condition_name

error_property:
    error_property_name = <error_property_value>

error_property_name:
    CLASS_ORIGIN
  | SUBCLASS_ORIGIN
  | MESSAGE_TEXT
  | MYSQL_ERRNO
  | CONSTRAINT_CATALOG
  | CONSTRAINT_SCHEMA
  | CONSTRAINT_NAME
  | CATALOG_NAME
  | SCHEMA_NAME
  | TABLE_NAME
  | COLUMN_NAME
  | CURSOR_NAME
```

Description

The syntax of `RESIGNAL` and its semantics are very similar to SIGNAL. This statement can only be used within an error HANDLER. It produces an error, like SIGNAL. `RESIGNAL` clauses are the same as `SIGNAL`, except that they all are optional, even SQLSTATE. All the properties which are not specified in `RESIGNAL`, will be identical to the

properties of the error that was received by the error HANDLER. For a description of the clauses, see diagnostics area.

> **Note:**
>
> `RESIGNAL` does not empty the diagnostics area. It just appends another error condition.

`RESIGNAL`, without any clauses, produces an error which is identical to the error that was received by HANDLER.

If used out of a HANDLER construct, RESIGNAL produces the following error:

```
ERROR 1645 (0K000): RESIGNAL when handler not active
```

CURRENT

If a HANDLER contains a CALL to another procedure, that procedure can use `RESIGNAL`, but trying to do this raises the above error.

< 5.6

If a HANDLER contains a CALL to another procedure, that procedure can use `RESIGNAL`.

For a list of `SQLSTATE` values and MariaDB error codes, see MariaDB Error Codes.

The following procedure tries to query two tables which don't exist, producing a 1146 error in both cases. Those errors will trigger the HANDLER. The first time the error will be ignored, and the client will not receive it, but the second time, the error is re-signaled, so the client will receive it.

```
CREATE PROCEDURE test_error( )
BEGIN
   DECLARE CONTINUE HANDLER
      FOR 1146
   BEGIN
   IF @hide_errors IS FALSE THEN
      RESIGNAL;
   END IF;
   END;
   SET @hide_errors = TRUE;
   SELECT 'Next error will be ignored' AS msg;
   SELECT `c` FROM `temptab_one`;
   SELECT 'Next error won''t be ignored' AS msg;
   SET @hide_errors = FALSE;
   SELECT `c` FROM `temptab_two`;
END;

CALL test_error( );

+----------------------------+
| msg                        |
+----------------------------+
| Next error will be ignored |
+----------------------------+

+----------------------------+
| msg                         |
+----------------------------+
| Next error won't be ignored |
+----------------------------+

ERROR 1146 (42S02): Table 'test.temptab_two' doesn't exist
```

The following procedure re-signals an error, modifying only the error message to clarify the cause of the problem.

```
CREATE PROCEDURE test_error()
BEGIN
   DECLARE CONTINUE HANDLER
   FOR 1146
   BEGIN
      RESIGNAL SET
      MESSAGE_TEXT = '`temptab` does not exist';
   END;
   SELECT `c` FROM `temptab`;
END;

CALL test_error( );
ERROR 1146 (42S02): `temptab` does not exist
```

```
CREATE PROCEDURE handle_error()
BEGIN
  RESIGNAL;
END;
CREATE PROCEDURE p()
BEGIN
  DECLARE EXIT HANDLER FOR SQLEXCEPTION CALL p();
  SIGNAL SQLSTATE '45000';
END;
```

See Also

- Diagnostics Area
- SIGNAL
- HANDLER
- Stored Routines
- MariaDB Error Codes

This page is licensed: CC BY-SA / Gnu FDL

RETURN

Syntax

```
RETURN expr
```

The `RETURN` statement terminates execution of a stored function and returns the value *`expr`* to the function caller. There must be at least one `RETURN` statement in a stored function. If the function has multiple exit points, all exit points must have a `RETURN`.

This statement is not used in stored procedures, triggers, or events. LEAVE can be used instead.

The following example shows that `RETURN` can return the result of a scalar subquery:

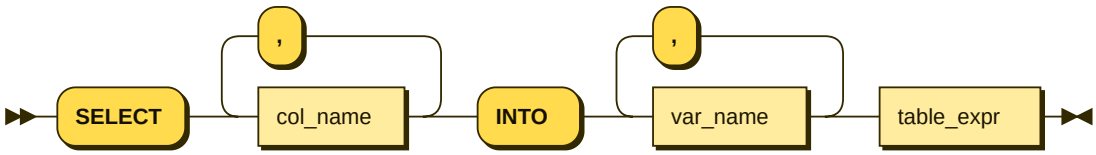
```
CREATE FUNCTION users_count() RETURNS BOOL
   READS SQL DATA
BEGIN
   RETURN (SELECT COUNT(DISTINCT User) FROM mysql.user);
END;
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SELECT INTO

Syntax

```
SELECT col_name [, col_name] ...
    INTO var_name [, var_name] ...
    table_expr
```



Railroad diagram of SELECT INTO — equivalent to the BNF above

Description

`SELECT ... INTO` enables selected columns to be stored directly into variables. No resultset is produced. The query should return a single row. If the query returns no rows, a warning with error code 1329 occurs (`No data`), and the variable values remain unchanged. If the query returns multiple rows, error 1172 occurs (Result consisted of more than one row). If it is possible that the statement may retrieve multiple rows, you can use `LIMIT 1` to limit the result set to a single row.

The `INTO` clause can also be specified at the end of the statement.

In the context of such statements that occur as part of events executed by the Event Scheduler, diagnostics messages (not only errors, but also warnings) are written to the error log, and, on Windows, to the application event log.

This statement can be used with both local variables and user-defined variables.

For the complete syntax, see SELECT.

Another way to set a variable's value is the SET statement.

`SELECT ... INTO` results are not stored in the query cache even if `SQL_CACHE` is specified.

Examples

```
SELECT id, data INTO @x,@y
FROM test.t1 LIMIT 1;
SELECT * FROM t1 WHERE t1.a=@x AND t1.b=@y
```

If you want to use this construct with `UNION` you have to use the syntax:

```
SELECT  * INTO @x FROM (SELECT t1.a FROM t1 UNION SELECT t2.a FROM t2) dt;
```

See Also

- SELECT - full SELECT syntax.
- SELECT INTO OUTFILE - formatting and writing the result to an external file.
- SELECT INTO DUMPFILE - binary-safe writing of the unformatted results to an external file.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SIGNAL

Syntax

```
SIGNAL error_condition
    [SET error_property
    [, error_property] ...]

error_condition:
    SQLSTATE [VALUE] 'sqlstate_value'
  | condition_name

error_property:
    error_property_name = <error_property_value>

error_property_name:
    CLASS_ORIGIN
  | SUBCLASS_ORIGIN
  | MESSAGE_TEXT
  | MYSQL_ERRNO
  | CONSTRAINT_CATALOG
  | CONSTRAINT_SCHEMA
  | CONSTRAINT_NAME
  | CATALOG_NAME
  | SCHEMA_NAME
  | TABLE_NAME
  | COLUMN_NAME
  | CURSOR_NAME
```

`SIGNAL` empties the diagnostics area and produces a custom error. This statement can be used anywhere but is generally useful when used inside a stored program. When the error is produced, it can be caught by a HANDLER. If not, the current stored program, or the current statement, will terminate with the specified error.

Sometimes an error HANDLER just needs to SIGNAL the same error it received, optionally with some changes. Usually, the RESIGNAL statement is the most convenient way to do this.

`error_condition` can be an SQLSTATE value, or a named error condition defined via DECLARE CONDITION. SQLSTATE must be a constant string consisting of five characters. These codes are standard to ODBC and ANSI SQL. For customized errors, the recommended SQLSTATE is '45000'. For a list of SQLSTATE values used by MariaDB, see the MariaDB Error Codes page. The SQLSTATE can be read via the API method `mysql_sqlstate()`.

To specify error properties user-defined variables and local variables can be used, as well as character set conversions (but you can't set a collation).

The error properties, their type and their default values are explained in the diagnostics area page.

Errors

If the `SQLSTATE` is not valid, the following error like this will be produced:

```
ERROR 1407 (42000): Bad SQLSTATE: '123456'
```

If a property is specified more than once, an error like this will be produced:

```
ERROR 1641 (42000): Duplicate condition information item 'MESSAGE_TEXT'
```

If you specify a condition name which is not declared, an error like this will be produced:

```
ERROR 1319 (42000): Undefined CONDITION: cond_name
```

If `MYSQL_ERRNO` is out of range, you will get an error like this:

```
ERROR 1231 (42000): Variable 'MYSQL_ERRNO' can't be set to the value of '0'
```

Examples

Here's what happens if SIGNAL is used in the client to generate errors:

```
SIGNAL SQLSTATE '01000';
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;

+---------+------+-----------------------------------------+
| Level   | Code | Message                                 |
+---------+------+-----------------------------------------+
| Warning | 1642 | Unhandled user-defined warning condition |
+---------+------+-----------------------------------------+
1 row in set (0.06 sec)

SIGNAL SQLSTATE '02000';
ERROR 1643 (02000): Unhandled user-defined not found condition
```

How to specify `MYSQL_ERRNO` and `MESSAGE_TEXT` properties:

```
SIGNAL SQLSTATE '45000' SET MYSQL_ERRNO=30001, MESSAGE_TEXT='H
ello, world!';

ERROR 30001 (45000): Hello, world!
```

The following code shows how to use user variables, local variables and character set conversion with `SIGNAL`:

```
CREATE PROCEDURE test_error(x INT)
BEGIN
   DECLARE errno SMALLINT UNSIGNED DEFAULT 31001;
   SET @errmsg = 'Hello, world!';
   IF x = 1 THEN
      SIGNAL SQLSTATE '45000' SET
      MYSQL_ERRNO = errno,
      MESSAGE_TEXT = @errmsg;
   ELSE
      SIGNAL SQLSTATE '45000' SET
      MYSQL_ERRNO = errno,
      MESSAGE_TEXT = _utf8'Hello, world!';
   END IF;
END;
```

How to use named error conditions:

```
CREATE PROCEDURE test_error(n INT)
BEGIN
   DECLARE `too_big` CONDITION FOR SQLSTATE '45000';
   IF n > 10 THEN
      SIGNAL `too_big`;
   END IF;
END;
```

In this example, we'll define a HANDLER for an error code. When the error occurs, we SIGNAL a more informative error which makes sense for our procedure:

```
CREATE PROCEDURE test_error()
BEGIN
   DECLARE EXIT HANDLER
   FOR 1146
   BEGIN
      SIGNAL SQLSTATE '45000' SET
      MESSAGE_TEXT = 'Temporary tables not found; did you call init() procedure?';
   END;
   -- this will produce a 1146 error
   SELECT `c` FROM `temptab`;
END;
```

See Also

- Diagnostics Area
- RESIGNAL
- DECLARE HANDLER
- Stored Routines
- MariaDB Error Codes

This page is licensed: CC BY-SA / Gnu FDL

WHILE

Syntax

```
[begin_label:] WHILE search_condition DO
    statement_list
END WHILE [end_label]
```

Description

The statement list within a `WHILE` statement is repeated as long as the `search_condition` is true. statement_list consists of one or more statements. If the loop must be executed at least once, REPEAT … LOOP can be used instead.

A `WHILE` statement can be labeled. `end_label` cannot be given unless `begin_label` also is present. If both are present, they must be the same.

Examples

```
CREATE PROCEDURE dowhile()
BEGIN
  DECLARE v1 INT DEFAULT 5;

  WHILE v1 > 0 DO
    ...
    SET v1 = v1 - 1;
  END WHILE;
END
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

Using Compound Statements Outside of Stored Programs

Compound statements can also be used outside of stored programs.

```
delimiter |
IF @have_innodb THEN
  CREATE TABLE IF NOT EXISTS innodb_index_stats (
    database_name    VARCHAR(64) NOT NULL,
    table_name       VARCHAR(64) NOT NULL,
    index_name       VARCHAR(64) NOT NULL,
    last_update      TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    stat_name        VARCHAR(64) NOT NULL,
    stat_value       BIGINT UNSIGNED NOT NULL,
    sample_size      BIGINT UNSIGNED,
    stat_description VARCHAR(1024) NOT NULL,
    PRIMARY KEY (database_name, table_name, index_name, stat_name)
  ) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8_bin STATS_PERSISTENT=0;
END IF|
Query OK, 0 rows affected, 2 warnings (0.00 sec)
```

Using compound statements this way is subject to following limitations:

- Only BEGIN, IF, CASE, LOOP, WHILE, REPEAT statements may start a compound statement outside of stored programs.
- BEGIN must use the `BEGIN NOT ATOMIC` syntax (otherwise it'll be confused with BEGIN that starts a transaction).
- A compound statement might not start with a label.
- A compound statement is parsed completely—note "2 warnings" in the above example, even if the condition was false (InnoDB was, indeed, disabled), and the CREATE TABLE statement was not executed, it was still parsed and the parser produced "Unknown storage engine" warning.

Inside a compound block first three limitations do not apply, one can use anything that can be used inside a stored program — including labels, condition handlers, variables, and so on:

```
BEGIN NOT ATOMIC
    DECLARE foo CONDITION FOR 1146;
    DECLARE x INT DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SET x=1;
    INSERT INTO test.t1 VALUES ("hndlr1", val, 2);
    END|
```

Example how to use `IF`:

```
IF (1>0) THEN BEGIN NOT ATOMIC SELECT 1; END ; END IF;;
```

Example of how to use `WHILE` loop:

```
DELIMITER |
BEGIN NOT ATOMIC
    DECLARE x INT DEFAULT 0;
    WHILE x <= 10 DO
        SET x = x + 1;
        SELECT x;
    END WHILE;
END|
DELIMITER ;
```

This page is licensed: CC BY-SA / Gnu FDL

Cursors

See Cursor Overview.

Introduces cursors in MariaDB stored programs, which are non-scrollable, read-only, and asensitive, and are declared, opened, fetched from, and closed to process result rows sequentially.

See CLOSE.

CLOSE closes a previously opened cursor, raising an error if it was not open; if not closed explicitly, a cursor closes at the end of its enclosing compound statement.

See DECLARE CURSOR.

DECLARE CURSOR declares a named cursor for a SELECT or, from MariaDB 12.3, a prepared statement name, optionally with parameters, for use within a stored program.

See FETCH.

FETCH retrieves the next row from an open cursor into local variables and advances the cursor, raising a No Data condition (SQLSTATE 02000) when no more rows remain.

See OPEN.

OPEN opens a previously declared cursor and executes its associated query, optionally binding variables or expressions, so that rows can be fetched.

This page is licensed: CC BY-SA / Gnu FDL

Cursor Overview

Description

A cursor is a structure that allows you to go over records sequentially, and perform processing based on the result.

MariaDB permits cursors inside stored programs, and MariaDB cursors are non-scrollable, read-only and case-insensitive.

- Non-scrollable means that the rows can only be fetched in the order specified by the `SELECT` statement. Rows cannot be skipped, you cannot jump to a specific row, and you cannot fetch rows in reverse order.
- Read-only means that data cannot be updated through the cursor.
- Assensitive means that the cursor points to the actual underlying data. This kind of cursor is quicker than the alternative, an insensitive cursor, as no data is copied to a temporary table. However, changes to the data being used by the cursor will affect the cursor data.

Cursors are created with a DECLARE CURSOR statement and opened with an OPEN statement. Rows are read with a FETCH statement before the cursor is finally closed with a CLOSE statement.

When FETCH is issued and there are no more rows to extract, the following error is produced:

```
ERROR 1329 (02000): No data - zero rows fetched, selected, or processed
```

To avoid problems, a DECLARE HANDLER statement is generally used. The `HANDLER` should handler the 1329 error, or the '02000' SQLSTATE, or the `NOT FOUND` error class.

> **CURRENT**
>
> Cursors can be declared for a **prepared statement** name, enabling full support for **Dynamic SQL** within stored routines. The SQL Standard syntax for `<dynamic declare cursor>` is supported: a cursor is declared with the `FOR` clause to bind it to a prepared statement name, which is then prepared and opened normally. Cursors may also still SELECT from views, which can be created dynamically via prepared statements.
>
> **< 12.3**

Only **SELECT** statements are allowed for cursors, and they cannot be contained in a variable - so, they cannot be composed dynamically. However, it is possible to **SELECT** from a view. Since the **CREATE VIEW** statement can be executed as a prepared statement, it is possible to dynamically create the view that is queried by the cursor.

CURRENT

Cursors can have parameters. Cursor parameters can appear in any part of the DECLARE CURSOR `select_statement` where a stored procedure variable is allowed (select list, `WHERE`, `HAVING`, `LIMIT` etc). See DECLARE CURSOR and OPEN for syntax, and below for an example.

< 10.3

Cursors cannot have parameters.

Examples

```
CREATE TABLE c1(i INT);

CREATE TABLE c2(i INT);

CREATE TABLE c3(i INT);

DELIMITER //

CREATE PROCEDURE p1()
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE x, y INT;
  DECLARE cur1 CURSOR FOR SELECT i FROM test.c1;
  DECLARE cur2 CURSOR FOR SELECT i FROM test.c2;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  OPEN cur1;
  OPEN cur2;

  read_loop: LOOP
    FETCH cur1 INTO x;
    FETCH cur2 INTO y;
    IF done THEN
      LEAVE read_loop;
    END IF;
    IF x < y THEN
      INSERT INTO test.c3 VALUES (x);
    ELSE
      INSERT INTO test.c3 VALUES (y);
    END IF;
  END LOOP;

  CLOSE cur1;
  CLOSE cur2;
END; //

DELIMITER ;

INSERT INTO c1 VALUES(5),(50),(500);

INSERT INTO c2 VALUES(10),(20),(30);

CALL p1;

SELECT * FROM c3;
```

```
+------+
| i    |
+------+
|    5 |
|   20 |
|   30 |
+------+
```

```
DROP PROCEDURE IF EXISTS p1;
DROP TABLE IF EXISTS t1;
CREATE TABLE t1 (a INT, b VARCHAR(10));

INSERT INTO t1 VALUES (1,'old'),(2,'old'),(3,'old'),(4,'old'),(5,'old');

DELIMITER //

CREATE PROCEDURE p1(min INT,max INT)
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE va INT;
  DECLARE cur CURSOR(pmin INT, pmax INT) FOR SELECT a FROM t1 WHERE a BETWEEN pmin AND pmax;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=TRUE;
  OPEN cur(min,max);
  read_loop: LOOP
    FETCH cur INTO va;
    IF done THEN
      LEAVE read_loop;
    END IF;
    INSERT INTO t1 VALUES (va,'new');
  END LOOP;
  CLOSE cur;
END;
//

DELIMITER ;

CALL p1(2,4);

SELECT * FROM t1;
```

```
+------+------+
| a    | b    |
+------+------+
|    1 | old  |
|    2 | old  |
|    3 | old  |
|    4 | old  |
|    5 | old  |
|    2 | new  |
|    3 | new  |
|    4 | new  |
+------+------+
```

See Also

- DECLARE CURSOR
- OPEN cursor_name
- FETCH cursor_name
- CLOSE cursor_name
- Cursors in Oracle mode

This page is licensed: CC BY-SA / Gnu FDL

CLOSE

Syntax

```
CLOSE cursor_name
```

Description

This statement closes a previously opened cursor. The cursor must have been previously opened or else an error occurs.

If not closed explicitly, a cursor is closed at the end of the compound statement in which it was declared.

See Cursor Overview for an example.

See Also

- Cursor Overview
- DECLARE CURSOR
- OPEN cursor_name
- FETCH cursor_name
- Cursors in Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

DECLARE CURSOR

Syntax

CURRENT

```
DECLARE cursor_name CURSOR [(cursor_formal_parameter[,...])] FOR {select_statement |
prepared_statement_name}

cursor_formal_parameter:
    [IN] name type [collate clause]
```

< 10.8

```
DECLARE cursor_name CURSOR [(cursor_formal_parameter[,...])] FOR select_statement

cursor_formal_parameter:
    name type [collate clause]
```

Description

This statement declares a cursor. Multiple cursors may be declared in a stored program, but each cursor in a given block must have a unique name.

`select_statement` is not executed until the OPEN statement is executed. It is important to remember this if the query produces an error, or calls functions which have side effects.

A `SELECT` associated to a cursor can use variables, but the query itself cannot be a variable, and cannot be dynamically composed. The `SELECT` statement cannot have an `INTO` clause.

Note:

Starting with MariaDB 12.3, the query can also be a prepared statement name, which allows the query to be dynamically composed. The `SELECT` statement cannot have an `INTO` clause.

Cursors must be declared before HANDLERs, but after local variables and CONDITIONs.

> The `DECLARE CURSOR` statement provides a cursor to a specified `SELECT` statement.
>
> To declare a cursor variable type that can be connected with multiple queries at runtime (REF CURSOR), use `DECLARE TYPE... IS REF CURSOR` in Oracle mode. See DECLARE TYPE.

Dynamic Cursors

Starting with MariaDB 12.3, a cursor can be declared for a prepared statement. This allows the use of Dynamic SQL within stored routines. The cursor is bound to a prepared statement name, which must be defined using the `PREPARE` statement before the cursor is opened.

```
CREATE OR REPLACE PROCEDURE p1(tab VARCHAR(64), min_id INTEGER)
BEGIN
  DECLARE v_id INT;
  DECLARE v_c1 VARCHAR(100);
  DECLARE no_data BOOL DEFAULT FALSE;

  -- 1. Declare cursor for the statement name 's1'
  DECLARE c1 CURSOR FOR s1;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET no_data = TRUE;

  -- 2. Prepare the statement dynamically
  PREPARE s1 FROM CONCAT('SELECT id, c1 FROM ', tab, ' WHERE id >= ?');

  -- 3. Open cursor and bind parameters
  OPEN c1 USING min_id;

  fetch_loop: LOOP
    FETCH c1 INTO v_id, v_c1;
    IF no_data THEN
      LEAVE fetch_loop;
    END IF;
    SELECT v_id, v_c1;
  END LOOP;

  CLOSE c1;
  DEALLOCATE PREPARE s1;
END;
$$

DELIMITER ;
```

Parameters

Cursors can have parameters. This is a non-standard SQL extension. Cursor parameters can appear in any part of the `DECLARE CURSOR select_statement` where a stored procedure variable is allowed (select list, `WHERE`, `HAVING`, `LIMIT`, and so forth).

IN

> **CURRENT**
>
> The `IN` qualifier is supported in the `cursor_formal_parameter` part of the syntax.
>
> **< 10.8**

> The `IN` qualifier is **not** supported in the `cursor_formal_parameter` part of the syntax.

See Cursor Overview for an example.

See Also

- Cursor Overview
- OPEN cursor_name
- FETCH cursor_name
- CLOSE cursor_name
- Cursors in Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

FETCH

Syntax

```
FETCH cursor_name INTO var_name [, var_name] ...
```

Description

This statement fetches the next row (if a row exists) using the specified open cursor, and advances the cursor pointer.

`var_name` can be a local variable, but *not* a user-defined variable.

If no more rows are available, a No Data condition occurs with `SQLSTATE` value `02000`. To detect this condition, you can set up a handler for it (or for a `NOT FOUND` condition).

See Cursor Overview for an example.

See Also

- Cursor Overview
- DECLARE CURSOR
- OPEN cursor_name
- CLOSE cursor_name
- Cursors in Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

OPEN

Syntax

CURRENT

```
OPEN cursor_name [[USING variable[,...]] | [expression[,...]]];
```

ORACLE MODE

`-- From MariaDB 12.3`

```
OPEN cursor_variable FOR dynamic_sql_string;
```

< 10.3

```
OPEN cursor_name
```

Description

This statement opens a cursor which was previously declared with DECLARE CURSOR.

The query associated to the `DECLARE CURSOR` is executed when `OPEN` is executed. It is important to remember this if the query produces an error, or calls functions which have side effects.

This is necessary in order to FETCH rows from a cursor.

See Cursor Overview for an example.

See Also

- Cursor Overview
- DECLARE CURSOR
- FETCH cursor_name
- CLOSE cursor_name
- Cursors in Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

Diagnostics

See Diagnostics Area.

Describes the diagnostics area that holds statement and condition information about errors, warnings, and notes produced by an SQL statement, including its properties and how it is populated and cleared.

See GET DIAGNOSTICS.

GET DIAGNOSTICS copies statement and condition information from the diagnostics area into user or local variables, reading properties such as NUMBER, RETURNED_SQLSTATE, MYSQL_ERRNO, and MESSAGE_TEXT.

See SQLSTATE.

Explains SQLSTATE, the five-character code identifying SQL error conditions by class and subclass, including the standard success, warning, and not-found classes and how they map to handler keywords.

This page is licensed: CC BY-SA / Gnu FDL

Diagnostics Area

The diagnostics area contains information about the error conditions which were produced by an SQL statement, as well as some information about the statement which generated them.

Statement Information

The statement information area contains the following data:

- `NUMBER` is the number of conditions which are present in the diagnostics area.
- `ROW_COUNT` has the same value as the ROW_COUNT() function for the statement that produced the conditions.

Condition Information

Each condition has several properties, which are explained here.

Data Types and Sizes

The following table shows the type and size of all the properties:

Property name	Property type
RETURNED_SQLSTATE	VARCHAR(5)
MYSQL_ERRNO	SMALLINT UNSIGNED
MESSAGE_TEXT	VARCHAR(512)
CLASS_ORIGIN	VARCHAR(64)
SUBCLASS_ORIGIN	VARCHAR(64)
CONSTRAINT_CATALOG	VARCHAR(64)
CONSTRAINT_SCHEMA	VARCHAR(64)
CONSTRAINT_NAME	VARCHAR(64)
CATALOG_NAME	VARCHAR(64)
SCHEMA_NAME	VARCHAR(64)
TABLE_NAME	VARCHAR(64)
COLUMN_NAME	VARCHAR(64)
CURSOR_NAME	VARCHAR(64)

These properties can never be set to `NULL`. If they are empty, the empty string is used.

Common Condition Properties

The most common ones have a value for all built-in errors, and can be read both via SQL and via the API:

`RETURNED_SQLSTATE` is the `SQLSTATE` of the condition. It is a five characters code, composed by a class (first two characters) and a subclass (last three characters). For more information about this property, refer to the SQLSTATE page.

`MYSQL_ERRNO` is the error code. Each built-in condition has a unique numeric code. 0 indicates success, but it cannot be explicitly set or read via SQL. For a list of built-in error codes, refer to MariaDB Error Codes. The API function to read it is mysql_errno().

`MESSAGE_TEXT` is a descriptive, human-readable message. For built-in errors, parsing this string is the only way to get more information about the error. For example, parsing a message like “Table ‘tab1’ already exists”, a program can find out that the missing table is tab1. The API function to read it is mysql_error().

For conditions generated by the user via `SIGNAL`, if `MYSQL_ERRNO` and `MESSAGE_TEXT` are not specified, their default values depend on the first two `SQLSTATE` characters:

- ‘00’ means ‘success’. It can not be set in any way, and can only be read via the API.
- For ‘01’ class, default `MYSQL_ERRNO` is 1642 and default `MESSAGE_TEXT` is ‘Unhandled user-defined warning condition’.
- For ‘02’ class, default `MYSQL_ERRNO` is 1643 and default `MESSAGE_TEXT` is ‘Unhandled user-defined not found condition’.
- For all other cases, including the ‘45000’ value, default `MYSQL_ERRNO` is 1644 and default `MESSAGE_TEXT` is ‘Unhandled user-defined exception condition’.

Special Condition Properties

There are more condition properties, which are never set for built-in errors. They can only be set via `SIGNAL` and `RESIGNAL` statements, and can only be read via `GET DIAGNOSTICS` - not via the API. Such properties are:

`CLASS_ORIGIN` indicates whether the `SQLSTATE` uses a standard class or a software-specific class. If it is defined in the SQL standards document ISO 9075-2 (section 24.1, SQLSTATE), this property's value is supposed to be 'ISO 9075', otherwise it is supposed to be 'MySQL'. However, any string is accepted.

`SUBCLASS_ORIGIN` indicates whether the `SQLSTATE` uses a standard subclass or a software-specific class. If the `SQLSTATE` is defined in the SQL standards document ISO 9075-2 (section 24.1, `SQLSTATE`), this property's value is supposed to be 'ISO 9075', otherwise it is supposed to be 'MySQL'. However, any string is accepted.

`SCHEMA_NAME` indicates in which schema (database) the error occurred.

`TABLE_NAME` indicates the name of the table which was accessed by the failed statement.

`COLUMN_NAME` indicates the name of the column which was accessed by the failed statement.

`CONSTRAINT_NAME` indicates the name of the constraint that was violated.

`CONSTRAINT_SCHEMA` indicates in which schema the violated constraint is located.

`CURSOR_NAME` indicates the name of the cursor which caused the error.

The following properties can be used and are defined in the standard SQL, but have no meaning because MariaDB doesn't currently support catalogs:

`CATALOG_NAME` is used by the standard SQL to indicate in which catalog the error occurred.

`CONSTRAINT_CATALOG` is used by the standard SQL to indicate in which catalog the violated constraint is located.

How the Diagnostics Area is Populated and Cleared

When a statement produces one or more error conditions (errors, warnings, notes) the conditions are inserted into the diagnostics area, and the statement information area is updated with that statement's information. Usually, this also clears all the old conditions from the diagnostics area, but there is an exception: if the new statement is a correctly parsed `RESIGNAL` or `GET DIAGNOSTICS`, the old contents will remain in the diagnostics area. `SIGNAL` clears the old conditions.

When a table-based statement (like `INSERT`) is executed, the old data in the diagnostics area is cleared even if the statement doesn't produce any condition. However, statements which don't access any table (like `SET`, or a `SELECT` with no `FROM` clause) is executed and produces no warnings, the diagnostics area remains unchanged.

The maximum number of conditions that can be in the diagnostics area is max_error_count. If this value is 0, the diagnostics area is empty. If this variable is changed, the new value takes effect with the next statement (that is, the diagnostics area is not immediately truncated).

How to Access the Diagnostics Area

The following statements explicitly add conditions to the diagnostics area:

- SIGNAL: produces a custom error.
- RESIGNAL: after an error is produced, generates a modified version of that error.

The following statements read contents from the diagnostics area:

- GET DIAGNOSTICS is the only way to read all information.
- SHOW WARNINGS shows a summary of errors, warnings and notes.
- SHOW ERRORS shows a summary of errors.

DECLARE HANDLER can be used to handle error conditions within stored programs.

DECLARE CONDITION can be used to associate an `SQLSTATE` or an error code to a name. That name can be referenced in `DECLARE HANDLER`, `SIGNAL` and `RESIGNAL` statements.

All these statements can also be executed inside a stored routine. However, only `SHOW WARNINGS` and `SHOW ERRORS` can be executed as a prepared statement. After an EXECUTE statement, the diagnostics area contains information about the prepared statement, if it produces error conditions.

See Also

- RESIGNAL
- SIGNAL
- HANDLER
- GET DIAGNOSTICS
- SHOW WARNINGS
- SHOW ERRORS
- DECLARE HANDLER
- MariaDB Error Codes

This page is licensed: CC BY-SA / Gnu FDL

GET DIAGNOSTICS

```
GET [CURRENT] DIAGNOSTICS
{
    statement_property
    [, statement_property] ...
  | CONDITION condition_number
    condition_property
    [, condition_property] ...
}

statement_property:
    variable = statement_property_name

condition_property:
    variable  = condition_property_name

statement_property_name:
    NUMBER
  | ROW_COUNT

condition_property_name:
    CLASS_ORIGIN
  | SUBCLASS_ORIGIN
  | RETURNED_SQLSTATE
  | MESSAGE_TEXT
  | MYSQL_ERRNO
  | CONSTRAINT_CATALOG
  | CONSTRAINT_SCHEMA
  | CONSTRAINT_NAME
  | CATALOG_NAME
  | SCHEMA_NAME
  | TABLE_NAME
  | COLUMN_NAME
  | CURSOR_NAME
  | ROW_NUMBER
```

The diagnostics area contains information about the errors, warnings and notes which were produced by the last SQL statement. If that statement didn't produce any warnings, the diagnostics area contains information about the

last executed statement which involved a table. The `GET DIAGNOSTICS` statement copies the requested information from the diagnostics area to the specified variables. It is possible to use both user variables or local variables.

To use `GET DIAGNOSTICS`, it is important to know how the diagnostics area is structured. It has two sub-areas: the statement information area and the error conditions information area. For details, please refer to the diagnostics area page.

Each single `GET DIAGNOSTICS` command can read information from the statement information area or from a single error condition. This means that, if you have two warnings and you want to know the number of warnings, and read both the warnings, you need to issue `GET DIAGNOSTICS` three times.

The `CURRENT` keywords adds nothing to the statement, because MariaDB has only one diagnostics area.

If `GET DIAGNOSTICS` produces an error condition (because the command is properly parsed but not correctly used), the diagnostics area is not emptied, and the new condition is added.

Getting Information from a Condition

To read information from a condition, the `CONDITION` keyword must be specified and it must be followed by the condition number. This number can be specified as a constant value or as a variable. The first condition's index is 1. If the error condition does not exist, the variables will not change their value and a 1758 error will be produced ("Invalid condition number").

The condition properties that can be read with `GET DIAGNOSTICS` are the same that can be set with `SIGNAL` and `RESIGNAL` statements. They are explained in the diagnostics area page. However, there is one more property: `RETURNED_SQLSTATE`, which indicates the condition's SQLSTATE.

For a list of `SQLSTATE` values and MariaDB error codes, see MariaDB Error Codes.

The type for all the condition properties is `VARCHAR`(64), except for `MYSQL_ERRNO`, whose valid range is 1 to 65534.

ROW_NUMBER

> **CURRENT**
>
> You can use the `ROW_NUMBER` property to retrieve the row number, too, even if the error text does not mention it. This property is named `ERROR_INDEX`. `ROW_NUMBER` is a reserved word.
>
> **< 10.7**
>
> There is no way, short of parsing the error text, to know in what row an error had happened.

Examples

In the following example, a statement generates two warnings, and `GET DIAGNOSTICS` is used to get the number of warnings:

```
CREATE TABLE `test`.`t` (`c` INT) ENGINE = x;
Query OK, 0 rows affected, 2 warnings (0.19 sec)

GET DIAGNOSTICS @num_conditions = NUMBER;

SELECT @num_conditions;
+-----------------+
| @num_conditions |
+-----------------+
|               2 |
+-----------------+
```

Then we can see the warnings:

```
GET DIAGNOSTICS CONDITION 1 @sqlstate = RETURNED_SQLSTATE,
  @errno = MYSQL_ERRNO, @text = MESSAGE_TEXT;

SELECT @sqlstate, @errno, @text;
+-----------+--------+---------------------------+
| @sqlstate | @errno | @text                     |
+-----------+--------+---------------------------+
| 42000     |   1286 | Unknown storage engine 'x' |
+-----------+--------+---------------------------+

GET DIAGNOSTICS CONDITION 2 @sqlstate = RETURNED_SQLSTATE,
  @errno = MYSQL_ERRNO, @text = MESSAGE_TEXT;

SELECT @sqlstate, @errno, @text;
+-----------+--------+-----------------------------------------+
| @sqlstate | @errno | @text                                   |
+-----------+--------+-----------------------------------------+
| HY000     |   1266 | Using storage engine InnoDB for table 't' |
+-----------+--------+-----------------------------------------+
```

```
INSERT INTO t1 VALUES (4,'d',1.00104),(1,'a',1.00101),(2,'b',1.00102);
ERROR 23000: Duplicate entry '1' for key 'PRIMARY'

GET DIAGNOSTICS CONDITION 1 @row_num= ROW_NUMBER; -- from MariaDB 10.7

SELECT @row_num;
+----------+
| @row_num |
+----------+
|        2 |
+----------+
```

See Also

- Diagnostics Area

This page is licensed: CC BY-SA / Gnu FDL

SQLSTATE

`SQLSTATE` is a code which identifies SQL error conditions. It composed by five characters, which can be numbers or uppercase ASCII letters. An `SQLSTATE` value consists of a class (first two characters) and a subclass (last three characters).

There are three important standard classes. They all indicate in which logical group of errors the condition falls. They match to a particular keyword which can be used with DECLARE HANDLER. Also, the `SQLSTATE` class determines the default value for the `MYSQL_ERRNO` and `MESSAGE_TEXT` condition properties.

- '00' means 'success'. It can not be set in any way and can only be read via the API.
- '01' contains all warnings and matches to the `SQLWARNING` keyword. The default `MYSQL_ERRNO` is 1642 and default `MESSAGE_TEXT` is 'Unhandled user-defined warning condition'.
- '02' is the `NOT FOUND` class. The default `MYSQL_ERRNO` is 1643 and default `MESSAGE_TEXT` is 'Unhandled user-defined not found condition'.
- All other classes match the `SQLEXCEPTION` keyword. The default `MYSQL_ERRNO` is 1644 and default `MESSAGE_TEXT` is 'Unhandled user-defined exception condition'.

The subclass, if it is set, indicates a particular condition, or a particular group of conditions within the class. However, the '000' sequence means 'no subclass'.

For example, if you try to SELECT from a table which does not exist, a 1109 error is produced, with a '42S02' `SQLSTATE`. '42' is the class and 'S02' is the subclass. This value matches to the `SQLEXCEPTION` keyword. When

`FETCH` is called for a cursor which has already reached the end, a 1329 error is produced, with a '02000' SQLSTATE. The class is '02' and there is no subclass (because '000' means 'no subclass'). It can be handled by a `NOT FOUND` handlers.

The standard SQL specification says that classes beginning with 0, 1, 2, 3, 4, A, B, C, D, E, F and G are reserved for standard-defined classes, while other classes are vendor-specific. It also says that, when the class is standard-defined, subclasses starting with those characters (except for '000') are standard-defined subclasses, while other subclasses are vendor-defined. However, MariaDB and MySQL do not strictly obey this rule.

To read the `SQLSTATE` of a particular condition which is in the diagnostics area, the GET DIAGNOSTICS statement can be used: the property is called `RETURNED_SQLSTATE`. For user-defined conditions (SIGNAL and RESIGNAL statements), a SQLSTATE value must be set via the `SQLSTATE` clause. However, SHOW WARNINGS and SHOW ERRORS do not display the `SQLSTATE`.

For user-defined conditions, MariaDB and MySQL recommend the '45000' `SQLSTATE` class.

'HY000' is called the "general error": it is the class used for built-in conditions which do not have a specific `SQLSTATE` class.

A partial list of error codes and matching `SQLSTATE` values can be found under MariaDB Error Codes.

This page is licensed: CC BY-SA / Gnu FDL

Stored Routine Statements

See CALL.

Invoke a stored procedure. This statement executes a previously created stored procedure, optionally passing parameters and returning results.

See DO.

Execute expressions without returning a result set. This statement runs functions or expressions, often used for side effects like releasing locks.

See CREATE PROCEDURE.

Define a new stored procedure, specifying its name, parameters (`IN` , `OUT` , `INOUT`), and the SQL statements it executes.

See ALTER PROCEDURE.

Modify the characteristics of an existing stored procedure, such as its security context or comment, without changing its logic.

See DROP PROCEDURE.

Remove a stored procedure and its associated privileges from the database.

See DROP FUNCTION.

Remove a stored function from the database, deleting its definition and associated privileges.

This page is licensed: CC BY-SA / Gnu FDL

CALL

Syntax

```
CALL sp_name([parameter[,...]])
CALL sp_name[()]
```

Description

The `CALL` statement invokes a stored procedure that was defined previously with CREATE PROCEDURE.

Stored procedure names can be specified as `database_name.procedure_name`. Procedure names and database names can be quoted with backticks (). This is necessary if they are reserved words, or contain special characters. See identifier qualifiers for details.

`CALL p()` and `CALL p` are equivalent.

If parentheses are used, any number of spaces, tab characters and newline characters are allowed between the procedure's name and the open parenthesis.

`CALL` can pass back values to its caller using parameters that are declared as `OUT` or `INOUT` parameters. If no value is assigned to an `OUT` parameter, `NULL` is assigned (and its former value is lost). To pass such values from another stored program you can use user-defined variables, local variables or routine's parameters; in other contexts, you can only use user-defined variables.

`CALL` can also be executed as a prepared statement. Placeholders can be used for `IN` parameters in all versions of MariaDB; for `OUT` and `INOUT` parameters, placeholders can be used since MariaDB 5.5.

When the procedure returns, a client program can also obtain the number of rows affected for the final statement executed within the routine: At the SQL level, call the ROW_COUNT() function; from the C API, call the `mysql_affected_rows()` function.

If the `CLIENT_MULTI_RESULTS` API flag is set, `CALL` can return any number of result sets and the called stored procedure can execute prepared statements. If it is not set, at most one result set can be returned and prepared statements cannot be used within procedures.

This page is licensed: GPLv2, originally from fill_help_tables.sql

DO

Syntax

```
DO expr [, expr] ...
```

Description

`DO` executes the expressions but does not return any results. In most respects, `DO` is shorthand for `SELECT expr, ...,` but has the advantage that it is slightly faster when you do not care about the result.

`DO` is useful primarily with functions that have side effects, such as RELEASE_LOCK().

This page is licensed: GPLv2, originally from fill_help_tables.sql

Table Statements

See ALTER.

Access the reference for `ALTER` statements. This section lists commands to modify existing database objects, including tables, databases, users, and servers.

See ALTER TABLE.

Complete `ALTER TABLE` guide for MariaDB. Complete syntax for modifying columns, indexes, constraints, and table properties with comprehensive examples.

See ANALYZE TABLE.

Analyze and store key distribution. This statement updates index statistics used by the optimizer to choose the best execution plan.

See CHECK TABLE.

Check tables or views for errors. This statement verifies the integrity of table structure and data for supported storage engines.

See CHECK VIEW.

Verify the validity of a view's algorithm. This statement checks if the view definition is correct and references existing tables.

See CHECKSUM TABLE.

Report a checksum for table contents. This statement calculates a value to compare tables, useful for verifying replication consistency.

See CREATE TABLE.

Complete guide to creating tables in MariaDB. Complete `CREATE TABLE` syntax for data types, constraints, indexes, and storage engines for production use.

See DELETE.

Complete guide to deleting data in MariaDB. Complete `DELETE` syntax with `WHERE` filtering, `JOIN` operations, and safety considerations for production use.

See DROP TABLE.

Complete `DROP TABLE` syntax: `TEMPORARY`, `IF EXISTS`, `WAIT` / `NOWAIT`, `RESTRICT` / `CASCADE` options, metadata locks, atomic `DROP`, and replication behavior.

See INSERT.

Complete guide to inserting data in MariaDB. Complete `INSERT` syntax for single rows, bulk operations, and `ON DUPLICATE KEY` handling for production use.

See OPTIMIZE TABLE.

Complete MariaDB performance optimization guide. Complete reference for query tuning, indexing strategies, and configuration improvements for production use.

See RENAME TABLE.

Change the name of one or more tables atomically. This command moves tables within or between databases while preserving their data and structure.

See REPAIR TABLE.

Repair corrupted tables. This statement fixes errors in tables for supported storage engines like MyISAM, Aria, and Archive.

See REPAIR VIEW.

Check and correct a view's algorithm. This statement is primarily used by upgrade scripts to ensure view definitions are compatible.

See REPLACE.

Insert or replace rows based on unique keys. This statement acts like `INSERT`, but if a duplicate key exists, it deletes the old row and inserts the new one.

See SHOW COLUMNS.

Complete guide to displaying table columns in MariaDB. Complete `SHOW COLUMNS` syntax with field types, keys, and filtering options for production use.

See SHOW CREATE TABLE.

Get the SQL statement to recreate a table. This statement shows the complete `CREATE TABLE` syntax, including column definitions and indexes.

See SHOW INDEX.

Complete `SHOW INDEX` reference: `SHOW INDEX FROM`` *tbl_name*` syntax, output fields, and `WHERE` / `LIKE` filters.

See TRUNCATE TABLE.

Complete `TRUNCATE TABLE` reference with `[WAIT n|NOWAIT]` syntax, InnoDB `FOREIGN KEY` constraints, implicit commit, and `AUTO_INCREMENT` reset.

See UPDATE.

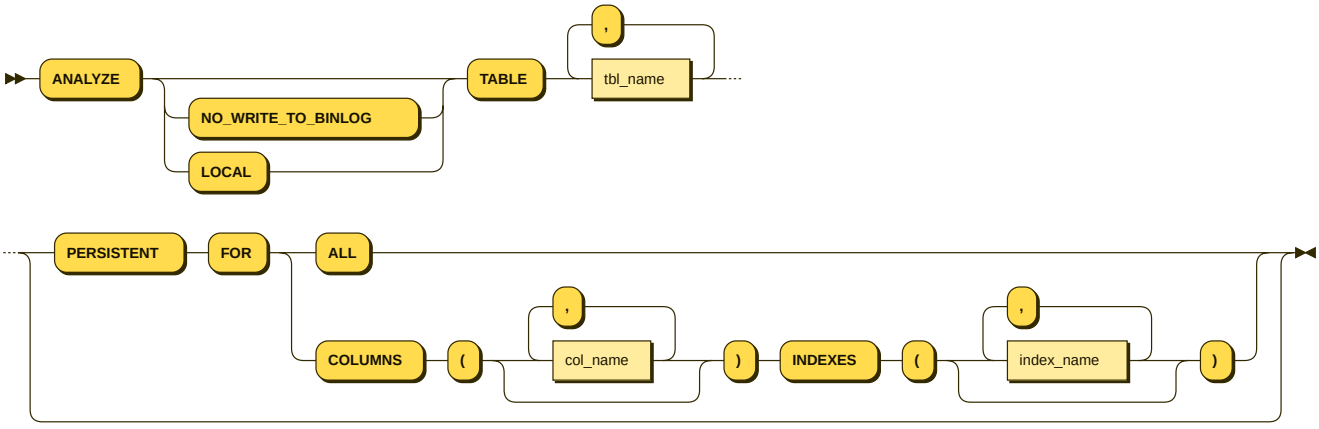
Complete `UPDATE` statement guide for MariaDB. Complete syntax reference with `WHERE` conditions, `JOIN` operations, and multi-table updates for production use.

This page is licensed: CC BY-SA / Gnu FDL

ANALYZE TABLE

Syntax

```
ANALYZE [NO_WRITE_TO_BINLOG | LOCAL] TABLE tbl_name [,tbl_name ...]
  [PERSISTENT FOR
    { ALL
      | COLUMNS ([col_name [,col_name ...]]) INDEXES ([index_name [,index_name ...]])
    }
  ]
```



Railroad diagram of ANALYZE TABLE — equivalent to the BNF above

Description

`ANALYZE TABLE` analyzes and stores the key distribution for a table (index statistics). This statement works with MyISAM, Aria, and InnoDB tables. During the analysis, InnoDB will allow reads/writes, and MyISAM/Aria reads/inserts. For MyISAM tables, this statement is equivalent to using myisamchk –analyze.

`ANALYZE` uses histograms, which can provide a better selectivity than InnoDB statistics offer. InnoDB statistics work with a limited sample set and are therefore not as accurate as persistent statistics can be. For more information on how the analysis works within InnoDB, see InnoDB Limitations.

MariaDB uses the stored key distribution to decide the order in which tables should be joined when you perform a join on something other than a constant. In addition, key distributions can be used when deciding which indexes to use for a specific table within a query.

This statement requires SELECT and INSERT privileges for the table.

By default, `ANALYZE TABLE` statements are written to the binary log and will be replicated. The `NO_WRITE_TO_BINLOG` keyword (`LOCAL` is an alias) will ensure the statement is not written to the binary log.

`ANALYZE TABLE` statements are not logged to the binary log if read_only is set. See also Read-Only Replicas.

> **CURRENT**
>
> `ANALYZE TABLE` is non-blocking and non-intrusive. A connection will start using new statistics for the query following the completion of the `ANALYZE TABLE`.
>
> **< 10.6.16**
>
> `ANALYZE TABLE` is blocking and intrusive.

`ANALYZE TABLE` is also supported for partitioned tables. You can use `ALTER TABLE ... ANALYZE PARTITION` to analyze one or more partitions.

The Aria storage engine supports progress reporting for the `ANALYZE TABLE` statement.

> **CURRENT**
>
> **Skipping Long CHAR/VARCHAR Columns**
>
> When using `ANALYZE TABLE PERSISTENT`, MariaDB skips long `CHAR` / `VARCHAR` columns during statistics collection if they exceed the value of the `analyze_max_length` system variable.
>
> This prevents excessive disk usage when analyzing tables with large text columns.
>
> - If a column is longer than `analyze_max_length`, it is excluded from stats.
> - If a long column is **explicitly specified** in `FOR COLUMNS()`, it is **still analyzed**, regardless of its size.
>
> Example:

```
SET GLOBAL analyze_max_length = 50000;
ANALYZE TABLE large_text_table PERSISTENT;
```

```
ANALYZE TABLE large_text_table PERSISTENT FOR COLUMNS(long_description);
```

```
CREATE TABLE product_data (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    description VARCHAR(50000), -- long column
    specs VARCHAR(1000)
);

-- Set limit
SET SESSION analyze_max_length = 10000;

-- Run analysis without explicitly selecting columns
ANALYZE TABLE product_data PERSISTENT;

-- 'description' will be skipped due to length > 10000

-- To include it anyway
ANALYZE TABLE product_data PERSISTENT FOR COLUMNS(description);
```

> **< 10.6.23 / 10.11.14 / 11.4.8 / 11.8.3 / 12.0.2 / 12.1.1 / 11.8 ENTERPRISE SERVER**
>
> The functionality for skipping long `CHAR` and `VARCHAR` columns isn't available.

Performance Impact

Note that analyzing tables with `ANALYZE` can have a performance impact and can use a lot of disk space for big tables. As column statistics usually do not change much over time, even when the table grows, there is no benefit to running `ANALYZE` very often.

Running `ANALYZE` is indicated:

- for newly populated tables,
- for tables that have additional columns added that are used in WHERE clauses,
- when a table has doubled in size,
- when you note that a query becomes slow because the table order has changed, and you can see from EXPLAIN or ANALYZE FORMAT=JSON that the selectivity is wrong for a table,
- when the distribution of data in a table has changed significantly, for example, if you have loaded a batch of data or deleted a large amount of data.

> **Warning:**
>
> `ANALYZE` isn't useful for table columns of type `UNIQUE`, `PRIMARY KEY`, `TIME`, or `CURRENT_TIME`. In `ANALYZE` queries, you should omit columns of those types.

EITS Statistics / PERSISTENT FOR

Overview

`ANALYZE TABLE` supports engine-independent table statistics (EITS). See Engine-Independent Table Statistics: Collecting Statistics with the ANALYZE TABLE Statement for more information.

You can run the statement on all columns with this statement — however, be aware that this can take a long time for very large (500+ GB) tables:

```
ANALYZE TABLE tbl PERSISTENT FOR ALL
```

Focusing on particular columns, the statement looks like this — it collects statistics only for the specified table columns:

```
ANALYZE TABLE tbl PERSISTENT FOR COLUMNS (column1, column2, ...) INDEXES (index1, ...)
```

Focusing on columns helps, among other things, avoid including `BLOB` columns, for which MariaDB doesn't collect statistics.

> **Note:**
>
> It's recommended to collect statistics for columns that do *not* have an index starting from that column (if the index starts with a column, the index itself is used as a source for statistics), and which are used in `WHERE` conditions of your queries.

Queries That Benefit

Queries that benefit most are those where the query plan depends on the optimizer knowing condition selectivities, the most important ones being `JOIN` queries and those with an `ORDER BY ... LIMIT` clause. To benefit, the query must have a condition that may or may not be selective, for example:

```
SELECT *
FROM orders, customer
WHERE
       orders.o_custkey = customer.c_custkey  -- join condition
  AND  customer.c_area = 'EMEA'  -- (1)
  AND  orders.o_priority= 'URGENT' -- (2)
```

Here, the optimizer benefits from knowing these statistics:

1. Which fraction of customers are in the EMEA region?
2. Which fraction of orders are URGENT?

For that situation, you can issue this statement:

```
ANALYZE TABLE orders, customer PERSISTENT FOR ALL
```

EITS vs. InnoDB-Internal Statistics

EITS (engine-independent table statistics) provide way more data than InnoDB-internal statistics. The downside is that EITS are never automatically updated, and it takes time to collect them.

InnoDB statistics, on the other hand, provide less data, but they are automatically updated.

Controlling Statistics

Engine-independent statistics can be controlled (enabled and disabled) using the use_stat_tables variable and the optimizer_use_condition_selectivity variable. InnoDB-persistent statistics are controlled with the innodb_stats_persistent variable (allowing you to turn InnoDB statistics on or off). Combining both kinds of statistics is possible.

The server relies on InnoDB statistics by default. That way, it can use some statistics even if `ANALYZE TABLE` is never run (or not often enough). This gives good enough results for the majority of queries. Some queries, however, need more statistical data so the optimizer can create a good plan. Slow queries indicate there aren't enough statistical data. Those queries can be accelerated by running `ANALYZE TABLE tbl PERSISTENT FOR ...`, where `tbl` indicates a table used by a slow query. You can also run `ANALYZE TABLE ... PERSISTENT FOR ALL`, but that has a significant performance impact.

Useful Variables

The following overview indicates when a particular variable was introduced. When multiple versions are given, it means variable options (like the default value) changed between the indicated versions.

Variable	Introduced in MariaDB version	Description
analyze_max_length	MariaDB 10.6.23 / 10.11.14 / 11.4.8 / 11.8.3 / 12.0.2 / 12.1.1	Prevents the collection of column statistics for `CHAR` / `VARCHAR` columns that can be analyzed automatically by `ANALYZE TABLE PERSISTENT`. Columns exceeding this threshold in bytes will be skipped unless included explicitly in `FOR COLUMNS(...)`.
analyze_sample_percentage	10.4.3	Percentage of rows from the table ANALYZE TABLE will sample to collect table statistics. Set to 0 to let MariaDB decide what percentage of rows to sample.
histogram_type	10.4.3-11.0	Specifies the type of histograms created by ANALYZE. Options are #SINGLE_PREC_HB,DOUBLE_PREC_HB or JSON_HB.
histogram_size	10.7	Number of bytes or buckets (in case of JSON_HB) used for storing the histogram. If set to 0, no histograms are created by ANALYZE.]]

Variable	Introduced in MariaDB version	Description
log_slow_admin_statements	11.0.1	Log slow OPTIMIZE, ANALYZE, ALTER, and other administrative statements to the slow log if it is open. Deprecated. Use log_slow_filter instead.
log_slow_filter	(all versions)	Having an admin in the log_slow_filter will add slow ANALYZE_TABLE statements to the slow log.
sort_buffer_size	(all versions)	For calculating the number of duplicates, ANALYZE TABLE uses a buffer of sort_buffer_size bytes per column. You can slightly increase the speed of ANALYZE TABLE by increasing this variable.

Examples

```
-- update all engine-independent statistics for all columns and indexes
ANALYZE TABLE tbl PERSISTENT FOR ALL;

-- update specific columns and indexes:
ANALYZE TABLE tbl PERSISTENT FOR COLUMNS (col1,col2,...) INDEXES (idx1,idx2,...);

-- empty lists are allowed:
ANALYZE TABLE tbl PERSISTENT FOR COLUMNS (col1,col2,...) INDEXES ();
ANALYZE TABLE tbl PERSISTENT FOR COLUMNS () INDEXES (idx1,idx2,...);

-- the following will only update mysql.table_stats fields:
ANALYZE TABLE tbl PERSISTENT FOR COLUMNS () INDEXES ();

-- when use_stat_tables is set to 'COMPLEMENTARY' or 'PREFERABLY',
-- a simple ANALYZE TABLE  collects engine-independent statistics for all columns and indexes.
SET SESSION use_stat_tables='COMPLEMENTARY';
ANALYZE TABLE tbl;
```

See Also

- This one trick can make MariaDB 30x faster! (mariadb.org blog)
- Index Statistics
- InnoDB Persistent Statistics
- Progress Reporting
- Engine-independent Statistics
- Histogram-based Statistics
- ANALYZE Statement

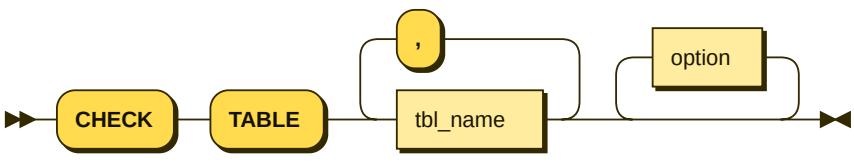
This page is licensed: GPLv2, originally from fill_help_tables.sql

CHECK TABLE

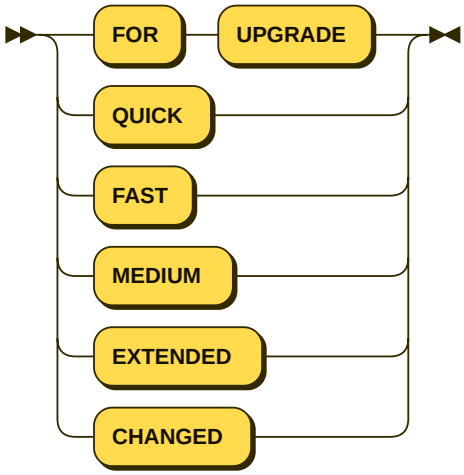
Syntax

```
CHECK TABLE tbl_name [, tbl_name] ... [option] ...

option = {FOR UPGRADE | QUICK | FAST | MEDIUM | EXTENDED | CHANGED}
```



Railroad diagram of CHECK TABLE



Railroad diagram of option

Description

`CHECK TABLE` checks a table or tables for errors. `CHECK TABLE` works for Archive, Aria, CSV, InnoDB, and MyISAM tables. For Aria and MyISAM tables, the key statistics are updated as well. For CSV, see also Checking and Repairing CSV Tables.

From MariaDB 12.0, `CHECK TABLE` also works on sequences. The sequence's underlying storage engine is checked first, then the sequence itself is validated.

As an alternative, myisamchk is a command-line tool for checking MyISAM tables when the tables are not being accessed. For Aria tables, there is a similar tool: aria_chk.

For checking dynamic columns' integrity, COLUMN_CHECK() can be used.

`CHECK TABLE` can also check views for problems, such as tables that are referenced in the view definition that no longer exist.

`CHECK TABLE` is also supported for partitioned tables. You can use `ALTER TABLE ... CHECK PARTITION` to check one or more partitions.

The meaning of the different options is as follows - note that this can vary a bit between storage engines:

Option	Explanation
FOR UPGRADE	Do a very quick check if the storage format for the table has changed so that one needs to do a `REPAIR`. This is only needed when one upgrades between major versions of MariaDB or MySQL. This is usually done by running mariadb-upgrade.
FAST	Only check tables that has not been closed properly or are marked as corrupt. Only supported by the MyISAM and Aria engines. For other engines the table is checked normally
CHANGED	Check only tables that have changed since the last `REPAIR` / `CHECK`. Only supported by the MyISAM and Aria engines. For other engines, the table is checked normally.
QUICK	Do a fast check. For MyISAM and Aria, this means skipping the check of the delete link chain, which may take some time.
MEDIUM	Scan also the data files. Checks the integrity between data and index files with checksums. In most cases, this should find all possible errors.
EXTENDED	Does a full check to verify every possible error. For InnoDB, Aria, and MyISAM, verify for each row that all its keys exist, and for those index keys, they point back to the primary clustered key. This may take a long time on large tables.

> **CURRENT**
>
> The `EXTENDED` option is available to InnoDB, too.
>
> **< 10.9.4 / 10.8.6 / 10.7.7 / 10.6.11**

> The `EXTENDED` option is ignored by InnoDB.

> **CURRENT**
>
> When the `EXTENDED` option is given, MariaDB also checks for referential integrity.
>
> **< 11.8**
>
> MariaDB does not check for referential integrity, even if the `EXTENDED` option is given.

For most cases, running `CHECK TABLE` without options or `MEDIUM` should be good enough.

The Aria storage engine supports progress reporting for this statement.

If you want to know if two tables are identical, take a look at CHECKSUM TABLE.

InnoDB

If `CHECK TABLE` finds an error in an InnoDB table, MariaDB might shut down to prevent the error propagation. In this case, the problem will be reported in the error log. Otherwise, the table or an index might be marked as corrupted, to prevent use. This does not happen with some minor problems, like a wrong number of entries in a secondary index. Those problems are reported in the output of `CHECK TABLE`.

Each tablespace contains a header with metadata. This header is not checked by this statement.

During the execution of `CHECK TABLE`, other threads may be blocked.

Examples

```
CHECK TABLE y EXTENDED;
+--------+-------+----------+----------+
| Table  | Op    | Msg_type | Msg_text |
+--------+-------+----------+----------+
| test.y | check | status   | OK       |
+--------+-------+----------+----------+
```

In newer MariaDB versions, the `EXTENDED` clause provides more information:

```
CHECK TABLE t1 EXTENDED;
+---------+-------+----------+-----------------------------------------------------------------------------+
| Table   | Op    | Msg_type | Msg_text                                                                    |
+---------+-------+----------+-----------------------------------------------------------------------------+
| test.t1 | check | Warning  | No suitable key found for foreign key t2_ibfk_1 in table test.t1    |
+---------+-------+----------+-----------------------------------------------------------------------------+

CHECK TABLE t2 EXTENDED;
+---------+-------+----------+-----------------------------------------------------------------------------------
----------------------+
| Table   | Op    | Msg_type | Msg_text
|
+---------+-------+----------+-----------------------------------------------------------------------------------
----------------------+
| test.t2 | check | status   | Cannot add or update a child row: a foreign key constraint fails (Key:
t2_ibfk_1, record: '2')  |
| test.t2 | check | status   | Cannot add or update a child row: a foreign key constraint fails (Key:
t2_ibfk_1, record: '2')  |
| test.t2 | check | status   | Cannot add or update a child row: a foreign key constraint fails (Key:
t2_ibfk_1, record: '3')  |
| test.t2 | check | error    | Corrupt
|
+---------+-------+----------+-----------------------------------------------------------------------------------
----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

CHECK VIEW

Syntax

```
CHECK VIEW view_name
```

Description

The `CHECK VIEW` statement checks whether the view algorithm is correct. It is run as part of mariadb-upgrade, and should not normally be required in regular use.

See Also

- REPAIR VIEW

This page is licensed: CC BY-SA / Gnu FDL

CHECKSUM TABLE

Syntax

```
CHECKSUM TABLE tbl_name [, tbl_name] ... [ QUICK | EXTENDED ]
```

Description

`CHECKSUM TABLE` reports a table checksum. This is very useful if you want to know if two tables are the same (for example on a master and a replica).

With `QUICK`, the live table checksum is reported if it is available, or `NULL` otherwise. This is very fast. A live checksum is enabled by specifying the `CHECKSUM=1` table option when you create the table; currently, this is supported only for Aria and MyISAM tables.

With `EXTENDED`, the entire table is read row by row and the checksum is calculated. This can be very slow for large tables.

If neither `QUICK` nor `EXTENDED` is specified, MariaDB returns a live checksum if the table storage engine supports it and scans the table otherwise.

`CHECKSUM TABLE` requires the SELECT privilege for the table.

For a nonexistent table, `CHECKSUM TABLE` returns `NULL` and generates a warning.

The table row format affects the checksum value. If the row format changes, the checksum will change. This means that when a table created with a MariaDB/MySQL version is upgraded to another version, the checksum value will probably change.

Two identical tables should always match to the same checksum value; however, also for non-identical tables there is a very slight chance that they will return the same value as the hashing algorithm is not completely collision-free.

Identical Tables

Identical tables mean that the CREATE statement is identical and that the following variable, which affects the storage formats, was the same when the tables were created:

- mysql56-temporal-format

Differences Between MariaDB and MySQL

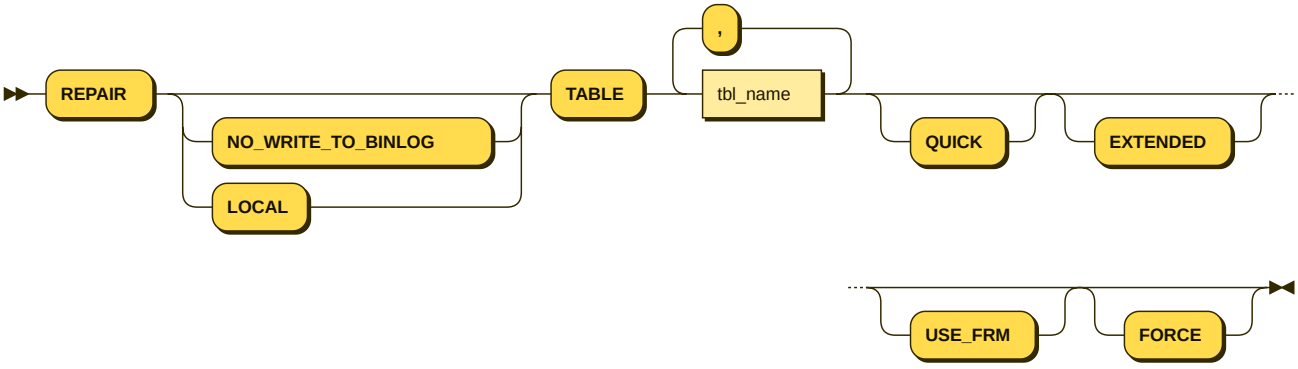
`CHECKSUM TABLE` may give a different result as MariaDB doesn't ignore `NULL`s in the columns like MySQL 5.1 does (later MySQL versions should calculate checksums the same way as MariaDB).

> **CURRENT**
>
> You can get the 'old style' checksum in MariaDB by setting old_mode to COMPAT_5_1_CHECKSUM. Note, however, that the MyISAM and Aria storage engines in MariaDB are using the new checksum internally, so if you are using this old mode, the `CHECKSUM` command will be slower as it needs to calculate the checksum row by row.
>
> **< 10.9**
>
> You can get the 'old style' checksum in MariaDB by starting mariadbd with the –old option. Note, however, that the MyISAM and Aria storage engines in MariaDB are using the new checksum internally, so if you are using this old mode, the `CHECKSUM` command will be slower as it needs to calculate the checksum row by row.

This page is licensed: GPLv2, originally from fill_help_tables.sql

REPAIR TABLE

Syntax

> **CURRENT**
>
> ```
> REPAIR [NO_WRITE_TO_BINLOG | LOCAL] TABLE
>     tbl_name [, tbl_name] ...
>     [QUICK] [EXTENDED] [USE_FRM] [FORCE]
> ```
>
> 
>
>
> *Railroad diagram of REPAIR TABLE — equivalent to the BNF above*
>
> **< 11.5**
>
> ```
> REPAIR [NO_WRITE_TO_BINLOG | LOCAL] TABLE
>     tbl_name [, tbl_name] ...
>     [QUICK] [EXTENDED] [USE_FRM]
> ```

Description

`REPAIR TABLE` repairs a possibly corrupted table. By default, it has the same effect as

```
myisamchk --recover tbl_name
```

or

```
aria_chk --recover tbl_name
```

See aria_chk and myisamchk for more.

`REPAIR TABLE` works for Archive, Aria, CSV, and MyISAM tables. For InnoDB, see recovery modes. For CSV, see also Checking and Repairing CSV Tables. For Archive, this statement also improves compression. If the storage engine does not support this statement, a warning is issued.

This statement requires SELECT and INSERT privileges for the table.

By default, `REPAIR TABLE` statements are written to the binary log and will be replicated. The `NO_WRITE_TO_BINLOG` keyword (`LOCAL` is an alias) will ensure the statement is not written to the binary log.

> **CURRENT**
>
> `REPAIR TABLE` statements are not logged to the binary log if read_only is set. See also Read-Only Replicas.
>
> **< 10.3.19**
>
> `REPAIR TABLE` statements are logged to the binary log.

When an index is recreated, the storage engine may use a configurable buffer in the process. Incrementing the buffer speeds up the index creation. Aria and MyISAM allocate a buffer whose size is defined by aria_sort_buffer_size or myisam_sort_buffer_size, also used for ALTER TABLE.

QUICK

When specified, `REPAIR TABLE` will not modify the data file, only attempting to repair the index file. The same behavior can be achieved with myisamchk –recover –quick.

EXTENDED

Creates the index row by row rather than sorting and creating a single index. Similar to myisamchk –safe-recover.

USE_FRM

For use only when the index file is missing or its header corrupted. MariaDB then attempts to recreate it using the `.frm` file. There is no equivalent myisamchk option.

FORCE

> **CURRENT**
>
> The `FORCE` argument allows to first run internal repair to fix damaged blocks and then follow it up with `ALTER TABLE` (MDEV-33449).
>
> **< 11.5**
>
> The `FORCE` option is not available.

Partitioned Tables

`REPAIR TABLE` is also supported for partitioned tables with the ALTER TABLE … REPAIR PARTITION statement. However, the `USE_FRM` option cannot be used with this statement on a partitioned table. See Repairing Partitions for details.

Progress Reporting

The Aria storage engine supports progress reporting for this statement.

See Also

- mariadb-check
- aria_chk
- myisamchk

This page is licensed: GPLv2, originally from fill_help_tables.sql

REPAIR VIEW

Syntax

```
REPAIR [NO_WRITE_TO_BINLOG | LOCAL] VIEW  view_name[, view_name] ... [FROM MYSQL]
```

Description

The `REPAIR VIEW` statement checks whether the view algorithm is correct. It is run as part of mariadb-upgrade, and should not normally be required in regular use.

By default it corrects the checksum and if necessary adds the mariadb-version field. If the optional `FROM MYSQL` clause is used, and no mariadb-version field is present, the `MERGE` and `TEMPTABLE` algorithms are toggled.

By default, `REPAIR VIEW` statements are written to the binary log and will be replicated. The `NO_WRITE_TO_BINLOG` keyword (`LOCAL` is an alias) will ensure the statement is not written to the binary log.

See Also

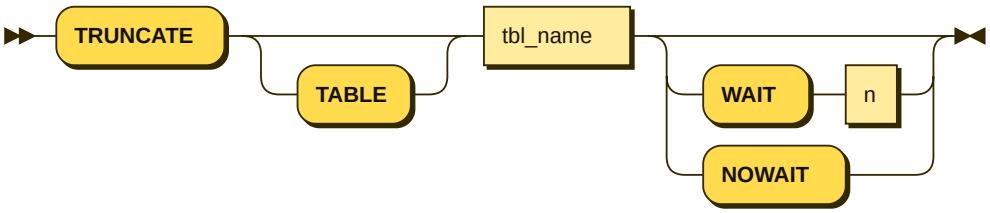
- CHECK VIEW

This page is licensed: CC BY-SA / Gnu FDL

TRUNCATE TABLE

Syntax

```
TRUNCATE [TABLE] tbl_name
  [WAIT n | NOWAIT]
```



Railroad diagram of TRUNCATE TABLE — equivalent to the BNF above

Description

`TRUNCATE TABLE` empties a table completely. It requires the `DROP` privilege. See GRANT.

`tbl_name` can also be specified in the form `db_name.tbl_name` (see Identifier Qualifiers).

Logically, `TRUNCATE TABLE` is equivalent to a DELETE statement that deletes all rows, but there are practical differences under some circumstances.

`TRUNCATE TABLE` will fail for an InnoDB table if any FOREIGN KEY constraints from other tables reference the table, returning the error:

```
ERROR 1701 (42000): Cannot truncate a table referenced in a foreign key constraint
```

Foreign Key constraints between columns in the same table are permitted.

For an InnoDB table, if there are no `FOREIGN KEY` constraints, InnoDB performs fast truncation by dropping the original table and creating an empty one with the same definition, which is much faster than deleting rows one by one. The AUTO_INCREMENT counter is reset by `TRUNCATE TABLE`, regardless of whether there is a `FOREIGN KEY` constraint.

The count of rows affected by `TRUNCATE TABLE` is accurate only when it is mapped to a `DELETE` statement.

For other storage engines, `TRUNCATE TABLE` differs from `DELETE` in the following ways:

- Truncate operations drop and re-create the table, which is much faster than deleting rows one by one, particularly for large tables.
- Truncate operations cause an implicit commit.
- Truncation operations cannot be performed if the session holds an active table lock.
- Truncation operations do not return a meaningful value for the number of deleted rows. The usual result is "0 rows affected," which should be interpreted as "no information."
- As long as the table format file `tbl_name.frm` is valid, the table can be re-created as an empty table with `TRUNCATE TABLE`, even if the data or index files have become corrupted.
- The table handler does not remember the last used AUTO_INCREMENT value, but starts counting from the beginning. This is true even for MyISAM and InnoDB, which normally do not reuse sequence values.
- When used with partitioned tables, `TRUNCATE TABLE` preserves the partitioning; that is, the data and index files are dropped and re-created, while the partition definitions (.par) file is unaffected.
- Since truncation of a table does not make any use of `DELETE`, the `TRUNCATE` statement does not invoke `ON DELETE` triggers.
- `TRUNCATE TABLE` will only reset the values in the Performance Schema summary tables to zero or null, and will not remove the rows.

For the purposes of binary logging and replication, `TRUNCATE TABLE` is treated as DROP TABLE followed by CREATE TABLE (DDL rather than DML).

`TRUNCATE TABLE` does not work on views. It also cannot be used on a system-versioned table, returning the error:

```
ERROR 4137 (HY000): System-versioned tables do not support TRUNCATE TABLE
```

To remove historical records from a system-versioned table, use DELETE HISTORY instead.

WAIT/NOWAIT

Set the lock wait timeout. See WAIT and NOWAIT.

Oracle-mode

Oracle-mode permits the optional keywords `REUSE STORAGE` or `DROP STORAGE` to be used.

```
TRUNCATE [TABLE] tbl_name [{DROP | REUSE} STORAGE] [WAIT n | NOWAIT]
```

These have no effect on the operation.

Performance

`TRUNCATE TABLE` is faster than DELETE, because it drops and re-creates a table.

With InnoDB, `TRUNCATE TABLE` is slower if innodb_file_per_table=ON is set (the default). This is because `TRUNCATE TABLE` unlinks the underlying tablespace file, which can be an expensive operation. See MDEV-8069 for more details.

The performance issues with innodb_file_per_table=ON can be exacerbated in cases where the InnoDB buffer pool is very large and innodb_adaptive_hash_index=ON is set. In that case, using DROP TABLE followed by CREATE TABLE instead of `TRUNCATE TABLE` may perform better. Setting innodb_adaptive_hash_index=OFF can also help.

Setting innodb_adaptive_hash_index=OFF can also improve `TRUNCATE TABLE` performance in general. See MDEV-16796 for more details.

See Also

- TRUNCATE function
- innodb_safe_truncate system variable
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

Transactions

See COMMIT.

Commit the current transaction. This statement permanently saves all changes made during the current transaction to the database.

See LOCK TABLES.

Complete LOCK TABLES reference: READ/WRITE/WRITE CONCURRENT lock syntax, table aliases, WAIT n|NOWAIT timeouts, UNLOCK TABLES, and innodb_table_locks behavior.

See Metadata Locking.

Understand how MariaDB manages concurrency. Metadata locks protect the structure of database objects from being modified while they are in use.

See READ COMMITTED.

Set the transaction isolation level to READ COMMITTED. In this mode, each query within a transaction sees only data committed before the query began.

See READ UNCOMMITTED.

Set the transaction isolation level to READ UNCOMMITTED. This lowest isolation level allows dirty reads, where a transaction can see uncommitted changes.

See REPEATABLE READ.

Set the transaction isolation level to REPEATABLE READ. This default InnoDB level ensures consistent results for repeated reads within the same transaction.

See ROLLBACK.

Undo changes in the current transaction. This statement reverts the database to its state before the transaction started or to a specific savepoint.

See SAVEPOINT.

Create a named marker within a transaction. Savepoints allow you to roll back part of a transaction without canceling the entire operation.

See SERIALIZABLE.

Set the transaction isolation level to SERIALIZABLE. This highest level ensures total isolation by converting plain SELECTs to locking reads.

See SQL statements Causing an Implicit Commit.

Identify statements that force a commit. Certain commands, like DDL statements, implicitly commit the current transaction before executing.

See START TRANSACTION.

Complete START TRANSACTION reference: BEGIN/COMMIT/ROLLBACK syntax, WITH CONSISTENT SNAPSHOT option, READ ONLY/WRITE modes, AND [NO] CHAIN/RELEASE modifiers.

See Transaction Timeouts.

Understand how timeouts affect transactions. This section explains system variables that control wait times for locks and transaction duration.

See UNLOCK TABLES.

Release explicit table locks. This statement releases all locks acquired by the current session with LOCK TABLES.

See WAIT and NOWAIT.

Control lock wait behavior. These clauses allow statements to wait for a specific timeout or fail immediately if a lock cannot be acquired.

See XA Transactions.

Manage distributed transactions. This section covers XA statements for coordinating two-phase commits across multiple resources.

This page is licensed: CC BY-SA / Gnu FDL

COMMIT

The `COMMIT` statement ends a transaction, saving any changes to the data so that they become visible to subsequent transactions. Also, unlocks metadata changed by current transaction. If autocommit is set to 1, an implicit commit is performed after each statement. Otherwise, all transactions which don't end with an explicit `COMMIT` are implicitly rollbacked and the changes are lost. The ROLLBACK statement can be used to do this explicitly.

The required syntax for the `COMMIT` statement is as follows:

```
COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
```

`COMMIT` is the more important transaction terminator, as well as the more interesting one. The basic form of the `COMMIT` statement is simply the keyword `COMMIT` (the keyword `WORK` is simply noise and can be omitted without changing the effect).

The optional `AND CHAIN` clause is a convenience for initiating a new transaction as soon as the old transaction terminates. If `AND CHAIN` is specified, then there is effectively nothing between the old and new transactions, although they remain separate. The characteristics of the new transaction will be the same as the characteristics of the old one — that is, the new transaction will have the same access mode, isolation level and diagnostics area size (we'll discuss all of these shortly) as the transaction just terminated.

`RELEASE` tells the server to disconnect the client immediately after the current transaction.

There are `NO RELEASE` and `AND NO CHAIN` options. By default, commits do not `RELEASE` or `CHAIN`, but it's possible to change this default behavior with the completion_type server system variable. In this case, the `AND NO CHAIN` and `NO RELEASE` options override the server default.

See Also

- autocommit - server system variable that determines whether statements are automatically committed.
- completion_type - server system variable that determines whether COMMIT's are standard, COMMIT AND CHAIN or COMMIT RELEASE.
- SQL statements that cause an implicit commit

This page is licensed: CC BY-SA / Gnu FDL

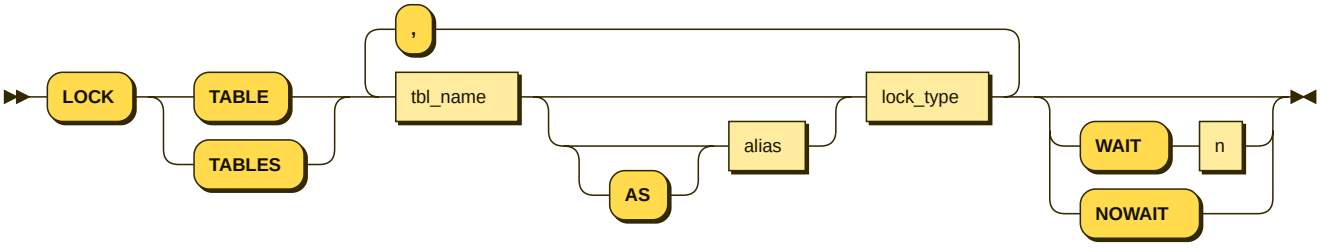
LOCK TABLES

Syntax

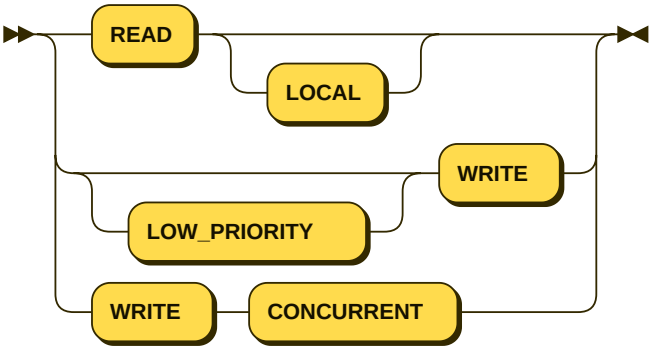
```
LOCK TABLE[S]
    tbl_name [[AS] alias] lock_type
    [, tbl_name [[AS] alias] lock_type] ...
    [WAIT n|NOWAIT]

lock_type:
    READ [LOCAL]
  | [LOW_PRIORITY] WRITE
  | WRITE CONCURRENT

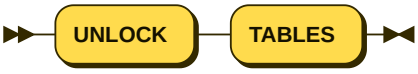
UNLOCK TABLES
```



Railroad diagram of LOCK TABLES



Railroad diagram of lock_type



Railroad diagram of UNLOCK TABLES

Description

The *lock_type* can be one of:

Option	Description
READ	Read lock, no writes allowed
READ LOCAL	Read lock, but allow concurrent inserts

Option	Description
WRITE	Exclusive write lock. No other connections can read or write to this table
LOW_PRIORITY WRITE	Exclusive write lock, but allow new read locks on the table until we get the write lock.
WRITE CONCURRENT	Exclusive write lock, but allow READ LOCAL locks to the table.

MariaDB enables client sessions to acquire table locks explicitly for the purpose of cooperating with other sessions for access to tables, or to prevent other sessions from modifying tables during periods when a session requires exclusive access to them. A session can acquire or release locks only for itself. One session cannot acquire locks for another session or release locks held by another session.

Locks may be used to emulate transactions or to get more speed when updating tables.

`LOCK TABLES` explicitly acquires table locks for the current client session. Table locks can be acquired for base tables or views. To use `LOCK TABLES`, you must have the `LOCK TABLES` privilege, and the `SELECT` privilege for each object to be locked. See GRANT.

For view locking, `LOCK TABLES` adds all base tables used in the view to the set of tables to be locked and locks them automatically. If you lock a table explicitly with `LOCK TABLES`, any tables used in triggers are also locked implicitly, as described in Triggers and Implicit Locks.

UNLOCK TABLES explicitly releases any table locks held by the current session.

Aliases

Aliases need to correspond to the aliases used in prior SQL statements in the session. For example:

```
LOCK TABLE t1 AS t1_alias1 READ;

SELECT * FROM t1;
ERROR 1100 (HY000): Table 't1' was not locked with LOCK TABLES

SELECT * FROM t1 AS t1_alias2;
ERROR 1100 (HY000): Table 't1_alias2' was not locked with LOCK TABLES

SELECT * FROM t1 AS t1_alias1;
```

WAIT/NOWAIT

Set the lock wait timeout. See WAIT and NOWAIT.

Limitations

- `LOCK TABLES` doesn't work when using Galera cluster. You may experience crashes or locks when used with Galera.
- `LOCK TABLES` works on XtraDB/InnoDB tables only if the innodb_table_locks system variable is set to 1 (the default) and autocommit is set to 0 (1 is default). Please note that no error message will be returned on LOCK TABLES with innodb_table_locks = 0.
- `LOCK TABLES` implicitly commits the active transaction, if any. Also, starting a transaction always releases all table locks acquired with LOCK TABLES. This means that there is no way to have table locks and an active transaction at the same time. The only exceptions are the transactions in autocommit mode. To preserve the data integrity between transactional and non-transactional tables, the GET_LOCK() function can be used.
- When using `LOCK TABLES` on a `TEMPORARY` table, it will always be locked with a `WRITE` lock.
- While a connection holds an explicit read lock on a table, it cannot modify it. If you try, the following error will be produced:

```
ERROR 1099 (HY000): Table 'tab_name' was locked with a READ lock and can't be updated
```

- While a connection holds an explicit lock on a table, it cannot access a non-locked table. If you try, the following error will be produced:

```
ERROR 1100 (HY000): Table 'tab_name' was not locked with LOCK TABLES
```

- While a connection holds an explicit lock on a table, it cannot issue the following: INSERT DELAYED, CREATE TABLE, CREATE TABLE … LIKE, and DDL statements involving stored programs and views (except for triggers). If you try, the following error will be produced:

```
ERROR 1192 (HY000): Can't execute the given command because you have active locked tables or an active
transaction
```

- `LOCK TABLES` can not be used in stored routines - if you try, the following error will be produced on creation:

```
ERROR 1314 (0A000): LOCK is not allowed in stored procedures
```

See Also

- UNLOCK TABLES

This page is licensed: GPLv2, originally from fill_help_tables.sql

Metadata Locking

MariaDB supports metadata locking. This means that when a transaction (including XA transactions) uses a table, it locks its metadata until the end of transaction. Non-transactional tables are also locked, as well as views and objects which are related to locked tables/views (stored functions, triggers, etc). When a connection tries to use a DDL statement (like an ALTER TABLE) which modifies a table that is locked, that connection is queued, and has to wait until it's unlocked. Using savepoints and performing a partial rollback does not release metadata locks.

LOCK TABLES … WRITE are also queued. Some wrong statements which produce an error may not need to wait for the lock to be freed.

The metadata lock's timeout is determined by the value of the lock_wait_timeout server system variable (in seconds). However, note that its default value is 31536000 (1 year). If this timeout is exceeded, the following error is returned:

```
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
```

If the metadata_lock_info plugin is installed, the Information Schema metadata_lock_info table stores information about existing metadata locks.

CURRENT

The Performance Schema metadata_locks table contains metadata lock information.

Example

Let's use the following MEMORY (non-transactional) table:

```
CREATE TABLE t (a INT) ENGINE = MEMORY;
```

Connection 1 starts a transaction, and INSERTs a row into t:

```
START TRANSACTION;

INSERT INTO t SET a=1;
```

`t`'s metadata is now locked by connection 1. Connection 2 tries to alter `t`, but has to wait:

```
ALTER TABLE t ADD COLUMN b INT;
```

Connection 2's prompt is blocked now.

Now connection 1 ends the transaction:

```
COMMIT;
```

…and connection 2 finally gets the output of its command:

```
Query OK, 1 row affected (35.23 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

< 10.5.2

The Performance Schema metadata_locks table does **not** contain metadata lock information.

This page is licensed: CC BY-SA / Gnu FDL

READ COMMITTED

`READ COMMITTED` is one of the transaction isolation levels. Each consistent read, even within the same transaction, sets and reads its own fresh snapshot.

See Isolation Levels for details.

This page is licensed: CC BY-SA / Gnu FDL

READ UNCOMMITTED

`READ UNCOMMITTED` is one of the transaction isolation levels. `SELECT` statements are performed in a non-locking fashion, but a possible earlier version of a row might be used.

See Isolation Levels for details.

This page is licensed: CC BY-SA / Gnu FDL

REPEATABLE READ

`REPEATABLE READ` is one of the transaction isolation levels. All consistent reads within the same transaction read the snapshot established by the first read.

See Isolation Levels for details.

This page is licensed: CC BY-SA / Gnu FDL

ROLLBACK

The `ROLLBACK` statement rolls back (ends) a transaction, destroying any changes to SQL-data so that they never become visible to subsequent transactions. The required syntax for the `ROLLBACK` statement is as follows.

```
ROLLBACK [ WORK ] [ AND [ NO ] CHAIN ]
[ TO [ SAVEPOINT ] {<savepoint name> | <simple target specification>} ]
```

The `ROLLBACK` statement will either end a transaction, destroying all data changes that happened during any of the transaction, or it will just destroy any data changes that happened since you established a savepoint. The basic form

of the `ROLLBACK` statement is just the keyword `ROLLBACK` (the keyword `WORK` is simply noise and can be omitted without changing the effect).

The optional `AND CHAIN` clause is a convenience for initiating a new transaction as soon as the old transaction terminates. If `AND CHAIN` is specified, then there is effectively nothing between the old and new transactions, although they remain separate. The characteristics of the new transaction will be the same as the characteristics of the old one — that is, the new transaction will have the same access mode, isolation level and diagnostics area size (we'll discuss all of these shortly) as the transaction just terminated. The `AND NO CHAIN` option just tells your DBMS to end the transaction — that is, these four SQL statements are equivalent:

```
ROLLBACK;
ROLLBACK WORK;
ROLLBACK AND NO CHAIN;
ROLLBACK WORK AND NO CHAIN;
```

All of them end a transaction without saving any transaction characteristics. The only other options, the equivalent statements …

```
ROLLBACK AND CHAIN;
ROLLBACK WORK AND CHAIN;
```

… both tell your DBMS to end a transaction, but to save that transaction's characteristics for the next transaction.

`ROLLBACK` is much simpler than `COMMIT`: it may involve no more than a few deletions (of Cursors, locks, prepared SQL statements and log-file entries). It's usually assumed that `ROLLBACK` can't fail, although such a thing is conceivable (for example, an encompassing transaction might reject an attempt to `ROLLBACK` because it's lining up for a `COMMIT`).

`ROLLBACK` cancels all effects of a transaction. It does not cancel effects on objects outside the DBMS's control (for example the values in host program variables or the settings made by some SQL/CLI function calls). But in general, it is a convenient statement for those situations when you say "oops, this isn't working" or when you simply don't care whether your temporary work becomes permanent or not.

Here is a moot question. If all you've been doing is `SELECT`s, so that there have been no data changes, should you end the transaction with `ROLLBACK` or `COMMIT`? It shouldn't really matter because both `ROLLBACK` and `COMMIT` do the same transaction-terminating job. However, the popular conception is that `ROLLBACK` implies failure, so after a successful series of `SELECT` statements the convention is to end the transaction with `COMMIT` rather than `ROLLBACK`.

MariaDB (and most other DBMSs) supports rollback of SQL-data change statements, but not of SQL-Schema statements. This means that if you use any of `CREATE`, `ALTER`, `DROP`, `GRANT`, `REVOKE`, you are implicitly committing at execution time.

```
INSERT INTO Table_2 VALUES(5);
DROP TABLE Table_3 CASCADE;
ROLLBACK;
```

The result will be that both the `INSERT` and the `DROP` will go through as separate transactions so the `ROLLBACK` will have no effect.

See Also

- Flashback, rollback transactions in binary log

This page is licensed: CC BY-SA / Gnu FDL

SAVEPOINT

Syntax

```
SAVEPOINT identifier
ROLLBACK [WORK] TO [SAVEPOINT] identifier
RELEASE SAVEPOINT identifier
```

Description

InnoDB supports the SQL statements `SAVEPOINT`, `ROLLBACK TO SAVEPOINT`, `RELEASE SAVEPOINT` and the optional `WORK` keyword for `ROLLBACK`.

Each savepoint must have a legal MariaDB identifier. A savepoint is a named sub-transaction.

Normally ROLLBACK undoes the changes performed by the whole transaction. When used with the `TO` clause, it undoes the changes performed after the specified savepoint, and erases all subsequent savepoints. However, all locks that have been acquired after the save point will survive. `RELEASE SAVEPOINT` does not rollback or commit any changes, but removes the specified savepoint.

When the execution of a trigger or a stored function begins, it is not possible to use statements which reference a savepoint which was defined from out of that stored program.

When a COMMIT (including implicit commits) or a `ROLLBACK` statement (with no `TO` clause) is performed, they act on the whole transaction, and all savepoints are removed.

Errors

If `COMMIT` or `ROLLBACK` is issued and no transaction was started, no error is reported.

If `SAVEPOINT` is issued and no transaction was started, no error is reported but no savepoint is created. When `ROLLBACK TO SAVEPOINT` or `RELEASE SAVEPOINT` is called for a savepoint that does not exist, an error like this is issued:

```
ERROR 1305 (42000): SAVEPOINT svp_name does not exist
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SERIALIZABLE

`SERIALIZABLE` is one of the transaction isolation levels. Similar to `REPEATABLE READ`, but InnoDB implicitly converts all plain `SELECT` statements to `SELECT ... LOCK IN SHARE MODE` if autocommit is disabled.

See Isolation Levels for details.

This page is licensed: CC BY-SA / Gnu FDL

SQL statements Causing an Implicit Commit

Some SQL statements cause an implicit commit. As a rule of thumb, such statements are DDL statements. The same statements (except for SHUTDOWN) produce a 1400 error (SQLSTATE 'XAE09') if a XA transaction is in effect.

Here is the list:

```
ALTER DATABASE ... UPGRADE DATA DIRECTORY NAME
ALTER EVENT
ALTER FUNCTION
ALTER PROCEDURE
ALTER SEQUENCE
ALTER SERVER
ALTER TABLE
ALTER VIEW
ANALYZE TABLE
BEGIN
CACHE INDEX
CHANGE MASTER TO
CHECK TABLE
CREATE DATABASE
CREATE EVENT
CREATE FUNCTION
CREATE INDEX
CREATE PROCEDURE
CREATE ROLE
CREATE SEQUENCE
CREATE SERVER
CREATE TABLE
CREATE TRIGGER
CREATE USER
CREATE VIEW
DROP DATABASE
DROP EVENT
DROP FUNCTION
DROP INDEX
DROP PROCEDURE
DROP ROLE
DROP SEQUENCE
DROP SERVER
DROP TABLE
DROP TRIGGER
DROP USER
DROP VIEW
FLUSH
GRANT
LOAD INDEX INTO CACHE
LOCK TABLES
OPTIMIZE TABLE
RENAME TABLE
RENAME USER
REPAIR TABLE
RESET
REVOKE
SET PASSWORD
SHUTDOWN
START SLAVE
START TRANSACTION
STOP SLAVE
TRUNCATE TABLE
```

`SET autocommit = 1` causes an implicit commit if the value was 0.

All these statements cause an implicit commit before execution. This means that, even if the statement fails with an error, the transaction is committed. Some of them, like `CREATE TABLE ... SELECT`, also cause a commit immediately after execution. Such statements couldn't be rollbacked in any case.

If you are not sure whether a statement has implicitly committed the current transaction, you can query the in_transaction server system variable.

Note that when a transaction starts (not in autocommit mode), all locks acquired with LOCK TABLES are released. And acquiring such locks always commits the current transaction. To preserve the data integrity between transactional and non-transactional tables, the GET_LOCK() function can be used.

Exceptions

These statements do not cause an implicit commit in the following cases:

- CREATE TABLE and DROP TABLE, when the `TEMPORARY` keyword is used.
  - However, TRUNCATE TABLE causes an implicit commit even when used on a temporary table.
- CREATE FUNCTION and DROP FUNCTION, when used to create a UDF (instead of a stored function). However, CREATE INDEX and DROP INDEX cause commits even when used with temporary tables.
- UNLOCK TABLES causes a commit only if a LOCK TABLES was used on non-transactional tables.

This page is licensed: CC BY-SA / Gnu FDL

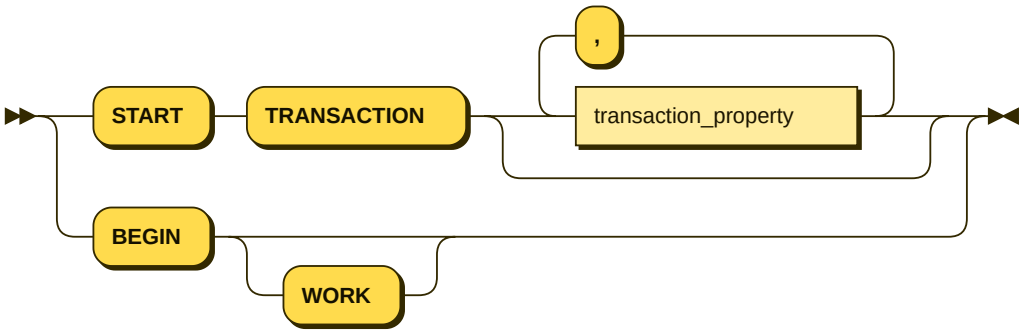
START TRANSACTION

Syntax

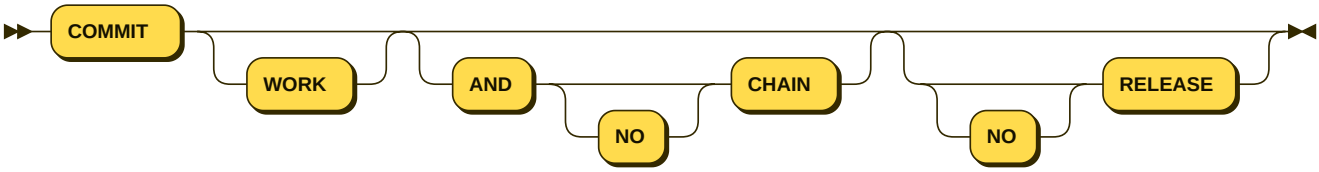
```
START TRANSACTION [transaction_property [, transaction_property] ...] | BEGIN [WORK]
COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
SET autocommit = {0 | 1}

transaction_property:
    WITH CONSISTENT SNAPSHOT
  | READ WRITE
  | READ ONLY
```

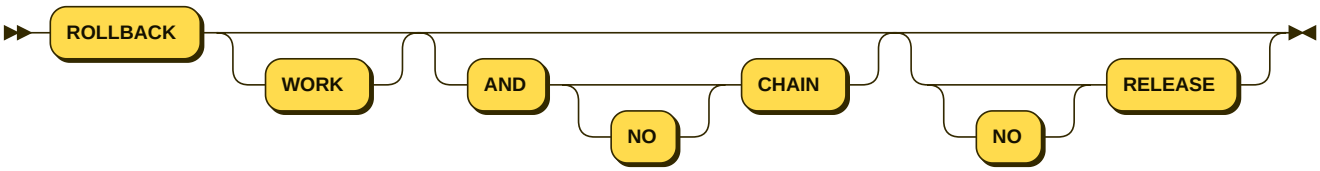
The BNF above documents four related statements together; each gets its own diagram below.



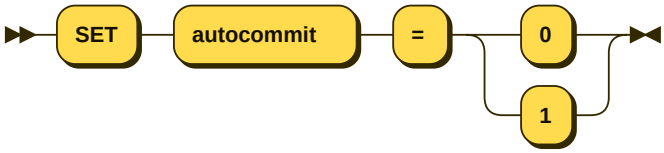
Railroad diagram of START TRANSACTION / BEGIN



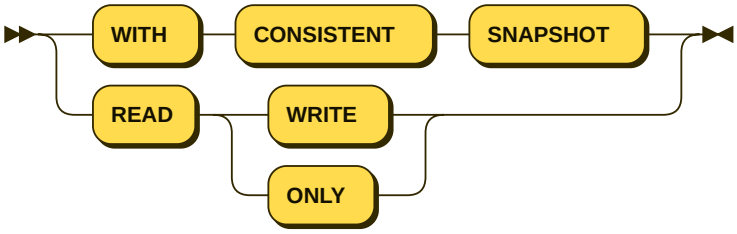
Railroad diagram of COMMIT



Railroad diagram of ROLLBACK



Railroad diagram of SET autocommit



Railroad diagram of transaction_property

Description

The `START TRANSACTION` or `BEGIN` statement begins a new transaction. COMMIT commits the current transaction, making its changes permanent. ROLLBACK rolls back the current transaction, canceling its changes. The SET autocommit statement disables or enables the default autocommit mode for the current session.

`START TRANSACTION` and `SET` autocommit = 1 implicitly commit the current transaction, if there is one.

The optional `WORK` keyword is supported for `COMMIT` and `ROLLBACK`, as are the `CHAIN` and `RELEASE` clauses. `CHAIN` and `RELEASE` can be used for additional control over transaction completion. The value of the completion_type system variable determines the default completion behavior.

The `AND CHAIN` clause causes a new transaction to begin as soon as the current one ends, and the new transaction has the same isolation level as the just-terminated transaction. The `RELEASE` clause causes the server to disconnect the current client session after terminating the current transaction. Including the `NO` keyword suppresses `CHAIN` or `RELEASE` completion, which can be useful if the completion_type system variable is set to cause chaining or release completion by default.

Access Mode

> **CURRENT**
>
> The access mode specifies whether the transaction is allowed to write data or not. By default, transactions are in `READ WRITE` mode (see the tx_read_only system variable). `READ ONLY` mode allows the storage engine to apply optimizations that cannot be used for transactions which write data. Note that, unlike the global read_only mode, the READ_ONLY ADMIN privilege doesn't allow writes and DDL statements on temporary tables are not allowed either.
>
> **< 10.11**
>
> The access mode specifies whether the transaction is allowed to write data or not. By default, transactions are in `READ WRITE` mode (see the tx_read_only system variable). `READ ONLY` mode allows the storage engine to apply optimizations that cannot be used for transactions which write data. Note that, unlike the global read_only mode, the SUPER privilege doesn't allow writes and DDL statements on temporary tables are not allowed either.

It is not permitted to specify both `READ WRITE` and `READ ONLY` in the same statement.

`READ WRITE` and `READ ONLY` can also be specified in the SET TRANSACTION statement, in which case the specified mode is valid for all sessions, or for all subsequent transaction used by the current session.

autocommit

By default, MariaDB runs with autocommit mode enabled. This means that as soon as you execute a statement that updates (modifies) a table, MariaDB stores the update on disk to make it permanent. To disable autocommit mode, use the following statement:

```
SET autocommit=0;
```

After disabling autocommit mode by setting the autocommit variable to zero, changes to transaction-safe tables (such as those for InnoDB or `NDBCLUSTER`) are not made permanent immediately. You must use `COMMIT` to store your changes to disk or `ROLLBACK` to ignore the changes.

To disable autocommit mode for a single series of statements, use the `START TRANSACTION` statement.

DDL Statements

DDL statements (`CREATE`, `ALTER`, `DROP`) and administrative statements (`FLUSH`, `RESET`, `OPTIMIZE`, `ANALYZE`, `CHECK`, `REPAIR`, `CACHE INDEX`), transaction management statements (`BEGIN`, `START TRANSACTION`) and `LOAD DATA INFILE`, cause an implicit `COMMIT` and start a new transaction. An exception to this rule are the DDL that operate on temporary tables: you can `CREATE`, `ALTER` and `DROP` them without causing any `COMMIT`, but those actions cannot be rolled back. This means that if you call `ROLLBACK`, the temporary tables you created in the transaction will remain, while the rest of the transaction will be rolled back.

Transactions cannot be used in Stored Functions or Triggers. In Stored Procedures and Events `BEGIN` is not allowed, so you should use `START TRANSACTION` instead.

A transaction acquires a metadata lock on every table it accesses to prevent other connections from altering their structure. The lock is released at the end of the transaction. This happens even with non-transactional storage engines (like MEMORY or CONNECT), so it makes sense to use transactions with non-transactional tables.

in_transaction

The in_transaction system variable is a session-only, read-only variable that returns `1` inside a transaction, and `0` if not in a transaction.

WITH CONSISTENT SNAPSHOT

The `WITH CONSISTENT SNAPSHOT` option starts a consistent read for storage engines such as InnoDB that can do so, the same as if a `START TRANSACTION` followed by a `SELECT` from any InnoDB table was issued.

See Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT.

Examples

```
START TRANSACTION;
SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summary=@A WHERE type=1;
COMMIT;
```

See Also

- Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT
- MyRocks and START TRANSACTION WITH CONSISTENT SNAPSHOT

This page is licensed: GPLv2, originally from fill_help_tables.sql

Transaction Timeouts

MariaDB has always had the wait_timeout and interactive_timeout settings, which close connections after a certain period of inactivity.

However, these are by default set to a long wait period. In situations where transactions may be started, but not committed or rolled back, more granular control and a shorter timeout may be desirable so as to avoid locks being held for too long.

> **CURRENT**
>
> These variables help handle this situation:
>
> - idle_transaction_timeout (all transactions)
> - idle_write_transaction_timeout (write transactions)
> - idle_readonly_transaction_timeout (read transactions)
>
> **< 10.3**
>
> There is no variables for more granular control.

These accept a time in seconds to time out, by closing the connection, transactions that are idle for longer than this period. By default all are set to zero, or no timeout.

idle_transaction_timeout affects all transactions, idle_write_transaction_timeout affects write transactions only and idle_readonly_transaction_timeout affects read transactions only. The latter two variables work independently. However, if either is set along with idle_transaction_timeout, the settings for idle_write_transaction_timeout or idle_readonly_transaction_timeout will take precedence.

Examples

```
SET SESSION idle_transaction_timeout=2;
BEGIN;
SELECT * FROM t;
Empty set (0.000 sec)
## wait 3 seconds
SELECT * FROM t;
ERROR 2006 (HY000): MySQL server has gone away
```

```
SET SESSION idle_write_transaction_timeout=2;
BEGIN;
SELECT * FROM t;
Empty set (0.000 sec)
## wait 3 seconds
SELECT * FROM t;
Empty set (0.000 sec)
INSERT INTO t VALUES(1);
## wait 3 seconds
SELECT * FROM t;
ERROR 2006 (HY000): MySQL server has gone away
```

```
SET SESSION idle_transaction_timeout=2, SESSION idle_readonly_transaction_timeout=10;
BEGIN;
SELECT * FROM t;
Empty set (0.000 sec)
 ## wait 3 seconds
SELECT * FROM t;
Empty set (0.000 sec)
## wait 11 seconds
SELECT * FROM t;
ERROR 2006 (HY000): MySQL server has gone away
```

This page is licensed: CC BY-SA / Gnu FDL

UNLOCK TABLES

Syntax

```
UNLOCK TABLES
```

Description

`UNLOCK TABLES` explicitly releases any table locks held by the current session. See LOCK TABLES for more information.

In addition to releasing table locks acquired by the LOCK TABLES statement, the `UNLOCK TABLES` statement also releases the global read lock acquired by the `FLUSH TABLES WITH READ LOCK` statement. The `FLUSH TABLES WITH READ LOCK` statement is very useful for performing backups. See FLUSH for more information about `FLUSH TABLES WITH READ LOCK`.

This page is licensed: CC BY-SA / Gnu FDL

WAIT and NOWAIT

Extended syntax so that it is possible to set innodb_lock_wait_timeout and lock_wait_timeout for the following statements:

Syntax

```
ALTER TABLE tbl_name [WAIT n|NOWAIT] ...
CREATE ... INDEX ON tbl_name (index_col_name, ...) [WAIT n|NOWAIT] ...
DROP INDEX ... [WAIT n|NOWAIT]
DROP TABLE tbl_name [WAIT n|NOWAIT] ...
LOCK TABLE ... [WAIT n|NOWAIT]
OPTIMIZE TABLE tbl_name [WAIT n|NOWAIT]
RENAME TABLE tbl_name [WAIT n|NOWAIT] ...
SELECT ... FOR UPDATE [WAIT n|NOWAIT]
SELECT ... LOCK IN SHARE MODE [WAIT n|NOWAIT]
TRUNCATE TABLE tbl_name [WAIT n|NOWAIT]
```

Description

The lock wait timeout can be explicitly set in the statement by using either `WAIT n` (to set the wait in seconds) or `NOWAIT`, in which case the statement will immediately fail if the lock cannot be obtained. `WAIT 0` is equivalent to `NOWAIT`.

See Also

- Query Limits and Timeouts
- ALTER TABLE
- CREATE INDEX
- DROP INDEX
- DROP TABLE
- LOCK TABLES and UNLOCK TABLES
- OPTIMIZE TABLE
- RENAME TABLE
- SELECT
- TRUNCATE TABLE

This page is licensed: CC BY-SA / Gnu FDL

XA Transactions

Overview

The MariaDB XA implementation is based on the X/Open CAE document Distributed Transaction Processing: The XA Specification. This document is published by The Open Group and available at c193.htm.

XA transactions are designed to allow distributed transactions, where a transaction manager (the application) controls a transaction which involves multiple resources. Such resources are usually DBMSs, but could be resources of any type. The whole set of required transactional operations is called a global transaction. Each subset of operations which involve a single resource is called a local transaction. XA used a 2-phases commit (2PC). With the first commit, the transaction manager tells each resource to prepare an effective commit, and waits for a confirm message. The changes are not still made effective at this point. If any of the resources encountered an error, the transaction manager will rollback the global transaction. If all resources communicate that the first commit is successful, the transaction manager can require a second commit, which makes the changes effective.

In MariaDB, XA transactions can only be used with storage engines that support them. At least InnoDB, SPIDER and MyRocks support them. XA transactions are always supported.

Like regular transactions, XA transactions create metadata locks on accessed tables.

XA transactions require REPEATABLE READ as a minimum isolation level. However, distributed transactions should always use SERIALIZABLE.

Trying to start more than one XA transaction at the same time produces a 1400 error (SQLSTATE 'XAE09'). The same error is produced when attempting to start an XA transaction while a regular transaction is in effect. Trying to start a regular transaction while an XA transaction is in effect produces a 1399 error (SQLSTATE 'XAE07').

The statements that cause an implicit COMMIT for regular transactions produce a 1400 error (SQLSTATE 'XAE09') if a XA transaction is in effect.

Internal XA vs External XA

XA transactions are an overloaded term in MariaDB. If a storage engine is XA-capable, it can mean one or both of these:

- It supports MariaDB's internal two-phase commit API. This is transparent to the user. Sometimes this is called "internal XA", since MariaDB's internal transaction coordinator log can handle coordinating these transactions.
- It supports XA transactions, with the `XA START`, `XA PREPARE`, `XA COMMIT`, etc. statements. Sometimes this is called "external XA", since it requires the use of an external transaction coordinator to use this feature properly.

Transaction Coordinator Log

If you have two or more XA-capable storage engines enabled, then a transaction coordinator log must be available.

There are currently two implementations of the transaction coordinator log:

- Binary log-based transaction coordinator log
- Memory-mapped file-based transaction coordinator log

If the binary log is enabled on a server, then the server will use the binary log-based transaction coordinator log. Otherwise, it will use the memory-mapped file-based transaction coordinator log.

See Transaction Coordinator Log for more information.

Syntax

```
XA {START|BEGIN} xid [JOIN|RESUME]

XA END xid [SUSPEND [FOR MIGRATE]]

XA PREPARE xid

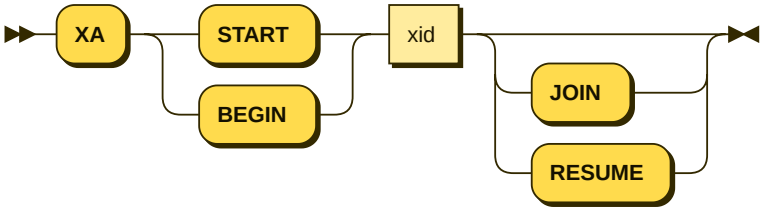
XA COMMIT xid [ONE PHASE]

XA ROLLBACK xid

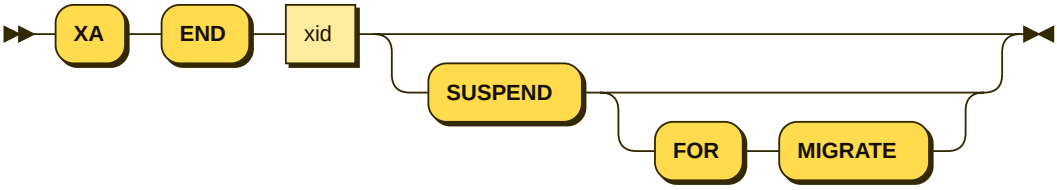
XA RECOVER [FORMAT=['RAW'|'SQL']]

xid: gtrid [, bqual [, formatID ]]
```

The BNF documents six XA statements together; each gets its own diagram.

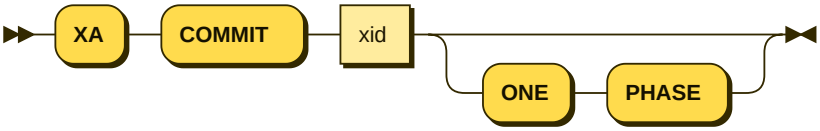


Railroad diagram of XA START / XA BEGIN



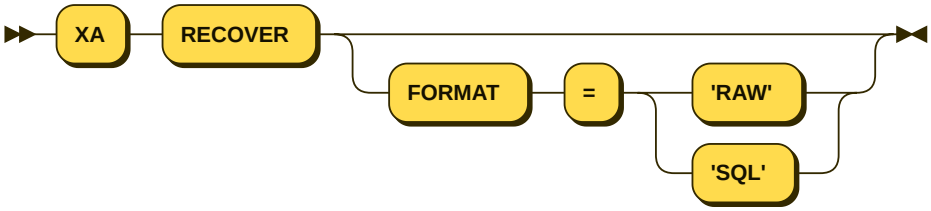
Railroad diagram of XA END

Railroad diagram of XA PREPARE

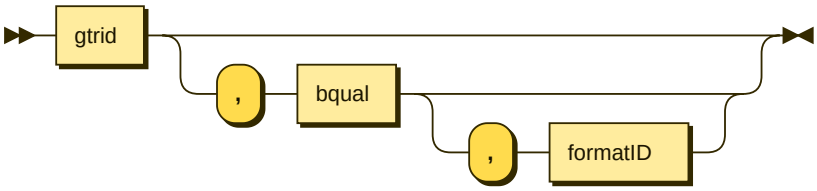


Railroad diagram of XA COMMIT

Railroad diagram of XA ROLLBACK



Railroad diagram of XA RECOVER



Railroad diagram of xid

The interface to XA transactions is a set of SQL statements starting with `XA`. Each statement changes a transaction's state, determining which actions it can perform. A transaction which does not exist is in the `NON-EXISTING` state.

When trying to execute an operation which is not allowed for the transaction's current state, an error is produced:

```
XA COMMIT 'test' ONE PHASE;
ERROR 1399 (XAE07): XAER_RMFAIL: The command cannot be executed when global transaction is in the  ACTIVE
state

XA COMMIT 'test2';
ERROR 1399 (XAE07): XAER_RMFAIL: The command cannot be executed when global transaction is in the  NON-
EXISTING state
```

XA START

```
XA {START|BEGIN} xid [JOIN|RESUME]
```

`XA START` (or `BEGIN`) starts a transaction and defines its `xid` (a transaction identifier). The new transaction will be in `ACTIVE` state.

The `xid` can have 3 components, though only the first one is mandatory. `gtrid` is a quoted string representing a global transaction identifier. `bqual` is a quoted string representing a local transaction identifier. `formatID` is an unsigned integer indicating the format used for the first two components; if not specified, defaults to 1. MariaDB does not interpret in any way these components, and only uses them to identify a transaction. `xid`s of transactions in effect must be unique.

Using the `JOIN` or `RESUME` keywords will currently cause an error to be returned.

```
XA START 'test' RESUME;
ERROR 1398 (XAE05): XAER_INVAL: Invalid arguments (or unsupported command)
XA START 'test' JOIN;
ERROR 1398 (XAE05): XAER_INVAL: Invalid arguments (or unsupported command)
```

An exception to this is that `XA START xid RESUME` will resume the transaction if the `xid` is the same as the previous `xid` used in `XA END xid`. This simply undoes the `XA END` and moves it from the `IDLE` state back into `ACTIVE`.

```
XA START 'test';
INSERT INTO t VALUES (1,2);
XA END 'test';
XA START 'test' RESUME; -- This reverts the XA END and continues the transaction
INSERT INTO t VALUES (3,4);
XA END 'test';
XA PREPARE 'test';
XA COMMIT 'test';
```

XA END

```
XA END xid [SUSPEND [FOR MIGRATE]]
```

`XA END` declares that the specified `ACTIVE` transaction is finished and it changes its state to `IDLE`. `SUSPEND [FOR MIGRATE]` has no effect.

XA PREPARE

```
XA PREPARE xid
```

`XA PREPARE` prepares an `IDLE` transaction for commit, changing its state to `PREPARED`. Prepared transactions are stored persistently and will survive disconnects and server crashes, and must be explicitly committed or rolled back.

> **CURRENT**
>
> Prepared transactions were automatically rolled back on client disconnect, but were not rolled back if the server was crashed or killed. This violated XA guarantees and could have caused inconsistent data, if the transaction in question was already irrevocably committed in another XA participant.
>
> **< 10.5**
>
> Prepared transactions are automatically rolled back on client disconnect, but are not rolled back if the server was crashed or killed. This violated XA guarantees and can cause inconsistent data, if the transaction in question was already irrevocably committed in another XA participant.

XA COMMIT

```
XA COMMIT xid [ONE PHASE]
```

`XA COMMIT` definitely commits and terminates a transaction which has already been `PREPARED`. If the `ONE PHASE` clause is specified, this statements performs a 1-phase commit on an `IDLE` transaction.

XA ROLLBACK

```
XA ROLLBACK xid
```

`XA ROLLBACK` rolls back and terminates an `IDLE` or `PREPARED` transaction.

XA RECOVER

```
XA RECOVER [FORMAT=['RAW'|'SQL']]
```

The `XA RECOVER` statement shows information about all transactions which are in the `PREPARED` state. It does not matter which connection created the transaction: if it has been `PREPARED`, it appears. But this does not mean that a connection can commit or rollback a transaction which was started by another connection. Note that transactions using a 1-phase commit are never in the `PREPARED` state, so they cannot be shown by `XA RECOVER`.

`XA RECOVER` produces four columns:

```
XA RECOVER;
+----------+--------------+--------------+------+
| formatID | gtrid_length | bqual_length | data |
+----------+--------------+--------------+------+
|        1 |            4 |            0 | test |
+----------+--------------+--------------+------+
```

You can use `XA RECOVER FORMAT='SQL'` to get the data in a human readable form that can be directly copy-pasted into `XA COMMIT` or `XA ROLLBACK`. This is particularly useful for binary `xid` generated by some transaction coordinators.

`formatID` is the `formatID` part of `xid`.

`data` are the `gtrid` and `bqual` parts of `xid`, concatenated.

`gtrid_length` and `bqual_length` are the lengths of `gtrid` and `bqual`, respectively.

Examples

2-phases commit:

```
XA START 'test';

INSERT INTO t VALUES (1,2);

XA END 'test';

XA PREPARE 'test';

XA COMMIT 'test';
```

1-phase commit:

```
XA START 'test';

INSERT INTO t VALUES (1,2);

XA END 'test';

XA COMMIT 'test' ONE PHASE;
```

Human-readable:

```
xa start '12\r34\t67\v78', 'abc\ndef', 3;

INSERT t1 values (40);

xa end '12\r34\t67\v78', 'abc\ndef', 3;

xa prepare '12\r34\t67\v78', 'abc\ndef', 3;

xa recover format='RAW';
+----------+--------------+--------------+--------------------+
| formatID | gtrid_length | bqual_length | data               |
+----------+--------------+--------------+--------------------+
34      67v78abc       11 |            7 | 12
def |
+----------+--------------+--------------+--------------------+

xa recover format='SQL';
+----------+--------------+--------------+-----------------------------------------------+
| formatID | gtrid_length | bqual_length | data                                          |
+----------+--------------+--------------+-----------------------------------------------+
|        3 |           11 |            7 | X'31320d3334093637763738',X'6162630a646566',3 |
+----------+--------------+--------------+-----------------------------------------------+

xa rollback X'31320d3334093637763738',X'6162630a646566',3;
```

Known Issues

MariaDB Galera Cluster

MariaDB Galera Cluster does not support XA transactions.

However, MariaDB Galera Cluster builds include a built-in plugin called `wsrep`. Consequently, these MariaDB Galera Cluster builds have multiple XA-capable storage engines by default, even if the only “real” storage engine that supports external XA transactions enabled on these builds by default is InnoDB. Therefore, when using one these builds MariaDB would be forced to use a transaction coordinator log by default, which could have performance implications.

See Transaction Coordinator Log Overview: MariaDB Galera Cluster for more information.

Incompatibility with XA behavior

> **Warning:**
>
> From MariaDB 10.5, `XA PREPARE` persists the XA transaction following the XA Specification. If an existing application relies on the previous behavior, upgrading to 10.5 or later can leave XA transactions in the `PREPARE`d state indefinitely after disconnect, causing such applications to no longer function correctly.

As a work-around, the variable legacy_xa_rollback_at_disconnect can be set to TRUE to re-enable the old behavior and roll back XA transactions in the `PREPARE`d state at disconnect. This is non-standard behaviour, and is not recommended for new applications. If rollback-at-disconnect is desired, it is better to use a normal (non-XA) transaction.

This page is licensed: CC BY-SA / Gnu FDL

SQL Functions

See Functions & Operators.

Comprehensive reference of all SQL functions and operators. This index lists built-in functions for string manipulation, math, date/time, and more.

See Aggregate Functions.

Perform calculations on multiple rows to return a single value. Includes standard SQL functions like SUM, AVG, COUNT, MIN, and MAX, often used with GROUP BY.

See Control Flow Functions.

Learn about control flow functions in MariaDB Server. This section details SQL functions like IF, CASE, and NULLIF, which enable conditional logic within your queries and stored routines.

See Date & Time Functions.

Complete MariaDB date and time functions guide. Complete reference for formatting, calculations, conversions, time zones, and operations for production use.

See Numeric Functions.

Learn about numeric functions in MariaDB Server. This section details SQL functions for performing mathematical calculations, rounding, and manipulating numeric values in your queries.

See Pseudo Columns.

MariaDB has pseudo columns that can be used for different purposes.

See Secondary Functions.

Explore secondary functions in MariaDB Server. This section details various SQL functions that support specialized operations, enhancing data manipulation and analysis beyond core functionalities.

See Special Functions.

Explore special functions in MariaDB Server. This section details unique SQL functions that provide specialized capabilities, often related to server internals, diagnostics, or specific data handling.

See String Functions.

Learn about string functions in MariaDB Server. This section details SQL functions for manipulating, searching, and formatting text strings, essential for data cleansing and presentation.

See Vector Functions.

Explore vector functions. This section details SQL functions for manipulating and querying vector data types, enabling efficient similarity search and AI/ML applications within your database.

This page is licensed: CC BY-SA / Gnu FDL

Functions & Operators

Name	Description	Added
+	Addition operator	
/	Division operator	
*	Multiplication operator	
%	Modulo operator. Returns the remainder of N divided by M	
-	Subtraction operator	
!=	Not equals	
<	Less than	
<=	Less than or equal	
<=>	NULL-safe equal	
=	Equal	
>	Greater than	
>=	Greater than or equal	
&	Bitwise AND	
<<	Shift left	
>>	Shift right	
^	Bitwise XOR	
!	Logical NOT	
&&	Logical AND	
XOR	Logical XOR	
|	Bitwise OR	
:=	Assignment operator	
=	Assignment and comparison operator	
~	Bitwise NOT	
ABS	Returns an absolute value	
ACOS	Returns an arc cosine	
ADD_MONTHS	Add months to a date	
ADDDATE	Add days or another interval to a date	
ADDTIME	Adds a time to a time or datetime	

Name	Description	Added
AES_DECRYPT	Decryption data encrypted with AES_ENCRYPT	
AES_ENCRYPT	Encrypts a string with the AES algorithm	
AREA	Synonym for ST_AREA	
AsBinary	Synonym for ST_AsBinary	
ASCII	Numeric ASCII value of leftmost character	
ASIN	Returns the arc sine	
AsText	Synonym for ST_AsText	
AsWKB	Synonym for ST_AsBinary	
AsWKT	Synonym for ST_AsText	
ATAN	Returns the arc tangent	
ATAN2	Returns the arc tangent of two variables	
AVG	Returns the average value	
BENCHMARK	Executes an expression repeatedly	
BETWEEN AND	True if expression between two values	
BIN	Returns binary value	
BINARY OPERATOR	Casts to a binary string	
BINLOG_GTID_POS	Returns a string representation of the corresponding GTID position	
BIT_AND	Bitwise AND	
BIT_COUNT	Returns the number of set bits	
BIT_LENGTH	Returns the length of a string in bits	
BIT_OR	Bitwise OR	
BIT_XOR	Bitwise XOR	
BOUNDARY	Synonym for ST_BOUNDARY	
BUFFER	Synonym for ST_BUFFER	
CASE	Returns the result where value=compare_value or for the first condition that is true	
CAST	Casts a value of one type to another type	
CEIL	Synonym for CEILING()	
CEILING	Returns the smallest integer not less than X	
CENTROID	Synonym for ST_CENTROID	
CHAR Function	Returns string based on the integer values for the individual characters	
CHARACTER_LENGTH	Synonym for CHAR_LENGTH()	
CHAR_LENGTH	Length of the string in characters	
CHARSET	Returns the character set	
CHR	Returns a string consisting of the character given by the code values of the integer	
COALESCE	Returns the first non-NULL parameter	
COERCIBILITY	Returns the collation coercibility value	
COLLATION	Collation of the string argument	

Name	Description	Added
COLUMN_ADD	Adds or updates dynamic columns	
COLUMN_CHECK	Checks if a dynamic column blob is valid	
COLUMN_CREATE	Returns a dynamic columns blob	
COLUMN_DELETE	Deletes a dynamic column	
COLUMN_EXISTS	Checks is a column exists	
COLUMN_GET	Gets a dynamic column value by name	
COLUMN_JSON	Returns a JSON representation of dynamic column blob data	
COLUMN_LIST	Returns comma-separated list	
COMPRESS	Returns a binary, compressed string	
CONCAT	Returns concatenated string	
CONCAT_WS	Concatenate with separator	
CONNECTION_ID	Connection thread ID	
CONTAINS	Whether one geometry contains another	
CONVERT	Convert a value from one type to another type	
CONV	Converts numbers between different number bases	
CONVERT_TZ	Converts a datetime from on time zone to another	
CONVEXHULL	Synonym for ST_CONVEXHULL	
COS	Returns the cosine	
COT	Returns the cotangent	
COUNT	Returns count of non-null values	
COUNT DISTINCT	Returns count of number of different non-NULL values	
CRC32	Computes a cyclic redundancy check value	
CRC32C	Computes a cyclic redundancy check value	MariaDB 10.8
CROSSES	Whether two geometries spatially cross	
CUME_DIST	Window function that returns the cumulative distribution of a given row	
CURDATE	Returns the current date	
CURRENT_DATE	Synonym for CURDATE()	
CURRENT_ROLE	Current role name	
CURRENT_TIME	Synonym for CURTIME()	
CURRENT_TIMESTAMP	Synonym for NOW()	
CURRENT_USER	Username/host that authenticated the current client	
CURTIME	Returns the current time	
DATABASE	Current default database	
DATE FUNCTION	Extracts the date portion of a datetime	
DATEDIFF	Difference in days between two date/time values	
DATE_ADD	Date arithmetic - addition	
DATE_FORMAT	Formats the date value according to the format string	
DATE_SUB	Date arithmetic - subtraction	
DAY	Synonym for DAYOFMONTH()	

Name	Description	Added
DAYNAME	Return the name of the weekday	
DAYOFMONTH	Returns the day of the month	
DAYOFWEEK	Returns the day of the week index	
DAYOFYEAR	Returns the day of the year	
DECODE	Decrypts a string encoded with ENCODE()	
DECODE_HISTOGRAM	Returns comma separated numerics corresponding to a probability distribution represented by a histogram	
DEFAULT	Returns column default	
DEGREES	Converts from radians to degrees	
DENSE_RANK	Rank of a given row with identical values receiving the same result, no skipping	
DES_DECRYPT	Decrypts a string encrypted with DES_ENCRYPT()	
DES_ENCRYPT	Encrypts a string using the Triple-DES algorithm	
DIMENSION	Synonym for ST_DIMENSION	
DISJOINT	Whether the two elements do not intersect	
DIV	Integer division	
ELT	Returns the N'th element from a set of strings	
ENCODE	Encrypts a string	
ENCRYPT	Encrypts a string with Unix crypt()	
ENDPOINT	Synonym for ST_ENDPOINT	
ENVELOPE	Synonym for ST_ENVELOPE	
EQUALS	Indicates whether two geometries are spatially equal	
EXP	e raised to the power of the argument	
EXPORT_SET	Returns an on string for every bit set, an off string for every bit not set	
ExteriorRing	Synonym for ST_ExteriorRing	
EXTRACT	Extracts a portion of the date	
EXTRACTVALUE	Returns the text of the first text node matched by the XPath expression	
FIELD	Returns the index position of a string in a list	
FIND_IN_SET	Returns the position of a string in a set of strings	
FLOOR	Largest integer value not greater than the argument	
FORMAT	Formats a number	
FORMAT_BYTES	Given a byte count, returns a string consisting of a value and the units in a human-readable format.	MariaDB 11.8
FORMAT_PICO_TIME	Given a time in picoseconds, returns a human-readable time value and unit indicator	MariaDB 11.0.2
FOUND_ROWS	Number of (potentially) returned rows	
FROM_BASE64	Given a base-64 encoded string, returns the decoded result as a binary string	
FROM_DAYS	Returns a date given a day	
FROM_UNIXTIME	Returns a datetime from a Unix timestamp	

Name	Description	Added
GeomCollFromText	Synonym for ST_GeomCollFromText	
GeomCollFromWKB	Synonym for ST_GeomCollFromWKB	
GEOMETRYCOLLECTION	Constructs a WKB GeometryCollection	
GeometryCollectionFromText	Synonym for ST_GeomCollFromText	
GeometryCollectionFromWKB	Synonym for ST_GeomCollFromWKB	
GeometryFromText	Synonym for ST_GeomFromText	
GeometryFromWKB	Synonym for ST_GeomFromWKB	
GeomFromText	Synonym for ST_GeomFromText	
GeomFromWKB	Synonym for ST_GeomFromWKB	
GeometryN	Synonym for ST_GeometryN	
GeometryType	Synonym for ST_GeometryType	
GET_FORMAT	Returns a format string	
GET_LOCK	Obtain LOCK	
GLENGTH	Length of a LineString value	
GREATEST	Returns the largest argument	
GROUP_CONCAT	Returns string with concatenated values from a group	
HEX	Returns hexadecimal value	
HOUR	Returns the hour	
IF	If expr1 is TRUE, returns expr2; otherwise returns expr3	
IFNULL	Check whether an expression is NULL	
IN	True if expression equals any of the values in the list	
INTERVAL	Index of the argument that is less than the first argument	
INET6_ATON	Given an IPv6 or IPv4 network address, returns a VARBINARY numeric value	
INET6_NTOA	Given an IPv6 or IPv4 network address, returns the address as a nonbinary string	
INET_ATON	Returns numeric value of IPv4 address	
INET_NTOA	Returns dotted-quad representation of IPv4 address	
INSERT Function	Replaces a part of a string with another string	
INSTR	Returns the position of a string within a string	
InteriorRingN	Synonym for ST_InteriorRingN	
INTERSECTS	Indicates whether two geometries spatially intersect	
IS	Tests whether a boolean is TRUE, FALSE, or UNKNOWN	
IsClosed	Synonym for ST_IsClosed	
IsEmpty	Synonym for ST_IsEmpty	
IS_FREE_LOCK	Checks whether lock is free to use	
IS_IPV4	Whether or not an expression is a valid IPv4 address	
IS_IPV4_COMPAT	Whether or not an IPv6 address is IPv4-compatible	

Name	Description	Added
IS_IPV4_MAPPED	Whether an IPv6 address is a valid IPv4-mapped address	
IS_IPV6	Whether or not an expression is a valid IPv6 address	
IS NOT	Tests whether a boolean value is not TRUE, FALSE, or UNKNOWN	
IS NOT NULL	Tests whether a value is not NULL	
IS NULL	Tests whether a value is NULL	
ISNULL	Checks if an expression is NULL	
IsRing	Synonym for ST_IsRing	
IsSimple	Synonym for ST_IsSimple	
IS_USED_LOCK	Check if lock is in use	
JSON_ARRAY	Returns a JSON array containing the listed values	
JSON_ARRAYAGG	Returns a JSON array containing an element for each value in a given set of JSON or SQL values.	
JSON_ARRAY_INTERSECT		MariaDB 11.2.0
JSON_ARRAY_APPEND	Appends values to the end of the given arrays within a JSON document	
JSON_ARRAY_INSERT	Inserts a value into a JSON document	
JSON_COMPACT	Removes all unnecessary spaces so the json document is as short as possible	
JSON_CONTAINS	Whether a value is found in a given JSON document or at a specified path within the document	
JSON_CONTAINS_PATH	Indicates whether the given JSON document contains data at the specified path or paths	
JSON_DEPTH	Maximum depth of a JSON document	
JSON_DETAILED	Represents JSON in the most understandable way emphasizing nested structures	
JSON_EQUALS	Check for equality between JSON objects.	MariaDB 10.7
JSON_EXISTS	Determines whether a specified JSON value exists in the given data	
JSON_EXTRACT	Extracts data from a JSON document.	
JSON_INSERT	Inserts data into a JSON document	
JSON_KEY_VALUE	Extracts key/value pairs from a JSON object	MariaDB 11.2
JSON_KEYS	Returns keys from top-level value of a JSON object or top-level keys from the path	
JSON_LENGTH	Returns the length of a JSON document, or the length of a value within the document	
JSON_LOOSE	Adds spaces to a JSON document to make it look more readable	
JSON_MERGE	Merges the given JSON documents	
JSON_MERGE_PATCH	RFC 7396-compliant merge of the given JSON documents	
JSON_MERGE_PRESERVE	Synonym for JSON_MERGE_PATCH.	

Name	Description	Added
JSON_NORMALIZE	Recursively sorts keys and removes spaces, allowing comparison of json documents for equality	
JSON_OBJECT	Returns a JSON object containing the given key/value pairs	
JSON_OBJECT_FILTER_KEYS		MariaDB 11.2.0
JSON_OBJECT_TO_ARRAY		MariaDB 11.2.0
JSON_OBJECTAGG	Returns a JSON object containing key-value pairs	
JSON_OVERLAPS	Compares two json documents for overlaps	MariaDB 10.9
JSON_PRETTY	Alias for json_detailed	MariaDB 10.10.3, MariaDB 10.9.5, MariaDB 10.8.7, MariaDB 10.7.8, MariaDB 10.6.12
JSON_QUERY	Given a JSON document, returns an object or array specified by the path	
JSON_QUOTE	Quotes a string as a JSON value	
JSON_REMOVE	Removes data from a JSON document	
JSON_REPLACE	Replaces existing values in a JSON document	
JSON_SCHEMA_VALID	Validates a JSON schema	MariaDB 11.1.0
JSON_SEARCH	Returns the path to the given string within a JSON document	
JSON_SET	Updates or inserts data into a JSON document	
JSON_TABLE	Returns a representation of a JSON document as a relational table	MariaDB 10.6
JSON_TYPE	Returns the type of a JSON value	
JSON_UNQUOTE	Unquotes a JSON value, returning a string	
JSON_VALID	Whether a value is a valid JSON document or not	
JSON_VALUE	Given a JSON document, returns the specified scalar	
KDF	Key derivation function	MariaDB 11.3.0
LAST_DAY	Returns the last day of the month	
LAST_INSERT_ID	Last inserted autoinc value	
LAST_VALUE	Returns the last value in a list	
LASTVAL	Get last value generated from a sequence	
LCASE	Synonym for [LOWER()	
LEAST	Returns the smallest argument	
LEFT	Returns the leftmost characters from a string	
LENGTH	Length of the string in bytes	
LIKE	Whether expression matches a pattern	
LineFromText	Synonym for ST_LineFromText	
LineFromWKB	Synonym for ST_LineFromWKB	
LINESTRING	Constructs a WKB LineString value from a number of WKB Point arguments	
LineStringFromText	Synonym for ST_LineFromText	
LineStringFromWKB	Synonym for ST_LineFromWKB	
LN	Returns natural logarithm	

Name	Description	Added
LOAD_FILE	Returns file contents as a string	
LOCALTIME	Synonym for NOW()	
LOCALTIMESTAMP	Synonym for NOW()	
LOCATE	Returns the position of a substring in a string	
LOG	Returns the natural logarithm	
LOG10	Returns the base-10 logarithm	
LOG2	Returns the base-2 logarithm	
LOWER	Returns a string with all characters changed to lowercase	
LPAD	Returns the string left-padded with another string to a given length	
LTRIM	Returns the string with leading space characters removed	
MAKE_SET	Make a set of strings that matches a bitmask	
MAKEDATE	Returns a date given a year and day	
MAKETIME	Returns a time	
MASTER_GTID_WAIT	Wait until slave reaches the GTID position	
MASTER_POS_WAIT	Blocks until the slave has applied all specified updates	
MATCH AGAINST	Perform a fulltext search on a fulltext index	
MAX	Returns the maximum value	
MBRContains	Whether one Minimum Bounding Rectangle contains another.	
MBRCoveredBy	Whether one Minimum Bounding Rectangle is covered by another.	MariaDB 11.8
MBRDisjoint	Whether the Minimum Bounding Rectangles of two geometries are disjoint	
MBREqual	Whether the Minimum Bounding Rectangles of two geometries are the same.	
MBREquals	Synonym for MBREqual.	
MBRIntersects	Indicates whether the Minimum Bounding Rectangles of the two geometries intersect	
MBROverlaps	Whether the Minimum Bounding Rectangles of two geometries overlap	
MBRTouches	Whether the Minimum Bounding Rectangles of two geometries touch.	
MBRWithin	Indicates whether one Minimum Bounding Rectangle is within another	
MD5	MD5 checksum	
MEDIAN	Window function that returns the median value of a range of values	
MICROSECOND	Returns microseconds from a date or datetime	
MID	Synonym for SUBSTRING(str,pos,len)	
MIN	Returns the minimum value	
MINUTE	Returns a minute from 0 to 59	
MLineFromText	Constructs MULTILINESTRING using its WKT representation and SRID	

Name	Description	Added
MLineFromWKB	Constructs a MULTILINESTRING	
MOD	Modulo operation. Remainder of N divided by M	
MONTH	Returns a month from 1 to 12	
MONTHNAME	Returns the full name of the month	
MPointFromText	Constructs a MULTIPOINT value using its WKT and SRID	
MPointFromWKB	Constructs a MULTIPOINT value using its WKB representation and SRID	
MPolyFromText	Constructs a MULTIPOLYGON value	
MPolyFromWKB	Constructs a MULTIPOLYGON value using its WKB representation and SRID	
MULTILINESTRING	Constructs a MultiLineString value	
MultiLineStringFromText	Synonym for MLineFromText	
MultiLineStringFromWKB	A synonym for MLineFromWKB	
MULTIPOINT	Constructs a WKB MultiPoint value	
MultiPointFromText	Synonym for MPointFromText	
MultiPointFromWKB	Synonym for MPointFromWKB	
MULTIPOLYGON	Constructs a WKB MultiPolygon	
MultiPolygonFromText	Synonym for MPolyFromText	
MultiPolygonFromWKB	Synonym for MPolyFromWKB	
NAME_CONST	Returns the given value	
NATURAL_SORT_KEY	Sorting that is more similar to natural human sorting	
NOT LIKE	Same as NOT(expr LIKE pat [ESCAPE ‘escape_char’])	
NOT REGEXP	Same as NOT (expr REGEXP pat)	
NULLIF	Returns NULL if expr1 = expr2	
NEXTVAL	Generate next value for sequence	
NOT BETWEEN	Same as NOT (expr BETWEEN min AND max)	
NOT IN	Same as NOT (expr IN (value,…))	
NOW	Returns the current date and time	
NTILE	Returns an integer indicating which group a given row falls into	
NumGeometries	Synonym for ST_NumGeometries	
NumInteriorRings	Synonym for NumInteriorRings	
NumPoints	Synonym for ST_NumPoints	
OCT	Returns octal value	
OCTET_LENGTH	Synonym for LENGTH()	
OLD_PASSWORD	Pre MySQL 4.1 password implementation	
ORD	Return ASCII or character code	
OVERLAPS	Indicates whether two elements spatially overlap	
PASSWORD	Calculates a password string	
PERCENT_RANK	Window function that returns the relative percent rank of a given row	

Name	Description	Added
PERCENTILE_CONT	Returns a value which corresponds to the given fraction in the sort order.	
PERCENTILE_DISC	Returns the first value in the set whose ordered position is the same or more than the specified fraction.	
PERIOD_ADD	Add months to a period	
PERIOD_DIFF	Number of months between two periods	
PI	Returns the value of π (pi)	
POINT	Constructs a WKB Point	
PointFromText	Synonym for ST_PointFromText	
PointFromWKB	Synonym for PointFromWKB	
PointN	Synonym for PointN	
PointOnSurface	Synonym for ST_PointOnSurface	
POLYGON	Constructs a WKB Polygon value from a number of WKB LineString arguments	
PolyFromText	Synonym for ST_PolyFromText	
PolyFromWKB	Synonym for ST_PolyFromWKB	
PolygonFromText	Synonym for ST_PolyFromText	
PolygonFromWKB	Synonym for ST_PolyFromWKB	
POSITION	Returns the position of a substring in a string	
POW	Returns X raised to the power of Y	
POWER	Synonym for POW()	
QUARTER	Returns year quarter from 1 to 4	
QUOTE	Returns quoted, properly escaped string	
RADIANS	Converts from degrees to radians	
RAND	Random floating-point value	
RANK	Rank of a given row with identical values receiving the same result	
REGEXP	Performs pattern matching	
REGEXP_INSTR	Position of the first appearance of a regex	
REGEXP_REPLACE	Replaces all occurrences of a pattern	
REGEXP_SUBSTR	Returns the matching part of a string	
RELEASE_LOCK	Releases lock obtained with GET_LOCK()	
REPEAT Function	Returns a string repeated a number of times	
REPLACE Function	Replace occurrences of a string	
REVERSE	Reverses the order of a string	
RIGHT	Returns the rightmost N characters from a string	
RLIKE	Synonym for REGEXP()	
RPAD	Returns the string right-padded with another string to a given length	
ROUND	Rounds a number	
ROW_COUNT	Number of rows affected by previous statement	

Name	Description	Added
ROW_NUMBER	Row number of a given row with identical values receiving a different result	
RTRIM	Returns the string with trailing space characters removed	
SCHEMA	Synonym for DATABASE()	
SECOND	Returns the second of a time	
SEC_TO_TIME	Converts a second to a time	
SETVAL	Set the next value to be returned by a sequence	
SESSION_USER	Synonym for USER()	
sha	Synonym for SHA1()	
SHA1	Calculates an SHA-1 checksum	
SHA2	Calculates an SHA-2 checksum	
SIGN	Returns 1, 0 or -1	
SIN	Returns the sine	
SLEEP	Pauses for the given number of seconds	
SOUNDEX	Returns a string based on how the string sounds	
SOUNDS LIKE	SOUNDEX(expr1) = SOUNDEX(expr2)	
SPACE	Returns a string of space characters	
SPIDER_BG_DIRECT_SQL	Background SQL execution	
SPIDER_COPY_TABLES	Copy table data	
SPIDER_DIRECT_SQL	Execute SQL on the remote server	
SPIDER_FLUSH_TABLE_MON_CACHE	Refreshing Spider monitoring server information	
SQRT	Square root	
SRID	Synonym for ST_SRID	
ST_AREA	Area of a Polygon	
ST_AsBinary	Converts a value to its WKB representation	
ST_AsGeoJson	Returns a GeoJSON element from a given geometry.	
ST_AsText	Converts a value to its WKT-Definition	
ST_AsWKB	Synonym for ST_AsBinary	
ST_ASWKT	Synonym for ST_ASTEXT()	
ST_BOUNDARY	Returns a geometry that is the closure of a combinatorial boundary	
ST_BUFFER	A new geometry with a buffer added to the original geometry	
ST_CENTROID	The mathematical centroid (geometric center) for a MultiPolygon	
ST_Collect	Returns the aggregation of the distinct geometry arguments	MariaDB 11.8
ST_CONTAINS	Whether one geometry is contained by another	
ST_CONVEXHULL	The minimum convex geometry enclosing all geometries within the set	
ST_CROSSES	Whether two geometries spatially cross	
ST_DIFFERENCE	Point set difference	

Name	Description	Added
ST_DIMENSION	Inherent dimension of a geometry value	
ST_DISJOINT	Whether one geometry is spatially disjoint from another	
ST_DISTANCE	The distance between two geometries	
ST_DISTANCE_SPHERE	The spherical distance between two geometries	
ST_ENDPOINT	Returns the endpoint of a LineString	
ST_ENVELOPE	Returns the Minimum Bounding Rectangle for a geometry value	
ST_EQUALS	Whether two geometries are spatoially equal	
ST_ExteriorRing	Returns the exterior ring of a Polygon as a LineString	
ST_GeoHash	Returns a geohash.	MariaDB 11.8
ST_GeomCollFromText	Constructs a GEOMETRYCOLLECTION value	
ST_GeomCollFromWKB	Constructs a GEOMETRYCOLLECTION value from a WKB	
ST_GeometryCollectionFromText	Synonym for ST_GeomCollFromText	
ST_GeometryCollectionFromWKB	Synonym for ST_GeomCollFromWKB	
ST_GeometryFromText	Synonym for ST_GeomFromText	
ST_GeometryFromWKB	Synonym for ST_GeomFromWKB	
ST_GEOMETRYN	Returns the N-th geometry in a GeometryCollection	
ST_GEOMETRYTYPE	Returns name of the geometry type of which a given geometry instance is a member	
ST_GeomFromGeoJSON	Returns a geometry object from a GeoJSON input	
ST_GeomFromText	Constructs a geometry value using its WKT and SRID	
ST_GeomFromWKB	Constructs a geometry value using its WKB representation and SRID	
ST_InteriorRingN	Returns the N-th interior ring for a Polygon	
ST_INTERSECTION	The intersection, or shared portion, of two geometries	
ST_INTERSECTS	Whether two geometries spatially intersect	
ST_ISCLOSED	Returns true if a given LINESTRING's start and end points are the same	
ST_ISEMPTY	Indicated validity of geometry value	
ST_IsRing	Returns true if a given LINESTRING is both ST_IsClosed and ST_IsSimple	
ST_IsSimple	Returns true if the given Geometry has no anomalous geometric points	
ST_IsValid	Returns 1 if the argument is geometrically valid, 0 if not.	MariaDB 11.8
ST_LatFromGeoHash	Returns a latitude from the given geohash.	MariaDB 11.8
ST_LongFromGeoHash	Returns a longitude from the given geohash.	MariaDB 11.8
ST_LENGTH	Length of a LineString value	
ST_LineFromText	Creates a linestring value	
ST_LineFromWKB	Constructs a LINESTRING using its WKB and SRID	
ST_LineStringFromText	Synonym for ST_LineFromText	

Name	Description	Added
ST_LineStringFromWKB	Synonym for ST_LineFromWKB	
ST_MLineFromText	Synonym for ST_GeomFromText	
ST_MLineFromWKB	Synonym for ST_GeomFromWKB	
ST_MPointFromText	Synonym for ST_GeomFromText	
ST_MPointFromWKB	Synonym for ST_GeomFromWKB	
ST_MPolyFromText	Synonym for ST_GeomFromText	
ST_MPolyFromWKB	Synonym for ST_GeomFromWKB	
ST_MultiLineStringFromText	Synonym for ST_GeomFromText	
ST_MultiLineStringFromWKB	Synonym for ST_GeomFromWKB	
ST_MultiPointFromText	Synonym for ST_GeomFromText	
ST_MultiPolygonFromText	Synonym for ST_GeomFromText	
ST_MultiPolygonFromWKB	Synonym for ST_GeomFromWKB	
ST_MultiPointFromWKB	Synonym for ST_GeomFromWKB	
ST_NUMGEOMETRIES	Number of geometries in a GeometryCollection	
ST_NumInteriorRings	Number of interior rings in a Polygon	
ST_NUMPOINTS	Returns the number of Point objects in a LineString	
ST_OVERLAPS	Whether two geometries overlap	
ST_PointFromGeoHash	Returns a point from the given geohash.	MariaDB 11.8
ST_PointFromText	Constructs a POINT value	
ST_PointFromWKB	Constructs POINT using its WKB and SRID	
ST_POINTN	Returns the N-th Point in the LineString	
ST_POINTONSURFACE	Returns a POINT guaranteed to intersect a surface	
ST_PolyFromText	Constructs a POLYGON value	
ST_PolyFromWKB	Constructs POLYGON value using its WKB representation and SRID	
ST_PolygonFromText	Synonym for ST_PolyFromText	
ST_PolygonFromWKB	Synonym for ST_PolyFromWKB	
ST_RELATE	Returns true if two geometries are related	
ST_Simplify	Applies the Ramer–Douglas–Peucker algorithm to provide generalized simplifications.	MariaDB 11.8
ST_SRID	Returns a Spatial Reference System ID	
ST_STARTPOINT	Returns the start point of a LineString	
ST_SYMDIFFERENCE	Portions of two geometries that don't intersect	
ST_TOUCHES	Whether one geometry g1 spatially touches another	
ST_UNION	Union of two geometries	
ST_Validate	Whether a geometry is compliant with the WKB format and SRID syntax, and is geometrically valid.	MariaDB 11.8
ST_WITHIN	Whether one geometry is within another	
ST_X	X-coordinate value for a point	

Name	Description	Added
ST_Y	Y-coordinate for a point	
STARTPOINT	Synonym for ST_StartPoint	
STD	Population standard deviation	
STDDEV	Population standard deviation	
STDDEV_POP	Returns the population standard deviation	
STDDEV_SAMP	Standard deviation	
STR_TO_DATE	Converts a string to date	
STRCMP	Compares two strings in sort order	
SUBDATE	Subtract a date unit or number of days	
SUBSTR	Returns a substring from string starting at a given position	
SUBSTRING	Returns a substring from string starting at a given position	
SUBSTRING_INDEX	Returns the substring from string before count occurrences of a delimiter	
SUBTIME	Subtracts a time from a date/time	
SUM	Sum total	
SYS.EXTRACT_SCHEMA_FROM_FILE_NAME	Given a file path, returns the schema (database) name	MariaDB 10.6
SYS.EXTRACT_TABLE_FROM_FILE_NAME	Given a file path, returns the table name	MariaDB 10.6
SYS.FORMAT_BYTES	Returns a string consisting of a value and the units in a human-readable format	MariaDB 10.6
SYS.FORMAT_PATH	Returns a modified path after replacing subpaths matching the values of various system variables with the variable name	MariaDB 10.6
SYS.FORMAT_STATEMENT	Returns a reduced length string	MariaDB 10.6
SYS.FORMAT_TIME	Returns a human-readable time value and unit indicator	MariaDB 10.6
SYS.LIST_ADD	Adds a value to a given list	MariaDB 10.6
SYS.LIST_DROP	Drops a value from a given list	MariaDB 10.6
SYS.PS_IS_ACCOUNT_ENABLED	Whether Performance Schema instrumentation for the given account is enabled	MariaDB 10.6
SYS.PS_IS_CONSUMER_ENABLED	Whether Performance Schema instrumentation for the given consumer is enabled	MariaDB 10.6
SYS.PS_IS_INSTRUMENT_DEFAULT_ENABLED	Whether a given Performance Schema instrument is enabled by default	MariaDB 10.6
SYS.PS_IS_INSTRUMENT_DEFAULT_TIMED	Returns whether a given Performance Schema instrument is timed by default	MariaDB 10.6
SYS.PS_IS_THREAD_INSTRUMENTED	Returns whether or not Performance Schema instrumentation for the given connection_id is enabled	MariaDB 10.6
SYS.PS_THREAD_ACCOUNT	Returns the account (username@hostname) associated with the given thread_id	MariaDB 10.6
SYS.PS_THREAD_ID	Returns the thread_id associated with the given connection_id	MariaDB 10.6
SYS.PS_THREAD_STACK	Returns all statements, stages, and events within the Performance Schema for a given thread_id	MariaDB 10.6

Name	Description	Added
SYS.PS_THREAD_TRX_INFO	Returns a JSON object with information about the thread specified by the given thread_id	MariaDB 10.6
SYS.QUOTE_IDENTIFIER	Quotes a string to produce a result that can be used as an identifier in an SQL statement	MariaDB 10.6
SYS.SYS_GET_CONFIG	Returns a configuration option value from the sys_config table	MariaDB 10.6
SYS.VERSION_MAJOR	Returns the MariaDB Server major release version	MariaDB 10.6
SYS.VERSION_MINOR	Returns the MariaDB Server minor release version	MariaDB 10.6
SYS.VERSION_PATCH	Returns the MariaDB Server patch release version	MariaDB 10.6
SYS_GUID	Generates a globally unique identifier	
SYSDATE	Returns the current date and time	
SYSTEM_USER	Synonym for USER()	
TAN	Returns the tangent	
TIME function	Extracts the time	
TIMEDIFF	Returns the difference between two date/times	
TIMESTAMP FUNCTION	Return the datetime, or add a time to a date/time	
TIMESTAMPADD	Add interval to a date or datetime	
TIMESTAMPDIFF	Difference between two datetimes	
TIME_FORMAT	Formats the time value according to the format string	
TIME_TO_SEC	Returns the time argument, converted to seconds	
TO_BASE64	Converts a string to its base-64 encoded form	
TO_CHAR	Converts a date/time type to a char	
TO_DAYS	Number of days since year 0	
TO_NUMBER	Converts and expression to NUMERIC	MariaDB 12.2
TO_SECONDS	Number of seconds since year 0	
TOUCHES	Whether two geometries spatially touch	
TRIM	Returns a string with all given prefixes or suffixes removed	
TRUNC	Returns a DATETIME truncated according to a given format.	MariaDB 12.2
TRUNCATE	Truncates X to D decimal places	
UCASE	Synonym for UPPER]]()	
UNHEX	Interprets pairs of hex digits as a number and converts to the character represented by the number	
UNCOMPRESS	Uncompresses string compressed with COMPRESS()	
UNCOMPRESSED_LENGTH	Returns length of a string before being compressed with COMPRESS()	
UNIX_TIMESTAMP	Returns a Unix timestamp	
UPDATEXML	Replace XML	
UPPER	Changes string to uppercase	
USER	Current user/host	
UTC_DATE	Returns the current UTC date	
UTC_TIME	Returns the current UTC time	
UTC_TIMESTAMP	Returns the current UTC date and time	

Name	Description	Added
UUID	Returns a Universal Unique Identifier v1	
UUIDv4	Returns a Universal Unique Identifier v4	MariaDB 11.7
UUIDv7	Returns a Universal Unique Identifier v7	MariaDB 11.7
UUID_SHORT	Return short universal identifier	
VALUES or VALUE	Refer to columns in INSERT … ON DUPLICATE KEY UPDATE	
VAR_POP	Population standard variance	
VAR_SAMP	Returns the sample variance	
VARIANCE	Population standard variance	
VEC_DISTANCE	Calculates either a Euclidean or Cosine distance between two vectors.	MariaDB 11.8
VEC_DISTANCE_COSINE	Calculates a Cosine distance between two vectors.	MariaDB 11.7
VEC_DISTANCE_EUCLIDEAN	Calculates a Euclidean (L2) distance between two points.	MariaDB 11.7
VEC_FromText	Converts a text representation of the vector to a vector.	MariaDB 11.6.0 Vector
VEC_ToText	Converts a binary vector into a json array of numbers (floats).	MariaDB 11.6.0 Vector
VERSION	MariaDB server version	
WEEK	Returns the week number	
WEEKDAY	Returns the weekday index	
WEEKOFYEAR	Returns the calendar week of the date as a number in the range from 1 to 53	
WEIGHT_STRING	Weight of the input string	
WITHIN	Indicate whether a geographic element is spacially within another	
WSREP_LAST_SEEN_GTID	Returns the Global Transaction ID of the most recent write transaction observed by the client.	
WSREP_LAST_WRITTEN_GTID	Returns the Global Transaction ID of the most recent write transaction performed by the client.	
WSREP_SYNC_WAIT_UPTO_GTID	Blocks the client until the transaction specified by the given Global Transaction ID is applied and committed by the node	
X	Synonym for ST_X	
Y	Synonym for ST_Y	
YEAR	Returns the year for the given date	
YEARWEEK	Returns year and week for a date	

This page is licensed: CC BY-SA / Gnu FDL

Aggregate Functions

See Stored Aggregate Functions.

Stored Aggregate Functions allow users to create custom aggregate functions that process a sequence of rows and return a single summary result. This page provides a general overview.

See AVG.

Calculate the average value. This function computes the arithmetic mean of a numeric expression, ignoring `NULL` values.

See BIT_AND.

Perform a bitwise `AND` operation. This function returns the result of performing a bitwise `AND` on all values in a given expression.

See BIT_OR.

Perform a bitwise `OR` operation. This function returns the result of performing a bitwise `OR` on all values in a given expression.

See BIT_XOR.

Perform a bitwise `XOR` operation. This function returns the result of performing a bitwise `XOR` on all values in a given expression.

See COUNT DISTINCT.

Count unique values. This function returns the number of distinct, non-`NULL` values found in the specified column or expression.

See COUNT.

Complete `COUNT()` function reference: `COUNT(*)` and `COUNT(expr)` syntax, `COUNT(DISTINCT)` usage, `GROUP BY` aggregation, and `OVER(PARTITION BY)` window syntax.

See GROUP_CONCAT.

Complete `GROUP_CONCAT` reference for MariaDB. Complete function guide with syntax, parameters, return values, and usage examples for production use.

See JSON_ARRAYAGG.

Aggregate values into a JSON array. This function aggregates a result set column into a single JSON array.

See JSON_OBJECTAGG.

Aggregate key-value pairs into a JSON object. This function aggregates two columns or expressions into a single JSON object.

See MAX.

Find the maximum value. This function returns the highest value in a set of values, applicable to numbers, strings, and dates.

See MIN.

Find the minimum value. This function returns the lowest value in a set of values, applicable to numbers, strings, and dates.

See STD.

Calculate population standard deviation. This function returns the square root of the population variance. It is a synonym for `STDDEV_POP()`.

See STDDEV.

Calculate population standard deviation. This function is a synonym for `STD()` and `STDDEV_POP()`, returning the square root of the population variance.

See STDDEV_POP.

Calculate population standard deviation. This function computes the standard deviation assuming the set of values represents the entire population.

See STDDEV_SAMP.

Calculate sample standard deviation. This function computes the standard deviation assuming the set of values represents a sample of the population.

See SUM.

Calculate the total sum. This function returns the sum of all values in a numeric expression, ignoring `NULL` values.

See VAR_POP.

Calculate population variance. This function computes the statistical variance for a set of values assumed to be the entire population.

See VAR_SAMP.

Calculate sample variance. This function computes the statistical variance for a set of values assumed to be a sample of the population.

See VARIANCE.

Calculate population variance. This function is a synonym for VAR_POP() and returns the variance of a set of values.

This page is licensed: CC BY-SA / Gnu FDL

AVG

Syntax

```
AVG([DISTINCT] expr)
```

Description

Returns the average value of expr. The DISTINCT option can be used to return the average of the distinct values of expr. NULL values are ignored. It is an aggregate function, and so can be used with the GROUP BY clause.

AVG() returns NULL if there were no matching rows.

AVG() can be used as a window function.

Examples

```
CREATE TABLE sales (sales_value INT);

INSERT INTO sales VALUES(10),(20),(20),(40);

SELECT AVG(sales_value) FROM sales;
+------------------+
| AVG(sales_value) |
+------------------+
|          22.5000 |
+------------------+

SELECT AVG(DISTINCT(sales_value)) FROM sales;
+---------------------------+
| AVG(DISTINCT(sales_value)) |
+---------------------------+
|                   23.3333 |
+---------------------------+
```

Commonly, AVG() is used with a GROUP BY clause:

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
  ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
  ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
  ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
  ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT name, AVG(score) FROM student GROUP BY name;
+---------+------------+
| name    | AVG(score) |
+---------+------------+
| Chun    |    74.0000 |
| Esben   |    37.0000 |
| Kaolin  |    72.0000 |
| Tatiana |    85.0000 |
+---------+------------+
```

Be careful to avoid this common mistake, not grouping correctly and returning mismatched data:

```
SELECT name,test,AVG(score) FROM student;
+------+------+------------+
| name | test | MIN(score) |
+------+------+------------+
| Chun | SQL  |         31 |
+------+------+------------+
```

As a window function:

```
CREATE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT name, test, score, AVG(score) OVER (PARTITION BY test)
    AS average_by_test FROM student_test;
+---------+--------+-------+-----------------+
| name    | test   | score | average_by_test |
+---------+--------+-------+-----------------+
| Chun    | SQL    |    75 |         65.2500 |
| Chun    | Tuning |    73 |         68.7500 |
| Esben   | SQL    |    43 |         65.2500 |
| Esben   | Tuning |    31 |         68.7500 |
| Kaolin  | SQL    |    56 |         65.2500 |
| Kaolin  | Tuning |    88 |         68.7500 |
| Tatiana | SQL    |    87 |         65.2500 |
| Tatiana | Tuning |    83 |         68.7500 |
+---------+--------+-------+-----------------+
```

See Also

- MAX (maximum)
- MIN (minimum)
- SUM (sum total)

This page is licensed: GPLv2, originally from fill_help_tables.sql

BIT_AND

Syntax

```
BIT_AND(expr) [over_clause]
```

Description

Returns the bitwise AND of all bits in *expr*. The calculation is performed with 64-bit (BIGINT) precision. It is an aggregate function, and so can be used with the GROUP BY clause.

If no rows match, `BIT_AND` will return a value with all bits set to 1. NULL values have no effect on the result unless all results are NULL, which is treated as no match.

`BIT_AND` can be used as a window function with the addition of the *over_clause*.

Examples

```
CREATE TABLE vals (x INT);

INSERT INTO vals VALUES(111),(110),(100);

SELECT BIT_AND(x), BIT_OR(x), BIT_XOR(x) FROM vals;
+------------+-----------+------------+
| BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+------------+-----------+------------+
|        100 |       111 |        101 |
+------------+-----------+------------+
```

As an aggregate function:

```
CREATE TABLE vals2 (category VARCHAR(1), x INT);

INSERT INTO vals2 VALUES
  ('a',111),('a',110),('a',100),
  ('b','000'),('b',001),('b',011);

SELECT category, BIT_AND(x), BIT_OR(x), BIT_XOR(x)
  FROM vals GROUP BY category;
+----------+------------+-----------+------------+
| category | BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+----------+------------+-----------+------------+
| a        |        100 |       111 |        101 |
| b        |          0 |        11 |         10 |
+----------+------------+-----------+------------+
```

No match:

```
SELECT BIT_AND(NULL);
+----------------------+
| BIT_AND(NULL)        |
+----------------------+
| 18446744073709551615 |
+----------------------+
```

See Also

- BIT_OR
- BIT_XOR

This page is licensed: GPLv2, originally from fill_help_tables.sql

BIT_OR

Syntax

```
BIT_OR(expr) [over_clause]
```

Description

Returns the bitwise OR of all bits in `expr`. The calculation is performed with 64-bit (BIGINT) precision. It is an aggregate function, and so can be used with the GROUP BY clause.

If no rows match, `BIT_OR` will return a value with all bits set to `0`. NULL values have no effect on the result unless all results are NULL, which is treated as no match.

`BIT_OR` can be used as a window function with the addition of the *over_clause*.

Examples

```
CREATE TABLE vals (x INT);

INSERT INTO vals VALUES(111),(110),(100);

SELECT BIT_AND(x), BIT_OR(x), BIT_XOR(x) FROM vals;
+------------+-----------+------------+
| BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+------------+-----------+------------+
|        100 |       111 |        101 |
+------------+-----------+------------+
```

As an aggregate function:

```
CREATE TABLE vals2 (category VARCHAR(1), x INT);

INSERT INTO vals2 VALUES
  ('a',111),('a',110),('a',100),
  ('b','000'),('b',001),('b',011);

SELECT category, BIT_AND(x), BIT_OR(x), BIT_XOR(x)
  FROM vals GROUP BY category;
+----------+------------+-----------+------------+
| category | BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+----------+------------+-----------+------------+
| a        |        100 |       111 |        101 |
| b        |          0 |        11 |         10 |
+----------+------------+-----------+------------+
```

No match:

```
SELECT BIT_OR(NULL);
+--------------+
| BIT_OR(NULL) |
+--------------+
|            0 |
+--------------+
```

See Also

- BIT_AND
- BIT_XOR

This page is licensed: GPLv2, originally from fill_help_tables.sql

BIT_XOR

Syntax

```
BIT_XOR(expr) [over_clause]
```

Description

Returns the bitwise XOR of all bits in `expr`. The calculation is performed with 64-bit (BIGINT) precision. It is an aggregate function, and so can be used with the GROUP BY clause.

If no rows match, `BIT_XOR` will return a value with all bits set to `0`. NULL values have no effect on the result unless all results are NULL, which is treated as no match.

`BIT_XOR` can be used as a window function with the addition of the *over_clause*.

Examples

```
CREATE TABLE vals (x INT);

INSERT INTO vals VALUES(111),(110),(100);

SELECT BIT_AND(x), BIT_OR(x), BIT_XOR(x) FROM vals;
+------------+-----------+------------+
| BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+------------+-----------+------------+
|        100 |       111 |        101 |
+------------+-----------+------------+
```

As an aggregate function:

```
CREATE TABLE vals2 (category VARCHAR(1), x INT);

INSERT INTO vals2 VALUES
  ('a',111),('a',110),('a',100),
  ('b','000'),('b',001),('b',011);

SELECT category, BIT_AND(x), BIT_OR(x), BIT_XOR(x)
  FROM vals GROUP BY category;
+----------+------------+-----------+------------+
| category | BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+----------+------------+-----------+------------+
| a        |        100 |       111 |        101 |
| b        |          0 |        11 |         10 |
+----------+------------+-----------+------------+
```

No match:

```
SELECT BIT_XOR(NULL);
+---------------+
| BIT_XOR(NULL) |
+---------------+
|             0 |
+---------------+
```

See Also

- BIT_AND
- BIT_OR

This page is licensed: GPLv2, originally from fill_help_tables.sql

COUNT DISTINCT

Syntax

```
COUNT(DISTINCT expr,[expr...])
```

Description

Returns a count of the number of different non-NULL values.

COUNT(DISTINCT) returns 0 if there were no matching rows.

Examples

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
  ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
  ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
  ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
  ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT COUNT(*) FROM student;
+----------+
| COUNT(*) |
+----------+
|        8 |
+----------+

SELECT COUNT(DISTINCT (name)) FROM student;
+------------------------+
| COUNT(DISTINCT (name)) |
+------------------------+
|                      4 |
+------------------------+
```

See Also

- SELECT
- COUNT

This page is licensed: GPLv2, originally from fill_help_tables.sql

COUNT

Syntax

```
COUNT(expr)
```

Description

Returns a count of the number of non-NULL values of expr in the rows retrieved by a SELECT statement. The result is a BIGINT value. It is an aggregate function, and so can be used with the GROUP BY clause.

COUNT(*) counts the total number of rows in a table.

COUNT() returns 0 if there were no matching rows.

COUNT() can be used as a window function.

Examples

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
  ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
  ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
  ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
  ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT COUNT(*) FROM student;
+----------+
| COUNT(*) |
+----------+
|        8 |
+----------+
```

COUNT(DISTINCT) example:

```
SELECT COUNT(DISTINCT (name)) FROM student;
+------------------------+
| COUNT(DISTINCT (name)) |
+------------------------+
|                      4 |
+------------------------+
```

As a window function

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, COUNT(score) OVER (PARTITION BY name)
    AS tests_written FROM student_test;
+---------+--------+-------+---------------+
| name    | test   | score | tests_written |
+---------+--------+-------+---------------+
| Chun    | SQL    |    75 |             2 |
| Chun    | Tuning |    73 |             2 |
| Esben   | SQL    |    43 |             2 |
| Esben   | Tuning |    31 |             2 |
| Kaolin  | SQL    |    56 |             2 |
| Kaolin  | Tuning |    88 |             2 |
| Tatiana | SQL    |    87 |             1 |
+---------+--------+-------+---------------+
```

See Also

- SELECT
- COUNT DISTINCT
- Window Functions

This page is licensed: GPLv2, originally from fill_help_tables.sql

GROUP_CONCAT

Syntax

```
GROUP_CONCAT(expr)
```

Description

This function returns a string result with the concatenated non-NULL values from a group. If any expr in GROUP_CONCAT evaluates to NULL, that tuple is not present in the list returned by GROUP_CONCAT.

It returns NULL if all arguments are NULL, or there are no matching rows.

The maximum returned length in bytes is determined by the group_concat_max_len server system variable, which defaults to 1M.

If group_concat_max_len <= 512, the return type is VARBINARY or VARCHAR; otherwise, the return type is BLOB or TEXT. The choice between binary or non-binary types depends from the input.

The full syntax is as follows:

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
             [ORDER BY {unsigned_integer | col_name | expr}
                 [ASC | DESC] [,col_name ...]]
             [SEPARATOR str_val]
             [LIMIT {[offset,] row_count | row_count OFFSET offset}])
```

`DISTINCT` eliminates duplicate values from the output string.

ORDER BY determines the order of returned values.

`SEPARATOR` specifies a separator between the values. The default separator is a comma (`,`). It is possible to avoid using a separator by specifying an empty string.

LIMIT

The LIMIT clause can be used with `GROUP_CONCAT`.

Examples

```
SELECT student_name,
       GROUP_CONCAT(test_score)
       FROM student
       GROUP BY student_name;
```

Get a readable list of MariaDB users from the mysql.user table:

```
SELECT GROUP_CONCAT(DISTINCT User ORDER BY User SEPARATOR '\n')
   FROM mysql.user;
```

In the former example, `DISTINCT` is used because the same user may occur more than once. The new line () used as a `SEPARATOR` makes the results easier to read.

Get a readable list of hosts from which each user can connect:

```
SELECT User, GROUP_CONCAT(Host ORDER BY Host SEPARATOR ', ')
   FROM mysql.user GROUP BY User ORDER BY User;
```

The former example shows the difference between the `GROUP_CONCAT`'s ORDER BY (which sorts the concatenated hosts), and the `SELECT`'s ORDER BY (which sorts the rows).

LIMIT can be used with `GROUP_CONCAT`, so, for example, given the following table:

```
CREATE TABLE d (dd DATE, cc INT);

INSERT INTO d VALUES ('2017-01-01',1);
INSERT INTO d VALUES ('2017-01-02',2);
INSERT INTO d VALUES ('2017-01-04',3);
```

the following query:

```
SELECT SUBSTRING_INDEX(GROUP_CONCAT(CONCAT_WS(":",dd,cc)
ORDER BY cc DESC),",",1) FROM d;
+-----------------------------------------------------------------------------+
| SUBSTRING_INDEX(GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC),",",1) |
+-----------------------------------------------------------------------------+
| 2017-01-04:3                                                                |
+-----------------------------------------------------------------------------+
```

can be more simply rewritten as:

```
SELECT GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC LIMIT 1) FROM d;
+-------------------------------------------------------------+
| GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC LIMIT 1) |
+-------------------------------------------------------------+
| 2017-01-04:3                                                |
+-------------------------------------------------------------+
```

NULLS:

```
CREATE OR REPLACE TABLE t1 (a int, b char);

INSERT INTO t1 VALUES (1, 'a'), (2, NULL);

SELECT GROUP_CONCAT(a, b) FROM t1;
+--------------------+
| GROUP_CONCAT(a, b) |
+--------------------+
| 1a                 |
+--------------------+
```

See Also

- CONCAT()
- CONCAT_WS()
- SELECT
- ORDER BY

This page is licensed: GPLv2, originally from fill_help_tables.sql

JSON_ARRAYAGG

> **Note:**
>
> `JSON_ARRAYAGG` is available from MariaDB 10.5.

Syntax

```
JSON_ARRAYAGG(column_or_expression)
```

Description

`JSON_ARRAYAGG` returns a JSON array containing an element for each value in a given set of JSON or SQL values. It acts on a column or an expression that evaluates to a single value.

The maximum returned length in bytes is determined by the group_concat_max_len server system variable.

Returns `NULL` in the case of an error, or if the result contains no rows.

`JSON_ARRAYAGG` cannot currently be used as a window function.

The full syntax is as follows:

```
JSON_ARRAYAGG([DISTINCT] expr
             [ORDER BY {unsigned_integer | col_name | expr}
                 [ASC | DESC] [,col_name ...]]
             [LIMIT {[offset,] row_count | row_count OFFSET offset}])
```

Examples

```
CREATE TABLE t1 (a INT, b INT);

INSERT INTO t1 VALUES (1, 1),(2, 1), (1, 1),(2, 1), (3, 2),(2, 2),(2, 2),(2, 2);

SELECT JSON_ARRAYAGG(a), JSON_ARRAYAGG(b) FROM t1;
+-------------------+-------------------+
| JSON_ARRAYAGG(a)  | JSON_ARRAYAGG(b)  |
+-------------------+-------------------+
| [1,2,1,2,3,2,2,2] | [1,1,1,1,2,2,2,2] |
+-------------------+-------------------+

SELECT JSON_ARRAYAGG(a), JSON_ARRAYAGG(b) FROM t1 GROUP BY b;
+------------------+------------------+
| JSON_ARRAYAGG(a) | JSON_ARRAYAGG(b) |
+------------------+------------------+
| [1,2,1,2]        | [1,1,1,1]        |
| [3,2,2,2]        | [2,2,2,2]        |
+------------------+------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_OBJECTAGG

Note:

`JSON_OBJECTAGG` is available from MariaDB 10.5.

Syntax

```
JSON_OBJECTAGG(key, value)
```

Description

`JSON_OBJECTAGG` returns a JSON object containing key-value pairs. It takes two expressions that evaluate to a single value, or two column names, as arguments, the first used as a key, and the second as a value.

The maximum returned length in bytes is determined by the group_concat_max_len server system variable.

Returns `NULL` in the case of an error, or if the result contains no rows.

`JSON_OBJECTAGG` cannot currently be used as a window function.

Examples

```
SELECT * FROM t1;
+------+-------+
| a    | b     |
+------+-------+
|    1 | Hello |
|    1 | World |
|    2 | This  |
+------+-------+

SELECT JSON_OBJECTAGG(a, b) FROM t1;
+----------------------------------------+
| JSON_OBJECTAGG(a, b)                   |
+----------------------------------------+
| {"1":"Hello", "1":"World", "2":"This"} |
+----------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

MAX

Syntax

```
MAX([DISTINCT] expr)
```

Description

Returns the largest, or maximum, value of *`expr`*. `MAX()` can also take a string argument in which case it returns the maximum string value. The `DISTINCT` keyword can be used to find the maximum of the distinct values of *`expr`*, however, this produces the same result as omitting `DISTINCT`.

Note that SET and ENUM fields are currently compared by their string value rather than their relative position in the set, so MAX() may produce a different highest result than ORDER BY DESC.

It is an aggregate function, and so can be used with the GROUP BY clause.

MAX() can be used as a window function.

`MAX()` returns `NULL` if there were no matching rows.

> **CURRENT**
>
> Not only ascending, but also descending indexes can be used to optimize `MAX`.
>
> **< 11.4**
>
> Only ascending indexes can be used to optimize `MAX`.

Examples

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
  ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
  ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
  ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
  ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT name, MAX(score) FROM student GROUP BY name;
+---------+------------+
| name    | MAX(score) |
+---------+------------+
| Chun    |         75 |
| Esben   |         43 |
| Kaolin  |         88 |
| Tatiana |         87 |
+---------+------------+
```

`MAX` string:

```
SELECT MAX(name) FROM student;
+-----------+
| MAX(name) |
+-----------+
| Tatiana   |
+-----------+
```

Be careful to avoid this common mistake, not grouping correctly and returning mismatched data:

```
SELECT name,test,MAX(SCORE) FROM student;
+------+------+------------+
| name | test | MAX(SCORE) |
+------+------+------------+
| Chun | SQL  |         88 |
+------+------+------------+
```

Difference between `ORDER BY DESC` and `MAX()`:

```
CREATE TABLE student2(name CHAR(10),grade ENUM('b','c','a'));

INSERT INTO student2 VALUES('Chun','b'),('Esben','c'),('Kaolin','a');

SELECT MAX(grade) FROM student2;
+------------+
| MAX(grade) |
+------------+
| c          |
+------------+

SELECT grade FROM student2 ORDER BY grade DESC LIMIT 1;
+-------+
| grade |
+-------+
| a     |
+-------+
```

As a window function:

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);
INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, MAX(score)
  OVER (PARTITION BY name) AS highest_score FROM student_test;
+---------+--------+-------+---------------+
| name    | test   | score | highest_score |
+---------+--------+-------+---------------+
| Chun    | SQL    |    75 |            75 |
| Chun    | Tuning |    73 |            75 |
| Esben   | SQL    |    43 |            43 |
| Esben   | Tuning |    31 |            43 |
| Kaolin  | SQL    |    56 |            88 |
| Kaolin  | Tuning |    88 |            88 |
| Tatiana | SQL    |    87 |            87 |
+---------+--------+-------+---------------+
```

See Also

- AVG (average)
- MIN (minimum)
- SUM (sum total)
- GREATEST() returns the largest value from a list

This page is licensed: GPLv2, originally from fill_help_tables.sql

MIN

Syntax

```
MIN([DISTINCT] expr)
```

Description

Returns the minimum value of *expr*. `MIN()` may take a string argument, in which case it returns the minimum string value. The `DISTINCT` keyword can be used to find the minimum of the distinct values of *expr*, however, this produces the same result as omitting `DISTINCT`.

Note that SET and ENUM fields are currently compared by their string value rather than their relative position in the set, so MIN() may produce a different lowest result than ORDER BY ASC.

It is an aggregate function, and so can be used with the GROUP BY clause.

`MIN()` can be used as a window function.

`MIN()` returns `NULL` if there were no matching rows.

> **CURRENT**
>
> Not only ascending, but also descending indexes can be used to optimize `MIN`.
>
> **< 11.4**
>
> Only ascending indexes can be used to optimize `MIN`.

Examples

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
  ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
  ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
  ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
  ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT name, MIN(score) FROM student GROUP BY name;
+---------+------------+
| name    | MIN(score) |
+---------+------------+
| Chun    |         73 |
| Esben   |         31 |
| Kaolin  |         56 |
| Tatiana |         83 |
+---------+------------+
```

`MIN()` with a string:

```
SELECT MIN(name) FROM student;
+-----------+
| MIN(name) |
+-----------+
| Chun      |
+-----------+
```

Be careful to avoid this common mistake, not grouping correctly and returning mismatched data:

```
SELECT name,test,MIN(score) FROM student;
+------+------+------------+
| name | test | MIN(score) |
+------+------+------------+
| Chun | SQL  |         31 |
+------+------+------------+
```

Difference between `ORDER BY ASC` and `MIN()`:

```
CREATE TABLE student2(name CHAR(10),grade ENUM('b','c','a'));

INSERT INTO student2 VALUES('Chun','b'),('Esben','c'),('Kaolin','a');

SELECT MIN(grade) FROM student2;
+------------+
| MIN(grade) |
+------------+
| a          |
+------------+

SELECT grade FROM student2 ORDER BY grade ASC LIMIT 1;
+-------+
| grade |
+-------+
| b     |
+-------+
```

As a window function:

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);
INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, MIN(score)
  OVER (PARTITION BY name) AS lowest_score FROM student_test;
+---------+--------+-------+--------------+
| name    | test   | score | lowest_score |
+---------+--------+-------+--------------+
| Chun    | SQL    |    75 |           73 |
| Chun    | Tuning |    73 |           73 |
| Esben   | SQL    |    43 |           31 |
| Esben   | Tuning |    31 |           31 |
| Kaolin  | SQL    |    56 |           56 |
| Kaolin  | Tuning |    88 |           56 |
| Tatiana | SQL    |    87 |           87 |
+---------+--------+-------+--------------+
```

See Also

- AVG (average)
- MAX (maximum)
- SUM (sum total)
- LEAST() returns the smallest value from a list.

This page is licensed: GPLv2, originally from fill_help_tables.sql

STD

Syntax

```
STD(expr)
```

Description

Returns the population standard deviation of `expr`. This is an extension to standard SQL. The standard SQL function STDDEV_POP() can be used instead.

It is an aggregate function, and so can be used with the GROUP BY clause.

`STD()` can be used as a window function.

This function returns `NULL` if there were no matching rows.

Examples

As an aggregate function:

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
  ('a',1),('a',2),('a',3),
  ('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+----------+--------------+----------------+------------+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+----------+--------------+----------------+------------+
| a        |        0.8165 |         1.0000 |     0.6667 |
| b        |       18.0400 |        20.1693 |   325.4400 |
+----------+--------------+----------------+------------+
```

As a window function:

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, STDDEV_POP(score)
  OVER (PARTITION BY test) AS stddev_results FROM student_test;
+---------+--------+-------+----------------+
| name    | test   | score | stddev_results |
+---------+--------+-------+----------------+
| Chun    | SQL    |    75 |        16.9466 |
| Chun    | Tuning |    73 |        24.1247 |
| Esben   | SQL    |    43 |        16.9466 |
| Esben   | Tuning |    31 |        24.1247 |
| Kaolin  | SQL    |    56 |        16.9466 |
| Kaolin  | Tuning |    88 |        24.1247 |
| Tatiana | SQL    |    87 |        16.9466 |
+---------+--------+-------+----------------+
```

See Also

- STDDEV_POP (equivalent, standard SQL)
- STDDEV (equivalent, Oracle-compatible non-standard SQL)
- VAR_POP (variance)
- STDDEV_SAMP (sample standard deviation)

This page is licensed: GPLv2, originally from fill_help_tables.sql

STDDEV

Syntax

```
STDDEV(expr)
```

Description

Returns the population standard deviation of `expr`. This function is provided for compatibility with Oracle. The standard SQL function STDDEV_POP() can be used instead.

It is an aggregate function, and so can be used with the GROUP BY clause.

`STDDEV()` can be used as a window function.

This function returns `NULL` if there were no matching rows.

Examples

As an aggregate function:

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
  ('a',1),('a',2),('a',3),
  ('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+----------+---------------+----------------+------------+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+----------+---------------+----------------+------------+
| a        |        0.8165 |         1.0000 |     0.6667 |
| b        |       18.0400 |        20.1693 |   325.4400 |
+----------+---------------+----------------+------------+
```

As a window function:

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, STDDEV_POP(score)
  OVER (PARTITION BY test) AS stddev_results FROM student_test;
+---------+--------+-------+----------------+
| name    | test   | score | stddev_results |
+---------+--------+-------+----------------+
| Chun    | SQL    |    75 |        16.9466 |
| Chun    | Tuning |    73 |        24.1247 |
| Esben   | SQL    |    43 |        16.9466 |
| Esben   | Tuning |    31 |        24.1247 |
| Kaolin  | SQL    |    56 |        16.9466 |
| Kaolin  | Tuning |    88 |        24.1247 |
| Tatiana | SQL    |    87 |        16.9466 |
+---------+--------+-------+----------------+
```

See Also

- STDDEV_POP (equivalent, standard SQL)
- STD (equivalent, non-standard SQL)
- VAR_POP (variance)
- STDDEV_SAMP (sample standard deviation)

This page is licensed: GPLv2, originally from fill_help_tables.sql

STDDEV_POP

Syntax

```
STDDEV_POP(expr)
```

Description

Returns the population standard deviation of *`expr`* (the square root of VAR_POP()). You can also use STD() or STDDEV(), which are equivalent but not standard SQL.

It is an aggregate function, and so can be used with the GROUP BY clause.

`STDDEV_POP()` can be used as a window function.

`STDDEV_POP()` returns `NULL` if there were no matching rows.

Examples

As an aggregate function:

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
  ('a',1),('a',2),('a',3),
  ('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+----------+---------------+----------------+------------+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+----------+---------------+----------------+------------+
| a        |        0.8165 |         1.0000 |     0.6667 |
| b        |       18.0400 |        20.1693 |   325.4400 |
+----------+---------------+----------------+------------+
```

As a window function:

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, STDDEV_POP(score)
  OVER (PARTITION BY test) AS stddev_results FROM student_test;
+---------+--------+-------+----------------+
| name    | test   | score | stddev_results |
+---------+--------+-------+----------------+
| Chun    | SQL    |    75 |        16.9466 |
| Chun    | Tuning |    73 |        24.1247 |
| Esben   | SQL    |    43 |        16.9466 |
| Esben   | Tuning |    31 |        24.1247 |
| Kaolin  | SQL    |    56 |        16.9466 |
| Kaolin  | Tuning |    88 |        24.1247 |
| Tatiana | SQL    |    87 |        16.9466 |
+---------+--------+-------+----------------+
```

See Also

- STD (equivalent, non-standard SQL)
- STDDEV (equivalent, Oracle-compatible non-standard SQL)
- VAR_POP (variance)
- STDDEV_SAMP (sample standard deviation)

This page is licensed: GPLv2, originally from fill_help_tables.sql

STDDEV_SAMP

Syntax

```
STDDEV_SAMP(expr)
```

Description

Returns the sample standard deviation of `expr` (the square root of VAR_SAMP()).

It is an aggregate function, and so can be used with the GROUP BY clause.

`STDDEV_SAMP()` can be used as a window function.

`STDDEV_SAMP()` returns `NULL` if there were no matching rows.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SUM

Syntax

```
SUM([DISTINCT] expr)
```

Description

Returns the sum of *`expr`*. If the return set has no rows, `SUM()` returns `NULL`. The `DISTINCT` keyword can be used to sum only the distinct values of `expr`.

`SUM()` can be used as a window function, although not with the `DISTINCT` specifier.

Examples

```
CREATE TABLE sales (sales_value INT);
INSERT INTO sales VALUES(10),(20),(20),(40);

SELECT SUM(sales_value) FROM sales;
+------------------+
| SUM(sales_value) |
+------------------+
|               90 |
+------------------+

SELECT SUM(DISTINCT(sales_value)) FROM sales;
+----------------------------+
| SUM(DISTINCT(sales_value)) |
+----------------------------+
|                         70 |
+----------------------------+
```

Commonly, `SUM` is used with a GROUP BY clause:

```
CREATE TABLE sales (name CHAR(10), month CHAR(10), units INT);

INSERT INTO sales VALUES
  ('Chun', 'Jan', 75), ('Chun', 'Feb', 73),
  ('Esben', 'Jan', 43), ('Esben', 'Feb', 31),
  ('Kaolin', 'Jan', 56), ('Kaolin', 'Feb', 88),
  ('Tatiana', 'Jan', 87), ('Tatiana', 'Feb', 83);

SELECT name, SUM(units) FROM sales GROUP BY name;
+---------+------------+
| name    | SUM(units) |
+---------+------------+
| Chun    |        148 |
| Esben   |         74 |
| Kaolin  |        144 |
| Tatiana |        170 |
+---------+------------+
```

The GROUP BY clause is required when using an aggregate function along with regular column data, otherwise the result will be a mismatch, as in the following common type of mistake:

```
SELECT name,SUM(units) FROM sales
;+------+------------+
| name | SUM(units) |
+------+------------+
| Chun |        536 |
+------+------------+
```

As a window function:

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);
INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, SUM(score) OVER (PARTITION BY name) AS total_score FROM student_test;
+---------+--------+-------+-------------+
| name    | test   | score | total_score |
+---------+--------+-------+-------------+
| Chun    | SQL    |    75 |         148 |
| Chun    | Tuning |    73 |         148 |
| Esben   | SQL    |    43 |          74 |
| Esben   | Tuning |    31 |          74 |
| Kaolin  | SQL    |    56 |         144 |
| Kaolin  | Tuning |    88 |         144 |
| Tatiana | SQL    |    87 |          87 |
+---------+--------+-------+-------------+
```

See Also

- AVG (average)
- MAX (maximum)
- MIN (minimum)

This page is licensed: GPLv2, originally from fill_help_tables.sql

VAR_POP

Syntax

```
VAR_POP(expr)
```

Description

Returns the population standard variance of `expr`. It considers rows as the whole population, not as a sample, so it has the number of rows as the denominator. You can also use VARIANCE(), which is equivalent but is not standard SQL.

Variance is calculated by

- working out the mean for the set;
- for each number, subtracting the mean and squaring the result;
- calculating the average of the resulting differences.

It is an aggregate function, and so can be used with the GROUP BY clause.

`VAR_POP()` can be used as a window function.

`VAR_POP()` returns `NULL` if there were no matching rows.

Examples

```
CREATE TABLE v(i tinyint);

INSERT INTO v VALUES(101),(99);

SELECT VAR_POP(i) FROM v;
+------------+
| VAR_POP(i) |
+------------+
|     1.0000 |
+------------+

INSERT INTO v VALUES(120),(80);

SELECT VAR_POP(i) FROM v;
+------------+
| VAR_POP(i) |
+------------+
|   200.5000 |
+------------+
```

As an aggregate function:

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
  ('a',1),('a',2),('a',3),
  ('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+----------+---------------+----------------+------------+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+----------+---------------+----------------+------------+
| a        |        0.8165 |         1.0000 |     0.6667 |
| b        |       18.0400 |        20.1693 |   325.4400 |
+----------+---------------+----------------+------------+
```

As a window function:

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, VAR_POP(score)
  OVER (PARTITION BY test) AS variance_results FROM student_test;
+---------+--------+-------+------------------+
| name    | test   | score | variance_results |
+---------+--------+-------+------------------+
| Chun    | SQL    |    75 |         287.1875 |
| Esben   | SQL    |    43 |         287.1875 |
| Kaolin  | SQL    |    56 |         287.1875 |
| Tatiana | SQL    |    87 |         287.1875 |
| Chun    | Tuning |    73 |         582.0000 |
| Esben   | Tuning |    31 |         582.0000 |
| Kaolin  | Tuning |    88 |         582.0000 |
+---------+--------+-------+------------------+
```

See Also

- VARIANCE (equivalent, non-standard SQL)
- STDDEV_POP (population standard deviation)
- STDDEV_SAMP (sample standard deviation)

This page is licensed: GPLv2, originally from fill_help_tables.sql

VAR_SAMP

Syntax

```
VAR_SAMP(expr)
```

Description

Returns the sample variance of *`expr`*. That is, the denominator is the number of rows minus one.

It is an aggregate function, and so can be used with the GROUP BY clause.

VAR_SAMP() can be used as a window function.

VAR_SAMP() returns `NULL` if there were no matching rows.

Examples

As an aggregate function:

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
  ('a',1),('a',2),('a',3),
  ('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+----------+--------------+---------------+------------+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+----------+--------------+---------------+------------+
| a        |       0.8165 |        1.0000 |     0.6667 |
| b        |      18.0400 |       20.1693 |   325.4400 |
+----------+--------------+---------------+------------+
```

As a window function:

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, VAR_SAMP(score)
  OVER (PARTITION BY test) AS variance_results FROM student_test;
+---------+--------+-------+------------------+
| name    | test   | score | variance_results |
+---------+--------+-------+------------------+
| Chun    | SQL    |    75 |         382.9167 |
| Chun    | Tuning |    73 |         873.0000 |
| Esben   | SQL    |    43 |         382.9167 |
| Esben   | Tuning |    31 |         873.0000 |
| Kaolin  | SQL    |    56 |         382.9167 |
| Kaolin  | Tuning |    88 |         873.0000 |
| Tatiana | SQL    |    87 |         382.9167 |
+---------+--------+-------+------------------+
```

See Also

- VAR_POP (variance)
- STDDEV_POP (population standard deviation)

This page is licensed: GPLv2, originally from fill_help_tables.sql

VARIANCE

Syntax

```
VARIANCE(expr)
```

Description

Returns the population standard variance of `expr`. This is an extension to standard SQL. The standard SQL function VAR_POP() can be used instead.

Variance is calculated by

- working out the mean for the set;
- for each number, subtracting the mean and squaring the result;
- calculating the average of the resulting differences.

It is an aggregate function, and so can be used with the GROUP BY clause.

`VARIANCE()` can be used as a window function.

`VARIANCE()` returns `NULL` if there were no matching rows.

Examples

```
CREATE TABLE v(i tinyint);

INSERT INTO v VALUES(101),(99);

SELECT VARIANCE(i) FROM v;
+-------------+
| VARIANCE(i) |
+-------------+
|      1.0000 |
+-------------+

INSERT INTO v VALUES(120),(80);

SELECT VARIANCE(i) FROM v;
+-------------+
| VARIANCE(i) |
+-------------+
|    200.5000 |
+-------------+
```

As an aggregate function:

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
  ('a',1),('a',2),('a',3),
  ('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+----------+---------------+----------------+------------+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+----------+---------------+----------------+------------+
| a        |        0.8165 |         1.0000 |     0.6667 |
| b        |       18.0400 |        20.1693 |   325.4400 |
+----------+---------------+----------------+------------+
```

As a window function:

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, VAR_POP(score)
  OVER (PARTITION BY test) AS variance_results FROM student_test;
+---------+--------+-------+------------------+
| name    | test   | score | variance_results |
+---------+--------+-------+------------------+
| Chun    | SQL    |    75 |         287.1875 |
| Chun    | Tuning |    73 |         582.0000 |
| Esben   | SQL    |    43 |         287.1875 |
| Esben   | Tuning |    31 |         582.0000 |
| Kaolin  | SQL    |    56 |         287.1875 |
| Kaolin  | Tuning |    88 |         582.0000 |
| Tatiana | SQL    |    87 |         287.1875 |
+---------+--------+-------+------------------+
```

See Also

- VAR_POP (equivalent, standard SQL)
- STDDEV_POP (population standard deviation)
- STDDEV_SAMP (sample standard deviation)

This page is licensed: GPLv2, originally from fill_help_tables.sql

Control Flow Functions

See CASE OPERATOR.

Implement conditional logic in SQL queries. This operator evaluates conditions and returns a specific value when the first true condition is met.

See DECODE_ORACLE.

Compare a value against a list of conditions. This Oracle-compatible function returns a corresponding result when a match is found, or a default value otherwise.

See IF Function.

Complete IF() function reference: IF(expr1,expr2,expr3) conditional syntax, TRUE/NULL evaluation rules, return type context (numeric/string), and examples.

See IFNULL.

Replace NULL values with a fallback. This function returns the first argument if it's not NULL; otherwise, it returns the specified replacement value.

See NULLIF.

Compare two expressions and return NULL if they are equal. If the expressions differ, the function returns the first expression.

See NVL.

Synonym for IFNULL. This Oracle-compatible function returns the first argument if it is not NULL, or the second argument if the first is NULL.

See NVL2.

Return values based on NULL status. This function returns the second argument if the first is not NULL, and the third argument if the first is NULL.

This page is licensed: CC BY-SA / Gnu FDL

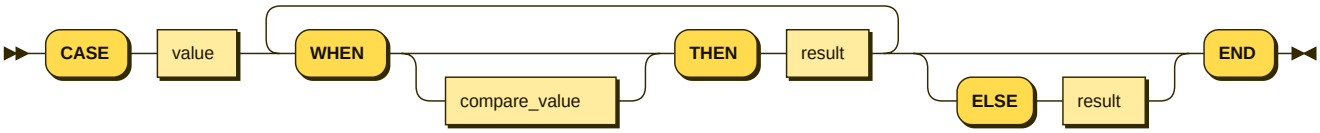
CASE OPERATOR

Syntax

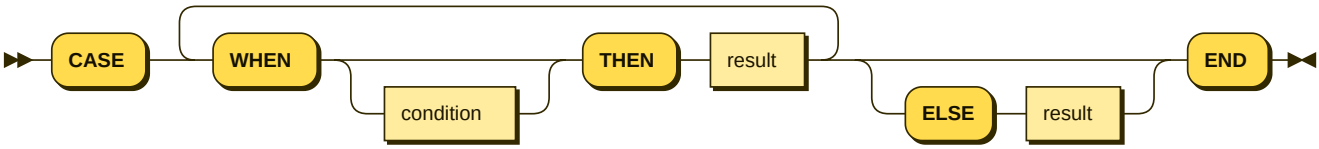
```
CASE value WHEN [compare_value] THEN result [WHEN [compare_value] THEN
result ...] [ELSE result] END

CASE WHEN [condition] THEN result [WHEN [condition] THEN result ...]
[ELSE result] END
```

The BNF documents the two CASE-operator forms (simple and searched); each gets its own diagram.



Railroad diagram of the simple CASE form



Railroad diagram of the searched CASE form

Description

The first version returns the result for the first `value=compare_value` comparison that is true. The second version returns the result for the first condition that is true. If there was no matching result value, the result after ELSE is returned, or NULL if there is no ELSE part.

There is also a CASE statement, which differs from the CASE operator described here.

Examples

```
SELECT CASE 1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE 'more' END;
+-------------------------------------------------------------+
| CASE 1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE 'more' END |
+-------------------------------------------------------------+
| one                                                         |
+-------------------------------------------------------------+

SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;
+----------------------------------------+
| CASE WHEN 1>0 THEN 'true' ELSE 'false' END |
+----------------------------------------+
| true                                   |
+----------------------------------------+

SELECT CASE BINARY 'B' WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;
+-----------------------------------------------+
| CASE BINARY 'B' WHEN 'a' THEN 1 WHEN 'b' THEN 2 END |
+-----------------------------------------------+
|                                          NULL |
+-----------------------------------------------+
```

Only the first matching condition is processed:

```
SELECT
  CASE true
     WHEN (1=1) THEN '1=1' -- result is returned
     WHEN (1=1 OR 2=2) THEN '1=1 OR 2=2' -- condition not processed
     ELSE 'else'
  END
;
+-------------------------------------------------------------------------------------+
| CASE true WHEN (1=1) THEN '1=1' WHEN (1=1 OR 2=2) THEN '1=1 OR 2=2' ELSE 'else' END |
+-------------------------------------------------------------------------------------+
+ 1=1                                                                                 +
+-------------------------------------------------------------------------------------+
```

See Also

- The CASE statement, which differs from the CASE operator described above.
- The IF() function.
- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

DECODE_ORACLE

`DECODE_ORACLE` is a synonym for the Oracle mode version of the DECODE function, and is available in all modes.

This page is licensed: CC BY-SA / Gnu FDL

IF Function

Syntax

```
IF(expr1,expr2,expr3)
```

Description

If `expr1` is `TRUE` (`expr1 <> 0` and `expr1 <> NULL`) then `IF()` returns `expr2`; otherwise it returns `expr3`. `IF()` returns a numeric or string value, depending on the context in which it is used.

Note: There is also an IF statement which differs from the `IF()` function described here.

Examples

```
SELECT IF(1>2,2,3);
+-------------+
| IF(1>2,2,3) |
+-------------+
|           3 |
+-------------+
```

```
SELECT IF(1<2,'yes','no');
+--------------------+
| IF(1<2,'yes','no') |
+--------------------+
| yes                |
+--------------------+
```

```
SELECT IF(STRCMP('test','test1'),'no','yes');
+---------------------------------------+
| IF(STRCMP('test','test1'),'no','yes') |
+---------------------------------------+
| no                                    |
+---------------------------------------+
```

See Also

There is also an IF statement, which differs from the `IF()` function described above.

This page is licensed: GPLv2, originally from fill_help_tables.sql

IFNULL

Syntax

```
IFNULL(expr1,expr2)
NVL(expr1,expr2)
```

Description

If *`expr1`* is not `NULL`, `IFNULL()` returns *`expr1`*; otherwise it returns_ `expr2` _. `IFNULL()` returns a numeric or string value, depending on the context in which it is used.

`NVL()` is an alias for `IFNULL()`.

Examples

```
SELECT IFNULL(1,0);
+-------------+
| IFNULL(1,0) |
+-------------+
|           1 |
+-------------+

SELECT IFNULL(NULL,10);
+-----------------+
| IFNULL(NULL,10) |
+-----------------+
|              10 |
+-----------------+

SELECT IFNULL(1/0,10);
+----------------+
| IFNULL(1/0,10) |
+----------------+
|        10.0000 |
+----------------+

SELECT IFNULL(1/0,'yes');
+-------------------+
| IFNULL(1/0,'yes') |
+-------------------+
| yes               |
+-------------------+
```

See Also

- NULL values
- IS NULL operator
- IS NOT NULL operator

- COALESCE function
- NULLIF function
- CONNECT data types

This page is licensed: GPLv2, originally from fill_help_tables.sql

NULLIF

Syntax

```
NULLIF(expr1,expr2)
```

Description

Returns `NULL` if expr1 = expr2 is true, otherwise returns expr1. This is the same as CASE `WHEN` expr1 = expr2 `THEN NULL ELSE` expr1 `END`.

Examples

```
SELECT NULLIF(1,1);
+-------------+
| NULLIF(1,1) |
+-------------+
|        NULL |
+-------------+

SELECT NULLIF(1,2);
+-------------+
| NULLIF(1,2) |
+-------------+
|           1 |
+-------------+
```

See Also

- NULL values
- IS NULL operator
- IS NOT NULL operator
- COALESCE function
- IFNULL function
- CONNECT data types

This page is licensed: GPLv2, originally from fill_help_tables.sql

NVL

`NVL` is a synonym for IFNULL.

This page is licensed: CC BY-SA / Gnu FDL

NVL2

Syntax

```
NVL2(expr1,expr2,expr3)
```

Description

The `NVL2` function returns a value based on whether a specified expression is `NULL` or not. If *expr1* is not `NULL`, then `NVL2` returns *expr2*. If *expr1* is `NULL`, then `NVL2` returns *expr3*.

Examples

```
SELECT NVL2(NULL,1,2);
+----------------+
| NVL2(NULL,1,2) |
+----------------+
|              2 |
+----------------+

SELECT NVL2('x',1,2);
+---------------+
| NVL2('x',1,2) |
+---------------+
|             1 |
+---------------+
```

See Also

- IFNULL (or NVL)

This page is licensed: CC BY-SA / Gnu FDL

Date & Time Functions

See Date and Time Units.

Reference keywords for date arithmetic. These units, such as DAY, HOUR, and MINUTE, specify the interval type used in functions like DATE_ADD and EXTRACT.

See ADD_MONTHS.

Add a specific number of months to a date. This Oracle-compatible function simplifies date calculations involving monthly intervals.

See ADDDATE.

Add a time interval to a date. This function performs date arithmetic, adding a specified value like days or hours to a starting date.

See ADDTIME.

Add a time value to a date or time expression. This function sums two time arguments, returning a new time or datetime result.

See CONVERT_TZ.

Convert a datetime value between time zones. This function shifts a timestamp from a source time zone to a target time zone.

See CURDATE.

Return the current date. This function outputs today's date as a value in 'YYYY-MM-DD' or YYYYMMDD format, depending on the context.

See CURRENT_DATE.

Synonym for CURDATE(). Returns the current date as a value in 'YYYY-MM-DD' or YYYYMMDD format.

See CURRENT_TIME.

Synonym for CURTIME(). Returns the current time as a value in ‘HH:MM:SS’ or HHMMSS format.

See CURRENT_TIMESTAMP.

Synonym for NOW(). Returns the current date and time as a value in ‘YYYY-MM-DD HH:MM:SS’ or YYYYMMDDHHMMSS format.

See CURTIME.

Return the current time. This function outputs the current time of day as a value in ‘HH:MM:SS’ or HHMMSS format.

See DATE FUNCTION.

Extract the date part from a datetime expression. This function returns the year, month, and day portions, discarding the time component.

See DATE_ADD.

Complete DATE_ADD() reference: DATE_ADD(date, INTERVAL expr unit) syntax, negative interval support, unit keywords (DAY/MONTH/YEAR), and return types.

See DATE_FORMAT.

Complete DATE_FORMAT reference for MariaDB. Complete function guide with syntax, parameters, return values, and usage examples with comprehensive examples.

See DATE_SUB.

Subtract a time interval from a date. This function calculates a past date by subtracting a specified unit, such as days, from a starting value.

See DATEDIFF.

Complete DATEDIFF() reference: DATEDIFF(expr1,expr2) syntax, date vs datetime expression handling, time component ignore, and positive/negative results.

See DAY.

Synonym for DAYOFMONTH(). Returns the day of the month (1-31) for a given date.

See DAYNAME.

Return the name of the weekday. This function returns the full name of the day, such as ‘Monday’ or ‘Sunday’, for a given date.

See DAYOFMONTH.

Return the day of the month. This function extracts the day portion of a date, returning a number from 1 to 31.

See DAYOFWEEK.

Return the weekday index. This function returns a number from 1 (Sunday) to 7 (Saturday) representing the day of the week.

See DAYOFYEAR.

Return the day of the year. This function returns a number from 1 to 366 indicating the day’s position within the year.

See EXTRACT.

Extract a specific part of a date. This function retrieves components like YEAR, MONTH, DAY, or HOUR from a date or datetime expression.

See FORMAT_PICO_TIME.

Format a time in picoseconds. This function converts a numeric picosecond value into a human-readable string with units like ps, ns, us, ms, s, m, h, or d.

See FROM_DAYS.

Convert a day number to a date. This function returns a DATE value corresponding to the number of days since year 0.

See FROM_UNIXTIME.

Convert a Unix timestamp to a datetime. This function formats a Unix timestamp as a date string or number in the current time zone.

See GET_FORMAT.

Return a format string. This function provides standard format strings for DATE_FORMAT and STR_TO_DATE based on regions like 'USA' or 'EUR'.

See HOUR.

Extract the hour. This function returns the hour portion of a time or datetime value as a number from 0 to 23.

See LAST_DAY.

Return the last day of the month. This function calculates the date of the final day for the month containing the given date.

See LOCALTIME.

Synonym for NOW(). Returns the current date and time in the session time zone.

See LOCALTIMESTAMP.

Synonym for NOW(). Returns the current date and time in the session time zone as a datetime value.

See MAKEDATE.

Create a date from a year and day of year. This function constructs a DATE value given a year and the day number within that year.

See MAKETIME.

Create a time from hour, minute, and second. This function constructs a TIME value from three numeric arguments.

See MICROSECOND.

Extract microseconds. This function returns the microsecond part of a time or datetime expression as a number from 0 to 999999.

See Microseconds in MariaDB.

Understand microsecond precision. This concept page explains how MariaDB stores and handles fractional seconds in time data types.

See MINUTE.

Extract the minute. This function returns the minute portion of a time or datetime value as a number from 0 to 59.

See MONTH.

Extract the month. This function returns the month portion of a date as a number from 1 (January) to 12 (December).

See MONTHNAME.

Return the name of the month. This function returns the full name of the month, such as ‘January’ or ‘December’, for a given date.

See MONTHS_BETWEEN.

Calculate the difference between two months.

See NOW.

Complete NOW() function reference: NOW(precision) and CURRENT_TIMESTAMP synonyms, TIMESTAMP vs DATETIME types, timezone/DST handling, and fractional seconds.

See PERIOD_ADD.

Add months to a period. This function adds a specified number of months to a period formatted as YYMM or YYYYMM.

See PERIOD_DIFF.

Calculate the difference between periods. This function returns the number of months between two periods formatted as YYMM or YYYYMM.

See QUARTER.

Return the quarter of the year. This function returns a number from 1 to 4 indicating the quarter for a given date.

See SEC_TO_TIME.

Convert seconds to time. This function returns a TIME value corresponding to the number of seconds elapsed from the start of the day.

See SECOND.

Extract the second. This function returns the second portion of a time or datetime value as a number from 0 to 59.

See STR_TO_DATE.

Complete STR_TO_DATE() reference: parse strings to DATE/TIME/DATETIME, format specifiers (%Y %m %d %H %i %s), invalid input handling, and SQL_MODE errors.

See SUBDATE.

Subtract a time interval from a date. This synonym for DATE_SUB calculates a past date by subtracting a specified unit from a starting value.

See SUBTIME.

Subtract a time value. This function subtracts one time or datetime expression from another and returns the result.

See SYSDATE.

Return the time of execution. Unlike NOW(), which returns the start time of the statement, SYSDATE() returns the time it executes.

See TIME.

Extract the time portion. This function returns the time part of a time or datetime expression.

See TIME_FORMAT.

Format a time. This function formats a time value according to a format string, similar to DATE_FORMAT but for time values.

See TIME_TO_SEC.

TIME_TO_SEC() converts a time value to the number of seconds, returning a DOUBLE that preserves microseconds.

See TIMEDIFF.

Subtract two time values. This function calculates the difference between two time or datetime expressions.

See TIMESTAMP.

Convert to datetime or add time. With one argument, it returns a datetime; with two, it adds a time expression to a date or datetime.

See TIMESTAMPADD.

Add an interval to a timestamp. This function adds a specified integer number of units (like MONTH or SECOND) to a datetime expression.

See TIMESTAMPDIFF.

Calculate the difference between timestamps. This function returns the difference between two datetime expressions in the specified unit.

See TO_DATE.

TO_DATE() converts a string to a date using a specified format, with optional handling for conversion errors.

See TO_DAYS.

Convert a date to a day number. This function returns the number of days between year 0 and the given date.

See TO_SECONDS.

Convert a date to seconds. This function returns the number of seconds from year 0 to the given date or datetime.

See TRUNC.

Truncate a date. In Oracle mode, this function truncates a date value to a specified unit of measure.

See UNIX_TIMESTAMP.

Return a Unix timestamp. This function returns the number of seconds since the Unix Epoch (‘1970-01-01 00:00:00’ UTC).

See UTC_DATE.

Return the current UTC date. This function returns the current Coordinated Universal Time date in ‘YYYY-MM-DD’ or YYYYMMDD format.

See UTC_TIME.

Return the current UTC time. This function returns the current Coordinated Universal Time in ‘HH:MM:SS’ or HHMMSS format.

See UTC_TIMESTAMP.

Return the current UTC timestamp. This function returns the current Coordinated Universal Time date and time.

See WEEK.

Return the week number. This function returns the week number for a date, with an optional mode to define the start of the week.

See WEEKDAY.

Return the weekday index. This function returns the index of the day of the week (0=Monday, 6=Sunday).

See WEEKOFYEAR.

Return the calendar week. This function returns the week number of the date (1-53), equivalent to WEEK(date, 3).

See YEAR.

Extract the year. This function returns the year portion of a date as a number from 1000 to 9999.

See YEARWEEK.

Return the year and week. This function returns the year and week number for a date, useful for grouping results by week.

This page is licensed: CC BY-SA / Gnu FDL

Date and Time Units

The `INTERVAL` keyword can be used to add or subtract a time interval of time to a DATETIME, DATE or TIME value.

The syntax is:

```
INTERVAL time_quantity time_unit
```

For example, the `SECOND` unit is used below by the DATE_ADD() function:

```
SELECT '2008-12-31 23:59:59' + INTERVAL 1 SECOND;
+----------------------------------------+
| '2008-12-31 23:59:59' + INTERVAL 1 SECOND |
+----------------------------------------+
| 2009-01-01 00:00:00                    |
+----------------------------------------+
```

The following units are valid:

Unit	Description
MICROSECOND	Microseconds
SECOND	Seconds
MINUTE	Minutes
HOUR	Hours
DAY	Days
WEEK	Weeks
MONTH	Months
QUARTER	Quarters
YEAR	Years
SECOND_MICROSECOND	Seconds.Microseconds
MINUTE_MICROSECOND	Minutes.Seconds.Microseconds
MINUTE_SECOND	Minutes.Seconds
HOUR_MICROSECOND	Hours.Minutes.Seconds.Microseconds
HOUR_SECOND	Hours.Minutes.Seconds
HOUR_MINUTE	Hours.Minutes
DAY_MICROSECOND	Days Hours.Minutes.Seconds.Microseconds
DAY_SECOND	Days Hours.Minutes.Seconds
DAY_MINUTE	Days Hours.Minutes

Unit	Description
DAY_HOUR	Days Hours
YEAR_MONTH	Years-Months

The time units containing an underscore are composite; that is, they consist of multiple base time units. For base time units, `time_quantity` is an integer number. For composite units, the quantity must be expressed as a string with multiple integer numbers separated by any punctuation character.

Example of composite units:

```
INTERVAL '2:2' YEAR_MONTH
INTERVAL '1:30:30' HOUR_SECOND
INTERVAL '1!30!30' HOUR_SECOND -- same as above
```

Time units can be used in the following contexts:

- after a + or a - operator;
- with the following `DATE` or `TIME` functions: ADDDATE(), SUBDATE(), DATE_ADD(), DATE_SUB(), TIMESTAMPADD(), TIMESTAMPDIFF(), EXTRACT();
- in the `ON SCHEDULE` clause of CREATE EVENT and ALTER EVENT;
- when defining a partitioning `BY SYSTEM_TIME` .

See Also

- Date and time literals
- Operator Precedence

This page is licensed: CC BY-SA / Gnu FDL

ADD_MONTHS

> **Note:**
>
> `ADD_MONTHS` is available from MariaDB 10.6.1.

The `ADD_MONTHS` function was introduced to enhance Oracle compatibility. Similar functionality can be achieved with the DATE_ADD function.

Syntax

```
ADD_MONTHS(date, months)
```

Description

`ADD_MONTHS` adds an integer *months* to a given *date* (DATE, DATETIME or TIMESTAMP), returning the resulting date.

`months` can be positive or negative. If months is not a whole number, it is rounded to the nearest whole number (not truncated).

The resulting day component will remain the same as that specified in *date*, unless the resulting month has fewer days than the day component of the given date, in which case the day will be the last day of the resulting month.

Returns `NULL` if given an invalid date or a `NULL` argument.

Examples

```
SELECT ADD_MONTHS('2012-01-31', 2);
+----------------------------+
| ADD_MONTHS('2012-01-31', 2) |
+----------------------------+
| 2012-03-31                 |
+----------------------------+

SELECT ADD_MONTHS('2012-01-31', -5);
+-----------------------------+
| ADD_MONTHS('2012-01-31', -5) |
+-----------------------------+
| 2011-08-31                  |
+-----------------------------+

SELECT ADD_MONTHS('2011-01-31', 1);
+----------------------------+
| ADD_MONTHS('2011-01-31', 1) |
+----------------------------+
| 2011-02-28                 |
+----------------------------+

SELECT ADD_MONTHS('2012-01-31', 1);
+----------------------------+
| ADD_MONTHS('2012-01-31', 1) |
+----------------------------+
| 2012-02-29                 |
+----------------------------+

SELECT ADD_MONTHS('2012-01-31', 2);
+----------------------------+
| ADD_MONTHS('2012-01-31', 2) |
+----------------------------+
| 2012-03-31                 |
+----------------------------+

SELECT ADD_MONTHS('2012-01-31', 3);
+----------------------------+
| ADD_MONTHS('2012-01-31', 3) |
+----------------------------+
| 2012-04-30                 |
+----------------------------+

SELECT ADD_MONTHS('2011-01-15', 2.5);
+-----------------------------+
| ADD_MONTHS('2011-01-15', 2.5) |
+-----------------------------+
| 2011-04-15                  |
+-----------------------------+
1 row in set (0.001 sec)

SELECT ADD_MONTHS('2011-01-15', 2.6);
+-----------------------------+
| ADD_MONTHS('2011-01-15', 2.6) |
+-----------------------------+
| 2011-04-15                  |
+-----------------------------+
1 row in set (0.001 sec)

SELECT ADD_MONTHS('2011-01-15', 2.1);
+-----------------------------+
| ADD_MONTHS('2011-01-15', 2.1) |
+-----------------------------+
| 2011-03-15                  |
+-----------------------------+
1 row in set (0.004 sec)
```

See Also

- SQL_MODE=ORACLE

This page is licensed: CC BY-SA / Gnu FDL

ADDDATE

Syntax

```
ADDDATE(date,INTERVAL expr unit), ADDDATE(expr,days)
```

Description

When invoked with the `INTERVAL` form of the second argument, `ADDDATE()` is a synonym for DATE_ADD(). The related function SUBDATE() is a synonym for DATE_SUB(). For information on the `INTERVAL` unit argument, see the discussion for DATE_ADD().

When invoked with the days form of the second argument, MariaDB treats it as an integer number of days to be added to *expr*.

Examples

```
SELECT DATE_ADD('2008-01-02', INTERVAL 31 DAY);
+----------------------------------------+
| DATE_ADD('2008-01-02', INTERVAL 31 DAY) |
+----------------------------------------+
| 2008-02-02                             |
+----------------------------------------+

SELECT ADDDATE('2008-01-02', INTERVAL 31 DAY);
+----------------------------------------+
| ADDDATE('2008-01-02', INTERVAL 31 DAY) |
+----------------------------------------+
| 2008-02-02                             |
+----------------------------------------+
```

```
SELECT ADDDATE('2008-01-02', 31);
+---------------------------+
| ADDDATE('2008-01-02', 31) |
+---------------------------+
| 2008-02-02                |
+---------------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d, ADDDATE(d, 10) from t1;
+---------------------+---------------------+
| d                   | ADDDATE(d, 10)      |
+---------------------+---------------------+
| 2007-01-30 21:31:07 | 2007-02-09 21:31:07 |
| 1983-10-15 06:42:51 | 1983-10-25 06:42:51 |
| 2011-04-21 12:34:56 | 2011-05-01 12:34:56 |
| 2011-10-30 06:31:41 | 2011-11-09 06:31:41 |
| 2011-01-30 14:03:25 | 2011-02-09 14:03:25 |
| 2004-10-07 11:19:34 | 2004-10-17 11:19:34 |
+---------------------+---------------------+

SELECT d, ADDDATE(d, INTERVAL 10 HOUR) from t1;
+---------------------+-----------------------------+
| d                   | ADDDATE(d, INTERVAL 10 HOUR) |
+---------------------+-----------------------------+
| 2007-01-30 21:31:07 | 2007-01-31 07:31:07          |
| 1983-10-15 06:42:51 | 1983-10-15 16:42:51          |
| 2011-04-21 12:34:56 | 2011-04-21 22:34:56          |
| 2011-10-30 06:31:41 | 2011-10-30 16:31:41          |
| 2011-01-30 14:03:25 | 2011-01-31 00:03:25          |
| 2004-10-07 11:19:34 | 2004-10-07 21:19:34          |
+---------------------+-----------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ADDTIME

Syntax

```
ADDTIME(expr1,expr2)
```

Description

`ADDTIME()` adds *expr2* to *expr1* and returns the result. *expr1* is a time or datetime expression, and *expr2* is a time expression.

Examples

```
SELECT ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002');
+----------------------------------------------------------+
| ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002') |
+----------------------------------------------------------+
| 2008-01-02 01:01:01.000001                               |
+----------------------------------------------------------+

SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');
+---------------------------------------------+
| ADDTIME('01:00:00.999999', '02:00:00.999998') |
+---------------------------------------------+
| 03:00:01.999997                             |
+---------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

CONVERT_TZ

Syntax

```
CONVERT_TZ(dt,from_tz,to_tz)
```

Description

`CONVERT_TZ()` converts a datetime value *dt* from the time zone given by *from_tz* to the time zone given by *to_tz* and returns the resulting value.

In order to use named time zones, such as GMT, MET or Africa/Johannesburg, the `time_zone` tables must be loaded (see mysql_tzinfo_to_sql).

No conversion takes place if the value falls outside of the supported `TIMESTAMP` range when converted from *from_tz* to UTC.

> **CURRENT**
>
> The supported range is `1970-01-01 00:00:00` to `2106-02-07 06:28:15` UTC.
>
> **< 11.5**
>
> The supported range is `1970-01-01 00:00:01` to `2038-01-19 05:14:07` UTC.

This function returns `NULL` if the arguments are invalid (or named time zones have not been loaded).

See time zones for more information.

Examples

```
SELECT CONVERT_TZ('2016-01-01 12:00:00','+00:00','+10:00');
+---------------------------------------------------+
| CONVERT_TZ('2016-01-01 12:00:00','+00:00','+10:00') |
+---------------------------------------------------+
| 2016-01-01 22:00:00                               |
+---------------------------------------------------+
```

Using named time zones (with the time zone tables loaded):

```
SELECT CONVERT_TZ('2016-01-01 12:00:00','GMT','Africa/Johannesburg');
+-------------------------------------------------------------+
| CONVERT_TZ('2016-01-01 12:00:00','GMT','Africa/Johannesburg') |
+-------------------------------------------------------------+
| 2016-01-01 14:00:00                                         |
+-------------------------------------------------------------+
```

The value is out of the `TIMESTAMP` range, so no conversion takes place:

```
SELECT CONVERT_TZ('1969-12-31 22:00:00','+00:00','+10:00');
+---------------------------------------------------+
| CONVERT_TZ('1969-12-31 22:00:00','+00:00','+10:00') |
+---------------------------------------------------+
| 1969-12-31 22:00:00                               |
+---------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

CURDATE

Syntax

```
CURDATE()
CURRENT_DATE
CURRENT_DATE()
```

Description

`CURDATE` returns the current date as a value in `YYYY-MM-DD` or `YYYYMMDD` format, depending on whether the function is used in a string or numeric context.

`CURRENT_DATE` and `CURRENT_DATE()` are synonyms.

Examples

```
SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2019-03-05 |
+------------+
```

In a numeric context (note this is not performing date calculations):

```
SELECT CURDATE() +0;
+--------------+
| CURDATE() +0 |
+--------------+
|     20190305 |
+--------------+
```

Date calculation:

```
SELECT CURDATE() - INTERVAL 5 DAY;
+----------------------------+
| CURDATE() - INTERVAL 5 DAY |
+----------------------------+
| 2019-02-28                 |
+----------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

CURRENT_DATE

Syntax

```
CURRENT_DATE, CURRENT_DATE()
```

Description

`CURRENT_DATE` and `CURRENT_DATE()` are synonyms for CURDATE().

This page is licensed: GPLv2, originally from fill_help_tables.sql

CURRENT_TIME

Syntax

```
CURRENT_TIME
CURRENT_TIME([precision])
```

Description

`CURRENT_TIME` and `CURRENT_TIME()` are synonyms for CURTIME().

See Also

- Microseconds in MariaDB

This page is licensed: GPLv2, originally from fill_help_tables.sql

CURRENT_TIMESTAMP

Syntax

```
CURRENT_TIMESTAMP
CURRENT_TIMESTAMP([precision])
```

Description

`CURRENT_TIMESTAMP` and `CURRENT_TIMESTAMP()` are synonyms for NOW().

See Also

- Microseconds in MariaDB
- The TIMESTAMP data type

This page is licensed: GPLv2, originally from fill_help_tables.sql

CURTIME

Syntax

```
CURTIME([precision])
```

Description

Returns the current time as a value in `HH:MM:SS` or `HHMMSS.uuuuuu` format, depending on whether the function is used in a string or numeric context. The value is expressed in the current time zone.

The optional *precision* determines the microsecond precision. See Microseconds in MariaDB.

Examples

```
SELECT CURTIME();
+-----------+
| CURTIME() |
+-----------+
| 12:45:39  |
+-----------+

SELECT CURTIME() + 0;
+---------------+
| CURTIME() + 0 |
+---------------+
| 124545.000000 |
+---------------+
```

With precision:

```
SELECT CURTIME(2);
+-------------+
| CURTIME(2)  |
+-------------+
| 09:49:08.09 |
+-------------+
```

See Also

- Microseconds in MariaDB

This page is licensed: GPLv2, originally from fill_help_tables.sql

DATE FUNCTION

Syntax

```
DATE(expr)
```

Description

Extracts the date part of the date or datetime expression *expr*. Returns NULL and throws a warning when passed an invalid date.

Examples

```
SELECT DATE('2013-07-18 12:21:32');
+----------------------------+
| DATE('2013-07-18 12:21:32') |
+----------------------------+
| 2013-07-18                 |
+----------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DATE_ADD

Syntax

```
DATE_ADD(date,INTERVAL expr unit)
```

Description

Performs date arithmetic. The *date* argument specifies the starting date or datetime value. *expr* is an expression specifying the interval value to be added to the starting date. *expr* is a string; it may start with a "`-`" for negative intervals. For composite interval units (such as DAY_HOUR, MINUTE_SECOND, and SECOND_MICROSECOND), *expr* can contain multiple components separated by spaces or punctuation. For example:

- '1 10' DAY_HOUR represents 1 day and 10 hours
- '1:1' MINUTE_SECOND represents 1 minute and 1 second
- '1.999999' SECOND_MICROSECOND represents seconds and microseconds

The exact format of *expr* depends on the *unit* specified.

unit is a keyword indicating the units in which the expression should be interpreted. See Date and Time Units for a complete list of permitted units.

The result type of `DATE_ADD()` is determined as follows:

- if the first argument is of the type `DATETIME`, the function returns `DATETIME`;
- if the first argument is `DATE` and the interval uses `HOUR` or smaller units, the function returns `DATETIME`;
- if the first argument is `DATE` and the interval uses `DAY` or larger units, the function returns `DATE`;
- similarly, if the first argument is `TIME` and the interval uses `DAY` or smaller units the function returns `TIME`, if the interval uses anything larger, the function returns `DATETIME`;

- if the first argument isn't a temporal type, the function returns a string.

Examples

```
SELECT '2008-12-31 23:59:59' + INTERVAL 1 SECOND;
+-----------------------------------------+
| '2008-12-31 23:59:59' + INTERVAL 1 SECOND |
+-----------------------------------------+
| 2009-01-01 00:00:00                     |
+-----------------------------------------+
```

```
SELECT INTERVAL 1 DAY + '2008-12-31';
+-------------------------------+
| INTERVAL 1 DAY + '2008-12-31' |
+-------------------------------+
| 2009-01-01                    |
+-------------------------------+
```

```
SELECT '2005-01-01' - INTERVAL 1 SECOND;
+---------------------------------+
| '2005-01-01' - INTERVAL 1 SECOND |
+---------------------------------+
| 2004-12-31 23:59:59             |
+---------------------------------+
```

```
SELECT DATE_ADD('2000-12-31 23:59:59', INTERVAL 1 SECOND);
+-------------------------------------------------+
| DATE_ADD('2000-12-31 23:59:59', INTERVAL 1 SECOND) |
+-------------------------------------------------+
| 2001-01-01 00:00:00                             |
+-------------------------------------------------+
```

```
SELECT DATE_ADD('2010-12-31 23:59:59', INTERVAL 1 DAY);
+----------------------------------------------+
| DATE_ADD('2010-12-31 23:59:59', INTERVAL 1 DAY) |
+----------------------------------------------+
| 2011-01-01 23:59:59                          |
+----------------------------------------------+
```

```
SELECT DATE_ADD('2100-12-31 23:59:59', INTERVAL '1:1' MINUTE_SECOND);
+---------------------------------------------------------------+
| DATE_ADD('2100-12-31 23:59:59', INTERVAL '1:1' MINUTE_SECOND) |
+---------------------------------------------------------------+
| 2101-01-01 00:01:00                                           |
+---------------------------------------------------------------+
```

```
SELECT DATE_ADD('1900-01-01 00:00:00', INTERVAL '-1 10' DAY_HOUR);
+-------------------------------------------------------------+
| DATE_ADD('1900-01-01 00:00:00', INTERVAL '-1 10' DAY_HOUR) |
+-------------------------------------------------------------+
| 1899-12-30 14:00:00                                         |
+-------------------------------------------------------------+
```

```
SELECT DATE_ADD('1992-12-31 23:59:59.000002', INTERVAL '1.999999' SECOND_MICROSECOND);
+----------------------------------------------------------------------------------+
| DATE_ADD('1992-12-31 23:59:59.000002', INTERVAL '1.999999' SECOND_MICROSECOND) |
+----------------------------------------------------------------------------------+
| 1993-01-01 00:00:01.000001                                                       |
+----------------------------------------------------------------------------------+
```

See Also

- DATE_SUB
- ADD_MONTHS

This page is licensed: GPLv2, originally from fill_help_tables.sql

DATE_FORMAT

Syntax

```
DATE_FORMAT(date, format[, locale])
```

Description

Formats the date value according to the format string.

The language used for the names is controlled by the value of the lc_time_names system variable. See server locale for more on the supported locales.

Date Formatting Options

The options that can be used by `DATE_FORMAT()`, as well as its inverse STR_TO_DATE() and the FROM_UNIXTIME() function, are:

Option	Description
%a	Short weekday name in current locale (Variable lc_time_names).
%b	Short form month name in current locale. For locale `en_US` this is one of: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, or Dec.
%c	Month with 1 or 2 digits.
%D	Day with English suffix ‘th’, ‘nd’, ‘st’ or ‘rd’’. (1st, 2nd, 3rd…).
%d	Day with 2 digits.
%e	Day with 1 or 2 digits.
%f	Microseconds 6 digits.
%H	Hour with 2 digits between 00-23.
%h	Hour with 2 digits between 01-12.
%I	Hour with 2 digits between 01-12.
%i	Minute with 2 digits.
%j	Day of the year (001-366)
%k	Hour with 1 digits between 0-23.
%l	Hour with 1 digits between 1-12.
%M	Full month name in current locale (Variable lc_time_names).
%m	Month with 2 digits.
%p	AM/PM according to current locale (Variable lc_time_names).
%r	Time in 12 hour format, followed by AM/PM. Short for ‘%I:%i:%S %p’.
%S	Seconds with 2 digits.
%s	Seconds with 2 digits.
%T	Time in 24 hour format. Short for ‘%H:%i:%S’.

Option	Description
%U	Week number (00-53), when first day of the week is Sunday.
%u	Week number (00-53), when first day of the week is Monday.
%V	Week number (01-53), when first day of the week is Sunday. Used with %X.
%v	Week number (01-53), when first day of the week is Monday. Used with %x.
%W	Full weekday name in current locale (Variable lc_time_names).
%w	Day of the week. 0 = Sunday, 6 = Saturday.
%X	Year with 4 digits when first day of the week is Sunday. Used with %V.
%x	Year with 4 digits when first day of the week is Monday. Used with %v.
%Y	Year with 4 digits.
%y	Year with 2 digits.
%Z	Timezone abbreviation. From MariaDB 11.3.0.
%z	Numeric timezone +hhmm or -hhmm presenting the hour and minute offset from UTC. From MariaDB 11.3.0.
%#	For str_to_date(), skip all numbers.
%.	For str_to_date(), skip all punctuation characters.
%@	For str_to_date(), skip all alpha characters.
%%	A literal % character.

To get a date in one of the standard formats, GET_FORMAT() can be used.

Examples

```
SELECT DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y');
+-----------------------------------------------+
| DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y') |
+-----------------------------------------------+
| Sunday October 2009                           |
+-----------------------------------------------+

SELECT DATE_FORMAT('2007-10-04 22:23:00', '%H:%i:%s');
+-----------------------------------------------+
| DATE_FORMAT('2007-10-04 22:23:00', '%H:%i:%s') |
+-----------------------------------------------+
| 22:23:00                                      |
+-----------------------------------------------+

SELECT DATE_FORMAT('1900-10-04 22:23:00', '%D %y %a %d %m %b %j');
+------------------------------------------------------------+
| DATE_FORMAT('1900-10-04 22:23:00', '%D %y %a %d %m %b %j') |
+------------------------------------------------------------+
| 4th 00 Thu 04 10 Oct 277                                   |
+------------------------------------------------------------+

SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w');
+------------------------------------------------------------+
| DATE_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w') |
+------------------------------------------------------------+
| 22 22 10 10:23:00 PM 22:23:00 00 6                         |
+------------------------------------------------------------+

SELECT DATE_FORMAT('1999-01-01', '%X %V');
+-----------------------------------+
| DATE_FORMAT('1999-01-01', '%X %V') |
+-----------------------------------+
| 1998 52                           |
+-----------------------------------+

SELECT DATE_FORMAT('2006-06-00', '%d');
+---------------------------------+
| DATE_FORMAT('2006-06-00', '%d') |
+---------------------------------+
| 00                              |
+---------------------------------+
```

Optionally, the locale can be explicitly specified as the third `DATE_FORMAT()` argument. Doing so makes the function independent from the session settings, and the three argument version of `DATE_FORMAT()` can be used in virtual indexed and persistent generated-columns:

```
SELECT DATE_FORMAT('2006-01-01', '%W', 'el_GR');
+-----------------------------------------+
| DATE_FORMAT('2006-01-01', '%W', 'el_GR') |
+-----------------------------------------+
| Κυριακή                                 |
+-----------------------------------------+
```

Format the current date and time in German date-time format:

```
SELECT DATE_FORMAT(NOW(),'%W, %e. %M %Y, %k:%i Uhr (%Z)','de_DE');
+------------------------------------------------------------+
| DATE_FORMAT(NOW(),'%W, %e. %M %Y, %k:%i Uhr (%Z)','de_DE') |
+------------------------------------------------------------+
| Donnerstag, 5. February 2026, 20:50 Uhr (CET)              |
+------------------------------------------------------------+
```

> **Note:**
>
> Timezone information from MariaDB 11.3:

```
SELECT DATE_FORMAT(NOW(), '%W %d %M %Y %H:%i:%s %Z %z');
+-------------------------------------------------+
| DATE_FORMAT(NOW(), '%W %d %M %Y %H:%i:%s %Z %z') |
+-------------------------------------------------+
| Wednesday 20 September 2023 15:00:23 SAST +0200  |
+-------------------------------------------------+
```

See Also

- STR_TO_DATE()
- FROM_UNIXTIME()

This page is licensed: GPLv2, originally from fill_help_tables.sql

DATE_SUB

Syntax

```
DATE_SUB(date,INTERVAL expr unit)
```

Description

Performs date arithmetic. The *date* argument specifies the starting date or datetime value. *expr* is an expression specifying the interval value to be subtracted from the starting date. *expr* is a string; it may start with a "`-`" for negative intervals. *unit* is a keyword indicating the units in which the expression should be interpreted. See Date and Time Units for a complete list of permitted units.

See also DATE_ADD().

Examples

```
SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
+----------------------------------------+
| DATE_SUB('1998-01-02', INTERVAL 31 DAY) |
+----------------------------------------+
| 1997-12-02                             |
+----------------------------------------+
```

```
SELECT DATE_SUB('2005-01-01 00:00:00', INTERVAL '1 1:1:1' DAY_SECOND);
+-----------------------------------------------------------------+
| DATE_SUB('2005-01-01 00:00:00', INTERVAL '1 1:1:1' DAY_SECOND) |
+-----------------------------------------------------------------+
| 2004-12-30 22:58:59                                             |
+-----------------------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DATEDIFF

Syntax

```
DATEDIFF(expr1,expr2)
```

Description

`DATEDIFF()` returns (*expr1* – *expr2*) expressed as a value in days from one date to the other. *expr1* and *expr2* are date or date-and-time expressions. Only the date parts of the values are used in the calculation.

Examples

```
SELECT DATEDIFF('2007-12-31 23:59:59','2007-12-30');
+----------------------------------------------+
| DATEDIFF('2007-12-31 23:59:59','2007-12-30') |
+----------------------------------------------+
|                                            1 |
+----------------------------------------------+

SELECT DATEDIFF('2010-11-30 23:59:59','2010-12-31');
+----------------------------------------------+
| DATEDIFF('2010-11-30 23:59:59','2010-12-31') |
+----------------------------------------------+
|                                          -31 |
+----------------------------------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2011-05-23 10:56:05 |
+---------------------+

SELECT d, DATEDIFF(NOW(),d) FROM t1;
+---------------------+-------------------+
| d                   | DATEDIFF(NOW(),d) |
+---------------------+-------------------+
| 2007-01-30 21:31:07 |              1574 |
| 1983-10-15 06:42:51 |             10082 |
| 2011-04-21 12:34:56 |                32 |
| 2011-10-30 06:31:41 |              -160 |
| 2011-01-30 14:03:25 |               113 |
| 2004-10-07 11:19:34 |              2419 |
+---------------------+-------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DAY

Syntax

```
DAY(date)
```

Description

`DAY()` is a synonym for DAYOFMONTH().

This page is licensed: GPLv2, originally from fill_help_tables.sql

DAYNAME

Syntax

```
DAYNAME(date)
```

Description

Returns the name of the weekday for date. The language used for the name is controlled by the value of the lc_time_names system variable. See server locale for more on the supported locales.

Examples

```
SELECT DAYNAME('2007-02-03');
+----------------------+
| DAYNAME('2007-02-03') |
+----------------------+
| Saturday              |
+----------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d, DAYNAME(d) FROM t1;
+---------------------+------------+
| d                   | DAYNAME(d) |
+---------------------+------------+
| 2007-01-30 21:31:07 | Tuesday    |
| 1983-10-15 06:42:51 | Saturday   |
| 2011-04-21 12:34:56 | Thursday   |
| 2011-10-30 06:31:41 | Sunday     |
| 2011-01-30 14:03:25 | Sunday     |
| 2004-10-07 11:19:34 | Thursday   |
+---------------------+------------+
```

Changing the locale:

```
SET lc_time_names = 'fr_CA';

SELECT DAYNAME('2013-04-01');
+----------------------+
| DAYNAME('2013-04-01') |
+----------------------+
| lundi                 |
+----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DAYOFMONTH

Syntax

```
DAYOFMONTH(date)
```

Description

Returns the day of the month for date, in the range `1` to `31`, or `0` for dates such as `'0000-00-00'` or `'2008-00-00'` which have a zero day part.

`DAY()` is a synonym.

Examples

```
SELECT DAYOFMONTH('2007-02-03');
+--------------------------+
| DAYOFMONTH('2007-02-03') |
+--------------------------+
|                        3 |
+--------------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d FROM t1 where DAYOFMONTH(d) = 30;
+---------------------+
| d                   |
+---------------------+
| 2007-01-30 21:31:07 |
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
+---------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DAYOFWEEK

Syntax

```
DAYOFWEEK(date)
```

Description

Returns the day of the week index for the date (1 = Sunday, 2 = Monday, …, 7 = Saturday). These index values correspond to the ODBC standard.

This contrasts with WEEKDAY() which follows a different index numbering (`0` = Monday, `1` = Tuesday, … `6` = Sunday).

Examples

```
SELECT DAYOFWEEK('2007-02-03');
+-------------------------+
| DAYOFWEEK('2007-02-03') |
+-------------------------+
|                       7 |
+-------------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d, DAYNAME(d), DAYOFWEEK(d), WEEKDAY(d) from t1;
+---------------------+------------+--------------+------------+
| d                   | DAYNAME(d) | DAYOFWEEK(d) | WEEKDAY(d) |
+---------------------+------------+--------------+------------+
| 2007-01-30 21:31:07 | Tuesday    |            3 |          1 |
| 1983-10-15 06:42:51 | Saturday   |            7 |          5 |
| 2011-04-21 12:34:56 | Thursday   |            5 |          3 |
| 2011-10-30 06:31:41 | Sunday     |            1 |          6 |
| 2011-01-30 14:03:25 | Sunday     |            1 |          6 |
| 2004-10-07 11:19:34 | Thursday   |            5 |          3 |
+---------------------+------------+--------------+------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DAYOFYEAR

Syntax

```
DAYOFYEAR(date)
```

Description

Returns the day of the year for date, in the range 1 to 366.

Examples

```
SELECT DAYOFYEAR('2018-02-16');
+-------------------------+
| DAYOFYEAR('2018-02-16') |
+-------------------------+
|                      47 |
+-------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

EXTRACT

Syntax

```
EXTRACT(unit FROM date)
```

Description

The EXTRACT() function extracts the required unit from the date. See Date and Time Units for a complete list of permitted units.

HOUR() is not a standard SQL function, so continues to adhere to the old behavior inherited from MySQL.

Examples

```
SELECT EXTRACT(YEAR FROM '2009-07-02');
+--------------------------------+
| EXTRACT(YEAR FROM '2009-07-02') |
+--------------------------------+
|                           2009 |
+--------------------------------+

SELECT EXTRACT(YEAR_MONTH FROM '2009-07-02 01:02:03');
+-----------------------------------------------+
| EXTRACT(YEAR_MONTH FROM '2009-07-02 01:02:03') |
+-----------------------------------------------+
|                                        200907 |
+-----------------------------------------------+

SELECT EXTRACT(DAY_MINUTE FROM '2009-07-02 01:02:03');
+---------------------------------------------+
| EXTRACT(DAY_MINUTE FROM '2009-07-02 01:02:03') |
+---------------------------------------------+
|                                       20102 |
+---------------------------------------------+

SELECT EXTRACT(MICROSECOND FROM '2003-01-02 10:30:00.000123');
+-------------------------------------------------------+
| EXTRACT(MICROSECOND FROM '2003-01-02 10:30:00.000123') |
+-------------------------------------------------------+
|                                                   123 |
+-------------------------------------------------------+
```

`EXTRACT (HOUR FROM...)` returns a value from 0 to 23, as per the SQL standard. `HOUR` is not a standard function, so continues to adhere to the old behaviour inherited from MySQL.

```
SELECT EXTRACT(HOUR FROM '26:30:00'), HOUR('26:30:00');
+-------------------------------+------------------+
| EXTRACT(HOUR FROM '26:30:00') | HOUR('26:30:00') |
+-------------------------------+------------------+
|                             2 |               26 |
+-------------------------------+------------------+
```

See Also

- Date and Time Units
- Date and Time Literals
- HOUR()

This page is licensed: GPLv2, originally from fill_help_tables.sql

FORMAT_PICO_TIME

> **Note:**
>
> FORMAT_PICO_TIME is available from MariaDB 11.0.2.

Syntax

```
FORMAT_PICO_TIME(time_val)
```

Description

Given a time in picoseconds, returns a human-readable time value and unit indicator. Resulting unit is dependent on the length of the argument, and can be:

- ps - picoseconds
- ns - nanoseconds
- us - microseconds
- ms - milliseconds
- s - seconds
- min - minutes
- h - hours
- d - days

With the exception of results under one nanosecond, which are not rounded and are represented as whole numbers, the result is rounded to 2 decimal places, with a minimum of 3 significant digits.

Returns NULL if the argument is NULL.

This function is very similar to the Sys Schema FORMAT_TIME function, but with the following differences:

- Represents minutes as `min` rather than `m`.
- Does not represent weeks.

Examples

```
SELECT
    FORMAT_PICO_TIME(43) AS ps,
    FORMAT_PICO_TIME(4321) AS ns,
    FORMAT_PICO_TIME(43211234) AS us,
    FORMAT_PICO_TIME(432112344321) AS ms,
    FORMAT_PICO_TIME(43211234432123) AS s,
    FORMAT_PICO_TIME(432112344321234) AS m,
    FORMAT_PICO_TIME(4321123443212345) AS h,
    FORMAT_PICO_TIME(432112344321234545) AS d;
+--------+---------+----------+-----------+---------+----------+--------+--------+
| ps     | ns      | us       | ms        | s       | m        | h      | d      |
+--------+---------+----------+-----------+---------+----------+--------+--------+
|  43 ps | 4.32 ns | 43.21 us | 432.11 ms | 43.21 s | 7.20 min | 1.20 h | 5.00 d |
+--------+---------+----------+-----------+---------+----------+--------+--------+
```

This page is licensed: CC BY-SA / Gnu FDL

FROM_DAYS

Syntax

```
FROM_DAYS(N)
```

Description

Given a day number N, returns a DATE value. The day count is based on the number of days from the start of the standard calendar (0000-00-00).

The function is not designed for use with dates before the advent of the Gregorian calendar in October 1582. Results will not be reliable since it doesn't account for the lost days when the calendar changed from the Julian calendar.

This is the converse of the TO_DAYS() function.

Examples

```
SELECT FROM_DAYS(730669);
+-------------------+
| FROM_DAYS(730669) |
+-------------------+
| 2000-07-03        |
+-------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

FROM_UNIXTIME

Syntax

```
FROM_UNIXTIME(unix_timestamp)
FROM_UNIXTIME(unix_timestamp,format)
```

Description

Converts the number of seconds from the epoch (1970-01-01 00:00:00 UTC) to a `TIMESTAMP` value, the opposite of what UNIX_TIMESTAMP() is doing. Returns NULL if the result would be outside of the valid range of `TIMESTAMP` values.

If format is given, the result is exactly equivalent to

```
DATE_FORMAT(FROM_UNIXTIME(unix_timestamp), format)
```

CURRENT

Timestamps in MariaDB have a maximum value of 4294967295, equivalent to `2106-02-07 06:28:15`. This is due to the underlying 32-bit limitation. Using the function on a timestamp beyond this will result in NULL being returned. Use DATETIME as a storage type if you require dates beyond this.

< 11.7

The one-argument form of `FROM_UNIXTIME()` returns a `DATETIME`. This means that it can return values outside of valid `TIMESTAMP` range, in particular `1970-01-01 00:00:00`. And it can return the same result for different values of unix_timestamp (around DST changes).

Timestamps in MariaDB have a maximum value of 4294967295, equivalent to `2106-02-07 06:28:15`. This is due to the underlying 32-bit limitation. Using the function on a timestamp beyond this will result in NULL being returned. Use DATETIME as a storage type if you require dates beyond this.

< 11.5

The one-argument form of `FROM_UNIXTIME()` returns a `DATETIME`. This means that it can return values outside of valid `TIMESTAMP` range, in particular `1970-01-01 00:00:00`. And it can return the same result for different values of unix_timestamp (around DST changes).

The maximum value is 2147483647, equivalent to `2038-01-19 05:14:07`.

The following options can be used by `FROM_UNIXTIME()`, as well as DATE_FORMAT() and STR_TO_DATE():

Option	Description
%a	Short weekday name in current locale (Variable lc_time_names).
%b	Short form month name in current locale. For locale en_US this is one of: Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov or Dec.

Option	Description
%c	Month with 1 or 2 digits.
%D	Day with English suffix ‘th’, ‘nd’, ‘st’ or ‘rd’’. (1st, 2nd, 3rd…).
%d	Day with 2 digits.
%e	Day with 1 or 2 digits.
%f	Microseconds 6 digits.
%H	Hour with 2 digits between 00-23.
%h	Hour with 2 digits between 01-12.
%I	Hour with 2 digits between 01-12.
%i	Minute with 2 digits.
%j	Day of the year (001-366)
%k	Hour with 1 digits between 0-23.
%l	Hour with 1 digits between 1-12.
%M	Full month name in current locale (Variable lc_time_names).
%m	Month with 2 digits.
%p	AM/PM according to current locale (Variable lc_time_names).
%r	Time in 12 hour format, followed by AM/PM. Short for ‘%I:%i:%S %p’.
%S	Seconds with 2 digits.
%s	Seconds with 2 digits.
%T	Time in 24 hour format. Short for ‘%H:%i:%S’.
%U	Week number (00-53), when first day of the week is Sunday.
%u	Week number (00-53), when first day of the week is Monday.
%V	Week number (01-53), when first day of the week is Sunday. Used with %X.
%v	Week number (01-53), when first day of the week is Monday. Used with %x.
%W	Full weekday name in current locale (Variable lc_time_names).
%w	Day of the week. 0 = Sunday, 6 = Saturday.
%X	Year with 4 digits when first day of the week is Sunday. Used with %V.
%x	Year with 4 digits when first day of the week is Sunday. Used with %v.
%Y	Year with 4 digits.
%y	Year with 2 digits.
%#	For str_to_date(), skip all numbers.
%.	For str_to_date(), skip all punctuation characters.
%@	For str_to_date(), skip all alpha characters.
%%	A literal % character.

Performance Considerations

If your session time zone is set to `SYSTEM` (the default), `FROM_UNIXTIME()` will call the OS function to convert the data using the system time zone. At least on Linux, the corresponding function (`localtime_r`) uses a global mutex inside glibc that can cause contention under high concurrent load.

Set your time zone to a named time zone to avoid this issue. See mysql time zone tables for details on how to do this.

Examples

```
SELECT FROM_UNIXTIME(1196440219);
+---------------------------+
| FROM_UNIXTIME(1196440219) |
+---------------------------+
| 2007-11-30 11:30:19       |
+---------------------------+

SELECT FROM_UNIXTIME(1196440219) + 0;
+-------------------------------+
| FROM_UNIXTIME(1196440219) + 0 |
+-------------------------------+
|         20071130113019.000000 |
+-------------------------------+

SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(), '%Y %D %M %h:%i:%s %x');
+-----------------------------------------------------------+
| FROM_UNIXTIME(UNIX_TIMESTAMP(), '%Y %D %M %h:%i:%s %x') |
+-----------------------------------------------------------+
| 2010 27th March 01:03:47 2010                             |
+-----------------------------------------------------------+
```

See Also

- UNIX_TIMESTAMP()
- DATE_FORMAT()
- STR_TO_DATE()

This page is licensed: GPLv2, originally from fill_help_tables.sql

GET_FORMAT

Syntax

```
GET_FORMAT({DATE|DATETIME|TIME}, {'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL'})
```

Description

Returns a format string. This function is useful in combination with the DATE_FORMAT() and the STR_TO_DATE() functions.

Possible result formats are:

Function Call	Result Format
GET_FORMAT(DATE,‘EUR’)	‘%d.%m.%Y’
GET_FORMAT(DATE,‘USA’)	‘%m.%d.%Y’
GET_FORMAT(DATE,‘JIS’)	‘%Y-%m-%d’
GET_FORMAT(DATE,‘ISO’)	‘%Y-%m-%d’
GET_FORMAT(DATE,‘INTERNAL’)	‘%Y%m%d’
GET_FORMAT(DATETIME,‘EUR’)	‘%Y-%m-%d %H.%i.%s’
GET_FORMAT(DATETIME,‘USA’)	‘%Y-%m-%d %H.%i.%s’
GET_FORMAT(DATETIME,‘JIS’)	‘%Y-%m-%d %H:%i:%s’
GET_FORMAT(DATETIME,‘ISO’)	‘%Y-%m-%d %H:%i:%s’
GET_FORMAT(DATETIME,‘INTERNAL’)	‘%Y%m%d%H%i%s’
GET_FORMAT(TIME,‘EUR’)	‘%H.%i.%s’

Function Call	Result Format
GET_FORMAT(TIME,‘USA’)	‘%h:%i:%s %p’
GET_FORMAT(TIME,‘JIS’)	‘%H:%i:%s’
GET_FORMAT(TIME,‘ISO’)	‘%H:%i:%s’
GET_FORMAT(TIME,‘INTERNAL’)	‘%H%i%s’

Examples

Obtaining the string matching to the standard European date format:

```
SELECT GET_FORMAT(DATE, 'EUR');
+-------------------------+
| GET_FORMAT(DATE, 'EUR') |
+-------------------------+
| %d.%m.%Y                |
+-------------------------+
```

Using the same string to format a date:

```
SELECT DATE_FORMAT('2003-10-03',GET_FORMAT(DATE,'EUR'));
+------------------------------------------------+
| DATE_FORMAT('2003-10-03',GET_FORMAT(DATE,'EUR')) |
+------------------------------------------------+
| 03.10.2003                                     |
+------------------------------------------------+

SELECT STR_TO_DATE('10.31.2003',GET_FORMAT(DATE,'USA'));
+------------------------------------------------+
| STR_TO_DATE('10.31.2003',GET_FORMAT(DATE,'USA')) |
+------------------------------------------------+
| 2003-10-31                                     |
+------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

HOUR

Syntax

```
HOUR(time)
```

Description

Returns the hour for time. The range of the return value is 0 to 23 for time-of-day values. However, the range of `TIME` values actually is much larger, so HOUR can return values greater than 23.

The return value is always positive, even if a negative `TIME` value is provided.

Examples

```
SELECT HOUR('10:05:03');
+------------------+
| HOUR('10:05:03') |
+------------------+
|               10 |
+------------------+

SELECT HOUR('272:59:59');
+-------------------+
| HOUR('272:59:59') |
+-------------------+
|               272 |
+-------------------+
```

See Also

- Date and Time Units
- Date and Time Literals
- EXTRACT()

This page is licensed: GPLv2, originally from fill_help_tables.sql

LAST_DAY

Syntax

```
LAST_DAY(date)
```

Description

Takes a date or datetime value and returns the corresponding value for the last day of the month. Returns `NULL` if the argument is invalid.

Examples

```
SELECT LAST_DAY('2003-02-05');
+------------------------+
| LAST_DAY('2003-02-05') |
+------------------------+
| 2003-02-28             |
+------------------------+

SELECT LAST_DAY('2004-02-05');
+------------------------+
| LAST_DAY('2004-02-05') |
+------------------------+
| 2004-02-29             |
+------------------------+

SELECT LAST_DAY('2004-01-01 01:01:01');
+---------------------------------+
| LAST_DAY('2004-01-01 01:01:01') |
+---------------------------------+
| 2004-01-31                      |
+---------------------------------+

SELECT LAST_DAY('2003-03-32');
+------------------------+
| LAST_DAY('2003-03-32') |
+------------------------+
| NULL                   |
+------------------------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1292): Incorrect datetime value: '2003-03-32'
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOCALTIME

Syntax

```
LOCALTIME
LOCALTIME([precision])
```

Description

`LOCALTIME` and `LOCALTIME()` are synonyms for NOW().

See Also

- Microseconds in MariaDB

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOCALTIMESTAMP

Syntax

```
LOCALTIMESTAMP
LOCALTIMESTAMP([precision])
```

Description

`LOCALTIMESTAMP` and `LOCALTIMESTAMP()` are synonyms for NOW().

See Also

- Microseconds in MariaDB

This page is licensed: GPLv2, originally from fill_help_tables.sql

MAKEDATE

Syntax

```
MAKEDATE(year,dayofyear)
```

Description

Returns a date, given `year` and `day-of-year values`. `dayofyear` must be greater than 0 or the result is `NULL`.

Examples

```
SELECT MAKEDATE(2011,31), MAKEDATE(2011,32);
+------------------+------------------+
| MAKEDATE(2011,31) | MAKEDATE(2011,32) |
+------------------+------------------+
| 2011-01-31        | 2011-02-01        |
+------------------+------------------+
```

2012 is a leap year:

```
SELECT MAKEDATE(2011,365), MAKEDATE(2012,365);
+------------------+------------------+
| MAKEDATE(2011,365) | MAKEDATE(2012,365) |
+------------------+------------------+
| 2011-12-31         | 2012-12-30         |
+------------------+------------------+

SELECT MAKEDATE(2011,366), MAKEDATE(2012,366);
+------------------+------------------+
| MAKEDATE(2011,366) | MAKEDATE(2012,366) |
+------------------+------------------+
| 2012-01-01         | 2012-12-31         |
+------------------+------------------+

SELECT MAKEDATE(2011,0);
+-----------------+
| MAKEDATE(2011,0) |
+-----------------+
| NULL             |
+-----------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MAKETIME

Syntax

```
MAKETIME(hour,minute,second)
```

Description

Returns a time value calculated from the `hour`, `minute`, and `second` arguments.

If `minute` or `second` are out of the range 0 to 60, `NULL` is returned. The `hour` can be in the range -838 to 838, outside of which the value is truncated with a warning.

Examples

```
SELECT MAKETIME(13,57,33);
+--------------------+
| MAKETIME(13,57,33) |
+--------------------+
| 13:57:33           |
+--------------------+

SELECT MAKETIME(-13,57,33);
+---------------------+
| MAKETIME(-13,57,33) |
+---------------------+
| -13:57:33           |
+---------------------+

SELECT MAKETIME(13,67,33);
+--------------------+
| MAKETIME(13,67,33) |
+--------------------+
| NULL               |
+--------------------+

SELECT MAKETIME(-1000,57,33);
+-----------------------+
| MAKETIME(-1000,57,33) |
+-----------------------+
| -838:59:59            |
+-----------------------+
1 row in set, 1 warning (0.00 sec)

SHOW WARNINGS;
+---------+------+-----------------------------------------------+
| Level   | Code | Message                                       |
+---------+------+-----------------------------------------------+
| Warning | 1292 | Truncated incorrect time value: '-1000:57:33' |
+---------+------+-----------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MICROSECOND

Syntax

```
MICROSECOND(expr)
```

Description

Returns the microseconds from the time or datetime expression *expr* as a number in the range from 0 to 999999.

If *expr* is a time with no microseconds, zero is returned, while if *expr* is a date with no time, zero with a warning is returned.

Examples

```
SELECT MICROSECOND('12:00:00.123456');
+-------------------------------+
| MICROSECOND('12:00:00.123456') |
+-------------------------------+
|                        123456 |
+-------------------------------+

SELECT MICROSECOND('2009-12-31 23:59:59.000010');
+----------------------------------------+
| MICROSECOND('2009-12-31 23:59:59.000010') |
+----------------------------------------+
|                                     10 |
+----------------------------------------+

SELECT MICROSECOND('2013-08-07 12:13:14');
+-----------------------------------+
| MICROSECOND('2013-08-07 12:13:14') |
+-----------------------------------+
|                                 0 |
+-----------------------------------+

SELECT MICROSECOND('2013-08-07');
+---------------------------+
| MICROSECOND('2013-08-07') |
+---------------------------+
|                         0 |
+---------------------------+
1 row in set, 1 warning (0.00 sec)

SHOW WARNINGS;
+---------+------+-----------------------------------------------+
| Level   | Code | Message                                       |
+---------+------+-----------------------------------------------+
| Warning | 1292 | Truncated incorrect time value: '2013-08-07' |
+---------+------+-----------------------------------------------+
```

See Also

- Microseconds in MariaDB

This page is licensed: GPLv2, originally from fill_help_tables.sql

Microseconds in MariaDB

The TIME, DATETIME, and TIMESTAMP types, along with the temporal functions, CAST and dynamic columns, support microseconds. The datetime precision of a column can be specified when creating the table with CREATE TABLE, for example:

```
CREATE TABLE example(
  col_microsec DATETIME(6),
  col_millisec TIME(3)
);
```

Generally, the precision can be specified for any `TIME`, `DATETIME`, or `TIMESTAMP` column, in parentheses, after the type name. The datetime precision specifies number of digits after the decimal dot and can be any integer number from 0 to 6. If no precision is specified it is assumed to be 0, for backward compatibility reasons.

A datetime precision can be specified wherever a type name is used. For example:

- when declaring arguments of stored routines;
- when specifying a return type of a stored function;
- when declaring variables;

- in a `CAST` function.

```
CREATE FUNCTION example(x DATETIME(5)) RETURNS TIME(4)
BEGIN
 DECLARE y TIMESTAMP(6);
 RETURN CAST(x AS time(2));
END;
```

`%f` is used as the formatting option for microseconds in the STR_TO_DATE, DATE_FORMAT and FROM_UNIXTIME functions, for example:

```
SELECT STR_TO_DATE('20200809 020917076','%Y%m%d %H%i%s%f');
+-----------------------------------------------------+
| STR_TO_DATE('20200809 020917076','%Y%m%d %H%i%s%f') |
+-----------------------------------------------------+
| 2020-08-09 02:09:17.076000                          |
+-----------------------------------------------------+
```

Additional Information

- When comparing anything to a temporal value (`DATETIME`, `TIME`, DATE, or `TIMESTAMP`), both values are compared as temporal values, not as strings.
- The INFORMATION_SCHEMA.COLUMNS table has a new column `DATETIME_PRECISION`
- NOW(), CURTIME(), UTC_TIMESTAMP(), UTC_TIME(), CURRENT_TIME(), CURRENT_TIMESTAMP(), LOCALTIME() and LOCALTIMESTAMP() accept datetime precision as an optional argument. For example:

```
SELECT CURTIME(4);
--> 10:11:12.3456
```

- TIME_TO_SEC() and UNIX_TIMESTAMP() preserve microseconds of the argument. These functions will return a decimal number if the result non-zero datetime precision and an integer otherwise (for backward compatibility).

```
SELECT TIME_TO_SEC('10:10:10.12345');
--> 36610.12345
```

- Current versions of this patch fix a bug in the following optimization: In certain queries with `DISTINCT` MariaDB can ignore this clause if it can prove that all result rows are unique anyway, for example, when a primary key is compared with a constant. Sometimes this optimization was applied incorrectly, though — for example, when comparing a string with a date constant. This is now fixed.
- `DATE_ADD()` and `DATE_SUB()` functions can now take a `TIME` expression as an argument (not just `DATETIME` as before).

```
SELECT TIME('10:10:10') + INTERVAL 100 MICROSECOND;
--> 10:10:10.000100
```

- The `event_time` field in the mysql.general_log table and the `start_time`, `query_time`, and `lock_time` fields in the mysql.slow_log table now store values with microsecond precision.
- The old syntax `TIMESTAMP(N)`, where `N` is the display width, is no longer supported.
- When a `DATETIME` value is compared to a `TIME` value, the latter is treated as a full datetime with a zero date part, similar to comparing `DATE` to a `DATETIME`, or to comparing `DECIMAL` numbers.
  Earlier versions of MariaDB used to compare only the time part of both operands in such a case.
- In MariaDB, an extra column TIME_MS has been added to the INFORMATION_SCHEMA.PROCESSLIST table, as well as to the output of SHOW FULL PROCESSLIST.

Note: When you convert a temporal value to a value with a smaller precision, it will be truncated, not rounded. This is done to guarantee that the date part is not changed. For example:

```
SELECT CAST('2009-12-31 23:59:59.998877' AS DATETIME(3));
-> 2009-12-31 23:59:59.998
```

See Also

- Data Type Storage Requirements

This page is licensed: CC BY-SA / Gnu FDL

MINUTE

Syntax

```
MINUTE(time)
```

Description

Returns the minute for *time*, in the range 0 to 59.

Examples

```
SELECT MINUTE('2013-08-03 11:04:03');
+------------------------------+
| MINUTE('2013-08-03 11:04:03') |
+------------------------------+
|                            4 |
+------------------------------+

 SELECT MINUTE ('23:12:50');
+---------------------+
| MINUTE ('23:12:50') |
+---------------------+
|                  12 |
+---------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MONTH

Syntax

```
MONTH(date)
```

Description

Returns the month for `date` in the range 1 to 12 for January to December, or 0 for dates such as `0000-00-00` or `2008-00-00` that have a zero month part.

Examples

```
SELECT MONTH('2019-01-03');
+---------------------+
| MONTH('2019-01-03') |
+---------------------+
|                   1 |
+---------------------+

SELECT MONTH('2019-00-03');
+---------------------+
| MONTH('2019-00-03') |
+---------------------+
|                   0 |
+---------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MONTHNAME

Syntax

```
MONTHNAME(date)
```

Description

Returns the full name of the month for date. The language used for the name is controlled by the value of the lc_time_names system variable. See server locale for more on the supported locales.

Examples

```
SELECT MONTHNAME('2019-02-03');
+-------------------------+
| MONTHNAME('2019-02-03') |
+-------------------------+
| February                |
+-------------------------+
```

Changing the locale:

```
SET lc_time_names = 'fr_CA';

SELECT MONTHNAME('2019-05-21');
+-------------------------+
| MONTHNAME('2019-05-21') |
+-------------------------+
| mai                     |
+-------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MONTHS_BETWEEN

`MONTHS_BETWEEN` returns the number of months between dates two dates. If the first date given is later than the second date, the result is positive; otherwise, the result is negative. If both dates are the same days of the month, or both are last days of months, the result is always an integer. Otherwise, the fractional portion of the result based on a 31-day month is calculated, and considered the difference in time components between the dates.

The following example calculates the months between two dates:

```
SELECT MONTHS_BETWEEN
       (TO_DATE('02-02-1995','MM-DD-YYYY'),
        TO_DATE('01-01-1995','MM-DD-YYYY')
       );
```

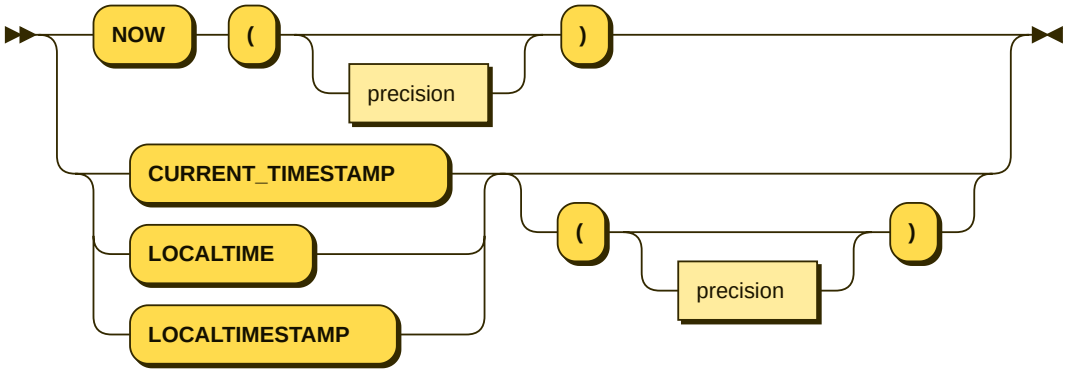
The result is `1.03225806`.

This page is licensed: CC BY-SA / Gnu FDL

NOW

Syntax

```
NOW([precision])
CURRENT_TIMESTAMP
CURRENT_TIMESTAMP([precision])
LOCALTIME
LOCALTIME([precision])
LOCALTIMESTAMP
LOCALTIMESTAMP([precision])
```



Railroad diagram of NOW and synonyms — equivalent to the BNF above

Description

Returns the current date and time as a value in `YYYY-MM-DD HH:MM:SS` or `YYYYMMDDHHMMSS.uuuuuu` format, depending on whether the function is used in a string or numeric context. The value is expressed in the current time zone.

MariaDB starting with 11.7

CURRENT

These functions return SQL standard compliant types:

- `NOW()` and `CURRENT_TIMESTAMP()` return a `TIMESTAMP` value (analogous to the standard type `TIMESTAMP WITH LOCAL TIME ZONE`) which corresponds to the current point in time and is unambiguous around DST changes.
- `LOCALTIMESTAMP` returns a `DATETIME` value (analogous to the standard type `TIMESTAMP WITHOUT TIME ZONE`). Storing its result in a `TIMESTAMP` column can result in a data loss around DST changes.

< 11.7

These functions do **not** return SQL standard compliant types:

- `NOW()`
- `CURRENT_TIMESTAMP()`

- `LOCALTIMESTAMP`

The optional *precision* determines the microsecond precision. See Microseconds in MariaDB.

`NOW()` (or its synonyms) can be used as the default value for TIMESTAMP columns as well as.

When displayed in the INFORMATION_SCHEMA.COLUMNS table, a default CURRENT TIMESTAMP is displayed as `current_timestamp()` .

Changing the timestamp system variable with a SET `timestamp` statement affects the value returned by `NOW()`, but not by SYSDATE().

Examples

```
SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2010-03-27 13:13:25 |
+---------------------+

SELECT NOW() + 0;
+-----------------------+
| NOW() + 0             |
+-----------------------+
| 20100327131329.000000 |
+-----------------------+
```

With precision:

```
SELECT CURRENT_TIMESTAMP(2);
+------------------------+
| CURRENT_TIMESTAMP(2)   |
+------------------------+
| 2018-07-10 09:47:26.24 |
+------------------------+
```

Used as a default TIMESTAMP:

```
CREATE TABLE t (createdTS TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP);
```

```
SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_SCHEMA='test'
  AND COLUMN_NAME LIKE '%ts%'\G
*************************** 1. row ***************************
           TABLE_CATALOG: def
            TABLE_SCHEMA: test
              TABLE_NAME: t
             COLUMN_NAME: ts
        ORDINAL_POSITION: 1
          COLUMN_DEFAULT: current_timestamp()
...
```

See Also

- Microseconds in MariaDB
- timestamp server system variable

This page is licensed: GPLv2, originally from fill_help_tables.sql

PERIOD_ADD

Syntax

```
PERIOD_ADD(P,N)
```

Description

Adds `N` months to period `P`. `P` is in the format YYMM or YYYYMM, and is not a date value. If `P` contains a two-digit year, values from 00 to 69 are converted to from 2000 to 2069, while values from 70 are converted to 1970 upwards.

Returns a value in the format `YYYYMM`.

Examples

```
SELECT PERIOD_ADD(200801,2);
+----------------------+
| PERIOD_ADD(200801,2) |
+----------------------+
|               200803 |
+----------------------+

SELECT PERIOD_ADD(6910,2);
+--------------------+
| PERIOD_ADD(6910,2) |
+--------------------+
|             206912 |
+--------------------+

SELECT PERIOD_ADD(7010,2);
+--------------------+
| PERIOD_ADD(7010,2) |
+--------------------+
|             197012 |
+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

PERIOD_DIFF

Syntax

```
PERIOD_DIFF(P1,P2)
```

Description

Returns the number of months between periods P1 and P2. P1 and P2 can be in the format `YYMM` or `YYYYMM`, and are not date values.

If P1 or P2 contains a two-digit year, values from 00 to 69 are converted to from 2000 to 2069, while values from 70 are converted to 1970 upwards.

Examples

```
SELECT PERIOD_DIFF(200802,200703);
+----------------------------+
| PERIOD_DIFF(200802,200703) |
+----------------------------+
|                         11 |
+----------------------------+

SELECT PERIOD_DIFF(6902,6803);
+------------------------+
| PERIOD_DIFF(6902,6803) |
+------------------------+
|                     11 |
+------------------------+

SELECT PERIOD_DIFF(7002,6803);
+------------------------+
| PERIOD_DIFF(7002,6803) |
+------------------------+
|                  -1177 |
+------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

QUARTER

Syntax

```
QUARTER(date)
```

Description

Returns the quarter of the year for `date`, in the range 1 to 4. Returns 0 if month contains a zero value, or `NULL` if the given value is not otherwise a valid date (zero values are accepted).

Examples

```
SELECT QUARTER('2008-04-01');
+-----------------------+
| QUARTER('2008-04-01') |
+-----------------------+
|                     2 |
+-----------------------+

SELECT QUARTER('2019-00-01');
+-----------------------+
| QUARTER('2019-00-01') |
+-----------------------+
|                     0 |
+-----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SEC_TO_TIME

Syntax

```
SEC_TO_TIME(seconds)
```

Description

Returns the seconds argument, converted to hours, minutes, and seconds, as a TIME value. The range of the result is constrained to that of the TIME data type. A warning occurs if the argument corresponds to a value outside that range.

The time will be returned in the format `hh:mm:ss`, or `hhmmss` if used in a numeric calculation.

Examples

```
SELECT SEC_TO_TIME(12414);
+-------------------+
| SEC_TO_TIME(12414) |
+-------------------+
| 03:26:54          |
+-------------------+

SELECT SEC_TO_TIME(12414)+0;
+----------------------+
| SEC_TO_TIME(12414)+0 |
+----------------------+
|                32654 |
+----------------------+

SELECT SEC_TO_TIME(9999999);
+----------------------+
| SEC_TO_TIME(9999999) |
+----------------------+
| 838:59:59            |
+----------------------+
1 row in set, 1 warning (0.00 sec)

SHOW WARNINGS;
+---------+------+-------------------------------------------+
| Level   | Code | Message                                   |
+---------+------+-------------------------------------------+
| Warning | 1292 | Truncated incorrect time value: '9999999' |
+---------+------+-------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SECOND

Syntax

```
SECOND(time)
```

Description

Returns the second for a given `time` (which can include microseconds), in the range 0 to 59, or `NULL` if not given a valid time value.

Examples

```
SELECT SECOND('10:05:03');
+-------------------+
| SECOND('10:05:03') |
+-------------------+
|                  3 |
+-------------------+

SELECT SECOND('10:05:01.999999');
+--------------------------+
| SECOND('10:05:01.999999') |
+--------------------------+
|                        1 |
+--------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

STR_TO_DATE

Syntax

```
STR_TO_DATE(str,format)
```

Description

This is the inverse of the DATE_FORMAT() function. It takes a string `str` and a format string `format`. `STR_TO_DATE()` returns a `DATETIME` value if the format string contains both date and time parts, or a `DATE` or `TIME` value if the string contains only date or time parts.

The date, time, or datetime values contained in `str` should be given in the format indicated by format. If str contains an illegal date, time, or datetime value, `STR_TO_DATE()` returns `NULL`. An illegal value also produces a warning.

Under specific SQL_MODE settings an error may also be generated if the `str` isn't a valid date:

- ALLOW_INVALID_DATES
- NO_ZERO_DATE
- NO_ZERO_IN_DATE

The options that can be used by STR_TO_DATE(), as well as its inverse DATE_FORMAT() and the FROM_UNIXTIME() function, are:

Option	Description
%a	Short weekday name in current locale (Variable lc_time_names).
%b	Short form month name in current locale. For locale en_US this is one of: Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov or Dec.
%c	Month with 1 or 2 digits.
%D	Day with English suffix 'th', 'nd', 'st' or 'rd''. (1st, 2nd, 3rd…).
%d	Day with 2 digits.
%e	Day with 1 or 2 digits.
%f	Microseconds 6 digits.
%H	Hour with 2 digits between 00-23.
%h	Hour with 2 digits between 01-12.
%I	Hour with 2 digits between 01-12.
%i	Minute with 2 digits.

Option	Description
%j	Day of the year (001-366)
%k	Hour with 1 digits between 0-23.
%l	Hour with 1 digits between 1-12.
%M	Full month name in current locale (Variable lc_time_names).
%m	Month with 2 digits.
%p	AM/PM according to current locale (Variable lc_time_names).
%r	Time in 12 hour format, followed by AM/PM. Short for '%I:%i:%S %p'.
%S	Seconds with 2 digits.
%s	Seconds with 2 digits.
%T	Time in 24 hour format. Short for '%H:%i:%S'.
%U	Week number (00-53), when first day of the week is Sunday.
%u	Week number (00-53), when first day of the week is Monday.
%V	Week number (01-53), when first day of the week is Sunday. Used with %X.
%v	Week number (01-53), when first day of the week is Monday. Used with %x.
%W	Full weekday name in current locale (Variable lc_time_names).
%w	Day of the week. 0 = Sunday, 6 = Saturday.
%X	Year with 4 digits when first day of the week is Sunday. Used with %V.
%x	Year with 4 digits when first day of the week is Monday. Used with %v.
%Y	Year with 4 digits.
%y	Year with 2 digits.
%#	For str_to_date(), skip all numbers.
%.	For str_to_date(), skip all punctuation characters.
%@	For str_to_date(), skip all alpha characters.
%%	A literal % character.

Examples

```
SELECT STR_TO_DATE('Wednesday, June 2, 2014', '%W, %M %e, %Y');
+-------------------------------------------------------+
| STR_TO_DATE('Wednesday, June 2, 2014', '%W, %M %e, %Y') |
+-------------------------------------------------------+
| 2014-06-02                                            |
+-------------------------------------------------------+

SELECT STR_TO_DATE('Wednesday23423, June 2, 2014', '%W, %M %e, %Y');
+-------------------------------------------------------------+
| STR_TO_DATE('Wednesday23423, June 2, 2014', '%W, %M %e, %Y') |
+-------------------------------------------------------------+
| NULL                                                        |
+-------------------------------------------------------------+
1 row in set, 1 warning (0.00 sec)

SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------------------------------+
| Level   | Code | Message                                                                                 |
+---------+------+-----------------------------------------------------------------------------------------+
| Warning | 1411 | Incorrect datetime value: 'Wednesday23423, June 2, 2014' for function str_to_date |
+---------+------+-----------------------------------------------------------------------------------------+

SELECT STR_TO_DATE('Wednesday23423, June 2, 2014', '%W%#, %M %e, %Y');
+-----------------------------------------------------------------+
| STR_TO_DATE('Wednesday23423, June 2, 2014', '%W%#, %M %e, %Y') |
+-----------------------------------------------------------------+
| 2014-06-02                                                      |
+-----------------------------------------------------------------+
```

See Also

- DATE_FORMAT()
- FROM_UNIXTIME()

This page is licensed: GPLv2, originally from fill_help_tables.sql

SUBDATE

Syntax

```
SUBDATE(date,INTERVAL expr unit), SUBDATE(expr,days)
```

Description

When invoked with the `INTERVAL` form of the second argument, `SUBDATE()` is a synonym for DATE_SUB(). See Date and Time Units for a complete list of permitted units.

The second form allows the use of an integer value for days. In such cases, it is interpreted as the number of days to be subtracted from the date or datetime expression expr.

Examples

```
SELECT DATE_SUB('2008-01-02', INTERVAL 31 DAY);
+----------------------------------------+
| DATE_SUB('2008-01-02', INTERVAL 31 DAY) |
+----------------------------------------+
| 2007-12-02                             |
+----------------------------------------+

SELECT SUBDATE('2008-01-02', INTERVAL 31 DAY);
+----------------------------------------+
| SUBDATE('2008-01-02', INTERVAL 31 DAY) |
+----------------------------------------+
| 2007-12-02                             |
+----------------------------------------+
```

```
SELECT SUBDATE('2008-01-02 12:00:00', 31);
+-----------------------------------+
| SUBDATE('2008-01-02 12:00:00', 31) |
+-----------------------------------+
| 2007-12-02 12:00:00                |
+-----------------------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d, SUBDATE(d, 10) from t1;
+---------------------+---------------------+
| d                   | SUBDATE(d, 10)      |
+---------------------+---------------------+
| 2007-01-30 21:31:07 | 2007-01-20 21:31:07 |
| 1983-10-15 06:42:51 | 1983-10-05 06:42:51 |
| 2011-04-21 12:34:56 | 2011-04-11 12:34:56 |
| 2011-10-30 06:31:41 | 2011-10-20 06:31:41 |
| 2011-01-30 14:03:25 | 2011-01-20 14:03:25 |
| 2004-10-07 11:19:34 | 2004-09-27 11:19:34 |
+---------------------+---------------------+

SELECT d, SUBDATE(d, INTERVAL 10 MINUTE) from t1;
+---------------------+-------------------------------+
| d                   | SUBDATE(d, INTERVAL 10 MINUTE) |
+---------------------+-------------------------------+
| 2007-01-30 21:31:07 | 2007-01-30 21:21:07           |
| 1983-10-15 06:42:51 | 1983-10-15 06:32:51           |
| 2011-04-21 12:34:56 | 2011-04-21 12:24:56           |
| 2011-10-30 06:31:41 | 2011-10-30 06:21:41           |
| 2011-01-30 14:03:25 | 2011-01-30 13:53:25           |
| 2004-10-07 11:19:34 | 2004-10-07 11:09:34           |
+---------------------+-------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SUBTIME

Syntax

```
SUBTIME(expr1,expr2)
```

Description

`SUBTIME()` returns `expr1` - `expr2` expressed as a value in the same format as `expr1`. `expr1` is a time or datetime expression, and expr2 is a time expression.

Examples

```
SELECT SUBTIME('2007-12-31 23:59:59.999999','1 1:1:1.000002');
+----------------------------------------------------------+
| SUBTIME('2007-12-31 23:59:59.999999','1 1:1:1.000002') |
+----------------------------------------------------------+
| 2007-12-30 22:58:58.999997                               |
+----------------------------------------------------------+

SELECT SUBTIME('01:00:00.999999', '02:00:00.999998');
+-----------------------------------------------+
| SUBTIME('01:00:00.999999', '02:00:00.999998') |
+-----------------------------------------------+
| -00:59:59.999999                              |
+-----------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SYSDATE

Syntax

```
SYSDATE([precision])
```

Description

Returns the current date and time as a value in `YYYY-MM-DD HH:MM:SS` or `YYYYMMDDHHMMSS.uuuuuu` format, depending on whether the function is used in a string or numeric context.

The optional *precision* determines the microsecond precision. See Microseconds in MariaDB.

`SYSDATE()` returns the time at which it executes. This differs from the behavior for NOW(), which returns a constant time that indicates the time at which the statement began to execute. (Within a stored routine or trigger, NOW() returns the time at which the routine or triggering statement began to execute.)

In addition, changing the timestamp system variable with a SET `timestamp` statement affects the value returned by `NOW()` but not by `SYSDATE()`. This means that timestamp settings in the binary log have no effect on invocations of `SYSDATE()`.

Because `SYSDATE()` can return different values even within the same statement, and is not affected by `SET TIMESTAMP`, it is non-deterministic and therefore unsafe for replication if statement-based binary logging is used. If that is a problem, you can use row-based logging, or start the server with the –sysdate-is-now mariadbd option to cause `SYSDATE()` to be an alias for `NOW()`. The non-deterministic nature of `SYSDATE()` also means that indexes cannot be used for evaluating expressions that refer to it, and that statements using the `SYSDATE()` function are unsafe for statement-based replication.

Examples

Difference between `NOW()` and `SYSDATE()`:

```
SELECT NOW(), SLEEP(2), NOW();
+---------------------+----------+---------------------+
| NOW()               | SLEEP(2) | NOW()               |
+---------------------+----------+---------------------+
| 2010-03-27 13:23:40 |        0 | 2010-03-27 13:23:40 |
+---------------------+----------+---------------------+

SELECT SYSDATE(), SLEEP(2), SYSDATE();
+---------------------+----------+---------------------+
| SYSDATE()           | SLEEP(2) | SYSDATE()           |
+---------------------+----------+---------------------+
| 2010-03-27 13:23:52 |        0 | 2010-03-27 13:23:54 |
+---------------------+----------+---------------------+
```

With precision:

```
SELECT SYSDATE(4);
+--------------------------+
| SYSDATE(4)               |
+--------------------------+
| 2018-07-10 10:17:13.1689 |
+--------------------------+
```

See Also

- Microseconds in MariaDB
- timestamp server system variable

This page is licensed: GPLv2, originally from fill_help_tables.sql

TIME

Syntax

```
TIME(expr)
```

Description

Extracts the time part of the time or datetime expression `expr` and returns it as a string.

Examples

```
SELECT TIME('2003-12-31 01:02:03');
+-----------------------------+
| TIME('2003-12-31 01:02:03') |
+-----------------------------+
| 01:02:03                    |
+-----------------------------+

SELECT TIME('2003-12-31 01:02:03.000123');
+----------------------------------+
| TIME('2003-12-31 01:02:03.000123') |
+----------------------------------+
| 01:02:03.000123                  |
+----------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TIME_FORMAT

Syntax

```
TIME_FORMAT(time,format)
```

Description

This is used like the DATE_FORMAT() function, but the format string may contain format specifiers only for hours, minutes, and seconds. Other specifiers produce a NULL value or 0.

Examples

```
SELECT TIME_FORMAT('100:00:00', '%H %k %h %I %l');
+----------------------------------------+
| TIME_FORMAT('100:00:00', '%H %k %h %I %l') |
+----------------------------------------+
| 100 100 04 04 4                        |
+----------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TIME_TO_SEC

Syntax

```
TIME_TO_SEC(time)
```

Description

Returns the time argument, converted to seconds.

The value returned by `TIME_TO_SEC` is of type DOUBLE. The returned value preserves microseconds of the argument. See also Microseconds in MariaDB.

Examples

```
SELECT TIME_TO_SEC('22:23:00');
+-------------------------+
| TIME_TO_SEC('22:23:00') |
+-------------------------+
|                   80580 |
+-------------------------+
```

```
SELECT TIME_TO_SEC('00:39:38');
+-------------------------+
| TIME_TO_SEC('00:39:38') |
+-------------------------+
|                    2378 |
+-------------------------+
```

```
SELECT TIME_TO_SEC('09:12:55.2355');
+-----------------------------+
| TIME_TO_SEC('09:12:55.2355') |
+-----------------------------+
|                   33175.2355 |
+-----------------------------+
1 row in set (0.000 sec)
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TIMEDIFF

Syntax

```
TIMEDIFF(expr1,expr2)
```

Description

TIMEDIFF() returns `expr1` - `expr2` expressed as a time value. `expr1` and `expr2` are time or date-and-time expressions, but both must be of the same type.

Examples

```
SELECT TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001');
+----------------------------------------------------------------+
| TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001') |
+----------------------------------------------------------------+
| -00:00:00.000001                                               |
+----------------------------------------------------------------+

SELECT TIMEDIFF('2008-12-31 23:59:59.000001', '2008-12-30 01:01:01.000002');
+-------------------------------------------------------------------+
| TIMEDIFF('2008-12-31 23:59:59.000001', '2008-12-30 01:01:01.000002') |
+-------------------------------------------------------------------+
| 46:58:57.999999                                                   |
+-------------------------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TIMESTAMP

> **Note:**
>
> For the timestamp data type, see TIMESTAMP.

Syntax

```
TIMESTAMP(expr), TIMESTAMP(expr1,expr2)
```

Description

With a single argument, this function returns the date or datetime expression `expr` as a datetime value. With two arguments, it adds the time expression `expr2` to the date or datetime expression `expr1` and returns the result as a datetime value.

Examples

```
SELECT TIMESTAMP('2003-12-31');
+-------------------------+
| TIMESTAMP('2003-12-31') |
+-------------------------+
| 2003-12-31 00:00:00     |
+-------------------------+

SELECT TIMESTAMP('2003-12-31 12:00:00','6:30:00');
+-----------------------------------------+
| TIMESTAMP('2003-12-31 12:00:00','6:30:00') |
+-----------------------------------------+
| 2003-12-31 18:30:00                     |
+-----------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TIMESTAMPADD

Syntax

```
TIMESTAMPADD(unit,interval,datetime_expr)
```

Description

Adds the integer expression interval to the date or datetime expression datetime_expr. The unit for interval is given by the unit argument, which should be one of the following values: `MICROSECOND`, `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `QUARTER`, or `YEAR`.

The unit value may be specified using one of keywords as shown, or with a prefix of `SQL_TSI_`. For example, `DAY` and `SQL_TSI_DAY` both are allowed.

Examples

```
SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');
+-------------------------------------+
| TIMESTAMPADD(MINUTE,1,'2003-01-02') |
+-------------------------------------+
| 2003-01-02 00:01:00                 |
+-------------------------------------+

SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');
+----------------------------------+
| TIMESTAMPADD(WEEK,1,'2003-01-02') |
+----------------------------------+
| 2003-01-09                       |
+----------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TIMESTAMPDIFF

Syntax

```
TIMESTAMPDIFF(unit,datetime_expr1,datetime_expr2)
```

Description

Returns `datetime_expr2 - datetime_expr1`, where `datetime_expr1` and `datetime_expr2` are date or datetime expressions. One expression may be a date and the other a datetime; a date value is treated as a datetime having the time part '00:00:00' where necessary. The unit for the result (an integer) is given by the unit argument. The legal values for unit are the same as those listed in the description of the TIMESTAMPADD() function, that is, `MICROSECOND`, `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `QUARTER`, or `YEAR`.

`TIMESTAMPDIFF` can also be used to calculate age.

Examples

```
SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');
+-------------------------------------------------+
| TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01') |
+-------------------------------------------------+
|                                               3 |
+-------------------------------------------------+

SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');
+------------------------------------------------+
| TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01') |
+------------------------------------------------+
|                                             -1 |
+------------------------------------------------+

SELECT TIMESTAMPDIFF(MINUTE,'2003-02-01','2003-05-01 12:05:55');
+--------------------------------------------------------+
| TIMESTAMPDIFF(MINUTE,'2003-02-01','2003-05-01 12:05:55') |
+--------------------------------------------------------+
|                                                 128885 |
+--------------------------------------------------------+
```

Calculating age:

```
SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2019-05-27 |
+------------+

SELECT TIMESTAMPDIFF(YEAR, '1971-06-06', CURDATE()) AS age;
+------+
| age  |
+------+
|   47 |
+------+

SELECT TIMESTAMPDIFF(YEAR, '1971-05-06', CURDATE()) AS age;
+------+
| age  |
+------+
|   48 |
+------+
```

Age as of 2014-08-02:

```
SELECT name, date_of_birth, TIMESTAMPDIFF(YEAR,date_of_birth,'2014-08-02') AS age
  FROM student_details;
+---------+---------------+------+
| name    | date_of_birth | age  |
+---------+---------------+------+
| Chun    | 1993-12-31    |   20 |
| Esben   | 1946-01-01    |   68 |
| Kaolin  | 1996-07-16    |   18 |
| Tatiana | 1988-04-13    |   26 |
+---------+---------------+------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TO_DATE

> **Note:**
>
> This function is available from MariaDB 12.3.

Syntax

```
TO_DATE(string_expression [DEFAULT string_expression ON CONVERSION ERROR],
        format_string [,NLS_FORMAT_STRING])
```

Description

`TO_DATE` was added for Oracle support.

- `format_string` has the same format elements as TO_CHAR(), except a few elements that are not supported (see below). `TO_DATE()` returns a datetime value.
- The allowed separators are the same as for `TO_CHAR()`: space ( ), tab (`\t`), and any of `!`, `#`, `%`, `'`, `(`, `)`, `*`, `+`, `-`, `.`, `/`, `:`, `;`, `<`, `=`, or `>`.
- `&` can be used if the next character is not a character in the `a-z` or `A-Z` range. `text` indicates a text string that is used verbatim in the format. You cannot use the double-quote (`"`) as a separator.
- `NLS_FORMAT_STRING` supports these options: `NLS_CALENDAR=GREGORIAN` and `NLS_DATE_LANGUAGE=` *`language`*, where *`language`* can be one of the following:
  - All MariaDB short locales, like `en_AU`.
  - The following Oracle language names: ALBANIAN, AMERICAN, ARABIC, BASQUE, BELARUSIAN, BRAZILIAN PORTUGUESE, BULGARIAN, CANADIAN FRENCH, CATALAN, CROATIAN, CYRILLIC SERBIAN, CZECH, DANISH, DUTCH, ENGLISH, ESTONIAN, FINNISH, FRENCH, GERMAN, GREEK, HEBREW, HINDI, HUNGARIAN, ICELANDIC, INDONESIAN, ITALIAN, JAPANESE, KANNADA, KOREAN, LATIN AMERICAN SPANISH, LATVIAN, LITHUANIAN, MACEDONIAN, MALAY, MEXICAN SPANISH, NORWEGIAN, POLISH, PORTUGUESE, ROMANIAN, RUSSIAN, SIMPLIFIED CHINESE, SLOVAK, SLOVENIAN, SPANISH, SWAHILI, SWEDISH, TAMIL, THAI, TRADITIONAL CHINESE, TURKISH, UKRAINIAN, VIETNAMESE

Supported Format Elements

- `AD` – Anno Domini (“in the year of the Lord”)
- `AD_DOT` – Anno Domini (“in the year of the Lord”)
- `AM` – Meridian indicator (before midday)
- `AM_DOT` – Meridian indicator (before midday)
- `DAY` – Name of day
- `DD` – Day (1-31)

- `DDD` – Day of year (1-336)
- `DY` – Abbreviated name of day
- `FF[1-6]` – Fractional seconds
- `HH` – Hour (1-12)
- `HH12` – Hour (1-12)
- `HH24` – Hour (0-23)
- `MI` – Minutes (0-59)
- `MM` – Month (1-12)
- `MON` – Abbreviated name of month
- `MONTH` – Name of Month
- `PM` – Meridian indicator (after midday)
- `PM_DOT` – Meridian indicator (after midday)
- `RR` – 20th century dates in the 21st century. 2 digits in the `50-99` range are assumed starting from the year 2000, and `0-49` is assumed from 1900.
- `RRRR` – 20th century dates in the 21st century. 4 digits
- `SS` – Seconds
- `SYYYY` – Signed 4-digit year; MariaDB only supports positive years
- `Y` – 1-digit year
- `YY` – 2-digit year
- `YYY` – 3-digit year
- `YYYY` – 4 digit year

> **Note:**
>
> If no datetime is given, the current datetime is used. For example, if `'MM-DD HH-MM-SS'` is given, the current year is used. (This is Oracle behavior.)

Unsupported Options

- `BC`, `D`, `DL`, `DS`, `E`, `EE`, `FM`, `FX`, `RM`, `SSSSS`, `TS`, `TZD`, `TZH`, `TZR`, `X`, and `SY BC` are not supported by MariaDB datetime.
- Most other formats do not make sense in a MariaDB context, as we return `datetime` with fractions, not as a string.
- `D` (day-of-week) is not supported as it is not clear how it would map to MariaDB. This element depends on the NLS territory of the session.
- `RR` only works with 2-digit years. (In Oracle, `RR` may also work with 4-digit years in some context, but the rules are not clear.)

Extensions and Differences Compared to Oracle

- MariaDB supports `FF` (fractional seconds). If `FF[#]` is used, `TO_DATE` returns a `datetime` with `#` of subseconds. If `FF` is not used, a `datetime` is returned. A warning (but no error) is issued if the string contains more digits than specified with `F(#)`.
- Names can be shortened to their unique prefixes. For example, both `January` and `Ja` work fine.
- No error is issued if the datetime string is shorter than `format_string` and the next unused character is not a number. This is useful to get a `datetime` from a mixed set of strings in `datetime` format. By contrast, Oracle gives an error if the `datetime` string is too short.
- MariaDB supports short locales as language names.
- `NLS_DATE_FORMAT` can use double-quote (`"`) and single-quote (`'`) for quoting.

- `NLS_DATE_FORMAT` must be a constant string. This is to ensure that the server knows which locale to use when executing the function.

Examples

Convert a textual date and time to `DATE` format:

```
SELECT TO_DATE('February 5, 2026, 20:56','%Y-%m-%e')
+-----------------------------------------------+
| TO_DATE('February 5, 2026, 20:56','%Y-%m-%e') |
+-----------------------------------------------+
| 2026-02-05                                    |
+-----------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

TO_DAYS

Syntax

```
TO_DAYS(date)
```

Description

Given a date `date`, returns the number of days since the start of the current calendar (`0000-00-00`).

The function is not designed for use with dates before the advent of the Gregorian calendar in October 1582. Results are not reliable since it doesn't account for the lost days when the calendar changed from the Julian calendar.

This is the converse of the FROM_DAYS() function.

Examples

```
SELECT TO_DAYS('2007-10-07');
+-----------------------+
| TO_DAYS('2007-10-07') |
+-----------------------+
|                733321 |
+-----------------------+

SELECT TO_DAYS('0000-01-01');
+-----------------------+
| TO_DAYS('0000-01-01') |
+-----------------------+
|                     1 |
+-----------------------+

SELECT TO_DAYS(950501);
+-----------------+
| TO_DAYS(950501) |
+-----------------+
|          728779 |
+-----------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TO_SECONDS

Syntax

```
TO_SECONDS(expr)
```

Description

Returns the number of seconds from year 0 till `expr`, or NULL if `expr` is not a valid date or datetime.

Examples

```
SELECT TO_SECONDS('2013-06-13');
+--------------------------+
| TO_SECONDS('2013-06-13') |
+--------------------------+
|              63538300800 |
+--------------------------+

SELECT TO_SECONDS('2013-06-13 21:45:13');
+-----------------------------------+
| TO_SECONDS('2013-06-13 21:45:13') |
+-----------------------------------+
|                       63538379113 |
+-----------------------------------+

SELECT TO_SECONDS(NOW());
+-------------------+
| TO_SECONDS(NOW()) |
+-------------------+
|       63543530875 |
+-------------------+

SELECT TO_SECONDS(20130513);
+----------------------+
| TO_SECONDS(20130513) |
+----------------------+
|          63535622400 |
+----------------------+
1 row in set (0.00 sec)

SELECT TO_SECONDS(130513);
+--------------------+
| TO_SECONDS(130513) |
+--------------------+
|        63535622400 |
+--------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

TRUNC

> **Note:**
>
> Introduced in MariaDB 12.2.

Syntax

```
TRUNC(date[,fmt])
```

Description

Returns a DATETIME truncated according to `fmt`.

Supported formats:

Truncate to day: `DD`, `DDD`, `J`
Truncate to month: `MM`, `MON`, `MONTH`, `RM`
Truncate to year: `SYEAR`, `SYYYY`, `Y`, `YEAR`, `YY`, `YYY`, `YYYY`

Examples

```
SELECT TRUNC('2025-09-24 12:43','DD');
+--------------------------------+
| TRUNC('2025-09-24 12:43','DD') |
+--------------------------------+
| 2025-09-24 00:00:00            |
+--------------------------------+

SELECT TRUNC('2025-09-24 12:43','MM');
+--------------------------------+
| TRUNC('2025-09-24 12:43','MM') |
+--------------------------------+
| 2025-09-01 00:00:00            |
+--------------------------------+

SELECT TRUNC('2025-09-24 12:43','YY');
+--------------------------------+
| TRUNC('2025-09-24 12:43','YY') |
+--------------------------------+
| 2025-01-01 00:00:00            |
+--------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

UNIX_TIMESTAMP

Syntax

```
UNIX_TIMESTAMP()
UNIX_TIMESTAMP(date)
```

Description

If called with no argument, returns a Unix timestamp (seconds since `1970-01-01 00:00:00` UTC) as an unsigned integer. If `UNIX_TIMESTAMP()` is called with a date argument, it returns the value of the argument as seconds since `1970-01-01 00:00:00` UTC. date may be a DATE string, a DATETIME string, a TIMESTAMP, or a number in the format `YYMMDD` or `YYYYMMDD`. The server interprets date as a value in the current time zone and converts it to an internal value in UTC. Clients can set their time zone as described in time zones.

The inverse function of `UNIX_TIMESTAMP()` is FROM_UNIXTIME()

`UNIX_TIMESTAMP()` supports microseconds.

> **CURRENT**
>
> Timestamps in MariaDB have a maximum value of `4294967295`, equivalent to `2106-02-07 06:28:15`. This is due to the underlying 32-bit limitation. Using the function on a timestamp beyond this will result in `NULL` being returned. Use DATETIME as a storage type if you require dates beyond this.
>
> **< 11.5**

> Timestamps in MariaDB have a maximum value of `2147483647`, equivalent to `2038-01-19 05:14:07`. This is due to the underlying 32-bit limitation. Using the function on a timestamp beyond this will result in `NULL` being returned. Use DATETIME as a storage type if you require dates beyond this.

Error Handling

Returns `NULL` for wrong arguments to `UNIX_TIMESTAMP()`.

Examples

```
SELECT UNIX_TIMESTAMP();
+------------------+
| UNIX_TIMESTAMP() |
+------------------+
|       1269711082 |
+------------------+

SELECT UNIX_TIMESTAMP('2007-11-30 10:30:19');
+---------------------------------------+
| UNIX_TIMESTAMP('2007-11-30 10:30:19') |
+---------------------------------------+
|                     1196436619.000000 |
+---------------------------------------+

SELECT UNIX_TIMESTAMP("2007-11-30 10:30:19.123456");
+-----------------------------------------------+
| unix_timestamp("2007-11-30 10:30:19.123456") |
+-----------------------------------------------+
|                            1196411419.123456 |
+-----------------------------------------------+

SELECT FROM_UNIXTIME(UNIX_TIMESTAMP('2007-11-30 10:30:19'));
+-------------------------------------------------------+
| FROM_UNIXTIME(UNIX_TIMESTAMP('2007-11-30 10:30:19')) |
+-------------------------------------------------------+
| 2007-11-30 10:30:19.000000                            |
+-------------------------------------------------------+

SELECT FROM_UNIXTIME(FLOOR(UNIX_TIMESTAMP('2007-11-30 10:30:19')));
+-------------------------------------------------------------+
| FROM_UNIXTIME(FLOOR(UNIX_TIMESTAMP('2007-11-30 10:30:19'))) |
+-------------------------------------------------------------+
| 2007-11-30 10:30:19                                         |
+-------------------------------------------------------------+
```

See Also

- FROM_UNIXTIME()

This page is licensed: GPLv2, originally from fill_help_tables.sql

UTC_DATE

Syntax

```
UTC_DATE, UTC_DATE()
```

Description

Returns the current UTC date as a value in `YYYY-MM-DD` or `YYYYMMDD` format, depending on whether the function is used in a string or numeric context.

Examples

```
SELECT UTC_DATE(), UTC_DATE() + 0;
+------------+----------------+
| UTC_DATE() | UTC_DATE() + 0 |
+------------+----------------+
| 2010-03-27 |       20100327 |
+------------+----------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

UTC_TIME

Syntax

```
UTC_TIME
UTC_TIME([precision])
```

Description

Returns the current UTC time as a value in `HH:MM:SS` or `HHMMSS.uuuuuu` format, depending on whether the function is used in a string or numeric context.

The optional *precision* determines the microsecond precision. See Microseconds in MariaDB.

Examples

```
SELECT UTC_TIME(), UTC_TIME() + 0;
+------------+----------------+
| UTC_TIME() | UTC_TIME() + 0 |
+------------+----------------+
| 17:32:34   |  173234.000000 |
+------------+----------------+
```

With precision:

```
SELECT UTC_TIME(5);
+----------------+
| UTC_TIME(5)    |
+----------------+
| 07:52:50.78369 |
+----------------+
```

See Also

- Microseconds in MariaDB

This page is licensed: GPLv2, originally from fill_help_tables.sql

UTC_TIMESTAMP

Syntax

```
UTC_TIMESTAMP
UTC_TIMESTAMP([precision])
```

Description

Returns the current UTC date and time as a value in `YYYY-MM-DD HH:MM:SS` or `YYYYMMDDHHMMSS.uuuuuu` format, depending on whether the function is used in a string or numeric context.

The optional *precision* determines the microsecond precision. See Microseconds in MariaDB.

Examples

```
SELECT UTC_TIMESTAMP(), UTC_TIMESTAMP() + 0;
+---------------------+----------------------+
| UTC_TIMESTAMP()     | UTC_TIMESTAMP() + 0   |
+---------------------+----------------------+
| 2010-03-27 17:33:16 | 20100327173316.000000 |
+---------------------+----------------------+
```

With precision:

```
SELECT UTC_TIMESTAMP(4);
+--------------------------+
| UTC_TIMESTAMP(4)         |
+--------------------------+
| 2018-07-10 07:51:09.1019 |
+--------------------------+
```

See Also

- Time Zones
- Microseconds in MariaDB

This page is licensed: GPLv2, originally from fill_help_tables.sql

WEEK

Syntax

```
WEEK(date[,mode])
```

Description

This function returns the week number for `date`. The two-argument form of `WEEK()` allows you to specify whether the week starts on Sunday or Monday and whether the return value should be in the range from 0 to 53 or from 1 to 53. If the `mode` argument is omitted, the value of the default_week_format system variable is used.

Modes

Mode	1st day of week	Range	Week 1 is the 1st week with
0	Sunday	0-53	a Sunday in this year
1	Monday	0-53	more than 3 days this year
2	Sunday	1-53	a Sunday in this year
3	Monday	1-53	more than 3 days this year
4	Sunday	0-53	more than 3 days this year
5	Monday	0-53	a Monday in this year
6	Sunday	1-53	more than 3 days this year
7	Monday	1-53	a Monday in this year

With the mode value of 3, which means ‘more than 3 days this year’, weeks are numbered according to ISO 8601:1988.

Examples

```
SELECT WEEK('2008-02-20');
+--------------------+
| WEEK('2008-02-20') |
+--------------------+
|                  7 |
+--------------------+

SELECT WEEK('2008-02-20',0);
+----------------------+
| WEEK('2008-02-20',0) |
+----------------------+
|                    7 |
+----------------------+

SELECT WEEK('2008-02-20',1);
+----------------------+
| WEEK('2008-02-20',1) |
+----------------------+
|                    8 |
+----------------------+

SELECT WEEK('2008-12-31',0);
+----------------------+
| WEEK('2008-12-31',0) |
+----------------------+
|                   52 |
+----------------------+

SELECT WEEK('2008-12-31',1);
+----------------------+
| WEEK('2008-12-31',1) |
+----------------------+
|                   53 |
+----------------------+

 SELECT WEEK('2019-12-30',3);
+----------------------+
| WEEK('2019-12-30',3) |
+----------------------+
|                    1 |
+----------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d, WEEK(d,0), WEEK(d,1) from t1;
+---------------------+-----------+-----------+
| d                   | WEEK(d,0) | WEEK(d,1) |
+---------------------+-----------+-----------+
| 2007-01-30 21:31:07 |         4 |         5 |
| 1983-10-15 06:42:51 |        41 |        41 |
| 2011-04-21 12:34:56 |        16 |        16 |
| 2011-10-30 06:31:41 |        44 |        43 |
| 2011-01-30 14:03:25 |         5 |         4 |
| 2004-10-07 11:19:34 |        40 |        41 |
+---------------------+-----------+-----------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

WEEKDAY

Syntax

```
WEEKDAY(date)
```

Description

Returns the weekday index for `date` (`0` = Monday, `1` = Tuesday, … `6` = Sunday).

This contrasts with DAYOFWEEK() which follows the ODBC standard (`1` = Sunday, `2` = Monday, …, `7` = Saturday).

Examples

```
SELECT WEEKDAY('2008-02-03 22:23:00');
+--------------------------------+
| WEEKDAY('2008-02-03 22:23:00') |
+--------------------------------+
|                              6 |
+--------------------------------+

SELECT WEEKDAY('2007-11-06');
+-----------------------+
| WEEKDAY('2007-11-06') |
+-----------------------+
|                     1 |
+-----------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d FROM t1 where WEEKDAY(d) = 6;
+---------------------+
| d                   |
+---------------------+
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
+---------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

WEEKOFYEAR

Syntax

```
WEEKOFYEAR(date)
```

Description

Returns the calendar week of the date as a number in the range from 1 sqto 53. `WEEKOFYEAR()` is a compatibility function that is equivalent to WEEK(date,3).

Examples

```
SELECT WEEKOFYEAR('2008-02-20');
+--------------------------+
| WEEKOFYEAR('2008-02-20') |
+--------------------------+
|                        8 |
+--------------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT * FROM t1;
+---------------------+
| d                   |
+---------------------+
| 2007-01-30 21:31:07 |
| 1983-10-15 06:42:51 |
| 2011-04-21 12:34:56 |
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
| 2004-10-07 11:19:34 |
+---------------------+
```

```
SELECT d, WEEKOFYEAR(d), WEEK(d,3) from t1;
+---------------------+---------------+-----------+
| d                   | WEEKOFYEAR(d) | WEEK(d,3) |
+---------------------+---------------+-----------+
| 2007-01-30 21:31:07 |             5 |         5 |
| 1983-10-15 06:42:51 |            41 |        41 |
| 2011-04-21 12:34:56 |            16 |        16 |
| 2011-10-30 06:31:41 |            43 |        43 |
| 2011-01-30 14:03:25 |             4 |         4 |
| 2004-10-07 11:19:34 |            41 |        41 |
+---------------------+---------------+-----------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

YEAR

Syntax

```
YEAR(date)
```

Description

Returns the year for the given date, in the range 1000 to 9999, or 0 for the “zero” date.

`SQL_TSI_YEAR` is a synonym for `YEAR`:

Examples

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT * FROM t1;
+---------------------+
| d                   |
+---------------------+
| 2007-01-30 21:31:07 |
| 1983-10-15 06:42:51 |
| 2011-04-21 12:34:56 |
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
| 2004-10-07 11:19:34 |
+---------------------+

SELECT * FROM t1 WHERE YEAR(d) = 2011;
+---------------------+
| d                   |
+---------------------+
| 2011-04-21 12:34:56 |
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
+---------------------+
```

```
SELECT YEAR('1987-01-01');
+--------------------+
| YEAR('1987-01-01') |
+--------------------+
|               1987 |
+--------------------+
```

YEAR Format

```
CREATE TABLE year_format_example (
  description VARCHAR(30),
  example YEAR
);
```

```
INSERT INTO year_format_example VALUES
  ('4-digit numeric year', 1966),
  ('2-digit numeric year', 66),
  ('4-digit string year', '1966'),
  ('2-digit string year', '66');
```

The resulting output would look like this:

```
SELECT * FROM year_format_example;

+----------------------+---------+
| description          | example |
+----------------------+---------+
| 4-digit numeric year |    1966 |
| 2-digit numeric year |    2066 |
| 4-digit string year  |    1966 |
| 2-digit string year  |    2066 |
+----------------------+---------+
```

YEAR Range

```
CREATE TABLE year_range_example (
  description VARCHAR(30),
  example YEAR
);
```

```
INSERT INTO year_range_example VALUES
  ('minimum', 1901),
  ('maximum', 2155),
  ('below minimum', 1900),
  ('above maximum', 2156);
```

If `SQL_MODE` is strict (the default), the example above generates the following error and no values are inserted:

```
ERROR 1264 (22003): Out of range value for column 'example' at row 3
```

If `SQL_MODE` is not strict, the example above generates a warning and (possibly modified) values are inserted:

```
Warning (sql 1264): Out of range value for column 'example' at row 3
Warning (sql 1264): Out of range value for column 'example' at row 4
The resulting data would look like this:
```

```
SELECT * FROM year_range_example;

+---------------+---------+
| description   | example |
+---------------+---------+
| minimum       |    1901 |
| maximum       |    2155 |
| below minimum |    0000 |
| above maximum |    0000 |
+---------------+---------+
```

Zero YEAR

```
CREATE TABLE year_zero_example (
  description VARCHAR(30),
  example YEAR
);
```

```
INSERT INTO year_zero_example VALUES
  ('4-digit numeric zero', 0000),
  ('3-digit numeric zero', 000),
  ('2-digit numeric zero', 00),
  ('1-digit numeric zero', 0),
  ('4-digit string zero', '0000'),
  ('3-digit string zero', '000'),
  ('2-digit string zero', '00'),
  ('1-digit string zero', '0');
```

The resulting data would look like this:

```
SELECT * FROM year_zero_example;

+----------------------+---------+
| description          | example |
+----------------------+---------+
| 4-digit numeric zero |    0000 |
| 3-digit numeric zero |    0000 |
| 2-digit numeric zero |    0000 |
| 1-digit numeric zero |    0000 |
| 4-digit string zero  |    0000 |
| 3-digit string zero  |    2000 |
| 2-digit string zero  |    2000 |
| 1-digit string zero  |    2000 |
+----------------------+---------+
```

See Also

- YEAR data type

This page is licensed: GPLv2, originally from fill_help_tables.sql

YEARWEEK

Syntax

```
YEARWEEK(date), YEARWEEK(date,mode)
```

Description

Returns year and week for a date. The mode argument works exactly like the mode argument to WEEK(). The year in the result may be different from the year in the date argument for the first and the last week of the year.

Examples

```
SELECT YEARWEEK('1987-01-01');
+-----------------------+
| YEARWEEK('1987-01-01') |
+-----------------------+
|                 198652 |
+-----------------------+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT * FROM t1;
+---------------------+
| d                   |
+---------------------+
| 2007-01-30 21:31:07 |
| 1983-10-15 06:42:51 |
| 2011-04-21 12:34:56 |
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
| 2004-10-07 11:19:34 |
+---------------------+
6 rows in set (0.02 sec)
```

```
SELECT YEARWEEK(d) FROM t1 WHERE YEAR(d) = 2011;
+-------------+
| YEARWEEK(d) |
+-------------+
|      201116 |
|      201144 |
|      201105 |
+-------------+
3 rows in set (0.03 sec)
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

Numeric Functions

See ABS.

Calculate absolute value. This function returns the non-negative value of a number, removing any negative sign.

See ACOS.

Calculate arc cosine. This function returns the angle in radians whose cosine is the given number.

See ASIN.

Calculate arc cosine. This function returns the angle in radians whose cosine is the given number.

See ATAN.

Calculate arc tangent. This function returns the angle in radians whose tangent is the given number.

See ATAN2.

Calculate arc tangent of two variables. This function returns the angle in radians between the positive x-axis and the point (X, Y).

See CEIL.

Synonym for CEILING(). Rounds a number up to the nearest integer.

See CEILING.

Round up to the nearest integer. This function returns the smallest integer value that is greater than or equal to the argument.

See CONV.

Convert numbers between bases. This function transforms a number from one numeric base system to another.

See COS.

Calculate cosine. This function returns the cosine of an angle given in radians.

See COT.

Calculate cotangent. This function returns the cotangent of an angle given in radians.

See CRC32.

Compute cyclic redundancy check. This function returns a 32-bit unsigned integer representing the CRC32 checksum of a string.

See CRC32C.

Compute CRC32C checksum. This function calculates a cyclic redundancy check value using the Castagnoli polynomial.

See DEGREES.

Convert radians to degrees. This function transforms an angle measured in radians to its equivalent in degrees.

See DIV.

Perform integer division. This operator divides one number by another and returns the integer result, discarding any remainder.

See EXP.

Calculate exponential value. This function returns the value of e (the base of natural logarithms) raised to the power of the argument.

See FLOOR.

Round down to the nearest integer. This function returns the largest integer value that is less than or equal to the argument.

See LN.

Calculate natural logarithm. This function returns the logarithm of a number to the base e.

See LOG.

Calculate logarithm. This function returns the natural logarithm of a number, or the logarithm to a specified base if two arguments are provided.

See LOG10.

Calculate base-10 logarithm. This function returns the logarithm of a number to the base 10.

See LOG2.

Calculate base-2 logarithm. This function returns the logarithm of a number to the base 2.

See MOD.

Calculate modulo. This function returns the remainder of a number divided by another number.

See OCT.

Convert to octal. This function returns the octal string representation of a numeric argument.

See PI.

Return the value of pi. This function returns the mathematical constant π (approximately 3.141593).

See POW.

Synonym for POWER(). Returns the value of a number raised to the power of another number.

See POWER.

Calculate power. This function returns the value of a number raised to the specified exponent.

See RADIANS.

Convert degrees to radians. This function transforms an angle measured in degrees to its equivalent in radians.

See RAND.

Generate a random number. This function returns a random floating-point value between 0 and 1.

See ROUND.

Round a number. This function rounds a number to a specified number of decimal places.

See SIGN.

Return the sign of a number. This function returns -1, 0, or 1 depending on whether the argument is negative, zero, or positive.

See SIN.

Calculate sine. This function returns the sine of an angle given in radians.

See SQRT.

Calculate square root. This function returns the non-negative square root of a number.

See TAN.

Calculate tangent. This function returns the tangent of an angle given in radians.

See TO_NUMBER.

Convert a string to a number. This Oracle-compatible function parses a string using a specified format and returns a numeric value.

See TRUNCATE.

Truncate a number. This function truncates a number to a specified number of decimal places.

This page is licensed: CC BY-SA / Gnu FDL

ABS

Syntax

```
ABS(X)
```

Description

Returns the absolute (non-negative) value of `X`. If `X` is not a number, it is converted to a numeric type.

Examples

```
SELECT ABS(42);
+---------+
| ABS(42) |
+---------+
|      42 |
+---------+

SELECT ABS(-42);
+----------+
| ABS(-42) |
+----------+
|       42 |
+----------+

SELECT ABS(DATE '1994-01-01');
+------------------------+
| ABS(DATE '1994-01-01') |
+------------------------+
|               19940101 |
+------------------------+
```

See Also

- SIGN()

This page is licensed: GPLv2, originally from fill_help_tables.sql

ACOS

Syntax

```
ACOS(X)
```

Description

Returns the arc cosine of `X`, that is, the value whose cosine is `X`. Returns `NULL` if `X` is not in the range `-1` to `1`.

Examples

```
SELECT ACOS(1);
+---------+
| ACOS(1) |
+---------+
|       0 |
+---------+

SELECT ACOS(1.0001);
+--------------+
| ACOS(1.0001) |
+--------------+
|         NULL |
+--------------+

SELECT ACOS(0);
+-----------------+
| ACOS(0)         |
+-----------------+
| 1.5707963267949 |
+-----------------+

SELECT ACOS(0.234);
+------------------+
| ACOS(0.234)      |
+------------------+
| 1.33460644244679 |
+------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ASIN

Syntax

```
ASIN(X)
```

Description

Returns the arc sine of X, that is, the value whose sine is X. Returns `NULL` if X is not in the range -1 to 1.

Examples

```
SELECT ASIN(0.2);
+--------------------+
| ASIN(0.2)          |
+--------------------+
| 0.2013579207903308 |
+--------------------+

SELECT ASIN('foo');
+-------------+
| ASIN('foo') |
+-------------+
|           0 |
+-------------+

SHOW WARNINGS;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1292 | Truncated incorrect DOUBLE value: 'foo' |
+---------+------+----------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ATAN

Syntax

```
ATAN(X)
```

Description

Returns the arc tangent of X, that is, the value whose tangent is X.

Examples

```
SELECT ATAN(2);
+-------------------+
| ATAN(2)           |
+-------------------+
| 1.1071487177940904 |
+-------------------+

SELECT ATAN(-2);
+--------------------+
| ATAN(-2)           |
+--------------------+
| -1.1071487177940904 |
+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ATAN2

Syntax

```
ATAN(Y,X), ATAN2(Y,X)
```

Description

Returns the arc tangent of the two variables X and Y. It is similar to calculating the arc tangent of Y / X, except that the signs of both arguments are used to determine the quadrant of the result.

Examples

```
SELECT ATAN(-2,2);
+--------------------+
| ATAN(-2,2)         |
+--------------------+
| -0.7853981633974483 |
+--------------------+

SELECT ATAN2(PI(),0);
+-------------------+
| ATAN2(PI(),0)     |
+-------------------+
| 1.5707963267948966 |
+-------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

CEIL

Syntax

```
CEIL(X)
```

Description

`CEIL()` is a synonym for CEILING().

This page is licensed: GPLv2, originally from fill_help_tables.sql

CEILING

Syntax

```
CEILING(X)
```

Description

Returns the smallest integer value not less than X.

Examples

```
SELECT CEILING(1.23);
+---------------+
| CEILING(1.23) |
+---------------+
|             2 |
+---------------+

SELECT CEILING(-1.23);
+----------------+
| CEILING(-1.23) |
+----------------+
|             -1 |
+----------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

CONV

Syntax

```
CONV(N,from_base,to_base)
```

Description

Converts numbers between different number bases. Returns a string representation of the number `N`, converted from base `from_base` to base `to_base`.

Returns `NULL` if any argument is `NULL`, or if the second or third argument are not in the allowed range.

> **CURRENT**
>
> The argument `N` is interpreted as an integer, but may be specified as an integer or a string. The minimum base is 2 and the maximum base is 62. If `to_base` is a negative number, `N` is regarded as a signed number. Otherwise, `N` is treated as unsigned. `CONV()` works with 64-bit precision.

< 11.4

The argument `N` is interpreted as an integer, but may be specified as an integer or a string. The minimum base is 2 and the maximum base is 36. If `to_base` is a negative number, `N` is regarded as a signed number. Otherwise, `N` is treated as unsigned. `CONV()` works with 64-bit precision.

Some shortcuts for this function are also available: BIN(), OCT(), HEX(), UNHEX(). Also, MariaDB allows binary literal values and hexadecimal literal values.

Examples

```
SELECT CONV('a',16,2);
+----------------+
| CONV('a',16,2) |
+----------------+
| 1010           |
+----------------+

SELECT CONV('6E',18,8);
+-----------------+
| CONV('6E',18,8) |
+-----------------+
| 172             |
+-----------------+

SELECT CONV(-17,10,-18);
+------------------+
| CONV(-17,10,-18) |
+------------------+
| -H               |
+------------------+

SELECT CONV(12+'10'+'10'+0xa,10,10);
+-------------------------------+
| CONV(12+'10'+'10'+0xa,10,10) |
+-------------------------------+
| 42                            |
+-------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

COS

Syntax

```
COS(X)
```

Description

Returns the cosine of X, where X is given in radians.

Examples

```
SELECT COS(PI());
+-----------+
| COS(PI()) |
+-----------+
|        -1 |
+-----------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

COT

Syntax

```
COT(X)
```

Description

Returns the cotangent of X.

Examples

```
SELECT COT(42);
+--------------------+
| COT(42)            |
+--------------------+
| 0.4364167060752729 |
+--------------------+

SELECT COT(12);
+---------------------+
| COT(12)             |
+---------------------+
| -1.5726734063976893 |
+---------------------+

SELECT COT(0);
ERROR 1690 (22003): DOUBLE value is out of range in 'cot(0)'
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

CRC32

Syntax

CURRENT

```
CRC32([par,]expr)
```

< 10.8

```
CRC32(expr)
```

Description

Computes a cyclic redundancy check (CRC) value and returns a 32-bit unsigned value. The result is `NULL` if the argument is `NULL`. The argument is expected to be a string and (if possible) is treated as one if it is not.

CURRENT

Uses the alternate Castagnoli polynomia.

Often, CRC is computed in pieces. To facilitate this, there's an optional parameter: CRC32('MariaDB')=CRC32(CRC32('Maria'),'DB').

< 10.8

Uses the ISO 3309 polynomial that used by zlib and many others.

Examples

CURRENT

```
SELECT CRC32(CRC32('Maria'),'DB');
+----------------------------+
| CRC32(CRC32('Maria'),'DB') |
+----------------------------+
|                 4227209140 |
+----------------------------+
```

< 10.8

```
SELECT CRC32('MariaDB');
+------------------+
| CRC32('MariaDB') |
+------------------+
|       4227209140 |
+------------------+

SELECT CRC32('mariadb');
+------------------+
| CRC32('mariadb') |
+------------------+
|       2594253378 |
+------------------+
```

See Also

- CRC32C()

This page is licensed: GPLv2, originally from fill_help_tables.sql

CRC32C

> **Note:**
>
> `CRC32C` is available from MariaDB 10.8.

`CRC32C` is used to compute a cyclic redundancy check (CRC) value using the Castagnoli polynomial.

Syntax

```
CRC32C([par,]expr)
```

Description

MariaDB has always included a native unary function CRC32() that computes the CRC-32 of a string using the ISO 3309 polynomial that used by zlib and many others.

InnoDB and MyRocks use a different polynomial, which was implemented in SSE4.2 instructions that were introduced in the Intel Nehalem microarchitecture. This is commonly called CRC-32C (Castagnoli).

The CRC32C function uses the Castagnoli polynomial.

This allows `SELECT…INTO DUMPFILE` to be used for the creation of files with valid checksums, such as a logically empty InnoDB redo log file `ib_logfile0` corresponding to a particular log sequence number.

The optional parameter allows the checksum to be computed in pieces: CRC32C('MariaDB')=CRC32C(CRC32C('Maria'),'DB').

Examples

```
SELECT CRC32C('MariaDB');
+-------------------+
| CRC32C('MariaDB') |
+-------------------+
|         809606978 |
+-------------------+

SELECT CRC32C(CRC32C('Maria'),'DB');
+------------------------------+
| CRC32C(CRC32C('Maria'),'DB') |
+------------------------------+
|                    809606978 |
+------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

DEGREES

Syntax

```
DEGREES(X)
```

Description

Returns the argument `X`, converted from radians to degrees.

This is the converse of the RADIANS() function.

Examples

```
SELECT DEGREES(PI());
+---------------+
| DEGREES(PI()) |
+---------------+
|           180 |
+---------------+

SELECT DEGREES(PI() / 2);
+-------------------+
| DEGREES(PI() / 2) |
+-------------------+
|                90 |
+-------------------+

SELECT DEGREES(45);
+-----------------+
| DEGREES(45)     |
+-----------------+
| 2578.3100780887 |
+-----------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DIV

Syntax

```
DIV
```

Description

Integer division. Similar to FLOOR(), but is safe with BIGINT values. Incorrect results may occur for non-integer operands that exceed the `BIGINT` range.

If the `ERROR_ON_DIVISION_BY_ZERO` SQL_MODE is used, a division by zero produces an error. Otherwise, it returns `NULL`.

The remainder of a division can be obtained using the MOD operator.

Examples

```
SELECT 300 DIV 7;
+-----------+
| 300 DIV 7 |
+-----------+
|        42 |
+-----------+

SELECT 300 DIV 0;
+-----------+
| 300 DIV 0 |
+-----------+
|      NULL |
+-----------+
```

See Also

- Division operator
- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

EXP

Syntax

```
EXP(X)
```

Description

Returns the value of e (the base of natural logarithms) raised to the power of X. The inverse of this function is LOG() (using a single argument only) or LN().

If `X` is `NULL`, this function returns `NULL`.

Examples

```
SELECT EXP(2);
+------------------+
| EXP(2)           |
+------------------+
| 7.38905609893065 |
+------------------+

SELECT EXP(-2);
+--------------------+
| EXP(-2)            |
+--------------------+
| 0.1353352832366127 |
+--------------------+

SELECT EXP(0);
+--------+
| EXP(0) |
+--------+
|      1 |
+--------+

SELECT EXP(NULL);
+-----------+
| EXP(NULL) |
+-----------+
|      NULL |
+-----------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

FLOOR

Syntax

```
FLOOR(X)
```

Description

Returns the largest integer value not greater than X.

Examples

```
SELECT FLOOR(1.23);
+-------------+
| FLOOR(1.23) |
+-------------+
|           1 |
+-------------+

SELECT FLOOR(-1.23);
+--------------+
| FLOOR(-1.23) |
+--------------+
|           -2 |
+--------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LN

Syntax

```
LN(X)
```

Description

Returns the natural logarithm of X; that is, the base-e logarithm of X. If X is less than or equal to 0, or `NULL`, then `NULL` is returned.

The inverse of this function is EXP().

Examples

```
SELECT LN(2);
+-------------------+
| LN(2)             |
+-------------------+
| 0.693147180559945 |
+-------------------+

SELECT LN(-2);
+--------+
| LN(-2) |
+--------+
|   NULL |
+--------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOG

Syntax

```
LOG(X), LOG(B,X)
```

Description

If called with one parameter, this function returns the natural logarithm of X. If X is less than or equal to 0, then `NULL` is returned.

If called with two parameters, it returns the logarithm of X to the base B. If B is <= 1 or X <= 0, the function returns `NULL`.

If any argument is `NULL`, the function returns `NULL`.

The inverse of this function (when called with a single argument) is the EXP() function.

Examples

LOG(X):

```
SELECT LOG(2);
+-------------------+
| LOG(2)            |
+-------------------+
| 0.693147180559945 |
+-------------------+

SELECT LOG(-2);
+---------+
| LOG(-2) |
+---------+
|    NULL |
+---------+
```

LOG(B,X)

```
SELECT LOG(2,16);
+-----------+
| LOG(2,16) |
+-----------+
|         4 |
+-----------+

SELECT LOG(3,27);
+-----------+
| LOG(3,27) |
+-----------+
|         3 |
+-----------+

SELECT LOG(3,1);
+----------+
| LOG(3,1) |
+----------+
|        0 |
+----------+

SELECT LOG(3,0);
+----------+
| LOG(3,0) |
+----------+
|     NULL |
+----------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOG10

Syntax

```
LOG10(X)
```

Description

Returns the base-10 logarithm of X.

Examples

```
SELECT LOG10(2);
+-------------------+
| LOG10(2)          |
+-------------------+
| 0.301029995663981 |
+-------------------+

SELECT LOG10(100);
+------------+
| LOG10(100) |
+------------+
|          2 |
+------------+

SELECT LOG10(-100);
+-------------+
| LOG10(-100) |
+-------------+
|        NULL |
+-------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOG2

Syntax

```
LOG2(X)
```

Description

Returns the base-2 logarithm of X.

Examples

```
SELECT LOG2(4398046511104);
+---------------------+
| LOG2(4398046511104) |
+---------------------+
|                  42 |
+---------------------+

SELECT LOG2(65536);
+-------------+
| LOG2(65536) |
+-------------+
|          16 |
+-------------+

SELECT LOG2(-100);
+------------+
| LOG2(-100) |
+------------+
|       NULL |
+------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MOD

Syntax

```
MOD(N,M), N % M, N MOD M
```

Description

Modulo operation. Returns the remainder of N divided by M. See also Modulo Operator.

If the `ERROR_ON_DIVISION_BY_ZERO` SQL_MODE is used, any number modulus zero produces an error. Otherwise, it returns `NULL`.

The integer part of a division can be obtained using DIV.

Examples

```
SELECT 1042 % 50;
+-----------+
| 1042 % 50 |
+-----------+
|        42 |
+-----------+

SELECT MOD(234, 10);
+--------------+
| MOD(234, 10) |
+--------------+
|            4 |
+--------------+

SELECT 253 % 7;
+---------+
| 253 % 7 |
+---------+
|       1 |
+---------+

SELECT MOD(29,9);
+-----------+
| MOD(29,9) |
+-----------+
|         2 |
+-----------+

SELECT 29 MOD 9;
+----------+
| 29 MOD 9 |
+----------+
|        2 |
+----------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

OCT

Syntax

```
OCT(N)
```

Description

Returns a string representation of the octal value of N, where N is a longlong (BIGINT) number. This is equivalent to CONV(N,10,8). Returns `NULL` if N is `NULL`.

Examples

```
SELECT OCT(34);
+---------+
| OCT(34) |
+---------+
| 42      |
+---------+

SELECT OCT(12);
+---------+
| OCT(12) |
+---------+
| 14      |
+---------+
```

See Also

- CONV()
- BIN()
- HEX()

This page is licensed: GPLv2, originally from fill_help_tables.sql

PI

Syntax

```
PI()
```

Description

Returns the value of π (pi). The default number of decimal places displayed is six, but MariaDB uses the full double-precision value internally.

Examples

```
SELECT PI();
+----------+
| PI()     |
+----------+
| 3.141593 |
+----------+

SELECT PI()+0.0000000000000000000000;
+-------------------------------+
| PI()+0.0000000000000000000000 |
+-------------------------------+
|      3.1415926535897931159980 |
+-------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

POW

Syntax

```
POW(X,Y)
```

Description

Returns the value of X raised to the power of Y.

`POWER()` is a synonym.

Examples

```
SELECT POW(2,3);
+----------+
| POW(2,3) |
+----------+
|        8 |
+----------+

SELECT POW(2,-2);
+-----------+
| POW(2,-2) |
+-----------+
|      0.25 |
+-----------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

POWER

Syntax

```
POWER(X,Y)
```

Description

This is a synonym for POW(), which returns the value of X raised to the power of Y.

This page is licensed: GPLv2, originally from fill_help_tables.sql

RADIANS

Syntax

```
RADIANS(X)
```

Description

Returns the argument *X*, converted from degrees to radians. Note that π radians equals 180 degrees.

This is the converse of the DEGREES() function.

Examples

```
SELECT RADIANS(45);
+------------------+
| RADIANS(45)      |
+------------------+
| 0.785398163397448 |
+------------------+

SELECT RADIANS(90);
+-----------------+
| RADIANS(90)     |
+-----------------+
| 1.5707963267949 |
+-----------------+

SELECT RADIANS(PI());
+--------------------+
| RADIANS(PI())      |
+--------------------+
| 0.0548311355616075 |
+--------------------+

SELECT RADIANS(180);
+------------------+
| RADIANS(180)     |
+------------------+
| 3.14159265358979 |
+------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

RAND

Syntax

```
RAND(), RAND(N)
```

Description

Returns a random DOUBLE precision floating point value v in the range 0 <= v < 1.0. If a constant integer argument N is specified, it is used as the seed value, which produces a repeatable sequence of column values. In the example below, note that the sequences of values produced by `RAND(3)` is the same both places where it occurs.

In a `WHERE` clause, `RAND()` is evaluated each time the `WHERE` is executed.

Statements using the `RAND()` function are not safe for statement-based replication.

Practical uses

The expression to get a random integer from a given range is the following:

```
FLOOR(min_value + RAND() * (max_value - min_value +1))
```

`RAND()` is often used to read random rows from a table, as follows:

```
SELECT * FROM my_table ORDER BY RAND() LIMIT 10;
```

Warning:

Note, however, that this technique should never be used **on a large table** as it will be extremely slow. MariaDB will read all rows in the table, generate a random value for each of them, order them, and finally will apply the `LIMIT` clause.

Examples

```
CREATE TABLE t (i INT);

INSERT INTO t VALUES(1),(2),(3);

SELECT i, RAND() FROM t;
+------+------------------+
| i    | RAND()           |
+------+------------------+
|    1 | 0.255651095188829 |
|    2 | 0.833920199269355 |
|    3 |  0.40264774151393 |
+------+------------------+

SELECT i, RAND(3) FROM t;
+------+------------------+
| i    | RAND(3)          |
+------+------------------+
|    1 |  0.90576975597606 |
|    2 | 0.373079058130345 |
|    3 | 0.148086053457191 |
+------+------------------+

SELECT i, RAND() FROM t;
+------+------------------+
| i    | RAND()           |
+------+------------------+
|    1 | 0.511478140495232 |
|    2 | 0.349447508668012 |
|    3 | 0.212803152588013 |
+------+------------------+
```

Using the same seed, the same sequence will be returned:

```
SELECT i, RAND(3) FROM t;
+------+------------------+
| i    | RAND(3)          |
+------+------------------+
|    1 |  0.90576975597606 |
|    2 | 0.373079058130345 |
|    3 | 0.148086053457191 |
+------+------------------+
```

Generating a random number from 5 to 15:

```
SELECT FLOOR(5 + (RAND() * 11));
```

See Also

- Techniques for Efficiently Finding a Random Row
- rand_seed1 and rand_seed2 system variables

This page is licensed: GPLv2, originally from fill_help_tables.sql

ROUND

Syntax

```
ROUND(X), ROUND(X,D)
```

Description

Rounds the argument `X` to `D` decimal places. `D` defaults to `0` if not specified. `D` can be negative to cause `D` digits left of the decimal point of the value `X` to become zero.

The rounding algorithm depends on the data type of `X`:

- For floating-point types (FLOAT, DOUBLE) the C libraries rounding function is used, so the behavior *may* differ between operating systems.
- For fixed-point types (DECIMAL, DEC/NUMBER/FIXED) the “round half up” rule is used, meaning that e.g. a value ending in exactly .5 is always rounded up.

Examples

```
SELECT ROUND(-1.23);
+--------------+
| ROUND(-1.23) |
+--------------+
|           -1 |
+--------------+

SELECT ROUND(-1.58);
+--------------+
| ROUND(-1.58) |
+--------------+
|           -2 |
+--------------+

SELECT ROUND(1.58);
+-------------+
| ROUND(1.58) |
+-------------+
|           2 |
+-------------+

SELECT ROUND(1.298, 1);
+-----------------+
| ROUND(1.298, 1) |
+-----------------+
|             1.3 |
+-----------------+

SELECT ROUND(1.298, 0);
+-----------------+
| ROUND(1.298, 0) |
+-----------------+
|               1 |
+-----------------+

SELECT ROUND(23.298, -1);
+------------------+
| ROUND(23.298, -1) |
+------------------+
|                20 |
+------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SIGN

Syntax

```
SIGN(X)
```

Description

Returns the sign of the argument as -1, 0, or 1, depending on whether X is negative, zero, or positive.

Examples

```
SELECT SIGN(-32);
+-----------+
| SIGN(-32) |
+-----------+
|        -1 |
+-----------+

SELECT SIGN(0);
+---------+
| SIGN(0) |
+---------+
|       0 |
+---------+

SELECT SIGN(234);
+-----------+
| SIGN(234) |
+-----------+
|         1 |
+-----------+
```

See Also

- ABS()

This page is licensed: GPLv2, originally from fill_help_tables.sql

SIN

Syntax

```
SIN(X)
```

Description

Returns the sine of X, where X is given in radians.

Examples

```
SELECT SIN(1.5707963267948966);
+-------------------------+
| SIN(1.5707963267948966) |
+-------------------------+
|                       1 |
+-------------------------+

SELECT SIN(PI());
+----------------------+
| SIN(PI())            |
+----------------------+
| 1.22460635382238e-16 |
+----------------------+

SELECT ROUND(SIN(PI()));
+------------------+
| ROUND(SIN(PI())) |
+------------------+
|                0 |
+------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SQRT

Syntax

```
SQRT(X)
```

Description

Returns the square root of X. If X is negative, `NULL` is returned.

Examples

```
SELECT SQRT(4);
+---------+
| SQRT(4) |
+---------+
|       2 |
+---------+

SELECT SQRT(20);
+------------------+
| SQRT(20)         |
+------------------+
| 4.47213595499958 |
+------------------+

SELECT SQRT(-16);
+-----------+
| SQRT(-16) |
+-----------+
|      NULL |
+-----------+

SELECT SQRT(1764);
+------------+
| SQRT(1764) |
+------------+
|         42 |
+------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TAN

Syntax

```
TAN(X)
```

Description

Returns the tangent of X, where X is given in radians.

Examples

```
SELECT TAN(0.7853981633974483);
+-------------------------+
| TAN(0.7853981633974483) |
+-------------------------+
|      0.9999999999999999 |
+-------------------------+

SELECT TAN(PI());
+----------------------+
| TAN(PI())            |
+----------------------+
| -1.22460635382238e-16 |
+----------------------+

SELECT TAN(PI()+1);
+-----------------+
| TAN(PI()+1)     |
+-----------------+
| 1.5574077246549 |
+-----------------+

SELECT TAN(RADIANS(PI()));
+--------------------+
| TAN(RADIANS(PI())) |
+--------------------+
| 0.0548861508080033 |
+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

TO_NUMBER

Note:

`TO_NUMBER` is available from MariaDB 12.2.

`TO_NUMBER` converts an expression to the NUMERIC data type.

Syntax

```
TO_NUMBER(number_or_string_subject)
TO_NUMBER(string_subject,string_format)
```

Description

The function returns the `DOUBLE` data type for all signatures and input data types.

The format parser understands the following components:

- Digits: 0, 9
- Hex digits: X
- Group separators: comma (`,`) and G
- Decimal delimiters: period (`.`) and D
- Approximate number signature: EEEE
- Currency/numeric flags: $ and B
- Currency signatures: C, L, U
- Sign signatures: S, MI, PR
- Special format signatures: V, TM, TM9, TME
- Format flag: FM

The function was introduced for Oracle compatibility, but does not include the following features:

- The `ON CONVERSION ERROR` clause
- The third parameter (nlsparam)
- Internationalized components: G, D, C, L, U

These features are planned to be implemented via MDEV-36978.

Examples

```
SELECT TO_NUMBER('100.00');
+---------------------+
| TO_NUMBER('100.00') |
+---------------------+
|                 100 |
+---------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

TRUNCATE

This page documents the `TRUNCATE` function. See TRUNCATE TABLE for the DDL statement.

Syntax

```
TRUNCATE(X,D)
```

Description

Returns the number X, truncated to D decimal places. If D is 0, the result has no decimal point or fractional part. D can be negative to cause D digits left of the decimal point of the value X to become zero.

Examples

```
SELECT TRUNCATE(1.223,1);
+------------------+
| TRUNCATE(1.223,1) |
+------------------+
|               1.2 |
+------------------+

SELECT TRUNCATE(1.999,1);
+------------------+
| TRUNCATE(1.999,1) |
+------------------+
|               1.9 |
+------------------+

SELECT TRUNCATE(1.999,0);
+------------------+
| TRUNCATE(1.999,0) |
+------------------+
|                 1 |
+------------------+

SELECT TRUNCATE(-1.999,1);
+-------------------+
| TRUNCATE(-1.999,1) |
+-------------------+
|               -1.9 |
+-------------------+

SELECT TRUNCATE(122,-2);
+-----------------+
| TRUNCATE(122,-2) |
+-----------------+
|              100 |
+-----------------+

SELECT TRUNCATE(10.28*100,0);
+----------------------+
| TRUNCATE(10.28*100,0) |
+----------------------+
|                 1028 |
+----------------------+
```

See Also

- TRUNCATE TABLE

This page is licensed: GPLv2, originally from fill_help_tables.sql

XXH3

> **Note:**
>
> `XXH3` was added in MariaDB 13.1.

Syntax

```
XXH3(expr)
```

Description

`XXH3` returns a fast, non-cryptographic xxHash of a text value as a 64-bit number (`BIGINT UNSIGNED`). It's useful for checksums, change detection, and grouping or bucketing rows. For a 32-bit result, use XXH32().

The argument must be a text value: a string, `BLOB` / `TEXT`, `JSON`, `ENUM`, or `SET`. Numbers, dates, and other non-text values are rejected with an error; CAST() them to a string first.

Hashing `NULL` returns `NULL`. Hashing an empty string returns `0`.

The result depends on the collation, not on the raw bytes. Values that count as equal under the collation hash to the same number (for example, `'abc'` and `'ABC'` under a case-insensitive collation). The same text in a different character set or collation gives a different number.

> **Warning:**
>
> When comparing a column to a value, keep both sides in the same collation. If the hashed column and the value use different collations, they won't match, so a hash-based index or lookup can quietly find nothing. Hash the binary form with `CAST(... AS BINARY)` (or the `BINARY` operator) when you want a byte-exact hash that ignores collation.

Predicting the partition for KEY partitioning

The same XXH3 algorithm is available as a KEY partitioning algorithm (`PARTITION BY KEY ALGORITHM=XXH3`). For a table partitioned on a single column, the partition a value lands in is `MOD(XXH3(value), number_of_partitions)` — handy for knowing where a row will go, or for partition pruning. The collation rule above still applies: the value must use the column's collation to predict correctly.

This shortcut holds only for a plain, single-column `KEY`. It doesn't apply to LINEAR KEY (unless the number of partitions is a power of two), to multi-column keys, or to `NULL` values (the server places those itself).

> **Note:**
>
> `XXH32` is the 32-bit variant. The `XXH64()` and `XXH128()` functions are not part of this release.

Examples

```
SELECT XXH3('abc');
+---------------------+
| XXH3('abc')         |
+---------------------+
| 5167207402487786768 |
+---------------------+
```

The result depends on the session collation, so a bare literal reproduces only under the same collation (the result above is for the default `utf8mb4_uca1400_ai_ci`). For a byte-exact hash that doesn't depend on collation, hash the binary form:

```
SELECT XXH3(CAST('abc' AS BINARY));
+-----------------------------+
| XXH3(CAST('abc' AS BINARY)) |
+-----------------------------+
|         8696274497037089104 |
+-----------------------------+
```

`NULL` and the empty string:

```
SELECT XXH3(NULL), XXH3('');
+------------+----------+
| XXH3(NULL) | XXH3('') |
+------------+----------+
|       NULL |        0 |
+------------+----------+
```

Equal values under a case-insensitive collation hash to the same number:

```
SELECT XXH3(_utf8mb4'ABC' COLLATE utf8mb4_general_ci)
     = XXH3(_utf8mb4'abc' COLLATE utf8mb4_general_ci) AS equal_ci;
+----------+
| equal_ci |
+----------+
|        1 |
+----------+
```

A non-text argument is rejected:

```
SELECT XXH3(11223344);
ERROR 4079 (HY000): Illegal parameter data type int for operation 'XXH3'
```

See Also

- XXH32() - the 32-bit variant
- CRC32(), CRC32C() - other non-cryptographic checksums

This page is licensed: CC BY-SA / Gnu FDL

XXH32

Note:

`XXH32` was added in MariaDB 13.1.

Syntax

```
XXH32(expr)
```

Description

`XXH32` returns a fast, non-cryptographic xxHash of a text value as a 32-bit number (`INT UNSIGNED`). It's useful for checksums, change detection, and grouping or bucketing rows. For a 64-bit result, use XXH3().

The argument must be a text value: a string, `BLOB` / `TEXT`, `JSON`, `ENUM`, or `SET`. Numbers, dates, and other non-text values are rejected with an error; CAST() them to a string first.

Hashing `NULL` returns `NULL`. Hashing an empty string returns `0`.

The result depends on the collation, not on the raw bytes. Values that count as equal under the collation hash to the same number (for example, `'abc'` and `'ABC'` under a case-insensitive collation). The same text in a different character set or collation gives a different number.

Warning:

When comparing a column to a value, keep both sides in the same collation. If the hashed column and the value use different collations, they won't match, so a hash-based index or lookup can quietly find nothing. Hash the binary form with `CAST(... AS BINARY)` (or the `BINARY` operator) when you want a byte-exact hash that ignores collation.

Predicting the partition for KEY partitioning

The same XXH32 algorithm is available as a KEY partitioning algorithm (`PARTITION BY KEY ALGORITHM=XXH32`). For a table partitioned on a single column, the partition a value lands in is `MOD(XXH32(value), number_of_partitions)` — handy for knowing where a row will go, or for partition pruning. The collation rule above still applies: the value must use the column's collation to predict correctly.

This shortcut holds only for a plain, single-column `KEY`. It doesn't apply to LINEAR KEY (unless the number of partitions is a power of two), to multi-column keys, or to `NULL` values (the server places those itself).

> **Note:**
>
> `XXH3` is the 64-bit variant. The `XXH64()` and `XXH128()` functions are not part of this release.

Examples

```
SELECT XXH32('abc');
+--------------+
| XXH32('abc') |
+--------------+
|    560666315 |
+--------------+
```

The result depends on the session collation, so a bare literal reproduces only under the same collation (the result above is for the default `utf8mb4_uca1400_ai_ci`). For a byte-exact hash that doesn't depend on collation, hash the binary form:

```
SELECT XXH32(CAST('abc' AS BINARY));
+------------------------------+
| XXH32(CAST('abc' AS BINARY)) |
+------------------------------+
|                    852579327 |
+------------------------------+
```

`NULL` and the empty string:

```
SELECT XXH32(NULL), XXH32('');
+-------------+-----------+
| XXH32(NULL) | XXH32('') |
+-------------+-----------+
|        NULL |         0 |
+-------------+-----------+
```

Equal values under a case-insensitive collation hash to the same number:

```
SELECT XXH32(_utf8mb4'ABC' COLLATE utf8mb4_general_ci)
     = XXH32(_utf8mb4'abc' COLLATE utf8mb4_general_ci) AS equal_ci;
+----------+
| equal_ci |
+----------+
|        1 |
+----------+
```

A non-text argument is rejected:

```
SELECT XXH32(11223344);
ERROR 4079 (HY000): Illegal parameter data type int for operation 'XXH32'
```

See Also

- XXH3() - the 64-bit variant
- CRC32(), CRC32C() - other non-cryptographic checksums

This page is licensed: CC BY-SA / Gnu FDL

Pseudo Columns

See _rowid.

Access the unique row identifier. This pseudo column often aliases the primary key, allowing direct row access even if the primary key column name is unknown.

This page is licensed: CC BY-SA / Gnu FDL

_rowid

Syntax

```
_rowid
```

Description

The `_rowid` pseudo column is mapped to the primary key in the related table. This can be used as a replacement of the `rowid` pseudo column in other databases. Another usage is to simplify sql queries as one doesn't have to know the name of the primary key.

Examples

```
CREATE TABLE t1 (a INT PRIMARY KEY, b VARCHAR(80));
INSERT INTO t1 VALUES (1,"one"),(2,"two");
SELECT * FROM t1 WHERE _rowid=1;
```

```
+---+------+
| a | b    |
+---+------+
| 1 | one  |
+---+------+
```

```
UPDATE t1 SET b="three" WHERE _rowid=2;
SELECT * FROM t1 WHERE _rowid>=1 AND _rowid<=10;
```

```
+---+-------+
| a | b     |
+---+-------+
| 1 | one   |
| 2 | three |
+---+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

Secondary Functions

See Bit Functions and Operators.

Perform bitwise logic and manipulation. This section covers operators like AND, OR, XOR, and functions for shifting bits or counting set bits in binary data.

See Encryption, Hashing and Compression Functions.

Learn about encryption, hashing, and compression functions. This section details SQL functions for securing data, generating hashes, and compressing/decompressing information within your database.

See Information Functions.

Retrieve system and session metadata. This section details functions to access database names, user details, version info, and query statistics like row counts.

See Miscellaneous Functions.

Explore specialized utilities. This category includes functions for benchmarking, managing UUIDs, converting IP addresses, and handling binary data formats.

This page is licensed: CC BY-SA / Gnu FDL

Bit Functions and Operators

See BIT_COUNT.

Count the set bits. This function returns the number of bits that are set to 1 in the binary representation of a number.

See ~.

Invert all bits. This unary operator flips every bit in the operand, changing 1s to 0s and 0s to 1s.

See |.

Perform bitwise OR. This operator compares bits and returns 1 if at least one of the corresponding bits is 1.

See ^.

Perform bitwise exclusive OR. This operator returns 1 only if the corresponding bits of the operands are different.

See &.

Perform bitwise AND. This operator compares bits of two operands and returns 1 only if both corresponding bits are 1.

See Parentheses.

Control operator precedence. Parentheses are used to group expressions and enforce the order of evaluation in complex operations.

See <<.

Shift bits to the left. This operator shifts the binary representation of a number to the left by a specified number of positions, filling with zeros.

See >>.

Shift bits to the right. This operator shifts the binary representation of a number to the right by a specified number of positions.

See TRUE FALSE.

Boolean literal for 1 = TRUE or 0 = FALSE, respectively.

This page is licensed: CC BY-SA / Gnu FDL

BIT_COUNT

Syntax

```
BIT_COUNT(N)
```

Description

Returns the number of bits that are set in the argument N.

Examples

```
SELECT BIT_COUNT(29), BIT_COUNT(b'101010');
+---------------+----------------------+
| BIT_COUNT(29) | BIT_COUNT(b'101010') |
+---------------+----------------------+
|             4 |                    3 |
+---------------+----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

~

Syntax

```
~
```

Description

Bitwise NOT. Converts the value to 4 bytes binary and inverts all bits.

Examples

```
SELECT 3 & ~1;
+--------+
| 3 & ~1 |
+--------+
|      2 |
+--------+

SELECT 5 & ~1;
+--------+
| 5 & ~1 |
+--------+
|      4 |
+--------+
```

See Also

- Operator Precedence

This page is licensed: CC BY-SA / Gnu FDL

|

Syntax

```
|
```

Description

Bitwise OR. Converts the values to binary and compares bits. If either of the corresponding bits has a value of 1, the resulting bit is also 1.

See also bitwise AND.

Examples

```
SELECT 2|1;
+-----+
| 2|1 |
+-----+
|   3 |
+-----+

SELECT 29 | 15;
+---------+
| 29 | 15 |
+---------+
|      31 |
+---------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

^

Syntax

```
^
```

Description

Bitwise XOR. Converts the values to binary and compares bits. If one (and only one) of the corresponding bits is 1 is the resulting bit also 1.

Examples

```
SELECT 1 ^ 1;
+-------+
| 1 ^ 1 |
+-------+
|     0 |
+-------+

SELECT 1 ^ 0;
+-------+
| 1 ^ 0 |
+-------+
|     1 |
+-------+

SELECT 11 ^ 3;
+--------+
| 11 ^ 3 |
+--------+
|      8 |
+--------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

&

Syntax

```
&
```

Description

Bitwise AND. Converts the values to binary and compares bits. Only if both the corresponding bits are 1 is the resulting bit also 1.

See also bitwise OR.

Examples

```
SELECT 2&1;
+-----+
| 2&1 |
+-----+
|   0 |
+-----+

SELECT 3&1;
+-----+
| 3&1 |
+-----+
|   1 |
+-----+

SELECT 29 & 15;
+---------+
| 29 & 15 |
+---------+
|      13 |
+---------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

Parentheses

Parentheses are sometimes called precedence operators - this means that they can be used to change the other operator's precedence in an expression. The expressions that are written between parentheses are computed before the expressions that are written outside. Parentheses must always contain an expression (that is, they cannot be empty), and can be nested.

For example, the following expressions could return different results:

- `NOT a OR b`
- `NOT (a OR b)`

In the first case, `NOT` applies to `a`, so if `a` is `FALSE` or `b` is `TRUE`, the expression returns `TRUE`. In the second case, `NOT` applies to the result of `a OR b`, so if at least one of `a` or `b` is `TRUE`, the expression is `TRUE`.

When the precedence of operators is not intuitive, you can use parentheses to make it immediately clear for whoever reads the statement.

The precedence of the `NOT` operator can also be affected by the `HIGH_NOT_PRECEDENCE` SQL_MODE flag.

Other uses

Parentheses must always be used to enclose subqueries.

Parentheses can also be used in a JOIN statement between multiple tables to determine which tables must be joined first.

Also, parentheses are used to enclose the list of parameters to be passed to built-in functions, user-defined functions and stored routines. However, when no parameter is passed to a stored procedure, parentheses are optional. For builtin functions and user-defined functions, spaces are not allowed between the function name and the open parenthesis, unless the `IGNORE_SPACE` SQL_MODE is set. For stored routines (and for functions if `IGNORE_SPACE` is set) spaces are allowed before the open parenthesis, including tab characters and new line characters.

Syntax errors

If there are more open parentheses than closed parentheses, the error usually looks like this:

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MariaDB server version for the right syntax to use near '' a
t line 1
```

Note the empty string.

If there are more closed parentheses than open parentheses, the error usually looks like this:

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MariaDB server version for the right syntax to use near ')'
at line 1
```

Note the quoted closed parenthesis.

This page is licensed: CC BY-SA / Gnu FDL

<<

Syntax

```
value1 << value2
```

Description

Converts a longlong (BIGINT) number (*value1*) to binary and shifts *value2* units to the left.

Examples

```
SELECT 1 << 2;
+--------+
| 1 << 2 |
+--------+
|      4 |
+--------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

>>

Syntax

```
value1 >> value2
```

Description

Converts a longlong (BIGINT) number (*value1*) to binary and shifts *value2* units to the right.

Examples

```
SELECT 4 >> 2;
+--------+
| 4 >> 2 |
+--------+
|      1 |
+--------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

TRUE FALSE

Description

The constants `TRUE` and `FALSE` evaluate to 1 and 0, respectively. The constant names can be written in any lettercase.

Examples

```
SELECT TRUE, true, FALSE, false;
+------+------+-------+-------+
| TRUE | TRUE | FALSE | FALSE |
+------+------+-------+-------+
|    1 |    1 |     0 |     0 |
+------+------+-------+-------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

Encryption, Hashing and Compression Functions

- AES_DECRYPT
- AES_ENCRYPT
- COMPRESS
- DECODE
- DES_DECRYPT
- DES_ENCRYPT
- ENCODE
- ENCRYPT
- KDF
- MD5
- OLD_PASSWORD

- PASSWORD
- RANDOM_BYTES
- SHA1
- SHA2
- UNCOMPRESS
- UNCOMPRESSED_LENGTH

This page is licensed: CC BY-SA / Gnu FDL

AES_DECRYPT

Syntax

CURRENT

```
AES_ENCRYPT(crypt_str, key_str, [, iv [, mode]])
```

< 11.2

```
AES_DECRYPT(crypt_str,key_str)
```

Description

This function allows decryption of data using the official AES (Advanced Encryption Standard) algorithm. For more information, see the description of AES_ENCRYPT().

CURRENT

The function supports an initialization vector, and control of the block encryption mode. The default mode is specified by the block_encryption_mode system variable, which can be changed when calling the function with a mode. *mode* is aes-{128,192,256}-{ecb,cbc,ctr} for example: “AES-128-cbc”.

For modes that require it, the initialization_vector *iv* should be 16 bytes (it can be longer, but the extra bytes are ignored). A shorter *iv*, where one is required, results in the function returning `NULL`. Calling RANDOM_BYTES(16) will generate a random series of bytes that can be used for the *iv*.

Examples

```
SELECT HEX(AES_ENCRYPT('foo', 'bar', '0123456789abcdef', 'aes-128-ctr')) AS x;
+--------+
| x      |
+--------+
| C57C4B |
+--------+

SELECT AES_DECRYPT(x'C57C4B', 'bar', '0123456789abcdef', 'aes-128-ctr');
+-----------------------------------------------------------------+
| AES_DECRYPT(x'C57C4B', 'bar', '0123456789abcdef', 'aes-128-ctr') |
+-----------------------------------------------------------------+
| foo                                                             |
+-----------------------------------------------------------------+
```

< 11.2

The function does **not** support an initialization vector.

This page is licensed: GPLv2, originally from fill_help_tables.sql

AES_ENCRYPT

Syntax

CURRENT

```
AES_ENCRYPT(str, key, [, iv [, mode]])
```

< 11.2

```
AES_ENCRYPT(str,key_str)
```

Description

`AES_ENCRYPT()` and AES_DECRYPT() allow encryption and decryption of data using the official AES (Advanced Encryption Standard) algorithm, previously known as "Rijndael." Encoding with a 128-bit key length is used (from MariaDB 11.2.0, this is the default, and can be changed). 128 bits is much faster and is secure enough for most purposes.

`AES_ENCRYPT()` encrypts a string *`str`* using the key *`key_str`*, and returns a binary string.

`AES_DECRYPT()` decrypts the encrypted string and returns the original string.

The input arguments may be any length. If either argument is `NULL`, the result of this function is also `NULL`.

Because AES is a block-level algorithm, padding is used to encode uneven length strings and so the result string length may be calculated using this formula:

```
16 x (trunc(string_length / 16) + 1)
```

If `AES_DECRYPT()` detects invalid data or incorrect padding, it returns `NULL`. However, it is possible for `AES_DECRYPT()` to return a non-`NULL` value (possibly garbage) if the input data or the key is invalid.

MariaDB starting with 11.2

CURRENT

The function supports an initialization vector, and control of the block encryption mode. The default mode is specified by the block_encryption_mode system variable, which can be changed when calling the function with a mode. *mode* is aes-{128,192,256}-{ecb,cbc,ctr} for example: "AES-128-cbc".
`AES_ENCRYPT(str, key)` can no longer be used in persistent virtual columns (and the like).

TAB 2

The function does **not** support an initialization vector.

Examples

CURRENT

```
SELECT HEX(AES_ENCRYPT('foo', 'bar', '0123456789abcdef', 'aes-256-cbc')) AS x;
+----------------------------------+
| x                                |
+----------------------------------+
| 42A3EB91E6DFC40A900D278F99E0726E |
+----------------------------------+
```

< 11.2

```
INSERT INTO t VALUES (AES_ENCRYPT('text',SHA2('password',512)));
```

See Also

- RANDOM_BYTES() is a function for generating good encryption keys for `AES_ENCRYPT`.
- KDF() is a key derivation function which is useful if an authentication validation against the value is required without data being able to be decrypted.

This page is licensed: GPLv2, originally from fill_help_tables.sql

COMPRESS

Syntax

```
COMPRESS(string_to_compress)
```

Description

Compresses a string and returns the result as a binary string. This function requires MariaDB to have been compiled with a compression library such as zlib. Otherwise, the return value is always `NULL`. The compressed string can be uncompressed with UNCOMPRESS().

The have_compress server system variable indicates whether a compression library is present.

Examples

```
SELECT LENGTH(COMPRESS(REPEAT('a',1000)));
+-----------------------------------+
| LENGTH(COMPRESS(REPEAT('a',1000))) |
+-----------------------------------+
|                                21 |
+-----------------------------------+

SELECT LENGTH(COMPRESS(''));
+----------------------+
| LENGTH(COMPRESS('')) |
+----------------------+
|                    0 |
+----------------------+

SELECT LENGTH(COMPRESS('a'));
+-----------------------+
| LENGTH(COMPRESS('a')) |
+-----------------------+
|                    13 |
+-----------------------+

SELECT LENGTH(COMPRESS(REPEAT('a',16)));
+---------------------------------+
| LENGTH(COMPRESS(REPEAT('a',16))) |
+---------------------------------+
|                              15 |
+---------------------------------+
```

See Also

- Automatic compression of columns

This page is licensed: GPLv2, originally from fill_help_tables.sql

DECODE

Syntax

```
DECODE(crypt_str,pass_str)
```

In Oracle mode:

```
DECODE(expr, search_expr, result_expr [, search_expr2, result_expr2 ...] [default_expr])
```

In all modes:

```
DECODE_ORACLE(expr, search_expr, result_expr [, search_expr2, result_expr2 ...] [default_expr])
```

Description

In the default mode, `DECODE` decrypts the encrypted string *`crypt_str`* using *`pass_str`* as the password. *crypt_str* should be a string returned from ENCODE(). The resulting string will be the original string only if *`pass_str`* is the same.

In Oracle mode, `DECODE` compares *`expr`* to the search expressions, in order. If it finds a match, the corresponding result expression is returned. If no matches are found, the default expression is returned, or `NULL` if no default is provided.

`NULL` values are treated as equivalent.

`DECODE_ORACLE` is a synonym for the Oracle-mode version of the function, and is available in all modes.

Examples

```
SELECT DECODE_ORACLE(2+1,3*1,'found1',3*2,'found2','default');
+-------------------------------------------------------+
| DECODE_ORACLE(2+1,3*1,'found1',3*2,'found2','default') |
+-------------------------------------------------------+
| found1                                                |
+-------------------------------------------------------+

SELECT DECODE_ORACLE(2+4,3*1,'found1',3*2,'found2','default');
+-------------------------------------------------------+
| DECODE_ORACLE(2+4,3*1,'found1',3*2,'found2','default') |
+-------------------------------------------------------+
| found2                                                |
+-------------------------------------------------------+

SELECT DECODE_ORACLE(2+2,3*1,'found1',3*2,'found2','default');
+-------------------------------------------------------+
| DECODE_ORACLE(2+2,3*1,'found1',3*2,'found2','default') |
+-------------------------------------------------------+
| default                                               |
+-------------------------------------------------------+
```

`NULL` values are treated as equivalent:

```
SELECT DECODE_ORACLE(NULL,NULL,'Nulls are equivalent','Nulls are not equivalent');
+-----------------------------------------------------------------------------+
| DECODE_ORACLE(NULL,NULL,'Nulls are equivalent','Nulls are not equivalent') |
+-----------------------------------------------------------------------------+
| Nulls are equivalent                                                        |
+-----------------------------------------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DES_DECRYPT

> **Warning:**
>
> `DES_DECRYPT` was **removed in MariaDB 13.0**, together with the `--des-key-file` option and the `FLUSH DES_KEY_FILE` statement. It was deprecated in earlier releases. Use AES_DECRYPT() instead. The description below applies to MariaDB releases before 13.0.

Syntax

```
DES_DECRYPT(crypt_str[,key_str])
```

Description

Decrypts a string encrypted with DES_ENCRYPT(). If an error occurs, this function returns `NULL`.

This function works only if MariaDB has been configured with TLS support.

If no `key_str` argument is given, `DES_DECRYPT()` examines the first byte of the encrypted string to determine the DES key number that was used to encrypt the original string, and then reads the key from the DES key file to decrypt the message. For this to work, the user must have the SUPER privilege. The key file can be specified with the `--des-key-file` server option.

If you pass this function a `key_str` argument, that string is used as the key for decrypting the message.

If the `crypt_str` argument does not appear to be an encrypted string, MariaDB returns the given `crypt_str`.

This page is licensed: GPLv2, originally from fill_help_tables.sql

DES_ENCRYPT

> **Warning:**
>
> `DES_ENCRYPT` was **removed in MariaDB 13.0**, together with the `--des-key-file` option and the `FLUSH DES_KEY_FILE` statement. It was deprecated in earlier releases. Use AES_ENCRYPT() instead. The description below applies to MariaDB releases before 13.0.

Syntax

```
DES_ENCRYPT(str[,{key_num|key_str}])
```

Description

Encrypts the string with the given key using the Triple-DES algorithm.

This function works only if MariaDB has been configured with TLS support.

The encryption key to use is chosen based on the second argument to `DES_ENCRYPT()`, if one was given. With no argument, the first key from the DES key file is used. With a *`key_num`* argument, the given key number (0-9) from the DES key file is used. With a *`key_str`* argument, the given key string is used to encrypt *`str`*.

The key file can be specified with the `--des-key-file` server option.

The return string is a binary string where the first character is `CHAR(128 | key_num)`. If an error occurs, `DES_ENCRYPT()` returns `NULL`.

The 128 is added to make it easier to recognize an encrypted key. If you use a string key, *`key_num`* is 127.

The string length for the result is given by this formula:

```
new_len = orig_len + (8 - (orig_len % 8)) + 1
```

Each line in the DES key file has the following format:

```
key_num des_key_str
```

Each *`key_num`* value must be a number in the range from 0 to 9. Lines in the file may be in any order. *`des_key_str`* is the string that is used to encrypt the message. There should be at least one space between the number and the key. The first key is the default key that is used if you do not specify any key argument to `DES_ENCRYPT()`.

You can tell MariaDB to read new key values from the key file with the `FLUSH DES_KEY_FILE` statement. This requires the RELOAD privilege.

One benefit of having a set of default keys is that it gives applications a way to check for the existence of encrypted column values, without giving the end user the right to decrypt those values.

Examples

```
SELECT customer_address FROM customer_table
   WHERE crypted_credit_card = DES_ENCRYPT('credit_card_number');
```

See Also

- DES_DECRYPT()

This page is licensed: GPLv2, originally from fill_help_tables.sql

ENCODE

Syntax

```
ENCODE(str,pass_str)
```

Description

> **Warning:**
>
> `ENCODE` is not considered cryptographically secure, and should not be used for password encryption.

Encrypt `str` using `pass_str` as the password. To decrypt the result, use DECODE().

The result is a binary string of the same length as `str`.

The strength of the encryption is based on how good the random generator is.

It is not recommended to rely on the encryption performed by the ENCODE function. Using a salt value (changed when a password is updated) will improve matters somewhat, but for storing passwords, consider a more cryptographically secure function, such as SHA2().

Examples

```
ENCODE('not so secret text', CONCAT('random_salt','password'))
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

ENCRYPT

Syntax

```
ENCRYPT(str[,salt])
```

Description

Encrypts a string using the Unix crypt() system call, returning an encrypted binary string. The `salt` argument should be a string with at least two characters or the returned result will be NULL. If no salt argument is given, a random value of sufficient length is used.

It is not recommended to use ENCRYPT() with utf16, utf32 or ucs2 multi-byte character sets because the crypt() system call expects a string terminated with a zero byte.

Note that the underlying crypt() system call may have some limitations, such as ignoring all but the first eight characters.

If the have_crypt system variable is set to `NO` (because the crypt() system call is not available), the ENCRYPT function will always return NULL.

Examples

```
SELECT ENCRYPT('encrypt me');
+-----------------------+
| ENCRYPT('encrypt me') |
+-----------------------+
| 4I5BsEx0lqTDk         |
+-----------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

KDF

> **Note:**
>
> `KDF()` is a key derivation function available from MariaDB 11.3.0.

Syntax

```
KDF(key_str, salt [, {info | iterations} [, kdf_name [, width ]]])
```

Description

`KDF` is a key derivation function, similar to OpenSSL's EVP_KDF_derive(). The purpose of a KDF is to be slow, so if the calculated value is lost/stolen, the original `key_str` is not achievable easily with modern GPU. KDFs are therefore an ideal replacement for password hashes. KDFs can also pad out a password secret to the number of bits used in encryption algorithms.

For generating good encryption keys for AES_ENCRYPT a less expensive but cryptographically secure function like RANDOM_BYTES is recommended.

- `kdf_name` is "hkdf" or "pbkdf2_hmac" (default).
- Width (in bits) can be any number divisible by 8, by default it's taken from @@block_encryption_mode.
- Iterations must be positive, and is 1000 by default.

Note that OpenSSL 1.0 doesn't support HKDF, so in this case `NULL` is returned. This OpenSSL version is still used in SLES 12 and CentOS 7.

Examples

```
select hex(kdf('foo', 'bar', 'infa', 'hkdf'));
+---------------------------------------+
| hex(kdf('foo', 'bar', 'infa', 'hkdf')) |
+---------------------------------------+
| 612875F859CFB4EE0DFEFF9F2A18E836       |
+---------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

MD5

Syntax

```
MD5(str)
```

Description

Calculates an MD5 128-bit checksum for the string.

The return value is a 32-hex digit string, and a nonbinary string in the connection character set and collation, determined by the values of the character_set_connection and collation_connection system variables.

`NULL` is returned if the argument was `NULL`.

> **Warning:**
>
> **Don't use this function as an encryption function.**
>
> MD5 can be used as a checksum to verify data integrity against unintentional corruption. Historically it was widely used as a cryptographic hash function; however it has been found to suffer from extensive vulnerabilities.
>
> See https://en.wikipedia.org/wiki/MD5 for details.

Examples

```
SELECT MD5('testing');
+----------------------------------+
| MD5('testing')                   |
+----------------------------------+
| ae2b1fca515949e5d54fb22b8ed95575 |
+----------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

OLD_PASSWORD

Syntax

```
OLD_PASSWORD(str)
```

Description

`OLD_PASSWORD()` was added to MySQL when the implementation of PASSWORD() was changed to improve security. `OLD_PASSWORD()` returns the value of the old (pre-MySQL 4.1) implementation of `PASSWORD()` as a string, and is intended to permit you to reset passwords for any pre-4.1 clients that need to connect to a more recent MySQL server version, or any version of MariaDB, without locking them out.

The return value is a nonbinary string in the connection character set and collation, determined by the values of the character_set_connection and collation_connection system variables.

The return value is 16 bytes in length, or `NULL` if the argument was `NULL`.

See Also

- PASSWORD()
- MySQL manual on password hashing

This page is licensed: GPLv2, originally from fill_help_tables.sql

PASSWORD

Syntax

```
PASSWORD(str)
```

Description

The `PASSWORD()` function is used for hashing passwords for use in authentication by the MariaDB server. It is not intended for use in other applications.

Calculates and returns a hashed password string from the plaintext password *str*. Returns an empty string if the argument is `NULL`.

The return value is a nonbinary string in the connection character set and collation, determined by the values of the character_set_connection and collation_connection system variables.

This is the function that is used for hashing MariaDB passwords for storage in the Password column of the user table (see privileges), usually used with the SET PASSWORD statement. It is not intended for use in other applications.

The function takes into account the authentication plugin where applicable (a CREATE USER or SET PASSWORD statement). For example, when used in conjunction with a user authenticated by the ed25519 plugin, the statement will create a longer hash:

```
CREATE USER edtest@localhost IDENTIFIED VIA ed25519 USING PASSWORD('secret');

CREATE USER edtest2@localhost IDENTIFIED BY 'secret';

SELECT CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)) FROM mysql.global_priv
  WHERE user LIKE 'edtest%'\G
*************************** 1. row ***************************
CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)): edtest@localhost => {
...
    "plugin": "ed25519",
    "authentication_string": "ZIgUREUg5PVgQ6LskhXmO+eZLS0nC8be6HPjYWR4YJY",
...
}
*************************** 2. row ***************************
CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)): edtest2@localhost => {
...
    "plugin": "mysql_native_password",
    "authentication_string": "*14E65567ABDB5135D0CFD9A70B3032C179A49EE7",
...
}
```

The behavior of this function is affected by the value of the old_passwords system variable. If this is set to `1` (`0` is default), MariaDB reverts to using the mysql_old_password authentication plugin by default for newly created users and passwords.

Examples

```
SELECT PASSWORD('notagoodpwd');
+-------------------------------------------+
| PASSWORD('notagoodpwd')                   |
+-------------------------------------------+
| *3A70EE9FC6594F88CE9E959CD51C5A1C002DC937 |
+-------------------------------------------+
```

```
SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('newpass');
```

See Also

- Password Validation Plugins - permits the setting of basic criteria for passwords
- OLD_PASSWORD() - pre-MySQL 4.1 password function

This page is licensed: GPLv2, originally from fill_help_tables.sql

RANDOM_BYTES

> **Note:**
>
> `RANDOM_BYTES` is available from MariaDB 10.10.0.

The `RANDOM_BYTES` function generates a binary string of random bytes.

Syntax

```
RANDOM_BYTES(length)
```

Description

Given a *length* from 1 to 1024, generates a binary string of *length* consisting of random bytes generated by the SSL library's random number generator.

See the `RAND_bytes()` function documentation of your SSL library for information on the random number generator. In the case of OpenSSL, a cryptographically secure pseudo random generator (CSPRNG) is used.

Statements containing the RANDOM_BYTES function are unsafe for statement-based replication.

An error occurs if *length* is outside the range 1 to 1024.

This page is licensed: CC BY-SA / Gnu FDL

SHA1

Syntax

```
SHA1(str), SHA(str)
```

Description

Calculates an SHA-1 160-bit checksum for the string `str`, as described in RFC 3174 (Secure Hash Algorithm).

The value is returned as a string of 40 hex digits, or `NULL` if the argument was NULL. The return value is a nonbinary string in the connection character set and collation, determined by the values of the character_set_connection and collation_connection system variables.

Examples

```
SELECT SHA1('some boring text');
+------------------------------------------+
| SHA1('some boring text')                 |
+------------------------------------------+
| af969fc2085b1bb6d31e517d5c456def5cdd7093 |
+------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SHA2

Syntax

```
SHA2(str,hash_len)
```

Description

Given a string `str`, calculates an SHA-2 checksum, which is considered more cryptographically secure than its SHA-1 equivalent. The SHA-2 family includes SHA-224, SHA-256, SHA-384, and SHA-512, and the `hash_len` must correspond to one of these, i.e. 224, 256, 384 or 512. 0 is equivalent to 256.

The return value is a nonbinary string in the connection character set and collation, determined by the values of the character_set_connection and collation_connection system variables.

`NULL` is returned if the hash length is not valid, or the string `str` is `NULL`.

> **Warning:**
>
> `SHA2` only works if MariaDB is configured with TLS support.

Examples

```
SELECT SHA2('Maria',224);
+----------------------------------------------------------+
| SHA2('Maria',224)                                        |
+----------------------------------------------------------+
| 6cc67add32286412efcab9d0e1675a43a5c2ef3cec8879f81516ff83 |
+----------------------------------------------------------+

SELECT SHA2('Maria',256);
+------------------------------------------------------------------+
| SHA2('Maria',256)                                                |
+------------------------------------------------------------------+
| 9ff18ebe7449349f358e3af0b57cf7a032c1c6b2272cb2656ff85eb112232f16 |
+------------------------------------------------------------------+

SELECT SHA2('Maria',0);
+------------------------------------------------------------------+
| SHA2('Maria',0)                                                  |
+------------------------------------------------------------------+
| 9ff18ebe7449349f358e3af0b57cf7a032c1c6b2272cb2656ff85eb112232f16 |
+------------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

UNCOMPRESS

Syntax

```
UNCOMPRESS(string_to_uncompress)
```

Description

Uncompresses a string compressed by the COMPRESS() function. If the argument is not a compressed value, the result is `NULL`. This function requires MariaDB to have been compiled with a compression library such as zlib. Otherwise, the return value is always `NULL`. The have_compress server system variable indicates whether a compression library is present.

Examples

```
SELECT UNCOMPRESS(COMPRESS('a string'));
+----------------------------------+
| UNCOMPRESS(COMPRESS('a string')) |
+----------------------------------+
| a string                         |
+----------------------------------+

SELECT UNCOMPRESS('a string');
+------------------------+
| UNCOMPRESS('a string') |
+------------------------+
| NULL                   |
+------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

UNCOMPRESSED_LENGTH

Syntax

```
UNCOMPRESSED_LENGTH(compressed_string)
```

Description

Returns the length that the compressed string had before being compressed with COMPRESS().

`UNCOMPRESSED_LENGTH()` returns `NULL` or an incorrect result if the string is not compressed.

Returns `MYSQL_TYPE_LONG`, or int(10), if the result fits within 32-bits.

Examples

```
SELECT UNCOMPRESSED_LENGTH(COMPRESS(REPEAT('a',30)));
+-----------------------------------------------+
| UNCOMPRESSED_LENGTH(COMPRESS(REPEAT('a',30))) |
+-----------------------------------------------+
|                                            30 |
+-----------------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

Information Functions

- BENCHMARK
- BINLOG_GTID_POS

- CHARSET
- COERCIBILITY
- COLLATION
- CONNECTION_ID
- CURRENT_ROLE
- CURRENT_USER
- DATABASE
- DECODE_HISTOGRAM
- DEFAULT
- FOUND_ROWS
- LAST_INSERT_ID
- LAST_VALUE
- PROCEDURE ANALYSE
- ROW_COUNT
- ROWNUM
- SCHEMA
- SESSION_USER
- SYSTEM_USER
- USER
- VERSION

This page is licensed: CC BY-SA / Gnu FDL

BENCHMARK

Syntax

```
BENCHMARK(count,expr)
```

Description

The BENCHMARK() function executes the expression `expr` repeatedly `count` times. It may be used to time how quickly MariaDB processes the expression. The result value is always 0. The intended use is from within the mariadb client, which reports query execution times.

Examples

```
SELECT BENCHMARK(1000000,ENCODE('hello','goodbye'));
+--------------------------------------------+
| BENCHMARK(1000000,ENCODE('hello','goodbye')) |
+--------------------------------------------+
|                                          0 |
+--------------------------------------------+
1 row in set (0.21 sec)
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

BINLOG_GTID_POS

Syntax

```
BINLOG_GTID_POS(binlog_filename,binlog_offset)
```

Description

The BINLOG_GTID_POS() function takes as input an old-style binary log position in the form of a file name and a file offset. It looks up the position in the current binlog, and returns a string representation of the corresponding GTID position. If the position is not found in the current binlog, NULL is returned.

Examples

```
SELECT BINLOG_GTID_POS("master-bin.000001", 600);
```

See Also

- SHOW BINLOG EVENTS - Show events and their positions in the binary log

This page is licensed: CC BY-SA / Gnu FDL

CHARSET

Syntax

```
CHARSET(str)
```

Description

Returns the character set of the string argument. If `str` is not a string, it is considered as a binary string (so the function returns 'binary'). This applies to `NULL`, too. The return value is a string in the utf8 character set.

Examples

```
SELECT CHARSET('abc');
+----------------+
| CHARSET('abc') |
+----------------+
| latin1         |
+----------------+

SELECT CHARSET(CONVERT('abc' USING utf8));
+-----------------------------------+
| CHARSET(CONVERT('abc' USING utf8)) |
+-----------------------------------+
| utf8                              |
+-----------------------------------+

SELECT CHARSET(USER());
+-----------------+
| CHARSET(USER()) |
+-----------------+
| utf8            |
+-----------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

COERCIBILITY

Syntax

```
COERCIBILITY(str)
```

Description

Returns the collation coercibility value of the string argument. Coercibility defines what will be converted to what in case of collation conflict, with an expression with higher coercibility being converted to the collation of an expression with lower coercibility.

Coercibility	Description	Example
0	Explicit	Value using a COLLATE clause
1	No collation	Concatenated strings using different collations
2	Implicit	A string data type column value, CAST to a string data type
3	System constant	DATABASE(), USER() return value
4	Coercible	Literal string
5	Numeric	Numeric and temporal values
6	Ignorable	NULL or derived from NULL

Examples

```
SELECT COERCIBILITY(_latin1 'abc' COLLATE latin1_swedish_ci);
+---------------------------------------------+
| COERCIBILITY(_latin1 'abc' COLLATE latin1_swedish_ci) |
+---------------------------------------------+
|                                           0 |
+---------------------------------------------+

CREATE TABLE t (a VARCHAR(30) COLLATE uca1400_swedish_ai_ci, b VARCHAR(30) COLLATE uca1400_german2_ai_ci)
CHARSET utf8mb4;
INSERT INTO t VALUES ('abc', 'def'); /* a 2 coercibility */
SELECT COERCIBILITY(CONCAT(a, b)) FROM t;
+----------------------------+
| COERCIBILITY(CONCAT(a, b)) |
+----------------------------+
|                          1 |
+----------------------------+

SELECT COERCIBILITY(CAST(1 AS CHAR));
+-------------------------------+
| COERCIBILITY(CAST(1 AS CHAR)) |
+-------------------------------+
|                             2 |
+-------------------------------+

SELECT COERCIBILITY(USER());
+----------------------+
| COERCIBILITY(USER()) |
+----------------------+
|                    3 |
+----------------------+

SELECT COERCIBILITY('abc');
+---------------------+
| COERCIBILITY('abc') |
+---------------------+
|                   4 |
+---------------------+

SELECT COERCIBILITY(1);
+-----------------+
| COERCIBILITY(1) |
+-----------------+
|               5 |
+-----------------+

SELECT COERCIBILITY(NULL);
+--------------------+
| COERCIBILITY(NULL) |
+--------------------+
|                  6 |
+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

COLLATION

Syntax

```
COLLATION(str)
```

Description

Returns the collation of the string argument. If `str` is not a string, it is considered as a binary string (so the function returns 'binary'). This applies to `NULL`, too. The return value is a string in the utf8 character set.

See Character Sets and Collations.

Examples

```
SELECT COLLATION('abc');
+------------------+
| COLLATION('abc')  |
+------------------+
| latin1_swedish_ci |
+------------------+

SELECT COLLATION(_utf8'abc');
+-----------------------+
| COLLATION(_utf8'abc') |
+-----------------------+
| utf8_general_ci       |
+-----------------------+
```

See Also

- String literals
- CAST()
- CONVERT()

This page is licensed: GPLv2, originally from fill_help_tables.sql

CONNECTION_ID

Syntax

```
CONNECTION_ID()
```

Description

Returns the connection ID for the connection. Every connection (including events) has an ID that is unique among the set of currently connected clients.

Returns `MYSQL_TYPE_LONG`, or int(10).

Examples

```
SELECT CONNECTION_ID();
+-----------------+
| CONNECTION_ID() |
+-----------------+
|               3 |
+-----------------+
```

See Also

- SHOW PROCESSLIST
- INFORMATION_SCHEMA.PROCESSLIST

This page is licensed: GPLv2, originally from fill_help_tables.sql

CURRENT_ROLE

Syntax

```
CURRENT_ROLE, CURRENT_ROLE()
```

Description

Returns the current role name. The return value is a string in the utf8 character set.

If there is no current role, `NULL` is returned.

USER() returns the combination of user and host used to login. CURRENT_USER() returns the account used to determine current connection's privileges.

Statements using the `CURRENT_ROLE` function are not safe for statement-based replication.

Examples

```
SELECT CURRENT_ROLE;
+--------------+
| CURRENT_ROLE |
+--------------+
| NULL         |
+--------------+

SET ROLE staff;

SELECT CURRENT_ROLE;
+--------------+
| CURRENT_ROLE |
+--------------+
| staff        |
+--------------+
```

This page is licensed: CC BY-SA / Gnu FDL

CURRENT_USER

Syntax

```
CURRENT_USER, CURRENT_USER()
```

Description

Returns the user name and host name combination for the MariaDB account that the server used to authenticate the current client. This account determines your access privileges. The return value is a string in the utf8 character set.

The value of `CURRENT_USER()` can differ from the value of USER(). CURRENT_ROLE() returns the current active role.

Statements using the `CURRENT_USER` function are not safe for statement-based replication.

Examples

```
shell> mysql --user="anonymous"

SELECT USER(),CURRENT_USER();
+---------------------+----------------+
| USER()              | CURRENT_USER() |
+---------------------+----------------+
| anonymous@localhost | @localhost     |
+---------------------+----------------+
```

When calling `CURRENT_USER()` in a stored procedure, it returns the owner of the stored procedure, as defined with `DEFINER`.

See Also

- USER()
- SESSION_USER()
- CREATE PROCEDURE

This page is licensed: GPLv2, originally from fill_help_tables.sql

DATABASE

Syntax

```
DATABASE()
SCHEMA()
```

Description

Returns the default (current) database name as a string in the utf8 character set. If there is no default database, `DATABASE()` returns `NULL`. Within a stored routine, the default database is the database that the routine is associated with, which is not necessarily the same as the database that is the default in the calling context.

`SCHEMA()` is a synonym for `DATABASE()`.

To select a default database, the USE statement can be run. Another way to set the default database is specifying its name at mariadb command line client startup.

Examples

```
SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| NULL       |
+------------+

USE test;
Database changed

SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| test       |
+------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DECODE_HISTOGRAM

Syntax

```
DECODE_HISTOGRAM(hist_type,histogram)
```

Description

Returns a string of comma separated numeric values corresponding to a probability distribution represented by the histogram of type `hist_type` (`SINGLE_PREC_HB` or `DOUBLE_PREC_HB`). The `hist_type` and `histogram` would be commonly used from the mysql.column_stats table.

See Histogram Based Statistics for details.

Examples

```
CREATE TABLE origin (
  i INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  v INT UNSIGNED NOT NULL
);

INSERT INTO origin(v) VALUES
  (1),(2),(3),(4),(5),(10),(20),
  (30),(40),(50),(60),(70),(80),
  (90),(100),(200),(400),(800);

SET histogram_size=10,histogram_type=SINGLE_PREC_HB;

ANALYZE TABLE origin PERSISTENT FOR ALL;
+-------------+---------+----------+----------------------------------------+
| Table       | Op      | Msg_type | Msg_text                               |
+-------------+---------+----------+----------------------------------------+
| test.origin | analyze | status   | Engine-independent statistics collected |
| test.origin | analyze | status   | OK                                     |
+-------------+---------+----------+----------------------------------------+

SELECT db_name,table_name,column_name,hist_type,
  hex(histogram),decode_histogram(hist_type,histogram)
  FROM mysql.column_stats WHERE db_name='test' and table_name='origin';
+---------+------------+-------------+----------------+----------------------+--------------------------
---------------------------------------+
| db_name | table_name | column_name | hist_type      | hex(histogram)       |
decode_histogram(hist_type,histogram)                          |
+---------+------------+-------------+----------------+----------------------+--------------------------
---------------------------------------+
| test    | origin     | i           | SINGLE_PREC_HB | 0F2D3C5A7887A5C3D2F0 |
0.059,0.118,0.059,0.118,0.118,0.059,0.118,0.118,0.059,0.118,0.059 |
| test    | origin     | v           | SINGLE_PREC_HB | 000001060C0F161C1F7F |
0.000,0.000,0.004,0.020,0.024,0.012,0.027,0.024,0.012,0.376,0.502 |
+---------+------------+-------------+----------------+----------------------+--------------------------
---------------------------------------+

SET histogram_size=20,histogram_type=DOUBLE_PREC_HB;

ANALYZE TABLE origin PERSISTENT FOR ALL;
+-------------+---------+----------+----------------------------------------+
| Table       | Op      | Msg_type | Msg_text                               |
+-------------+---------+----------+----------------------------------------+
| test.origin | analyze | status   | Engine-independent statistics collected |
| test.origin | analyze | status   | OK                                     |
+-------------+---------+----------+----------------------------------------+

SELECT db_name,table_name,column_name,
  hist_type,hex(histogram),decode_histogram(hist_type,histogram)
  FROM mysql.column_stats WHERE db_name='test' and table_name='origin';
+---------+------------+-------------+----------------+----------------------------------------+-------
-------------------------------------------------------------------------------+
| db_name | table_name | column_name | hist_type      | hex(histogram)                         |
decode_histogram(hist_type,histogram)                                          |
+---------+------------+-------------+----------------+----------------------------------------+-------
-------------------------------------------------------------------------------+
| test    | origin     | i           | DOUBLE_PREC_HB | 0F0F2D2D3C3C5A5A78788787A5A5C3C3D2D2F0F0 |
0.05882,0.11765,0.05882,0.11765,0.11765,0.05882,0.11765,0.11765,0.05882,0.11765,0.05882 |
| test    | origin     | v           | DOUBLE_PREC_HB | 5200F600480116067E0CB30F1B16831CB81FD67F |
0.00125,0.00250,0.00125,0.01877,0.02502,0.01253,0.02502,0.02502,0.01253,0.37546,0.50063 |
```

This page is licensed: CC BY-SA / Gnu FDL

DEFAULT

Syntax

```
DEFAULT(col_name)
```

Description

Returns the default value for a table column. If the column has no default value (and is not `NULLABLE` - `NULLABLE` fields have a `NULL` default), an error is returned.

For integer columns using AUTO_INCREMENT, `0` is returned.

When using `DEFAULT` as a value to set in an INSERT or UPDATE statement, you can use the bare keyword `DEFAULT` without the parentheses and argument to refer to the column in context. You can only use `DEFAULT` as a bare keyword if you are using it alone without a surrounding expression or function.

Examples

Select only non-default values for a column:

```
SELECT i FROM t WHERE i != DEFAULT(i);
```

Update values to be one greater than the default value:

```
UPDATE t SET i = DEFAULT(i)+1 WHERE i < 100;
```

When referring to the default value exactly in `UPDATE` or `INSERT`, you can omit the argument:

```
INSERT INTO t (i) VALUES (DEFAULT);
UPDATE t SET i = DEFAULT WHERE i < 100;
```

```
CREATE OR REPLACE TABLE t (
  i INT NOT NULL AUTO_INCREMENT,
  j INT NOT NULL,
  k INT DEFAULT 3,
  l INT NOT NULL DEFAULT 4,
  m INT,
  PRIMARY KEY (i)
);

DESC t;
+-------+---------+------+-----+---------+----------------+
| Field | Type    | Null | Key | Default | Extra          |
+-------+---------+------+-----+---------+----------------+
| i     | int(11) | NO   | PRI | NULL    | auto_increment |
| j     | int(11) | NO   |     | NULL    |                |
| k     | int(11) | YES  |     | 3       |                |
| l     | int(11) | NO   |     | 4       |                |
| m     | int(11) | YES  |     | NULL    |                |
+-------+---------+------+-----+---------+----------------+

INSERT INTO t (j) VALUES (1);
INSERT INTO t (j,m) VALUES (2,2);
INSERT INTO t (j,l,m) VALUES (3,3,3);

SELECT * FROM t;
+---+---+------+---+------+
| i | j | k    | l | m    |
+---+---+------+---+------+
| 1 | 1 |    3 | 4 | NULL |
| 2 | 2 |    3 | 4 |    2 |
| 3 | 3 |    3 | 3 |    3 |
+---+---+------+---+------+

SELECT DEFAULT(i), DEFAULT(k), DEFAULT (l), DEFAULT(m) FROM t;
+------------+------------+-------------+------------+
| DEFAULT(i) | DEFAULT(k) | DEFAULT (l) | DEFAULT(m) |
+------------+------------+-------------+------------+
|          0 |          3 |           4 |       NULL |
|          0 |          3 |           4 |       NULL |
|          0 |          3 |           4 |       NULL |
+------------+------------+-------------+------------+

SELECT DEFAULT(i), DEFAULT(k), DEFAULT (l), DEFAULT(m), DEFAULT(j)  FROM t;
ERROR 1364 (HY000): Field 'j' doesn't have a default value

SELECT * FROM t WHERE i = DEFAULT(i);
Empty set (0.001 sec)

SELECT * FROM t WHERE j = DEFAULT(j);
ERROR 1364 (HY000): Field 'j' doesn't have a default value

SELECT * FROM t WHERE k = DEFAULT(k);
+---+---+------+---+------+
| i | j | k    | l | m    |
+---+---+------+---+------+
| 1 | 1 |    3 | 4 | NULL |
| 2 | 2 |    3 | 4 |    2 |
| 3 | 3 |    3 | 3 |    3 |
+---+---+------+---+------+

SELECT * FROM t WHERE l = DEFAULT(l);
+---+---+------+---+------+
| i | j | k    | l | m    |
+---+---+------+---+------+
| 1 | 1 |    3 | 4 | NULL |
| 2 | 2 |    3 | 4 |    2 |
+---+---+------+---+------+
```

```
SELECT * FROM t WHERE m = DEFAULT(m);
Empty set (0.001 sec)

SELECT * FROM t WHERE m <=> DEFAULT(m);
+---+---+------+---+------+
| i | j | k    | l | m    |
+---+---+------+---+------+
| 1 | 1 |    3 | 4 | NULL |
+---+---+------+---+------+
```

See Also

- CREATE TABLE DEFAULT Clause

This page is licensed: GPLv2, originally from fill_help_tables.sql

FOUND_ROWS

Syntax

```
FOUND_ROWS()
```

Description

A SELECT statement may include a LIMIT clause to restrict the number of rows the server returns to the client. In some cases, it is desirable to know how many rows the statement would have returned without the LIMIT, but without running the statement again. To obtain this row count, include an SQL_CALC_FOUND_ROWS option in the `SELECT` statement, and then invoke `FOUND_ROWS()` afterwards.

You can also use `FOUND_ROWS()` to obtain the number of rows returned by a `SELECT` which does not contain a `LIMIT` clause. In this case you don't need to use the `SQL_CALC_FOUND_ROWS` option. This can be useful for example in a stored procedure.

Also, this function works with some other statements which return a result set, including SHOW, DESC and HELP. For DELETE … RETURNING you should use ROW_COUNT(). It also works as a prepared statement, or after executing a prepared statement.

Statements which don't return any results don't affect `FOUND_ROWS()` - the previous value will still be returned.

Warning: When used after a CALL statement, this function returns the number of rows selected by the last query in the procedure, not by the whole procedure.

Statements using the `FOUND_ROWS()` function are not safe for statement-based replication.

Examples

```
SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: CSV
     Support: YES
     Comment: Stores tables as CSV files
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 2. row ***************************
      Engine: MRG_MyISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO

...

*************************** 8. row ***************************
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
8 rows in set (0.000 sec)

SELECT FOUND_ROWS();
+--------------+
| FOUND_ROWS() |
+--------------+
|           8 |
+--------------+
```

```
SELECT SQL_CALC_FOUND_ROWS * FROM tbl_name WHERE id > 100 LIMIT 10;
...
SELECT FOUND_ROWS();
+--------------+
| FOUND_ROWS() |
+--------------+
|           23 |
+--------------+
```

See Also

- ROW_COUNT()

This page is licensed: GPLv2, originally from fill_help_tables.sql

LAST_INSERT_ID

Syntax

```
LAST_INSERT_ID(), LAST_INSERT_ID(expr)
```

Description

`LAST_INSERT_ID()` (no arguments) returns the first automatically generated value successfully inserted for an AUTO_INCREMENT column as a result of the most recently executed `INSERT` statement. The value of `LAST_INSERT_ID()` remains unchanged if no rows are successfully inserted.

If one gives an argument to `LAST_INSERT_ID()`, then it will return the value of the expression and the next call to `LAST_INSERT_ID()` will return the same value. The value is also sent to the client and can be accessed by the mysql_insert_id function.

For example, after inserting a row that generates an `AUTO_INCREMENT` value, you can get the value like this:

```
SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                9 |
+------------------+
```

You can also use `LAST_INSERT_ID()` to delete the last inserted row:

```
DELETE FROM product WHERE id = LAST_INSERT_ID();
```

If no rows were successfully inserted, `LAST_INSERT_ID()` returns 0.

You can also use INSERT…RETURNING for this purpose.

The value of `LAST_INSERT_ID()` will be consistent across all versions if all rows in the INSERT or UPDATE statement were successful.

The currently executing statement does not affect the value of `LAST_INSERT_ID()`. Suppose that you generate an `AUTO_INCREMENT` value with one statement, and then refer to `LAST_INSERT_ID()` in a multiple-row `INSERT` statement that inserts rows into a table with its own `AUTO_INCREMENT` column. The value of `LAST_INSERT_ID()` will remain stable in the second statement; its value for the second and later rows is not affected by the earlier row insertions. (However, if you mix references to `LAST_INSERT_ID()` and `LAST_INSERT_ID( expr )`, the effect is undefined.)

If the previous statement returned an error, the value of `LAST_INSERT_ID()` is undefined. For transactional tables, if the statement is rolled back due to an error, the value of `LAST_INSERT_ID()` is left undefined. For manual ROLLBACK, the value of `LAST_INSERT_ID()` is not restored to that before the transaction; it remains as it was at the point of the `ROLLBACK`.

Within the body of a stored routine (procedure or function) or a trigger, the value of `LAST_INSERT_ID()` changes the same way as for statements executed outside the body of these kinds of objects. The effect of a stored routine or trigger upon the value of `LAST_INSERT_ID()` that is seen by following statements depends on the kind of routine:

- If a stored procedure executes statements that change the value of `LAST_INSERT_ID()`, the new value will be seen by statements that follow the procedure call.
- For stored functions and triggers that change the value, the value is restored when the function or trigger ends, so following statements will not see a changed value.

Examples

```
CREATE TABLE t (
  id INTEGER UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  f VARCHAR(1))
ENGINE = InnoDB;

INSERT INTO t(f) VALUES('a');

SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                1 |
+------------------+

INSERT INTO t(f) VALUES('b');

INSERT INTO t(f) VALUES('c');

SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                3 |
+------------------+

INSERT INTO t(f) VALUES('d'),('e');

SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                4 |
+------------------+

SELECT * FROM t;
+----+------+
| id | f    |
+----+------+
|  1 | a    |
|  2 | b    |
|  3 | c    |
|  4 | d    |
|  5 | e    |
+----+------+

SELECT LAST_INSERT_ID(12);
+--------------------+
| LAST_INSERT_ID(12) |
+--------------------+
|                 12 |
+--------------------+

SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|               12 |
+------------------+

INSERT INTO t(f) VALUES('f');

SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                6 |
+------------------+
```

```
SELECT * FROM t;
+----+------+
| id | f    |
+----+------+
|  1 | a    |
|  2 | b    |
|  3 | c    |
|  4 | d    |
|  5 | e    |
|  6 | f    |
+----+------+

SELECT LAST_INSERT_ID(12);
+--------------------+
| LAST_INSERT_ID(12) |
+--------------------+
|                 12 |
+--------------------+

INSERT INTO t(f) VALUES('g');

SELECT * FROM t;
+----+------+
| id | f    |
+----+------+
|  1 | a    |
|  2 | b    |
|  3 | c    |
|  4 | d    |
|  5 | e    |
|  6 | f    |
|  7 | g    |
+----+------+
```

See Also

- mysql_insert_id
- AUTO_INCREMENT
- AUTO_INCREMENT handling in InnoDB
- Sequences - an alternative to auto_increment
- INSERT…RETURNING

This page is licensed: GPLv2, originally from fill_help_tables.sql

LAST_VALUE

Syntax

```
LAST_VALUE(expr,[expr,...])
```

```
LAST_VALUE(expr) OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```

Description

`LAST_VALUE()` evaluates all expressions and returns the last. This is useful together with setting user variables to a value with @var:=expr, for example when you want to get data of rows updated/deleted without having to do two queries against the table.

`LAST_VALUE` can be used as a window function.

Returns `NULL` if no last value exists.

Examples

```
CREATE TABLE t1 (a int, b int);
INSERT INTO t1 VALUES(1,10),(2,20);
DELETE FROM t1 WHERE a=1 AND last_value(@a:=a,@b:=b,1);
SELECT @a,@b;
+------+------+
| @a   | @b   |
+------+------+
|    1 |   10 |
+------+------+
```

As a window function:

```
CREATE TABLE t1 (
  pk int primary key,
  a int,
  b int,
  c char(10),
  d decimal(10, 3),
  e real
);

INSERT INTO t1 VALUES
( 1, 0, 1,    'one',    0.1,  0.001),
( 2, 0, 2,    'two',    0.2,  0.002),
( 3, 0, 3,    'three',  0.3,  0.003),
( 4, 1, 2,    'three',  0.4,  0.004),
( 5, 1, 1,    'two',    0.5,  0.005),
( 6, 1, 1,    'one',    0.6,  0.006),
( 7, 2, NULL, 'n_one',  0.5,  0.007),
( 8, 2, 1,    'n_two',  NULL, 0.008),
( 9, 2, 2,    NULL,     0.7,  0.009),
(10, 2, 0,    'n_four', 0.8,  0.010),
(11, 2, 10,   NULL,     0.9,  NULL);

SELECT pk, FIRST_VALUE(pk) OVER (ORDER BY pk) AS first_asc,
           LAST_VALUE(pk) OVER (ORDER BY pk) AS last_asc,
           FIRST_VALUE(pk) OVER (ORDER BY pk DESC) AS first_desc,
           LAST_VALUE(pk) OVER (ORDER BY pk DESC) AS last_desc
FROM t1
ORDER BY pk DESC;

+----+-----------+----------+------------+-----------+
| pk | first_asc | last_asc | first_desc | last_desc |
+----+-----------+----------+------------+-----------+
| 11 |         1 |       11 |         11 |        11 |
| 10 |         1 |       10 |         11 |        10 |
|  9 |         1 |        9 |         11 |         9 |
|  8 |         1 |        8 |         11 |         8 |
|  7 |         1 |        7 |         11 |         7 |
|  6 |         1 |        6 |         11 |         6 |
|  5 |         1 |        5 |         11 |         5 |
|  4 |         1 |        4 |         11 |         4 |
|  3 |         1 |        3 |         11 |         3 |
|  2 |         1 |        2 |         11 |         2 |
|  1 |         1 |        1 |         11 |         1 |
+----+-----------+----------+------------+-----------+
```

```
CREATE OR REPLACE TABLE t1 (i int);
INSERT INTO t1 VALUES (1),(2),(3),(4),(5),(6),(7),(8),(9),(10);

SELECT i,
  FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN CURRENT ROW and 1 FOLLOWING) AS f_1f,
  LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN CURRENT ROW and 1 FOLLOWING) AS l_1f,
  FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS f_1p1f,
  LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS l_1p1f,
  FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 2 PRECEDING AND 1 PRECEDING) AS f_2p1p,
  LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 2 PRECEDING AND 1 PRECEDING) AS l_2p1p,
  FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 FOLLOWING AND 2 FOLLOWING) AS f_1f2f,
  LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 FOLLOWING AND 2 FOLLOWING) AS l_1f2f
FROM t1;

+------+------+------+--------+--------+--------+--------+--------+--------+
| i    | f_1f | l_1f | f_1p1f | l_1p1f | f_2p1p | l_2p1p | f_1f2f | l_1f2f |
+------+------+------+--------+--------+--------+--------+--------+--------+
|    1 |    1 |    2 |      1 |      2 |   NULL |   NULL |      2 |      3 |
|    2 |    2 |    3 |      1 |      3 |      1 |      1 |      3 |      4 |
|    3 |    3 |    4 |      2 |      4 |      1 |      2 |      4 |      5 |
|    4 |    4 |    5 |      3 |      5 |      2 |      3 |      5 |      6 |
|    5 |    5 |    6 |      4 |      6 |      3 |      4 |      6 |      7 |
|    6 |    6 |    7 |      5 |      7 |      4 |      5 |      7 |      8 |
|    7 |    7 |    8 |      6 |      8 |      5 |      6 |      8 |      9 |
|    8 |    8 |    9 |      7 |      9 |      6 |      7 |      9 |     10 |
|    9 |    9 |   10 |      8 |     10 |      7 |      8 |     10 |     10 |
|   10 |   10 |   10 |      9 |     10 |      8 |      9 |   NULL |   NULL |
+------+------+------+--------+--------+--------+--------+--------+--------+
```

See Also

- Setting a variable to a value

This page is licensed: CC BY-SA / Gnu FDL

PROCEDURE ANALYSE

Syntax

```
ANALYSE([max_elements[,max_memory]])
```

Description

This procedure is defined in the `sql/sql_analyse.cc` file. It examines the result from a query and returns an analysis of the results that suggests optimal data types for each column. To obtain this analysis, append `PROCEDURE ANALYSE` to the end of a SELECT statement:

```
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max_elements,[max_memory]])
```

For example:

```
SELECT col1, col2 FROM table1 PROCEDURE ANALYSE(10, 2000);
```

The results show some statistics for the values returned by the query, and propose an optimal data type for the columns. This can be helpful for checking your existing tables, or after importing new data. You may need to try different settings for the arguments so that `PROCEDURE ANALYSE()` does not suggest the `ENUM` data type when it is not appropriate.

The arguments are optional and are used as follows:

- max_elements (default 256) is the maximum number of distinct values that analyse notices per column. This is used by analyse to check whether the optimal data type should be of type `ENUM`; if there are more than max_elements distinct values, then `ENUM` is not a suggested type.
- max_memory (default 8192) is the maximum amount of memory that analyse should allocate per column while trying to find all distinct values.

See Also

- PROCEDURE
- SELECT

This page is licensed: GPLv2, originally from fill_help_tables.sql

ROW_COUNT

Syntax

```
ROW_COUNT()
```

Description

`ROW_COUNT()` returns the number of rows updated, inserted or deleted by the preceding statement. This is the same as the row count that the mariadb client displays and the value from the mysql_affected_rows() C API function.

Generally:

- For statements which return a result set (such as SELECT, SHOW, DESC or HELP), returns -1, even when the result set is empty. This is also true for administrative statements, such as OPTIMIZE.
- For DML statements other than SELECT and for ALTER TABLE, returns the number of affected rows.
- For DDL statements (including TRUNCATE) and for other statements which don't return any result set (such as USE, DO, SIGNAL or DEALLOCATE PREPARE), returns 0.

For UPDATE, affected rows is by default the number of rows that were actually changed. If the CLIENT_FOUND_ROWS flag to mysql_real_connect() is specified when connecting to mariadbd, affected rows is instead the number of rows matched by the WHERE clause.

For REPLACE, deleted rows are also counted. So, if REPLACE deletes a row and adds a new row, `ROW_COUNT()` returns 2.

For INSERT … ON DUPLICATE KEY, values returned are as follows:

- 0: an existing row is set to its current values, and the `CLIENT_FOUND_ROWS` is not set.
- 1: the values are inserted as a new row, or an existing row is set to its current values, and the `CLIENT_FOUND_ROWS` is set.
- 2: an existing row is updated with new values.

`ROW_COUNT()` does not take into account rows that are not directly deleted/updated by the last statement. This means that rows deleted by foreign keys or triggers are not counted.

Warning: You can use `ROW_COUNT()` with prepared statements, but you need to call it after `EXECUTE`, not after DEALLOCATE PREPARE, because the row count for allocate prepare is always 0.

Warning: When used after a CALL statement, this function returns the number of rows affected by the last statement in the procedure, not by the whole procedure.

Warning: After INSERT DELAYED, `ROW_COUNT()` returns the number of the rows you tried to insert, not the number of the successful writes.

This information can also be found in the diagnostics area.

Statements using the `ROW_COUNT()` function are not safe for statement-based replication.

Examples

```
CREATE TABLE t (A INT);

INSERT INTO t VALUES(1),(2),(3);

SELECT ROW_COUNT();
+-------------+
| ROW_COUNT() |
+-------------+
|           3 |
+-------------+

DELETE FROM t WHERE A IN(1,2);

SELECT ROW_COUNT();
+-------------+
| ROW_COUNT() |
+-------------+
|           2 |
+-------------+
```

Example with prepared statements:

```
SET @q = 'INSERT INTO t VALUES(1),(2),(3);';

PREPARE stmt FROM @q;

EXECUTE stmt;
Query OK, 3 rows affected (0.39 sec)
Records: 3  Duplicates: 0  Warnings: 0

SELECT ROW_COUNT();
+-------------+
| ROW_COUNT() |
+-------------+
|           3 |
+-------------+
```

See Also

- FOUND_ROWS()

This page is licensed: GPLv2, originally from fill_help_tables.sql

ROWNUM

> **Note:**
>
> `ROWNUM()` is available from MariaDB 10.6.1.

Syntax

```
ROWNUM()
```

In Oracle mode one can just use `ROWNUM`, without the parentheses.

Description

`ROWNUM()` returns the current number of accepted rows in the current context. It main purpose is to emulate the ROWNUM pseudo column in Oracle. For MariaDB native applications, we recommend the usage of LIMIT, as it is easier to use and gives more predictable results than the usage of `ROWNUM()`.

The main difference between using `LIMIT` and `ROWNUM()` to limit the rows in the result is that `LIMIT` works on the result set while `ROWNUM` works on the number of accepted rows (before any `ORDER` or `GROUP BY` clauses).

The following queries will return the same results:

```
SELECT * FROM t1 LIMIT 10;
SELECT * FROM t1 WHERE ROWNUM() <= 10;
```

While the following may return different results based on in which orders the rows are found:

```
SELECT * FROM t1 ORDER BY a LIMIT 10;
SELECT * FROM t1 ORDER BY a WHERE ROWNUM() <= 10;
```

The recommended way to use `ROWNUM` to limit the number of returned rows and get predictable results is to have the query in a subquery and test for `ROWNUM()` in the outer query:

```
SELECT * FROM (SELECT * FROM t1 ORDER BY a) WHERE ROWNUM() <= 10;
```

`ROWNUM()` can be used in the following contexts:

- SELECT
- INSERT
- UPDATE
- DELETE
- LOAD DATA INFILE

Used in other contexts, `ROWNUM()` will return 0.

Examples

```
INSERT INTO t1 VALUES (1,ROWNUM()),(2,ROWNUM()),(3,ROWNUM());

INSERT INTO t1 VALUES (1),(2) RETURNING a, ROWNUM();

UPDATE t1 SET row_num_column=ROWNUM();

DELETE FROM t1 WHERE a < 10 AND ROWNUM() < 2;

LOAD DATA INFILE 'filename' INTO TABLE t1 fields terminated BY ','
  lines terminated BY "\r\n" (a,b) SET c=ROWNUM();
```

Optimizations

In many cases where `ROWNUM()` is used, MariaDB will use the same optimizations it uses with LIMIT.

`LIMIT` optimization is possible when using `ROWNUM` in the following manner:

- When one is in a top level `WHERE` clause comparing `ROWNUM()` with a numerical constant using any of the following expressions:
  - `ROWNUM()` < number.
  - `ROWNUM()` <= number.
  - `ROWNUM()` = 1 `ROWNUM()` can be also be the right argument to the comparison function.

In the above cases, `LIMIT` optimization can be done in the following cases:

- For the current subquery when the `ROWNUM` comparison is done on the top level:

```
SELECT * FROM t1 WHERE ROWNUM() <= 2 AND t1.a > 0
```

- For an inner subquery, when the upper level has only a `ROWNUM()` comparison in the `WHERE` clause:

```
SELECT * FROM (SELECT * FROM t1) AS t WHERE ROWNUM() <= 2
```

Other Changes Related to ROWNUM

When `ROWNUM()` is used anywhere in a query, the optimization to ignore `ORDER BY` in subqueries are disabled.

This was done to get the following common Oracle query to work as expected:

```
SELECT * FROM (SELECT * FROM t1 ORDER BY a DESC) AS t WHERE rownum() <= 2;
```

By default, MariaDB ignores any `ORDER BY` in subqueries both because the SQL standard defines results sets in subqueries to be un-ordered and because of performance reasons (especially when using views in subqueries). See MDEV-3926 "Wrong result with `GROUP BY ... WITH ROLLUP`" for a discussion of this topic.

Other Considerations

While MariaDB tries to emulate Oracle's usage of `ROWNUM()` as closely as possible, there are cases where the result is different:

- When the optimizer finds rows in a different order (because of different storage methods or optimization). This may also happen in Oracle if one adds or deletes an index, in which case the rows may be found in a different order.

Note that usage of `ROWNUM()` in functions or stored procedures will use their own context, not the caller's context.

See Also

- MDEV-24089 support Oracle syntax: rownum
- LIMIT clause

This page is licensed: CC BY-SA / Gnu FDL

SCHEMA

Syntax

```
SCHEMA()
```

Description

This function is a synonym for DATABASE().

This page is licensed: GPLv2, originally from fill_help_tables.sql

SESSION_USER

Syntax

```
SESSION_USER()
```

Description

CURRENT

Shows the value of CURRENT_USER() when the session was created, that is, it shows a `user@host` pair from the mysql.global_priv table, like `CURRENT_USER()`, but unlike `CURRENT_USER()` it will not change inside stored routines and views. This is SQL Standard behavior for the `SESSION_USER` function.

< 11.7

`SESSION_USER()` is a synonym for USER().

Backward-compatible behavior can be restored by setting old mode to SESSION_USER_IS_USER.

This page is licensed: GPLv2, originally from fill_help_tables.sql

SYSTEM_USER

Syntax

```
SYSTEM_USER()
```

Description

`SYSTEM_USER()` is a synonym for USER().

This page is licensed: GPLv2, originally from fill_help_tables.sql

USER

Syntax

```
USER()
```

Description

Returns the current MariaDB user name and host name, given when authenticating to MariaDB, as a string in the utf8 character set.

> **Warning:**
>
> The value of `USER()` may differ from the value of CURRENT_USER(), which is the user used to authenticate the current client. CURRENT_ROLE() returns the currently active role.

CURRENT

`SYSTEM_USER()` is a synonym for `USER()`.

< 11.7

`SYSTEM_USER()` and SESSION_USER are synonyms for `USER()`.

Statements using the `USER()` function or one of its synonyms are not safe for statement level replication.

Examples

```
shell> mysql --user="anonymous"

SELECT USER(),CURRENT_USER();
+---------------------+----------------+
| USER()              | CURRENT_USER() |
+---------------------+----------------+
| anonymous@localhost | @localhost     |
+---------------------+----------------+
```

To select only the IP address, use SUBSTRING_INDEX(),

```
SELECT SUBSTRING_INDEX(USER(), '@', -1);
+---------------------------------+
| SUBSTRING_INDEX(USER(), '@', -1) |
+---------------------------------+
| 192.168.0.101                   |
+---------------------------------+
```

See Also

- CURRENT_USER()

This page is licensed: GPLv2, originally from fill_help_tables.sql

VERSION

Syntax

```
VERSION()
```

Description

Returns a string that indicates the MariaDB server version. The string uses the utf8 character set.

Examples

```
SELECT VERSION();
+----------------+
| VERSION()      |
+----------------+
| 10.4.7-MariaDB |
+----------------+
```

The `VERSION()` string may have one or more of the following suffixes:

Suffix	Description
-embedded	The server is an embedded server (libmariadbd).
-log	General logging, slow logging or binary (replication) logging is enabled.
-debug	The server is compiled for debugging.
-valgrind	The server is compiled to be instrumented with valgrind.

Changing the Version String

Some old legacy code may break because they are parsing the `VERSION` string and expecting a MySQL string or a simple version string like Joomla til API17, see MDEV-7780.

You can fool these applications by setting the version string from the command line or the `my.cnf` files with –version=….

This page is licensed: GPLv2, originally from fill_help_tables.sql

Miscellaneous Functions

- DBMS_OUTPUT
- FORMAT_BYTES
- GET_LOCK
- INET6_ATON
- INET6_NTOA
- INET_ATON
- INET_NTOA
- IS_FREE_LOCK
- IS_IPV4
- IS_IPV4_COMPAT
- IS_IPV4_MAPPED
- IS_IPV6
- IS_USED_LOCK
- MASTER_GTID_WAIT
- MASTER_POS_WAIT
- NAME_CONST
- OLD_VALUE
- RELEASE_ALL_LOCKS
- RELEASE_LOCK
- SLEEP
- SYS_GUID
- UUID
- UUID_SHORT
- UUID_v4
- UUID_v7
- VALUES / VALUE

This page is licensed: CC BY-SA / Gnu FDL

FORMAT_BYTES

> **Note:**
>
> `FORMAT_BYTES` is available from MariaDB 11.8.

Syntax

```
FORMAT_BYTES(double)
```

Description

Given a byte count, returns a string consisting of a value and the units in a human-readable format. The units will be in bytes, KiB (kibibytes), MiB (mebibytes), GiB (gibibytes), TiB (tebibytes), PiB (pebibytes) or EiB (exbibytes).

The binary prefixes (kibi, mebi, gibi, tebi, pebi and exbi) were created in December 1998 by the International Electrotechnical Commission to avoid possible ambiguity, as the widely-used prefixes kilo, mega, giga, tera, peta and exa can be used to refer to both the power-of-10 decimal system multipliers and the power-of-two binary system multipliers.

This function is similar to the Sys Schema format_bytes function, except that function does not display exbibytes.

Examples

```
SELECT FORMAT_BYTES(1000)FORMAT_BYTES(1024);
+-------------------+-------------------+
| FORMAT_BYTES(1000) | FORMAT_BYTES(1024) |
+-------------------+-------------------+
| 1000 bytes         | 1.00 KiB           |
+-------------------+-------------------+

SELECT FORMAT_BYTES(1000000),FORMAT_BYTES(1048576);
+----------------------+----------------------+
| FORMAT_BYTES(1000000) | FORMAT_BYTES(1048576) |
+----------------------+----------------------+
| 976.56 KiB            | 1.00 MiB              |
+----------------------+----------------------+

SELECT FORMAT_BYTES(1000000000),FORMAT_BYTES(1073741874);
+--------------------------+--------------------------+
| FORMAT_BYTES(1000000000) | FORMAT_BYTES(1073741874) |
+--------------------------+--------------------------+
| 953.67 MiB               | 1.00 GiB                    |
+--------------------------+--------------------------+

SELECT FORMAT_BYTES(1000000000000),FORMAT_BYTES(1099511627776);
+-----------------------------+-----------------------------+
| FORMAT_BYTES(1000000000000) | FORMAT_BYTES(1099511627776) |
+-----------------------------+-----------------------------+
| 931.32 GiB                  | 1.00 TiB                    |
+-----------------------------+-----------------------------+

SELECT FORMAT_BYTES(1000000000000000),FORMAT_BYTES(1125899906842624);
+--------------------------------+--------------------------------+
| FORMAT_BYTES(1000000000000000) | FORMAT_BYTES(1125899906842624) |
+--------------------------------+--------------------------------+
| 909.49 TiB                     | 1.00 PiB                       |
+--------------------------------+--------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

GET_LOCK

Syntax

```
GET_LOCK(str,timeout)
```

Description

Tries to obtain a lock with a name given by the string `str`, using a timeout of `timeout` seconds. Returns `1` if the lock was obtained successfully, `0` if the attempt timed out (for example, because another client has previously locked the name), or `NULL` if an error occurred (such as running out of memory or the thread was killed with mariadb-admin kill).

A lock is released with RELEASE_LOCK(), when the connection terminates (either normally or abnormally). A connection can hold multiple locks at the same time, so a lock that is no longer needed needs to be explicitly released.

The IS_FREE_LOCK function returns whether a specified lock is free or not, and the IS_USED_LOCK whether the function is in use or not.

Locks obtained with `GET_LOCK()` do not interact with transactions. That is, committing a transaction does not release any such locks obtained during the transaction.

It is also possible to recursively set the same lock. If a lock with the same name is set `n` times, it needs to be released `n` times as well.

`str` is case insensitive for `GET_LOCK()` and related functions. If `str` is an empty string or `NULL`, `GET_LOCK()` returns `NULL` and does nothing. `timeout` supports microseconds.

If the metadata_lock_info plugin is installed, locks acquired with this function are visible in the Information Schema METADATA_LOCK_INFO table.

This function can be used to implement application locks or to simulate record locks. Names are locked on a server-wide basis. If a name has been locked by one client, `GET_LOCK()` blocks any request by another client for a lock with the same name. This allows clients that agree on a given lock name to use the name to perform cooperative advisory locking. But be aware that it also allows a client that is not among the set of cooperating clients to lock a name, either inadvertently or deliberately, and thus prevent any of the cooperating clients from locking that name. One way to reduce the likelihood of this is to use lock names that are database-specific or application-specific. For example, use lock names of the form `db_name.str` or `app_name.str`.

Statements using the `GET_LOCK` function are not safe for statement-based replication.

The patch to permit multiple locks was contributed by Konstantin “Kostja” Osipov (MDEV-3917).

Examples

```
SELECT GET_LOCK('lock1',10);
+----------------------+
| GET_LOCK('lock1',10) |
+----------------------+
|                    1 |
+----------------------+

SELECT IS_FREE_LOCK('lock1'), IS_USED_LOCK('lock1');
+-----------------------+-----------------------+
| IS_FREE_LOCK('lock1') | IS_USED_LOCK('lock1') |
+-----------------------+-----------------------+
|                     0 |                    46 |
+-----------------------+-----------------------+

SELECT IS_FREE_LOCK('lock2'), IS_USED_LOCK('lock2');
+-----------------------+-----------------------+
| IS_FREE_LOCK('lock2') | IS_USED_LOCK('lock2') |
+-----------------------+-----------------------+
|                     1 |                  NULL |
+-----------------------+-----------------------+
```

Multiple locks can be held:

```
SELECT GET_LOCK('lock2',10);
+---------------------+
| GET_LOCK('lock2',10) |
+---------------------+
|                   1 |
+---------------------+

SELECT IS_FREE_LOCK('lock1'), IS_FREE_LOCK('lock2');
+----------------------+----------------------+
| IS_FREE_LOCK('lock1') | IS_FREE_LOCK('lock2') |
+----------------------+----------------------+
|                    0 |                    0 |
+----------------------+----------------------+

SELECT RELEASE_LOCK('lock1'), RELEASE_LOCK('lock2');
+----------------------+----------------------+
| RELEASE_LOCK('lock1') | RELEASE_LOCK('lock2') |
+----------------------+----------------------+
|                    1 |                    1 |
+----------------------+----------------------+
```

It is possible to hold the same lock recursively. This example is viewed using the metadata_lock_info plugin:

```
SELECT GET_LOCK('lock3',10);
+---------------------+
| GET_LOCK('lock3',10) |
+---------------------+
|                   1 |
+---------------------+

SELECT GET_LOCK('lock3',10);
+---------------------+
| GET_LOCK('lock3',10) |
+---------------------+
|                   1 |
+---------------------+

SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
+-----------+--------------------+---------------+-----------+--------------+------------+
| THREAD_ID | LOCK_MODE          | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----------+--------------------+---------------+-----------+--------------+------------+
|        46 | MDL_SHARED_NO_WRITE | NULL          | User lock | lock3        |            |
+-----------+--------------------+---------------+-----------+--------------+------------+

SELECT RELEASE_LOCK('lock3');
+----------------------+
| RELEASE_LOCK('lock3') |
+----------------------+
|                    1 |
+----------------------+

SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
+-----------+--------------------+---------------+-----------+--------------+------------+
| THREAD_ID | LOCK_MODE          | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----------+--------------------+---------------+-----------+--------------+------------+
|        46 | MDL_SHARED_NO_WRITE | NULL          | User lock | lock3        |            |
+-----------+--------------------+---------------+-----------+--------------+------------+

SELECT RELEASE_LOCK('lock3');
+----------------------+
| RELEASE_LOCK('lock3') |
+----------------------+
|                    1 |
+----------------------+

SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
Empty set (0.000 sec)
```

Timeout example: Connection 1:

```
SELECT GET_LOCK('lock4',10);
+----------------------+
| GET_LOCK('lock4',10) |
+----------------------+
|                    1 |
+----------------------+
```

Connection 2:

```
SELECT GET_LOCK('lock4',10);
```

After 10 seconds…

```
+----------------------+
| GET_LOCK('lock4',10) |
+----------------------+
|                    0 |
+----------------------+
```

Deadlocks are automatically detected and resolved. Connection 1:

```
SELECT GET_LOCK('lock5',10);
+----------------------+
| GET_LOCK('lock5',10) |
+----------------------+
|                    1 |
+----------------------+
```

Connection 2:

```
SELECT GET_LOCK('lock6',10);
+----------------------+
| GET_LOCK('lock6',10) |
+----------------------+
|                    1 |
+----------------------+
```

Connection 1:

```
SELECT GET_LOCK('lock6',10);
+----------------------+
| GET_LOCK('lock6',10) |
+----------------------+
|                    0 |
+----------------------+
```

Connection 2:

```
SELECT GET_LOCK('lock5',10);
ERROR 1213 (40001): Deadlock found when trying to get lock; try restarting transaction
```

See Also

- RELEASE_LOCK
- IS_FREE_LOCK
- IS_USED_LOCK
- RELEASE_ALL_LOCKS

This page is licensed: GPLv2, originally from fill_help_tables.sql

INET6_ATON

Syntax

```
INET6_ATON(expr)
```

Description

Given an IPv6 or IPv4 network address as a string, returns a binary string that represents the numeric value of the address.

No trailing zone ID's or trailing network masks are permitted. For IPv4 addresses, or IPv6 addresses with IPv4 address parts, no classful addresses or trailing port numbers are permitted and octal numbers are not supported.

The returned binary string will be VARBINARY(16) or VARBINARY(4) for IPv6 and IPv4 addresses respectively.

Returns `NULL` if the argument is not understood.

MariaDB starting with 10.5.0

> **CURRENT**
>
> `INET6_ATON` can take INET6 as an argument.
>
> **< 10.5**
>
> `INET6_ATON` **cannot** take INET6 as an argument.

Examples

```
SELECT HEX(INET6_ATON('10.0.1.1'));
+----------------------------+
| HEX(INET6_ATON('10.0.1.1')) |
+----------------------------+
| 0A000101                   |
+----------------------------+

SELECT HEX(INET6_ATON('48f3::d432:1431:ba23:846f'));
+-------------------------------------------+
| HEX(INET6_ATON('48f3::d432:1431:ba23:846f')) |
+-------------------------------------------+
| 48F3000000000000D4321431BA23846F             |
+-------------------------------------------+
```

See Also

- INET6_NTOA()
- INET_ATON()
- INET6 Data Type

This page is licensed: CC BY-SA / Gnu FDL

INET6_NTOA

Syntax

```
INET6_NTOA(expr)
```

Description

Given an IPv6 or IPv4 network address as a numeric binary string, returns the address as a nonbinary string in the connection character set.

The return string is lowercase, and is platform independent, since it does not use functions specific to the operating system. It has a maximum length of 39 characters.

Returns `NULL` if the argument is not understood.

Examples

```
SELECT INET6_NTOA(UNHEX('0A000101'));
+------------------------------+
| INET6_NTOA(UNHEX('0A000101')) |
+------------------------------+
| 10.0.1.1                     |
+------------------------------+

SELECT INET6_NTOA(UNHEX('48F3000000000000D4321431BA23846F'));
+-------------------------------------------------------+
| INET6_NTOA(UNHEX('48F3000000000000D4321431BA23846F')) |
+-------------------------------------------------------+
| 48f3::d432:1431:ba23:846f                             |
+-------------------------------------------------------+
```

See Also

- INET6_ATON()
- INET_NTOA()

This page is licensed: CC BY-SA / Gnu FDL

INET_ATON

Syntax

```
INET_ATON(expr)
```

Description

Given the dotted-quad representation of an IPv4 network address as a string, returns an integer that represents the numeric value of the address. Addresses may be 4- or 8-byte addresses.

Returns `NULL` if the argument is not understood.

Examples

```
SELECT INET_ATON('192.168.1.1');
+--------------------------+
| INET_ATON('192.168.1.1') |
+--------------------------+
|               3232235777 |
+--------------------------+
```

This is calculated as follows: 192 x 2563 + 168 x 256 2 + 1 x 256 + 1.

See Also

- INET6_ATON()
- INET_NTOA()

This page is licensed: GPLv2, originally from fill_help_tables.sql

INET_NTOA

Syntax

```
INET_NTOA(expr)
```

Description

Given a numeric IPv4 network address in network byte order (4 or 8 byte), returns the dotted-quad representation of the address as a string.

Examples

```
SELECT INET_NTOA(3232235777);
+-----------------------+
| INET_NTOA(3232235777) |
+-----------------------+
| 192.168.1.1           |
+-----------------------+
```

192.168.1.1 corresponds to 3232235777 since 192 x 2563 + 168 x 256 2 + 1 x 256 + 1 = 3232235777.

See Also

- INET6_NTOA()
- INET_ATON()

This page is licensed: GPLv2, originally from fill_help_tables.sql

IS_FREE_LOCK

Syntax

```
IS_FREE_LOCK(str)
```

Description

Checks whether the lock named `str` is free to use (that is, not locked). Returns `1` if the lock is free (no one is using the lock), `0` if the lock is in use, and `NULL` if an error occurs (such as an incorrect argument, like an empty string or `NULL`). `str` is case insensitive.

If the metadata_lock_info plugin is installed, the Information Schema metadata_lock_info table contains information about locks of this kind (as well as metadata locks).

Statements using the `IS_FREE_LOCK` function are not safe for statement-based replication.

See Also

- GET_LOCK
- RELEASE_LOCK
- IS_USED_LOCK
- RELEASE_ALL_LOCKS

This page is licensed: GPLv2, originally from fill_help_tables.sql

IS_IPV4

Syntax

```
IS_IPV4(expr)
```

Description

If the expression is a valid IPv4 address, returns 1, otherwise returns 0.

`IS_IPV4()` is stricter than INET_ATON(), but as strict as INET6_ATON(), in determining the validity of an IPv4 address. This implies that if IS_IPV4 returns 1, the same expression will always return a non-`NULL` result when passed to `INET_ATON()`, but that the reverse may not apply.

Examples

```
SELECT IS_IPV4('1110.0.1.1');
+----------------------+
| IS_IPV4('1110.0.1.1') |
+----------------------+
|                    0 |
+----------------------+

SELECT IS_IPV4('48f3::d432:1431:ba23:846f');
+--------------------------------------+
| IS_IPV4('48f3::d432:1431:ba23:846f') |
+--------------------------------------+
|                                    0 |
+--------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

IS_IPV4_COMPAT

Syntax

```
IS_IPV4_COMPAT(expr)
```

Description

Returns 1 if a given numeric binary string IPv6 address, such as returned by INET6_ATON(), is IPv4-compatible, otherwise returns 0.

> **CURRENT**
>
> When the argument is not INET6, automatic implicit CAST to INET6 is applied. As a consequence, `IS_IPV4_COMPAT` now understands arguments in both text representation and binary(16) representation.
>
> **< 10.5**
>
> The function understands only binary(16) representation.

Examples

```
SELECT IS_IPV4_COMPAT(INET6_ATON('::10.0.1.1'));
+-----------------------------------------+
| IS_IPV4_COMPAT(INET6_ATON('::10.0.1.1')) |
+-----------------------------------------+
|                                       1 |
+-----------------------------------------+

SELECT IS_IPV4_COMPAT(INET6_ATON('::48f3::d432:1431:ba23:846f'));
+------------------------------------------------------------+
| IS_IPV4_COMPAT(INET6_ATON('::48f3::d432:1431:ba23:846f')) |
+------------------------------------------------------------+
|                                                          0 |
+------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

IS_IPV4_MAPPED

Syntax

```
IS_IPV4_MAPPED(expr)
```

Description

Returns 1 if a given a numeric binary string IPv6 address, such as returned by INET6_ATON(), is a valid IPv4-mapped address, otherwise returns 0.

> **CURRENT**
>
> When the argument is not INET6, automatic implicit CAST to INET6 is applied. As a consequence, `IS_IPV4_MAPPED` now understands arguments in both text representation and binary(16) representation.
>
> **< 10.5**
>
> The function understands only binary(16) representation.

Examples

```
SELECT IS_IPV4_MAPPED(INET6_ATON('::10.0.1.1'));
+-----------------------------------------+
| IS_IPV4_MAPPED(INET6_ATON('::10.0.1.1')) |
+-----------------------------------------+
|                                       0 |
+-----------------------------------------+

SELECT IS_IPV4_MAPPED(INET6_ATON('::ffff:10.0.1.1'));
+-----------------------------------------------+
| IS_IPV4_MAPPED(INET6_ATON('::ffff:10.0.1.1')) |
+-----------------------------------------------+
|                                             1 |
+-----------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

IS_IPV6

Syntax

```
IS_IPV6(expr)
```

Description

Returns 1 if the expression is a valid IPv6 address specified as a string, otherwise returns 0. Does not consider IPv4 addresses to be valid IPv6 addresses.

Examples

```
SELECT IS_IPV6('48f3::d432:1431:ba23:846f');
+--------------------------------------+
| IS_IPV6('48f3::d432:1431:ba23:846f') |
+--------------------------------------+
|                                    1 |
+--------------------------------------+
1 row in set (0.02 sec)

SELECT IS_IPV6('10.0.1.1');
+---------------------+
| IS_IPV6('10.0.1.1') |
+---------------------+
|                   0 |
+---------------------+
```

See Also

- INET6 data type
- INET6_ATON
- INET6_NTOA

This page is licensed: CC BY-SA / Gnu FDL

IS_USED_LOCK

Syntax

```
IS_USED_LOCK(str)
```

Description

Checks whether the lock named `str` is in use (that is, locked). If so, it returns the connection identifier of the client that holds the lock. Otherwise, it returns `NULL`. `str` is case insensitive.

If the metadata_lock_info plugin is installed, the Information Schema metadata_lock_info table contains information about locks of this kind (as well as metadata locks).

Statements using the `IS_USED_LOCK` function are not safe for statement-based replication.

See Also

- GET_LOCK
- RELEASE_LOCK
- IS_FREE_LOCK
- RELEASE_ALL_LOCKS

This page is licensed: GPLv2, originally from fill_help_tables.sql

MASTER_GTID_WAIT

Syntax

```
MASTER_GTID_WAIT(gtid-list[, timeout)
```

Description

This function takes a string containing a comma-separated list of global transaction id's (similar to the value of, for example, gtid_binlog_pos). It waits until the value of gtid_slave_pos has the same or higher `seq_no` within all replication domains specified in the gtid-list; in other words, it waits until the replica has reached the specified GTID position.

An optional second argument gives a timeout in seconds. If the timeout expires before the specified GTID position is reached, then the function returns -1. Passing `NULL` or a negative number for the timeout means no timeout, and the function will wait indefinitely.

If the wait completes without a timeout, 0 is returned. Passing `NULL` for the gtid-list makes the function return `NULL` immediately, without waiting.

The gtid-list may be the empty string, in which case `MASTER_GTID_WAIT()` returns immediately. If the gtid-list contains fewer domains than gtid_slave_pos, then only those domains are waited upon. If gtid-list contains a domain that is not present in `@@gtid_slave_pos`, then `MASTER_GTID_WAIT()` will wait until an event containing such domain_id arrives on the replica (or until timed out or killed).

`MASTER_GTID_WAIT()` can be useful to ensure that a replica has caught up to a master. Simply take the value of gtid_binlog_pos on the master, and use it in a `MASTER_GTID_WAIT()` call on the replica; when the call completes, the replica will have caught up with that master position.

`MASTER_GTID_WAIT()` can also be used in client applications together with the last_gtid session variable. This is useful in a read-scaleout replication setup, where the application writes to a single master but divides the reads out to a number of replica to distribute the load. In such a setup, there is a risk that an application could first do an update on the master, and then a bit later do a read on a replica, and if the replica is not fast enough, the data read from the slave might not include the update just made, possibly confusing the application and/or the end-user. One way to avoid this is to request the value of last_gtid on the master just after the update. Then before doing the read on the replica, do a `MASTER_GTID_WAIT()` on the value obtained from the master; this will ensure that the read is not performed until the replica has replicated sufficiently far for the update to have become visible.

Note that `MASTER_GTID_WAIT()` can be used even if the replica is configured not to use GTID for connections (CHANGE MASTER TO master_use_gtid=no). This is because from MariaDB 10, GTIDs are always logged on the master server, and always recorded on the replica servers.

Differences to MASTER_POS_WAIT()

- `MASTER_GTID_WAIT()` is global; it waits for any master connection to reach the specified GTID position. MASTER_POS_WAIT() works only against a specific connection. This also means that while `MASTER_POS_WAIT()` aborts if its master connection is terminated with STOP REPLICA or due to an error, `MASTER_GTID_WAIT()` continues to wait while replicas are stopped.
- `MASTER_GTID_WAIT()` can take its timeout as a floating-point value, so a timeout in fractional seconds is supported, eg. `MASTER_GTID_WAIT("0-1-100", 0.5)`. (The minimum wait is one microsecond, 0.000001 seconds).
- `MASTER_GTID_WAIT()` allows one to specify a timeout of zero in order to do a non-blocking check to see if the replicas have progressed to a specific GTID position (`MASTER_POS_WAIT()` takes a zero timeout as meaning an

infinite wait). To do an infinite `MASTER_GTID_WAIT()`, specify a negative timeout, or omit the timeout argument.

- `MASTER_GTID_WAIT()` does not return the number of events executed since the wait started, nor does it return `NULL` if a replica thread is stopped. It always returns either 0 for successful wait completed, or -1 for timeout reached (or `NULL` if the specified gtid-pos is `NULL`).

Since `MASTER_GTID_WAIT()` looks only at the seq_no part of the GTIDs, not the server_id, care is needed if a replica becomes diverged from another server so that two different GTIDs with the same seq_no (in the same domain) arrive at the same server. This situation is in any case best avoided; setting gtid_strict_mode is recommended, as this will prevent any such out-of-order sequence numbers from ever being replicated on a replica.

This page is licensed: CC BY-SA / Gnu FDL

MASTER_POS_WAIT

Syntax

```
MASTER_POS_WAIT(log_name,log_pos[,timeout,["connection_name"]])
```

Description

This function is useful in replication for controlling primary/replica synchronization. It blocks until the replica has read and applied all updates up to the specified position (`log_name,log_pos`) in the primary's binary log. The return value is the number of log events the replica had to wait for to advance to the specified position. The function returns `NULL` if the replica SQL thread is not started, the replica's primary information is not initialized, the arguments are incorrect, or an error occurs. It returns -1 if the timeout has been exceeded. If the replica SQL thread stops while `MASTER_POS_WAIT()` is waiting, the function returns `NULL`. If the replica is past the specified position, the function returns immediately.

If a `timeout` value is specified, `MASTER_POS_WAIT()` stops waiting when `timeout` seconds have elapsed. `timeout` must be greater than 0; a zero or negative `timeout` means no `timeout`.

The `connection_name` is used when you are using multi-source-replication. If you don't specify it, it's set to the value of the default_master_connection system variable.

Statements using the `MASTER_POS_WAIT()` function are not safe for statement-based replication.

This page is licensed: GPLv2, originally from fill_help_tables.sql

NAME_CONST

Syntax

```
NAME_CONST(name,value)
```

Description

Returns the given value. When used to produce a result set column, `NAME_CONST()` causes the column to have the given name. The arguments should be constants.

This function is used internally when replicating stored procedures. It makes little sense to use it explicitly in SQL statements, and it was not supposed to be used like that.

```
SELECT NAME_CONST('myname', 14);
+--------+
| myname |
+--------+
|     14 |
+--------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

OLD_VALUE

Note:

This function is available from MariaDB 13.0.

Syntax

```
OLD_VALUE(val)
```

Description

In the `RETURNING` clause of an `UPDATE` statement, `OLD_VALUE()` returns the value before the update. The function is meaningful only in this context.

Examples

```
UPDATE t SET a=a+1 RETURNING OLD_VALUE(a) AS old, a as new;
+------+------+
| old  | new  |
+------+------+
|    1 |    2 |
+------+------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

RELEASE_ALL_LOCKS

Note:

`RELEASE_ALL_LOCKS` is available from MariaDB 10.5.2.

Syntax

```
RELEASE_ALL_LOCKS()
```

Description

Releases all named locks held by the current session. Returns the number of locks released, or `0` if none were held.

Statements using the `RELEASE_ALL_LOCKS` function are not safe for statement-based replication.

Examples

```
SELECT RELEASE_ALL_LOCKS();
+---------------------+
| RELEASE_ALL_LOCKS() |
+---------------------+
|                   0 |
+---------------------+

SELECT GET_LOCK('lock1',10);
+----------------------+
| GET_LOCK('lock1',10) |
+----------------------+
|                    1 |
+----------------------+

SELECT RELEASE_ALL_LOCKS();
+---------------------+
| RELEASE_ALL_LOCKS() |
+---------------------+
|                   1 |
+---------------------+
```

See Also

- GET_LOCK
- IS_FREE_LOCK
- IS_USED_LOCK
- RELEASE_LOCK

This page is licensed: CC BY-SA / Gnu FDL

RELEASE_LOCK

Syntax

```
RELEASE_LOCK(str)
```

Description

Releases the lock named by the string `str` that was obtained with GET_LOCK(). Returns `1` if the lock was released, `0` if the lock was not established by this connection (in which case the lock is not released), and `NULL` if the named lock did not exist. The lock does not exist if it was never obtained by a call to `GET_LOCK()` or if it has previously been released.

`str` is case insensitive. If `str` is an empty string or `NULL`, `RELEASE_LOCK()` returns `NULL` and does nothing.

Statements using the `RELEASE_LOCK` function are not safe for statement-based replication.

The DO statement is convenient to use with `RELEASE_LOCK()`.

Examples

Connection1:

```
SELECT GET_LOCK('lock1',10);
+----------------------+
| GET_LOCK('lock1',10) |
+----------------------+
|                    1 |
+----------------------+
```

Connection 2:

```
SELECT GET_LOCK('lock2',10);
+----------------------+
| GET_LOCK('lock2',10) |
+----------------------+
|                    1 |
+----------------------+
```

Connection 1:

```
SELECT RELEASE_LOCK('lock1'), RELEASE_LOCK('lock2'), RELEASE_LOCK('lock3');
+----------------------+----------------------+----------------------+
| RELEASE_LOCK('lock1') | RELEASE_LOCK('lock2') | RELEASE_LOCK('lock3') |
+----------------------+----------------------+----------------------+
|                    1 |                    0 |                 NULL |
+----------------------+----------------------+----------------------+
```

It is possible to hold the same lock recursively. This example is viewed using the metadata_lock_info plugin:

```
SELECT GET_LOCK('lock3',10);
+----------------------+
| GET_LOCK('lock3',10) |
+----------------------+
|                    1 |
+----------------------+

SELECT GET_LOCK('lock3',10);
+----------------------+
| GET_LOCK('lock3',10) |
+----------------------+
|                    1 |
+----------------------+

SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
+-----------+---------------------+---------------+-----------+--------------+------------+
| THREAD_ID | LOCK_MODE           | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----------+---------------------+---------------+-----------+--------------+------------+
|        46 | MDL_SHARED_NO_WRITE | NULL          | User lock | lock3        |            |
+-----------+---------------------+---------------+-----------+--------------+------------+

SELECT RELEASE_LOCK('lock3');
+----------------------+
| RELEASE_LOCK('lock3') |
+----------------------+
|                    1 |
+----------------------+

SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
+-----------+---------------------+---------------+-----------+--------------+------------+
| THREAD_ID | LOCK_MODE           | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----------+---------------------+---------------+-----------+--------------+------------+
|        46 | MDL_SHARED_NO_WRITE | NULL          | User lock | lock3        |            |
+-----------+---------------------+---------------+-----------+--------------+------------+

SELECT RELEASE_LOCK('lock3');
+----------------------+
| RELEASE_LOCK('lock3') |
+----------------------+
|                    1 |
+----------------------+

SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
Empty set (0.000 sec)
```

See Also

- GET_LOCK
- IS_FREE_LOCK
- IS_USED_LOCK
- RELEASE_ALL_LOCKS

This page is licensed: GPLv2, originally from fill_help_tables.sql

SLEEP

Syntax

```
SLEEP(duration)
```

Description

Sleeps (pauses) for the number of seconds given by the duration argument, then returns `0`. If `SLEEP()` is interrupted, it returns `1`. The duration may have a fractional part given in microseconds.

Statements using the SLEEP() function are not safe for statement-based replication.

Example

```
SELECT SLEEP(5.5);
+------------+
| SLEEP(5.5) |
+------------+
|          0 |
+------------+
1 row in set (5.50 sec)
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SYS_GUID

> **Note:**
>
> `SYS_GUID` is available from MariaDB 10.6.1.

The `SYS_GUID` function was introduced to enhance Oracle compatibility. Similar functionality can be achieved with the UUID function.

Syntax

```
SYS_GUID()
```

Description

Returns a 16-byte globally unique identifier (GUID), similar to the UUID function, but without the `-` character.

Example

```
SELECT SYS_GUID();
+----------------------------------+
| SYS_GUID()                       |
+----------------------------------+
| 2C574E45BA2811EBB265F859713E4BE4 |
+----------------------------------+
```

See Also

- UUID
- UUID_SHORT
- UUID data type

This page is licensed: CC BY-SA / Gnu FDL

UUID

Syntax

```
UUID()
```

Description

Returns a Universally Unique Identifier (UUID) version 1. Functions to generate v4 and v7 UUIDs are available from MariaDB 11.7. See UUIDv4 and UUIDv7 respectively.

A UUID is designed as a number that is globally unique in space and time. Two calls to `UUID()` are expected to generate two different values, even if these calls are performed on two separate computers that are not connected to each other.

UUID() results are intended to be unique, but cannot always be relied upon to be unpredictable and unguessable.

A UUID is a 128-bit number represented by a utf8 string of five hexadecimal numbers in `aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee` format:

- The first three numbers are generated from a timestamp.
- The fourth number preserves temporal uniqueness in case the timestamp value loses monotonicity (for example, due to daylight saving time).
- The fifth number is an IEEE 802 node number that provides spatial uniqueness.
  A random number is substituted if the latter is not available (for example, because the host computer has no Ethernet card, or we do not know how to find the hardware address of an interface on your operating system). In this case, spatial uniqueness cannot be guaranteed. Nevertheless, a collision should have very low probability.

Currently, the MAC address of an interface is taken into account only on FreeBSD and Linux. On other operating systems, MariaDB uses a randomly generated 48-bit number.

Statements using the UUID() function are not safe for statement-based replication.

The function generates a UUIDv1 and the results are generated according to the “DCE 1.1:Remote Procedure Call” (Appendix A) CAE (Common Applications Environment) Specifications published by The Open Group in October 1997 (Document Number C706).

Examples

```
SELECT UUID();
+--------------------------------------+
| UUID()                               |
+--------------------------------------+
| cd41294a-afb0-11df-bc9b-00241dd75637 |
+--------------------------------------+
```

See Also

- UUIDv4
- UUIDv7
- UUID_SHORT() - Return short (64 bit) Universal Unique Identifier
- SYS_GUID - UUID without the `-` character for Oracle compatibility
- UUID data type
- MDEV-11339 (Support UUID v4 generation)
- MDEV-32637 (Implement native UUID7 function)

This page is licensed: GPLv2, originally from fill_help_tables.sql

UUID_SHORT

Syntax

```
UUID_SHORT()
```

Description

Returns a “short” universally unique identifier as a 64-bit unsigned integer (rather than a string-form 128-bit identifier as returned by the UUID() function).

The value of `UUID_SHORT()` is guaranteed to be unique if the following conditions hold:

- The server_id of the current host is unique among your set of master and replica servers.
- `server_id` is between 0 and 255.
- You don’t set back your system time for your server between mariadbd restarts.
- You do not invoke `UUID_SHORT()` on average more than 16 million times per second between mariadbd restarts

The `UUID_SHORT()` return value is constructed this way:

```
(server_id & 255) << 56
+ (server_startup_time_in_seconds << 24)
+ incremented_variable++;
```

Statements using the UUID_SHORT() function are not safe for statement-based replication.

Examples

```
SELECT UUID_SHORT();
+-------------------+
| UUID_SHORT()      |
+-------------------+
| 21517162376069120 |
+-------------------+
```

```
CREATE TABLE t1 (a BIGINT UNSIGNED DEFAULT(uuid_short()) PRIMARY KEY);
INSERT INTO t1 VALUES(),();
SELECT * FROM t1;
+-------------------+
| a                 |
+-------------------+
| 98113699159474176 |
| 98113699159474177 |
+-------------------+
```

See Also

- UUID() ; Return full (128 bit) Universally Unique Identifier
- AUTO_INCREMENT
- Sequences - an alternative to auto_increment available from MariaDB 10.3
- SYS_GUID - UUID without the `-` character for Oracle compatibility
- UUID data type

This page is licensed: GPLv2, originally from fill_help_tables.sql

UUID_v4

> **Note:**
>
> `UUID_v4` is available from MariaDB 11.7.

It is possible to generate UUIDv4 and UUIDv7, in addition to UUIDv1.

Syntax

```
UUID_v4()
```

Description

Returns a Universally Unique Identifier (UUID) version 4. To generate a version 1 UUID, see the UUID function. To generate a version 7 UUID, see UUIDv7.

A UUID is designed as a number that is globally unique in space and time. Two calls to `UUID()` are expected to generate two different values, even if these calls are performed on two separate computers that are not connected to each other.

UUID_v4() results are intended to be unique, but cannot always be relied upon to be unpredictable and unguessable.

A UUID is a 128-bit number represented by a utf8 string of five hexadecimal numbers in `aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee` format.

Statements using the UUID_v4() function are not safe for statement-based replication.

Examples

```
SELECT UUID(),UUID_v4(),UUID_v7()\G
*************************** 1. row ***************************
  UUID(): 63ae8c92-799a-11ef-98b2-f859713e4be4
UUID_v4(): a2443495-1b94-415b-b6fa-fe8e79ba4812
UUID_v7(): 01921e85-f198-7490-9b89-7dd0d468543b

CREATE TABLE t1 (a INT PRIMARY KEY NOT NULL, u UUID DEFAULT UUID_v4(), UNIQUE KEY(u));
```

See Also

- UUID() v1
- UUID_v7()
- UUID_SHORT() - Return short (64 bit) Universal Unique Identifier
- SYS_GUID - UUID without the `-` character for Oracle compatibility
- UUID data type
- MDEV-11339 (Support UUID v4 generation)
- MDEV-32637 (Implement native UUID7 function)

This page is licensed: CC BY-SA / Gnu FDL

UUID_v7

> **Note:**
>
> `UUID_v7` is available from MariaDB 11.7.

It is possible to generate UUIDv4 and UUIDv7, in addition to UUIDv1.

Syntax

```
UUID_v7()
```

Description

Returns a Universally Unique Identifier (UUID) version 7. To generate a version 1 UUID, see the UUID function. To generate a version 4 UUID, see UUID_v4.

A UUID is designed as a number that is globally unique in space and time. Two calls to `UUID()` are expected to generate two different values, even if these calls are performed on two separate computers that are not connected to each other.

A UUID is a 128-bit number represented by a utf8 string of five hexadecimal numbers in `aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee` format.

Statements using the UUID_v7() function are not safe for statement-based replication.

Examples

```
SELECT UUID(),UUID_v4(),UUID_v7()\G
*************************** 1. row ***************************
  UUID(): 63ae8c92-799a-11ef-98b2-f859713e4be4
UUID_v4(): a2443495-1b94-415b-b6fa-fe8e79ba4812
UUID_v7(): 01921e85-f198-7490-9b89-7dd0d468543b
```

See Also

- UUID() v1
- UUID_v4()
- UUID_SHORT() - Return short (64 bit) Universal Unique Identifier
- SYS_GUID - UUID without the `-` character for Oracle compatibility
- UUID data type
- MDEV-11339 (Support UUID v4 generation)
- MDEV-32637 (Implement native UUID7 function)

- uuid7.com

This page is licensed: CC BY-SA / Gnu FDL

VALUES / VALUE

Syntax

```
VALUE(col_name)
```

Description

In an INSERT … ON DUPLICATE KEY UPDATE statement, you can use the `VALUES(col_name)` function in the UPDATE clause to refer to column values from the INSERT portion of the statement. In other words, `VALUES(col_name)` in the `UPDATE` clause refers to the value of col_name that would be inserted, had no duplicate-key conflict occurred. This function is especially useful in multiple-row inserts.

The `VALUES()` function is meaningful only in `INSERT ... ON DUPLICATE KEY UPDATE` statements and returns `NULL` otherwise.

This function was renamed to `VALUE()`, because it's incompatible with the standard Table Value Constructors syntax.

The `VALUES()` function can still be used but only in `INSERT ... ON DUPLICATE KEY UPDATE` statements; it's a syntax error otherwise.

Examples

```
INSERT INTO t (a,b,c) VALUES (1,2,3),(4,5,6)
    ON DUPLICATE KEY UPDATE c=VALUE(a)+VALUE(b);
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

Special Functions

See Dynamic Columns Functions.

Manage schema-less data within relational tables. These functions, such as COLUMN_CREATE and COLUMN_GET, allow you to store and retrieve variable sets of columns in a single BLOB field.

See Galera Functions.

Interact with the Galera Cluster plugin. These functions provide internal status information and control mechanisms specific to synchronous multi-master replication nodes.

See Geographic Functions.

Process geospatial data. This collection of functions allows you to create, analyze, and manipulate geometric shapes like points, lines, and polygons within your database.

See JSON Functions.

Complete JSON Functions reference: JSON_EXTRACT(), JSON_SET(), JSON_REPLACE(), JSON_SEARCH() syntax for path queries, document updates, and value retrieval.

See Window Functions.

Explore window functions in MariaDB Server. This section details SQL functions that perform calculations across a set of table rows related to the current row, enabling advanced analytical queries.

This page is licensed: CC BY-SA / Gnu FDL

Dynamic Columns Functions

See COLUMN_ADD.

Update dynamic columns. This function adds or updates values within a dynamic column blob, returning the new blob content.

See COLUMN_CHECK.

Verify dynamic column integrity. This function checks if a blob containing dynamic columns is valid and returns 1 if it is, 0 otherwise.

See COLUMN_CREATE.

Create a dynamic column blob. This function generates a binary string containing specified column names and values for storage in a BLOB.

See COLUMN_DELETE.

Remove dynamic columns. This function deletes specified columns from a dynamic column blob and returns the updated blob.

See COLUMN_EXISTS.

Check for a dynamic column. This function returns 1 if a specified column exists within a dynamic column blob, and 0 otherwise.

See COLUMN_GET.

Retrieve a dynamic column value. This function extracts a specific column's value from a dynamic column blob, casting it to a specified type.

See COLUMN_JSON.

Convert dynamic columns to JSON. This function returns a JSON string representation of the data stored in a dynamic column blob.

See COLUMN_LIST.

List dynamic column names. This function returns a comma-separated list of all column names contained within a dynamic column blob.

This page is licensed: CC BY-SA / Gnu FDL

COLUMN_ADD

Syntax

```
COLUMN_ADD(dyncol_blob, column_nr, value [as type], [column_nr, value [as type]]...)
COLUMN_ADD(dyncol_blob, column_name, value [as type], [column_name, value [as type]]...)
```

Description

Adds or updates dynamic columns.

- `dyncol_blob` must be either a valid dynamic columns blob (for example, `COLUMN_CREATE` returns such blob), or an empty string.
- `column_name` specifies the name of the column to be added. If `dyncol_blob` already has a column with this name, it will be overwritten.

- `value` specifies the new value for the column. Passing a NULL value will cause the column to be deleted.
- `as type` is optional. See #datatypes section for a discussion about types.

The return value is a dynamic column blob after the modifications.

Examples

```
UPDATE t1 SET dyncol_blob=COLUMN_ADD(dyncol_blob, "column_name", "value") WHERE id=1;
```

Note: `COLUMN_ADD()` is a regular function (just like CONCAT()), hence, in order to update the value in the table you have to use the `UPDATE ... SET dynamic_col=COLUMN_ADD(dynamic_col, ....)` pattern.

This page is licensed: CC BY-SA / Gnu FDL

COLUMN_CHECK

Syntax

```
COLUMN_CHECK(dyncol_blob);
```

Description

Check if `dyncol_blob` is a valid packed dynamic columns blob. Return value of 1 means the blob is valid, return value of 0 means it is not.

Rationale:
Normally, one works with valid dynamic column blobs. Functions like COLUMN_CREATE, COLUMN_ADD, COLUMN_DELETE always return valid dynamic column blobs. However, if a dynamic column blob is accidentally truncated, or transcoded from one character set to another, it will be corrupted. This function can be used to check if a value in a blob field is a valid dynamic column blob.

This page is licensed: CC BY-SA / Gnu FDL

COLUMN_CREATE

Syntax

```
COLUMN_CREATE(column_nr, value [as type], [column_nr, value [as type]]...)
COLUMN_CREATE(column_name, value [as type], [column_name, value [as type]]...)
```

Description

Returns a dynamic columns blob that stores the specified columns with values.

The return value is suitable for

- storing in a table;
- further modification with other dynamic columns functions.

The **`as type`** part allows one to specify the value type. In most cases, this is redundant because MariaDB will be able to deduce the type of the value. Explicit type specification may be needed when the type of the value is not apparent. For example, a literal `'2012-12-01'` has a CHAR type by default, one will need to specify `'2012-12-01' AS DATE` to have it stored as a date. See Dynamic Columns:Datatypes for further details.

Examples

```
INSERT INTO tbl SET dyncol_blob=COLUMN_CREATE("column_name", "value");
```

This page is licensed: CC BY-SA / Gnu FDL

COLUMN_DELETE

Syntax

```
COLUMN_DELETE(dyncol_blob, column_nr, column_nr...)
COLUMN_DELETE(dyncol_blob, column_name, column_name...)
```

Description

Deletes a dynamic column with the specified name. Multiple names can be given. The return value is a dynamic column blob after the modification.

This page is licensed: CC BY-SA / Gnu FDL

COLUMN_EXISTS

Syntax

```
COLUMN_EXISTS(dyncol_blob, column_nr)
COLUMN_EXISTS(dyncol_blob, column_name)
```

Description

Checks if a column with name `column_name` exists in `dyncol_blob`. If yes, return `1`, otherwise return `0`. See dynamic columns for more information.

This page is licensed: CC BY-SA / Gnu FDL

COLUMN_GET

Syntax

```
COLUMN_GET(dyncol_blob, column_nr as type)
COLUMN_GET(dyncol_blob, column_name as type)
```

Description

Gets the value of a dynamic column by its name. If no column with the given name exists, `NULL` will be returned.

`column_name as type` requires that one specify the datatype of the dynamic column they are reading.

This may seem counter-intuitive: why would one need to specify which datatype they're retrieving? Can't the dynamic columns system figure the datatype from the data being stored?

The answer is: SQL is a statically-typed language. The SQL interpreter needs to know the datatypes of all expressions before the query is run (for example, when one is using prepared statements and runs `"select COLUMN_GET(...)"`, the prepared statement API requires the server to inform the client about the datatype of the column being read before the query is executed and the server can see what datatype the column actually has).

Lengths

Suppose running a query like this:

```
SELECT COLUMN_GET(BLOB, 'colname' AS CHAR) ...
```

Without specifying a maximum length (i.e. using `as CHAR`, not `as CHAR(n)`), MariaDB will report the maximum length of the result set column to be 16,777,216. This may cause excessive memory usage in some client libraries, because they try to pre-allocate a buffer of maximum result set width. To avoid this problem, use CHAR(n) whenever you're using COLUMN_GET in the select list.

See Dynamic Columns:Datatypes for more information about datatypes.

This page is licensed: CC BY-SA / Gnu FDL

COLUMN_JSON

Syntax

```
COLUMN_JSON(dyncol_blob)
```

Description

Returns a JSON representation of data in `dyncol_blob`. Can also be used to display nested columns. See dynamic columns for more information.

Example

```
SELECT item_name, COLUMN_JSON(dynamic_cols) FROM assets;
+-----------------+----------------------------------------+
| item_name       | COLUMN_JSON(dynamic_cols)              |
+-----------------+----------------------------------------+
| MariaDB T-shirt | {"size":"XL","color":"blue"}           |
| Thinkpad Laptop | {"color":"black","warranty":"3 years"} |
+-----------------+----------------------------------------+
```

Warning:

Limitation: `COLUMN_JSON` decodes nested dynamic columns at a nesting level of not more than 10 levels deep. Dynamic columns that are nested deeper than 10 levels are shown as `BINARY` string, without encoding.

This page is licensed: CC BY-SA / Gnu FDL

COLUMN_LIST

Syntax

```
COLUMN_LIST(dyncol_blob);
```

Description

Returns a comma-separated list of column names. The names are quoted with backticks.

See dynamic columns for more information.

This page is licensed: CC BY-SA / Gnu FDL

Galera Functions

See WSREP_LAST_SEEN_GTID.

Return the GTID of the most recent write transaction. This function helps determine the transaction ID to use for synchronization waiting.

See WSREP_LAST_WRITTEN_GTID.

Return the GTID of the client's last write. This function identifies the specific transaction ID generated by the current client's most recent write operation.

See WSREP_SYNC_WAIT_UPTO_GTID.

Wait for a specific GTID to be applied. This function blocks the client until the node has committed the transaction with the specified Global Transaction ID.

This page is licensed: CC BY-SA / Gnu FDL

WSREP_LAST_SEEN_GTID

Syntax

```
WSREP_LAST_SEEN_GTID()
```

Description

Returns the Global Transaction ID of the most recent write transaction observed by the client.

The result can be useful to determine the transaction to provide to WSREP_SYNC_WAIT_UPTO_GTID for waiting and unblocking purposes.

This page is licensed: CC BY-SA / Gnu FDL

WSREP_LAST_WRITTEN_GTID

Syntax

```
WSREP_LAST_WRITTEN_GTID()
```

Description

Returns the Global Transaction ID of the most recent write transaction performed by the client.

This page is licensed: CC BY-SA / Gnu FDL

WSREP_SYNC_WAIT_UPTO_GTID

Syntax

```
WSREP_SYNC_WAIT_UPTO_GTID(gtid[,timeout])
```

Description

Blocks the client until the transaction specified by the given Global Transaction ID is applied and committed by the node.

The optional *timeout* argument can be used to specify a block timeout in seconds. If not provided, the timeout will be indefinite.

Returns the node that applied and committed the Global Transaction ID, `ER_LOCAL_WAIT_TIMEOUT` if the function is timed out before this, or `ER_WRONG_ARGUMENTS` if the function is given an invalid GTID.

The result from WSREP_LAST_SEEN_GTID can be useful to determine the transaction to provide to `WSREP_SYNC_WAIT_UPTO_GTID` for waiting and unblocking purposes.

This page is licensed: CC BY-SA / Gnu FDL

Geographic Functions

Geographic and geometry functions. See Geographic Features for a full discussion of MariaDB's spatial extensions.

This page is licensed: CC BY-SA / Gnu FDL

JSON Functions

- JSONPath Expressions
- Differences between JSON_QUERY and JSON_VALUE
- JSON_ARRAY
- JSON_ARRAYAGG
- JSON_ARRAY_APPEND
- JSON_ARRAY_INSERT
- JSON_ARRAY_INTERSECT
- JSON_COMPACT
- JSON_CONTAINS
- JSON_CONTAINS_PATH
- JSON_DEPTH
- JSON_DETAILED
- JSON_EQUALS
- JSON_EXISTS
- JSON_EXTRACT
- JSON_INSERT
- JSON_KEY_VALUE
- JSON_KEYS
- JSON_LENGTH
- JSON_LOOSE
- JSON_MERGE
- JSON_MERGE_PATCH
- JSON_MERGE_PRESERVE
- JSON_NORMALIZE
- JSON_OBJECT
- JSON_OBJECT_FILTER_KEYS
- JSON_OBJECT_TO_ARRAY
- JSON_OBJECTAGG
- JSON_OVERLAPS
- JSON_PRETTY
- JSON_QUERY
- JSON_QUOTE
- JSON_REMOVE
- JSON_REPLACE
- JSON_SCHEMA_VALID
- JSON_SEARCH
- JSON_SET
- JSON_TABLE
- JSON_TYPE

- JSON_UNQUOTE
- JSON_VALID
- JSON_VALUE
- JSON Validation Function

This page is licensed: CC BY-SA / Gnu FDL

JSONPath Expressions

A number of JSON functions accept JSON Path expressions. MariaDB defines this path as follows:

JSON Path Syntax

```
path : ['lax'] '$' [step]*
```

The path starts with an optional *path mode*. At the moment, MariaDB supports only the "lax" mode, which is also the mode that is used when it is not explicitly specified.

The `$` symbol represents the context item. The search always starts from the context item; because of that, the path always starts with `$`.

Then, it is followed by zero or more steps, which select element(s) in the JSON document. A step may be one of the following:

- Object member selector.
- Array element selector.
- Wildcard selector.

Object Member Selector

To select member(s) in a JSON object, one can use one of the following:

- `.memberName` selects the value of the member with name memberName.
- `."memberName"` - the same as above but allows one to select a member with a name that's not a valid identifier (that is, has space, dot, and/or other characters).
- `.*` - selects the values of all members of the object.

If the current item is an array (instead of an object), nothing will be selected.

Array Element Selector

To select elements of an array, one can use one of the following:

- `[N]` selects element number N in the array. The elements are counted from zero.
- `[*]` selects all elements in the array.

If the current item is an object (instead of an array), nothing will be selected.

> **CURRENT**
>
> JSON path supports negative indexes in an array, 'last' keyword and range notation ('to' keyword) for accessing array elements. Negative indexes start from -1.
>
> - `[-N]` selects n th element from end.
> - `[last-N]` selects n th element from the last element.
> - `[M to N]` selects range of elements starting from index M to N.
>
> Example:

```
SET @json='{
            "A": [0,
                  [1, 2, 3],
                  [4, 5, 6],
                  "seven",
                   0.8,
                   true,
                   false,
                   "eleven",
                  [12, [13, 14], {"key1":"value1"},[15]],
                  true],
            "B": {"C": 1},
            "D": 2
           }';
SELECT JSON_EXTRACT(@json, '$.A[-8][1]');
+-------------------------------------------------+
| JSON_EXTRACT(@json, '$.A[-8][1]')               |
+-------------------------------------------------+
| 5                                               |
+-------------------------------------------------+

SELECT JSON_EXTRACT(@json, '$.A[last-7][1]');
+-----------------------------------------------+
| SELECT JSON_EXTRACT(@json, '$.A[last-7][1]'); |
+-----------------------------------------------+
| 5                                             |
+-----------------------------------------------+

SET @json= '[
             [1, {"key1": "value1"}, 3],
             [false, 5, 6],
             [7, 8, [9, {"key2": 2}, 11]],
             [15, 1.34, [14], ["string1", [16, {"key1":[1,2,3,[4,5,6]]}, 18]]],
             [19, 20],
             21, 22
            ]';

SELECT JSON_EXTRACT(@json, '$[0 to 3][2]');
+-----------------------------------------------+
| JSON_EXTRACT(@json, '$[0 to 3][2]')           |
+-----------------------------------------------+
| [3, 6, [9, {"key2": 2}, 11], [14]]            |
+-----------------------------------------------+
```

< 10.9

JSON path does **not** support negative indexes in an array.

This produces output for first index of eighth from last element of a two dimensional array.

Note: In range notation, when M > N (when M,N are greater than or equal to 0) or (size of array - M or size of array - N when M, N are less than 0), then it is treated as an impossible range and `NULL` is returned.

```
SET @json= '[1, 2, 3, 4, 5]';
SELECT JSON_EXTRACT(@json, '$[4 to 2]');
+----------------------------------+
| JSON_EXTRACT(@json, '$[4 to 2]') |
+----------------------------------+
| NULL                             |
+----------------------------------+
```

Wildcard

The wildcard step, `**`, recursively selects all child elements of the current element. Both array elements and object members are selected.

The wildcard step must not be the last step in the JSONPath expression. It must be followed by an array or object member selector step.

For example, this query selects all object members named `price` in the document:

```
SELECT json_extract(@json_doc, '$**.price');
```

This query, however, selects the second element in each of the arrays present in the document:

```
SELECT json_extract(@json_doc, '$**[2]');
```

Compatibility

MariaDB's JSONPath syntax supports a subset of JSON Path's definition in the SQL Standard. The most notable things not supported are the `strict` mode and filters.

MariaDB's JSONPath is close to MySQL's JSONPath. The wildcard step (`**`) is a non-standard extension that has the same meaning as in MySQL. The difference between MariaDB and MySQL's JSONPath is: MySQL doesn't allow one to specify the mode explicitly (but uses `lax` mode implicitly).

This page is licensed: CC BY-SA / Gnu FDL

Differences between JSON_QUERY and JSON_VALUE

The primary difference between the two functions is that *JSON_QUERY* returns an object or an array, while *JSON_VALUE* returns a scalar.

Take the following JSON document as an example:

```
SET @json='{ "x": [0,1], "y": "[0,1]", "z": "Monty" }';
```

Note that data member "x" is an array, and data members "y" and "z" are strings. The following examples demonstrate the differences between the two functions.

```
SELECT JSON_QUERY(@json,'$'), JSON_VALUE(@json,'$');
+-----------------------------------------+----------------------+
| JSON_QUERY(@json,'$')                   | JSON_VALUE(@json,'$') |
+-----------------------------------------+----------------------+
| { "x": [0,1], "y": "[0,1]", "z": "Monty" } | NULL                  |
+-----------------------------------------+----------------------+

SELECT JSON_QUERY(@json,'$.x'), JSON_VALUE(@json,'$.x');
+-------------------------+-------------------------+
| JSON_QUERY(@json,'$.x') | JSON_VALUE(@json,'$.x') |
+-------------------------+-------------------------+
| [0,1]                   | NULL                    |
+-------------------------+-------------------------+

SELECT JSON_QUERY(@json,'$.y'), JSON_VALUE(@json,'$.y');
+-------------------------+-------------------------+
| JSON_QUERY(@json,'$.y') | JSON_VALUE(@json,'$.y') |
+-------------------------+-------------------------+
| NULL                    | [0,1]                   |
+-------------------------+-------------------------+

SELECT JSON_QUERY(@json,'$.z'), JSON_VALUE(@json,'$.z');
+-------------------------+-------------------------+
| JSON_QUERY(@json,'$.z') | JSON_VALUE(@json,'$.z') |
+-------------------------+-------------------------+
| NULL                    | Monty                   |
+-------------------------+-------------------------+

SELECT JSON_QUERY(@json,'$.x[0]'), JSON_VALUE(@json,'$.x[0]');
+----------------------------+----------------------------+
| JSON_QUERY(@json,'$.x[0]') | JSON_VALUE(@json,'$.x[0]') |
+----------------------------+----------------------------+
| NULL                       | 0                          |
+----------------------------+----------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON Arrow Operators

Syntax

```
json_doc -> path
json_doc ->> path
```

Description

> **Note:**
>
> The `->` and `->>` operators were introduced in MariaDB 13.1.

The `->` and `->>` operators are shorthand for extracting a value from a JSON document by a JSONPath expression:

- `json_doc -> path` is equivalent to `JSON_EXTRACT(json_doc, path)`.
- `json_doc ->> path` is equivalent to `JSON_UNQUOTE ( JSON_EXTRACT (json_doc, path))`.

MariaDB supports these operators for compatibility with MySQL 5.7 and later, where they originated. Each operator builds the same expression as its equivalent function call and inherits the same `NULL` and error behavior. On the left, `json_doc` is any JSON expression — typically a column, a function result, or a JSON literal.

The two operators differ only in quoting:

- `->` returns the extracted value in its JSON representation, so a matched string keeps its surrounding double quotes (for example, `"Alice"`).
- `->>` additionally unquotes the result, returning the raw value (for example, `Alice`). Use `->>` when comparing against, sorting by, or displaying scalar values.

The following caveats apply and distinguish the operators from the underlying functions:

- **The path must be a string literal.** Unlike `JSON_EXTRACT()`, which accepts any expression as its path argument, the right-hand side of `->` and `->>` must be a literal string. A column, user variable (`data -> @path`), or placeholder (`data -> ?`) will not parse.
- **Chaining requires parentheses.** The operators do not chain directly; `col -> '$.a' -> '$.b'` is a syntax error. Wrap the intermediate result in parentheses instead: `(col -> '$.a') -> '$.b'`.
- **Precedence** is higher than the arithmetic and comparison operators, so `col ->> '$.age' = 30` is evaluated as `(col ->> '$.age') = 30`.

Examples

Before MariaDB 13.1, only the function form (`JSON_EXTRACT()` / `JSON_UNQUOTE(JSON_EXTRACT())`) worked in MariaDB; the `->` and `->>` operator form was accepted only by MySQL. As of MariaDB 13.1 both forms work, so queries written for MySQL 5.7+ port across unchanged.

The function form works in both MySQL and MariaDB:

```
SELECT JSON_EXTRACT('{"id":"1", "name":"Name"}', '$.name') `name`;
+--------+
| name   |
+--------+
| "Name" |
+--------+

SELECT JSON_UNQUOTE(JSON_EXTRACT('{"id":"1", "name":"Name"}', '$.name')) `name`;
+------+
| name |
+------+
| Name |
+------+
```

The equivalent operator form worked only in MySQL before MariaDB 13.1, and now works in MariaDB too:

```
SELECT '{"id":"1", "name":"Name"}'->'$.name' `name`;
+--------+
| name   |
+--------+
| "Name" |
+--------+

SELECT '{"id":"1", "name":"Name"}'->>'$.name' `name`;
+------+
| name |
+------+
| Name |
+------+
```

As with the function form, `->` keeps the JSON quoting (`"Name"`) while `->>` returns the unquoted value (`Name`).

The operators can be used anywhere an expression is allowed, such as in a `WHERE` clause:

```
CREATE TABLE t1 (id INT, data JSON);
INSERT INTO t1 VALUES (1, '{"id":"1", "name":"Name"}');

SELECT id FROM t1 WHERE data->>'$.name' = 'Name';
+------+
| id   |
+------+
|    1 |
+------+
```

See Also

- JSON_EXTRACT — the function `->` is shorthand for.
- JSON_UNQUOTE — combined with `JSON_EXTRACT`, the function `->>` is shorthand for.
- JSONPath Expressions — the path syntax used on the right-hand side.
- JSON_VALUE — related, but *not* equivalent: returns only scalar values, always unquoted.
- JSON_QUERY — related, but *not* equivalent: returns only objects or arrays.

This page is licensed: CC BY-SA / Gnu FDL

JSON_ARRAY

Syntax

```
JSON_ARRAY([value[, value2] ...])
```

Description

Returns a JSON array containing the listed values. The list can be empty.

Example

```
SELECT Json_Array(56, 3.1416, 'My name is "Foo"', NULL);
+-------------------------------------------------+
| Json_Array(56, 3.1416, 'My name is "Foo"', NULL) |
+-------------------------------------------------+
| [56, 3.1416, "My name is \"Foo\"", null]         |
+-------------------------------------------------+
```

See also

- JSON_MAKE_ARRAY, the CONNECT storage engine function

This page is licensed: CC BY-SA / Gnu FDL

JSON_ARRAY_APPEND

Syntax

```
JSON_ARRAY_APPEND(json_doc, path, value[, path, value] ...)
```

Description

Appends values to the end of the specified arrays within a JSON document, returning the result, or `NULL` if any of the arguments are `NULL`.

Evaluation is performed from left to right, with the resulting document from the previous pair becoming the new value against which the next pair is evaluated.

If the `json_doc` is not a valid JSON document, or if any of the paths are not valid, or contain a `*` or `**` wildcard, an error is returned.

Examples

```
SET @json = '[1, 2, [3, 4]]';

SELECT JSON_ARRAY_APPEND(@json, '$[0]', 5)
+-------------------------------------+
| JSON_ARRAY_APPEND(@json, '$[0]', 5) |
+-------------------------------------+
| [[1, 5], 2, [3, 4]]                 |
+-------------------------------------+

SELECT JSON_ARRAY_APPEND(@json, '$[1]', 6);
+-------------------------------------+
| JSON_ARRAY_APPEND(@json, '$[1]', 6) |
+-------------------------------------+
| [1, [2, 6], [3, 4]]                 |
+-------------------------------------+

SELECT JSON_ARRAY_APPEND(@json, '$[1]', 6, '$[2]', 7);
+-----------------------------------------------+
| JSON_ARRAY_APPEND(@json, '$[1]', 6, '$[2]', 7) |
+-----------------------------------------------+
| [1, [2, 6], [3, 4, 7]]                        |
+-----------------------------------------------+

SELECT JSON_ARRAY_APPEND(@json, '$', 5);
+----------------------------------+
| JSON_ARRAY_APPEND(@json, '$', 5) |
+----------------------------------+
| [1, 2, [3, 4], 5]                |
+----------------------------------+

SET @json = '{"A": 1, "B": [2], "C": [3, 4]}';

SELECT JSON_ARRAY_APPEND(@json, '$.B', 5);
+-------------------------------------+
| JSON_ARRAY_APPEND(@json, '$.B', 5) |
+-------------------------------------+
| {"A": 1, "B": [2, 5], "C": [3, 4]} |
+-------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_ARRAY_INSERT

Syntax

```
JSON_ARRAY_INSERT(json_doc, path, value[, path, value] ...)
```

Description

Inserts a value into a JSON document, returning the modified document, or `NULL` if any of the arguments are `NULL`.

Evaluation is performed from left to right, with the resulting document from the previous pair becoming the new value against which the next pair is evaluated.

If the `json_doc` is not a valid JSON document, or if any of the paths are not valid, or contain a `*` or `**` wildcard, an error is returned.

Examples

```
SET @json = '[1, 2, [3, 4]]';

SELECT JSON_ARRAY_INSERT(@json, '$[0]', 5);
+-----------------------------------+
| JSON_ARRAY_INSERT(@json, '$[0]', 5) |
+-----------------------------------+
| [5, 1, 2, [3, 4]]                 |
+-----------------------------------+

SELECT JSON_ARRAY_INSERT(@json, '$[1]', 6);
+-----------------------------------+
| JSON_ARRAY_INSERT(@json, '$[1]', 6) |
+-----------------------------------+
| [1, 6, 2, [3, 4]]                 |
+-----------------------------------+

SELECT JSON_ARRAY_INSERT(@json, '$[1]', 6, '$[2]', 7);
+-----------------------------------------------+
| JSON_ARRAY_INSERT(@json, '$[1]', 6, '$[2]', 7) |
+-----------------------------------------------+
| [1, 6, 7, 2, [3, 4]]                          |
+-----------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_ARRAY_INTERSECT

> **Note:**
>
> `JSON_ARRAY_INTERSECT` is available from MariaDB 11.2.

Syntax

```
JSON_ARRAY_INTERSECT(arr1, arr2)
```

Description

Finds intersection between two json arrays and returns an array of items found in both array.

Examples

```
SET @json1= '[1,2,3]';
SET @json2= '[1,2,4]';

SELECT json_array_intersect(@json1, @json2);
+-------------------------------------+
| json_array_intersect(@json1, @json2) |
+-------------------------------------+
| [1, 2]                              |
+-------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_COMPACT

Syntax

```
JSON_COMPACT(json_doc)
```

Description

Removes all unnecessary spaces so the json document is as short as possible.

Example

```
SET @j = '{ "A": 1, "B": [2, 3]}';

SELECT JSON_COMPACT(@j), @j;
+-------------------+-----------------------+
| JSON_COMPACT(@j)  | @j                    |
+-------------------+-----------------------+
| {"A":1,"B":[2,3]} | { "A": 1, "B": [2, 3]} |
+-------------------+-----------------------+
```

See Also

- JSON video tutorial covering JSON_COMPACT.

This page is licensed: CC BY-SA / Gnu FDL

JSON_CONTAINS

Syntax

```
JSON_CONTAINS(json_doc, val[, path])
```

Description

Returns whether or not the specified value is found in the given JSON document or, optionally, at the specified path within the document. Returns `1` if it does, `0` if not and `NULL` if any of the arguments are null. An error occurs if the document or path is not valid, or contains the `*` or `**` wildcards.

Examples

```
SET @json = '{"A": 0, "B": {"C": 1}, "D": 2}';

SELECT JSON_CONTAINS(@json, '2', '$.A');
+---------------------------------+
| JSON_CONTAINS(@json, '2', '$.A') |
+---------------------------------+
|                               0 |
+---------------------------------+

SELECT JSON_CONTAINS(@json, '2', '$.D');
+---------------------------------+
| JSON_CONTAINS(@json, '2', '$.D') |
+---------------------------------+
|                               1 |
+---------------------------------+

SELECT JSON_CONTAINS(@json, '{"C": 1}', '$.A');
+---------------------------------------+
| JSON_CONTAINS(@json, '{"C": 1}', '$.A') |
+---------------------------------------+
|                                     0 |
+---------------------------------------+

SELECT JSON_CONTAINS(@json, '{"C": 1}', '$.B');
+---------------------------------------+
| JSON_CONTAINS(@json, '{"C": 1}', '$.B') |
+---------------------------------------+
|                                     1 |
+---------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_CONTAINS_PATH

Syntax

```
JSON_CONTAINS_PATH(json_doc, return_arg, path[, path] ...)
```

Description

Indicates whether the given JSON document contains data at the specified path or paths. Returns `1` if it does, `0` if not and `NULL` if any of the arguments are null.

The *return_arg* can be `one` or `all`:

- `one` - Returns `1` if at least one path exists within the JSON document.
- `all` - Returns `1` only if all paths exist within the JSON document.

Examples

```
SET @json = '{"A": 1, "B": [2], "C": [3, 4]}';

SELECT JSON_CONTAINS_PATH(@json, 'one', '$.A', '$.D');
+-----------------------------------------------+
| JSON_CONTAINS_PATH(@json, 'one', '$.A', '$.D') |
+-----------------------------------------------+
|                                             1 |
+-----------------------------------------------+
1 row in set (0.00 sec)

SELECT JSON_CONTAINS_PATH(@json, 'all', '$.A', '$.D');
+-----------------------------------------------+
| JSON_CONTAINS_PATH(@json, 'all', '$.A', '$.D') |
+-----------------------------------------------+
|                                             0 |
+-----------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_DEPTH

Syntax

```
JSON_DEPTH(json_doc)
```

Description

Returns the maximum depth of the given JSON document, or `NULL` if the argument is null. An error occurs if the argument is an invalid JSON document.

- Scalar values or empty arrays or objects have a depth of 1.
- Arrays with only scalar values and objects with only scalar values for all keys have depth of 1.
- In all other cases, the depth can be 2 or greater.

> **12.2**
>
> There is no maximum depth level — it's unlimited.
>
> For more information, see this blog post.
>
> **< 12.2**
>
> The maximum depth is 32.

Examples

```
SELECT JSON_DEPTH('[]'), JSON_DEPTH('true'), JSON_DEPTH('{}');
+------------------+--------------------+------------------+
| JSON_DEPTH('[]') | JSON_DEPTH('true') | JSON_DEPTH('{}') |
+------------------+--------------------+------------------+
|                1 |                  1 |                1 |
+------------------+--------------------+------------------+

SELECT JSON_DEPTH('[1, 2, 3]'), JSON_DEPTH('[[], {}, []]');
+-------------------------+----------------------------+
| JSON_DEPTH('[1, 2, 3]') | JSON_DEPTH('[[], {}, []]') |
+-------------------------+----------------------------+
|                       2 |                          2 |
+-------------------------+----------------------------+

SELECT JSON_DEPTH('[1, 2, [3, 4, 5, 6], 7]');
+---------------------------------------+
| JSON_DEPTH('[1, 2, [3, 4, 5, 6], 7]') |
+---------------------------------------+
|                                     3 |
+---------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_DETAILED

Syntax

```
JSON_DETAILED(json_doc[, tab_size])
JSON_PRETTY(json_doc[, tab_size])
```

Description

Represents JSON in the most understandable way emphasizing nested structures.

> **CURRENT**
>
> `JSON_PRETTY` is an alias for `JSON_DETAILED` .
>
> **< 10.10.3 / 10.9.5 / 10.8.7 / 10.7.8 / 10.6.12 / 10.5.19 / 10.4.28**
>
> `JSON_PRETTY` is **not** available as an alias for `JSON_DETAILED` .

Example

```
SET @j = '{ "A":1,"B":[2,3]}';

SELECT @j;
+--------------------+
| @j                 |
+--------------------+
| { "A":1,"B":[2,3]} |
+--------------------+

SELECT JSON_DETAILED(@j);
+------------------------------------------------------------+
| JSON_DETAILED(@j)                                          |
+------------------------------------------------------------+
| {
    "A": 1,
    "B":
    [
        2,
        3
    ]
} |
+------------------------------------------------------------+
```

See Also

- JSON video tutorial covering JSON_DETAILED.

This page is licensed: CC BY-SA / Gnu FDL

JSON_EQUALS

> **Note:**
>
> `JSON_EQUALS` is available from MariaDB 10.7.

Syntax

```
JSON_EQUALS(json1, json2)
```

Description

Checks if there is equality between two json objects. Returns `1` if it there is, `0` if not, or NULL if any of the arguments are null.

Examples

```
SELECT JSON_EQUALS('{"a"   :[1, 2, 3],"b":[4]}', '{"b":[4],"a":[1, 2, 3.0]}');
+-------------------------------------------------------------------------+
| JSON_EQUALS('{"a"   :[1, 2, 3],"b":[4]}', '{"b":[4],"a":[1, 2, 3.0]}') |
+-------------------------------------------------------------------------+
|                                                                       1 |
+-------------------------------------------------------------------------+

SELECT JSON_EQUALS('{"a":[1, 2, 3]}', '{"a":[1, 2, 3.01]}');
+-----------------------------------------------------+
| JSON_EQUALS('{"a":[1, 2, 3]}', '{"a":[1, 2, 3.01]}') |
+-----------------------------------------------------+
|                                                   0 |
+-----------------------------------------------------+
```

See Also

- JSON_NORMALIZE

This page is licensed: CC BY-SA / Gnu FDL

JSON_EXISTS

Syntax

```
JSON_EXISTS(json_doc, json_path)
```

Description

Determines whether `json_doc` has an element pointed to by path `json_path`. Returns `1` if the element exists, `0` if not, or `NULL` if any of the inputs were `NULL`.

Examples

```
SELECT JSON_EXISTS('{"key1":"xxxx", "key2":[1, 2, 3]}', "$.key2");
+-----------------------------------------------------------+
| JSON_EXISTS('{"key1":"xxxx", "key2":[1, 2, 3]}', "$.key2") |
+-----------------------------------------------------------+
|                                                         1 |
+-----------------------------------------------------------+

SELECT JSON_EXISTS('{"key1":"xxxx", "key2":[1, 2, 3]}', "$.key3");
+-----------------------------------------------------------+
| JSON_EXISTS('{"key1":"xxxx", "key2":[1, 2, 3]}', "$.key3") |
+-----------------------------------------------------------+
|                                                         0 |
+-----------------------------------------------------------+

SELECT JSON_EXISTS('{"key1":"xxxx", "key2":[1, 2, 3]}', "$.key2[1]");
+--------------------------------------------------------------+
| JSON_EXISTS('{"key1":"xxxx", "key2":[1, 2, 3]}', "$.key2[1]") |
+--------------------------------------------------------------+
|                                                            1 |
+--------------------------------------------------------------+

SELECT JSON_EXISTS('{"key1":"xxxx", "key2":[1, 2, 3]}', "$.key2[10]");
+---------------------------------------------------------------+
| JSON_EXISTS('{"key1":"xxxx", "key2":[1, 2, 3]}', "$.key2[10]") |
+---------------------------------------------------------------+
|                                                             0 |
+---------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_EXTRACT

Syntax

```
JSON_EXTRACT(json_doc, path[, path] ...)
```

Description

Extracts data from a JSON document. The extracted data is selected from the parts matching the path arguments. Returns all matched values; either as a single matched value, or, if the arguments could return multiple values, a result autowrapped as an array in the matching order.

Returns `NULL` if no paths match or if any of the arguments are `NULL`.

An error occurs if any path argument is not a valid path, or if the json_doc argument is not a valid JSON document.

The path expression be a JSONPath expression as supported by MariaDB

Examples

```
SET @json = '[1, 2, [3, 4]]';

SELECT JSON_EXTRACT(@json, '$[1]');
+----------------------------+
| JSON_EXTRACT(@json, '$[1]') |
+----------------------------+
| 2                          |
+----------------------------+

SELECT JSON_EXTRACT(@json, '$[2]');
+----------------------------+
| JSON_EXTRACT(@json, '$[2]') |
+----------------------------+
| [3, 4]                     |
+----------------------------+

SELECT JSON_EXTRACT(@json, '$[2][1]');
+-------------------------------+
| JSON_EXTRACT(@json, '$[2][1]') |
+-------------------------------+
| 4                             |
+-------------------------------+
```

See Also

- JSON Arrow Operators — the `->` operator is shorthand for `JSON_EXTRACT()`.
- JSON video tutorial covering JSON_EXTRACT.

This page is licensed: CC BY-SA / Gnu FDL

JSON_INSERT

Syntax

```
JSON_INSERT(json_doc, path, val[, path, val] ...)
```

Description

Inserts data into a JSON document, returning the resulting document or `NULL` if either of the *json_doc* or *path* arguments are null.

An error occurs if the JSON document is invalid, or if any of the paths are invalid or contain a `*` or `**` wildcard.

JSON_INSERT can only insert data, while JSON_REPLACE can only update. JSON_SET can update or insert data.

Examples

```
SET @json = '{ "A": 0, "B": [1, 2]}';

SELECT JSON_INSERT(@json, '$.C', '[3, 4]');
+-------------------------------------+
| JSON_INSERT(@json, '$.C', '[3, 4]')  |
+-------------------------------------+
| { "A": 0, "B": [1, 2], "C":"[3, 4]"} |
+-------------------------------------+
```

See Also

- JSON video tutorial covering JSON_INSERT.

This page is licensed: CC BY-SA / Gnu FDL

JSON_KEY_VALUE

> **Note:**
>
> JSON_KEY_VALUE is available from MariaDB 11.2.

Syntax

```
JSON_KEY_VALUE(obj, json_path)
```

Description

`JSON_KEY_VALUE` extracts key/value pairs from a JSON object. The JSON path parameter is used to only return key/value pairs for matching JSON objects.

Example

```
SELECT JSON_KEY_VALUE('[[1, {"key1":"val1", "key2":"val2"}, 3], 2, 3]', '$[0][1]');
+-------------------------------------------------------------------------+
| JSON_KEY_VALUE('[[1, {"key1":"val1", "key2":"val2"}, 3], 2, 3]', '$[0][1]') |
+-------------------------------------------------------------------------+
| [{"key": "key1", "value": "val1"}, {"key": "key2", "value": "val2"}]        |
+-------------------------------------------------------------------------+
```

`JSON_KEY_VALUE()` can be used as an argument to JSON_TABLE(), which allows adding the key to a result set.

```
SELECT jt.* FROM JSON_TABLE(
JSON_KEY_VALUE('[[1, {"key1":"val1", "key2":"val2"}, 3], 2, 3]', '$[0][1]'),'$[*]'
COLUMNS (
k VARCHAR(20) PATH '$.key',
v VARCHAR(20) PATH '$.value',
id FOR ORDINALITY )) AS jt;
+------+------+------+
| k    | v    | id   |
+------+------+------+
| key1 | val1 |    1 |
| key2 | val2 |    2 |
+------+------+------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_KEYS

Syntax

```
JSON_KEYS(json_doc[, path])
```

Description

Returns the keys as a JSON array from the top-level value of a JSON object or, if the optional path argument is provided, the top-level keys from the path.

Excludes keys from nested sub-objects in the top level value. The resulting array will be empty if the selected object is empty.

Returns `NULL` if any of the arguments are null, a given path does not locate an object, or if the json_doc argument is not an object.

An error will occur if JSON document is invalid, the path is invalid or if the path contains a `*` or `**` wildcard.

Examples

```
SELECT JSON_KEYS('{"A": 1, "B": {"C": 2}}');
+--------------------------------------+
| JSON_KEYS('{"A": 1, "B": {"C": 2}}') |
+--------------------------------------+
| ["A", "B"]                           |
+--------------------------------------+

SELECT JSON_KEYS('{"A": 1, "B": 2, "C": {"D": 3}}', '$.C');
+-----------------------------------------------------+
| JSON_KEYS('{"A": 1, "B": 2, "C": {"D": 3}}', '$.C') |
+-----------------------------------------------------+
| ["D"]                                               |
+-----------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_LENGTH

Syntax

```
JSON_LENGTH(json_doc[, path])
```

Description

Returns the length of a JSON document, or, if the optional path argument is given, the length of the value within the document specified by the path.

Returns `NULL` if any of the arguments argument are null or the path argument does not identify a value in the document.

An error occurs if the JSON document is invalid, the path is invalid or if the path contains a `*` or `**` wildcard.

Length will be determined as follow:

- A scalar's length is always 1.
- If an array, the number of elements in the array.
- If an object, the number of members in the object.

The length of nested arrays or objects are not counted.

Examples

```

```

This page is licensed: CC BY-SA / Gnu FDL

JSON_LOOSE

Syntax

```
JSON_LOOSE(json_doc)
```

Description

Adds spaces to a JSON document to make it look more readable.

Example

```
SET @j = '{ "A":1,"B":[2,3]}';

SELECT JSON_LOOSE(@j), @j;
+----------------------+--------------------+
| JSON_LOOSE(@j)       | @j                 |
+----------------------+--------------------+
| {"A": 1, "B": [2, 3]} | { "A":1,"B":[2,3]} |
+----------------------+--------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_MERGE

Syntax

```
JSON_MERGE(json_doc, json_doc[, json_doc] ...)
```

Description

Merges the given JSON documents.

Returns the merged result, or `NULL` if any argument is `NULL`.

An error occurs if any of the arguments are not valid JSON documents.

> **Warning:**
>
> `JSON_MERGE` is deprecated. JSON_MERGE_PATCH is an RFC 7396-compliant replacement, and JSON_MERGE_PRESERVE is a synonym.

Example

```
SET @json1 = '[1, 2]';
SET @json2 = '[3, 4]';

SELECT JSON_MERGE(@json1,@json2);
+---------------------------+
| JSON_MERGE(@json1,@json2) |
+---------------------------+
| [1, 2, 3, 4]              |
+---------------------------+
```

See Also

- JSON_MERGE_PATCH
- JSON_MERGE_PRESERVE

This page is licensed: CC BY-SA / Gnu FDL

JSON_MERGE_PATCH

Syntax

```
JSON_MERGE_PATCH(json_doc, json_doc[, json_doc] ...)
```

Description

Merges the given JSON documents, returning the merged result, or `NULL` if any argument is `NULL`.

`JSON_MERGE_PATCH` is an RFC 7396-compliant replacement for JSON_MERGE, which is deprecated.

Unlike JSON_MERGE_PRESERVE, members with duplicate keys are not preserved.

Example

```
SET @json1 = '[1, 2]';
SET @json2 = '[2, 3]';
SELECT JSON_MERGE_PATCH(@json1,@json2),JSON_MERGE_PRESERVE(@json1,@json2);
+---------------------------------+-----------------------------------+
| JSON_MERGE_PATCH(@json1,@json2) | JSON_MERGE_PRESERVE(@json1,@json2) |
+---------------------------------+-----------------------------------+
| [2, 3]                          | [1, 2, 2, 3]                      |
+---------------------------------+-----------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_MERGE_PRESERVE

Syntax

```
JSON_MERGE_PRESERVE(json_doc, json_doc[, json_doc] ...)
```

Description

Merges the given JSON documents, returning the merged result, or `NULL` if any argument is `NULL`.

`JSON_MERGE_PRESERVE` is a synonym for JSON_MERGE, which has been deprecated.

Unlike JSON_MERGE_PATCH, members with duplicate keys are preserved.

Example

```
SET @json1 = '[1, 2]';
SET @json2 = '[2, 3]';
SELECT JSON_MERGE_PATCH(@json1,@json2),JSON_MERGE_PRESERVE(@json1,@json2);
+---------------------------------+-----------------------------------+
| JSON_MERGE_PATCH(@json1,@json2) | JSON_MERGE_PRESERVE(@json1,@json2) |
+---------------------------------+-----------------------------------+
| [2, 3]                          | [1, 2, 2, 3]                      |
+---------------------------------+-----------------------------------+
```

See Also

- JSON_MERGE_PATCH

This page is licensed: CC BY-SA / Gnu FDL

JSON_NORMALIZE

> **Note:**
>
> `JSON_NORMALIZE` is available from MariaDB 10.7.

Syntax

```
JSON_NORMALIZE(json)
```

Description

Recursively sorts keys and removes spaces, allowing comparison of json documents for equality.

Examples

We may wish our application to use the database to enforce a unique constraint on the JSON contents, and we can do so using the `JSON_NORMALIZE` function in combination with a unique key.

For example, if we have a table with a JSON column:

```
CREATE TABLE t1 (
 id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
 val JSON,
 /* other columns here */
 PRIMARY KEY (id)
);
```

Add a unique constraint using JSON_NORMALIZE like this:

```
ALTER TABLE t1
   ADD COLUMN jnorm JSON AS (JSON_NORMALIZE(val)) VIRTUAL,
   ADD UNIQUE KEY (jnorm);
```

We can test this by first inserting a row as normal:

```
INSERT INTO t1 (val) VALUES ('{"name":"alice","color":"blue"}');
```

And then seeing what happens with a different string which would produce the same JSON object:

```
INSERT INTO t1 (val) VALUES ('{ "color": "blue", "name": "alice" }');
ERROR 1062 (23000): Duplicate entry '{"color":"blue","name":"alice"}' for key 'jnorm'
```

See Also

- JSON_EQUALS

This page is licensed: CC BY-SA / Gnu FDL

JSON_OBJECT

Syntax

```
JSON_OBJECT([key, value[, key, value] ...])
```

Description

Returns a JSON object containing the given key/value pairs. The key/value list can be empty.

An error will occur if there are an odd number of arguments, or any key name is `NULL`.

Example

```
SELECT JSON_OBJECT("id", 1, "name", "Monty");
+---------------------------------------+
| JSON_OBJECT("id", 1, "name", "Monty") |
+---------------------------------------+
| {"id": 1, "name": "Monty"}            |
+---------------------------------------+
```

See also

- JSON_MAKE_OBJECT, the CONNECT storage engine function

This page is licensed: CC BY-SA / Gnu FDL

JSON_OBJECT_FILTER_KEYS

> **Note:**
>
> JSON_OBJECT_FILTER_KEYS is available from MariaDB 11.2.

Syntax

```
JSON_OBJECT_FILTER_KEYS(obj, array_keys)
```

Description

`JSON_OBJECT_FILTER_KEYS` returns a JSON object with keys from the object that are also present in the array as string. It is used when one wants to get key-value pair such that the keys are common but the values may not be common.

Example

```
SET @obj1= '{ "a": 1, "b": 2, "c": 3}';
SET @obj2= '{"b" : 10, "c": 20, "d": 30}';
SELECT JSON_OBJECT_FILTER_KEYS (@obj1, JSON_ARRAY_INTERSECT(JSON_KEYS(@obj1), JSON_KEYS(@obj2)));
+-----------------------------------------------------------------------------------------+
| JSON_OBJECT_FILTER_KEYS (@obj1, JSON_ARRAY_INTERSECT(JSON_KEYS(@obj1), JSON_KEYS(@obj2))) |
+-----------------------------------------------------------------------------------------+
| {"b": 2, "c": 3}                                                                        |
+-----------------------------------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_OBJECT_TO_ARRAY

> **Note:**
>
> JSON_OBJECT_TO_ARRAY is available from MariaDB 11.2.

Syntax

```
JSON_OBJECT_TO_ARRAY(Obj)
```

Description

It is used to convert all JSON objects found in a JSON document to JSON arrays where each item in the outer array represents a single key-value pair from the object. It is used when we want not just common keys, but also common values. It can be used in conjunction with `JSON_ARRAY_INTERSECT()`.

Examples

```
SET @obj1= '{ "a": [1, 2, 3], "b": { "key1":"val1", "key2": {"key3":"val3"} }}';

SELECT JSON_OBJECT_TO_ARRAY(@obj1);
+-----------------------------------------------------------------------+
| JSON_OBJECT_TO_ARRAY(@obj1)                                           |
+-----------------------------------------------------------------------+
| [["a", [1, 2, 3]], ["b", {"key1": "val1", "key2": {"key3": "val3"}}]] |
+-----------------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_OVERLAPS

> **Note:**
>
> `JSON_OVERLAPS` is available from MariaDB 10.9.

Syntax

```
JSON_OVERLAPS(json_doc1, json_doc2)
```

Description

`JSON_OVERLAPS()` compares two json documents and returns true if they have at least one common key-value pair between two objects, array element common between two arrays, or array element common with scalar if one of the arguments is a scalar and other is an array. If two json documents are scalars, it returns true if they have same type and value.

If none of the above conditions are satisfied then it returns false.

Examples

```
SELECT JSON_OVERLAPS('false', 'false');
+---------------------------------+
| JSON_OVERLAPS('false', 'false') |
+---------------------------------+
| 1                               |
+---------------------------------+

SELECT JSON_OVERLAPS('true', '["abc", 1, 2, true, false]');
+---------------------------------------------------+
| JSON_OVERLAPS('true','["abc", 1, 2, true, false]') |
+---------------------------------------------------+
| 1                                                 |
+---------------------------------------------------+

SELECT JSON_OVERLAPS('{"A": 1, "B": {"C":2}}', '{"A": 2, "B": {"C":2}}') AS is_overlap;
+---------------------+
| is_overlap          |
+---------------------+
| 1                   |
+---------------------+
```

Partial match is considered as no-match.

Examples

```
SELECT JSON_OVERLAPS('[1, 2, true, false, null]', '[3, 4, [1]]') AS is_overlap;
+--------------------- +
| is_overlap           |
+----------------------+
| 0                    |
+----------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_PRETTY

> **Note:**
>
> JSON_PRETTY is available from MariaDB 10.10.3, 10.9.5, 10.8.7, 10.7.8, 10.6.12, 10.5.19, and 10.4.28.

`JSON_PRETTY` is an alias for JSON_DETAILED.

This page is licensed: CC BY-SA / Gnu FDL

JSON_QUERY

Syntax

```
JSON_QUERY(json_doc, path)
```

Description

Given a JSON document, returns an object or array specified by the path. Returns `NULL` if not given a valid JSON document, or if there is no match.

Examples

```
SELECT json_query('{"key1":{"a":1, "b":[1,2]}}', '$.key1');
+-----------------------------------------------------+
| json_query('{"key1":{"a":1, "b":[1,2]}}', '$.key1') |
+-----------------------------------------------------+
| {"a":1, "b":[1,2]}                                  |
+-----------------------------------------------------+

SELECT json_query('{"key1":123, "key1": [1,2,3]}', '$.key1');
+-----------------------------------------------------+
| json_query('{"key1":123, "key1": [1,2,3]}', '$.key1') |
+-----------------------------------------------------+
| [1,2,3]                                             |
+-----------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_QUOTE

Syntax

```
JSON_QUOTE(json_value)
```

Description

Quotes a string as a JSON value, usually for producing valid JSON string literals for inclusion in JSON documents. Wraps the string with double quote characters and escapes interior quotes and other special characters, returning a

utf8mb4 string.

Returns `NULL` if the argument is `NULL`.

Examples

```
SELECT JSON_QUOTE('A'), JSON_QUOTE("B"), JSON_QUOTE('"C"');
+-----------------+-----------------+------------------+
| JSON_QUOTE('A') | JSON_QUOTE("B") | JSON_QUOTE('"C"') |
+-----------------+-----------------+------------------+
| "A"             | "B"             | "\"C\""          |
+-----------------+-----------------+------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_REMOVE

Syntax

```
JSON_REMOVE(json_doc, path[, path] ...)
```

Description

Removes data from a JSON document returning the result, or `NULL` if any of the arguments are null. If the element does not exist in the document, no changes are made.

The function returns `NULL` and throws a warning if the JSON document is invalid, the path is invalid, contains a range, or contains a `*` or `**` wildcard.

Path arguments are evaluated from left to right, with the result from the earlier evaluation being used as the value for the next.

Examples

```
SELECT JSON_REMOVE('{"A": 1, "B": 2, "C": {"D": 3}}', '$.C');
+-----------------------------------------------------+
| JSON_REMOVE('{"A": 1, "B": 2, "C": {"D": 3}}', '$.C') |
+-----------------------------------------------------+
| {"A": 1, "B": 2}                                    |
+-----------------------------------------------------+

SELECT JSON_REMOVE('["A", "B", ["C", "D"], "E"]', '$[1]');
+--------------------------------------------------+
| JSON_REMOVE('["A", "B", ["C", "D"], "E"]', '$[1]') |
+--------------------------------------------------+
| ["A", ["C", "D"], "E"]                           |
+--------------------------------------------------+
```

See Also

- JSON video tutorial covering JSON_REMOVE.

This page is licensed: CC BY-SA / Gnu FDL

JSON_REPLACE

Syntax

```
JSON_REPLACE(json_doc, path, val[, path, val] ...)
```

Description

Replaces existing values in a JSON document, returning the result, or `NULL` if any of the arguments are `NULL`.

An error will occur if the JSON document is invalid, the path is invalid or if the path contains a `*` or `**` wildcard.

Paths and values are evaluated from left to right, with the result from the earlier evaluation being used as the value for the next.

JSON_REPLACE can only update data, while JSON_INSERT can only insert. JSON_SET can update or insert data.

Examples

```
SELECT JSON_REPLACE('{ "A": 1, "B": [2, 3]}', '$.B[1]', 4);
+-----------------------------------------------------+
| JSON_REPLACE('{ "A": 1, "B": [2, 3]}', '$.B[1]', 4) |
+-----------------------------------------------------+
| { "A": 1, "B": [2, 4]}                              |
+-----------------------------------------------------+
```

See Also

- JSON video tutorial covering JSON_REPLACE.

This page is licensed: CC BY-SA / Gnu FDL

JSON_SCHEMA_VALID

> **Note:**
>
> `JSON_SCHEMA_VALID` is available from MariaDB 11.1.

Syntax

```
JSON_SCHEMA_VALID(schema, json);
```

Description

`JSON_SCHEMA_VALID` allows MariaDB to support JSON schema validation. If a given json is valid against a schema it returns true. When JSON does not validate against the schema, it does not return a message about which keyword it failed against and only returns false.

The function supports JSON Schema Draft 2020 with a few exceptions:

- External resources are not supported.
- Hyper schema keywords are not supported.
- Formats like date, email etc are treated as annotations.

Examples

To create validation rules for json field, do this:

```
CREATE TABLE obj_table(val_obj JSON CHECK(JSON_SCHEMA_VALID('{
  "type":"object",
    "properties": {
      "number1":{
        "type":"number",
        "maximum":5,
        "const":4
      },
      "string1":{
        "type":"string",
        "maxLength":5,
        "minLength":3
      },
    "object1":{
      "type":"object",
       "properties":{
         "key1": {"type":"string"},
         "key2":{"type":"array"},
         "key3":{"type":"number", "minimum":3}
       },
       "dependentRequired": { "key1":["key3"] }
     }
  },
  "required":["number1","object1"]
  }', val_obj)));

INSERT INTO obj_table VALUES(
  '{"number1":4, "string1":"abcd",
  "object1":{"key1":"val1", "key2":[1,2,3, "string1"], "key3":4}}'
);

INSERT INTO obj_table VALUES(
  '{"number1":3, "string1":"abcd",
  "object1":{"key1":"val1", "key2":[1,2,3, "string1"], "key3":4}}'
);
ERROR 4025 (23000): CONSTRAINT `obj_table.val_obj` failed for `test`.`obj_table`

SELECT * FROM obj_table;
+-----------------------------------------------------------------------------------------------------+
| val_obj                                                                                             |
+-----------------------------------------------------------------------------------------------------+
| {"number1":4, "string1":"abcd", "object1":{"key1":"val1", "key2":[1,2,3, "string1"], "key3":4}} |
+-----------------------------------------------------------------------------------------------------+

SET @schema= '{
  "properties" : {
    "number1":{ "maximum":10 },
    "string1" : { "maxLength": 3}
  }
}';

SELECT JSON_SCHEMA_VALID(@schema, '{ "number1":25, "string1":"ab" }');
+-------------------------------------------------------------------+
| JSON_SCHEMA_VALID(@schema, '{ "number1":25, "string1":"ab" }') |
+-------------------------------------------------------------------+
|                                                                 0 |
+-------------------------------------------------------------------+

SELECT JSON_SCHEMA_VALID(@schema, '{ "number1":10, "string1":"ab" }');
+-------------------------------------------------------------------+
| JSON_SCHEMA_VALID(@schema, '{ "number1":10, "string1":"ab" }') |
+-------------------------------------------------------------------+
|                                                                 1 |
+-------------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_SEARCH

Syntax

```
JSON_SEARCH(json_doc, return_arg, search_str[, escape_char[, path] ...])
```

Description

Returns the path to the given string within a JSON document, or `NULL` if any of *json_doc*, *search_str* or a path argument is `NULL`; if the search string is not found, or if no path exists within the document.

A warning will occur if the JSON document is not valid, any of the path arguments are not valid, if *return_arg* is neither *one* nor *all*, or if the escape character is not a constant. `NULL` will be returned.

return_arg can be one of two values:

- `'one`: Terminates after finding the first match, so will return one path string. If there is more than one match, it is undefined which is considered first.
- `all`: Returns all matching path strings, without duplicates. Multiple strings are autowrapped as an array. The order is undefined.

Examples

```
SET @json = '["A", [{"B": "1"}], {"C":"AB"}, {"D":"BC"}]';

SELECT JSON_SEARCH(@json, 'one', 'AB');
+---------------------------------+
| JSON_SEARCH(@json, 'one', 'AB') |
+---------------------------------+
| "$[2].C"                        |
+---------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_SET

Syntax

```
JSON_SET(json_doc, path, val[, path, val] ...)
```

Description

Updates or inserts data into a JSON document, returning the result, or `NULL` if any of the arguments are `NULL` or the optional path fails to find an object.

An error will occur if the JSON document is invalid, the path is invalid or if the path contains a `*` or a wildcard**.**

`JSON_SET` can update or insert data, while JSON_REPLACE can only update, and JSON_INSERT only insert.

Examples

```
SELECT JSON_SET(Priv, '$.locked', 'true') FROM mysql.global_priv
```

This page is licensed: CC BY-SA / Gnu FDL

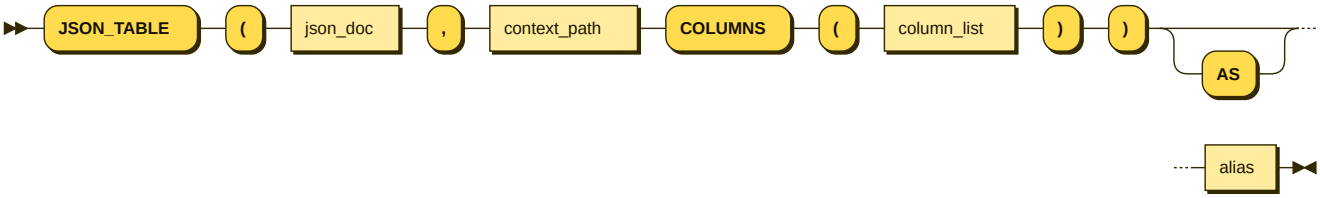
JSON_TABLE

> **Note:**
>
> `JSON_TABLE` is available from MariaDB 10.6.

`JSON_TABLE` is a table function that converts JSON data into a relational form.

Syntax

```
JSON_TABLE(json_doc,
          context_path COLUMNS (column_list)
) [AS] alias
```



Railroad diagram of JSON_TABLE — equivalent to the BNF above

```
column_list:
    column[, column][, ...]
```

```
column:
    name FOR ORDINALITY
    |  name type PATH path_str [on_empty] [on_error]
    |  name type FORMAT JSON PATH path_str [on_empty] [on_error]
    |  name type EXISTS PATH path_str
    |  NESTED PATH path_str COLUMNS (column_list)
```

```
on_empty:
    {NULL | DEFAULT string | ERROR} ON EMPTY
```

```
on_error:
    {NULL | DEFAULT string | ERROR} ON ERROR
```

Description

`JSON_TABLE` can be used in contexts where a table reference can be used; in the `FROM` clause of a SELECT statement, and in multi-table UPDATE/DELETE statements.

`json_doc` is the JSON document to extract data from. In the simplest case, it is a string literal containing JSON. In more complex cases it can be an arbitrary expression returning JSON. The expression may have references to columns of other tables. However, one can only refer to tables that precede this JSON_TABLE invocation. For RIGHT JOIN, it is assumed that its outer side precedes the inner. All tables in outer selects are also considered preceding.

`context_path` is a JSON Path expression pointing to a collection of nodes in `json_doc` that will be used as the source of rows.

The `COLUMNS` clause declares the names and types of the columns that JSON_TABLE returns, as well as how the values of the columns are produced.

Column Definitions

The following types of columns are supported:

Path Columns

```
name type PATH path_str [on_empty] [on_error]
```

Locates the JSON node pointed to by `path_str` and returns its value. The path_str is evaluated using the current row source node as the context node.

```
SET @json='
[
  {"name":"Laptop", "color":"black", "price":"1000"},
  {"name":"Jeans",  "color":"blue"}
]';

SELECT * FROM json_table(@json, '$[*]'
  COLUMNS(
   name  VARCHAR(10) path '$.name',
   color VARCHAR(10) path '$.color',
   price DECIMAL(8,2) path '$.price' )
) AS jt;
+--------+-------+---------+
| name   | color | price   |
+--------+-------+---------+
| Laptop | black | 1000.00 |
| Jeans  | blue  |    NULL |
+--------+-------+---------+
```

The `on_empty` and `on_error` clauses specify the actions to be performed when the value was not found or there was an error condition. See the `ON EMPTY` and `ON ERROR` clauses section for details.

FORMAT JSON Columns

> **Note:**
>
> `FORMAT JSON` columns are available from MariaDB 13.1.1.

```
name type FORMAT JSON PATH path_str [on_empty] [on_error]
```

Like a path column, but returns the JSON value located by `path_str` as-is, serialized to text — including arrays and objects, not only scalar values. This is useful for pulling a nested array or object into a single column without expanding it into multiple rows with `NESTED PATH`.

The column must be declared with a string type, such as VARCHAR or TEXT. Using `FORMAT JSON` with a non-string type raises an error.

```
SET @json='
[
  {"name":"Jeans",   "sizes": [32, 34, 36]},
  {"name":"T-Shirt", "sizes":["Medium", "Large"]},
  {"name":"Cellphone"}
]';

SELECT * FROM JSON_TABLE(@json, '$[*]'
  COLUMNS(
    name  VARCHAR(10)  PATH '$.name',
    sizes VARCHAR(100) FORMAT JSON PATH '$.sizes'
  )
) AS jt;
+-----------+---------------------+
| name      | sizes               |
+-----------+---------------------+
| Jeans     | [32, 34, 36]        |
| T-Shirt   | ["Medium", "Large"] |
| Cellphone | NULL                |
+-----------+---------------------+
```

Without `FORMAT JSON`, a path column returns only scalar values; locating an array or object instead triggers the `ON ERROR` behavior (`NULL` by default). A column declared with the JSON type applies `FORMAT JSON` implicitly — see Extracting a Subdocument into a Column.

ORDINALITY Columns

```
name FOR ORDINALITY
```

Counts the rows, starting from 1.

Example:

```
set @json='
[
  {"name":"Laptop", "color":"black"},
  {"name":"Jeans",  "color":"blue"}
]';

select * from json_table(@json, '$[*]'
  columns(
   id for ordinality,
   name  varchar(10) path '$.name')
) as jt;
+------+--------+
| id   | name   |
+------+--------+
|    1 | Laptop |
|    2 | Jeans  |
+------+--------+
```

EXISTS PATH Columns

```
name type EXISTS PATH path_str
```

Checks whether the node pointed to by `value_path` exists. The `value_path` is evaluated using the current row source node as the context node.

```
set @json='
[
  {"name":"Laptop", "color":"black", "price":1000},
  {"name":"Jeans",  "color":"blue"}
]';

select * from json_table(@json, '$[*]'
  columns(
   name  varchar(10) path '$.name',
   has_price integer exists path '$.price')
) as jt;
+--------+-----------+
| name   | has_price |
+--------+-----------+
| Laptop |         1 |
| Jeans  |         0 |
+--------+-----------+
```

NESTED PATHs

NESTED PATH converts nested JSON structures into multiple rows.

```
NESTED PATH path COLUMNS (column_list)
```

It finds the sequence of JSON nodes pointed to by `path` and uses it to produce rows. For each found node, a row is generated with column values as specified by the NESTED PATH's COLUMNS clause. If `path` finds no nodes, only one row is generated with all columns having NULL values.

For example, consider a JSON document that contains an array of items, and each item, in turn, is expected to have an array of its available sizes:

```
SET @json='
[
  {"name":"Jeans",  "sizes": [32, 34, 36]},
  {"name":"T-Shirt", "sizes":["Medium", "Large"]},
  {"name":"Cellphone"}
]';
```

NESTED PATH allows one to produce a separate row for each size each item has:

```
select * from json_table(@json, '$[*]'
  columns(
    name  varchar(10) path '$.name',
    nested path '$.sizes[*]' columns (
      size varchar(32) path '$'
    )
  )
) as jt;
+-----------+--------+
| name      | size   |
+-----------+--------+
| Jeans     | 32     |
| Jeans     | 34     |
| Jeans     | 36     |
| T-Shirt   | Medium |
| T-Shirt   | Large  |
| Cellphone | NULL   |
+-----------+--------+
```

`NESTED PATH` clauses can be nested within one another. They can also be located next to each other. In that case, the nested path clauses will produce records one at a time. The ones that are not producing records will have all columns set to `NULL`.

Example:

```
set @json='
[
  {"name":"Jeans",  "sizes": [32, 34, 36], "colors":["black", "blue"]}
]';

select * from json_table(@json, '$[*]'
  columns(
    name  varchar(10) path '$.name',
    nested path '$.sizes[*]' columns (
      size varchar(32) path '$'
    ),
    nested path '$.colors[*]' columns (
      color varchar(32) path '$'
    )
  )
) as jt;

+-------+------+-------+
| name  | size | color |
+-------+------+-------+
| Jeans | 32   | NULL  |
| Jeans | 34   | NULL  |
| Jeans | 36   | NULL  |
| Jeans | NULL | black |
| Jeans | NULL | blue  |
+-------+------+-------+
```

ON EMPTY and ON ERROR Clauses

The ON EMPTY clause specifies what will be done when the element specified by the search path is missing in the JSON document.

```
on_empty:
    {NULL | DEFAULT string | ERROR} ON EMPTY
```

When `ON EMPTY` clause is not present, `NULL ON EMPTY` is implied.

```
on_error:
    {NULL | DEFAULT string | ERROR} ON ERROR
```

The ON ERROR clause specifies what should be done if a JSON structure error occurs when trying to extract the value pointed to by the path expression. A JSON structure error here occurs only when one attempts to convert a JSON non-scalar (array or object) into a scalar value. When the `ON ERROR` clause is not present, `NULL ON ERROR` is implied.

Note: A datatype conversion error (e.g. attempt to store a non-integer value into an integer field, or a varchar column being truncated) is not considered a JSON error and so will not trigger the `ON ERROR` behavior. It will produce warnings, in the same way as CAST(value AS datatype) would.

Replication

In the current code, evaluation of `JSON_TABLE` is deterministic, that is, for a given input string `JSON_TABLE` will always produce the same set of rows in the same order. However, one can think of JSON documents that one can consider identical which will produce different output. In order to be future-proof and withstand changes like

- sorting JSON object members by name (like MySQL does);
- changing the way duplicate object members are handled the function is marked as unsafe for statement-based replication.

Extracting a Subdocument into a Column

CURRENT

```
SELECT * FROM JSON_TABLE('{"foo": [1,2,3,4]}','$' columns( jscol json path '$.foo') ) AS T;
+-----------+
| jscol     |
+-----------+
| [1,2,3,4] |
+-----------+
```

< 10.6.9

`JSON_TABLE` does not allow to extract a JSON "subdocument" into a JSON column.

```
SELECT * FROM JSON_TABLE('{"foo": [1,2,3,4]}','$' columns( jscol json path '$.foo') ) AS T;
+-------+
| jscol |
+-------+
| NULL  |
+-------+
```

See Also

- JSON Support (video)

This page is licensed: CC BY-SA / Gnu FDL

JSON_TYPE

Syntax

```
JSON_TYPE(json_val)
```

Description

Returns the type of a JSON value (as a string), or `NULL` if the argument is null.

An error will occur if the argument is an invalid JSON value.

The following is a complete list of the possible return types:

Return type	Value	Example
ARRAY	JSON array	[1, 2, {"key": "value"}]
OBJECT	JSON object	{"key":"value"}
BOOLEAN	JSON true/false literals	true, false
DOUBLE	A number with at least one floating point decimal.	1.2
INTEGER	A number without a floating point decimal.	1
NULL	JSON null literal (this is returned as a string, not to be confused with the SQL NULL value!)	null
STRING	JSON String	"a sample string"

Examples

```
SELECT JSON_TYPE('{"A": 1, "B": 2, "C": 3}');
+--------------------------------------+
| JSON_TYPE('{"A": 1, "B": 2, "C": 3}') |
+--------------------------------------+
| OBJECT                               |
+--------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON_UNQUOTE

Syntax

```
JSON_UNQUOTE(val)
```

Description

Unquotes a JSON value, returning a string, or `NULL` if the argument is null.

An error will occur if the given value begins and ends with double quotes and is an invalid JSON string literal.

If the given value is not a JSON string, value is passed through unmodified.

Certain character sequences have special meanings within a string. Usually, a backslash is ignored, but the escape sequences in the table below are recognised by MariaDB, unless the SQL Mode is set to `NO_BACKSLASH_ESCAPES` .

Escape sequence	Character
”	Double quote (“)
\b	Backslash
\f	Formfeed
\n	Newline (linefeed)
\r	Carriage return
\t	Tab
\	Backslash ()
\uXXXX	UTF-8 bytes for Unicode value XXXX

Examples

```
SELECT JSON_UNQUOTE('"Monty"');
+-------------------------+
| JSON_UNQUOTE('"Monty"') |
+-------------------------+
| Monty                   |
+-------------------------+
```

With the default SQL Mode:

```
SELECT JSON_UNQUOTE('Si\bng\ting');
+----------------------------+
| JSON_UNQUOTE('Si\bng\ting') |
+----------------------------+
| Sng   ing                  |
+----------------------------+
```

Setting `NO_BACKSLASH_ESCAPES` :

```
SET @@sql_mode = 'NO_BACKSLASH_ESCAPES';

SELECT JSON_UNQUOTE('Si\bng\ting');
+----------------------------+
| JSON_UNQUOTE('Si\bng\ting') |
+----------------------------+
| Si\bng\ting                |
+----------------------------+
```

See Also

- JSON Arrow Operators — the `->>` operator is shorthand for `JSON_UNQUOTE(JSON_EXTRACT())`.

This page is licensed: CC BY-SA / Gnu FDL

JSON_VALID

Syntax

```
JSON_VALID(value)
```

Description

Indicates whether the given value is a valid JSON document or not. Returns `1` if valid, `0` if not, and `NULL` if the argument is `NULL`.

> **CURRENT**
>
> The `JSON_VALID` function is automatically used as a CHECK constraint for the JSON data type alias in order to ensure that a valid json document is inserted.
>
> **< 10.4.3**
>
> The `JSON_VALID` function is **not** automatically used as a CHECK constraint for the JSON data type alias.

Examples

```
SELECT JSON_VALID('{"id": 1, "name": "Monty"}');
+-----------------------------------------+
| JSON_VALID('{"id": 1, "name": "Monty"}') |
+-----------------------------------------+
|                                       1 |
+-----------------------------------------+

SELECT JSON_VALID('{"id": 1, "name": "Monty", "oddfield"}');
+-----------------------------------------------------+
| JSON_VALID('{"id": 1, "name": "Monty", "oddfield"}') |
+-----------------------------------------------------+
|                                                   0 |
+-----------------------------------------------------+
```

See Also

- JSON video tutorial covering JSON_VALID.

This page is licensed: CC BY-SA / Gnu FDL

JSON_VALUE

Syntax

```
JSON_VALUE(json_doc, path)
```

Description

Given a JSON document, returns the scalar specified by the path. Returns NULL if not given a valid JSON document, or if there is no match.

Examples

```
SELECT json_value('{"key1":123}', '$.key1');
+-------------------------------------+
| json_value('{"key1":123}', '$.key1') |
+-------------------------------------+
| 123                                 |
+-------------------------------------+

SELECT json_value('{"key1": [1,2,3], "key1":123}', '$.key1');
+-------------------------------------------------------+
| json_value('{"key1": [1,2,3], "key1":123}', '$.key1') |
+-------------------------------------------------------+
| 123                                                   |
+-------------------------------------------------------+
```

In the `SET` statement below, two escape characters are needed, as a single escape character would be applied by the SQL parser in the `SET` statement, and the escaped character would not form part of the saved value.

```
SET @json = '{"key1":"60\\" Table", "key2":"1"}';

SELECT JSON_VALUE(@json,'$.key1') AS Name , json_value(@json,'$.key2') as ID;
+-----------+------+
| Name      | ID   |
+-----------+------+
| 60" Table | 1    |
+-----------+------+
```

This page is licensed: CC BY-SA / Gnu FDL

JSON Validation Function

IS JSON

Test whether a JSON expression includes valid data.

Syntax

```
<expr> IS [ NOT ] JSON
       [ { VALUE | ARRAY | OBJECT | SCALAR } ]
       [ { WITH | WITHOUT } UNIQUE [ KEYS ] ]
```

Description

The `IS JSON` is a variant of IS predicate that checks whether an expression contains valid JSON data, with optional constraints on JSON type (VALUE, ARRAY, OBJECT, SCALAR) and key uniqueness.

Example

```
SELECT '{"a": 42}' IS JSON OBJECT;
```

Output

```
+----------------------------+
| '{"a": 42}' IS JSON OBJECT |
+----------------------------+
|                          1|
+----------------------------+
```

See Also

- IS JSON (Comparison Operator) for details, including syntax and examples.

This page is licensed: CC BY-SA / Gnu FDL

Window Functions

- Window Functions Overview
- Window Frames
- ColumnStore Window Functions
- CUME_DIST
- DENSE_RANK
- FIRST_VALUE
- LAG
- LEAD
- MEDIAN
- NTH_VALUE
- NTILE
- PERCENT_RANK
- PERCENTILE_CONT
- PERCENTILE_DISC
- RANK
- ROW_NUMBER

This page is licensed: CC BY-SA / Gnu FDL

Window Functions Overview

Window Functions

Window functions calculate across related rows without collapsing them. Unlike `GROUP BY`, they return one result for each input row.

When to Use Window Functions

Use window functions when you need to:

- Rank rows inside a group.
- Calculate running totals or moving averages.
- Compare a row with earlier or later rows.
- Return the top *N* rows per group.

Basic Syntax

```
window_function(expr) OVER (
  [PARTITION BY expr [, ...]]
  [ORDER BY expr [ASC | DESC] [, ...]]
  [{ROWS | RANGE} frame_clause]
)

frame_clause:
  {frame_border | BETWEEN frame_border AND frame_border}

frame_border:
  UNBOUNDED PRECEDING
  | UNBOUNDED FOLLOWING
  | CURRENT ROW
  | expr PRECEDING
  | expr FOLLOWING
```

You can also define a named window and reuse it:

```
SELECT
  SUM(score) OVER w AS running_total,
  AVG(score) OVER w AS running_avg
FROM student
WINDOW w AS (
  ORDER BY score
  ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
);
```

Window Functions vs `GROUP BY`

Use `GROUP BY` when you want one output row per group.

Use a window function when you want to keep the original rows and add per-group or per-sequence calculations beside them.

```
SELECT test, AVG(score)
FROM student
GROUP BY test;
```

This returns one row per `test`.

```
SELECT
  name,
  test,
  score,
  AVG(score) OVER (PARTITION BY test) AS avg_by_test
FROM student;
```

This returns every row, plus the average for that row's `test`.

How `OVER` Works

`PARTITION BY`

`PARTITION BY` starts a new calculation for each group.

`ORDER BY`

`ORDER BY` defines the row sequence inside each partition.

Frame

The frame controls which rows contribute to the current result. Aggregate window functions use frames. Ranking functions such as `ROW_NUMBER()` and `RANK()` do not.

> **Warning:**
>
> `OVER ()` uses the whole result set.
>
> For aggregate window functions, `OVER (ORDER BY ...)` uses a running frame by default. In MariaDB, that default is `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`.

`ROWS` vs `RANGE`

`ROWS` counts physical rows. `RANGE` groups peer rows that share the same `ORDER BY` value.

Use `ROWS` when you need strict row-by-row stepping. Use `RANGE` when ties should be treated as one peer group.

```
CREATE TABLE t (score INT);
INSERT INTO t VALUES (10), (10), (20);

SELECT
  score,
  SUM(score) OVER (
    ORDER BY score
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS rows_sum,
  SUM(score) OVER (
    ORDER BY score
    RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS range_sum
FROM t;
```

This produces different results for the duplicate `10` values:

```
+-------+----------+-----------+
| score | rows_sum | range_sum |
+-------+----------+-----------+
|    10 |       10 |        20 |
|    10 |       20 |        20 |
|    20 |       40 |        40 |
+-------+----------+-----------+
```

Supported Functions

Dedicated window functions include CUME_DIST, DENSE_RANK, FIRST_VALUE, LAG, LAST_VALUE, LEAD, MEDIAN, NTH_VALUE, NTILE, PERCENTILE_CONT, PERCENTILE_DISC, PERCENT_RANK, RANK, and ROW_NUMBER.

Aggregate functions that also work with `OVER (...)` include AVG, BIT_AND, BIT_OR, BIT_XOR, COUNT, MAX, MIN, STD, STDDEV, STDDEV_POP, STDDEV_SAMP, SUM, VAR_POP, VAR_SAMP, and VARIANCE.

Aggregate window functions do not support `DISTINCT`.

Aggregate Functions as Window Functions

It is possible to use aggregate functions as window functions. An aggregate function used as a window function must have the `OVER` clause. For example, here's COUNT() used as a window function:

```
SELECT COUNT(*) OVER (ORDER BY column) FROM table;
```

Common Pitfalls

- You cannot reference a window function in `WHERE`.
- Compute the window result in a subquery or CTE first.
- MariaDB does not support `GROUPS` frames.
- MariaDB does not support frame exclusion.
- MariaDB does not support explicit `NULLS FIRST` or `NULLS LAST`.
- `RANGE` frames do not support `DATE` or `DATETIME` arithmetic.

> **Note:**
>
> Window functions are evaluated after `WHERE`, `GROUP BY`, and `HAVING`. Filter the computed result in an outer query or CTE.

Optimization

Window function evaluation invokes a sort pass per group of compatible windows. Each such pass appears as a `filesort` node in the execution plan. Query shape decides how many passes are needed and how cheap each one is to perform.

`GROUP BY` Comes First

If a query uses both `GROUP BY` and window functions, MariaDB executes the grouping step first. The window step runs on the grouped result.

This affects index usage. An index that helps the base table scan does not automatically avoid later sorting for the window stage.

> **Note:**
>
> When tuning this pattern, optimize the `GROUP BY` step first. Then check `EXPLAIN FORMAT=JSON` to confirm the window step's sort cost is low.

Sort Reuse Depends on Sort Keys

If the `GROUP BY` definition and the window's `PARTITION BY` and `ORDER BY` definition use different sort keys, the window's sort pass cannot reuse the grouped output's order — it has to re-sort the rows. Aligning the keys keeps the sort cheap; misaligning them forces a full sort.

For example, this shape can require an extra sort:

```
SELECT
  dept,
  month,
  SUM(sales) AS monthly_sales,
  ROW_NUMBER() OVER (
    PARTITION BY dept
    ORDER BY month
  ) AS row_num
FROM revenue
GROUP BY month, dept;
```

The grouped result is ordered by `month, dept`. The window step needs `dept, month`. Those orders do not match.

Multiple Window Functions Can Share One Sort

Multiple window functions can share the same sort pass when their `PARTITION BY` and `ORDER BY` clauses are compatible. Two specs are compatible if they are identical, or if one is a prefix of the other — the longer specification then becomes the canonical sort order.

Identical specs:

```
SELECT
  dept,
  month,
  sales,
  ROW_NUMBER() OVER (
    PARTITION BY dept
    ORDER BY month
  ) AS row_num,
  SUM(sales) OVER (
    PARTITION BY dept
    ORDER BY month
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS running_sales
FROM revenue;
```

Both window functions use the same partitioning and ordering. MariaDB reuses the same sorted stream for both — one `filesort` pass.

Prefix-compatible specs:

```
SELECT
  dept,
  month,
  day,
  ROW_NUMBER() OVER (PARTITION BY dept ORDER BY month)      AS rn,
  SUM(sales)   OVER (PARTITION BY dept ORDER BY month, day) AS running_sales
FROM revenue;
```

The first window's order key (`dept` , `month`) is a prefix of the second's (`dept` , `month` , `day`). MariaDB sorts once by the longer key (`dept, month, day`) and both window functions consume the result.

Practical Tuning Tips

- Expect `GROUP BY` to shape the input seen by window functions.
- Align `GROUP BY` keys with the window sort keys when possible — the sort still runs, but it is cheap when the data is already in the right order.
- Reuse the same `PARTITION BY` and `ORDER BY` across multiple window functions, or make one a prefix of another, so MariaDB can collapse them into a single sort pass.
- Use `EXPLAIN FORMAT=JSON` (or `ANALYZE FORMAT=JSON` for real timings) to count the window-function sort passes — each pass appears as a `filesort` node under the aggregation step.

Examples

Given the following data:

```
CREATE TABLE student (
  name CHAR(10),
  test CHAR(10),
  score TINYINT
);

INSERT INTO student VALUES
  ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
  ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
  ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
  ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);
```

Average by Test

```
SELECT
  name,
  test,
  score,
  AVG(score) OVER (PARTITION BY test) AS avg_by_test
FROM student
ORDER BY test, name;
```

```
+---------+--------+-------+-------------+
| name    | test   | score | avg_by_test |
+---------+--------+-------+-------------+
| Chun    | SQL    |    75 |     65.2500 |
| Esben   | SQL    |    43 |     65.2500 |
| Kaolin  | SQL    |    56 |     65.2500 |
| Tatiana | SQL    |    87 |     65.2500 |
| Chun    | Tuning |    73 |     68.7500 |
| Esben   | Tuning |    31 |     68.7500 |
| Kaolin  | Tuning |    88 |     68.7500 |
| Tatiana | Tuning |    83 |     68.7500 |
+---------+--------+-------+-------------+
```

Running Total by Score

```
SELECT
  name,
  score,
  SUM(score) OVER (
    ORDER BY score
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS running_total
FROM student
ORDER BY score;
```

```
+---------+-------+---------------+
| name    | score | running_total |
+---------+-------+---------------+
| Esben   |    31 |            31 |
| Esben   |    43 |            74 |
| Kaolin  |    56 |           130 |
| Chun    |    73 |           203 |
| Chun    |    75 |           278 |
| Tatiana |    83 |           361 |
| Tatiana |    87 |           448 |
| Kaolin  |    88 |           536 |
+---------+-------+---------------+
```

Top 2 Scores per Test

```
WITH ranked AS (
  SELECT
    name,
    test,
    score,
    RANK() OVER (PARTITION BY test ORDER BY score DESC) AS rnk
  FROM student
)
SELECT
  name,
  test,
  score,
  rnk
FROM ranked
WHERE rnk <= 2
ORDER BY test, rnk, name;
```

```
+---------+--------+-------+-----+
| name    | test   | score | rnk |
+---------+--------+-------+-----+
| Tatiana | SQL    |    87 |   1 |
| Chun    | SQL    |    75 |   2 |
| Kaolin  | Tuning |    88 |   1 |
| Tatiana | Tuning |    83 |   2 |
+---------+--------+-------+-----+
```

MariaDB Support and Limitations

MariaDB:

- Supports `ROWS` and `RANGE` frames.
- Supports all frame bounds, including `RANGE n PRECEDING` and `RANGE n FOLLOWING`.
- Does not support `GROUPS` frames.
- Does not support frame exclusion.
- Does not support explicit `NULLS FIRST` or `NULLS LAST`.
- Does not support `DATE` or `DATETIME` arithmetic for `RANGE` frames.
- Does not support `DISTINCT` for aggregate window functions.

See Also

- Window Frames
- Aggregate Functions as Window Functions
- ROW_NUMBER
- RANK
- AVG
- SUM
- Introduction to Window Functions in MariaDB Server 10.2

This page is licensed: CC BY-SA / Gnu FDL

Window Frames

Basic Syntax

```
frame_clause:
  {ROWS | RANGE} {frame_border | BETWEEN frame_border AND frame_border}

frame_border:
  | UNBOUNDED PRECEDING
  | UNBOUNDED FOLLOWING
  | CURRENT ROW
  | expr PRECEDING
  | expr FOLLOWING
```

How Window Frames Work

A basic overview of window functions is described in Window Functions Overview. Window frames define which rows contribute to the current result.

Window frames are used by aggregate window functions. They are defined by the frame clause inside `OVER (...)`.

> **Warning:**
>
> `OVER ()` uses the whole result set.
>
> For aggregate window functions, `OVER (ORDER BY ...)` uses a running frame by default. In MariaDB, that default is `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`.

Frame Bound Types

Common frame bounds include:

- All rows before the current row (`UNBOUNDED PRECEDING`), for example `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`.
- All rows after the current row (UNBOUNDED FOLLOWING), for example `RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING`.
- A set number of rows before the current row (`expr PRECEDING`), for example `RANGE BETWEEN 6 PRECEDING AND CURRENT ROW`.
- A set number of rows after the current row (`expr FOLLOWING`), for example `RANGE BETWEEN CURRENT ROW AND 2 FOLLOWING`.
- A specified number of rows both before and after the current row, for example `RANGE BETWEEN 6 PRECEDING AND 3 FOLLOWING`.

`ROWS` vs `RANGE`

`ROWS` counts physical rows. `RANGE` groups peer rows that share the same `ORDER BY` value.

Use `ROWS` when you want strict row-by-row stepping. Use `RANGE` when tied sort values should be treated as one peer group.

Aggregate Functions Using Frames

See list here.

Examples

Take the following example:

```
CREATE TABLE `student_test` (
  name char(10),
  test char(10),
  score tinyint(4)
);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, SUM(score)
  OVER () AS total_score
  FROM student_test;
```

```
+---------+--------+-------+-------------+
| name    | test   | score | total_score |
+---------+--------+-------+-------------+
| Chun    | SQL    |    75 |         453 |
| Chun    | Tuning |    73 |         453 |
| Esben   | SQL    |    43 |         453 |
| Esben   | Tuning |    31 |         453 |
| Kaolin  | SQL    |    56 |         453 |
| Kaolin  | Tuning |    88 |         453 |
| Tatiana | SQL    |    87 |         453 |
+---------+--------+-------+-------------+
```

By not specifying an `ORDER BY` clause, SUM runs over the entire result set. If we add `ORDER BY score`, the default running frame is used instead:

```
SELECT name, test, score, SUM(score)
  OVER (ORDER BY score) AS total_score
  FROM student_test ORDER BY score;
```

```
+---------+--------+-------+-------------+
| name    | test   | score | total_score |
+---------+--------+-------+-------------+
| Esben   | Tuning |    31 |          31 |
| Esben   | SQL    |    43 |          74 |
| Kaolin  | SQL    |    56 |         130 |
| Chun    | Tuning |    73 |         203 |
| Chun    | SQL    |    75 |         278 |
| Tatiana | SQL    |    87 |         365 |
| Kaolin  | Tuning |    88 |         453 |
+---------+--------+-------+-------------+
```

The `total_score` column now represents a running total of the current row and all previous rows. The frame expands as the query proceeds.

The previous query relies on the default frame. Written explicitly, it looks like this:

```
SELECT name, test, score, SUM(score)
  OVER (ORDER BY score RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS total_score
  FROM student_test ORDER BY score;
```

```
+---------+--------+-------+-------------+
| name    | test   | score | total_score |
+---------+--------+-------+-------------+
| Esben   | Tuning |    31 |          31 |
| Esben   | SQL    |    43 |          74 |
| Kaolin  | SQL    |    56 |         130 |
| Chun    | Tuning |    73 |         203 |
| Chun    | SQL    |    75 |         278 |
| Tatiana | SQL    |    87 |         365 |
| Kaolin  | Tuning |    88 |         453 |
+---------+--------+-------+-------------+
```

Applying the window function to the current row and all following rows can be done with `UNBOUNDED FOLLOWING`:

```
SELECT name, test, score, SUM(score)
  OVER (ORDER BY score RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS total_score
  FROM student_test ORDER BY score;
```

```
+---------+--------+-------+-------------+
| name    | test   | score | total_score |
+---------+--------+-------+-------------+
| Esben   | Tuning |    31 |         453 |
| Esben   | SQL    |    43 |         422 |
| Kaolin  | SQL    |    56 |         379 |
| Chun    | Tuning |    73 |         323 |
| Chun    | SQL    |    75 |         250 |
| Tatiana | SQL    |    87 |         175 |
| Kaolin  | Tuning |    88 |          88 |
+---------+--------+-------+-------------+
```

You can also specify a fixed number of rows instead of an unbounded frame. The following example uses the current row and the previous row:

```
SELECT name, test, score, SUM(score)
  OVER (ORDER BY score ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) AS total_score
  FROM student_test ORDER BY score;
```

```
+---------+--------+-------+-------------+
| name    | test   | score | total_score |
+---------+--------+-------+-------------+
| Esben   | Tuning |    31 |          31 |
| Esben   | SQL    |    43 |          74 |
| Kaolin  | SQL    |    56 |          99 |
| Chun    | Tuning |    73 |         129 |
| Chun    | SQL    |    75 |         148 |
| Tatiana | SQL    |    87 |         162 |
| Kaolin  | Tuning |    88 |         175 |
+---------+--------+-------+-------------+
```

The next example uses the previous row, the current row, and the following row:

```
SELECT name, test, score, SUM(score)
  OVER (ORDER BY score ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS total_score
  FROM student_test ORDER BY score;
```

```
+---------+--------+-------+-------------+
| name    | test   | score | total_score |
+---------+--------+-------+-------------+
| Esben   | Tuning |    31 |          74 |
| Esben   | SQL    |    43 |         130 |
| Kaolin  | SQL    |    56 |         172 |
| Chun    | Tuning |    73 |         204 |
| Chun    | SQL    |    75 |         235 |
| Tatiana | SQL    |    87 |         250 |
| Kaolin  | Tuning |    88 |         175 |
+---------+--------+-------+-------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ColumnStore Window Functions

Introduction

MariaDB ColumnStore provides support for window functions broadly following the SQL 2003 specification. A window function allows for calculations relating to a window of data surrounding the current row in a result set. This capability provides for simplified queries in support of common business questions such as cumulative totals, rolling averages, and top 10 lists.

Aggregate functions are utilized for window functions however differ in behavior from a group by query because the rows remain ungrouped. This provides support for cumulative sums and rolling averages, for example.

Two key concepts for window functions are Partition and Frame:

- A Partition is a group of rows, or window, that have the same value for a specific column, for example a Partition can be created over a time period such as a quarter or lookup values.
- The Frame for each row is a subset of the row's Partition. The frame typically is dynamic allowing for a sliding frame of rows within the Partition. The Frame determines the range of rows for the windowing function. A Frame could be defined as the last X rows and next Y rows all the way up to the entire Partition.

Window functions are applied after joins, group by, and having clauses are calculated.

Syntax

A window function is applied in the select clause using the following syntax:

```
function_name ([expression [, expression ... ]]) OVER ( window_definition )
```

where *window_definition* is defined as:

```
[ PARTITION BY expression [, ...] ]
[ ORDER BY expression [ ASC | DESC ] [ NULLS { FIRST | LAST } ] [, ...] ]
[ frame_clause ]
```

`PARTITION BY`:

- Divides the window result set into groups based on one or more expressions.
- An expression may be a constant, column, and non window function expressions.
- A query is not limited to a single partition by clause. Different partition clauses can be used across different window function applications.
- The partition by columns do not need to be in the select list but do need to be available from the query result set.
- If there is no `PARTITION BY` clause, all rows of the result set define the group.

`ORDER BY`:

- Defines the ordering of values within the partition.
- Can be ordered by multiple keys which may be a constant, column or non window function expression.
- The order by columns do not need to be in the select list but need to be available from the query result set.
- Use of a select column alias from the query is not supported.
- `ASC` (default) and `DESC` options allow for ordering ascending or descending.
- `NULLS FIRST` and `NULL_LAST` options specify whether null values come first or last in the ordering sequence. `NULLS_FIRST` is the default for ASC order, and `NULLS_LAST` is the default for `DESC` order.

and the optional *frame_clause* is defined as:

```
{ RANGE | ROWS } frame_start
{ RANGE | ROWS } BETWEEN frame_start AND frame_end
```

and the optional *frame_start* and *frame_end* are defined as (value being a numeric expression):

```
UNBOUNDED PRECEDING
value PRECEDING
CURRENT ROW
value FOLLOWING
UNBOUNDED FOLLOWING
```

RANGE/ROWS:

- Defines the windowing clause for calculating the set of rows that the function applies to for calculating a given rows window function result.
- Requires an `ORDER BY` clause to define the row order for the window.
- `ROWS` specify the window in physical units, i.e. result set rows and must be a constant or expression evaluating to a positive numeric value.
- `RANGE` specifies the window as a logical offset. If the expression evaluates to a numeric value then the `ORDER BY` expression must be a numeric or `DATE` type. If the expression evaluates to an interval value then the `ORDER BY` expression must be a `DATE` data type.
- `UNBOUNDED PRECEDING` indicates the window starts at the first row of the partition.
- `UNBOUNDED FOLLOWING` indicates the window ends at the last row of the partition.
- `CURRENT ROW` specifies the window start or ends at the current row or value.
- If omitted, the default is `ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`.

Supported Functions

Function	Description
AVG()	The average of all input values.
COUNT()	Number of input rows.
CUME_DIST()	Calculates the cumulative distribution, or relative rank, of the current row to other rows in the same partition. Number of peer or preceding rows / number of rows in partition.
DENSE_RANK()	Ranks items in a group leaving no gaps in ranking sequence when there are ties.
FIRST_VALUE()	The value evaluated at the row that is the first row of the window frame (counting from 1); null if no such row.

Function	Description
LAG()	The value evaluated at the row that is offset rows before the current row within the partition; if there is no such row, instead return default. Both offset and default are evaluated with respect to the current row. If omitted, offset defaults to 1 and default to null. LAG provides access to more than one row of a table at the same time without a self-join. Given a series of rows returned from a query and a position of the cursor, LAG provides access to a row at a given physical offset prior to that position.
LAST_VALUE()	The value evaluated at the row that is the last row of the window frame (counting from 1); null if no such row.
LEAD()	Provides access to a row at a given physical offset beyond that position. Returns value evaluated at the row that is offset rows after the current row within the partition; if there is no such row, instead return default. Both offset and default are evaluated with respect to the current row. If omitted, offset defaults to 1 and default to null.
MAX()	Maximum value of expression across all input values.
MEDIAN()	An inverse distribution function that assumes a continuous distribution model. It takes a numeric or datetime value and returns the middle value or an interpolated value that would be the middle value once the values are sorted. Nulls are ignored in the calculation.
MIN()	Minimum value of expression across all input values.
NTH_VALUE()	The value evaluated at the row that is the nth row of the window frame (counting from 1); null if no such row.
NTILE()	Divides an ordered data set into a number of buckets indicated by expr and assigns the appropriate bucket number to each row. The buckets are numbered 1 through expr. The expr value must resolve to a positive constant for each partition. Integer ranging from 1 to the argument value, dividing the partition as equally as possible.
PERCENT_RANK()	relative rank of the current row: (rank - 1) / (total rows - 1).
PERCENTILE_CONT()	An inverse distribution function that assumes a continuous distribution model. It takes a percentile value and a sort specification, and returns an interpolated value that would fall into that percentile value with respect to the sort specification. Nulls are ignored in the calculation.
PERCENTILE_DISC()	An inverse distribution function that assumes a discrete distribution model. It takes a percentile value and a sort specification and returns an element from the set. Nulls are ignored in the calculation.
RANK()	rank of the current row with gaps; same as row_number of its first peer.
ROW_NUMBER()	number of the current row within its partition, counting from 1
STDDEV() STDDEV_POP()	Computes the population standard deviation and returns the square root of the population variance.
STDDEV_SAMP()	Computes the cumulative sample standard deviation and returns the square root of the sample variance.
SUM()	Sum of expression across all input values.
VARIANCE() VAR_POP()	Population variance of the input values (square of the population standard deviation).
VAR_SAMP()	Sample variance of the input values (square of the sample standard deviation).

Examples

Example Schema

The examples are all based on the following simplified sales opportunity table:

```
CREATE TABLE opportunities (
id INT,
accountName VARCHAR(20),
name VARCHAR(128),
owner VARCHAR(7),
amount DECIMAL(10,2),
closeDate DATE,
stageName VARCHAR(11)
) ENGINE=columnstore;
```

Some example values are (thanks to www.mockaroo.com for sample data generation):

id	accountName	name	owner	amount	closeDate	stageName
1	Browseblab	Multi-lateral executive function	Bob	26444.86	2016-10-20	Negotiating
2	Mita	Organic demand-driven benchmark	Maria	477878.41	2016-11-28	ClosedWon
3	Miboo	De-engineered hybrid groupware	Olivier	80181.78	2017-01-05	ClosedWon
4	Youbridge	Enterprise-wide bottom-line Graphic Interface	Chris	946245.29	2016-07-02	ClosedWon
5	Skyba	Reverse-engineered fresh-thinking standardization	Maria	696241.82	2017-02-17	Negotiating
6	Eayo	Fundamental well-modulated artificial intelligence	Bob	765605.52	2016-08-27	Prospecting
7	Yotz	Extended secondary infrastructure	Chris	319624.20	2017-01-06	ClosedLost
8	Oloo	Configurable web-enabled data-warehouse	Chris	321016.26	2017-03-08	ClosedLost
9	Kaymbo	Multi-lateral web-enabled definition	Bob	690881.01	2017-01-02	Developing
10	Rhyloo	Public-key coherent infrastructure	Chris	965477.74	2016-11-07	Prospecting

The schema, sample data, and queries are available as an attachment to this article.

Cumulative Sum and Running Max Example

Window functions can be used to achieve cumulative / running calculations on a detail report. In this case a won opportunity report for a 7 day period adds columns to show the accumulated won amount as well as the current highest opportunity amount in preceding rows.

```
SELECT owner,
accountName,
CloseDate,
amount,
SUM(amount) OVER (ORDER BY CloseDate ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) cumeWon,
MAX(amount) OVER (ORDER BY CloseDate ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) runningMax
FROM opportunities
WHERE stageName='ClosedWon'
AND closeDate >= '2016-10-02' AND closeDate <= '2016-10-09'
ORDER BY CloseDate;
```

with example results:

owner	accountName	CloseDate	amount	cumeWon	runningMax
Bill	Babbleopia	2016-10-02	437636.47	437636.47	437636.47
Bill	Thoughtworks	2016-10-04	146086.51	583722.98	437636.47
Olivier	Devpulse	2016-10-05	834235.93	1417958.91	834235.93

owner	accountName	CloseDate	amount	cumeWon	runningMax
Chris	Linkbridge	2016-10-07	539977.45	2458738.65	834235.93
Olivier	Trupe	2016-10-07	500802.29	1918761.20	834235.93
Bill	Latz	2016-10-08	857254.87	3315993.52	857254.87
Chris	Avamm	2016-10-09	699566.86	4015560.38	857254.87

Partitioned Cumulative Sum and Running Max Example

The above example can be partitioned, so that the window functions are over a particular field grouping such as owner and accumulate within that grouping. This is achieved by adding the syntax “partition by” in the window function clause.

```
SELECT owner,
accountName,
CloseDate,
amount,
SUM(amount) OVER (PARTITION BY owner ORDER BY CloseDate ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
cumeWon,
MAX(amount) OVER (PARTITION BY owner ORDER BY CloseDate ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
runningMax
FROM opportunities
WHERE stageName='ClosedWon'
AND closeDate >= '2016-10-02' AND closeDate <= '2016-10-09'
ORDER BY owner, CloseDate;
```

With example results:

owner	accountName	CloseDate	amount	cumeWon	runningMax
Bill	Babbleopia	2016-10-02	437636.47	437636.47	437636.47
Bill	Thoughtworks	2016-10-04	146086.51	583722.98	437636.47
Bill	Latz	2016-10-08	857254.87	1440977.85	857254.87
Chris	Linkbridge	2016-10-07	539977.45	539977.45	539977.45
Chris	Avamm	2016-10-09	699566.86	1239544.31	699566.86
Olivier	Devpulse	2016-10-05	834235.93	834235.93	834235.93
Olivier	Trupe	2016-10-07	500802.29	1335038.22	834235.93

Ranking / Top Results

The rank window function allows for ranking or assigning a numeric order value based on the window function definition. Using the `Rank()` function will result in the same value for ties / equal values and the next rank value skipped. The `Dense_Rank()` function behaves similarly except the next consecutive number is used after a tie rather than skipped. The `Row_Number()` function will provide a unique ordering value. The example query shows the `Rank()` function being applied to rank sales reps by the number of opportunities for Q4 2016.

```
SELECT owner,
wonCount,
rank() OVER (ORDER BY wonCount DESC) rank
FROM (
  SELECT owner,
  COUNT(*) wonCount
  FROM opportunities
  WHERE stageName='ClosedWon'
  AND closeDate >= '2016-10-01' AND closeDate < '2016-12-31'
  GROUP BY owner
) t
ORDER BY rank;
```

with example results (note the query is technically incorrect by using closeDate < '2016-12-31' however this creates a tie scenario for illustrative purposes):

owner	wonCount	rank
Bill	19	1
Chris	15	2
Maria	14	3
Bob	14	3
Olivier	10	5

If the `dense_rank` function is used the rank values would be 1,2,3,3,4 and for the `row_number` function the values would be 1,2,3,4,5.

First and Last Values

The first_value and last_value functions allow determining the first and last values of a given range. Combined with a group by this allows summarizing opening and closing values. The example shows a more complex case where detailed information is presented for first and last opportunity by quarter.

```
SELECT a.YEAR,
a.quarter,
f.accountName firstAccountName,
f.owner firstOwner,
f.amount firstAmount,
l.accountName lastAccountName,
l.owner lastOwner,
l.amount lastAmount
FROM (
  SELECT YEAR,
  QUARTER,
  MIN(firstId) firstId,
  MIN(lastId) lastId
  FROM (
    SELECT YEAR(closeDate) YEAR,
    quarter(closeDate) QUARTER,
    first_value(id) OVER (PARTITION BY YEAR(closeDate), quarter(closeDate) ORDER BY closeDate ROWS
BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) firstId,
    last_value(id) OVER (PARTITION BY YEAR(closeDate), quarter(closeDate) ORDER BY closeDate ROWS BETWEEN
CURRENT ROW AND UNBOUNDED FOLLOWING) lastId
    FROM opportunities  WHERE stageName='ClosedWon'
  ) t
  GROUP BY YEAR, QUARTER ORDER BY YEAR,QUARTER
) a
JOIN opportunities f ON a.firstId = f.id
JOIN opportunities l ON a.lastId = l.id
ORDER BY YEAR, QUARTER;
```

with example results:

year	quarter	firstAccountName	firstOwner	firstAmount	lastAccountName	lastOwner	lastAmount
2016	3	Skidoo	Bill	523295.07	Skipstorm	Bill	151420.86
2016	4	Skimia	Chris	961513.59	Avamm	Maria	112493.65
2017	1	Yombu	Bob	536875.51	Skaboo	Chris	270273.08

Prior and Next Example

Sometimes it useful to understand the previous and next values in the context of a given row. The lag and lead window functions provide this capability. By default the offset is one providing the prior or next value but can also be provided to get a larger offset. The example query is a report of opportunities by account name showing the opportunity amount, and the prior and next opportunity amount for that account by close date.

```
SELECT accountName,
closeDate,
amount currentOppAmount,
lag(amount) OVER (PARTITION BY accountName ORDER BY closeDate) priorAmount, lead(amount) OVER (PARTITION
BY accountName ORDER BY closeDate) nextAmount
FROM opportunities
ORDER BY accountName, closeDate
LIMIT 9;
```

With example results:

accountName	closeDate	currentOppAmount	priorAmount	nextAmount
Abata	2016-09-10	645098.45	NULL	161086.82
Abata	2016-10-14	161086.82	645098.45	350235.75
Abata	2016-12-18	350235.75	161086.82	878595.89
Abata	2016-12-31	878595.89	350235.75	922322.39
Abata	2017-01-21	922322.39	878595.89	NULL
Abatz	2016-10-19	795424.15	NULL	NULL
Agimba	2016-07-09	288974.84	NULL	914461.49
Agimba	2016-09-07	914461.49	288974.84	176645.52
Agimba	2016-09-20	176645.52	914461.49	NULL

Quartiles Example

The NTile window function allows for breaking up a data set into portions assigned a numeric value to each portion of the range. NTile(4) breaks the data up into quartiles (4 sets). The example query produces a report of all opportunities summarizing the quartile boundaries of amount values.

```
SELECT t.quartile,
MIN(t.amount) MIN,
MAX(t.amount) MAX
FROM (
  SELECT amount,
  ntile(4) OVER (ORDER BY amount ASC) quartile
  FROM opportunities
  WHERE closeDate >= '2016-10-01' AND closeDate <= '2016-12-31'
  ) t
GROUP BY quartile
ORDER BY quartile;
```

With example results:

quartile	min	max
1	6337.15	287634.01
2	288796.14	539977.45
3	540070.04	748727.51
4	753670.77	998864.47

Percentile Example

The percentile functions have a slightly different syntax from other window functions as can be seen in the example below. These functions can be only applied against numeric values. The argument to the function is the percentile to evaluate. Following ‘within group’ is the sort expression which indicates the sort column and optionally order. Finally after ‘over’ is an optional partition by clause, for no partition clause use ‘over ()’. The example below utilizes the value 0.5 to calculate the median opportunity amount in the rows. The values differ sometimes because percentile_cont will return the average of the 2 middle rows for an even data set while percentile_desc returns the first encountered in the sort.

```
SELECT owner,
accountName,
CloseDate,
amount,
percentile_cont(0.5) within GROUP (ORDER BY amount) OVER (PARTITION BY owner) pct_cont,
percentile_disc(0.5) within GROUP (ORDER BY amount) OVER (PARTITION BY owner) pct_disc
FROM opportunities
WHERE stageName='ClosedWon'
AND closeDate >= '2016-10-02' AND closeDate <= '2016-10-09'
ORDER BY owner, CloseDate;
```

With example results:

owner	accountName	CloseDate	amount	pct_cont	pct_disc
Bill	Babbleopia	2016-10-02	437636.47	437636.4700000000	437636.47
Bill	Thoughtworks	2016-10-04	146086.51	437636.4700000000	437636.47
Bill	Latz	2016-10-08	857254.87	437636.4700000000	437636.47
Chris	Linkbridge	2016-10-07	539977.45	619772.1550000000	539977.45
Chris	Avamm	2016-10-09	699566.86	619772.1550000000	539977.45
Olivier	Devpulse	2016-10-05	834235.93	667519.1100000000	500802.29
Olivier	Trupe	2016-10-07	500802.29	667519.1100000000	500802.29

This page is licensed: CC BY-SA / Gnu FDL

CUME_DIST

Syntax

```
CUME_DIST() OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```

Description

`CUME_DIST()` is a window function that returns the cumulative distribution of a given row. The following formula is used to calculate the value:

```
(number of rows <= current row) / (total rows)
```

Examples

```
create table t1 (
  pk int primary key,
  a int,
  b int
);

insert into t1 values
( 1 , 0, 10),
( 2 , 0, 10),
( 3 , 1, 10),
( 4 , 1, 10),
( 8 , 2, 10),
( 5 , 2, 20),
( 6 , 2, 20),
( 7 , 2, 20),
( 9 , 4, 20),
(10 , 4, 20);

select pk, a, b,
    rank() over (order by a) as rank,
    percent_rank() over (order by a) as pct_rank,
    cume_dist() over (order by a) as cume_dist
from t1;
+----+------+------+------+--------------+--------------+
| pk | a    | b    | rank | pct_rank     | cume_dist    |
+----+------+------+------+--------------+--------------+
|  1 |    0 |   10 |    1 | 0.0000000000 | 0.2000000000 |
|  2 |    0 |   10 |    1 | 0.0000000000 | 0.2000000000 |
|  3 |    1 |   10 |    3 | 0.2222222222 | 0.4000000000 |
|  4 |    1 |   10 |    3 | 0.2222222222 | 0.4000000000 |
|  5 |    2 |   20 |    5 | 0.4444444444 | 0.8000000000 |
|  6 |    2 |   20 |    5 | 0.4444444444 | 0.8000000000 |
|  7 |    2 |   20 |    5 | 0.4444444444 | 0.8000000000 |
|  8 |    2 |   10 |    5 | 0.4444444444 | 0.8000000000 |
|  9 |    4 |   20 |    9 | 0.8888888889 | 1.0000000000 |
| 10 |    4 |   20 |    9 | 0.8888888889 | 1.0000000000 |
+----+------+------+------+--------------+--------------+

select pk, a, b,
       percent_rank() over (order by pk) as pct_rank,
       cume_dist() over (order by pk) as cume_dist
from t1 order by pk;
+----+------+------+--------------+--------------+
| pk | a    | b    | pct_rank     | cume_dist    |
+----+------+------+--------------+--------------+
|  1 |    0 |   10 | 0.0000000000 | 0.1000000000 |
|  2 |    0 |   10 | 0.1111111111 | 0.2000000000 |
|  3 |    1 |   10 | 0.2222222222 | 0.3000000000 |
|  4 |    1 |   10 | 0.3333333333 | 0.4000000000 |
|  5 |    2 |   20 | 0.4444444444 | 0.5000000000 |
|  6 |    2 |   20 | 0.5555555556 | 0.6000000000 |
|  7 |    2 |   20 | 0.6666666667 | 0.7000000000 |
|  8 |    2 |   10 | 0.7777777778 | 0.8000000000 |
|  9 |    4 |   20 | 0.8888888889 | 0.9000000000 |
| 10 |    4 |   20 | 1.0000000000 | 1.0000000000 |
+----+------+------+--------------+--------------+

select pk, a, b,
        percent_rank() over (partition by a order by a) as pct_rank,
        cume_dist() over (partition by a order by a) as cume_dist
from t1;
+----+------+------+--------------+--------------+
| pk | a    | b    | pct_rank     | cume_dist    |
+----+------+------+--------------+--------------+
|  1 |    0 |   10 | 0.0000000000 | 1.0000000000 |
|  2 |    0 |   10 | 0.0000000000 | 1.0000000000 |
|  3 |    1 |   10 | 0.0000000000 | 1.0000000000 |
```

```
|  4 |    1 |   10 | 0.0000000000 | 1.0000000000 |
|  5 |    2 |   20 | 0.0000000000 | 1.0000000000 |
|  6 |    2 |   20 | 0.0000000000 | 1.0000000000 |
|  7 |    2 |   20 | 0.0000000000 | 1.0000000000 |
|  8 |    2 |   10 | 0.0000000000 | 1.0000000000 |
|  9 |    4 |   20 | 0.0000000000 | 1.0000000000 |
| 10 |    4 |   20 | 0.0000000000 | 1.0000000000 |
+----+------+------+--------------+--------------+
```

See Also

- PERCENT_RANK()

This page is licensed: CC BY-SA / Gnu FDL

DENSE_RANK

Syntax

```
DENSE_RANK() OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```

Description

`DENSE_RANK()` is a window function that displays the number of a given row, starting at one and following the ORDER BY sequence of the window function, with identical values receiving the same result. Unlike the RANK() function, there are no skipped values if the preceding results are identical. It is also similar to the ROW_NUMBER() function except that in that function, identical values will receive a different row number for each result.

Examples

The distinction between `DENSE_RANK()`, RANK() and ROW_NUMBER():

```
CREATE TABLE student(course VARCHAR(10), mark int, name varchar(10));

INSERT INTO student VALUES
  ('Maths', 60, 'Thulile'),
  ('Maths', 60, 'Pritha'),
  ('Maths', 70, 'Voitto'),
  ('Maths', 55, 'Chun'),
  ('Biology', 60, 'Bilal'),
   ('Biology', 70, 'Roger');

SELECT
  RANK() OVER (PARTITION BY course ORDER BY mark DESC) AS rank,
  DENSE_RANK() OVER (PARTITION BY course ORDER BY mark DESC) AS dense_rank,
  ROW_NUMBER() OVER (PARTITION BY course ORDER BY mark DESC) AS row_num,
  course, mark, name
FROM student ORDER BY course, mark DESC;
+------+------------+---------+---------+------+---------+
| rank | dense_rank | row_num | course  | mark | name    |
+------+------------+---------+---------+------+---------+
|    1 |          1 |       1 | Biology |   70 | Roger   |
|    2 |          2 |       2 | Biology |   60 | Bilal   |
|    1 |          1 |       1 | Maths   |   70 | Voitto  |
|    2 |          2 |       2 | Maths   |   60 | Thulile |
|    2 |          2 |       3 | Maths   |   60 | Pritha  |
|    4 |          3 |       4 | Maths   |   55 | Chun    |
+------+------------+---------+---------+------+---------+
```

See Also

- RANK()
- ROW_NUMBER()
- ORDER BY

This page is licensed: CC BY-SA / Gnu FDL

FIRST_VALUE

Syntax

```
FIRST_VALUE(expr) OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```

Description

`FIRST_VALUE` returns the first result from an ordered set, or NULL if no such result exists.

Examples

```
CREATE TABLE t1 (
  pk int primary key,
  a int,
  b int,
  c char(10),
  d decimal(10, 3),
  e real
);

INSERT INTO t1 VALUES
( 1, 0, 1,    'one',    0.1,  0.001),
( 2, 0, 2,    'two',    0.2,  0.002),
( 3, 0, 3,    'three',  0.3,  0.003),
( 4, 1, 2,    'three',  0.4,  0.004),
( 5, 1, 1,    'two',    0.5,  0.005),
( 6, 1, 1,    'one',    0.6,  0.006),
( 7, 2, NULL, 'n_one',  0.5,  0.007),
( 8, 2, 1,    'n_two',  NULL, 0.008),
( 9, 2, 2,    NULL,     0.7,  0.009),
(10, 2, 0,    'n_four', 0.8,  0.010),
(11, 2, 10,   NULL,     0.9,  NULL);

SELECT pk, FIRST_VALUE(pk) OVER (ORDER BY pk) AS first_asc,
           LAST_VALUE(pk) OVER (ORDER BY pk) AS last_asc,
           FIRST_VALUE(pk) OVER (ORDER BY pk DESC) AS first_desc,
           LAST_VALUE(pk) OVER (ORDER BY pk DESC) AS last_desc
FROM t1
ORDER BY pk DESC;

+----+-----------+----------+------------+-----------+
| pk | first_asc | last_asc | first_desc | last_desc |
+----+-----------+----------+------------+-----------+
| 11 |         1 |       11 |         11 |        11 |
| 10 |         1 |       10 |         11 |        10 |
|  9 |         1 |        9 |         11 |         9 |
|  8 |         1 |        8 |         11 |         8 |
|  7 |         1 |        7 |         11 |         7 |
|  6 |         1 |        6 |         11 |         6 |
|  5 |         1 |        5 |         11 |         5 |
|  4 |         1 |        4 |         11 |         4 |
|  3 |         1 |        3 |         11 |         3 |
|  2 |         1 |        2 |         11 |         2 |
|  1 |         1 |        1 |         11 |         1 |
+----+-----------+----------+------------+-----------+
```

```
CREATE OR REPLACE TABLE t1 (i int);
INSERT INTO t1 VALUES (1),(2),(3),(4),(5),(6),(7),(8),(9),(10);

SELECT i,
  FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN CURRENT ROW and 1 FOLLOWING) AS f_1f,
  LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN CURRENT ROW and 1 FOLLOWING) AS l_1f,
  FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS f_1p1f,
  LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS l_1p1f,
  FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 2 PRECEDING AND 1 PRECEDING) AS f_2p1p,
  LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 2 PRECEDING AND 1 PRECEDING) AS l_2p1p,
  FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 FOLLOWING AND 2 FOLLOWING) AS f_1f2f,
  LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 FOLLOWING AND 2 FOLLOWING) AS l_1f2f
FROM t1;

+------+------+------+--------+--------+--------+--------+--------+--------+
| i    | f_1f | l_1f | f_1p1f | l_1p1f | f_2p1p | l_2p1p | f_1f2f | l_1f2f |
+------+------+------+--------+--------+--------+--------+--------+--------+
|    1 |    1 |    2 |      1 |      2 |   NULL |   NULL |      2 |      3 |
|    2 |    2 |    3 |      1 |      3 |      1 |      1 |      3 |      4 |
|    3 |    3 |    4 |      2 |      4 |      1 |      2 |      4 |      5 |
|    4 |    4 |    5 |      3 |      5 |      2 |      3 |      5 |      6 |
|    5 |    5 |    6 |      4 |      6 |      3 |      4 |      6 |      7 |
|    6 |    6 |    7 |      5 |      7 |      4 |      5 |      7 |      8 |
|    7 |    7 |    8 |      6 |      8 |      5 |      6 |      8 |      9 |
|    8 |    8 |    9 |      7 |      9 |      6 |      7 |      9 |     10 |
|    9 |    9 |   10 |      8 |     10 |      7 |      8 |     10 |     10 |
|   10 |   10 |   10 |      9 |     10 |      8 |      9 |   NULL |   NULL |
+------+------+------+--------+--------+--------+--------+--------+--------+
```

See Also

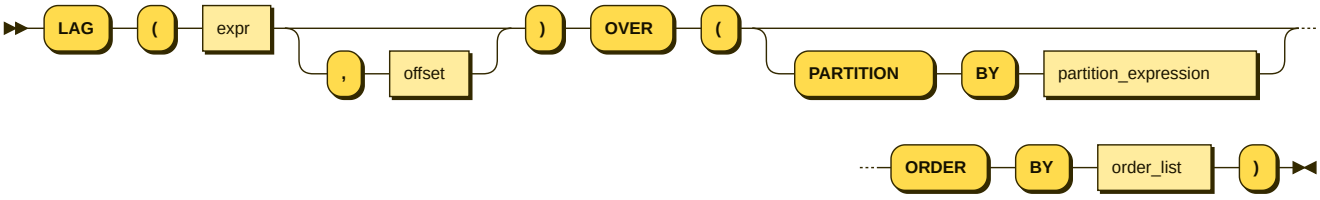
- LAST_VALUE

This page is licensed: CC BY-SA / Gnu FDL

LAG

Syntax

```
LAG (expr[, offset]) OVER (
  [ PARTITION BY partition_expression ]
  < ORDER BY order_list >
)
```



Railroad diagram of LAG — equivalent to the BNF above

Description

The `LAG` function accesses data from a previous row according to the `ORDER BY` clause without the need for a self-join. The specific row is determined by the *offset* (default *1*), which specifies the number of rows behind the current row to use. An offset of *0* is the current row.

Examples

```
CREATE TABLE t1 (pk int primary key, a int, b int, c char(10), d decimal(10, 3), e real);

INSERT INTO t1 VALUES
 ( 1, 0, 1,    'one',    0.1,  0.001),
 ( 2, 0, 2,    'two',    0.2,  0.002),
 ( 3, 0, 3,    'three',  0.3,  0.003),
 ( 4, 1, 2,    'three',  0.4,  0.004),
 ( 5, 1, 1,    'two',    0.5,  0.005),
 ( 6, 1, 1,    'one',    0.6,  0.006),
 ( 7, 2, NULL, 'n_one',  0.5,  0.007),
 ( 8, 2, 1,    'n_two',  NULL, 0.008),
 ( 9, 2, 2,    NULL,     0.7,  0.009),
 (10, 2, 0,    'n_four', 0.8,  0.010),
 (11, 2, 10,   NULL,     0.9,  NULL);

SELECT pk, LAG(pk) OVER (ORDER BY pk) AS l,
  LAG(pk,1) OVER (ORDER BY pk) AS l1,
  LAG(pk,2) OVER (ORDER BY pk) AS l2,
  LAG(pk,0) OVER (ORDER BY pk) AS l0,
  LAG(pk,-1) OVER (ORDER BY pk) AS lm1,
  LAG(pk,-2) OVER (ORDER BY pk) AS lm2
FROM t1;
+----+------+------+------+------+------+------+
| pk | l    | l1   | l2   | l0   | lm1  | lm2  |
+----+------+------+------+------+------+------+
|  1 | NULL | NULL | NULL |    1 |    2 |    3 |
|  2 |    1 |    1 | NULL |    2 |    3 |    4 |
|  3 |    2 |    2 |    1 |    3 |    4 |    5 |
|  4 |    3 |    3 |    2 |    4 |    5 |    6 |
|  5 |    4 |    4 |    3 |    5 |    6 |    7 |
|  6 |    5 |    5 |    4 |    6 |    7 |    8 |
|  7 |    6 |    6 |    5 |    7 |    8 |    9 |
|  8 |    7 |    7 |    6 |    8 |    9 |   10 |
|  9 |    8 |    8 |    7 |    9 |   10 |   11 |
| 10 |    9 |    9 |    8 |   10 |   11 | NULL |
| 11 |   10 |   10 |    9 |   11 | NULL | NULL |
+----+------+------+------+------+------+------+
```

See Also

- LEAD - Window function to access a following row

This page is licensed: CC BY-SA / Gnu FDL

LEAD

Syntax

```
LEAD (expr[, offset]) OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```

Description

The `LEAD` function accesses data from a following row in the same result set without the need for a self-join. The specific row is determined by the *offset* (default *1*), which specifies the number of rows ahead the current row to use. An offset of *0* is the current row.

Example

```
CREATE TABLE t1 (pk int primary key, a int, b int, c char(10), d decimal(10, 3), e real);

INSERT INTO t1 VALUES
 ( 1, 0, 1,    'one',    0.1,  0.001),
 ( 2, 0, 2,    'two',    0.2,  0.002),
 ( 3, 0, 3,    'three',  0.3,  0.003),
 ( 4, 1, 2,    'three',  0.4,  0.004),
 ( 5, 1, 1,    'two',    0.5,  0.005),
 ( 6, 1, 1,    'one',    0.6,  0.006),
 ( 7, 2, NULL, 'n_one',  0.5,  0.007),
 ( 8, 2, 1,    'n_two',  NULL, 0.008),
 ( 9, 2, 2,    NULL,     0.7,  0.009),
 (10, 2, 0,    'n_four', 0.8,  0.010),
 (11, 2, 10,   NULL,     0.9,  NULL);

SELECT pk, LEAD(pk) OVER (ORDER BY pk) AS l,
  LEAD(pk,1) OVER (ORDER BY pk) AS l1,
  LEAD(pk,2) OVER (ORDER BY pk) AS l2,
  LEAD(pk,0) OVER (ORDER BY pk) AS l0,
  LEAD(pk,-1) OVER (ORDER BY pk) AS lm1,
  LEAD(pk,-2) OVER (ORDER BY pk) AS lm2
FROM t1;
+----+------+------+------+------+------+------+
| pk | l    | l1   | l2   | l0   | lm1  | lm2  |
+----+------+------+------+------+------+------+
|  1 |    2 |    2 |    3 |    1 | NULL | NULL |
|  2 |    3 |    3 |    4 |    2 |    1 | NULL |
|  3 |    4 |    4 |    5 |    3 |    2 |    1 |
|  4 |    5 |    5 |    6 |    4 |    3 |    2 |
|  5 |    6 |    6 |    7 |    5 |    4 |    3 |
|  6 |    7 |    7 |    8 |    6 |    5 |    4 |
|  7 |    8 |    8 |    9 |    7 |    6 |    5 |
|  8 |    9 |    9 |   10 |    8 |    7 |    6 |
|  9 |   10 |   10 |   11 |    9 |    8 |    7 |
| 10 |   11 |   11 | NULL |   10 |    9 |    8 |
| 11 | NULL | NULL | NULL |   11 |   10 |    9 |
+----+------+------+------+------+------+------+
```

See Also

- LAG - Window function to access a previous row

This page is licensed: CC BY-SA / Gnu FDL

MEDIAN

Syntax

```
MEDIAN(median expression) OVER (
  [ PARTITION BY partition_expression ]
)
```

Description

`MEDIAN()` is a window function that returns the median value of a range of values.

It is a specific case of PERCENTILE_CONT, with an argument of 0.5 and the ORDER BY column the one in `MEDIAN`'s argument.

```
MEDIAN(<median-arg>) OVER ( [ PARTITION BY partition_expression] )
```

Is equivalent to:

```
PERCENTILE_CONT(0.5) WITHIN
  GROUP (ORDER BY <median-arg>) OVER ( [ PARTITION BY partition_expression ])
```

Examples

```
CREATE TABLE book_rating (name CHAR(30), star_rating TINYINT);

INSERT INTO book_rating VALUES ('Lord of the Ladybirds', 5);
INSERT INTO book_rating VALUES ('Lord of the Ladybirds', 3);
INSERT INTO book_rating VALUES ('Lady of the Flies', 1);
INSERT INTO book_rating VALUES ('Lady of the Flies', 2);
INSERT INTO book_rating VALUES ('Lady of the Flies', 5);

SELECT name, median(star_rating) OVER (PARTITION BY name) FROM book_rating;
+----------------------+-------------------------------------------+
| name                 | median(star_rating) OVER (PARTITION BY name) |
+----------------------+-------------------------------------------+
| Lord of the Ladybirds |                                4.0000000000 |
| Lord of the Ladybirds |                                4.0000000000 |
| Lady of the Flies     |                                2.0000000000 |
| Lady of the Flies     |                                2.0000000000 |
| Lady of the Flies     |                                2.0000000000 |
+----------------------+-------------------------------------------+
```

See Also

- PERCENTILE_CONT

This page is licensed: CC BY-SA / Gnu FDL

NTH_VALUE

Syntax

```
NTH_VALUE (expr[, num_row]) OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```

Description

The `NTH_VALUE` function returns the value evaluated at row number `num_row` of the window frame, starting from 1, or `NULL` if the row does not exist.

This page is licensed: CC BY-SA / Gnu FDL

NTILE

Syntax

```
NTILE (expr) OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```

Description

`NTILE()` is a window function that returns an integer indicating which group a given row falls into. The number of groups is specified in the argument (*expr*), starting at one. Ordered rows in the partition are divided into the specified number of groups with as equal a size as possible.

Examples

```
create table t1 (
    pk int primary key,
    a int,
    b int
  );

insert into t1 values
    (11 , 0, 10),
    (12 , 0, 10),
    (13 , 1, 10),
    (14 , 1, 10),
    (18 , 2, 10),
    (15 , 2, 20),
    (16 , 2, 20),
    (17 , 2, 20),
    (19 , 4, 20),
    (20 , 4, 20);

select pk, a, b,
    ntile(1) over (order by pk)
  from t1;
+----+------+------+-----------------------------+
| pk | a    | b    | ntile(1) over (order by pk) |
+----+------+------+-----------------------------+
| 11 |    0 |   10 |                           1 |
| 12 |    0 |   10 |                           1 |
| 13 |    1 |   10 |                           1 |
| 14 |    1 |   10 |                           1 |
| 15 |    2 |   20 |                           1 |
| 16 |    2 |   20 |                           1 |
| 17 |    2 |   20 |                           1 |
| 18 |    2 |   10 |                           1 |
| 19 |    4 |   20 |                           1 |
| 20 |    4 |   20 |                           1 |
+----+------+------+-----------------------------+

select pk, a, b,
    ntile(4) over (order by pk)
 from t1;
+----+------+------+-----------------------------+
| pk | a    | b    | ntile(4) over (order by pk) |
+----+------+------+-----------------------------+
| 11 |    0 |   10 |                           1 |
| 12 |    0 |   10 |                           1 |
| 13 |    1 |   10 |                           1 |
| 14 |    1 |   10 |                           2 |
| 15 |    2 |   20 |                           2 |
| 16 |    2 |   20 |                           2 |
| 17 |    2 |   20 |                           3 |
| 18 |    2 |   10 |                           3 |
| 19 |    4 |   20 |                           4 |
| 20 |    4 |   20 |                           4 |
+----+------+------+-----------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

PERCENT_RANK

Syntax

```
PERCENT_RANK() OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```

Description

`PERCENT_RANK()` is a window function that returns the relative percent rank of a given row. The following formula is used to calculate the percent rank:

```
(rank - 1) / (number of rows in the window or partition - 1)
```

Examples

```
create table t1 (
  pk int primary key,
  a int,
  b int
);

insert into t1 values
( 1 , 0, 10),
( 2 , 0, 10),
( 3 , 1, 10),
( 4 , 1, 10),
( 8 , 2, 10),
( 5 , 2, 20),
( 6 , 2, 20),
( 7 , 2, 20),
( 9 , 4, 20),
(10 , 4, 20);

select pk, a, b,
    rank() over (order by a) as rank,
    percent_rank() over (order by a) as pct_rank,
    cume_dist() over (order by a) as cume_dist
from t1;
+----+------+------+------+--------------+--------------+
| pk | a    | b    | rank | pct_rank     | cume_dist    |
+----+------+------+------+--------------+--------------+
|  1 |    0 |   10 |    1 | 0.0000000000 | 0.2000000000 |
|  2 |    0 |   10 |    1 | 0.0000000000 | 0.2000000000 |
|  3 |    1 |   10 |    3 | 0.2222222222 | 0.4000000000 |
|  4 |    1 |   10 |    3 | 0.2222222222 | 0.4000000000 |
|  5 |    2 |   20 |    5 | 0.4444444444 | 0.8000000000 |
|  6 |    2 |   20 |    5 | 0.4444444444 | 0.8000000000 |
|  7 |    2 |   20 |    5 | 0.4444444444 | 0.8000000000 |
|  8 |    2 |   10 |    5 | 0.4444444444 | 0.8000000000 |
|  9 |    4 |   20 |    9 | 0.8888888889 | 1.0000000000 |
| 10 |    4 |   20 |    9 | 0.8888888889 | 1.0000000000 |
+----+------+------+------+--------------+--------------+

select pk, a, b,
       percent_rank() over (order by pk) as pct_rank,
       cume_dist() over (order by pk) as cume_dist
from t1 order by pk;
+----+------+------+--------------+--------------+
| pk | a    | b    | pct_rank     | cume_dist    |
+----+------+------+--------------+--------------+
|  1 |    0 |   10 | 0.0000000000 | 0.1000000000 |
|  2 |    0 |   10 | 0.1111111111 | 0.2000000000 |
|  3 |    1 |   10 | 0.2222222222 | 0.3000000000 |
|  4 |    1 |   10 | 0.3333333333 | 0.4000000000 |
|  5 |    2 |   20 | 0.4444444444 | 0.5000000000 |
|  6 |    2 |   20 | 0.5555555556 | 0.6000000000 |
|  7 |    2 |   20 | 0.6666666667 | 0.7000000000 |
|  8 |    2 |   10 | 0.7777777778 | 0.8000000000 |
|  9 |    4 |   20 | 0.8888888889 | 0.9000000000 |
| 10 |    4 |   20 | 1.0000000000 | 1.0000000000 |
+----+------+------+--------------+--------------+

select pk, a, b,
        percent_rank() over (partition by a order by a) as pct_rank,
        cume_dist() over (partition by a order by a) as cume_dist
from t1;
+----+------+------+--------------+--------------+
| pk | a    | b    | pct_rank     | cume_dist    |
+----+------+------+--------------+--------------+
|  1 |    0 |   10 | 0.0000000000 | 1.0000000000 |
|  2 |    0 |   10 | 0.0000000000 | 1.0000000000 |
|  3 |    1 |   10 | 0.0000000000 | 1.0000000000 |
```

```
|  4 |    1 |   10 | 0.0000000000 | 1.0000000000 |
|  5 |    2 |   20 | 0.0000000000 | 1.0000000000 |
|  6 |    2 |   20 | 0.0000000000 | 1.0000000000 |
|  7 |    2 |   20 | 0.0000000000 | 1.0000000000 |
|  8 |    2 |   10 | 0.0000000000 | 1.0000000000 |
|  9 |    4 |   20 | 0.0000000000 | 1.0000000000 |
| 10 |    4 |   20 | 0.0000000000 | 1.0000000000 |
+----+------+------+--------------+--------------+
```

See Also

- CUME_DIST()

This page is licensed: CC BY-SA / Gnu FDL

PERCENTILE_CONT

Syntax

Description

`PERCENTILE_CONT()` (standing for continuous percentile) is a window function which returns a value which corresponds to the given fraction in the sort order. If required, it will interpolate between adjacent input items.

Essentially, the following process is followed to find the value to return:

- Get the number of rows in the partition, denoted by N
- RN = p*(N-1), where p denotes the argument to the PERCENTILE_CONT function
- Calculate FRN as FRN=floor(RN) and CRN as CRN=ceil(RN)
- Look up rows FRN and CRN
- If (CRN = FRN = RN) then the result is (value of expression from row at RN)
- Otherwise the result is
- (CRN - RN) * (value of expression for row at FRN) +
- (RN - FRN) * (value of expression for row at CRN)

The MEDIAN function is a specific case of `PERCENTILE_CONT`, equivalent to `PERCENTILE_CONT(0.5)`.

Examples

```
CREATE TABLE book_rating (name CHAR(30), star_rating TINYINT);

INSERT INTO book_rating VALUES ('Lord of the Ladybirds', 5);
INSERT INTO book_rating VALUES ('Lord of the Ladybirds', 3);
INSERT INTO book_rating VALUES ('Lady of the Flies', 1);
INSERT INTO book_rating VALUES ('Lady of the Flies', 2);
INSERT INTO book_rating VALUES ('Lady of the Flies', 5);

SELECT name, PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY star_rating)
  OVER (PARTITION BY name) AS pc
  FROM book_rating;
+----------------------+--------------+
| name                 | pc           |
+----------------------+--------------+
| Lord of the Ladybirds | 4.0000000000 |
| Lord of the Ladybirds | 4.0000000000 |
| Lady of the Flies     | 2.0000000000 |
| Lady of the Flies     | 2.0000000000 |
| Lady of the Flies     | 2.0000000000 |
+----------------------+--------------+

SELECT name, PERCENTILE_CONT(1) WITHIN GROUP (ORDER BY star_rating)
  OVER (PARTITION BY name) AS pc
  FROM book_rating;
+----------------------+--------------+
| name                 | pc           |
+----------------------+--------------+
| Lord of the Ladybirds | 5.0000000000 |
| Lord of the Ladybirds | 5.0000000000 |
| Lady of the Flies     | 5.0000000000 |
| Lady of the Flies     | 5.0000000000 |
| Lady of the Flies     | 5.0000000000 |
+----------------------+--------------+

SELECT name, PERCENTILE_CONT(0) WITHIN GROUP (ORDER BY star_rating)
  OVER (PARTITION BY name) AS pc
  FROM book_rating;
+----------------------+--------------+
| name                 | pc           |
+----------------------+--------------+
| Lord of the Ladybirds | 3.0000000000 |
| Lord of the Ladybirds | 3.0000000000 |
| Lady of the Flies     | 1.0000000000 |
| Lady of the Flies     | 1.0000000000 |
| Lady of the Flies     | 1.0000000000 |
+----------------------+--------------+

SELECT name, PERCENTILE_CONT(0.6) WITHIN GROUP (ORDER BY star_rating)
  OVER (PARTITION BY name) AS pc
  FROM book_rating;
+----------------------+--------------+
| name                 | pc           |
+----------------------+--------------+
| Lord of the Ladybirds | 4.2000000000 |
| Lord of the Ladybirds | 4.2000000000 |
| Lady of the Flies     | 2.6000000000 |
| Lady of the Flies     | 2.6000000000 |
| Lady of the Flies     | 2.6000000000 |
+----------------------+--------------+
```

See Also

- MEDIAN() - a special case of `PERCENTILE_CONT` equivalent to `PERCENTILE_CONT(0.5)`

This page is licensed: CC BY-SA / Gnu FDL

PERCENTILE_DISC

Syntax

Description

`PERCENTILE_DISC()` (standing for discrete percentile) is a window function which returns the first value in the set whose ordered position is the same or more than the specified fraction.

Essentially, the following process is followed to find the value to return:

- Get the number of rows in the partition.
- Walk through the partition, in order, until finding the first row with CUME_DIST() >= function_argument.

Examples

```
CREATE TABLE book_rating (name CHAR(30), star_rating TINYINT);

INSERT INTO book_rating VALUES ('Lord of the Ladybirds', 5);
INSERT INTO book_rating VALUES ('Lord of the Ladybirds', 3);
INSERT INTO book_rating VALUES ('Lady of the Flies', 1);
INSERT INTO book_rating VALUES ('Lady of the Flies', 2);
INSERT INTO book_rating VALUES ('Lady of the Flies', 5);

SELECT name, PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY star_rating)
  OVER (PARTITION BY name) AS pc FROM book_rating;
+-----------------------+------+
| name                  | pc   |
+-----------------------+------+
| Lord of the Ladybirds |    3 |
| Lord of the Ladybirds |    3 |
| Lady of the Flies     |    2 |
| Lady of the Flies     |    2 |
| Lady of the Flies     |    2 |
+-----------------------+------+
5 rows in set (0.000 sec)

SELECT name, PERCENTILE_DISC(0) WITHIN GROUP (ORDER BY star_rating)
 OVER (PARTITION BY name) AS pc FROM book_rating;
+-----------------------+------+
| name                  | pc   |
+-----------------------+------+
| Lord of the Ladybirds |    3 |
| Lord of the Ladybirds |    3 |
| Lady of the Flies     |    1 |
| Lady of the Flies     |    1 |
| Lady of the Flies     |    1 |
+-----------------------+------+
5 rows in set (0.000 sec)

SELECT name, PERCENTILE_DISC(1) WITHIN GROUP (ORDER BY star_rating)
  OVER (PARTITION BY name) AS pc FROM book_rating;
+-----------------------+------+
| name                  | pc   |
+-----------------------+------+
| Lord of the Ladybirds |    5 |
| Lord of the Ladybirds |    5 |
| Lady of the Flies     |    5 |
| Lady of the Flies     |    5 |
| Lady of the Flies     |    5 |
+-----------------------+------+
5 rows in set (0.000 sec)

SELECT name, PERCENTILE_DISC(0.6) WITHIN GROUP (ORDER BY star_rating)
  OVER (PARTITION BY name) AS pc FROM book_rating;
+-----------------------+------+
| name                  | pc   |
+-----------------------+------+
| Lord of the Ladybirds |    5 |
| Lord of the Ladybirds |    5 |
| Lady of the Flies     |    2 |
| Lady of the Flies     |    2 |
| Lady of the Flies     |    2 |
+-----------------------+------
```

See Also

- CUME_DIST()

This page is licensed: CC BY-SA / Gnu FDL

RANK

Syntax

```
RANK() OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```

Description

RANK() is a window function that displays the number of a given row, starting at one and following the ORDER BY sequence of the window function, with identical values receiving the same result. It is similar to the ROW_NUMBER() function except that in that function, identical values will receive a different row number for each result.

Examples

The distinction between DENSE_RANK(), RANK() and ROW_NUMBER():

```
CREATE TABLE student(course VARCHAR(10), mark int, name varchar(10));

INSERT INTO student VALUES
  ('Maths', 60, 'Thulile'),
  ('Maths', 60, 'Pritha'),
  ('Maths', 70, 'Voitto'),
  ('Maths', 55, 'Chun'),
  ('Biology', 60, 'Bilal'),
   ('Biology', 70, 'Roger');

SELECT
  RANK() OVER (PARTITION BY course ORDER BY mark DESC) AS rank,
  DENSE_RANK() OVER (PARTITION BY course ORDER BY mark DESC) AS dense_rank,
  ROW_NUMBER() OVER (PARTITION BY course ORDER BY mark DESC) AS row_num,
  course, mark, name
FROM student ORDER BY course, mark DESC;
+------+------------+---------+---------+------+---------+
| rank | dense_rank | row_num | course  | mark | name    |
+------+------------+---------+---------+------+---------+
|    1 |          1 |       1 | Biology |   70 | Roger   |
|    2 |          2 |       2 | Biology |   60 | Bilal   |
|    1 |          1 |       1 | Maths   |   70 | Voitto  |
|    2 |          2 |       2 | Maths   |   60 | Thulile |
|    2 |          2 |       3 | Maths   |   60 | Pritha  |
|    4 |          3 |       4 | Maths   |   55 | Chun    |
+------+------------+---------+---------+------+---------+
```

See Also

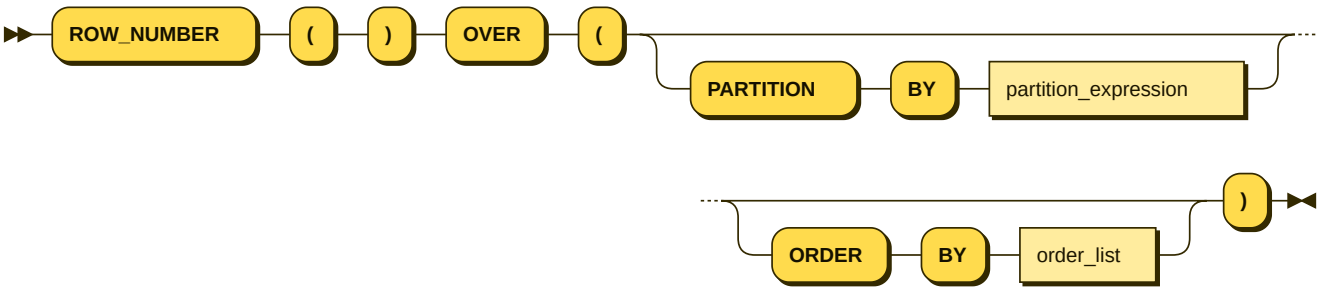
- DENSE_RANK()
- ROW_NUMBER()
- ORDER BY

This page is licensed: CC BY-SA / Gnu FDL

ROW_NUMBER

Syntax

```
ROW_NUMBER() OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)
```



Railroad diagram of ROW_NUMBER — equivalent to the BNF above

Description

`ROW_NUMBER()` is a window function that displays the number of a given row, starting at one and following the ORDER BY sequence of the window function, with identical values receiving different row numbers. It is similar to the RANK() and DENSE_RANK() functions except that in that function, identical values will receive the same rank for each result.

Examples

The distinction between DENSE_RANK(), RANK() and ROW_NUMBER():

```
CREATE TABLE student(course VARCHAR(10), mark int, name varchar(10));

INSERT INTO student VALUES
  ('Maths', 60, 'Thulile'),
  ('Maths', 60, 'Pritha'),
  ('Maths', 70, 'Voitto'),
  ('Maths', 55, 'Chun'),
  ('Biology', 60, 'Bilal'),
   ('Biology', 70, 'Roger');

SELECT
  RANK() OVER (PARTITION BY course ORDER BY mark DESC) AS rank,
  DENSE_RANK() OVER (PARTITION BY course ORDER BY mark DESC) AS dense_rank,
  ROW_NUMBER() OVER (PARTITION BY course ORDER BY mark DESC) AS row_num,
  course, mark, name
FROM student ORDER BY course, mark DESC;
+------+------------+---------+---------+------+---------+
| rank | dense_rank | row_num | course  | mark | name    |
+------+------------+---------+---------+------+---------+
|    1 |          1 |       1 | Biology |   70 | Roger   |
|    2 |          2 |       2 | Biology |   60 | Bilal   |
|    1 |          1 |       1 | Maths   |   70 | Voitto  |
|    2 |          2 |       2 | Maths   |   60 | Thulile |
|    2 |          2 |       3 | Maths   |   60 | Pritha  |
|    4 |          3 |       4 | Maths   |   55 | Chun    |
+------+------------+---------+---------+------+---------+
```

See Also

- RANK()
- DENSE_RANK()

- ORDER BY

This page is licensed: CC BY-SA / Gnu FDL

String Functions

See ASCII.

Return the ASCII value of the first character. This function returns the numeric ASCII code for the leftmost character of the input string.

See BIN.

Return the binary representation of a number. This function converts a number to its binary string equivalent.

See BINARY Operator.

Cast a string to a binary string. This operator converts a character string to a binary string, often used for case-sensitive comparisons.

See BIT_LENGTH.

Return the length of a string in bits. This function calculates the size of the string in bits (length in bytes multiplied by 8).

See CAST.

Complete CAST reference for MariaDB. Complete function guide with syntax, parameters, return values, and usage examples with comprehensive examples and best.

See CHAR Function.

Return the character for each integer passed. This function interprets arguments as integer ASCII values and returns a string of those characters.

See CHAR_LENGTH.

Return the length of a string in characters. This function counts the number of characters in the string, treating multi-byte characters as single units.

See CHARACTER_LENGTH.

Synonym for CHAR_LENGTH(). Returns the number of characters in the string.

See CHR.

Return the character for a specific ASCII value. This function is similar to CHAR() but accepts a single integer argument.

See CONCAT.

Complete CONCAT reference for MariaDB. Complete function guide with syntax, parameters, return values, and usage examples with comprehensive examples and.

See CONCAT_WS.

Concatenate with separator. This function joins strings with a specified separator. It skips NULL values during concatenation.

See CONVERT.

Complete CONVERT() reference: CONVERT(expr,type) and CONVERT(expr USING charset) syntax, SIGNED/UNSIGNED/BINARY/CHAR types, and CAST() differences.

See ELT.

Return the string at a specific index. This function returns the N-th string from a list of arguments.

See EXPORT_SET.

Return a string representation of bits. This function generates a string based on the bits set in a number, using specified ‘on’ and ‘off’ strings.

See EXTRACTVALUE.

Extract a value from XML. This function returns the text content of an XML fragment matching a given XPath expression.

See FIELD.

Return the index of a string in a list. This function returns the position of the first argument within the subsequent list of arguments.

See FIND_IN_SET.

Return the index of a string in a comma-separated list. This function finds the position of a string within a list of strings separated by commas.

See FORMAT.

Format a number. This function formats a number to a format like ‘#,###,###.##’, rounded to a specified number of decimal places.

See FROM_BASE64.

Decode a base-64 encoded string. This function takes a base-64 string and returns the decoded binary result.

See HEX.

Return the hexadecimal representation. This function converts a number or string to its hexadecimal string equivalent.

See INSERT Function.

Insert a substring into a string. This function inserts a string within another string at a specified position and length, replacing existing characters.

See INSTR.

Return the position of the first occurrence of a substring. This function locates a substring within a string and returns its index.

See LCASE.

Synonym for LOWER(). Converts a string to lowercase characters.

See LEFT.

Return the leftmost characters. This function returns the specified number of characters from the beginning (left) of a string.

See LENGTH.

Return the length of a string in bytes. This function counts the number of bytes in the string, which may differ from character count for multi-byte strings.

See LENGTHB.

Return the length of a string in bytes. This function is a synonym for LENGTH() in default mode, returning the byte count.

See LIKE.

Complete LIKE operator reference: LIKE/NOT LIKE pattern matching syntax, % and _ wildcards, ESCAPE clause, NULL behavior, and collation case-sensitivity.

See LOAD_FILE.

Read a file from the server. This function reads the content of a file located on the server host and returns it as a string.

See LOCATE.

Return the position of the first occurrence of a substring. This function finds the starting position of a substring within a string.

See LOWER.

Convert a string to lowercase. This function returns the string with all characters converted to lowercase.

See LPAD.

Left-pad a string. This function pads a string on the left side with a specified string until it reaches a certain length.

See LTRIM.

Remove leading spaces. This function returns the string with any leading whitespace characters removed.

See MAKE_SET.

Return a set of comma-separated strings. This function returns a string consisting of substrings corresponding to the set bits in a given number.

See MATCH AGAINST.

Perform a full-text search. This construct searches for a text query against a set of columns indexed with a FULLTEXT index.

See MID.

Synonym for SUBSTRING(). Returns a substring starting at a specified position for a given length.

See NATURAL_SORT_KEY.

Generate a sort key for natural ordering. This function produces a key that allows strings containing numbers to be sorted in a human-readable order.

See NOT LIKE.

Negated pattern matching. This operator tests whether a string does NOT match a specified SQL pattern.

See NOT REGEXP.

Negated regular expression matching. This operator tests whether a string does NOT match a specified regular expression pattern.

See OCTET_LENGTH.

Return the length of a string in bytes. This function is a synonym for LENGTH() and returns the number of bytes in the string.

See ORD.

Return the numeric value of the first character. This function returns the code for the leftmost character, supporting multi-byte characters.

See POSITION.

Synonym for LOCATE(). Returns the position of the first occurrence of a substring within a string.

See QUOTE.

Quote a string for SQL usage. This function produces a string ready for use as a data value in an SQL statement, escaping special characters.

See REPEAT Function.

Repeat a string. This function returns a string consisting of the input string repeated a specified number of times.

See REPLACE Function.

Complete REPLACE Function reference for MariaDB. Complete function guide with syntax, parameters, return values, and usage examples for production use.

See REVERSE.

Reverse a string. This function returns the string with the order of its characters reversed.

See RIGHT.

Return the rightmost characters. This function returns the specified number of characters from the end (right) of a string.

See RPAD.

Right-pad a string. This function pads a string on the right side with a specified string until it reaches a certain length.

See RTRIM.

Remove trailing spaces. This function returns the string with any trailing whitespace characters removed.

See SFORMAT.

Format strings with arbitrary patterns. This function allows complex string formatting using a pattern string and arguments.

See SOUNDEX.

Return the Soundex string. This function calculates the Soundex key for a string, allowing comparison of words that sound similar.

See SOUNDS LIKE.

Compare strings by sound. This operator tests if two strings have the same Soundex value, useful for fuzzy matching.

See SPACE.

Return a string of spaces. This function returns a string consisting of a specified number of space characters.

See STRCMP.

Compare two strings. This function returns 0 if strings are equal, -1 if the first is smaller, and 1 if the first is larger.

See SUBSTR.

SUBSTR() is a synonym for SUBSTRING(), returning a substring from a string.

See SUBSTRING.

Complete SUBSTRING() reference: syntax forms (pos,len, FROM/FOR), negative position from end, NULL handling, and Oracle sql_mode position 0 behavior.

See SUBSTRING_INDEX.

Complete SUBSTRING_INDEX() reference: SUBSTRING_INDEX(str,delim,count) syntax, positive/negative count, case-sensitive delimiter, NULL rules.

See TO_BASE64.

Encode a string to base-64. This function converts a string argument to its base-64 encoded form.

See TO_CHAR.

Convert to string. This function converts a value (often date/time) to a string, potentially using a format mask.

See TRIM.

Remove spaces from both ends. This function removes leading and trailing whitespace (or other specified characters) from a string.

See TRIM_ORACLE.

Oracle-compatible TRIM function. This version of TRIM provides compatibility with Oracle's syntax for removing characters from a string.

See Type Conversion.

Understand Type Conversion in MariaDB. Learn the rules for implicit conversion during comparisons and arithmetic, and how to use CAST for predictable results.

See UCASE.

Synonym for UPPER(). Converts a string to uppercase characters.

See UNHEX.

Convert hexadecimal to string. This function interprets pairs of hexadecimal digits as numbers and converts them to the characters they represent.

See UPDATEXML.

Replace a portion of XML. This function replaces a section of XML markup matching an XPath expression with a new XML fragment.

See UPPER.

Convert a string to uppercase. This function returns the string with all characters converted to uppercase.

See WEIGHT_STRING.

Return the weight string. This function returns the binary string that represents the sorting and comparison value of the input string.

See Regular Expressions Functions.

Learn about regular expression functions in MariaDB Server. This section details SQL functions for powerful pattern matching and manipulation of string data using regular expressions.

This page is licensed: CC BY-SA / Gnu FDL

ASCII

Syntax

```
ASCII(str)
```

Description

Returns the numeric ASCII value of the leftmost character of the string argument. Returns `0` if the given string is empty and `NULL` if it is `NULL`.

`ASCII()` works for 8-bit characters.

Examples

```
SELECT ASCII(9);
+----------+
| ASCII(9) |
+----------+
|       57 |
+----------+

SELECT ASCII('9');
+------------+
| ASCII('9') |
+------------+
|         57 |
+------------+

SELECT ASCII('abc');
+--------------+
| ASCII('abc') |
+--------------+
|           97 |
+--------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

BIN

Syntax

```
BIN(N)
```

Description

Returns a string representation of the binary value of the given longlong (that is, BIGINT) number. This is equivalent to CONV(N,10,2). The argument should be positive. If it is a `FLOAT`, it will be truncated. Returns `NULL` if the argument is `NULL`.

Examples

```
SELECT BIN(12);
+---------+
| BIN(12) |
+---------+
| 1100    |
+---------+
```

See Also

- Binary literals
- CONV()
- OCT()
- HEX()

This page is licensed: GPLv2, originally from fill_help_tables.sql

BINARY Operator

This page describes the BINARY operator. For details about the data type, see Binary Data Type.

Syntax

```
BINARY
```

Description

The `BINARY` operator casts the string following it to a binary string. This is an easy way to force a column comparison to be done byte by byte rather than character by character. This causes the comparison to be case sensitive even if the column isn't defined as BINARY or BLOB.

`BINARY` also causes trailing spaces to be significant.

Examples

```
SELECT 'a' = 'A';
+-----------+
| 'a' = 'A' |
+-----------+
|         1 |
+-----------+

SELECT BINARY 'a' = 'A';
+------------------+
| BINARY 'a' = 'A' |
+------------------+
|                0 |
+------------------+

SELECT 'a' = 'a ';
+------------+
| 'a' = 'a ' |
+------------+
|          1 |
+------------+

SELECT BINARY 'a' = 'a ';
+-------------------+
| BINARY 'a' = 'a ' |
+-------------------+
|                 0 |
+-------------------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

BIT_LENGTH

Syntax

```
BIT_LENGTH(str)
```

Description

Returns the length of the given string argument in bits. If the argument is not a string, it will be converted to string. If the argument is `NULL`, it returns `NULL`.

Examples

```
SELECT BIT_LENGTH('text');
+--------------------+
| BIT_LENGTH('text') |
+--------------------+
|                 32 |
+--------------------+
```

```
SELECT BIT_LENGTH('');
+----------------+
| BIT_LENGTH('') |
+----------------+
|              0 |
+----------------+
```

Compatibility

PostgreSQL and Sybase support `BIT_LENGTH()`.

This page is licensed: GPLv2, originally from fill_help_tables.sql

CAST

Syntax

```
CAST(expr AS type)
```

Description

The `CAST()` function takes a value of one data type and produces a value of another data type, similar to the CONVERT() function.

The type can be one of the following values:

- BINARY
- CHAR
- DATE
- DATETIME
- DECIMAL
- DOUBLE
- FLOAT
- INTEGER (short for `SIGNED INTEGER`)
- SIGNED [INTEGER]
- UNSIGNED [INTEGER]
- TIME
- VARCHAR
- XMLTYPE (Available starting with MariaDB 12.3)

The main difference between `CAST` and CONVERT() is that CONVERT(expr,type) is ODBC syntax, while `CAST(expr as type)` and CONVERT(… USING …) are SQL92 syntax.

To cast a value to a string data type while specifying the character set, use this extended syntax:

```
CAST(expr AS CHAR CHARACTER SET name)
```

See the examples for casting to character sets.

> **Note:**
>
> Using an introducer like `_utf8mb4'text'` is often more efficient than `CAST('text' AS CHAR CHARACTER SET utf8mb4)`, but `CAST` is necessary when converting data types (like numbers to strings) into a specific encoding. See this example, too.

You can use the `CAST()` function with the `INTERVAL` keyword.

This introduced an incompatibility with previous versions of MariaDB, and all versions of MySQL (see the example below).

Examples

Simple Casts

```
SELECT CAST("abc" AS BINARY);
SELECT CAST("1" AS UNSIGNED INTEGER);
SELECT CAST(123 AS CHAR CHARACTER SET utf8)
```

> **Warning:**
>
> When you casts to CHAR without specifying the character set, the collation_connection character set collation is used. When used with `CHAR CHARACTER SET`, the default collation for that character set is used.

```
SELECT COLLATION(CAST(123 AS CHAR));
+-----------------------------+
| COLLATION(CAST(123 AS CHAR)) |
+-----------------------------+
| latin1_swedish_ci            |
+-----------------------------+

SELECT COLLATION(CAST(123 AS CHAR CHARACTER SET utf8));
+-----------------------------------------------+
| COLLATION(CAST(123 AS CHAR CHARACTER SET utf8)) |
+-----------------------------------------------+
| utf8_general_ci                               |
+-----------------------------------------------+
```

If you also want to change the collation, you have to use the `COLLATE` operator:

```
SELECT COLLATION(CAST(123 AS CHAR CHARACTER SET utf8)
  COLLATE utf8_unicode_ci);
+-------------------------------------------------------------------------+
| COLLATION(CAST(123 AS CHAR CHARACTER SET utf8) COLLATE utf8_unicode_ci) |
+-------------------------------------------------------------------------+
| utf8_unicode_ci                                                         |
+-------------------------------------------------------------------------+
```

Using `CAST()` to order an ENUM field as a CHAR rather than the internal numerical value:

```
CREATE TABLE enum_list (enum_field enum('c','a','b'));

INSERT INTO enum_list (enum_field)
VALUES('c'),('a'),('c'),('b');

SELECT * FROM enum_list
ORDER BY enum_field;
+------------+
| enum_field |
+------------+
| c          |
| c          |
| a          |
| b          |
+------------+

SELECT * FROM enum_list
ORDER BY CAST(enum_field AS CHAR);
+------------+
| enum_field |
+------------+
| a          |
| b          |
| c          |
| c          |
+------------+
```

The following `CAST()` gives warnings, because `x'aa'` and `'X'aa'` don't behave as a number. In all versions of MySQL, no warnings are triggered because they erroneously behave as a number:

```
SELECT CAST(0xAA AS UNSIGNED), CAST(x'aa' AS UNSIGNED), CAST(X'aa' AS UNSIGNED);
+------------------------+-------------------------+-------------------------+
| CAST(0xAA AS UNSIGNED) | CAST(x'aa' AS UNSIGNED) | CAST(X'aa' AS UNSIGNED) |
+------------------------+-------------------------+-------------------------+
|                    170 |                       0 |                       0 |
+------------------------+-------------------------+-------------------------+
1 row in set, 2 warnings (0.00 sec)

Warning (Code 1292): Truncated incorrect INTEGER value: '\xAA'
Warning (Code 1292): Truncated incorrect INTEGER value: '\xAA'
```

Casting to Intervals

```
SELECT CAST(2019-01-04 AS INTERVAL DAY_SECOND(2)) AS "Cast";

+-------------+
| Cast        |
+-------------+
| 00:20:14.00 |
+-------------+
```

Casting to Character Sets

Casting a Literal to a Specific Character Set

```
-- Converts the literal string 'abc' to the utf8mb4 character set
SELECT CAST('abc' AS CHAR CHARACTER SET utf8mb4);
```

Casting a Binary Literal to a Readable Character Set

```
-- Converts a hexadecimal (binary) literal to a UTF-8 string
SELECT CAST(0x68656c6c6f AS CHAR CHARACTER SET utf8mb4);
```

Using an Introducer Inside a CAST

```
-- Telling the parser the input is latin1, then casting the result to utf8mb4
SELECT CAST(_latin1'text' AS CHAR CHARACTER SET utf8mb4);
```

See Also

- Supported data types
- Microseconds in MariaDB
- String literals
- COLLATION()
- CONVERT()

This page is licensed: GPLv2, originally from fill_help_tables.sql

CHAR Function

Syntax

```
CHAR(N,... [USING charset_name])
```

Description

`CHAR()` interprets each argument as an INT and returns a string consisting of the characters given by the code values of those integers. `NULL` values are skipped. By default, `CHAR()` returns a binary string. To produce a string in a given character set, use the optional `USING` clause:

```
SELECT CHARSET(CHAR(0x65)), CHARSET(CHAR(0x65 USING utf8));
+--------------------+------------------------------+
| CHARSET(CHAR(0x65)) | CHARSET(CHAR(0x65 USING utf8)) |
+--------------------+------------------------------+
| binary              | utf8                         |
+--------------------+------------------------------+
```

If `USING` is given and the result string is illegal for the given character set, a warning is issued. Also, if strict SQL mode is enabled, the result from `CHAR()` becomes `NULL`.

Examples

```
SELECT CHAR(77,97,114,'105',97,'68',66);
+----------------------------------+
| CHAR(77,97,114,'105',97,'68',66) |
+----------------------------------+
| MariaDB                          |
+----------------------------------+

SELECT CHAR(77,77.3,'77.3');
+----------------------+
| CHAR(77,77.3,'77.3') |
+----------------------+
| MMM                  |
+----------------------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1292): Truncated incorrect INTEGER value: '77.3'
```

See Also

- Character Sets and Collations

- ASCII() - Return ASCII value of first character
- ORD() - Return value for character in single or multi-byte character sets
- CHR - Similar, Oracle-compatible, function

This page is licensed: GPLv2, originally from fill_help_tables.sql

CHAR_LENGTH

Syntax

```
CHAR_LENGTH(str)
CHARACTER_LENGTH(str)
```

Description

Returns the length of the given string argument, measured in characters. A multi-byte character counts as a single character. This means that for a string containing five two-byte characters, LENGTH() (or OCTET_LENGTH() in Oracle mode) returns 10, whereas `CHAR_LENGTH()` returns 5. If the argument is `NULL`, it returns `NULL`.

If the argument is not a string value, it is converted into a string.

It is synonymous with the `CHARACTER_LENGTH()` function.

Examples

```
SELECT CHAR_LENGTH('MariaDB');
+------------------------+
| CHAR_LENGTH('MariaDB') |
+------------------------+
|                      7 |
+------------------------+
```

When Oracle mode is not set:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+-------------------+--------------+---------------+--------------------+
| CHAR_LENGTH('π')  | LENGTH('π')  | LENGTHB('π')  | OCTET_LENGTH('π')  |
+-------------------+--------------+---------------+--------------------+
|                 1 |            2 |             2 |                  2 |
+-------------------+--------------+---------------+--------------------+
```

In Oracle mode:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+-------------------+--------------+---------------+--------------------+
| CHAR_LENGTH('π')  | LENGTH('π')  | LENGTHB('π')  | OCTET_LENGTH('π')  |
+-------------------+--------------+---------------+--------------------+
|                 1 |            1 |             2 |                  2 |
+-------------------+--------------+---------------+--------------------+
```

See Also

- LENGTH()
- LENGTHB()
- OCTET_LENGTH()
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

CHARACTER_LENGTH

Syntax

```
CHARACTER_LENGTH(str)
```

Description

`CHARACTER_LENGTH()` is a synonym for CHAR_LENGTH().

This page is licensed: GPLv2, originally from fill_help_tables.sql

CHR

Syntax

```
CHR(N)
```

Description

`CHR()` interprets each argument N as an integer and returns a VARCHAR(1) string consisting of the character given by the code values of the integer. The character set and collation of the string are set according to the values of the character_set_database and collation_database system variables.

`CHR()` is similar to the CHAR() function, but only accepts a single argument.

`CHR()` is available in all sql_modes.

Examples

```
SELECT CHR(67);
+---------+
| CHR(67) |
+---------+
| C       |
+---------+

SELECT CHR('67');
+-----------+
| CHR('67') |
+-----------+
| C         |
+-----------+

SELECT CHR('C');
+----------+
| CHR('C') |
+----------+
|          |
+----------+
1 row in set, 1 warning (0.000 sec)

SHOW WARNINGS;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1292 | Truncated incorrect INTEGER value: 'C' |
+---------+------+----------------------------------------+
```

See Also

- Character Sets and Collations

- ASCII() - Return ASCII value of first character
- ORD() - Return value for character in single or multi-byte character sets
- CHAR() - Similar function which accepts multiple integers

This page is licensed: CC BY-SA / Gnu FDL

CONCAT

Syntax

```
CONCAT(str1,str2,...)
```

Description

Returns the string that results from concatenating the arguments. May have one or more arguments. If all arguments are non-binary strings, the result is a non-binary string. If the arguments include any binary strings, the result is a binary string. A numeric argument is converted to its equivalent binary string form; if you want to avoid that, you can use an explicit type cast, as in this example:

```
SELECT CONCAT(CAST(int_col AS CHAR), char_col);
```

`CONCAT()` returns `NULL` if any argument is `NULL`.

A `NULL` parameter hides all information contained in other parameters from the result. Sometimes this is not desirable; to avoid this, you can:

- Use the CONCAT_WS() function with an empty separator, because that function is `NULL`-safe.
- Use IFNULL() to turn NULLs into empty strings.

Oracle Mode

In Oracle mode, `CONCAT` ignores null.

Examples

```
SELECT CONCAT('Ma', 'ria', 'DB');
+--------------------------+
| CONCAT('Ma', 'ria', 'DB') |
+--------------------------+
| MariaDB                  |
+--------------------------+

SELECT CONCAT('Ma', 'ria', NULL, 'DB');
+---------------------------------+
| CONCAT('Ma', 'ria', NULL, 'DB') |
+---------------------------------+
| NULL                            |
+---------------------------------+

SELECT CONCAT(42.0);
+-------------+
| CONCAT(42.0) |
+-------------+
| 42.0         |
+-------------+
```

Using `IFNULL()` to handle `NULL` values:

```
SELECT CONCAT('The value of @v is: ', IFNULL(@v, ''));
+-----------------------------------------------+
| CONCAT('The value of @v is: ', IFNULL(@v, '')) |
+-----------------------------------------------+
| The value of @v is:                           |
+-----------------------------------------------+
```

In Oracle mode:

```
SELECT CONCAT('Ma', 'ria', NULL, 'DB');
+---------------------------------+
| CONCAT('Ma', 'ria', NULL, 'DB') |
+---------------------------------+
| MariaDB                         |
+---------------------------------+
```

See Also

- GROUP_CONCAT()
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

CONCAT_WS

Syntax

```
CONCAT_WS(separator,str1,str2,...)
```

Description

`CONCAT_WS()` stands for Concatenate With Separator and is a special form of CONCAT(). The first argument is the separator for the rest of the arguments. The separator is added between the strings to be concatenated. The separator can be a string, as can the rest of the arguments.

If the separator is `NULL`, the result is `NULL`; all other `NULL` values are skipped. This makes `CONCAT_WS()` suitable when you want to concatenate some values and avoid losing all information if one of them is `NULL`.

Examples

```
SELECT CONCAT_WS(',','First name','Second name','Last Name');
+------------------------------------------------------+
| CONCAT_WS(',','First name','Second name','Last Name') |
+------------------------------------------------------+
| First name,Second name,Last Name                     |
+------------------------------------------------------+

SELECT CONCAT_WS('-','Floor',NULL,'Room');
+-----------------------------------+
| CONCAT_WS('-','Floor',NULL,'Room') |
+-----------------------------------+
| Floor-Room                        |
+-----------------------------------+
```

In some cases, remember to include a space in the separator string:

```
SET @a = 'gnu', @b = 'penguin', @c = 'sea lion';
Query OK, 0 rows affected (0.00 sec)

SELECT CONCAT_WS(', ', @a, @b, @c);
+-----------------------------+
| CONCAT_WS(', ', @a, @b, @c) |
+-----------------------------+
| gnu, penguin, sea lion      |
+-----------------------------+
```

Using `CONCAT_WS()` to handle `NULL`s:

```
SET @a = 'a', @b = NULL, @c = 'c';

SELECT CONCAT_WS('', @a, @b, @c);
+---------------------------+
| CONCAT_WS('', @a, @b, @c) |
+---------------------------+
| ac                        |
+---------------------------+
```

See Also

- GROUP_CONCAT()

This page is licensed: GPLv2, originally from fill_help_tables.sql

CONVERT

Syntax

```
CONVERT(expr,type), CONVERT(expr USING transcoding_name)
```

Description

The `CONVERT()` and CAST() functions take a value of one type and produce a value of another type.

The type can be one of the following values:

- BINARY
- CHAR
- DATE
- DATETIME
- [DECIMAL (M[,D])]
- DOUBLE
- FLOAT
- INTEGER
  - Short for `SIGNED INTEGER`
- SIGNED [INTEGER]
- UNSIGNED [INTEGER]
- TIME
- VARCHAR (in Oracle mode)

Note that in MariaDB, `INT` and `INTEGER` are the same thing.

`BINARY` produces a string with the BINARY data type. If the optional length is given, `BINARY(N)` causes the cast to use no more than `N` bytes of the argument. Values shorter than the given number in bytes are padded with 0x00

bytes to make them equal the length value.

`CHAR(N)` causes the cast to use no more than the number of characters given in the argument.

The main difference between the CAST() and `CONVERT()` is that `CONVERT(expr,type)` is ODBC syntax while CAST(expr as type) and `CONVERT(... USING ...)` are SQL92 syntax.

`CONVERT()` with `USING` is used to convert data between different character sets. In MariaDB, transcoding names are the same as the corresponding character set names. For example, this statement converts the string 'abc' in the default character set to the corresponding string in the utf8 character set:

```
SELECT CONVERT('abc' USING utf8);
```

Examples

```
SELECT enum_col FROM tbl_name
ORDER BY CAST(enum_col AS CHAR);
```

Converting a BINARY to string to permit the LOWER function to work:

```
SET @x = 'AardVark';

SET @x = BINARY 'AardVark';

SELECT LOWER(@x), LOWER(CONVERT (@x USING latin1));
+-----------+---------------------------------+
| LOWER(@x) | LOWER(CONVERT (@x USING latin1)) |
+-----------+---------------------------------+
| AardVark  | aardvark                        |
+-----------+---------------------------------+
```

See Also

- Character Sets and Collations

This page is licensed: GPLv2, originally from fill_help_tables.sql

ELT

Syntax

```
ELT(N, str1[, str2, str3,...])
```

Description

Takes a numeric argument and a series of string arguments. Returns the string that corresponds to the given numeric position. For instance, it returns `str1` if `N` is 1, `str2` if `N` is 2, and so on. If the numeric argument is a FLOAT, MariaDB rounds it to the nearest INTEGER. If the numeric argument is less than 1, greater than the total number of arguments, or not a number, `ELT()` returns `NULL`. It must have at least two arguments.

It is complementary to the FIELD() function.

Examples

```
SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');
+-----------------------------------+
| ELT(1, 'ej', 'Heja', 'hej', 'foo') |
+-----------------------------------+
| ej                                |
+-----------------------------------+

SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');
+-----------------------------------+
| ELT(4, 'ej', 'Heja', 'hej', 'foo') |
+-----------------------------------+
| foo                               |
+-----------------------------------+
```

See also

- FIND_IN_SET() function. Returns the position of a string in a set of strings.
- FIELD() function. Returns the index position of a string in a list.

This page is licensed: GPLv2, originally from fill_help_tables.sql

EXPORT_SET

Syntax

```
EXPORT_SET(bits, on, off[, separator[, number_of_bits]])
```

Description

Takes a minimum of three arguments. Returns a string where each bit in the given `bits` argument is returned, with the string values given for `on` and `off`.

Bits are examined from right to left, (from low-order to high-order bits). Strings are added to the result from left to right, separated by a separator string (defaults as '`,`'). You can optionally limit the number of bits the `EXPORT_SET()` function examines using the `number_of_bits` option.

If any of the arguments are set as `NULL`, the function returns `NULL`.

Examples

```
SELECT EXPORT_SET(5,'Y','N',',',4);
+----------------------------+
| EXPORT_SET(5,'Y','N',',',4) |
+----------------------------+
| Y,N,Y,N                    |
+----------------------------+

SELECT EXPORT_SET(6,'1','0',',',10);
+-----------------------------+
| EXPORT_SET(6,'1','0',',',10) |
+-----------------------------+
| 0,1,1,0,0,0,0,0,0,0          |
+-----------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

EXTRACTVALUE

Syntax

```
EXTRACTVALUE(xml_frag, xpath_expr)
```

Description

The `EXTRACTVALUE()` function takes two string arguments: a fragment of XML markup and an XPath expression, (also known as a locator). It returns the text (That is, CDDATA), of the first text node which is a child of the element or elements matching the XPath expression.

In cases where a valid XPath expression does not match any text nodes in a valid XML fragment, (including the implicit `/text()` expression), the `EXTRACTVALUE()` function returns an empty string.

Invalid Arguments

When either the XML fragment or the XPath expression is `NULL`, the `EXTRACTVALUE()` function returns `NULL`. When the XML fragment is invalid, it raises a warning Code 1525:

```
Warning (Code 1525): Incorrect XML value: 'parse error at line 1 pos 11: unexpected END-OF-INPUT'
```

When the XPath value is invalid, it generates an Error 1105:

```
ERROR 1105 (HY000): XPATH syntax error: ')'
```

Explicit text() Expressions

This function is the equivalent of performing a match using the XPath expression after appending `/text()`. In other words:

```
SELECT
   EXTRACTVALUE('<cases><case>example</case></cases>', '/cases/case')
    AS 'Base Example',
   EXTRACTVALUE('<cases><case>example</case></cases>', '/cases/case/text()')
    AS 'text() Example';
+--------------+----------------+
| Base Example | text() Example |
+--------------+----------------+
| example      | example        |
+--------------+----------------+
```

Count Matches

When `EXTRACTVALUE()` returns multiple matches, it returns the content of the first child text node of each matching element, in the matched order, as a single, space-delimited string.

By design, the `EXTRACTVALUE()` function makes no distinction between a match on an empty element and no match at all. If you need to determine whether no matching element was found in the XML fragment or if an element was found that contained no child text nodes, use the XPath `count()` function.

For instance, when looking for a value that exists, but contains no child text nodes, you would get a count of the number of matching instances:

```
SELECT
   EXTRACTVALUE('<cases><case/></cases>', '/cases/case')
    AS 'Empty Example',
   EXTRACTVALUE('<cases><case/></cases>', 'count(/cases/case)')
    AS 'count() Example';
+---------------+-----------------+
| Empty Example | count() Example |
+---------------+-----------------+
|               |               1 |
+---------------+-----------------+
```

Alternatively, when looking for a value that doesn't exist, `count()` returns 0.

```
SELECT
   EXTRACTVALUE('<cases><case/></cases>', '/cases/person')
    AS 'No Match Example',
   EXTRACTVALUE('<cases><case/></cases>', 'count(/cases/person)')
    AS 'count() Example';
+------------------+-----------------+
| No Match Example | count() Example |
+------------------+-----------------+
|                  |               0|
+------------------+-----------------+
```

Matches

> **Warning:**
>
> **Important**: The `EXTRACTVALUE()` function only returns `CDDATA`. It does not return tags that the element might contain or the text that these child elements contain.

```
SELECT
  EXTRACTVALUE('<cases><case>Person<email>x@example.com</email></case></cases>', '/cases')
   AS Case;
+--------+
| Case   |
+--------+
| Person |
+--------+
```

Note, in the above example, while the XPath expression matches to the parent `<case>` instance, it does not return the contained `<email>` tag or its content.

Examples

```
SELECT
    ExtractValue('<a>ccc<b>ddd</b></a>', '/a')            AS val1,
    ExtractValue('<a>ccc<b>ddd</b></a>', '/a/b')          AS val2,
    ExtractValue('<a>ccc<b>ddd</b></a>', '//b')           AS val3,
    ExtractValue('<a>ccc<b>ddd</b></a>', '/b')            AS val4,
    ExtractValue('<a>ccc<b>ddd</b><b>eee</b></a>', '//b') AS val5;
+------+------+------+------+---------+
| val1 | val2 | val3 | val4 | val5    |
+------+------+------+------+---------+
| ccc  | ddd  | ddd  |      | ddd eee |
+------+------+------+------+---------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

FIELD

Syntax

```
FIELD(pattern, str1[,str2,...])
```

Description

Returns the index position of the string or number matching the given pattern. Returns `0` in the event that none of the arguments match the pattern. Raises an Error 1582 if not given at least two arguments.

When all arguments given to the `FIELD()` function are strings, they are treated as case-insensitive. When all the arguments are numbers, they are treated as numbers. Otherwise, they are treated as doubles.

If the given pattern occurs more than once, the `FIELD()` function only returns the index of the first instance. If the given pattern is `NULL`, the function returns `0`, as a `NULL` pattern always fails to match.

This function is complementary to the ELT() function.

Examples

```
SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo')
   AS 'Field Results';
+---------------+
| Field Results |
+---------------+
|             2 |
+---------------+

SELECT FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo')
   AS 'Field Results';
+---------------+
| Field Results |
+---------------+
|             0 |
+---------------+

SELECT FIELD(1, 2, 3, 4, 5, 1) AS 'Field Results';
+---------------+
| Field Results |
+---------------+
|             5 |
+---------------+

SELECT FIELD(NULL, 2, 3) AS 'Field Results';
+---------------+
| Field Results |
+---------------+
|             0 |
+---------------+

SELECT FIELD('fail') AS 'Field Results';
Error 1582 (42000): Incorrect parameter count in call
to native function 'field'
```

See also

- ELT() function. Returns the N'th element from a set of strings.

This page is licensed: GPLv2, originally from fill_help_tables.sql

FIND_IN_SET

Syntax

```
FIND_IN_SET(pattern, strlist)
```

Description

Returns the index position where the given pattern occurs in a string list. The first argument is the pattern you want to search for. The second argument is a string containing comma-separated variables. If the second argument is of the SET data-type, the function is optimized to use bit arithmetic.

If the pattern does not occur in the string list or if the string list is an empty string, the function returns `0`. If either argument is `NULL`, the function returns `NULL`. The function does not return the correct result if the pattern contains a comma (“`,`”) character.

Examples

```
SELECT FIND_IN_SET('b','a,b,c,d') AS "Found Results";
+---------------+
| Found Results |
+---------------+
|             2 |
+---------------+
```

See Also

- ELT() function. Returns the N’th element from a set of strings.

This page is licensed: GPLv2, originally from fill_help_tables.sql

FORMAT

Syntax

```
FORMAT(num, decimal_position[, locale])
```

Description

Formats the given number for display as a string, adding separators to appropriate position and rounding the results to the given decimal position. For instance, it would format `15233.345` to `15,233.35`.

If the given decimal position is `0`, it rounds to return no decimal point or fractional part. You can optionally specify a locale value to format numbers to the pattern appropriate for the given region.

Examples

```
SELECT FORMAT(1234567890.09876543210, 4) AS 'Format';
+--------------------+
| Format             |
+--------------------+
| 1,234,567,890.0988 |
+--------------------+

SELECT FORMAT(1234567.89, 4) AS 'Format';
+----------------+
| Format         |
+----------------+
| 1,234,567.8900 |
+----------------+

SELECT FORMAT(1234567.89, 0) AS 'Format';
+-----------+
| Format    |
+-----------+
| 1,234,568 |
+-----------+

-- Format number to German number formatting
SELECT FORMAT(123456789,2,'de_DE') AS 'Format';
+----------------+
| Format         |
+----------------+
| 123.456.789,00 |
+----------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

FROM_BASE64

Syntax

```
FROM_BASE64(str)
```

Description

Decodes the given base-64 encode string, returning the result as a binary string. Returns `NULL` if the given string is `NULL` or if it's invalid.

It is the reverse of the TO_BASE64 function.

There are numerous methods to base-64 encode a string. MariaDB uses the following:

- It encodes alphabet value 64 as '`+`'.
- It encodes alphabet value 63 as '`/`'.
- It codes output in groups of four printable characters. Each three byte of data encoded uses four characters. If the final group is incomplete, it pads the difference with the '`=`' character.
- It divides long output, adding a new line very 76 characters.
- In decoding, it recognizes and ignores newlines, carriage returns, tabs and space whitespace characters.

```
SELECT TO_BASE64('Maria') AS 'Input';
+-----------+
| Input     |
+-----------+
| TWFyaWE=  |
+-----------+

SELECT FROM_BASE64('TWFyaWE=') AS 'Output';
+--------+
| Output |
+--------+
| Maria  |
+--------+
```

This page is licensed: CC BY-SA / Gnu FDL

HEX

Syntax

```
HEX(N_or_S)
```

Description

If `N_or_S` is a number, returns a string representation of the hexadecimal value of `N`, where `N` is a `longlong` (BIGINT) number. This is equivalent to CONV(N,10,16).

If `N_or_S` is a string, returns a hexadecimal string representation of `N_or_S` where each byte of each character in `N_or_S` is converted to two hexadecimal digits. If `N_or_S` is `NULL`, returns `NULL`. The inverse of this operation is performed by the UNHEX() function.

`HEX()` with an INET6 argument returns a hexadecimal representation of the underlying 16-byte binary string.

Examples

```
SELECT HEX(255);
+----------+
| HEX(255) |
+----------+
| FF       |
+----------+

SELECT 0x4D617269614442;
+------------------+
| 0x4D617269614442 |
+------------------+
| MariaDB          |
+------------------+

SELECT HEX('MariaDB');
+----------------+
| HEX('MariaDB') |
+----------------+
| 4D617269614442 |
+----------------+
```

```
SELECT HEX(CAST('2001:db8::ff00:42:8329' AS INET6));
+-------------------------------------------+
| HEX(CAST('2001:db8::ff00:42:8329' AS INET6)) |
+-------------------------------------------+
| 20010DB8000000000000FF0000428329            |
+-------------------------------------------+
```

See Also

- Hexadecimal literals
- UNHEX()
- CONV()
- BIN()
- OCT()

This page is licensed: GPLv2, originally from fill_help_tables.sql

INSERT Function

Syntax

```
INSERT(str,pos,len,newstr)
```

Description

Returns the string `str`, with the substring beginning at position `pos` and `len` characters long replaced by the string `newstr`. Returns the original string if `pos` is not within the length of the string.
Replaces the rest of the string from position `pos` if `len` is not within the length of the rest of the string. Returns `NULL` if any argument is `NULL`.

Examples

```
SELECT INSERT('Quadratic', 3, 4, 'What');
+-----------------------------------+
| INSERT('Quadratic', 3, 4, 'What') |
+-----------------------------------+
| QuWhattic                         |
+-----------------------------------+

SELECT INSERT('Quadratic', -1, 4, 'What');
+-----------------------------------+
| INSERT('Quadratic', -1, 4, 'What') |
+-----------------------------------+
| Quadratic                          |
+-----------------------------------+

SELECT INSERT('Quadratic', 3, 100, 'What');
+-------------------------------------+
| INSERT('Quadratic', 3, 100, 'What') |
+-------------------------------------+
| QuWhat                              |
+-------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

INSTR

Syntax

```
INSTR(str,substr)
```

Description

Returns the position of the first occurrence of substring *substr* in string *str*. This is the same as the two-argument form of LOCATE(), except that the order of the arguments is reversed.

`INSTR()` performs a case-insensitive search.

If any argument is `NULL`, returns `NULL`.

Examples

```
SELECT INSTR('foobarbar', 'bar');
+---------------------------+
| INSTR('foobarbar', 'bar') |
+---------------------------+
|                         4 |
+---------------------------+

SELECT INSTR('My', 'Maria');
+----------------------+
| INSTR('My', 'Maria') |
+----------------------+
|                    0 |
+----------------------+
```

See Also

- LOCATE() ; Returns the position of a string within a string
- SUBSTRING_INDEX() ; Returns the substring from string before count occurrences of a delimiter

This page is licensed: GPLv2, originally from fill_help_tables.sql

LCASE

Syntax

```
LCASE(str)
```

Description

`LCASE()` is a synonym for LOWER().

This page is licensed: GPLv2, originally from fill_help_tables.sql

LEFT

Syntax

```
LEFT(str,len)
```

Description

Returns the leftmost `len` characters from the string `str`, or `NULL` if any argument is `NULL`.

Examples

```
SELECT LEFT('MariaDB', 5);
+--------------------+
| LEFT('MariaDB', 5) |
+--------------------+
| Maria              |
+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LENGTH

Syntax

```
LENGTH(str)
```

Description

Returns the length of the string `str`.

In the default mode, when Oracle mode is not set, the length is measured in bytes. In this case, a multi-byte character counts as multiple bytes. `LENGTH()` returns the length in bytes, whereas `CHAR_LENGTH()` returns the number of characters. This means that, for a string containing five two-byte characters, `LENGTH()` returns 10, whereas CHAR_LENGTH() returns 5. When running Oracle mode, the length is measured in characters, and `LENGTH` is a synonym for CHAR_LENGTH().

If `str` is not a string value, it is converted into a string. If `str` is `NULL`, the function returns `NULL`.

Examples

```
SELECT LENGTH('MariaDB');
+-------------------+
| LENGTH('MariaDB') |
+-------------------+
|                 7 |
+-------------------+
```

When Oracle mode is not set:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+-------------------+--------------+---------------+--------------------+
| CHAR_LENGTH('π')  | LENGTH('π')  | LENGTHB('π')  | OCTET_LENGTH('π')  |
+-------------------+--------------+---------------+--------------------+
|                 1 |            2 |             2 |                  2 |
+-------------------+--------------+---------------+--------------------+
```

In Oracle mode:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+-------------------+--------------+---------------+--------------------+
| CHAR_LENGTH('π')  | LENGTH('π')  | LENGTHB('π')  | OCTET_LENGTH('π')  |
+-------------------+--------------+---------------+--------------------+
|                 1 |            1 |             2 |                  2 |
+-------------------+--------------+---------------+--------------------+
```

See Also

- CHAR_LENGTH()
- LENGTHB()
- OCTET_LENGTH()
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

LENGTHB

Syntax

```
LENGTHB(str)
```

Description

`LENGTHB()` returns the length of the given string, in bytes. When Oracle mode is not set, this is a synonym for LENGTH.

A multi-byte character counts as multiple bytes. This means that for a string containing five two-byte characters, `LENGTHB()` returns 10, whereas CHAR_LENGTH() returns 5.

If `str` is not a string value, it is converted into a string. If `str` is `NULL`, the function returns `NULL`.

Examples

When Oracle mode is not set:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+------------------+--------------+---------------+--------------------+
| CHAR_LENGTH('π')  | LENGTH('π')  | LENGTHB('π')  | OCTET_LENGTH('π')  |
+------------------+--------------+---------------+--------------------+
|                1 |            2 |             2 |                  2 |
+------------------+--------------+---------------+--------------------+
```

In Oracle mode:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+------------------+--------------+---------------+--------------------+
| CHAR_LENGTH('π')  | LENGTH('π')  | LENGTHB('π')  | OCTET_LENGTH('π')  |
+------------------+--------------+---------------+--------------------+
|                1 |            1 |             2 |                  2 |
+------------------+--------------+---------------+--------------------+
```

See Also

- CHAR_LENGTH()
- LENGTH()
- OCTET_LENGTH()

This page is licensed: CC BY-SA / Gnu FDL

LIKE

Syntax

```
expr LIKE pat [ESCAPE 'escape_char']
expr NOT LIKE pat [ESCAPE 'escape_char']
```

Description

Tests whether *expr* matches the pattern *pat*. Returns either 1 (`TRUE`) or 0 (`FALSE`).
Both *expr* and *pat* may be any valid expression and are evaluated to strings.
Patterns may use the following wildcard characters:

- `%` matches any number of characters, including zero.
- `_` matches any single character.

Use `NOT LIKE` to test if a string does not match a pattern. This is equivalent to using the NOT operator on the entire `LIKE` expression.

If either the expression or the pattern is `NULL`, the result is `NULL`.

`LIKE` performs case-insensitive substring matches if the collation for the expression and pattern is case-insensitive. For case-sensitive matches, declare either argument to use a binary collation using collate, or coerce either of them to a BINARY string using CAST. Use SHOW COLLATION to get a list of available collations. Collations ending in `_bin` are case-sensitive.

Numeric arguments are coerced to binary strings.

The `_` wildcard matches a single character, not byte. It will only match a multi-byte character if it is valid in the expression's character set. For example, `_` will match `_utf8"€"`, but it will not match `_latin1"€"` because the Euro sign is not a valid latin1 character. If necessary, use CONVERT to use the expression in a different character set.

If you need to match the characters `_` or `%`, you must escape them. By default, you can prefix the wildcard characters the backslash character `\` to escape them. The backslash is used both to encode special characters like newlines when a string is parsed as well as to escape wildcards in a pattern after parsing. Thus, to match an actual backslash, you sometimes need to double-escape it as `"\``\``\``\"`.

To avoid difficulties with the backslash character, you can change the wildcard escape character using `ESCAPE` in a `LIKE` expression. The argument to `ESCAPE` must be a single-character string.

Examples

Select the days that begin with "T":

```
CREATE TABLE t1 (d VARCHAR(16));
INSERT INTO t1 VALUES
  ("Monday"), ("Tuesday"), ("Wednesday"),
  ("Thursday"), ("Friday"), ("Saturday"), ("Sunday");
SELECT * FROM t1 WHERE d LIKE "T%";
```

```
SELECT * FROM t1 WHERE d LIKE "T%";
+----------+
| d        |
+----------+
| Tuesday  |
| Thursday |
+----------+
```

Select the days that contain the substring "es":

```
SELECT * FROM t1 WHERE d LIKE "%es%";
```

```
SELECT * FROM t1 WHERE d LIKE "%es%";
+-----------+
| d         |
+-----------+
| Tuesday   |
| Wednesday |
+-----------+
```

Select the six-character day names:

```
SELECT * FROM t1 WHERE d like "___day";
```

```
SELECT * FROM t1 WHERE d like "___day";
+---------+
| d       |
+---------+
| Monday  |
| Friday  |
| Sunday  |
+---------+
```

With the default collations, `LIKE` is case-insensitive:

```
SELECT * FROM t1 WHERE d like "t%";
```

```
SELECT * FROM t1 where d like "t%";
+----------+
| d        |
+----------+
| Tuesday  |
| Thursday |
+----------+
```

Use collate to specify a binary collation, forcing case-sensitive matches:

```
SELECT * FROM t1 WHERE d like "t%" COLLATE latin1_bin;
```

```
SELECT * FROM t1 WHERE d like "t%" COLLATE latin1_bin;
Empty SET (0.00 sec)
```

You can include functions and operators in the expression to match. Select dates based on their day name:

```
CREATE TABLE t2 (d DATETIME);
INSERT INTO t2 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
SELECT * FROM t2 WHERE DAYNAME(d) LIKE "T%";
```

```
SELECT * FROM t2 WHERE DAYNAME(d) LIKE "T%";
+------------------+
| d                |
+------------------+
| 2007-01-30 21:31 |
| 2011-04-21 12:34 |
| 2004-10-07 11:19 |
+------------------+
3 rows in set, 7 warnings (0.00 sec)
```

Optimizing LIKE

- MariaDB can use indexes for `LIKE` on string columns in the case where the LIKE doesn't start with `%` or `_`.
- You can set the optimizer_use_condition_selectivity variable to 5. If this is done, then the optimizer will read optimizer_selectivity_sampling_limit rows to calculate the selectivity of the `LIKE` expression before starting to calculate the query plan. This can help speed up some `LIKE` queries by providing the optimizer with more information about your data.

See Also

- For searches on text columns, with results sorted by relevance, see full-text indexes.
- For more complex searches and operations on strings, you can use regular expressions, which were enhanced in MariaDB 10 (see PCRE Regular Expressions).
- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOAD_FILE

Syntax

```
LOAD_FILE(file_name)
```

Description

Reads the file and returns the file contents as a string. To use this function, the file must be located on the server host, you must specify the full path name to the file, and you must have the FILE privilege. The file must be readable by all and it must be less than the size, in bytes, of the max_allowed_packet system variable. If the secure_file_priv system variable is set to a non-empty directory name, the file to be loaded must be located in that directory.

If the file does not exist or cannot be read because one of the preceding conditions is not satisfied, the function returns `NULL`.

The character_set_filesystem system variable has controlled interpretation of file names that are given as literal strings.

Statements using the `LOAD_FILE()` function are not safe for statement based replication. This is because the slave will execute the `LOAD_FILE()` command itself. If the file doesn't exist on the slave, the function will return `NULL`.

Examples

```
UPDATE t SET blob_col=LOAD_FILE('/tmp/picture') WHERE id=1;
```

See Also

- SELECT INTO DUMPFILE

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOCATE

Syntax

```
LOCATE(substr,str), LOCATE(substr,str,pos)
```

Description

The first syntax returns the position of the first occurrence of substring `substr` in string `str`. The second syntax returns the position of the first occurrence of substring `substr` in string `str`, starting at position `pos`. Returns 0 if `substr` is not in `str`.

`LOCATE()` performs a case-insensitive search.

If any argument is `NULL`, returns `NULL`.

INSTR() is the same as the two-argument form of `LOCATE()`, except that the order of the arguments is reversed.

Examples

```
SELECT LOCATE('bar', 'foobarbar');
+----------------------------+
| LOCATE('bar', 'foobarbar') |
+----------------------------+
|                          4 |
+----------------------------+

SELECT LOCATE('My', 'Maria');
+-----------------------+
| LOCATE('My', 'Maria') |
+-----------------------+
|                     0 |
+-----------------------+

SELECT LOCATE('bar', 'foobarbar', 5);
+-------------------------------+
| LOCATE('bar', 'foobarbar', 5) |
+-------------------------------+
|                             7 |
+-------------------------------+
```

See Also

- INSTR() ; Returns the position of a string within a string
- SUBSTRING_INDEX() ; Returns the substring from string before count occurrences of a delimiter

This page is licensed: GPLv2, originally from fill_help_tables.sql

LOWER

Syntax

```
LOWER(str)
LCASE(str)
```

Description

Returns the string `str` with all characters changed to lowercase according to the current character set mapping. The default is latin1 (cp1252 West European).

`LCASE` is a synonym for `LOWER` .

Examples

```
SELECT LOWER('QUADRATICALLY');
+------------------------+
| LOWER('QUADRATICALLY') |
+------------------------+
| quadratically          |
+------------------------+
```

`LOWER()` (and UPPER()) are ineffective when applied to binary strings (BINARY, VARBINARY, BLOB). To perform letter case conversion, CONVERT the string to a non-binary string:

```
SET @str = BINARY 'North Carolina';

SELECT LOWER(@str), LOWER(CONVERT(@str USING latin1));
+---------------+----------------------------------+
| LOWER(@str)    | LOWER(CONVERT(@str USING latin1)) |
+---------------+----------------------------------+
| North Carolina | north carolina                    |
+---------------+----------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

LPAD

Syntax

```
LPAD(str, len [,padstr])
```

Description

Returns the string `str`, left-padded with the string `padstr` to a length of `len` characters. If `str` is longer than `len`, the return value is shortened to `len` characters. If `padstr` is omitted, the LPAD function pads spaces.

Returns `NULL` if given a `NULL` argument. If the result is empty (zero length), returns either an empty string or with SQL_MODE=Oracle, `NULL`.

The Oracle mode version of the function can be accessed outside of Oracle mode by using `LPAD_ORACLE` as the function name.

Examples

```
SELECT LPAD('hello',10,'.');
+----------------------+
| LPAD('hello',10,'.') |
+----------------------+
| .....hello           |
+----------------------+

SELECT LPAD('hello',2,'.');
+---------------------+
| LPAD('hello',2,'.') |
+---------------------+
| he                  |
+---------------------+
```

With the pad string defaulting to space:

```
SELECT LPAD('hello',10);
+------------------+
| LPAD('hello',10) |
+------------------+
|      hello       |
+------------------+
```

Oracle mode:

```
SELECT LPAD('',0),LPAD_ORACLE('',0);
+------------+-------------------+
| LPAD('',0) | LPAD_ORACLE('',0) |
+------------+-------------------+
|            | NULL              |
+------------+-------------------+
```

See Also

- RPAD - Right-padding instead of left-padding.

This page is licensed: GPLv2, originally from fill_help_tables.sql

LTRIM

Syntax

```
LTRIM(str)
```

Description

Returns the string `str` with leading space characters removed.

Returns `NULL` if given a `NULL` argument. If the result is empty, returns either an empty string, or with SQL_MODE=Oracle, `NULL`.

The Oracle mode version of the function can be accessed outside of Oracle mode by using `LTRIM_ORACLE` as the function name.

Examples

```
SELECT QUOTE(LTRIM('   MariaDB   '));
+------------------------------+
| QUOTE(LTRIM('   MariaDB   ')) |
+------------------------------+
| 'MariaDB   '                 |
+------------------------------+
```

Oracle mode version:

```
SELECT LTRIM(''),LTRIM_ORACLE('');
+-----------+-----------------+
| LTRIM('') | LTRIM_ORACLE('') |
+-----------+-----------------+
|           | NULL            |
+-----------+-----------------+
```

See Also

- RTRIM - trailing spaces removed
- TRIM - removes all given prefixes or suffixes

This page is licensed: GPLv2, originally from fill_help_tables.sql

MAKE_SET

Syntax

```
MAKE_SET(bits,str1,str2,...)
```

Description

Returns a set value (a string containing substrings separated by “,” characters) consisting of the strings that have the corresponding bit in bits set. `str1` corresponds to bit 0, `str2` to bit 1, and so on. `NULL` values in `str1`, `str2`, … are not appended to the result.

Examples

```
SELECT MAKE_SET(1,'a','b','c');
+------------------------+
| MAKE_SET(1,'a','b','c') |
+------------------------+
| a                      |
+------------------------+

SELECT MAKE_SET(1 | 4,'hello','nice','world');
+--------------------------------------+
| MAKE_SET(1 | 4,'hello','nice','world') |
+--------------------------------------+
| hello,world                          |
+--------------------------------------+

SELECT MAKE_SET(1 | 4,'hello','nice',NULL,'world');
+-------------------------------------------+
| MAKE_SET(1 | 4,'hello','nice',NULL,'world') |
+-------------------------------------------+
| hello                                     |
+-------------------------------------------+

SELECT QUOTE(MAKE_SET(0,'a','b','c'));
+--------------------------------+
| QUOTE(MAKE_SET(0,'a','b','c')) |
+--------------------------------+
| ''                             |
+--------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MATCH AGAINST

Syntax

```
MATCH (col1,col2,...) AGAINST (expr [search_modifier])
```

Description

A special construct used to perform a fulltext search on a fulltext index.

See Fulltext Index Overview for a full description, and Full-text Indexes for more articles on the topic.

Examples

```
CREATE TABLE ft_myisam(copy TEXT,FULLTEXT(copy)) ENGINE=MyISAM;

INSERT INTO ft_myisam(copy) VALUES ('Once upon a time'), ('There was a wicked witch'),
 ('Who ate everybody up');

SELECT * FROM ft_myisam WHERE MATCH(copy) AGAINST('wicked');
+--------------------------+
| copy                     |
+--------------------------+
| There was a wicked witch |
+--------------------------+
```

```
SELECT id, body, MATCH (title,body) AGAINST
    ('Security implications of running MySQL as root'
    IN NATURAL LANGUAGE MODE) AS score
    FROM articles WHERE MATCH (title,body) AGAINST
    ('Security implications of running MySQL as root'
    IN NATURAL LANGUAGE MODE);
+----+-------------------------------------+-----------------+
| id | body                                | score           |
+----+-------------------------------------+-----------------+
|  4 | 1. Never run mysqld as root. 2. ... | 1.5219271183014 |
|  6 | When configured properly, MySQL ... | 1.3114095926285 |
+----+-------------------------------------+-----------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MID

Syntax

```
MID(str,pos,len)
```

Description

MID(str,pos,len) is a synonym for SUBSTRING(str,pos,len)!

Examples

```
SELECT MID('abcd',4,1);
+-----------------+
| MID('abcd',4,1) |
+-----------------+
| d               |
+-----------------+

SELECT MID('abcd',2,2);
+-----------------+
| MID('abcd',2,2) |
+-----------------+
| bc              |
+-----------------+
```

A negative starting position:

```
SELECT MID('abcd',-2,4);
+------------------+
| MID('abcd',-2,4) |
+------------------+
| cd               |
+------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

NATURAL_SORT_KEY

Note:

`NATURAL_SORT_KEY` is available from MariaDB 10.7.

Syntax

```
NATURAL_SORT_KEY(str)
```

Description

The `NATURAL_SORT_KEY` function is used for sorting that is closer to natural sorting. Strings are sorted in alphabetical order, while numbers are treated in a way such that, for example, `10` is greater than `2`, whereas in other forms of sorting, `2` would be greater than `10`, just like `z` is greater than `ya`.

There are multiple natural sort implementations, differing in the way they handle leading zeroes, fractions, i18n, negatives, decimals and so on.

MariaDB's implementation ignores leading zeroes when performing the sort.

You can use also use `NATURAL_SORT_KEY` with generated columns. The value is not stored permanently in the table. When using a generated column, the virtual column must be longer than the base column to cater for embedded numbers in the string and MDEV-24582.

Examples

Strings and Numbers

```
CREATE TABLE t1 (c TEXT);

INSERT INTO t1 VALUES ('b1'),('a2'),('a11'),('a1');

SELECT c FROM t1;
+------+
| c    |
+------+
| b1   |
| a2   |
| a11  |
| a1   |
+------+

SELECT c FROM t1 ORDER BY c;
+------+
| c    |
+------+
| a1   |
| a11  |
| a2   |
| b1   |
+------+
```

Unsorted, regular sort and natural sort:

```
TRUNCATE t1;

INSERT INTO t1 VALUES
  ('5.5.31'),('10.7.0'),('10.2.1'),
  ('10.1.22'),('10.3.32'),('10.2.12');

SELECT c FROM t1;
+---------+
| c       |
+---------+
| 5.5.31  |
| 10.7.0  |
| 10.2.1  |
| 10.1.22 |
| 10.3.32 |
| 10.2.12 |
+---------+

SELECT c FROM t1 ORDER BY c;
+---------+
| c       |
+---------+
| 10.1.22 |
| 10.2.1  |
| 10.2.12 |
| 10.3.32 |
| 10.7.0  |
| 5.5.31  |
+---------+

SELECT c FROM t1 ORDER BY NATURAL_SORT_KEY(c);
+---------+
| c       |
+---------+
| 5.5.31  |
| 10.1.22 |
| 10.2.1  |
| 10.2.12 |
| 10.3.32 |
| 10.7.0  |
+---------+
```

IPs

Sorting IPs, unsorted, regular sort and natural sort::

```
TRUNCATE t1;

INSERT INTO t1 VALUES
  ('192.167.3.1'),('192.167.1.12'),('100.200.300.400'),
  ('100.50.60.70'),('100.8.9.9'),('127.0.0.1'),('0.0.0.0');

SELECT c FROM t1;
+-----------------+
| c               |
+-----------------+
| 192.167.3.1     |
| 192.167.1.12    |
| 100.200.300.400 |
| 100.50.60.70    |
| 100.8.9.9       |
| 127.0.0.1       |
| 0.0.0.0         |
+-----------------+

SELECT c FROM t1 ORDER BY c;
+-----------------+
| c               |
+-----------------+
| 0.0.0.0         |
| 100.200.300.400 |
| 100.50.60.70    |
| 100.8.9.9       |
| 127.0.0.1       |
| 192.167.1.12    |
| 192.167.3.1     |
+-----------------+

SELECT c FROM t1 ORDER BY NATURAL_SORT_KEY(c);
+-----------------+
| c               |
+-----------------+
| 0.0.0.0         |
| 100.8.9.9       |
| 100.50.60.70    |
| 100.200.300.400 |
| 127.0.0.1       |
| 192.167.1.12    |
| 192.167.3.1     |
+-----------------+
```

Generated Columns

Using with a generated column:

```
CREATE TABLE t(c VARCHAR(3), k VARCHAR(4) AS (NATURAL_SORT_KEY(c)) INVISIBLE);

INSERT INTO t(c) VALUES ('b1'),('a2'),('a11'),('a10');

SELECT * FROM t ORDER by k;
+------+
| c    |
+------+
| a2   |
| a10  |
| a11  |
| b1   |
+------+
```

Note that if the virtual column is not longer, results may not be as expected:

```
CREATE TABLE t2(c VARCHAR(3), k VARCHAR(3) AS (NATURAL_SORT_KEY(c)) INVISIBLE);

INSERT INTO t2(c) VALUES ('b1'),('a2'),('a11'),('a10');

SELECT * FROM t2 ORDER by k;
+------+
| c    |
+------+
| a2   |
| a11  |
| a10  |
| b1   |
+------+
```

Leading Zeroes

Ignoring leading zeroes can lead to undesirable results in certain contexts. For example:

```
CREATE TABLE t3 (a VARCHAR(4));

INSERT INTO t3 VALUES
  ('a1'), ('a001'), ('a10'), ('a001'), ('a10'),
  ('a01'), ('a01'), ('a01b'), ('a01b'), ('a1');

SELECT a FROM t3 ORDER BY a;
+------+
| a    |
+------+
| a001 |
| a001 |
| a01  |
| a01  |
| a01b |
| a01b |
| a1   |
| a1   |
| a10  |
| a10  |
+------+
10 rows in set (0.000 sec)

SELECT a FROM t3 ORDER BY NATURAL_SORT_KEY(a);
+------+
| a    |
+------+
| a1   |
| a01  |
| a01  |
| a001 |
| a001 |
| a1   |
| a01b |
| a01b |
| a10  |
| a10  |
+------+
```

This may not be what we were hoping for in a ‘natural’ sort. A workaround is to sort by both `NATURAL_SORT_KEY` and regular sort.

```
SELECT a FROM t3 ORDER BY NATURAL_SORT_KEY(a), a;
+------+
| a    |
+------+
| a001 |
| a001 |
| a01  |
| a01  |
| a1   |
| a1   |
| a01b |
| a01b |
| a10  |
| a10  |
+------+
```

This page is licensed: CC BY-SA / Gnu FDL

NOT LIKE

Syntax

```
expr NOT LIKE pat [ESCAPE 'escape_char']
```

Description

This is the same as NOT (expr LIKE pat [ESCAPE 'escape_char']).

This page is licensed: GPLv2, originally from fill_help_tables.sql

NOT REGEXP

Syntax

```
expr NOT REGEXP pat, expr NOT RLIKE pat
```

Description

This is the same as NOT (expr REGEXP pat).

This page is licensed: GPLv2, originally from fill_help_tables.sql

OCTET_LENGTH

Syntax

```
OCTET_LENGTH(str)
```

Description

`OCTET_LENGTH()` returns the length of the given string, in octets (bytes). This is a synonym for LENGTHB(), and, when Oracle mode is not set, a synonym for LENGTH().

A multi-byte character counts as multiple bytes. This means that for a string containing five two-byte characters, `OCTET_LENGTH()` returns 10, whereas CHAR_LENGTH() returns 5.

If `str` is not a string value, it is converted into a string. If `str` is `NULL`, the function returns `NULL`.

Examples

When Oracle mode is not set:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+------------------+--------------+---------------+--------------------+
| CHAR_LENGTH('π')  | LENGTH('π')  | LENGTHB('π')  | OCTET_LENGTH('π')  |
+------------------+--------------+---------------+--------------------+
|                1 |            2 |             2 |                  2 |
+------------------+--------------+---------------+--------------------+
```

In Oracle mode:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+------------------+--------------+---------------+--------------------+
| CHAR_LENGTH('π')  | LENGTH('π')  | LENGTHB('π')  | OCTET_LENGTH('π')  |
+------------------+--------------+---------------+--------------------+
|                1 |            1 |             2 |                  2 |
+------------------+--------------+---------------+--------------------+
```

See Also

- CHAR_LENGTH()
- LENGTH()
- LENGTHB()
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

ORD

Syntax

```
ORD(str)
```

Description

If the leftmost character of the string `str` is a multi-byte character, returns the code for that character, calculated from the numeric values of its constituent bytes using this formula:

```
(1st byte code)
+ (2nd byte code x 256)
+ (3rd byte code x 256 x 256) ...
```

If the leftmost character is not a multi-byte character, `ORD()` returns the same value as the ASCII() function.

Examples

```
SELECT ORD('2');
+----------+
| ORD('2') |
+----------+
|       50 |
+----------+
```

See Also

- ASCII() - Return ASCII value of first character
- CHAR() - Create a character from an integer value

This page is licensed: GPLv2, originally from fill_help_tables.sql

POSITION

Syntax

```
POSITION(substr IN str)
```

Description

`POSITION(substr IN str)` is a synonym for LOCATE(substr,str).

The function is part of ODBC 3.0.

This page is licensed: GPLv2, originally from fill_help_tables.sql

QUOTE

Syntax

```
QUOTE(str)
```

Description

Quotes a string to produce a result that can be used as a properly escaped data value in an SQL statement. The string is returned enclosed by single quotes and with each instance of single quote (“ `'` ”), backslash (“ `\` ”), `ASCII NUL`, and Control-Z preceded by a backslash. If the argument is `NULL`, the return value is the word “ `NULL` ” without enclosing single quotes.

Examples

```
SELECT QUOTE("Don't!");
+-----------------+
| QUOTE("Don't!") |
+-----------------+
| 'Don\'t!'       |
+-----------------+

SELECT QUOTE(NULL);
+-------------+
| QUOTE(NULL) |
+-------------+
| NULL        |
+-------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

REPEAT Function

Syntax

```
REPEAT(str,count)
```

Description

Returns a string consisting of the string `str` repeated `count` times. If `count` is less than 1, returns an empty string. Returns NULL if `str` or `count` are `NULL`.

Examples

```
SELECT QUOTE(REPEAT('MariaDB ',4));
+-----------------------------------+
| QUOTE(REPEAT('MariaDB ',4))       |
+-----------------------------------+
| 'MariaDB MariaDB MariaDB MariaDB ' |
+-----------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

REPLACE Function

Syntax

```
REPLACE(str,from_str,to_str)
```

Description

Returns the string `str` with all occurrences of the string `from_str` replaced by the string `to_str`. `REPLACE()` performs a case-sensitive match when searching for `from_str`. If any argument is `NULL`, the function returns `NULL`.

Examples

```
SELECT REPLACE('www.mariadb.org', 'w', 'Ww');
+---------------------------------------+
| REPLACE('www.mariadb.org', 'w', 'Ww') |
+---------------------------------------+
| WwWwWw.mariadb.org                    |
+---------------------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

REVERSE

Syntax

```
REVERSE(str)
```

Description

Returns the string `str` with the order of the characters reversed. If the input string is `NULL`, the function returns `NULL`. ##### Examples

```
SELECT REVERSE('desserts');
+--------------------+
| REVERSE('desserts') |
+--------------------+
| stressed           |
+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

RIGHT

Syntax

```
RIGHT(str,len)
```

Description

Returns the rightmost *`len`* characters from the string *`str`*, or `NULL` if any argument is `NULL`.

Examples

```
SELECT RIGHT('MariaDB', 2);
+--------------------+
| RIGHT('MariaDB', 2) |
+--------------------+
| DB                 |
+--------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

RPAD

Syntax

```
RPAD(str, len [, padstr])
```

Description

Returns the string `str`, right-padded with the string `padstr` to a length of `len` characters. If `str` is longer than `len`, the return value is shortened to `len` characters. If `padstr` is omitted, the RPAD function pads spaces.

Returns `NULL` if given a `NULL` argument. If the result is empty (a length of zero), returns either an empty string, or, with SQL_MODE=Oracle, `NULL`.

The Oracle mode version of the function can be accessed outside of Oracle mode by using `RPAD_ORACLE` as the function name.

Examples

```
SELECT RPAD('hello',10,'.');
+----------------------+
| RPAD('hello',10,'.') |
+----------------------+
| hello.....           |
+----------------------+

SELECT RPAD('hello',2,'.');
+--------------------+
| RPAD('hello',2,'.') |
+--------------------+
| he                 |
+--------------------+
```

With the pad string defaulting to space:

```
SELECT RPAD('hello',30);
+--------------------------------+
| RPAD('hello',30)               |
+--------------------------------+
| hello                          |
+--------------------------------+
```

Oracle mode:

```
SELECT RPAD('',0),RPAD_ORACLE('',0);
+------------+-------------------+
| RPAD('',0) | RPAD_ORACLE('',0) |
+------------+-------------------+
|            | NULL              |
+------------+-------------------+
```

See Also

- LPAD - Left-padding instead of right-padding.

This page is licensed: GPLv2, originally from fill_help_tables.sql

RTRIM

Syntax

```
RTRIM(str)
```

Description

Returns the string `str` with trailing space characters removed.

Returns NULL if given a NULL argument. If the result is empty, returns either an empty string, or, with SQL_MODE=Oracle, `NULL`.

The Oracle mode version of the function can be accessed outside of Oracle mode by using `RTRIM_ORACLE` as the function name.

Examples

```
SELECT QUOTE(RTRIM('MariaDB    '));
+-----------------------------+
| QUOTE(RTRIM('MariaDB    ')) |
+-----------------------------+
| 'MariaDB'                   |
+-----------------------------+
```

Oracle mode:

```
SELECT RTRIM(''),RTRIM_ORACLE('');
+-----------+------------------+
| RTRIM('') | RTRIM_ORACLE('') |
+-----------+------------------+
|           | NULL             |
+-----------+------------------+
```

See Also

- LTRIM - leading spaces removed
- TRIM - removes all given prefixes or suffixes

This page is licensed: GPLv2, originally from fill_help_tables.sql

SFORMAT

> **Note:**
>
> `SFORMAT` is available from MariaDB 10.7.

Description

The `SFORMAT` function takes an input string and a formatting specification and returns the string formatted using the rules the user passed in the specification.

It uses the fmtlib library for Python-like (as well as Rust, C++20, etc) string formatting.

Only fmtlib 7.0.0+ is supported.

There is no native support for temporal and decimal values:

- `TIME_RESULT` is handled as `STRING_RESULT`.
- `DECIMAL_RESULT` is handled as `REAL_RESULT`.

Examples

```
SELECT SFORMAT("The answer is {}.", 42);
+---------------------------------+
| SFORMAT("The answer is {}.", 42) |
+---------------------------------+
| The answer is 42.               |
+---------------------------------+

CREATE TABLE test_sformat(mdb_release char(6), mdev int, feature char(20));

INSERT INTO test_sformat VALUES('10.7.0', 25015, 'Python style sformat'),
  ('10.7.0', 4958, 'UUID');

SELECT * FROM test_sformat;
+-------------+-------+----------------------+
| mdb_release | mdev  | feature              |
+-------------+-------+----------------------+
| 10.7.0      | 25015 | Python style sformat |
| 10.7.0      |  4958 | UUID                 |
+-------------+-------+----------------------+

SELECT SFORMAT('MariaDB Server {} has a preview for MDEV-{} which is about {}',
  mdb_release, mdev, feature) AS 'Preview Release Examples'
  FROM test_sformat;
+-----------------------------------------------------------------------------------------+
| Preview Release Examples                                                                |
+-----------------------------------------------------------------------------------------+
| MariaDB Server 10.7.0 has a preview for MDEV-25015 which is about Python style sformat |
| MariaDB Server 10.7.0 has a preview for MDEV-4958 which is about UUID                   |
+-----------------------------------------------------------------------------------------+
```

See Also

- 10.7 preview feature: Python-like string formatting

This page is licensed: CC BY-SA / Gnu FDL

SOUNDEX

Syntax

```
SOUNDEX(str)
```

Description

Returns a soundex string from *str*. Two strings that sound almost the same should have identical soundex strings. A standard soundex string is four characters long, but the `SOUNDEX()` function returns an arbitrarily long string. You can use `SUBSTRING()` on the result to get a standard soundex string. All non-alphabetic characters in *str* are ignored. All international alphabetic characters outside the A-Z range are treated as vowels.

Important: When using `SOUNDEX()`, you should be aware of the following details:

- This function, as currently implemented, is intended to work well with strings that are in the English language only. Strings in other languages may not produce reasonable results.
- This function implements the original Soundex algorithm, not the more popular enhanced version (also described by D. Knuth). The difference is that original version discards vowels first and duplicates second, whereas the enhanced version discards duplicates first and vowels second.

Examples

```
SOUNDEX('Hello');
+------------------+
| SOUNDEX('Hello') |
+------------------+
| H400             |
+------------------+
```

```
SELECT SOUNDEX('MariaDB');
+--------------------+
| SOUNDEX('MariaDB') |
+--------------------+
| M631               |
+--------------------+
```

```
SELECT SOUNDEX('Knowledgebase');
+--------------------------+
| SOUNDEX('Knowledgebase') |
+--------------------------+
| K543212                  |
+--------------------------+
```

```
SELECT givenname, surname FROM users WHERE SOUNDEX(givenname) = SOUNDEX("robert");
+-----------+---------+
| givenname | surname |
+-----------+---------+
| Roberto   | Castro  |
+-----------+---------+
```

See Also

- SOUNDS LIKE()

This page is licensed: GPLv2, originally from fill_help_tables.sql

SOUNDS LIKE

Syntax

```
expr1 SOUNDS LIKE expr2
```

Description

This is the same as SOUNDEX(expr1) = `SOUNDEX(expr2)`.

Example

```
SELECT givenname, surname FROM users WHERE givenname SOUNDS LIKE "robert";
+-----------+---------+
| givenname | surname |
+-----------+---------+
| Roberto   | Castro  |
+-----------+---------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SPACE

Syntax

```
SPACE(N)
```

Description

Returns a string consisting of `N` space characters. If `N` is `NULL`, returns `NULL`.

Examples

```
SELECT QUOTE(SPACE(6));
+-----------------+
| QUOTE(SPACE(6)) |
+-----------------+
| '      '        |
+-----------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

STRCMP

Syntax

```
STRCMP(expr1,expr2)
```

Description

`STRCMP()` returns `0` if the strings are the same, `-1` if the first argument is smaller than the second according to the current sort order, and `1` if the strings are otherwise not the same. Returns `NULL` is either argument is `NULL`.

Examples

```
SELECT STRCMP('text', 'text2');
+-------------------------+
| STRCMP('text', 'text2') |
+-------------------------+
|                      -1 |
+-------------------------+

SELECT STRCMP('text2', 'text');
+-------------------------+
| STRCMP('text2', 'text') |
+-------------------------+
|                       1 |
+-------------------------+

SELECT STRCMP('text', 'text');
+------------------------+
| STRCMP('text', 'text') |
+------------------------+
|                      0 |
+------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

SUBSTR

Description

`SUBSTR()` is a synonym for SUBSTRING().

This page is licensed: GPLv2, originally from fill_help_tables.sql

SUBSTRING

Syntax

```
SUBSTRING(str,pos),
SUBSTRING(str FROM pos),
SUBSTRING(str,pos,len),
SUBSTRING(str FROM pos FOR len)

SUBSTR(str,pos),
SUBSTR(str FROM pos),
SUBSTR(str,pos,len),
SUBSTR(str FROM pos FOR len)
```

Description

The forms without a *`len`* argument return a substring from string *`str`* starting at position *`pos`*.

The forms with a *`len`* argument return a substring *`len`* characters long from string *`str`*, starting at position *`pos`*.

The forms that use *`FROM`* are standard SQL syntax.

It is also possible to use a negative value for *`pos`*. In this case, the beginning of the substring is *`pos`* characters from the end of the string, rather than the beginning. A negative value may be used for *`pos`* in any of the forms of this function.

By default, the position of the first character in the string from which the substring is to be extracted is 1. If the value of *pos* is 0, the result is empty string. For Oracle compatibility, when `sql_mode` is set to '`oracle`', position 0 is treated as position 1.

If any argument is `NULL`, returns `NULL`.

> **CURRENT**
>
> The optimizer can make use of an index for conditions like `SUBSTR(indexed_column, 1, n) = const_string`.
>
> **< 11.8**
>
> The optimizer cannot make use of an index if an indexed column is an argument of `SUBSTR()`.

Examples

```
SELECT SUBSTRING('Knowledgebase',5);
+-----------------------------+
| SUBSTRING('Knowledgebase',5) |
+-----------------------------+
| ledgebase                   |
+-----------------------------+

SELECT SUBSTRING('MariaDB' FROM 6);
+----------------------------+
| SUBSTRING('MariaDB' FROM 6) |
+----------------------------+
| DB                         |
+----------------------------+

SELECT SUBSTRING('Knowledgebase',3,7);
+-------------------------------+
| SUBSTRING('Knowledgebase',3,7) |
+-------------------------------+
| owledge                       |
+-------------------------------+

SELECT SUBSTRING('Knowledgebase', -4);
+--------------------------------+
| SUBSTRING('Knowledgebase', -4) |
+--------------------------------+
| base                           |
+--------------------------------+

SELECT SUBSTRING('Knowledgebase', 0);
+--------------------------------+
| SUBSTRING('Knowledgebase', 0)  |
+--------------------------------+
|                                |
+--------------------------------+

SELECT SUBSTRING('Knowledgebase', -8, 4);
+-----------------------------------+
| SUBSTRING('Knowledgebase', -8, 4) |
+-----------------------------------+
| edge                              |
+-----------------------------------+

SELECT SUBSTRING('Knowledgebase' FROM -8 FOR 4);
+-----------------------------------------+
| SUBSTRING('Knowledgebase' FROM -8 FOR 4) |
+-----------------------------------------+
| edge                                    |
+-----------------------------------------+
```

Oracle mode:

```
SELECT SUBSTR('abc',1,2);
+------------------+
| SUBSTR('abc',1,2) |
+------------------+
| ab                |
+------------------+

SET sql_mode='oracle';

SELECT SUBSTR('abc',0,3);
+------------------+
| SUBSTR('abc',0,3) |
+------------------+
| abc               |
+------------------+

SELECT SUBSTR('abc',1,2);
+------------------+
| SUBSTR('abc',1,2) |
+------------------+
| ab                |
+------------------+
```

See Also

- INSTR() - Returns the position of a string within a string
- LOCATE() - Returns the position of a string within a string
- SUBSTRING_INDEX() - Returns a string based on substring

This page is licensed: GPLv2, originally from fill_help_tables.sql

SUBSTRING_INDEX

Syntax

```
SUBSTRING_INDEX(str,delim,count)
```

Description

Returns the substring from string *`str`* before count occurrences of the delimiter *`delim`*. If *`count`* is positive, everything to the left of the final delimiter (counting from the left) is returned. If *`count`* is negative, everything to the right of the final delimiter (counting from the right) is returned. `SUBSTRING_INDEX()` performs a case-sensitive match when searching for *`delim`*.

If any argument is `NULL`, returns `NULL`.

For example:

```
SUBSTRING_INDEX('www.mariadb.org', '.', 2)
```

It means “Return all of the characters up to the 2nd occurrence of .”

Examples

```
SELECT SUBSTRING_INDEX('www.mariadb.org', '.', 2);
+--------------------------------------------+
| SUBSTRING_INDEX('www.mariadb.org', '.', 2) |
+--------------------------------------------+
| www.mariadb                                |
+--------------------------------------------+

SELECT SUBSTRING_INDEX('www.mariadb.org', '.', -2);
+---------------------------------------------+
| SUBSTRING_INDEX('www.mariadb.org', '.', -2) |
+---------------------------------------------+
| mariadb.org                                 |
+---------------------------------------------+
```

See Also

- INSTR() - Returns the position of a string within a string
- LOCATE() - Returns the position of a string within a string
- SUBSTRING() - Returns a string based on position

This page is licensed: GPLv2, originally from fill_help_tables.sql

TO_BASE64

Syntax

```
TO_BASE64(str)
```

Description

Converts the string argument `str` to its base-64 encoded form, returning the result as a character string in the connection character set and collation.

The argument `str` will be converted to string first if it is not a string. A `NULL` argument will return a `NULL` result.

The reverse function, FROM_BASE64(), decodes an encoded base-64 string.

There are a numerous different methods to base-64 encode a string. The following are used by MariaDB and MySQL:

- Alphabet value 64 is encoded as '+'.
- Alphabet value 63 is encoded as '/'.
- Encoding output is made up of groups of four printable characters, with each three bytes of data encoded using four characters. If the final group is not complete, it is padded with '=' characters to make up a length of four.
- To divide long output, a newline is added after every 76 characters.
- Decoding will recognize and ignore newlines, carriage returns, tabs, and spaces.

Examples

```
SELECT TO_BASE64('Maria');
+--------------------+
| TO_BASE64('Maria') |
+--------------------+
| TWFyaWE=           |
+--------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

TO_CHAR

> **Note:**
>
> `TO_CHAR` is available from MariaDB 10.6.

Syntax

```
TO_CHAR(expr[, fmt])
```

Description

CURRENT

The `TO_CHAR` function converts an *expr* of type date, datetime, time or timestamp to a string. The optional *fmt* argument supports `YYY/YYYY/YY/RRRR/RR/MM/MON/MONTH/MI/DD/DY/HH/HH12/HH24/SS` and special characters. The default value is `YYYY-MM-DD HH24:MI:SS`. `TO_CHAR` also accepts `FM` in the format string, which disables padding of all components following it.

`FM` can be specified multiple times, with each time disabling the previous state:

- An odd number of `FM`s disables padding.
- An even number of `FM`s enables padding.

These additional formats (for *`fmt`*) are available:

- `FF[1-6]` – Fractional seconds
- `DDD` – Day (1-366)
- `IW` – Week 1-53 according to ISO 8601
- `I` – 1-digit year according to ISO 8601
- `IY` – 2-digit year according to ISO 8601
- `IYY` – 3-digit year according to ISO 8601
- `IYYY` – 4-digit year according to ISO 8601
- `SYYY` – 4-digit year according to ISO 8601 (Oracle can use signed years)

< 12.3

The `TO_CHAR` function converts an *expr* of type date, datetime, time or timestamp to a string. The optional *fmt* argument supports `YYY/YYYY/YY/RRRR/RR/MM/MON/MONTH/MI/DD/DY/HH/HH12/HH24/SS` and special characters. The default value is `YYYY-MM-DD HH24:MI:SS`. `TO_CHAR` also accepts `FM` in the format string, which disables padding of all components following it.

`FM` can be specified multiple times, with each time disabling the previous state:

- An odd number of `FM`s disables padding.
- An even number of `FM`s enables padding.

< 12.0

The `TO_CHAR` function converts an *expr* of type date, datetime, time or timestamp to a string. The optional *`fmt`* argument supports `YYY/YYYY/YY/RRRR/RR/MM/MON/MONTH/MI/DD/DY/HH/HH12/HH24/SS` and special characters. The default value is `YYYY-MM-DD HH24:MI:SS`.

In Oracle, `TO_CHAR` can also be used to convert numbers to strings, but this is not supported in MariaDB and gives an error.

Examples

CURRENT

```
SELECT TO_CHAR('1980-01-11 04:50:39', 'YYYY-MM-DD');
+-------------------------------------------+
| TO_CHAR('1980-01-11 04:50:39', 'YYYY-MM-DD') |
+-------------------------------------------+
| 1980-01-11                                |
+-------------------------------------------+

SELECT TO_CHAR('1980-01-11 04:50:39', 'HH24-MI-SS');
+-------------------------------------------+
| TO_CHAR('1980-01-11 04:50:39', 'HH24-MI-SS') |
+-------------------------------------------+
| 04-50-39                                  |
+-------------------------------------------+

SELECT TO_CHAR('00-01-01 00:00:00', 'YY-MM-DD HH24:MI:SS');
+-----------------------------------------------------+
| TO_CHAR('00-01-01 00:00:00', 'YY-MM-DD HH24:MI:SS') |
+-----------------------------------------------------+
| 00-01-01 00:00:00                                   |
+-----------------------------------------------------+

SELECT TO_CHAR('99-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS');
+-----------------------------------------------------+
| TO_CHAR('99-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS') |
+-----------------------------------------------------+
| 99-12-31 23:59:59                                   |
+-----------------------------------------------------+

SELECT TO_CHAR('9999-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS');
+-----------------------------------------------------+
| TO_CHAR('9999-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS') |
+-----------------------------------------------------+
| 99-12-31 23:59:59                                    |
+-----------------------------------------------------+

SELECT TO_CHAR('21-01-03 08:30:00', 'Y-MONTH-DY HH:MI:SS');
+-----------------------------------------------------+
| TO_CHAR('21-01-03 08:30:00', 'Y-MONTH-DY HH:MI:SS') |
+-----------------------------------------------------+
| 1-January  -Sun 08:30:00                            |
+-----------------------------------------------------+

From MariaDB 12.0, FM removes following padding:

SELECT CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'DAY'), '/');
+-----------------------------------------------------------+
| CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'DAY'), '/') |
+-----------------------------------------------------------+
| /Monday   /                                              |
+-----------------------------------------------------------+

SELECT CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'FMDAY'), '/');
+-------------------------------------------------------------+
| CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'FMDAY'), '/') |
+-------------------------------------------------------------+
| /Monday/                                                   |
+-------------------------------------------------------------+

Even numbers of FM enable padding, while odd numbers disable it:

SELECT CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'FMFMDAY'), '/');
+-------------------------------------------------------------+
| CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'FMFMDAY'), '/') |
+-------------------------------------------------------------+
| /Monday   /                                                |
+-------------------------------------------------------------+

SELECT CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'FMFMFMDAY'), '/');
```

```
+-------------------------------------------------------------+
| CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'FMFMFMDAY'), '/') |
+-------------------------------------------------------------+
| /Monday/                                                    |
+-------------------------------------------------------------+

FM only suppresses following padding:

SELECT CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'DAYFM'), '/');
+-------------------------------------------------------+
| CONCAT('/', TO_CHAR('2020-01-06 10:11:12', 'DAYFM'), '/') |
+-------------------------------------------------------+
| /Monday   /                                           |
+-------------------------------------------------------+
```

< 12.0

```
SELECT TO_CHAR('1980-01-11 04:50:39', 'YYYY-MM-DD');
+-----------------------------------------+
| TO_CHAR('1980-01-11 04:50:39', 'YYYY-MM-DD') |
+-----------------------------------------+
| 1980-01-11                              |
+-----------------------------------------+

SELECT TO_CHAR('1980-01-11 04:50:39', 'HH24-MI-SS');
+-----------------------------------------+
| TO_CHAR('1980-01-11 04:50:39', 'HH24-MI-SS') |
+-----------------------------------------+
| 04-50-39                                |
+-----------------------------------------+

SELECT TO_CHAR('00-01-01 00:00:00', 'YY-MM-DD HH24:MI:SS');
+-----------------------------------------------------+
| TO_CHAR('00-01-01 00:00:00', 'YY-MM-DD HH24:MI:SS') |
+-----------------------------------------------------+
| 00-01-01 00:00:00                                   |
+-----------------------------------------------------+

SELECT TO_CHAR('99-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS');
+-----------------------------------------------------+
| TO_CHAR('99-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS') |
+-----------------------------------------------------+
| 99-12-31 23:59:59                                   |
+-----------------------------------------------------+

SELECT TO_CHAR('9999-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS');
+-----------------------------------------------------+
| TO_CHAR('9999-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS') |
+-----------------------------------------------------+
| 99-12-31 23:59:59                                    |
+-----------------------------------------------------+

SELECT TO_CHAR('21-01-03 08:30:00', 'Y-MONTH-DY HH:MI:SS');
+-----------------------------------------------------+
| TO_CHAR('21-01-03 08:30:00', 'Y-MONTH-DY HH:MI:SS') |
+-----------------------------------------------------+
| 1-January  -Sun 08:30:00                            |
+-----------------------------------------------------+
```

See Also

- SQL_MODE=ORACLE

This page is licensed: CC BY-SA / Gnu FDL

TRIM

Syntax

```
TRIM_ORACLE([{BOTH | LEADING | TRAILING} [remstr] FROM] str), TRIM([remstr FROM] str)
```

Description

Returns the string `str` with all `remstr` prefixes or suffixes removed. If none of the specifiers `BOTH`, `LEADING`, or `TRAILING` is given, `BOTH` is assumed. `remstr` is optional and, if not specified, spaces are removed.

Returns `NULL` if given a `NULL` argument. If the result is empty, returns either an empty string, or, with SQL_MODE=Oracle, `NULL`. `SQL_MODE=Oracle` is not set by default.

The Oracle mode version of the function can be accessed in any mode by using `TRIM_ORACLE` as the function name.

Examples

```
SELECT TRIM('  bar   ')\G
*************************** 1. row ***************************
TRIM('  bar   '): bar

SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx')\G
*************************** 1. row ***************************
TRIM(LEADING 'x' FROM 'xxxbarxxx'): barxxx

SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx')\G
*************************** 1. row ***************************
TRIM(BOTH 'x' FROM 'xxxbarxxx'): bar

SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz')\G
*************************** 1. row ***************************
TRIM(TRAILING 'xyz' FROM 'barxxyz'): barx
```

With SQL_MODE=Oracle not set:

```
SELECT TRIM(''),TRIM_ORACLE('');
+----------+-----------------+
| TRIM('') | TRIM_ORACLE('') |
+----------+-----------------+
|          | NULL            |
+----------+-----------------+
```

With SQL_MODE=Oracle set:

```
SELECT TRIM(''),TRIM_ORACLE('');
+----------+-----------------+
| TRIM('') | TRIM_ORACLE('') |
+----------+-----------------+
| NULL     | NULL            |
+----------+-----------------+
```

See Also

- LTRIM - leading spaces removed
- RTRIM - trailing spaces removed

This page is licensed: GPLv2, originally from fill_help_tables.sql

TRIM_ORACLE

`TRIM_ORACLE` is a synonym for the Oracle mode version of the TRIM function, and is available in all modes.

This page is licensed: CC BY-SA / Gnu FDL

Type Conversion

Implicit type conversion takes place when MariaDB is using operands or different types, in order to make the operands compatible.

It is best practice not to rely upon implicit conversion; rather use CAST to explicitly convert types.

Rules for Conversion on Comparison

- If either argument is NULL, the result of the comparison is NULL unless the NULL-safe <=> equality comparison operator is used.
- If both arguments are integers, they are compared as integers.
- If both arguments are strings, they are compared as strings.
- If one argument is decimal and the other argument is decimal or integer, they are compared as decimals.
- If one argument is decimal and the other argument is a floating point, they are compared as floating point values.
- If one argument is string and the other argument is integer, they are compared as decimals.
- If a hexadecimal argument is not compared to a number, it is treated as a binary string.
- If a constant is compared to a TIMESTAMP or DATETIME, the constant is converted to a timestamp, unless used as an argument to the IN function.
- In other cases, arguments are compared as floating point, or real, numbers.

Note that if a string column is being compared with a numeric value, MariaDB will not use the index on the column, as there are numerous alternatives that may evaluate as equal (see examples below).

Comparison Examples

Converting a string to a number:

```
SELECT 15+'15';
+---------+
| 15+'15' |
+---------+
|      30 |
+---------+
```

Converting a number to a string:

```
SELECT CONCAT(15,'15');
+-----------------+
| CONCAT(15,'15') |
+-----------------+
| 1515            |
+-----------------+
```

Floating point number errors:

```
SELECT '9746718491924563214' = 9746718491924563213;
+-------------------------------------------+
| '9746718491924563214' = 9746718491924563213 |
+-------------------------------------------+
|                                         1 |
+-------------------------------------------+
```

Numeric equivalence with strings:

```
SELECT '5' = 5;
+---------+
| '5' = 5 |
+---------+
|       1 |
+---------+

SELECT '   5' = 5;
+------------+
| '   5' = 5 |
+------------+
|          1 |
+------------+

SELECT '   5  ' = 5;
+--------------+
| '   5  ' = 5 |
+--------------+
|            1 |
+--------------+
1 row in set, 1 warning (0.000 sec)

SHOW WARNINGS;
+-------+------+-------------------------------------------+
| Level | Code | Message                                   |
+-------+------+-------------------------------------------+
| Note  | 1292 | Truncated incorrect DOUBLE value: '   5  ' |
+-------+------+-------------------------------------------+
```

As a result of the above, MariaDB cannot use the index when comparing a string with a numeric value in the example below:

```
CREATE TABLE t (a VARCHAR(10), b VARCHAR(10), INDEX idx_a (a));

INSERT INTO t VALUES
  ('1', '1'), ('2', '2'), ('3', '3'),
  ('4', '4'), ('5', '5'), ('1', '5');

EXPLAIN SELECT * FROM t WHERE a = '3' \G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t
         type: ref
possible_keys: idx_a
          key: idx_a
      key_len: 13
          ref: const
         rows: 1
        Extra: Using index condition

EXPLAIN SELECT * FROM t WHERE a = 3 \G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t
         type: ALL
possible_keys: idx_a
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 6
        Extra: Using where
```

Rules for Conversion on Dyadic Arithmetic Operations

Implicit type conversion also takes place on dyadic arithmetic operations (+,-,*,/). MariaDB chooses the minimum data type that is guaranteed to fit the result and converts both arguments to the result data type.

For addition (+), subtraction (-) and multiplication (*), the result data type is chosen as follows:

- If either of the arguments is an approximate number (float, double), the result is double.
- If either of the arguments is a string (char, varchar, text), the result is double.
- If either of the arguments is a decimal number, the result is decimal.
- If either of the arguments is of a temporal type with a non-zero fractional second precision (time(N), datetime(N), timestamp(N)), the result is decimal.
- If either of the arguments is of a temporal type with a zero fractional second precision (time(0), date, datetime(0), timestamp(0)), the result may vary between int, int unsigned, bigint or bigint unsigned, depending on the exact data type combination.
- If both arguments are integer numbers (tinyint, smallint, mediumint, bigint), the result may vary between int, int unsigned, bigint or bigint unsigned, depending of the exact data types and their signs.

For division (/), the result data type is chosen as follows:

- If either of the arguments is an approximate number (float, double), the result is double.
- If either of the arguments is a string (char, varchar, text), the result is double.
- Otherwise, the result is decimal.

Arithmetic Examples

Note, the above rules mean that when an argument of a temporal data type appears in addition or subtraction, it's treated as a number by default.

```
SELECT TIME'10:20:30' + 1;
+--------------------+
| TIME'10:20:30' + 1 |
+--------------------+
|             102031 |
+--------------------+
```

In order to do temporal addition or subtraction instead, use the DATE_ADD() or DATE_SUB() functions, or an INTERVAL expression as the second argument:

```
SELECT TIME'10:20:30' + INTERVAL 1 SECOND;
+-----------------------------------+
| TIME'10:20:30' + INTERVAL 1 SECOND |
+-----------------------------------+
| 10:20:31                          |
+-----------------------------------+
```

```
SELECT "2.2" + 3;
+-----------+
| "2.2" + 3 |
+-----------+
|       5.2 |
+-----------+

SELECT 2.2 + 3;
+---------+
| 2.2 + 3 |
+---------+
| 5.2     |
+---------+

SELECT 2.2 / 3;
+---------+
| 2.2 / 3 |
+---------+
| 0.73333 |
+---------+

SELECT "2.2" / 3;
+--------------------+
| "2.2" / 3          |
+--------------------+
| 0.7333333333333334 |
+--------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

UCASE

Syntax

```
UCASE(str)
```

Description

`UCASE()` is a synonym for UPPER().

This page is licensed: GPLv2, originally from fill_help_tables.sql

UNHEX

Syntax

```
UNHEX(str)
```

Description

Performs the inverse operation of HEX(str). That is, it interprets each pair of hexadecimal digits in the argument as a number and converts it to the character represented by the number. The resulting characters are returned as a binary string.

If `str` is `NULL`, `UNHEX()` returns `NULL`.

Examples

```
SELECT HEX('MariaDB');
+----------------+
| HEX('MariaDB') |
+----------------+
| 4D617269614442 |
+----------------+

SELECT UNHEX('4D617269614442');
+-------------------------+
| UNHEX('4D617269614442') |
+-------------------------+
| MariaDB                 |
+-------------------------+

SELECT 0x4D617269614442;
+------------------+
| 0x4D617269614442 |
+------------------+
| MariaDB          |
+------------------+

SELECT UNHEX(HEX('string'));
+----------------------+
| UNHEX(HEX('string')) |
+----------------------+
| string               |
+----------------------+

SELECT HEX(UNHEX('1267'));
+--------------------+
| HEX(UNHEX('1267')) |
+--------------------+
| 1267               |
+--------------------+
```

See Also

- Hexadecimal literals
- HEX()
- CONV()

This page is licensed: GPLv2, originally from fill_help_tables.sql

UPDATEXML

Syntax

```
UpdateXML(xml_target, xpath_expr, new_xml)
```

Description

This function replaces a single portion of a given fragment of XML markup `xml_target` with a new XML fragment `new_xml`, and then returns the changed XML. The portion of `xml_target` that is replaced matches an XPath expression `xpath_expr` supplied by the user. If no expression matching `xpath_expr` is found, or if multiple matches are found, the function returns the original `xml_target` XML fragment. All three arguments should be strings.

Examples

```
SELECT
    UpdateXML('<a><b>ccc</b><d></d></a>', '/a', '<e>fff</e>') AS val1,
    UpdateXML('<a><b>ccc</b><d></d></a>', '/b', '<e>fff</e>') AS val2,
    UpdateXML('<a><b>ccc</b><d></d></a>', '//b', '<e>fff</e>') AS val3,
    UpdateXML('<a><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val4,
    UpdateXML('<a><d></d><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val5
    \G
*************************** 1. row ***************************
val1: <e>fff</e>
val2: <a><b>ccc</b><d></d></a>
val3: <a><e>fff</e><d></d></a>
val4: <a><b>ccc</b><e>fff</e></a>
val5: <a><d></d><b>ccc</b><d></d></a>
1 row in set (0.00 sec)
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

UPPER

Syntax

```
UPPER(str)
UCASE(str)
```

Description

Returns the string `str` with all characters changed to uppercase according to the current character set mapping. The default is latin1 (cp1252 West European). The result may depend on the collation and character set used. `UCASE` is a synonym.

```
SELECT UPPER(surname), givenname FROM users ORDER BY surname;
+----------------+------------+
| UPPER(surname) | givenname  |
+----------------+------------+
| ABEL           | Jacinto    |
| CASTRO         | Robert     |
| COSTA          | Phestos    |
| MOSCHELLA      | Hippolytos |
+----------------+------------+
```

`UPPER()` is ineffective when applied to binary strings (BINARY, VARBINARY, BLOB). The description of LOWER() shows how to perform lettercase conversion of binary strings.

This page is licensed: GPLv2, originally from fill_help_tables.sql

WEIGHT_STRING

Syntax

```
WEIGHT_STRING(str [AS {CHAR|BINARY}(N)] [LEVEL levels] [flags])
  levels: N [ASC|DESC|REVERSE] [, N [ASC|DESC|REVERSE]] ...
```

Description

Returns a binary string representing the string's sorting and comparison value. A string with a lower result means that for sorting purposes the string appears before a string with a higher result.

`WEIGHT_STRING()` is particularly useful when adding new collations, for testing purposes.

If `str` is a non-binary string (CHAR, VARCHAR or TEXT), WEIGHT_STRING returns the string's collation weight. If `str` is a binary string (BINARY, VARBINARY or BLOB), the return value is simply the input value, since the weight for each byte in a binary string is the byte value.

`WEIGHT_STRING()` returns `NULL` if given a `NULL` input.

The optional AS clause permits casting the input string to a binary or non-binary string, as well as to a particular length.

`AS BINARY(N)` measures the length in bytes rather than characters, and right pads with 0x00 bytes to the desired length.

`AS CHAR(N)` measures the length in characters, and right pads with spaces to the desired length.

N has a minimum value of 1, and if it is less than the length of the input string, the string is truncated without warning.

The optional `LEVEL` clause specifies that the return value should contain weights for specific collation levels. The `levels` specifier can either be a single integer, a comma-separated list of integers, or a range of integers separated by a dash (whitespace is ignored). Integers can range from 1 to a maximum of 6, dependent on the collation, and need to be listed in ascending order.

If the `LEVEL` clause is no provided, a default of 1 to the maximum for the collation is assumed.

If the `q` is specified without using a range, an optional modifier is permitted.

`ASC`, the default, returns the weights without any modification.

`DESC` returns bitwise-inverted weights.

`REVERSE` returns the weights in reverse order.

Examples

The examples below use the HEX() function to represent non-printable results in hexadecimal format.

```
SELECT HEX(WEIGHT_STRING('x'));
+-------------------------+
| HEX(WEIGHT_STRING('x')) |
+-------------------------+
| 0058                    |
+-------------------------+

SELECT HEX(WEIGHT_STRING('x' AS BINARY(4)));
+--------------------------------------+
| HEX(WEIGHT_STRING('x' AS BINARY(4))) |
+--------------------------------------+
| 78000000                             |
+--------------------------------------+

SELECT HEX(WEIGHT_STRING('x' AS CHAR(4)));
+-----------------------------------+
| HEX(WEIGHT_STRING('x' AS CHAR(4))) |
+-----------------------------------+
| 0058002000200020                  |
+-----------------------------------+

SELECT HEX(WEIGHT_STRING(0xaa22ee LEVEL 1));
+--------------------------------------+
| HEX(WEIGHT_STRING(0xaa22ee LEVEL 1)) |
+--------------------------------------+
| AA22EE                               |
+--------------------------------------+

SELECT HEX(WEIGHT_STRING(0xaa22ee LEVEL 1 DESC));
+-----------------------------------------+
| HEX(WEIGHT_STRING(0xaa22ee LEVEL 1 DESC)) |
+-----------------------------------------+
| 55DD11                                  |
+-----------------------------------------+

SELECT HEX(WEIGHT_STRING(0xaa22ee LEVEL 1 REVERSE));
+-------------------------------------------+
| HEX(WEIGHT_STRING(0xaa22ee LEVEL 1 REVERSE)) |
+-------------------------------------------+
| EE22AA                                    |
+-------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Regular Expressions Functions

See Regular Expressions Overview.

Get an overview of regex usage. This page introduces the pattern matching capabilities and common metacharacters used in MariaDB regular expressions.

See PCRE - Perl Compatible Regular Expressions.

Understand MariaDB's regex support. This concept page explains the PCRE library integration, detailing supported syntax, character classes, and special characters.

See REGEXP.

Complete REGEXP operator reference: expr REGEXP/RLIKE pat syntax, 1/0/NULL return values, NOT REGEXP aliases, BINARY case-sensitivity, default_regex_flags.

See NOT REGEXP.

Negated regular expression matching. This operator tests whether a string does NOT match a specified regular expression pattern.

See REGEXP_INSTR.

Return the index of a regex match. This function finds the starting position of the first substring that matches the given pattern.

See REGEXP_REPLACE.

Replace regex matches in a string. This function substitutes occurrences of a pattern with a specified replacement string.

See REGEXP_SUBSTR.

Return the substring matching a regex. This function extracts the actual part of the string that matches the given pattern.

See RLIKE.

Synonym for REGEXP. This operator performs a regular expression match against a string argument.

This page is licensed: CC BY-SA / Gnu FDL

Regular Expressions Overview

Regular Expressions allow MariaDB to perform complex pattern matching on a string. In many cases, the simple pattern matching provided by LIKE is sufficient. `LIKE` performs two kinds of matches:

- `_` - the underscore, matching a single character
- `%` - the percentage sign, matching any number of characters.

In other cases you may need more control over the returned matches, and will need to use regular expressions.

Regular expression matches are performed with the REGEXP function. `RLIKE` is a synonym for `REGEXP`.

Comparisons are performed on the byte value, so characters that are treated as equivalent by a collation, but do not have the same byte-value, such as accented characters, could evaluate as unequal.

Without any special characters, a regular expression match is true if the characters match. The match is case-insensitive, except in the case of BINARY strings.

```
SELECT 'Maria' REGEXP 'Maria';
+------------------------+
| 'Maria' REGEXP 'Maria' |
+------------------------+
|                      1 |
+------------------------+

SELECT 'Maria' REGEXP 'maria';
+------------------------+
| 'Maria' REGEXP 'maria' |
+------------------------+
|                      1 |
+------------------------+

SELECT BINARY 'Maria' REGEXP 'maria';
+-------------------------------+
| BINARY 'Maria' REGEXP 'maria' |
+-------------------------------+
|                             0 |
+-------------------------------+
```

Note that the word being matched must match the whole pattern:

```
SELECT 'Maria' REGEXP 'Mari';
+----------------------+
| 'Maria' REGEXP 'Mari' |
+----------------------+
|                    1 |
+----------------------+

SELECT 'Mari' REGEXP 'Maria';
+----------------------+
| 'Mari' REGEXP 'Maria' |
+----------------------+
|                    0 |
+----------------------+
```

The first returns true because the pattern “Mari” exists in the expression “Maria”. When the order is reversed, the result is false, as the pattern “Maria” does not exist in the expression “Mari”

A match can be performed against more than one word with the `|` character. For example:

```
SELECT 'Maria' REGEXP 'Monty|Maria';
+-----------------------------+
| 'Maria' REGEXP 'Monty|Maria' |
+-----------------------------+
|                           1 |
+-----------------------------+
```

Special Characters

The above examples introduce the syntax, but are not very useful on their own. It’s the special characters that give regular expressions their power.

^

`^` matches the beginning of a string (inside square brackets it can also mean `NOT` - see below):

```
SELECT 'Maria' REGEXP '^Ma';
+---------------------+
| 'Maria' REGEXP '^Ma' |
+---------------------+
|                   1 |
+---------------------+
```

$

`$` matches the end of a string:

```
SELECT 'Maria' REGEXP 'ia$';
+---------------------+
| 'Maria' REGEXP 'ia$' |
+---------------------+
|                   1 |
+---------------------+
```

.

`.` matches any single character:

```
SELECT 'Maria' REGEXP 'Ma.ia';
+-----------------------+
| 'Maria' REGEXP 'Ma.ia' |
+-----------------------+
|                     1 |
+-----------------------+

SELECT 'Maria' REGEXP 'Ma..ia';
+------------------------+
| 'Maria' REGEXP 'Ma..ia' |
+------------------------+
|                      0 |
+------------------------+
```

*****

`x*` matches zero or more of a character `x`. In the examples below, it's the `r` character.

```
SELECT 'Maria' REGEXP 'Mar*ia';
+------------------------+
| 'Maria' REGEXP 'Mar*ia' |
+------------------------+
|                      1 |
+------------------------+

SELECT 'Maia' REGEXP 'Mar*ia';
+-----------------------+
| 'Maia' REGEXP 'Mar*ia' |
+-----------------------+
|                     1 |
+-----------------------+

SELECT 'Marrria' REGEXP 'Mar*ia';
+--------------------------+
| 'Marrria' REGEXP 'Mar*ia' |
+--------------------------+
|                        1 |
+--------------------------+
```

+

`x+` matches one or more of a character `x`. In the examples below, it's the `r` character.

```
SELECT 'Maria' REGEXP 'Mar+ia';
+------------------------+
| 'Maria' REGEXP 'Mar+ia' |
+------------------------+
|                      1 |
+------------------------+

SELECT 'Maia' REGEXP 'Mar+ia';
+-----------------------+
| 'Maia' REGEXP 'Mar+ia' |
+-----------------------+
|                     0 |
+-----------------------+

SELECT 'Marrria' REGEXP 'Mar+ia';
+--------------------------+
| 'Marrria' REGEXP 'Mar+ia' |
+--------------------------+
|                        1 |
+--------------------------+
```

?

`x?` matches zero or one of a character `x`. In the examples below, it's the `r` character.

```
SELECT 'Maria' REGEXP 'Mar?ia';
+-------------------------+
| 'Maria' REGEXP 'Mar?ia' |
+-------------------------+
|                       1 |
+-------------------------+

SELECT 'Maia' REGEXP 'Mar?ia';
+------------------------+
| 'Maia' REGEXP 'Mar?ia' |
+------------------------+
|                      1 |
+------------------------+

SELECT 'Marrria' REGEXP 'Mar?ia';
+--------------------------+
| 'Marrria' REGEXP 'Mar?ia' |
+--------------------------+
|                        0 |
+--------------------------+
```

()

`(xyz)` - combine a sequence, for example `(xyz)+` or `(xyz)*`

```
SELECT 'Maria' REGEXP '(ari)+';
+-------------------------+
| 'Maria' REGEXP '(ari)+' |
+-------------------------+
|                       1 |
+-------------------------+
```

`x{n}` and `x{m,n}`
This notation is used to match many instances of the `x`. In the case of `x{n}` the match must be exactly that many times. In the case of `x{m,n}`, the match can occur from `m` to `n` times. For example, to match zero or one instance of the string `ari` (which is identical to `(ari)?`), the following can be used:

```
SELECT 'Maria' REGEXP '(ari){0,1}';
+-----------------------------+
| 'Maria' REGEXP '(ari){0,1}' |
+-----------------------------+
|                           1 |
+-----------------------------+
```

[]

`[xy]` groups characters for matching purposes. For example, to match either the `p` or the `r` character:

```
SELECT 'Maria' REGEXP 'Ma[pr]ia';
+---------------------------+
| 'Maria' REGEXP 'Ma[pr]ia' |
+---------------------------+
|                         1 |
+---------------------------+
```

The square brackets also permit a range match, for example, to match any character from a-z, `[a-z]` is used. Numeric ranges are also permitted.

```
SELECT 'Maria' REGEXP 'Ma[a-z]ia';
+---------------------------+
| 'Maria' REGEXP 'Ma[a-z]ia' |
+---------------------------+
|                         1 |
+---------------------------+
```

The following does not match, as `r` falls outside of the range `a-p`.

```
SELECT 'Maria' REGEXP 'Ma[a-p]ia';
+---------------------------+
| 'Maria' REGEXP 'Ma[a-p]ia' |
+---------------------------+
|                         0 |
+---------------------------+
```

^

The `^` character means does `NOT` match, for example:

```
SELECT 'Maria' REGEXP 'Ma[^p]ia';
+---------------------------+
| 'Maria' REGEXP 'Ma[^p]ia' |
+---------------------------+
|                         1 |
+---------------------------+

SELECT 'Maria' REGEXP 'Ma[^r]ia';
+---------------------------+
| 'Maria' REGEXP 'Ma[^r]ia' |
+---------------------------+
|                         0 |
+---------------------------+
```

The `[` and `]` characters on their own can be literally matched inside a `[]` block, without escaping, as long as they immediately match the opening bracket:

```
SELECT '[Maria' REGEXP '[[]';
+-----------------------+
| '[Maria' REGEXP '[[]' |
+-----------------------+
|                     1 |
+-----------------------+

SELECT '[Maria' REGEXP '[]]';
+-----------------------+
| '[Maria' REGEXP '[]]' |
+-----------------------+
|                     0 |
+-----------------------+

SELECT ']Maria' REGEXP '[]]';
+-----------------------+
| ']Maria' REGEXP '[]]' |
+-----------------------+
|                     1 |
+-----------------------+

SELECT ']Maria' REGEXP '[]a]';
+-----------------------+
| ']Maria' REGEXP '[]a]' |
+-----------------------+
|                      1 |
+-----------------------+
```

Incorrect order, so no match:

```
SELECT ']Maria' REGEXP '[a]]';
+-----------------------+
| ']Maria' REGEXP '[a]]' |
+-----------------------+
|                     0 |
+-----------------------+
```

The `-` character can also be matched in the same way:

```
SELECT '-Maria' REGEXP '[1-10]';
+-------------------------+
| '-Maria' REGEXP '[1-10]' |
+-------------------------+
|                       0 |
+-------------------------+

SELECT '-Maria' REGEXP '[-1-10]';
+--------------------------+
| '-Maria' REGEXP '[-1-10]' |
+--------------------------+
|                        1 |
+--------------------------+
```

Word boundaries

The :<: and :>: patterns match the beginning and the end of a word respectively. For example:

```
SELECT 'How do I upgrade MariaDB?' REGEXP '[[:<:]]MariaDB[[:>:]]';
+-----------------------------------------------------------+
| 'How do I upgrade MariaDB?' REGEXP '[[:<:]]MariaDB[[:>:]]' |
+-----------------------------------------------------------+
|                                                         1 |
+-----------------------------------------------------------+

SELECT 'How do I upgrade MariaDB?' REGEXP '[[:<:]]Maria[[:>:]]';
+-----------------------------------------------------------+
| 'How do I upgrade MariaDB?' REGEXP '[[:<:]]Maria[[:>:]]' |
+-----------------------------------------------------------+
|                                                       0 |
+-----------------------------------------------------------+
```

Character Classes

There are a number of shortcuts to match particular preset character classes. These are matched with the `[:character_class:]` pattern (inside a `[]` set). The following character classes exist:

Character Class	Description
alnum	Alphanumeric
alpha	Alphabetic
blank	Whitespace
cntrl	Control characters
digit	Digits
graph	Graphic characters
lower	Lowercase alphabetic
print	Graphic or space characters
punct	Punctuation

Character Class	Description
space	Space, tab, newline, and carriage return
upper	Uppercase alphabetic
xdigit	Hexadecimal digit

For example:

```
SELECT 'Maria' REGEXP 'Mar[[:alnum:]]*';
+--------------------------------+
| 'Maria' REGEXP 'Mar[:alnum:]*' |
+--------------------------------+
|                              1 |
+--------------------------------+
```

Remember that matches are by default case-insensitive, unless a binary string is used, so the following example, specifically looking for an uppercase, counter-intuitively matches a lowercase character:

```
SELECT 'Mari' REGEXP 'Mar[[:upper:]]+';
+--------------------------------+
| 'Mari' REGEXP 'Mar[[:upper:]]+' |
+--------------------------------+
|                               1 |
+--------------------------------+

SELECT BINARY 'Mari' REGEXP 'Mar[[:upper:]]+';
+---------------------------------------+
| BINARY 'Mari' REGEXP 'Mar[[:upper:]]+' |
+---------------------------------------+
|                                     0 |
+---------------------------------------+
```

Character Names

There are also number of shortcuts to match particular preset character names. These are matched with the `[.character.]` pattern (inside a `[]` set). The following character classes exist:

Name	Character
NUL	0
SOH	001
STX	002
ETX	003
EOT	004
ENQ	005
ACK	006
BEL	007
alert	007
BS	010
backspace	'\b'
HT	011
tab	'\t'
LF	012
newline	'\n'

Name	Character
VT	013
vertical-tab	‘\v’
FF	014
form-feed	‘\f’
CR	015
carriage-return	‘\r’
SO	016
SI	017
DLE	020
DC1	021
DC2	022
DC3	023
DC4	024
NAK	025
SYN	026
ETB	027
CAN	030
EM	031
SUB	032
ESC	033
IS4	034
FS	034
IS3	035
GS	035
IS2	036
RS	036
IS1	037
US	037
space	’ ’
exclamation-mark	‘!’
quotation-mark	‘“’
number-sign	‘#’
dollar-sign	‘$’
percent-sign	‘%’
ampersand	‘&’
apostrophe	‘’’
left-parenthesis	‘(’
right-parenthesis	‘)’
asterisk	‘*’

Name	Character
plus-sign	‘+’
comma	‘,’
hyphen	‘-’
hyphen-minus	‘-’
period	‘.’
full-stop	‘.’
slash	‘/’
solidus	‘/’
zero	‘0’
one	‘1’
two	‘2’
three	‘3’
four	‘4’
five	‘5’
six	‘6’
seven	‘7’
eight	‘8’
nine	‘9’
colon	‘:’
semicolon	‘;’
less-than-sign	‘<’
equals-sign	‘=’
greater-than-sign	‘>’
question-mark	‘?’
commercial-at	‘@’
left-square-bracket	‘[’
backslash	”
reverse-solidus	”
right-square-bracket	‘]’
circumflex	‘^’
circumflex-accent	‘^’
underscore	‘_’
low-line	‘_’
grave-accent	‘`’
left-brace	‘{’
left-curly-bracket	‘{’
vertical-line	’
right-brace	‘}’
right-curly-bracket	‘}’

Name	Character
tilde	”
DEL	177

For example:

```
SELECT '|' REGEXP '[[.vertical-line.]]';
+---------------------------------+
| '|' REGEXP '[[.vertical-line.]]' |
+---------------------------------+
|                               1 |
+---------------------------------+
```

Combining

The true power of regular expressions is unleashed when the above is combined, to form more complex examples. Regular expression’s reputation for complexity stems from the seeming complexity of multiple combined regular expressions, when in reality, it’s simply a matter of understanding the characters and how they apply:

The first example fails to match, as while the `Ma` matches, either `i` or `r` only matches once before the `ia` characters at the end.

```
SELECT 'Maria' REGEXP 'Ma[ir]{2}ia';
+-----------------------------+
| 'Maria' REGEXP 'Ma[ir]{2}ia' |
+-----------------------------+
|                           0 |
+-----------------------------+
```

This example matches, as either `i` or `r` match exactly twice after the `Ma`, in this case one `r` and one `i`.

```
SELECT 'Maria' REGEXP 'Ma[ir]{2}';
+---------------------------+
| 'Maria' REGEXP 'Ma[ir]{2}' |
+---------------------------+
|                         1 |
+---------------------------+
```

Escaping

With the large number of special characters, care needs to be taken to properly escape characters. Two backslash characters, `` (one for the MariaDB parser, one for the regex library), are required to properly escape a character. For example:

To match the literal `(Ma`:

```
SELECT '(Maria)' REGEXP '(Ma';
ERROR 1139 (42000): Got error 'parentheses not balanced' from regexp

SELECT '(Maria)' REGEXP '\(Ma';
ERROR 1139 (42000): Got error 'parentheses not balanced' from regexp

SELECT '(Maria)' REGEXP '\\(Ma';
+-------------------------+
| '(Maria)' REGEXP '\\(Ma' |
+-------------------------+
|                       1 |
+-------------------------+
```

To match `r+`: The first two examples are incorrect, as they match `r` one or more times, not `r+`:

```
SELECT 'Mar+ia' REGEXP 'r+';
+----------------------+
| 'Mar+ia' REGEXP 'r+' |
+----------------------+
|                    1 |
+----------------------+

SELECT 'Maria' REGEXP 'r+';
+--------------------+
| 'Maria' REGEXP 'r+' |
+--------------------+
|                  1 |
+--------------------+

SELECT 'Maria' REGEXP 'r\\+';
+----------------------+
| 'Maria' REGEXP 'r\\+' |
+----------------------+
|                    0 |
+----------------------+

SELECT 'Maria' REGEXP 'r+';
+--------------------+
| 'Maria' REGEXP 'r+' |
+--------------------+
|                  1 |
+--------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

PCRE - Perl Compatible Regular Expressions

PCRE Versions

PCRE Version	Introduced	Maturity
PCRE2 10.34	MariaDB 10.5.1	Stable
PCRE 8.43	MariaDB 10.1.39	Stable
PCRE 8.42	MariaDB 10.2.15, MariaDB 10.1.33, MariaDB 10.0.35	Stable
PCRE 8.41	MariaDB 10.2.8, MariaDB 10.1.26, MariaDB 10.0.32	Stable
PCRE 8.40	MariaDB 10.2.5, MariaDB 10.1.22, MariaDB 10.0.30	Stable
PCRE 8.39	MariaDB 10.1.15, MariaDB 10.0.26	Stable
PCRE 8.38	MariaDB 10.1.10, MariaDB 10.0.23	Stable
PCRE 8.37	MariaDB 10.1.5, MariaDB 10.0.18	Stable
PCRE 8.36	MariaDB 10.1.2, MariaDB 10.0.15	Stable
PCRE 8.35	MariaDB 10.1.0, MariaDB 10.0.12	Stable
PCRE 8.34	MariaDB 10.0.8	Stable

PCRE Enhancements

MariaDB uses the PCRE library, which significantly improves the power of the REGEXP/RLIKE operator.

The switch to PCRE added a number of features, including recursive patterns, named capture, look-ahead and look-behind assertions, non-capturing groups, non-greedy quantifiers, Unicode character properties, extended syntax for characters and character classes, multi-line matching, and many other.

These functions work with regular expressions: REGEXP_REPLACE(), REGEXP_INSTR(), and REGEXP_SUBSTR().

Also, `REGEXP/RLIKE`, and the new functions, work correctly with all multi-byte character sets supported by MariaDB, including East-Asian character sets (big5, gb2313, gbk, eucjp, eucjpms, cp932, ujis, euckr), and Unicode character sets (utf8, utf8mb4, ucs2, utf16, utf16le, utf32).

New Regular Expression Functions

- REGEXP_REPLACE(subject, pattern, replace) - Replaces all occurrences of a pattern.
- REGEXP_INSTR(subject, pattern) - Position of the first appearance of a regex.
- REGEXP_SUBSTR(subject,pattern) - Returns the matching part of a string.

See the individual articles for more details and examples.

PCRE Syntax

In most cases PCRE is backward compatible with the old POSIX 1003.2 compliant regexp library (see Regular Expressions Overview), so you won't need to change your applications that use SQL queries with the REGEXP/RLIKE predicate.

This section briefly describes the most important extended PCRE features. For more details, please refer to the documentation on the PCRE site, or to the documentation which is bundled in the /pcre/doc/html/ directory of a MariaDB sources distribution. The pages pcresyntax.html and pcrepattern.html should be a good start. Regular-Expressions.Info is another good resource to learn about PCRE and regular expressions generally.

Special Characters

PCRE supports the following escape sequences to match special characters:

Sequence	Description
\a	0x07 (BEL)
\cx	"control-x", where x is any ASCII character
\e	0x1B (escape)
\f	0x0C (form feed)
	0x0A (newline)
	0x0D (carriage return)
	0x09 (TAB)
\ddd	character with octal code ddd
\xhh	character with hex code hh
\x{hhh..}	character with hex code hhh..

Note, the backslash characters (here, and in all examples in the sections below) must be escaped with another backslash, unless you're using the SQL_MODE `NO_BACKSLASH_ESCAPES`.

This example tests if a character has hex code 0x61:

```
SELECT 'a' RLIKE '\\x{61}';
-> 1
```

Character Classes

PCRE supports the standard POSIX character classes such as `alnum`, `alpha`, `blank`, `cntrl`, `digit`, `graph`, `lower`, `print`, `punct`, `space`, `upper`, `xdigit`, with the following additional classes:

Class	Description
ascii	any ASCII character (0x00..0x7F)
word	any “word” character (a letter, a digit, or an underscore)

This example checks if the string consists of ASCII characters only:

```
SELECT 'abc' RLIKE '^[[:ascii:]]+$';
-> 1
```

Generic Character Types

Generic character types complement the POSIX character classes and serve to simplify writing patterns:

Class	Description
\d	a decimal digit (same as [:digit:])
\D	a character that is not a decimal digit
\h	a horizontal white space character
\H	a character that is not a horizontal white space character
	a character that is not a new line
	a newline sequence
\s	a white space character
\S	a character that is not a white space character
\v	a vertical white space character
\V	a character that is not a vertical white space character
\w	a “word” character (same as [:word:])
\W	a “non-word” character

This example checks if the string consists of “word” characters only:

```
SELECT 'abc' RLIKE '^\\w+$';
-> 1
```

Unicode Character Properties

`\p{xx}` is a character with the `xx` property, and `\P{xx}` is a character without the `xx` property.

The property names represented by `xx` above are limited to the Unicode script names, the general category properties, and “Any”, which matches any character (including newline). Those that are not part of an identified script are lumped together as “Common”.

General Category Properties For \p and \P

Property	Description
C	Other
Cc	Control
Cf	Format
Cn	Unassigned
Co	Private use
Cs	Surrogate

Property	Description
L	Letter
Ll	Lower case letter
Lm	Modifier letter
Lo	Other letter
Lt	Title case letter
Lu	Upper case letter
L&	Ll, Lu, or Lt
M	Mark
Mc	Spacing mark
Me	Enclosing mark
Mn	Non-spacing mark
N	Number
Nd	Decimal number
Nl	Letter number
No	Other number
P	Punctuation
Pc	Connector punctuation
Pd	Dash punctuation
Pe	Close punctuation
Pf	Final punctuation
Pi	Initial punctuation
Po	Other punctuation
Ps	Open punctuation
S	Symbol
Sc	Currency symbol
Sk	Modifier symbol
Sm	Mathematical symbol
So	Other symbol
Z	Separator
Zl	Line separator
Zp	Paragraph separator
Zs	Space separator

This example checks if the string consists only of characters with property N (number):

```
SELECT '1¼①' RLIKE '^\\p{N}+$';
-> 1
```

Special Category Properties For \p and \P

Property	Description
Xan	Alphanumeric: union of properties L and N

Property	Description
Xps	POSIX space: property Z or tab, NL, VT, FF, CR
Xsp	Perl space: property Z or tab, NL, FF, CR
Xuc	A character than can be represented by a Universal Character Name
Xwd	Perl word: property Xan or underscore

The property `Xuc` matches any character that can be represented by a Universal Character Name (in C++ and other programming languages). These include `$`, `@`, ```, and all characters with Unicode code points greater than `U+00A0`, excluding the surrogates `U+D800`..`U+DFFF`.

Script Names For \p and \P

Arabic, Armenian, Avestan, Balinese, Bamum, Batak, Bengali, Bopomofo, Brahmi, Braille, Buginese, Buhid, Canadian_Aboriginal, Carian, Chakma, Cham, Cherokee, Common, Coptic, Cuneiform, Cypriot, Cyrillic, Deseret, Devanagari, Egyptian_Hieroglyphs, Ethiopic, Georgian, Glagolitic, Gothic, Greek, Gujarati, Gurmukhi, Han, Hangul, Hanunoo, Hebrew, Hiragana, Imperial_Aramaic, Inherited, Inscriptional_Pahlavi, Inscriptional_Parthian, Javanese, Kaithi, Kannada, Katakana, Kayah_Li, Kharoshthi, Khmer, Lao, Latin, Lepcha, Limbu, Linear_B, Lisu, Lycian, Lydian, Malayalam, Mandaic, Meetei_Mayek, Meroitic_Cursive, Meroitic_Hieroglyphs, Miao, Mongolian, Myanmar, New_Tai_Lue, Nko, Ogham, Old_Italic, Old_Persian, Old_South_Arabian, Old_Turkic, Ol_Chiki, Oriya, Osmanya, Phags_Pa, Phoenician, Rejang, Runic, Samaritan, Saurashtra, Sharada, Shavian, Sinhala, Sora_Sompeng, Sundanese, Syloti_Nagri, Syriac, Tagalog, Tagbanwa, Tai_Le, Tai_Tham, Tai_Viet, Takri, Tamil, Telugu, Thaana, Thai, Tibetan, Tifinagh, Ugaritic, Vai, Yi.

This example checks if the string consists only of Greek characters:

```
SELECT 'ΣΦΩ' RLIKE '^\\p{Greek}+$';
-> 1
```

Extended Unicode Grapheme Sequence

The `\X` escape sequence matches a character sequence that makes an “extended grapheme cluster”, i.e. a composite character that consists of multiple Unicode code points.

One of the examples of a composite character can be a letter followed by non-spacing accent marks. This example demonstrates that `U+0045 LATIN CAPITAL LETTER E` followed by `U+0302 COMBINING CIRCUMFLEX ACCENT` followed by `U+0323 COMBINING DOT BELOW` together form an extended grapheme cluster:

```
SELECT _ucs2 0x004503020323 RLIKE '^\\X$';
-> 1
```

See the PCRE documentation for the other types of extended grapheme clusters.

Simple Assertions

An assertion specifies a certain condition that must match at a particular point, but without consuming characters from the subject string. In addition to the standard POSIX simple assertions `^` (that matches at the beginning of a line) and `$` (that matches at the end of a line), PCRE supports a number of other assertions:

Assertion	Description
\b	matches at a word boundary
\B	matches when not at a word boundary
\A	matches at the start of the subject
\Z	matches at the end of the subject, also matches before a newline at the end of the subject

Assertion	Description
\z	matches only at the end of the subject
\G	matches at the first matching position in the subject

This example cuts a word that consists only of 3 characters from a string:

```
SELECT REGEXP_SUBSTR('---abcd---xyz---', '\\b\\w{3}\\b');
-> xyz
```

Notice that the two `\b` assertions checked the word boundaries but did not get into the matching pattern.

The `\b` assertions work well in the beginning and the end of the subject string:

```
SELECT REGEXP_SUBSTR('xyz', '\\b\\w{3}\\b');
-> xyz
```

By default, the `^` and `$` assertions have the same meaning with `\A`, `\Z`, and `\z`. However, the meanings of `^` and `$` can change in multiline mode (see below). By contrast, the meanings of `\A`, `\Z`, and `\z` are always the same; they are independent of the multiline mode.

Option Setting

A number of options that control the default match behavior can be changed within the pattern by a sequence of option letters enclosed between `(?` and `)`.

Option	Description
(?i)	case insensitive match
(?m)	multiline mode
(?s)	dotall mode (dot matches newline characters)
(?x)	extended (ignore white space)
(?U)	ungreedy (lazy) match
(?J)	allow named subpatterns with duplicate names
(?X)	extra PCRE functionality (e.g. force error on unknown escaped character)
(?-…)	unset option(s)

For example, `(?im)` sets case insensitive multiline matching.

A hyphen followed by the option letters unset the options. For example, `(?-im)` means case sensitive single line match.

A combined setting and unsetting is also possible, e.g. `(?im-sx)`.

If an option is set outside of subpattern parentheses, the option applies to the remainder of the pattern that follows the option. If an option is set inside a subpattern, it applies to the part of this subpattern that follows the option.

In this example the pattern `(?i)m((?-i)aria)db` matches the words `MariaDB`, `Mariadb`, `mariadb`, but not `MARIADB`:

```
SELECT 'MariaDB' RLIKE '(?i)m((?-i)aria)db';
-> 1

SELECT 'mariaDB' RLIKE '(?i)m((?-i)aria)db';
-> 1

SELECT 'Mariadb' RLIKE '(?i)m((?-i)aria)db';
-> 1

SELECT 'MARIADB' RLIKE '(?i)m((?-i)aria)db';
-> 0
```

This example demonstrates that the `(?x)` option makes the regexp engine ignore all white spaces in the pattern (other than in a class).

```
SELECT 'ab' RLIKE '(?x)a b';
-> 1
```

Note, putting spaces into a pattern in combination with the (?x) option can be useful to split different logical parts of a complex pattern, to make it more readable.

Multiline Matching

Multiline matching changes the meaning of `^` and `$` from “the beginning of the subject string” and “the end of the subject string” to “the beginning of any line in the subject string” and “the end of any line in the subject string” respectively.

This example checks if the subject string contains two consequent lines that fully consist of digits:

```
SELECT 'abc\n123\n456\nxyz\n' RLIKE '(?m)^\\d+\\R\\d+$';
-> 1
```

Notice the `(?m)` option in the beginning of the pattern, which switches to the multiline matching mode.

Newline Conventions

PCRE supports five line break conventions:

- `CR (\r)` - a single carriage return character
- `LF (\n)` - a single linefeed character
- `CRLF (\r\n)` - a carriage return followed by a linefeed
- any of the previous three
- any Unicode newline sequence

By default, the newline convention is set to any Unicode newline sequence, which includes:

Sequence	Description
LF	(U+000A, carriage return)
CR	(U+000D, carriage return)
CRLF	(a carriage return followed by a linefeed)
VT	(U+000B, vertical tab)
FF	(U+000C, form feed)
NEL	(U+0085, next line)
LS	(U+2028, line separator)
PS	(U+2029, paragraph separator)

The newline convention can be set by starting a pattern with one of the following sequences:

Sequence	Description
(*CR)	carriage return
(*LF)	linefeed
(*CRLF)	carriage return followed by linefeed
(*ANYCRLF)	any of the previous three
(*ANY)	all Unicode newline sequences

The newline conversion affects the `^` and `$` assertions, the interpretation of the dot metacharacter, and the behavior of .

Note, the new line convention does not affect the meaning of .

This example demonstrates that the dot metacharacter matches , because it is not a newline sequence anymore:

```
SELECT 'a\nb' RLIKE '(*CR)a.b';
-> 1
```

Newline Sequences

By default, the escape sequence matches any Unicode newline sequences.

The meaning of can be set by starting a pattern with one of the following sequences:

Sequence	Description
(*BSR_ANYCRLF)	any of CR, LF or CRLF
(*BSR_UNICODE)	any Unicode newline sequence

Comments

It's possible to include comments inside a pattern. Comments do not participate in the pattern matching. Comments start at the `(?`

sequence and continue up to the next closing parenthesis:

```
SELECT 'ab12' RLIKE 'ab(?#expect digits)12';
-> 1
```

Quoting

POSIX uses the backslash to remove a special meaning from a character. PCRE introduces a syntax to remove special meaning from a sequence of characters. The characters inside `\Q` … `\E` are treated literally, without their special meaning.

This example checks if the string matches a dollar sign followed by a parenthesized name (a variable reference in some languages):

```
SELECT '$(abc)' RLIKE '^\\Q$(\\E\\w+\\Q)\\E$';
-> 1
```

Note that the leftmost dollar sign and the parentheses are used literally, while the rightmost dollar sign is still used to match the end of the string.

Resetting the Match Start

The escape sequence `\K` causes any previously matched characters to be excluded from the final matched sequence. For example, the pattern: `(foo)\Kbar` matches `foobar`, but reports that it has matched `bar`. This feature is similar to a look-behind assertion. However, in this case, the part of the subject before the real match does not have to be of fixed length:

```
SELECT REGEXP_SUBSTR('aaa123', '[a-z]*\\K[0-9]*');
-> 123
```

Non-Capturing Groups

The question mark and the colon after the opening parenthesis create a non-capturing group: `(?:...)`.

This example removes an optional article from a word, for example for better sorting of the results.

```
SELECT REGEXP_REPLACE('The King','(?:the|an|a)[^a-z]([a-z]+)','\\1');
-> King
```

Note that the articles are listed inside the left parentheses using the alternation operator `|` but they do not produce a captured subpattern, so the word followed by the article is referenced by `'1'` in the third argument to the function. Using non-capturing groups can be useful to save numbers on the sup-patterns that won't be used in the third argument of REGEXP_REPLACE(), as well as for performance purposes.

Non-Greedy Quantifiers

By default, the repetition quantifiers `?`, `*`, `+` and `{n,m}` are "greedy", that is, they try to match as much as possible. Adding a question mark after a repetition quantifier makes it "non-greedy", so the pattern matches the minimum number of times possible.

This example cuts C comments from a line:

```
SELECT REGEXP_REPLACE('/* Comment1 */ i+= 1; /* Comment2 */', '/[*].*?[*]/','');
->  i+= 1;
```

The pattern without the non-greedy flag to the quantifier `/[*].*[*]/` would match the entire string between the leftmost `/*` and the rightmost `*/`.

Atomic Groups

A sequence inside `(?>`…`)` makes an atomic group. Backtracking inside an atomic group is prevented once it has matched; however, backtracking past to the previous items works normally.

Consider the pattern `\d+foo` applied to the subject string `123bar`. Once the engine scans `123` and fails on the letter `b`, it would normally backtrack to `2` and try to match again, then fail and backtrack to `1` and try to match and fail again, and finally fail the entire pattern. In case of an atomic group `(?>\d+)foo` with the same subject string `123bar`, the engine gives up immediately after the first failure to match `foo`. An atomic group with a quantifier can match all or nothing.

Atomic groups produce faster false results (i.e. in case when a long subject string does not match the pattern), because the regexp engine saves performance on backtracking. However, don't hurry to put everything into atomic groups. This example demonstrates the difference between atomic and non-atomic match:

```
SELECT 'abcc' RLIKE 'a(?>bc|b)c' AS atomic1;
-> 1

SELECT 'abc' RLIKE 'a(?>bc|b)c' AS atomic2;
-> 0

SELECT 'abcc' RLIKE 'a(bc|b)c' AS non_atomic1;
-> 1

SELECT 'abc' RLIKE 'a(bc|b)c' AS non_atomic2;
-> 1
```

The non-atomic pattern matches both `abbc` and `abc`, while the atomic pattern matches `abbc` only.

The atomic group `(?>bc|b)` in the above example can be "translated" as "if there is `bc`, then don't try to match as `b`". So `b` can match only if `bc` is not found.

Atomic groups are not capturing. To make an atomic group capturing, put it into parentheses:

```
SELECT REGEXP_REPLACE('abcc','a((?>bc|b))c','\\1');
-> bc
```

Possessive quantifiers

An atomic group which ends with a quantifier can be rewritten using a so called "possessive quantifier" syntax by putting an additional `+` sign following the quantifier.

The pattern `(?>\d+)foo` from the previous section's example can be rewritten as `\d++foo`.

Absolute and Relative Numeric Backreferences

Backreferences match the same text as previously matched by a capturing group. Backreferences can be written using:

- a backslash followed by a digit
- the `\g` escape sequence followed by a positive or negative number
- the `\g` escape sequence followed by a positive or negative number enclosed in braces

The following backreferences are identical and refer to the first capturing group:

- `\1`
- `\g1`
- `\g{1}`

This example demonstrates a pattern that matches "sense and sensibility" and "response and responsibility", but not "sense and responsibility":

```
SELECT 'sense and sensibility' RLIKE '(sens|respons)e and \\1ibility';
-> 1
```

This example removes doubled words that can unintentionally creep in when you edit a text in a text editor:

```
SELECT REGEXP_REPLACE('using using the the regexp regexp',
 '\\b(\\w+)\\s+\\1\\b','\\1');
-> USING the regexp
```

Note that all double words were removed, in the beginning, in the middle and in the end of the subject string.

A negative number in a `\g` sequence means a relative reference. Relative references can be helpful in long patterns, and also in patterns that are created by joining fragments together that contain references within

themselves. The sequence `\g{-1}` is a reference to the most recently started capturing subpattern before `\g`.

In this example `\g{-1}` is equivalent to `\2`:

```
SELECT 'abc123def123' RLIKE '(abc(123)def)\\g{-1}';
-> 1

SELECT 'abc123def123' RLIKE '(abc(123)def)\\2';
-> 1
```

Named Subpatterns and Backreferences

Using numeric backreferences for capturing groups can be hard to track in a complicated regular expression. Also, the numbers can change if an expression is modified. To overcome these difficulties, PCRE supports named subpatterns.

A subpattern can be named in one of three ways: `(?<name>…)` or `(?'name'…)` as in Perl, or `(?P<name>…)` as in Python. References to capturing subpatterns from other parts of the pattern, can be made by name as well as by number.

Backreferences to a named subpattern can be written using the .NET syntax `\k{name}`, the Perl syntax `\k<name>` or `\k'name'` or `\g{name}`, or the Python syntax `(?P=name)`.

This example tests if the string is a correct HTML tag:

```
SELECT '<a href="../">Up</a>' RLIKE '<(?<tag>[a-z][a-z0-9]*)[^>]*>[^<]*</(?P=tag)>';
-> 1
```

Positive and Negative Look-Ahead and Look-Behind Assertions

Look-ahead and look-behind assertions serve to specify the context for the searched regular expression pattern. Note that the assertions only check the context, they do not capture anything themselves!

This example finds the letter which is not followed by another letter (negative look-ahead):

```
SELECT REGEXP_SUBSTR('ab1','[a-z](?![a-z])');
-> b
```

This example finds the letter which is followed by a digit (positive look-ahead):

```
SELECT REGEXP_SUBSTR('ab1','[a-z](?=[0-9])');
-> b
```

This example finds the letter which does not follow a digit character (negative look-behind):

```
SELECT REGEXP_SUBSTR('1ab','(?<![0-9])[a-z]');
-> b
```

This example finds the letter which follows another letter character (positive look-behind):

```
SELECT REGEXP_SUBSTR('1ab','(?<=[a-z])[a-z]');
-> b
```

Note that look-behind assertions can only be of fixed length; you cannot have repetition operators or alternations with different lengths:

```
SELECT 'aaa' RLIKE '(?<=(a|bc))a';
ERROR 1139 (42000): Got error 'lookbehind assertion is not fixed length at offset 10' from regexp
```

Subroutine Reference and Recursive Patterns

PCRE supports a special syntax to recourse the entire pattern or its individual subpatterns:

Syntax	Description
(?R)	Recourse the entire pattern
(?n)	call subpattern by absolute number
(?+n)	call subpattern by relative number
(?-n)	call subpattern by relative number
(?&name)	call subpattern by name (Perl)
(?P>name)	call subpattern by name (Python)
\g	call subpattern by name (Oniguruma)
\g'name'	call subpattern by name (Oniguruma)
\g	call subpattern by absolute number (Oniguruma)
\g'n'	call subpattern by absolute number (Oniguruma)
\g<+n>	call subpattern by relative number
\g<-n>	call subpattern by relative number
\g'+n'	call subpattern by relative number
\g'-n'	call subpattern by relative number

This example checks for a correct additive arithmetic expression consisting of numbers, unary plus and minus, binary plus and minus, and parentheses:

```
SELECT '1+2-3+(+(4-1)+(-2)+(+1))' RLIKE  '^(([+-]?(\\d+|[(](?1)[)]))(([+-](?1))*))$';
-> 1
```

The recursion is done using `(?1)` to call for the first parenthesized subpattern, which includes everything except the leading `^` and the trailing `$`.

The regular expression in the above example implements the following BNF grammar:

1. `<expression> ::= <term> [(<sign> <term>)...]`
2. `<term> ::= [ <sign> ] <primary>`
3. `<primary> ::= <number> | <left paren> <expression> <right paren>`
4. `<sign> ::= <plus sign> | <minus sign>`

Defining Subpatterns For Use By Reference

Use the `(?(DEFINE)…)` syntax to define subpatterns that can be referenced from elsewhere.

This example defines a subpattern with the name `letters` that matches one or more letters, which is further reused two times:

```
SELECT 'abc123xyz' RLIKE '^(?(DEFINE)(?<letters>[a-z]+))(?&letters)[0-9]+(?&letters)$';
-> 1
```

The above example can also be rewritten to define the digit part as a subpattern as well:

```
SELECT 'abc123xyz' RLIKE
 '^(?(DEFINE)(?<letters>[a-z]+)(?<digits>[0-9]+))(?&letters)(?&digits)(?&letters)$';
-> 1
```

Conditional Subpatterns

There are two forms of conditional subpatterns:

```
(?(condition)yes-pattern)
(?(condition)yes-pattern|no-pattern)
```

The `yes-pattern` is used if the condition is satisfied, otherwise the `no-pattern` (if any) is used.

Conditions With Subpattern References

If a condition consists of a number, it makes a condition with a subpattern reference. Such a condition is true if a capturing subpattern corresponding to the number has previously matched.

This example finds an optionally parenthesized number in a string:

```
SELECT REGEXP_SUBSTR('a(123)b', '([(])?[0-9]+(?(1)[)])');
-> (123)
```

The `([(])?` part makes a capturing subpattern that matches an optional opening parenthesis; the `[0-9]+` part matches a number, and the `(?(1)[)])` part matches a closing parenthesis, but only if the opening parenthesis has been previously found.

Other Kinds of Conditions

The other possible condition kinds are: recursion references and assertions. See the PCRE documentation for details.

Matching Zero Bytes (0x00)

PCRE correctly works with zero bytes in the subject strings:

```
SELECT 'a\0b' RLIKE '^a.b$';
-> 1
```

Zero bytes, however, are not supported literally in the pattern strings and should be escaped using the `\xhh` or `\x{hh}` syntax:

```
SELECT 'a\0b' RLIKE '^a\\x{00}b$';
-> 1
```

Other PCRE Features

PCRE provides other extended features that were not covered in this document, such as duplicate subpattern numbers, backtracking control, breaking utf-8 sequences into individual bytes, setting the match limit, setting the recursion limit, optimization control, recursion conditions, assertion conditions and more types of extended grapheme clusters. Please refer to the PCRE documentation for details.

Enhanced regex was implemented as a GSoC 2013 project by Sudheera Palihakkara.

default_regex_flags Examples

The default_regex_flags variable was introduced to address the remaining incompatibilities between PCRE and the old regex library. Here are some examples of its usage:

The default behaviour (multiline match is off)

```
SELECT 'a\nb\nc' RLIKE '^b$';
+--------------------------+
| '(?m)a\nb\nc' RLIKE '^b$' |
+--------------------------+
|                        0 |
+--------------------------+
```

Enabling the multiline option using the PCRE option syntax:

```
SELECT 'a\nb\nc' RLIKE '(?m)^b$';
+-------------------------+
| 'a\nb\nc' RLIKE '(?m)^b$' |
+-------------------------+
|                       1 |
+-------------------------+
```

Enabling the miltiline option using default_regex_flags

```
SET default_regex_flags='MULTILINE';
SELECT 'a\nb\nc' RLIKE '^b$';
+----------------------+
| 'a\nb\nc' RLIKE '^b$' |
+----------------------+
|                    1 |
+----------------------+
```

See Also

- MariaDB upgrades to PCRE-8.34

This page is licensed: CC BY-SA / Gnu FDL

REGEXP

Syntax

```
expr REGEXP pat, expr RLIKE pat
```

Description

Performs a pattern match of a string expression `expr` against a pattern `pat`. The pattern can be an extended regular expression. See Regular Expressions Overview for details on the syntax for regular expressions (see also PCRE Regular Expressions).

Returns `1` if `expr` matches `pat` or `0` if it doesn't match. If either `expr` or `pat` are `NULL`, the result is `NULL`.

The negative form NOT REGEXP also exists, as an alias for `NOT (string REGEXP pattern)`. `RLIKE` and `NOT RLIKE` are synonyms for `REGEXP` and `NOT REGEXP`, originally provided for mSQL compatibility.

The pattern need not be a literal string. For example, it can be specified as a string expression or table column.

Note: Because MariaDB uses the C escape syntax in strings (for example, "\n" to represent the newline character), you must double any "" that you use in your `REGEXP` strings.

`REGEXP` is not case sensitive, except when used with binary strings.

The default_regex_flags variable addresses the remaining compatibilities between PCRE and the old regex library.

Examples

```
SELECT 'Monty!' REGEXP 'm%y%%';
+-------------------------+
| 'Monty!' REGEXP 'm%y%%' |
+-------------------------+
|                       0 |
+-------------------------+

SELECT 'Monty!' REGEXP '.*';
+----------------------+
| 'Monty!' REGEXP '.*' |
+----------------------+
|                    1 |
+----------------------+

SELECT 'new*\n*line' REGEXP 'new\\*.\\*line';
+---------------------------------------+
| 'new*\n*line' REGEXP 'new\\*.\\*line' |
+---------------------------------------+
|                                     1 |
+---------------------------------------+

SELECT 'a' REGEXP 'A', 'a' REGEXP BINARY 'A';
+----------------+-----------------------+
| 'a' REGEXP 'A' | 'a' REGEXP BINARY 'A' |
+----------------+-----------------------+
|              1 |                     0 |
+----------------+-----------------------+

SELECT 'a' REGEXP '^[a-d]';
+--------------------+
| 'a' REGEXP '^[a-d]' |
+--------------------+
|                  1 |
+--------------------+
```

default_regex_flags examples

MariaDB uses the default_regex_flags variable to address the remaining compatibilities between PCRE and the old regex library.

The default behavior (multiline match is off)

```
SELECT 'a\nb\nc' RLIKE '^b$';
+---------------------------+
| '(?m)a\nb\nc' RLIKE '^b$' |
+---------------------------+
|                         0 |
+---------------------------+
```

Enabling the multiline option using the PCRE option syntax:

```
SELECT 'a\nb\nc' RLIKE '(?m)^b$';
+---------------------------+
| 'a\nb\nc' RLIKE '(?m)^b$' |
+---------------------------+
|                         1 |
+---------------------------+
```

Enabling the multiline option using default_regex_flags

```
SET default_regex_flags='MULTILINE';
SELECT 'a\nb\nc' RLIKE '^b$';
+-----------------------+
| 'a\nb\nc' RLIKE '^b$' |
+-----------------------+
|                     1 |
+-----------------------+
```

See Also

- Operator Precedence

This page is licensed: GPLv2, originally from fill_help_tables.sql

REGEXP_INSTR

Syntax

```
REGEXP_INSTR(subject, pattern)
```

Returns the position of the first occurrence of the regular expression `pattern` in the string `subject`, or 0 if pattern was not found.

The positions start with 1 and are measured in characters (i.e. not in bytes), which is important for multi-byte character sets. You can cast a multi-byte character set to BINARY to get offsets in bytes.

The function follows the case sensitivity rules of the effective collation. Matching is performed case insensitively for case insensitive collations, and case sensitively for case sensitive collations and for binary data.

The collation case sensitivity can be overwritten using the (?i) and (?-i) PCRE flags.

MariaDB uses the PCRE regular expression library for enhanced regular expression performance, and `REGEXP_INSTR` was introduced as part of this enhancement.

Examples

```
SELECT REGEXP_INSTR('abc','b');
-> 2

SELECT REGEXP_INSTR('abc','x');
-> 0

SELECT REGEXP_INSTR('BJÖRN','N');
-> 5
```

Casting a multi-byte character set as BINARY to get offsets in bytes:

```
SELECT REGEXP_INSTR(BINARY 'BJÖRN','N') AS cast_utf8_to_binary;
-> 6
```

Case sensitivity:

```
SELECT REGEXP_INSTR('ABC','b');
-> 2

SELECT REGEXP_INSTR('ABC' COLLATE utf8_bin,'b');
-> 0

SELECT REGEXP_INSTR(BINARY'ABC','b');
-> 0

SELECT REGEXP_INSTR('ABC','(?-i)b');
-> 0

SELECT REGEXP_INSTR('ABC' COLLATE utf8_bin,'(?i)b');
-> 2
```

This page is licensed: CC BY-SA / Gnu FDL

REGEXP_REPLACE

Syntax

```
REGEXP_REPLACE(subject, pattern, replace)
```

Description

`REGEXP_REPLACE` returns the string `subject` with all occurrences of the regular expression `pattern` replaced by the string `replace`. If no occurrences are found, then `subject` is returned as is.

The replace string can have backreferences to the subexpressions in the form \N, where N is a number from 1 to 9.

The function follows the case sensitivity rules of the effective collation. Matching is performed case insensitively for case insensitive collations, and case sensitively for case sensitive collations and for binary data.

The collation case sensitivity can be overwritten using the (?i) and (?-i) PCRE flags.

MariaDB uses the PCRE regular expression library for enhanced regular expression performance, and `REGEXP_REPLACE` was introduced as part of this enhancement.

The default_regex_flags variable addresses the remaining compatibilities between PCRE and the old regex library.

Examples

```
SELECT REGEXP_REPLACE('ab12cd','[0-9]','') AS remove_digits;
-> abcd

SELECT REGEXP_REPLACE('<html><head><title>title</title><body>body</body></htm>', '<.+?>',' ')
AS strip_html;
-> title  body
```

Backreferences to the subexpressions in the form , where N is a number from 1 to 9:

```
SELECT REGEXP_REPLACE('James Bond','^(.*) (.*)$','\\2, \\1') AS reorder_name;
-> Bond, James
```

Case insensitive and case sensitive matches:

```
SELECT REGEXP_REPLACE('ABC','b','-') AS case_insensitive;
-> A-C

SELECT REGEXP_REPLACE('ABC' COLLATE utf8_bin,'b','-') AS case_sensitive;
-> ABC

SELECT REGEXP_REPLACE(BINARY 'ABC','b','-') AS binary_data;
-> ABC
```

Overwriting the collation case sensitivity using the (?i) and (?-i) PCRE flags.

```
SELECT REGEXP_REPLACE('ABC','(?-i)b','-') AS force_case_sensitive;
-> ABC

SELECT REGEXP_REPLACE(BINARY 'ABC','(?i)b','-') AS force_case_insensitive;
-> A-C
```

This page is licensed: CC BY-SA / Gnu FDL

REGEXP_SUBSTR

Syntax

```
REGEXP_SUBSTR(subject,pattern)
```

Description

Returns the part of the string `subject` that matches the regular expression `pattern`, or an empty string if `pattern` was not found.

The function follows the case sensitivity rules of the effective collation. Matching is performed case insensitively for case insensitive collations, and case sensitively for case sensitive collations and for binary data.

The collation case sensitivity can be overwritten using the (?i) and (?-i) PCRE flags.

MariaDB uses the PCRE regular expression library for enhanced regular expression performance, and `REGEXP_SUBSTR` was introduced as part of this enhancement.

The default_regex_flags variable addresses the remaining compatibilities between PCRE and the old regex library.

Examples

```
SELECT REGEXP_SUBSTR('ab12cd','[0-9]+');
-> 12

SELECT REGEXP_SUBSTR(
  'See https://mariadb.org/en/foundation/ for details',
  'https?://[^/]*');
-> https://mariadb.org
```

This page is licensed: CC BY-SA / Gnu FDL

RLIKE

Syntax

```
expr REGEXP pat, expr RLIKE pat
```

Description

`RLIKE` is a synonym for REGEXP.

This page is licensed: CC BY-SA / Gnu FDL

VEC_DISTANCE

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

Syntax

```
VEC_DISTANCE(v, s)
```

Description

`VEC_DISTANCE` is a generic function that behaves either as VEC_DISTANCE_EUCLIDEAN, calculating the Euclidean (L2) distance between two points. Or VEC_DISTANCE_COSINE, calculating the Cosine distance between two vectors, depending on the underlying index type.

If the underlying index cannot be determined, an error 4206 is returned:

```
ERROR 4206 (HY000): Cannot determine distance type for VEC_DISTANCE, index is not found
```

Example

```
INSERT INTO v VALUES
     (1, x'e360d63ebe554f3fcdbc523f4522193f5236083d'),
     (2, x'f511303f72224a3fdd05fe3eb22a133ffae86a3f'),
     (3,x'f09baa3ea172763f123def3e0c7fe53e288bf33e'),
     (4,x'b97a523f2a193e3eb4f62e3f2d23583e9dd60d3f'),
     (5,x'f7c5df3e984b2b3e65e59d3d7376db3eac63773e'),
     (6,x'de01453ffa486d3f10aa4d3fdd66813c71cb163f'),
     (7,x'76edfc3e4b57243f10f8423fb158713f020bda3e'),
     (8,x'56926c3fdf098d3e2c8c5e3d1ad4953daa9d0b3e'),
     (9,x'7b713f3e5258323f80d1113d673b2b3f66e3583f'),
     (10,x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e');

SELECT id FROM v
  ORDER BY VEC_DISTANCE(v, x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e') LIMIT 10;
+----+
| id |
+----+
| 10 |
|  7 |
|  3 |
|  9 |
|  2 |
|  1 |
|  5 |
|  4 |
|  6 |
|  8 |
+----+
```

> **Note:**
>
> The vector index is only used for nearest-neighbor search when the query combines `ORDER BY VEC_DISTANCE(...)` with a `LIMIT`. Without `LIMIT`, the distance is computed for every row (a full table scan). See Vector Overview.

See Also

- VEC_DISTANCE_COSINE
- VEC_DISTANCE_EUCLIDEAN
- Vector Overview
- CREATE TABLE with Vectors

This page is licensed: CC BY-SA / Gnu FDL

VEC_DISTANCE_COSINE

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

Syntax

```
VEC_DISTANCE_COSINE(v, s)
```

Description

`VEC_Distance_Cosine` is an SQL function that calculates the Cosine distance between two (not necessarily normalized) vectors.

Vectors must be of the same length. A distance between two vectors of different lengths is not defined, and `VEC_Distance_Cosine` will return `NULL` in such cases.

If the vector index was not built for the cosine function (see CREATE TABLE with Vectors), the index is not used — a full table scan is performed instead. The VEC_DISTANCE function is a generic function that behaves either as VEC_DISTANCE_EUCLIDEAN or `VEC_DISTANCE_COSINE`, depending on the underlying index type.

Example

```
SELECT VEC_DISTANCE_COSINE(vec_fromtext('[1,2,3]'), vec_fromtext('[3,5,7]'));
+-----------------------------------------------------------------------+
| VEC_DISTANCE_COSINE(vec_fromtext('[1,2,3]'), vec_fromtext('[3,5,7]')) |
+-----------------------------------------------------------------------+
|                                                  0.00258509695694209 |
+-----------------------------------------------------------------------+
```

See Also

- VEC_DISTANCE
- VEC_DISTANCE_EUCLIDEAN
- Vector Overview
- CREATE TABLE with Vectors

This page is licensed: CC BY-SA / Gnu FDL

VEC_DISTANCE_EUCLIDEAN

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

Syntax

```
VEC_DISTANCE_EUCLIDEAN(v, s)
```

Description

`VEC_Distance_Euclidean` is an SQL function that calculates a Euclidean (L2) distance between two points.

Vectors must be of the same length, a distance between two vectors of different lengths is not defined and `VEC_Distance_Euclidean` returns `NULL` in such cases.

If the vector index was not built for the euclidean function (see CREATE TABLE with Vectors), the index is not used, and a full table scan performed instead. The VEC_DISTANCE function is a generic function that behaves either as `VEC_DISTANCE_EUCLIDEAN` or VEC_DISTANCE_COSINE, depending on the underlying index type.

Example

```
INSERT INTO v VALUES
     (1, x'e360d63ebe554f3fcdbc523f4522193f5236083d'),
     (2, x'f511303f72224a3fdd05fe3eb22a133ffae86a3f'),
     (3,x'f09baa3ea172763f123def3e0c7fe53e288bf33e'),
     (4,x'b97a523f2a193e3eb4f62e3f2d23583e9dd60d3f'),
     (5,x'f7c5df3e984b2b3e65e59d3d7376db3eac63773e'),
     (6,x'de01453ffa486d3f10aa4d3fdd66813c71cb163f'),
     (7,x'76edfc3e4b57243f10f8423fb158713f020bda3e'),
     (8,x'56926c3fdf098d3e2c8c5e3d1ad4953daa9d0b3e'),
     (9,x'7b713f3e5258323f80d1113d673b2b3f66e3583f'),
     (10,x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e');

SELECT id FROM v
  ORDER BY VEC_Distance_Euclidean(v, x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e') LIMIT 10;
+----+
| id |
+----+
| 10 |
|  7 |
|  3 |
|  9 |
|  2 |
|  1 |
|  5 |
|  4 |
|  6 |
|  8 |
+----+
```

Note:

The vector index is only used for nearest-neighbor search when the query combines `ORDER BY VEC_DISTANCE_EUCLIDEAN(...)` with a `LIMIT`. Without `LIMIT`, the distance is computed for every row (a full table scan). See Vector Overview.

See Also

- VEC_DISTANCE
- VEC_DISTANCE_COSINE
- Vector Overview
- CREATE TABLE with Vectors

This page is licensed: CC BY-SA / Gnu FDL

VEC_FromText

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

Syntax

```
VEC_FromText(s)
```

Description

`VEC_FromText` converts a text representation of the vector (json array of numbers) to a vector (little-endian IEEE float sequence of bytes, 4 bytes per float).

Example

```
SELECT HEX(vec_fromtext('[1,2,3]'));
+------------------------------+
| HEX(vec_fromtext('[1,2,3]')) |
+------------------------------+
| 0000803F0000004000004040     |
+------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

VEC_ToText

> **Note:**
>
> Vectors are available from MariaDB Community Server 11.7 and from MariaDB Enterprise Server 11.4.5-3.

Syntax

```
VEC_ToText(v)
```

Description

`VEC_ToText` converts a binary vector into a json array of numbers (floats). Returns `NULL` and throws a warning Error 4201 if given an invalid vector.

Example

```
SELECT VEC_ToText(x'e360d63ebe554f3fcdbc523f4522193f5236083d');
+-----------------------------------------------------------+
| VEC_ToText(x'e360d63ebe554f3fcdbc523f4522193f5236083d') |
+-----------------------------------------------------------+
| [0.418708,0.809902,0.823193,0.598179,0.033255]          |
+-----------------------------------------------------------+
```

Invalid vector:

```
SELECT VEC_ToText(x'aabbcc');
+-----------------------+
| VEC_ToText(x'aabbcc') |
+-----------------------+
| NULL                  |
+-----------------------+
1 row in set, 1 warning (0.000 sec)

Warning (Code 4201): Invalid binary vector format. Must use IEEE standard float
  representation in little-endian format. Use VEC_FromText() to generate it.
```

See Also

- Error 4201: Invalid binary vector format

This page is licensed: CC BY-SA / Gnu FDL

Data Type Storage Requirements

The following tables indicate the approximate data storage requirements for each data type.

Numeric Data Types

Data Type	Storage Requirement
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT	4 bytes
BIGINT	8 bytes
FLOAT(p)	4 bytes if p <= 24, otherwise 8 bytes
DOUBLE	8 bytes
DECIMAL	See table below
BIT(M)	(M+7)/8 bytes

Note that `MEDIUMINT` columns will require 4 bytes in memory (for example, in InnoDB buffer pool).

Decimal

Decimals are stored using a binary format, with the integer and the fraction stored separately. Each nine-digit multiple requires 4 bytes, followed by a number of bytes for whatever remains, as follows:

Remaining digits	Storage Requirement
0	0 bytes
1	1 byte
2	1 byte
3	2 bytes
4	2 bytes
5	3 bytes
6	3 bytes
7	4 bytes
8	4 bytes

String Data Types

In the descriptions below, `M` is the declared column length (in characters or in bytes), while `len` is the actual length in bytes of the value.

Data Type	Storage Requirement
ENUM	1 byte for up to 255 enum values, 2 bytes for 256 to 65,535 enum values
CHAR(M)	M × w bytes, where w is the number of bytes required for the maximum-length character in the character set
BINARY(M)	M bytes
VARCHAR(M), VARBINARY(M)	len + 1 bytes if column is 0 – 255 bytes, len + 2 bytes if column may require more than 255 bytes
TINYBLOB, TINYTEXT	len + 1 bytes
BLOB, TEXT	len + 2 bytes
MEDIUMBLOB, MEDIUMTEXT	len + 3 bytes
LONGBLOB, LONGTEXT	len + 4 bytes
XMLTYPE	len + 4 bytes (same as LONGBLOB)
SET	Given M members of the set, (M+7)/8 bytes, rounded up to 1, 2, 3, 4, or 8 bytes
INET6	16 bytes
UUID	16 bytes

Note: Introduced in MariaDB 12.3, the `XMLTYPE` data type has a maximum storage capacity of 4GB, similar to `LONGBLOB`.

In some character sets, not all characters use the same number of bytes. utf8 encodes characters with one to three bytes per character, while `utf8mb4` requires one to four bytes per character.

When using field the `COMPRESSED` attribute, 1 byte is reserved for metadata. For example, `VARCHAR(255)` will use +2 bytes instead of +1.

Examples

Assuming a single-byte character-set:

Value	CHAR(2)	Storage Required	VARCHAR(2)	Storage Required
''	' '	2 bytes	''	1 byte
'1'	'1'	2 bytes	'1'	2 bytes
'12'	'12'	2 bytes	'12'	3 bytes

Date and Time Data Types

Data Type	Storage Requirement
DATE	3 bytes
TIME	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
YEAR	1 byte

Microseconds

MariaDB defaults to the MySQL format (by means of the mysql56_temporal_format variable). Microseconds have the following additional storage requirements:

Precision	Storage Requirement
0	0 bytes
1,2	1 byte
3,4	2 bytes
5,6	3 bytes

NULL Values

For the InnoDB COMPACT, DYNAMIC and COMPRESSED row formats, a number of bytes will be allocated in the record header for the nullable fields. If there are between 1 and 8 nullable fields, 1 such byte will be allocated. In the record payload area, no space will be reserved for values that are `NULL`.

For the InnoDB REDUNDANT row format, the overhead is 1 bit in the record header (as a part of the 1-byte or 2-byte "end of field" pointer). In that format, a `NULL` fixed-length field will consume the same amount of space as any `NOT NULL` value in the record payload area. The motivation is that it is possible to update in place between `NOT NULL` and `NULL` values.

In other formats, `NULL` values usually require 1 bit in the data file, 1 byte in the index file.

This page is licensed: CC BY-SA / Gnu FDL

AUTO_INCREMENT

Description

The `AUTO_INCREMENT` attribute can be used to generate a unique identity for new rows. When you insert a new record into the table (or upon adding an AUTO_INCREMENT attribute with the ALTER TABLE statement), and the `AUTO_INCREMENT` field is NULL or DEFAULT (in the case of an INSERT), automatically be incremented. This also applies to `0`, unless the NO_AUTO_VALUE_ON_ZERO SQL_MODE is enabled.

`AUTO_INCREMENT` columns start from 1 by default. The automatically generated value can never be lower than `0`.

Each table can have only one `AUTO_INCREMENT` column. It must defined as a key (not necessarily the `PRIMARY KEY` or `UNIQUE` key). In some storage engines (including the default InnoDB), if the key consists of multiple columns, the `AUTO_INCREMENT` column must be the first column. Storage engines that permit the column to be placed elsewhere are Aria, MyISAM, MERGE, Spider, BLACKHOLE, FederatedX and Federated.

```
CREATE TABLE animals (
     id MEDIUMINT NOT NULL AUTO_INCREMENT,
     name CHAR(30) NOT NULL,
     PRIMARY KEY (id)
 );

INSERT INTO animals (name) VALUES
    ('dog'),('cat'),('penguin'),
    ('fox'),('whale'),('ostrich');
```

```
SELECT * FROM animals;
+----+---------+
| id | name    |
+----+---------+
|  1 | dog     |
|  2 | cat     |
|  3 | penguin |
|  4 | fox     |
|  5 | whale   |
|  6 | ostrich |
+----+---------+
```

`SERIAL` is an alias for `BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE`.

```
CREATE TABLE t (id SERIAL, c CHAR(1)) ENGINE=InnoDB;

SHOW CREATE TABLE t \G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE `t` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `c` char(1) DEFAULT NULL,
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Setting or Changing the Auto_Increment Value

You can use an ALTER TABLE statement to assign a new value to the `auto_increment` table option, or set the insert_id server system variable to change the next `AUTO_INCREMENT` value inserted by the current session.

LAST_INSERT_ID() can be used to see the last `AUTO_INCREMENT` value inserted by the current session.

```
ALTER TABLE animals AUTO_INCREMENT=8;

INSERT INTO animals (name) VALUES ('aardvark');

SELECT * FROM animals;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | dog       |
|  2 | cat       |
|  3 | penguin   |
|  4 | fox       |
|  5 | whale     |
|  6 | ostrich   |
|  8 | aardvark  |
+----+-----------+

SET insert_id=12;

INSERT INTO animals (name) VALUES ('gorilla');

SELECT * FROM animals;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | dog       |
|  2 | cat       |
|  3 | penguin   |
|  4 | fox       |
|  5 | whale     |
|  6 | ostrich   |
|  8 | aardvark  |
| 12 | gorilla   |
+----+-----------+
```

InnoDB

`AUTO_INCREMENT` is persistent in InnoDB.

See also AUTO_INCREMENT Handling in InnoDB.

Setting Explicit Values

It is possible to specify a value for an `AUTO_INCREMENT` column. If the key is primary or unique, the value must not already exist in the key.

If the new value is higher than the current maximum value, the `AUTO_INCREMENT` value is updated, so the next value will be higher. If the new value is lower than the current maximum value, the `AUTO_INCREMENT` value remains unchanged.

The following example demonstrates these behaviors:

```
CREATE TABLE t (id INTEGER UNSIGNED AUTO_INCREMENT PRIMARY KEY) ENGINE = InnoDB;

INSERT INTO t VALUES (NULL);
SELECT id FROM t;
+----+
| id |
+----+
|  1 |
+----+

INSERT INTO t VALUES (10); -- higher value
SELECT id FROM t;
+----+
| id |
+----+
|  1 |
| 10 |
+----+

INSERT INTO t VALUES (2); -- lower value
INSERT INTO t VALUES (NULL); -- auto value
SELECT id FROM t;
+----+
| id |
+----+
|  1 |
|  2 |
| 10 |
| 11 |
+----+
```

Warning:

The ARCHIVE storage engine does not allow to insert a value that is lower than the current maximum.

Missing Values

An `AUTO_INCREMENT` column normally has missing values. This happens because if a row is deleted, or an `AUTO_INCREMENT` value is explicitly updated, old values are never reused. The `REPLACE` statement also deletes a row, and its value is wasted. With InnoDB, values can be reserved by a transaction; but if the transaction fails (for example, because of a `ROLLBACK`) the reserved value will be lost.

Thus `AUTO_INCREMENT` values can be used to sort results in a chronological order, but not to create a numeric sequence.

Replication

To make master-master or Galera safe to use `AUTO_INCREMENT` , you should use the system variables auto_increment_increment and auto_increment_offset to generate unique values for each server.

```
SET @@auto_increment_increment=3;

SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 3     |
| auto_increment_offset    | 1     |
+--------------------------+-------+

CREATE TABLE t (c INT NOT NULL AUTO_INCREMENT PRIMARY KEY);

INSERT INTO t VALUES (NULL), (NULL), (NULL);

SELECT * FROM t;
+---+
| c |
+---+
| 1 |
| 4 |
| 7 |
+---+

CREATE TABLE t2 (c INT NOT NULL AUTO_INCREMENT PRIMARY KEY);

SET @@auto_increment_offset=2;

SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 3     |
| auto_increment_offset    | 2     |
+--------------------------+-------+

INSERT INTO t2 VALUES (NULL), (NULL), (NULL);

SELECT * FROM t2;
+---+
| c |
+---+
| 2 |
| 5 |
| 8 |
+---+
```

If `auto_increment_offset` is larger than `auto_increment_increment`, the value of `auto_increment_offset` is ignored, and the offset reverts to the default of 1 instead:

```
SET @@auto_increment_offset=5;

SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 3     |
| auto_increment_offset    | 5     |
+--------------------------+-------+

CREATE TABLE t3 (c INT NOT NULL AUTO_INCREMENT PRIMARY KEY);

INSERT INTO t3 VALUES (NULL), (NULL), (NULL);

SELECT * FROM t3;
+---+
| c |
+---+
| 1 |
| 4 |
| 5 |
+---+

+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 3     |
| auto_increment_offset    | 3     |
+--------------------------+-------+

INSERT INTO t4 VALUES (NULL), (NULL), (NULL);

SELECT * FROM t4;
+---+
| c |
+---+
| 3 |
| 6 |
| 9 |
+---+
```

Changing `auto_increment_increment` and `auto_incremenet_offset` when adding a new master to a multi-master setup

The purpose of auto_increment_increment and auto_increment_offset is to ensure that in a multi-master or multi-source setup, all masters generate unique values for `auto_increment` fields or for sequences with `INCREMENT=0`.

If `auto_increment_increment` is larger than the current number of masters, then one can configure the new master with the not used `auto_increment_offset`. The easiest way to add a new master is to stop all MariaDB servers, update `auto_increment_increment` and `auto_increment_offset` in the configuration files, and restart.\

This has to be done if `auto_increment_increment` is 1. If one has more than one master (`auto_increment_increment` > 1), there is a way to add more masters with only having to restart one of the masters. The 'trick' is to configure one of the masters to not use all the values in its current sequence.

The following example illustrates how to do it. Assume you have two masters, A and B.

In this case you will have auto_increment_increment=2 for both masters and A would have `auto_increment_offset=1` and B would have `auto_increment_offset=2`. For A, all `auto_increment` and generated sequence values will be odd.

```
1
3
5
7
```

For B, all values will be even:

```
2
4
6
8
```

See the Replication section above.

If we change `auto_increment_increment` from `2` to `4` in A, it will now generate values from this sequence:

```
1
5
9
13
```

As you can see, values `3`, `7`, `11` are not going to be used.We can get C to use values from this sequence by configuring `auto_increment_increment=4` and `auto_increment_offset=3`.

```
3
7
11
```

If we would like to add a 4'th master (D), we can do that by changing 'B' to use `auto_increment_increment=4` and then configure D to have `auto_increment_increment=4` and `auto_increment_offset=4`.

> **Note:**
>
> Note that when changing the `auto_increment_increment` or `auto_increment_offset` on a server, you have to either restart the server or ensure that all current connections are killed. This is needed to force all connections to use the new values.
>
> Also ensure that, if you want to use the above trick, you always double the value of `auto_increment_increment`. This is needed to ensure that the sequence used will not conflict with numbers generated by any other master.

CHECK Constraints, DEFAULT Values and Virtual Columns

`AUTO_INCREMENT` columns are not permitted in CHECK constraints, DEFAULT value expressions and virtual columns.

Generating Auto_Increment Values When Adding the Attribute

```
CREATE OR REPLACE TABLE t1 (a INT);
INSERT t1 VALUES (0),(0),(0);
ALTER TABLE t1 MODIFY a INT NOT NULL AUTO_INCREMENT PRIMARY KEY;
SELECT * FROM t1;
+---+
| a |
+---+
| 1 |
| 2 |
| 3 |
+---+
```

```
CREATE OR REPLACE TABLE t1 (a INT);
INSERT t1 VALUES (5),(0),(8),(0);
ALTER TABLE t1 MODIFY a INT NOT NULL AUTO_INCREMENT PRIMARY KEY;
SELECT * FROM t1;
+---+
| a |
+---+
| 5 |
| 6 |
| 8 |
| 9 |
+---+
```

If the NO_AUTO_VALUE_ON_ZERO SQL_MODE is set, zero values will not be automatically incremented:

```
SET SQL_MODE='no_auto_value_on_zero';
CREATE OR REPLACE TABLE t1 (a INT);
INSERT t1 VALUES (3), (0);
ALTER TABLE t1 MODIFY a INT NOT NULL AUTO_INCREMENT PRIMARY KEY;
SELECT * FROM t1;
+---+
| a |
+---+
| 0 |
| 3 |
+---+
```

See Also

- Getting Started with Indexes
- Sequences - an alternative to auto_increment available from MariaDB 10.3
- AUTO_INCREMENT FAQ
- LAST_INSERT_ID()
- AUTO_INCREMENT handling in InnoDB
- BLACKHOLE and AUTO_INCREMENT
- UUID_SHORT() - Generate unique ids
- Generating Identifiers – from AUTO_INCREMENT to Sequence (percona.com)

This page is licensed: GPLv2, originally from fill_help_tables.sql

AUTO_INCREMENT FAQ

How do I get the last inserted auto_increment value?

Use the LAST_INSERT_ID() function:

```
SELECT LAST_INSERT_ID();
```

What if someone else inserts before I select my id?

LAST_INSERT_ID() is connection specific, so there is no problem from race conditions.

How do I get the next value to be inserted?

You don't. Insert, then find out what you did with LAST_INSERT_ID().

How do I change what number auto_increment starts with?

Run a query, `ALTER TABLE yourTable AUTO_INCREMENT = x;` . The next insert will contain `x` or `MAX(autoField) + 1` , whichever is higher.

As an alternative, run `INSERT INTO yourTable (autoField) VALUES (x);` . The next insert will contain `x+1` or `MAX(autoField) + 1` , whichever is higher.

Issuing TRUNCATE TABLE will delete all the rows in the table, and will reset the auto_increment value to 0 in most cases.

How do I renumber rows once I deleted some in the middle?

Typically, you don't want to. Gaps are hardly ever a problem; if your application can't handle gaps in the sequence, you probably should rethink your application.

Can I do group-wise auto_increment?

Yes, if you use the MyISAM engine.

How do I get the AUTO_INCREMENT value in a BEFORE INSERT trigger?

This isn't possible. It's only available after insert.

How do I assign two fields the same auto_increment value in one query?

You can't, not even with an `AFTER INSERT` trigger. Insert, then go back and update using `LAST_INSERT_ID()` . Those two statements could be wrapped into one stored procedure if you wish.

However, you can mimic this behavior with a `BEFORE INSERT` trigger and a second table to store the sequence position:

```
CREATE TABLE sequence (table_name VARCHAR(255), position INT UNSIGNED);
INSERT INTO sequence VALUES ('testTable', 0);
CREATE TABLE testTable (firstAuto INT UNSIGNED, secondAuto INT UNSIGNED);
DELIMITER //
CREATE TRIGGER testTable_BI BEFORE INSERT ON testTable FOR EACH ROW BEGIN
  UPDATE sequence SET position = LAST_INSERT_ID(position + 1) WHERE table_name = 'testTable';
  SET NEW.firstAuto = LAST_INSERT_ID();
  SET NEW.secondAuto = LAST_INSERT_ID();
END//
DELIMITER ;
INSERT INTO testTable VALUES (NULL, NULL), (NULL, NULL);
SELECT * FROM testTable;

+-----------+------------+
| firstAuto | secondAuto |
+-----------+------------+
|         1 |          1 |
|         2 |          2 |
+-----------+------------+
```

The same sequence table can maintain separate sequences for multiple tables (or separate sequences for different fields in the same table) by adding extra rows.

Does the auto_increment field have to be primary key?

No, it only has to be indexed. It doesn't even have to be unique.

How does InnoDB handle AUTO_INCREMENT?

See AUTO_INCREMENT handling in InnoDB.

General Information To Read

AUTO_INCREMENT

Manual Notes

There can be only one `AUTO_INCREMENT` column per table, it must be indexed, and it cannot have a `DEFAULT` value. An `AUTO_INCREMENT` column works properly only if it contains only positive values. Inserting a negative number is regarded as inserting a very large positive number. This is done to avoid precision problems when numbers wrap over from positive to negative and also to ensure that you do not accidentally get an `AUTO_INCREMENT` column that contains 0.

How to start a table with a set AUTO_INCREMENT value?

```
CREATE TABLE autoinc_test (
  h INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
  m INT UNSIGNED
) AUTO_INCREMENT = 100;

INSERT INTO autoinc_test ( m ) VALUES ( 1 );

SELECT * FROM autoinc_test;
+-----+------+
| h   | m    |
+-----+------+
| 100 |    1 |
+-----+------+
```

See Also

- AUTO_INCREMENT
- AUTO_INCREMENT handling in XtraDB/InnoDB
- LAST_INSERT_ID()
- BLACKHOLE and AUTO_INCREMENT
- Sequences - an alternative to `AUTO_INCREMENT`

The initial version of this article was copied, with permission, from Autoincrement_FAQ on 2012-10-05.

This page is licensed: CC BY-SA / Gnu FDL

NULL Values

`NULL` represents an unknown value. It is *not* an empty string (by default), or a zero value. These are all valid values, and are not `NULL` values.

When a table is created or the format altered, columns can be specified as accepting `NULL` values, or not accepting them, with the `NULL` and `NOT NULL` clauses respectively.

For example, a customer table could contain dates of birth. For some customers, this information is unknown, so the value could be `NULL`.

The same system could allocate a customer ID for each customer record, and in this case a `NULL` value would not be permitted.

```
CREATE TABLE customer (
 id INT NOT NULL,
 date_of_birth DATE NULL
...
)
```

User-defined variables are `NULL` until a value is explicitly assigned.

Stored routines parameters and local variables can always be set to `NULL`. If no `DEFAULT` value is specified for a local variable, its initial value will be `NULL`. If no value is assigned to an `OUT` parameter in a stored procedure, `NULL` is assigned at the end of the procedure.

Syntax

The case of `NULL` is not relevant. (uppercase) is an alias for `NULL`.

The IS operator accepts `UNKNOWN` as an alias for `NULL`, which is meant for boolean contexts.

Comparison Operators

`NULL` values cannot be used with most comparison operators. For example, =, >, >=, <=, <, or != cannot be used, as any comparison with a NULL always returns a NULL value, never true (1) or false (0).

```
SELECT NULL = NULL;
+-------------+
| NULL = NULL |
+-------------+
|        NULL |
+-------------+

SELECT 99 = NULL;
+-----------+
| 99 = NULL |
+-----------+
|      NULL |
+-----------+
```

To overcome this, certain operators are specifically designed for use with `NULL` values. To cater for testing equality between two values that may contain `NULL` values, there's <=>, `NULL`-safe equal.

```
SELECT 99 <=> NULL, NULL <=> NULL;
+-------------+---------------+
| 99 <=> NULL | NULL <=> NULL |
+-------------+---------------+
|           0 |             1 |
+-------------+---------------+
```

Other operators for working with `NULL` values include IS NULL and IS NOT NULL, ISNULL (for testing an expression) and COALESCE (for returning the first non-NULL parameter).

Ordering

When you order by a field that may contain `NULL` values, any `NULL` values are considered to have the lowest value. So ordering in `DESC` order will see the `NULL` values appearing last. To force `NULL` values to be regarded as highest values, one can add another column which has a higher value when the main field is `NULL`. Example:

```
SELECT col1 FROM tab ORDER BY ISNULL(col1), col1;
```

Descending order, with `NULL` values first:

```
SELECT col1 FROM tab ORDER BY IF(col1 IS NULL, 0, 1), col1 DESC;
```

All `NULL` values are also regarded as equivalent for the purposes of the `DISTINCT` and `GROUP BY` clauses.

Functions

In most cases, functions will return `NULL` if any of the parameters are NULL. There are also functions specifically for handling `NULL` values. These include IFNULL(), NULLIF() and COALESCE().

```
SELECT IFNULL(1,0);
+-------------+
| IFNULL(1,0) |
+-------------+
|           1 |
+-------------+

SELECT IFNULL(NULL,10);
+-----------------+
| IFNULL(NULL,10) |
+-----------------+
|              10 |
+-----------------+

SELECT COALESCE(NULL,NULL,1);
+-----------------------+
| COALESCE(NULL,NULL,1) |
+-----------------------+
|                     1 |
+-----------------------+
```

Aggregate functions, such as SUM and AVG ignore `NULL` values.

```
CREATE TABLE t(x INT);

INSERT INTO t VALUES (1),(9),(NULL);

SELECT SUM(x) FROM t;
+--------+
| SUM(x) |
+--------+
|     10 |
+--------+

SELECT AVG(x) FROM t;
+--------+
| AVG(x) |
+--------+
| 5.0000 |
+--------+
```

The one exception is COUNT(*), which counts rows, and doesn't look at whether a value is NULL or not. Compare for example, `COUNT(x)`, which ignores the `NULL`, and `COUNT(*)`, which counts it:

```
SELECT COUNT(x) FROM t;
+----------+
| COUNT(x) |
+----------+
|        2 |
+----------+

SELECT COUNT(*) FROM t;
+----------+
| COUNT(*) |
+----------+
|        3 |
+----------+
```

AUTO_INCREMENT, TIMESTAMP and Virtual Columns

MariaDB handles `NULL` values in a special way if the field is an AUTO_INCREMENT, a TIMESTAMP or a virtual column. Inserting a `NULL` value into a numeric `AUTO_INCREMENT` column will result in the next number in the auto increment sequence being inserted instead. This technique is frequently used with `AUTO_INCREMENT` fields, which are left to take care of themselves.

```
CREATE TABLE t2(id INT PRIMARY KEY AUTO_INCREMENT, letter CHAR(1));

INSERT INTO t2(letter) VALUES ('a'),('b');

SELECT * FROM t2;
+----+--------+
| id | letter |
+----+--------+
|  1 | a      |
|  2 | b      |
+----+--------+
```

Similarly, if a `NULL` value is assigned to a `TIMESTAMP` field, the current date and time is assigned instead.

```
CREATE TABLE t3 (x INT, ts TIMESTAMP);

INSERT INTO t3(x) VALUES (1),(2);
```

After a pause:

```
INSERT INTO t3(x) VALUES (3);

SELECT* FROM t3;
+------+---------------------+
| x    | ts                  |
+------+---------------------+
|    1 | 2013-09-05 10:14:18 |
|    2 | 2013-09-05 10:14:18 |
|    3 | 2013-09-05 10:14:29 |
+------+---------------------+
```

If a `NULL` is assigned to a `VIRTUAL` or `PERSISTENT` column, the default value is assigned instead.

```
CREATE TABLE virt (c INT, v INT AS (c+10) PERSISTENT) ENGINE=InnoDB;

INSERT INTO virt VALUES (1, NULL);

SELECT c, v FROM virt;
+------+------+
| c    | v    |
+------+------+
|    1 |   11 |
+------+------+
```

In all these special cases, `NULL` is equivalent to the `DEFAULT` keyword.

Inserting

If a `NULL` value is single-row inserted into a column declared as `NOT NULL`, an error will be returned. However, if the SQL mode is not strict (strict is the default), if a `NULL` value is multi-row inserted into a column declared as `NOT NULL`, the implicit default for the column type will be inserted (and `NOT` the default value in the table definition). The implicit defaults are an empty string for string types, and the zero value for numeric, date and time types.

By default, both cases will result in an error.

Examples

```
CREATE TABLE nulltest (
  a INT(11),
  x VARCHAR(10) NOT NULL DEFAULT 'a',
  y INT(11) NOT NULL DEFAULT 23
);
```

Single-row insert:

```
INSERT INTO nulltest (a,x,y) VALUES (1,NULL,NULL);
ERROR 1048 (23000): Column 'x' cannot be null
```

Multi-row insert with SQL mode not strict:

```
show variables like 'sql_mode%';
+---------------+--------------------------------------------+
| Variable_name | Value                                      |
+---------------+--------------------------------------------+
| sql_mode      | NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+---------------+--------------------------------------------+

INSERT INTO nulltest (a,x,y) VALUES (1,NULL,NULL),(2,NULL,NULL);
Query OK, 2 rows affected, 4 warnings (0.08 sec)
Records: 2  Duplicates: 0  Warnings: 4
```

The specified defaults have not been used; rather, the implicit column type defaults have been inserted.

```
SELECT * FROM nulltest;
+------+---+---+
| a    | x | y |
+------+---+---+
|    1 |   | 0 |
|    2 |   | 0 |
+------+---+---+
```

Primary Keys and UNIQUE Indexes

`UNIQUE` indexes can contain multiple `NULL` values. Primary keys are never nullable.

Oracle Compatibility

In Oracle mode, `NULL` can be used as a statement:

```
IF a=10 THEN NULL; ELSE NULL; END IF
```

In Oracle mode, CONCAT and the Logical OR operator || ignore null.

When setting sql_mode=EMPTY_STRING_IS_NULL, empty strings and `NULL` values are the same thing.

For example:

```
SET sql_mode=EMPTY_STRING_IS_NULL;
SELECT '' IS NULL; -- returns TRUE
INSERT INTO t1 VALUES (''); -- inserts NULL
```

See Also

- Primary Keys with Nullable Columns
- IS NULL operator
- IS NOT NULL operator

- ISNULL function
- COALESCE function
- IFNULL function
- NULLIF function
- CONNECT data types
- Oracle mode

This page is licensed: CC BY-SA / Gnu FDL

ROW TYPE OF

Overview

This is special declaration only available inside a stored procedure.

EXAMPLES

```
CREATE TABLE rowtypeof_table(
  descr VARCHAR(20),
  val INT
);
```

```
INSERT INTO rowtypeof_table VALUES ('Life', 42);
```

```
DELIMITER $$
CREATE PROCEDURE rowtypeof_proc()
BEGIN
  DECLARE rec1 ROW TYPE OF rowtypeof_table;
  SELECT * INTO rec1 FROM rowtypeof_table;
  SELECT rec1.descr, rec1.val;
END;
$$
DELIMITER ;
```

```
CALL rowtypeof_proc();

+------------+----------+
| rec1.descr | rec1.val |
+------------+----------+
| Life       |       42 |
+------------+----------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

SERIAL

Overview

This is an alias for `BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE`.

EXAMPLES

```
CREATE TABLE serial_example (
  id SERIAL,
  data VARCHAR(32)
);
```

```
SHOW CREATE TABLE serial_example\G
```

```
*************************** 1. row ***************************
       Table: serial_example
Create Table: CREATE TABLE `serial_example` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `data` varchar(32) DEFAULT NULL,
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

EXTERNAL REFERENCES

Additional information is available here.

This page is: Copyright © 2026 MariaDB. All rights reserved.

TYPE OF

Overview

This is special declaration only available inside a stored procedure.

EXAMPLES

```
CREATE TABLE typeof_table(
  descr VARCHAR(20),
  val INT
);
```

```
INSERT INTO typeof_table VALUES ('Life', 42);
```

```
DELIMITER $$
CREATE PROCEDURE typeof_proc()
BEGIN
  DECLARE descr TYPE OF typeof_table.descr;
  DECLARE val TYPE OF typeof_table.val;
  SELECT * INTO descr, val FROM typeof_table;
  SELECT descr, val;
END;
$$
DELIMITER ;
```

```
CALL typeof_proc();

+-------+------+
| descr | val  |
+-------+------+
| Life  |   42 |
+-------+------+
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Numeric Data Types

See Numeric Data Type Overview.

General introduction to numeric data types. This page summarizes the available integer, fixed-point, and floating-point types and their storage characteristics.

See BIGINT.

Large integer type. A BIGINT uses 8 bytes and can store values from -9223372036854775808 to 9223372036854775807 (signed) or 0 to 18446744073709551615 (unsigned).

See BIT.

Bit-field data type. A BIT(M) column stores M bits per value, allowing storage of binary values from 1 to 64 bits in length.

See BOOL.

Synonym for TINYINT(1). This type is commonly used to represent boolean values, where 0 is considered false and non-zero values are true.

See BOOLEAN.

Complete BOOLEAN type reference: TINYINT(1) synonym, TRUE/FALSE aliases, CREATE TABLE/SHOW CREATE TABLE output, and IF()/IS operator evaluation.

See DEC, NUMERIC, FIXED.

Synonyms for DECIMAL. These keywords declare fixed-point numbers, which store exact numeric data with a defined precision and scale.

See DEC.

Synonym for DECIMAL. This keyword creates a fixed-point column with exact precision, suitable for financial calculations.

See DECIMAL.

Complete DECIMAL type reference: DECIMAL(M,D) syntax, precision limits (M≤65, D≤38), SIGNED/UNSIGNED/ZEROFILL options, and rounding behavior.

See DOUBLE PRECISION.

Synonym for DOUBLE. This keyword declares a normal-size (8-byte) floating-point number with double precision.

See DOUBLE.

Double precision floating-point number. A DOUBLE column uses 8 bytes to store large or precise approximate values.

See FIXED.

Synonym for DECIMAL. This keyword is used to define columns that require exact numeric precision, such as currency.

See FLOAT.

Single precision floating-point number. A FLOAT column uses 4 bytes and stores approximate values with less precision than DOUBLE.

See FLOAT4.

Synonym for FLOAT. This keyword declares a single-precision floating-point column.

See FLOAT8.

Synonym for DOUBLE. This keyword declares a double-precision floating-point column.

See Floating-point Accuracy.

Explanation of floating-point precision issues. This page details why FLOAT and DOUBLE types are approximate and how rounding errors occur.

See INT.

Complete INT type reference: INT(M) syntax, signed (-2147483648 to 2147483647) vs unsigned (0 to 4294967295) ranges, and ZEROFILL padding guidelines.

See INT1.

Synonym for TINYINT. This type uses 1 byte of storage.

See INT2.

Synonym for SMALLINT. This type uses 2 bytes of storage.

See INT3.

Synonym for MEDIUMINT. This type uses 3 bytes of storage.

See INT4.

Synonym for INT. This type uses 4 bytes of storage.

See INT8.

Synonym for BIGINT. This type uses 8 bytes of storage.

See INTEGER.

Synonym for INT. This keyword declares a standard 4-byte integer column.

See MEDIUMINT.

Medium-sized integer. A MEDIUMINT column uses 3 bytes and stores values from -8388608 to 8388607 (signed) or 0 to 16777215 (unsigned).

See MIDDLEINT.

Synonym for MEDIUMINT. This keyword refers to the 3-byte integer type.

See NUMBER.

Oracle-compatible synonym for DECIMAL. This type is used for fixed-point arithmetic.

See NUMERIC.

Synonym for DECIMAL. This type stores exact numeric data values.

See REAL.

Synonym for DOUBLE. In standard SQL mode, REAL is a double-precision floating-point number.

See SMALLINT.

Small integer type. A SMALLINT column uses 2 bytes and stores values from -32768 to 32767 (signed) or 0 to 65535 (unsigned).

See TINYINT.

Very small integer type. A TINYINT column uses 1 byte and stores values from -128 to 127 (signed) or 0 to 255 (unsigned).

This page is licensed: CC BY-SA / Gnu FDL

Numeric Data Type Overview

There are a number of numeric data types:

- BOOLEAN - Synonym for TINYINT(1)

- INT1 - Synonym for TINYINT
- INT2 - Synonym for SMALLINT
- INT3 - Synonym for MEDIUMINT
- INT, INTEGER
- INT4 - Synonym for INT
- INT8 - Synonym for BIGINT
- TINYINT
- SMALLINT
- MEDIUMINT
- BIGINT
- DECIMAL, DEC, NUMERIC, FIXED
- FLOAT
- DOUBLE, DOUBLE PRECISION, REAL
- BIT
- VECTOR

See the specific articles for detailed information on each.

SIGNED, UNSIGNED and ZEROFILL

Most numeric types can be defined as `SIGNED`, `UNSIGNED` or `ZEROFILL`, for example:

```
TINYINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

If `SIGNED`, or no attribute, is specified, a portion of the numeric type will be reserved for the sign (plus or minus). For example, a `TINYINT SIGNED` can range from -128 to 127.

If `UNSIGNED` is specified, no portion of the numeric type is reserved for the sign, so for integer types range can be larger. For example, a `TINYINT UNSIGNED` can range from 0 to 255. Floating point and fixed-point types also can be `UNSIGNED`, but this only prevents negative values from being stored and doesn't alter the range.

If `ZEROFILL` is specified, the column will be set to `UNSIGNED` and the spaces used by default to pad the field are replaced with zeros. `ZEROFILL` is ignored in expressions or as part of a UNION. `ZEROFILL` is a non-standard MySQL and MariaDB enhancement.

Note that although the preferred syntax indicates that the attributes are exclusive, more than one attribute can be specified.

Only the following combinations are supported:

- `SIGNED`
- `UNSIGNED`
- `ZEROFILL`
- `UNSIGNED ZEROFILL`
- `ZEROFILL UNSIGNED`

The latter two should be replaced with simply `ZEROFILL`, but are still accepted by the parser.

Examples

```
CREATE TABLE zf (
  i1 TINYINT SIGNED,
  i2 TINYINT UNSIGNED,
  i3 TINYINT ZEROFILL
);

INSERT INTO zf VALUES (2,2,2);

SELECT * FROM zf;
+------+------+------+
| i1   | i2   | i3   |
+------+------+------+
|    2 |    2 |  002 |
+------+------+------+
```

Range

When attempting to add a value that is out of the valid range for the numeric type, MariaDB will react depending on the strict SQL_MODE setting.

If strict_mode has been set (the default), MariaDB will return an error.

If strict_mode has not been set, MariaDB will adjust the number to fit in the field, returning a warning.

Examples

With `strict_mode` set:

```
SHOW VARIABLES LIKE 'sql_mode';
+---------------+-----------------------------------------------------------------------------------------------
---+
| Variable_name | Value
|
+---------------+-----------------------------------------------------------------------------------------------
---+
| sql_mode      |
STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+---------------+-----------------------------------------------------------------------------------------------
---+

CREATE TABLE ranges (i1 TINYINT, i2 SMALLINT, i3 TINYINT UNSIGNED);

INSERT INTO ranges VALUES (257,257,257);
ERROR 1264 (22003): Out of range value for column 'i1' at row 1

SELECT * FROM ranges;
Empty set (0.10 sec)
```

With `strict_mode` unset:

```
SHOW VARIABLES LIKE 'sql_mode%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| sql_mode      |       |
+---------------+-------+

CREATE TABLE ranges (i1 TINYINT, i2 SMALLINT, i3 TINYINT UNSIGNED);

INSERT INTO ranges VALUES (257,257,257);
Query OK, 1 row affected, 2 warnings (0.00 sec)

SHOW WARNINGS;
+---------+------+--------------------------------------------+
| Level   | Code | Message                                    |
+---------+------+--------------------------------------------+
| Warning | 1264 | Out of range value for column 'i1' at row 1 |
| Warning | 1264 | Out of range value for column 'i3' at row 1 |
+---------+------+--------------------------------------------+
2 rows in set (0.00 sec)

SELECT * FROM ranges;
+------+------+------+
| i1   | i2   | i3   |
+------+------+------+
|  127 |  257 |  255 |
+------+------+------+
```

AUTO_INCREMENT

The `AUTO_INCREMENT` attribute can be used to generate a unique identity for new rows. For more details, see auto_increment.

This page is licensed: CC BY-SA / Gnu FDL

BIGINT

Syntax

```
BIGINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

A large integer. The signed range is `-9223372036854775808` to `9223372036854775807`. The unsigned range is `0` to `18446744073709551615`.

If a column has been set to `ZEROFILL`, all values will be prepended by zeros so that the `BIGINT` value contains a number of M digits.

Note: If the `ZEROFILL` attribute has been specified, the column will automatically become `UNSIGNED`.

For more details on the attributes, see Numeric Data Type Overview.

`SERIAL` is an alias for:

```
BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE
```

`INT8` is a synonym for `BIGINT`.

EXAMPLES

With strict_mode set

```
CREATE TABLE bigints (a BIGINT,b BIGINT UNSIGNED,c BIGINT ZEROFILL);

INSERT INTO bigints VALUES (-10,-10,-10);
ERROR 1264 (22003): Out of range value for column 'b' at row 1

INSERT INTO bigints VALUES (-10,10,-10);
ERROR 1264 (22003): Out of range value for column 'c' at row 1

INSERT INTO bigints VALUES (-10,10,10);

INSERT INTO bigints VALUES (9223372036854775808,9223372036854775808,9223372036854775808);
ERROR 1264 (22003): Out of range value for column 'a' at row 1

INSERT INTO bigints VALUES (9223372036854775807,9223372036854775808,9223372036854775808);

SELECT * FROM bigints;
+---------------------+---------------------+----------------------+
| a                   | b                   | c                    |
+---------------------+---------------------+----------------------+
|                 -10 |                  10 | 00000000000000000010 |
| 9223372036854775807 | 9223372036854775808 | 09223372036854775808 |
+---------------------+---------------------+----------------------+
```

SIGNED and UNSIGNED

The `BIGINT` data type may be `SIGNED` (allowing negative values) or `UNSIGNED` (not allowing negative values).

Example of `BIGINT SIGNED` (the default):

```
CREATE TABLE bigint_signed_example (
   description VARCHAR(20),
   example BIGINT SIGNED
);
```

```
INSERT INTO bigint_signed_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', -9223372036854775808),
   ('Maximum', 9223372036854775807);
```

Example of BIGINT UNSIGNED:

```
CREATE TABLE bigint_unsigned_example (
   description VARCHAR(20),
   example BIGINT UNSIGNED
);
```

```
INSERT INTO bigint_unsigned_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 18446744073709551615);
```

Out of Range

A value is considered “out of range” when it is too small or too large to be stored in a data type. When `sql_mode=STRICT_TRANS_TABLES` (the default) is set, an out-of-range value generates an error. If strict mode is not

in effect, the value is rounded to the nearest valid value and a warning is generated (which might be hidden, depending on your warning settings).

An example of non-strict out-of-range behavior:

```
TRUNCATE bigint_signed_example;

-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO bigint_signed_example VALUES
   ('Underflow', -9223372036854775809),
   ('Overflow', 9223372036854775808);

Warning (Code 1264): Out of range value for column 'example' at row 1
Warning (Code 1264): Out of range value for column 'example' at row 2

SELECT * FROM bigint_signed_example;

+-------------+----------------------+
| description | example              |
+-------------+----------------------+
| Underflow   | -9223372036854775808 |
| Overflow    |  9223372036854775807 |
+-------------+----------------------+

TRUNCATE bigint_unsigned_example;

-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO bigint_unsigned_example VALUES
   ('Underflow', -1),
   ('Overflow', 18446744073709551616);

Warning (Code 1264): Out of range value for column 'example' at row 1
Warning (Code 1264): Out of range value for column 'example' at row 2

SELECT * FROM bigint_unsigned_example;

+-------------+----------------------+
| description | example              |
+-------------+----------------------+
| Underflow   |                    0 |
| Overflow    | 18446744073709551615 |
+-------------+----------------------+
```

BIGINT ZEROFILL

A special type of `BIGINT UNSIGNED is BIGINT ZEROFILL`, which pads out the values with leading zeros in `SELECT` results. The number of leading zeros are just enough to pad the field out to the length of the type's maximum unsigned value, but the zeros are not included in an expression result or in a `UNION SELECT` column.

Using `BIGINT ZEROFILL` works the same way as `BIGINT UNSIGNED` for most operations except a simple `SELECT`. For example, with the following test table setup:

```
CREATE TABLE bigint_zerofill_example (
   description VARCHAR(20),
   example BIGINT ZEROFILL
);

INSERT INTO bigint_zerofill_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 18446744073709551615);

-- Turn off strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO bigint_zerofill_example VALUES
   ('Underflow', -1),
   ('Overflow', 18446744073709551616);

Warning (Code 1264): Out of range value for column 'example' at row 1
Warning (Code 1264): Out of range value for column 'example' at row 2
```

The resulting data would look like this:

```
SELECT *, example + 0 FROM bigint_zerofill_example;

+-------------+----------------------+----------------------+
| description | example              | example + 0          |
+-------------+----------------------+----------------------+
| Zero        | 00000000000000000000 |                    0 |
| Forty-Two   | 00000000000000000042 |                   42 |
| Minimum     | 00000000000000000000 |                    0 |
| Maximum     | 18446744073709551615 | 18446744073709551615 |
| Underflow   | 00000000000000000000 |                    0 |
| Overflow    | 18446744073709551615 | 18446744073709551615 |
+-------------+----------------------+----------------------+
```

See Also

- Numeric Data Type Overview
- TINYINT
- SMALLINT
- MEDIUMINT
- INTEGER

This page is licensed: GPLv2, originally from fill_help_tables.sql

BIT

Syntax

```
BIT[(M)]
```

Description

A bit-field type. `M` indicates the number of bits per value, from `1` to `64`. The default is `1` if `M` is omitted.

Bit values can be inserted with `b'value'` notation, where `value` is the bit value in 0's and 1's.

Bit fields are automatically zero-padded from the left to the full length of the bit, so for example in a `BIT(4)` field, '10' is equivalent to '0010'.

Bits are returned as binary, so to display them, either add 0, or use a function such as HEX, OCT or BIN to convert them.

Examples

Example of BIT:

```
CREATE TABLE bit_example (
  description VARCHAR(20),
  b1 BIT,
  b4 BIT(4),
  b16 BIT(16)
);
```

```
INSERT INTO bit_example VALUES
  ('Zero', 0, 0, 0),
  ('One', 1, 1, 1),
  ('Two', 0, 2, 2),
  ('Eight', 0, 8, b'1000'),
  ('All on', 1, 15, b'1111111111111111');
SELECT description, b1+0, LPAD(BIN(b4), 4, 0) AS b4, HEX(b16)
  FROM bit_example;

+-------------+------+------+----------+
| description | b1+0 | b4   | HEX(b16) |
+-------------+------+------+----------+
| Zero        |    0 | 0000 | 0        |
| One         |    1 | 0001 | 1        |
| Two         |    0 | 0010 | 2        |
| Eight       |    0 | 1000 | 8        |
| All on      |    1 | 1111 | FFFF     |
+-------------+------+------+----------+
```

With strict_mode set

```
CREATE TABLE b ( b1 BIT(8) );
```

```
INSERT INTO b VALUES (b'11111111');

INSERT INTO b VALUES (b'01010101');

INSERT INTO b VALUES (b'1111111111111');
ERROR 1406 (22001): Data too long for column 'b1' at row 1

SELECT b1+0, HEX(b1), OCT(b1), BIN(b1) FROM b;
+------+---------+---------+----------+
| b1+0 | HEX(b1) | OCT(b1) | BIN(b1)  |
+------+---------+---------+----------+
|  255 | FF      | 377     | 11111111 |
|   85 | 55      | 125     | 1010101  |
+------+---------+---------+----------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

BOOL

Overview

See TINYINT.

EXAMPLES

```
CREATE TABLE bool_example (
  example BOOL
) DEFAULT CHARSET=latin1;
```

```
SHOW CREATE TABLE bool_example\G
```

```
*************************** 1. row ***************************
       Table: bool_example
Create Table: CREATE TABLE `bool_example` (
  `example` tinyint(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

BOOLEAN

Syntax

```
BOOL, BOOLEAN
```

Description

These types are synonyms for TINYINT(1). A value of zero is considered false. Non-zero values are considered true.

However, the values `TRUE` and `FALSE` are merely aliases for 1 and 0. See Boolean Literals, as well as the IS operator for testing values against a boolean.

Examples

```
CREATE TABLE boolean_example (
  example BOOLEAN
);
```

```
SHOW CREATE TABLE boolean_example\G
```

```
*************************** 1. row ***************************
       Table: boolean_example
Create Table: CREATE TABLE `boolean_example` (
  `example` tinyint(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

```
CREATE TABLE boo (i BOOLEAN);

DESC boo;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| i     | tinyint(1) | YES  |     | NULL    |       |
+-------+------------+------+-----+---------+-------+
```

```
SELECT IF(0, 'true', 'false');
+------------------------+
| IF(0, 'true', 'false') |
+------------------------+
| false                  |
+------------------------+

SELECT IF(1, 'true', 'false');
+------------------------+
| IF(1, 'true', 'false') |
+------------------------+
| true                   |
+------------------------+

SELECT IF(2, 'true', 'false');
+------------------------+
| IF(2, 'true', 'false') |
+------------------------+
| true                   |
+------------------------+
```

`TRUE` and `FALSE` as aliases for 1 and 0:

```
SELECT IF(0 = FALSE, 'true', 'false');

+--------------------------------+
| IF(0 = FALSE, 'true', 'false') |
+--------------------------------+
| true                           |
+--------------------------------+

SELECT IF(1 = TRUE, 'true', 'false');
+-------------------------------+
| IF(1 = TRUE, 'true', 'false') |
+-------------------------------+
| true                          |
+-------------------------------+

SELECT IF(2 = TRUE, 'true', 'false');
+-------------------------------+
| IF(2 = TRUE, 'true', 'false') |
+-------------------------------+
| false                         |
+-------------------------------+

SELECT IF(2 = FALSE, 'true', 'false');
+--------------------------------+
| IF(2 = FALSE, 'true', 'false') |
+--------------------------------+
| false                          |
+--------------------------------+
```

The last two statements display the results shown because 2 is equal to neither 1 nor 0.

See Also

- Boolean Literals
- IS operator

This page is licensed: GPLv2, originally from fill_help_tables.sql

DEC, NUMERIC, FIXED

Syntax

```
DEC[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]

NUMERIC[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]

FIXED[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

These types are synonyms for DECIMAL. The `FIXED` synonym is available for compatibility with other database systems.

This page is licensed: GPLv2, originally from fill_help_tables.sql

DEC

See DECIMAL.

EXAMPLES

```
CREATE TABLE dec_example (
  example DEC
);
```

```
SHOW CREATE TABLE dec_example\G
```

```
*************************** 1. row ***************************
       Table: dec_example
Create Table: CREATE TABLE `dec_example` (
  `example` decimal(10,0) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

DECIMAL

Syntax

```
DECIMAL[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

A packed “exact” fixed-point number. `M` is the total number of digits (the precision), and `D` is the number of digits after the decimal point (the scale).

- The decimal point and (for negative numbers) the “-” sign are not counted in `M`.
- If `D` is `0`, values have no decimal point or fractional part, and on INSERT, the value will be rounded to the nearest `DECIMAL`.
- The maximum number of digits (`M`) for `DECIMAL` is 65.
- The maximum number of supported decimals (`D`) is `30` before MariaDB 10.2.1 and `38` afterwards.
- If `D` is omitted, the default is `0`. If `M` is omitted, the default is `10`.

`UNSIGNED`, if specified, disallows negative values.

`ZEROFILL`, if specified, pads the number with zeros, up to the total number of digits specified by `M`.

All basic calculations (+, -, *, /) with `DECIMAL` columns are done with a precision of 65 digits.

For more details on the attributes, see Numeric Data Type Overview.

`DEC`, `NUMERIC`, and `FIXED` are synonyms, as well as `NUMBER` in Oracle mode.

Examples

```
CREATE TABLE t1 (d DECIMAL UNSIGNED ZEROFILL);

INSERT INTO t1 VALUES (1),(2),(3),(4.0),(5.2),(5.7);
Query OK, 6 rows affected, 2 warnings (0.16 sec)
Records: 6  Duplicates: 0  Warnings: 2

Note (sql 1265): Data truncated for column 'd' at row 5
Note (sql 1265): Data truncated for column 'd' at row 6

SELECT * FROM t1;
+------------+
| d          |
+------------+
| 0000000001 |
| 0000000002 |
| 0000000003 |
| 0000000004 |
| 0000000005 |
| 0000000006 |
+------------+
```

With strict_mode set

```
INSERT INTO t1 VALUES (-7);
ERROR 1264 (22003): Out of range value for column 'd' at row 1
```

SIGNED and UNSIGNED

The `DECIMAL` data type may be `SIGNED` (allowing negative values) or `UNSIGNED` (not allowing negative values).

Example of `DECIMAL SIGNED` (`SIGNED` is the default):

```
CREATE TABLE decimal_signed_example (
  description VARCHAR(20),
  sz10_0 DECIMAL,
  sz6_2 DECIMAL(6,2),
  sz20_19 DECIMAL(20,19) SIGNED
);
```

```
SET @pi = 3.14159265358979323846264338327950;
```

```
INSERT INTO decimal_signed_example VALUES
  ('Pi', @pi, @pi, @pi),
  ('Series', 1234.567890123, 1234.567890123, 1.234567890123),
  ('Negative', -1234.567890123, -1234.567890123, -1.234567890123),
  ('Various', 1234567890, 9999.99, 9.9999999999999999999);
```

```
SELECT * FROM decimal_signed_example;

+-------------+------------+----------+-----------------------+
| description | sz10_0     | sz6_2    | sz20_19               |
+-------------+------------+----------+-----------------------+
| Pi          |          3 |     3.14 |  3.1415926535897932385 |
| Series      |       1235 |  1234.57 |  1.2345678901230000000 |
| Negative    |      -1235 | -1234.57 | -1.2345678901230000000 |
| Various     | 1234567890 |  9999.99 |  9.9999999999999999999 |
+-------------+------------+----------+-----------------------+
```

Example of `DECIMAL UNSIGNED`:

```
CREATE TABLE decimal_unsigned_example (
  description VARCHAR(20),
  sz10_0 DECIMAL UNSIGNED,
  sz6_2 DECIMAL(6,2) UNSIGNED,
  sz20_19 DECIMAL(20,19) UNSIGNED
);
```

```
SET @pi = 3.14159265358979323846264338327950;
```

```
INSERT INTO decimal_unsigned_example VALUES
  ('Pi', @pi, @pi, @pi),
  ('Series', 1234.567890123, 1234.567890123, 1.234567890123),
  ('Various', 1234567890, 9999.99, 9.9999999999999999999);
```

```
SELECT * FROM decimal_unsigned_example;

+-------------+------------+---------+-----------------------+
| description | sz10_0     | sz6_2   | sz20_19               |
+-------------+------------+---------+-----------------------+
| Pi          |          3 |    3.14 | 3.1415926535897932385 |
| Series      |       1235 | 1234.57 | 1.2345678901230000000 |
| Various     | 1234567890 | 9999.99 | 9.9999999999999999999 |
+-------------+------------+---------+-----------------------+
```

Out of Range

A value is considered “out of range” when it is too small or too large to be stored in a data type. The size specified when creating the column is the strict limit for what values can be represented. When `SQL_MODE` is strict (the default), an out-of-range value generates an error, and the operation fails. If strict mode is not in effect, the value is rounded to the nearest valid value, and a warning is generated (which might be hidden, depending on your warning settings).

A value whose significant digits must be rounded to fit only generates a warning note about data truncation, since it is only an out-of-range value if the rounding causes the value to overflow.

```
TRUNCATE decimal_signed_example;

-- Disable strict mode
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO decimal_signed_example VALUES
  ('OK', 9999999999.4, NULL, NULL),
  ('Rounded overflow', 9999999999.5, NULL, NULL),
  ('Overflow', 10000000000, NULL, NULL),
  ('OK', -9999999999.4, NULL, NULL),
  ('Rounded underflow', -9999999999.5, NULL, NULL),
  ('Underflow', -10000000000, NULL, NULL),
  ('OK', NULL, 9999.994, NULL),
  ('Rounded overflow', NULL, 9999.995, NULL),
  ('Overflow', NULL, 10000, NULL),
  ('OK', NULL, -9999.994, NULL),
  ('Rounded underflow', NULL, -9999.995, NULL),
  ('Underflow', NULL, -10000, NULL),
  ('OK', NULL, NULL, 9.99999999999999999994),
  ('Rounded overflow', NULL, NULL, 9.99999999999999999995),
  ('Overflow', NULL, NULL, 10),
  ('OK', NULL, NULL, -9.99999999999999999994),
  ('Rounded underflow', NULL, NULL, -9.99999999999999999995),
  ('Underflow', NULL, NULL, -10);
```

```
Note (sql 1265): Data truncated for column 'sz10_0' at row 1
Warning (sql 1264): Out of range value for column 'sz10_0' at row 2
Warning (sql 1264): Out of range value for column 'sz10_0' at row 3
Note (sql 1265): Data truncated for column 'sz10_0' at row 4
Warning (sql 1264): Out of range value for column 'sz10_0' at row 5
Warning (sql 1264): Out of range value for column 'sz10_0' at row 6
Note (sql 1265): Data truncated for column 'sz6_2' at row 7
Warning (sql 1264): Out of range value for column 'sz6_2' at row 8
Warning (sql 1264): Out of range value for column 'sz6_2' at row 9
Note (sql 1265): Data truncated for column 'sz6_2' at row 10
Warning (sql 1264): Out of range value for column 'sz6_2' at row 11
Warning (sql 1264): Out of range value for column 'sz6_2' at row 12
Note (sql 1265): Data truncated for column 'sz20_19' at row 13
Warning (sql 1264): Out of range value for column 'sz20_19' at row 14
Warning (sql 1264): Out of range value for column 'sz20_19' at row 15
Note (sql 1265): Data truncated for column 'sz20_19' at row 16
Warning (sql 1264): Out of range value for column 'sz20_19' at row 17
Warning (sql 1264): Out of range value for column 'sz20_19' at row 18
```

```
SELECT * FROM decimal_signed_example;
```

```
+-------------------+-------------+----------+------------------------+
| description       | sz10_0      | sz6_2    | sz20_19                |
+-------------------+-------------+----------+------------------------+
| OK                |  9999999999 |     NULL |                   NULL |
| Rounded overflow  |  9999999999 |     NULL |                   NULL |
| Overflow          |  9999999999 |     NULL |                   NULL |
| OK                | -9999999999 |     NULL |                   NULL |
| Rounded underflow | -9999999999 |     NULL |                   NULL |
| Underflow         | -9999999999 |     NULL |                   NULL |
| OK                |        NULL |  9999.99 |                   NULL |
| Rounded overflow  |        NULL |  9999.99 |                   NULL |
| Overflow          |        NULL |  9999.99 |                   NULL |
| OK                |        NULL | -9999.99 |                   NULL |
| Rounded underflow |        NULL | -9999.99 |                   NULL |
| Underflow         |        NULL | -9999.99 |                   NULL |
| OK                |        NULL |     NULL |  9.9999999999999999999 |
| Rounded overflow  |        NULL |     NULL |  9.9999999999999999999 |
| Overflow          |        NULL |     NULL |  9.9999999999999999999 |
| OK                |        NULL |     NULL | -9.9999999999999999999 |
| Rounded underflow |        NULL |     NULL | -9.9999999999999999999 |
| Underflow         |        NULL |     NULL | -9.9999999999999999999 |
+-------------------+-------------+----------+------------------------+
```

DECIMAL ZEROFILL

A special type of `DECIMAL UNSIGNED` is `DECIMAL ZEROFILL`, which pads out the values with leading zeros in `SELECT` results. The number of leading zeros is just enough to pad the field out to the length of the type's field size (not counting the decimal place), but the zeros are not included in an expression result or in a `UNION SELECT` column.

Using `DECIMAL ZEROFILL` works the same way as `DECIMAL UNSIGNED` for most operations except a simple `SELECT`. For example, with the following test table setup:

```
CREATE TABLE decimal_zerofill_example (
  description VARCHAR(20),
  sz10_0 DECIMAL ZEROFILL,
  sz6_2 DECIMAL(6,2) ZEROFILL,
  sz20_19 DECIMAL(20,19) ZEROFILL
);
```

```
SET @pi = 3.14159265358979323846264338327950;
```

```
INSERT INTO decimal_zerofill_example VALUES
  ('Pi', @pi, @pi, @pi),
  ('Series', 1234.567890123, 1234.567890123, 1.234567890123),
  ('Various', 1234567890, 9999.99, 9.9999999999999999999);
```

```
SELECT * FROM decimal_zerofill_example;

+-------------+------------+---------+------------------------+
| description | sz10_0     | sz6_2   | sz20_19                |
+-------------+------------+---------+------------------------+
| Pi          | 0000000003 | 0003.14 | 3.1415926535897932385 |
| Series      | 0000001235 | 1234.57 | 1.2345678901230000000 |
| Various     | 1234567890 | 9999.99 | 9.9999999999999999999 |
+-------------+------------+---------+------------------------+
```

See Also

- Numeric Data Type Overview
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

DOUBLE PRECISION

Syntax

```
DOUBLE PRECISION[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
REAL[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

`REAL` and `DOUBLE PRECISION` are synonyms for DOUBLE.

Exception: If the `REAL_AS_FLOAT` SQL mode is enabled, `REAL` is a synonym for FLOAT rather than DOUBLE.

EXAMPLES

```
CREATE TABLE double_precision_example (
  example DOUBLE PRECISION
);
```

```
SHOW CREATE TABLE double_precision_example\G
<</code>>

<<sql>>

*************************** 1. row ***************************
       Table: double_precision_example
Create Table: CREATE TABLE `double_precision_example` (
  `example` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

DOUBLE

Syntax

```
DOUBLE[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
DOUBLE PRECISION[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
REAL[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

A normal-size (double-precision) floating-point number (see FLOAT for a single-precision floating-point number). Allowable values are:

- `-1.7976931348623157E+308` to `-2.2250738585072014E-308`
- `0`
- `2.2250738585072014E-308` to `1.7976931348623157E+308`

These are the theoretical limits, based on the IEEE standard. The actual range might be slightly smaller depending on your hardware or operating system.

`M` is the total number of digits and `D` is the number of digits following the decimal point. If `M` and `D` are omitted, values are stored to the limits allowed by the hardware. A double-precision floating-point number is accurate to approximately 15 decimal places.

`UNSIGNED`, if specified, disallows negative values.

`ZEROFILL`, if specified, pads the number with zeros, up to the total number of digits specified by `M`.

`REAL` and `DOUBLE PRECISION` are synonyms, unless the `REAL_AS_FLOAT` SQL mode is enabled, in which case `REAL` is a synonym for FLOAT rather than `DOUBLE`.

See Floating Point Accuracy for issues when using floating-point numbers.

For more details on the attributes, see Numeric Data Type Overview.

Examples

```
CREATE TABLE t1 (d DOUBLE(5,0) zerofill);

INSERT INTO t1 VALUES (1),(2),(3),(4);

SELECT * FROM t1;
+-------+
| d     |
+-------+
| 00001 |
| 00002 |
| 00003 |
| 00004 |
+-------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

FIXED

Overview

See DECIMAL.

EXAMPLES

```
CREATE TABLE fixed_example (
  example FIXED
);
```

```
SHOW CREATE TABLE fixed_example\G

*************************** 1. row ***************************
       Table: fixed_example
Create Table: CREATE TABLE `fixed_example` (
  `example` decimal(10,0) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

FLOAT

Syntax

```
FLOAT[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

A small (single-precision) floating-point number (see DOUBLE for a regular-size floating point number). Allowable values are:

- -3.402823466E+38 to -1.175494351E-38

- 0
- 1.175494351E-38 to 3.402823466E+38.

These are the theoretical limits, based on the IEEE standard. The actual range might be slightly smaller depending on your hardware or operating system.

M is the total number of digits and D is the number of digits following the decimal point. If M and D are omitted, values are stored to the limits allowed by the hardware. A single-precision floating-point number is accurate to approximately 7 decimal places.

`UNSIGNED`, if specified, disallows negative values.

Using `FLOAT` might give you some unexpected problems because all calculations in MariaDB are done with double precision. See Floating Point Accuracy.

For more details on the attributes, see Numeric Data Type Overview.

EXAMPLES

SIGNED and UNSIGNED

The `FLOAT` data type may be `SIGNED` (allowing negative values) or `UNSIGNED` (not allowing negative values).

Example of `FLOAT SIGNED` (`SIGNED` is the default):

```
CREATE TABLE float_signed_example (
  description VARCHAR(20),
  example FLOAT,
  sz6_2 FLOAT(6,2),
  sz20_19 FLOAT(20,19) SIGNED
);
```

```
SET @pi = 3.14159265358979323846264338327795;

INSERT INTO float_signed_example VALUES
  ('Pi', @pi, @pi, @pi),
  ('Series', 1234.567890123, 1234.567890123, 1.234567890123),
  ('Negative', -1234.567890123, -1234.567890123, -1.234567890123),
  ('Various', 1234567890, 9999.99, 9.9999999999999999);
```

```
SELECT * FROM float_signed_example;

+-------------+------------+----------+------------------------+
| description | example    | sz6_2    | sz20_19                |
+-------------+------------+----------+------------------------+
| Pi          |    3.14159 |     3.14 |  3.1415927410125732000 |
| Series      |    1234.57 |  1234.57 |  1.2345678806304932000 |
| Negative    |   -1234.57 | -1234.57 | -1.2345678806304932000 |
| Various     | 1234570000 |  9999.99 | 10.0000000000000000000 |
+-------------+------------+----------+------------------------+
```

Example of `FLOAT UNSIGNED`:

```
CREATE TABLE float_unsigned_example (
  description VARCHAR(20),
  example FLOAT UNSIGNED,
  sz6_2 FLOAT(6,2) UNSIGNED,
  sz20_19 FLOAT(20,19) UNSIGNED
);
```

```
SET @pi = 3.14159265358979323846264338327795;
```

```
INSERT INTO float_unsigned_example VALUES
  ('Pi', @pi, @pi, @pi),
  ('Series', 1234.567890123, 1234.567890123, 1.234567890123),
  ('Various', 1234567890, 9999.99, 9.9999999999999999);
```

```
SELECT * FROM float_unsigned_example;

+-------------+------------+---------+------------------------+
| description | example    | sz6_2   | sz20_19                |
+-------------+------------+---------+------------------------+
| Pi          |    3.14159 |    3.14 |  3.1415927410125732000 |
| Series      |    1234.57 | 1234.57 |  1.2345678806304932000 |
| Various     | 1234570000 | 9999.99 | 10.0000000000000000000 |
+-------------+------------+---------+------------------------+
```

Out of Range

A value is considered “out of range” when it is too small or too large to be stored in a data type. The size specified when creating the column is the limit for what values can be represented. The limits can also vary based on your hardware and operating system. When `SQL_MODE` is strict (the default) an out-of-range value generates an error and the operation fails. If strict mode is not in effect, the value is rounded to the nearest valid value and a warning is generated (which might be hidden, depending on your warning settings).

A value whose significant digits must be rounded to fit only generates a warning note about data truncation, since it is only an out-of-range value if the rounding causes the value to overflow. A somewhat strange exception happens when the decimal places are 16 digits or larger: at that point the value can round up to be one digit larger than you would expect to be accepted, but only for the next larger power of 10. For instance, a `FLOAT(17,16)` should max out at 9.9999999999999999, but that value rounds up to being equivalent to 10 (and 11 overflows).

```
TRUNCATE float_signed_example;

-- Disable strict mode
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO float_signed_example VALUES
  ('OK', 3.402823466e38, NULL, NULL),
  ('Overflow', 4.0e38, NULL, NULL),
  ('OK', -3.402823466e38, NULL, NULL),
  ('Underflow', -4.0e38, NULL, NULL),
  ('OK', NULL, 9999.99, NULL),
  ('Rounded overflow', NULL, 9999.995, NULL),
  ('Overflow', NULL, 10000, NULL),
  ('OK', NULL, -9999.99, NULL),
  ('Rounded underflow', NULL, -9999.995, NULL),
  ('Underflow', NULL, -10000, NULL),
  ('OK', NULL, NULL, 9.9999999999999999),
  ('Rounded OK', NULL, NULL, 9.99999999999999995),
  ('Overflow', NULL, NULL, 11),
  ('OK', NULL, NULL, -9.9999999999999999),
  ('Rounded OK', NULL, NULL, -9.99999999999999995),
  ('Underflow', NULL, NULL, -11);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 2
Warning (sql 1264): Out of range value for column 'example' at row 4
Warning (sql 1264): Out of range value for column 'sz6_2' at row 6
Warning (sql 1264): Out of range value for column 'sz6_2' at row 7
Warning (sql 1264): Out of range value for column 'sz6_2' at row 9
Warning (sql 1264): Out of range value for column 'sz6_2' at row 10
Warning (sql 1264): Out of range value for column 'sz20_19' at row 13
Warning (sql 1264): Out of range value for column 'sz20_19' at row 16
```

```
SELECT * FROM float_signed_example;

+-------------------+-------------+----------+-------------------------+
| description       | example     | sz6_2    | sz20_19                 |
+-------------------+-------------+----------+-------------------------+
| OK                |  3.40282e38 |     NULL |                    NULL |
| Overflow          |  3.40282e38 |     NULL |                    NULL |
| OK                | -3.40282e38 |     NULL |                    NULL |
| Underflow         | -3.40282e38 |     NULL |                    NULL |
| OK                |        NULL |  9999.99 |                    NULL |
| Rounded overflow  |        NULL |  9999.99 |                    NULL |
| Overflow          |        NULL |  9999.99 |                    NULL |
| OK                |        NULL | -9999.99 |                    NULL |
| Rounded underflow |        NULL | -9999.99 |                    NULL |
| Underflow         |        NULL | -9999.99 |                    NULL |
| OK                |        NULL |     NULL |  10.0000000000000000000 |
| Rounded OK        |        NULL |     NULL |  10.0000000000000000000 |
| Overflow          |        NULL |     NULL |  10.0000000000000000000 |
| OK                |        NULL |     NULL | -10.0000000000000000000 |
| Rounded OK        |        NULL |     NULL | -10.0000000000000000000 |
| Underflow         |        NULL |     NULL | -10.0000000000000000000 |
+-------------------+-------------+----------+-------------------------+
```

```
TRUNCATE float_unsigned_example;

-- Disable strict mode
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO float_unsigned_example VALUES
  ('OK', 3.402823466e38, NULL, NULL),
  ('Overflow', 4.0e38, NULL, NULL),
  ('Underflow', -1, NULL, NULL);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 2
Warning (sql 1264): Out of range value for column 'example' at row 3
```

```
SELECT * FROM float_unsigned_example;

+-------------+------------+-------+---------+
| description | example    | sz6_2 | sz20_19 |
+-------------+------------+-------+---------+
| OK          | 3.40282e38 |  NULL |    NULL |
| Overflow    | 3.40282e38 |  NULL |    NULL |
| Underflow   |          0 |  NULL |    NULL |
+-------------+------------+-------+---------+
```

FLOAT ZEROFILL

A special type of `FLOAT UNSIGNED` is `FLOAT ZEROFILL`, which pads out the values with leading zeros in `SELECT` results. The number of leading zeros are just enough to pad the field out to the length of the type's field size (counting the decimal point), but the zeros are not included in an expression result or in a `UNION SELECT` column.

Using `FLOAT ZEROFILL` works the same way as `FLOAT UNSIGNED` for most operations except a simple `SELECT`. For example, with the following test table setup:

```
CREATE TABLE float_zerofill_example (
  description VARCHAR(20),
  example FLOAT ZEROFILL,
  sz6_2 FLOAT(6,2) ZEROFILL,
  sz20_19 FLOAT(20,19) ZEROFILL
);
```

```
SET @pi = 3.14159265358979323846264338327795;
```

```
INSERT INTO float_zerofill_example VALUES
  ('Pi', @pi, @pi, @pi),
  ('Series', 1234.567890123, 1234.567890123, 1.234567890123),
  ('Various', 1234567890, 9999.99, 9.9999999999999999999);
```

```
SELECT * FROM float_zerofill_example;

+-------------+--------------+---------+------------------------+
| description | example      | sz6_2   | sz20_19                |
+-------------+--------------+---------+------------------------+
| Pi          | 000003.14159 |  003.14 |  3.1415927410125732000 |
| Series      | 000001234.57 | 1234.57 |  1.2345678806304932000 |
| Various     | 001234570000 | 9999.99 | 10.0000000000000000000 |
+-------------+--------------+---------+------------------------+
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

FLOAT4

Overview

See FLOAT.

EXAMPLES

```
CREATE TABLE float4_example (
  example FLOAT4
);
```

```
SHOW CREATE TABLE float4_example\G
```

```
*************************** 1. row ***************************
       Table: float4_example
Create Table: CREATE TABLE `float4_example` (
  `example` float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

FLOAT8

Overview

See DOUBLE.

EXAMPLES

```
CREATE TABLE float8_example (
  example FLOAT8
);
```

```
SHOW CREATE TABLE float8_example\G

*************************** 1. row ***************************
       Table: float8_example
Create Table: CREATE TABLE `float8_example` (
  `example` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

Floating-point Accuracy

Due to their nature, not all floating-point numbers can be stored with exact precision. Hardware architecture, the CPU or even the compiler version and optimization level may affect the precision.

If you are comparing DOUBLEs or FLOATs with numeric decimals, it is not safe to use the equality operator.

Sometimes, changing a floating-point number from single-precision (FLOAT) to double-precision (DOUBLE) will fix the problem.

Example

In the following query, f1, f2 and f3 have seemingly identical values across each row, but due to floating point accuracy, the results may be unexpected.

```
CREATE TABLE fpn (id INT, f1 FLOAT, f2 DOUBLE, f3 DECIMAL (10,3));
INSERT INTO fpn VALUES (1,2,2,2),(2,0.1,0.1,0.1);

SELECT * FROM fpn WHERE f1*f1 = f2*f2;
+------+------+------+-------+
| id   | f1   | f2   | f3    |
+------+------+------+-------+
|    1 |    2 |    2 | 2.000 |
+------+------+------+-------+
```

The reason why only one instead of two rows was returned becomes clear when we see how the floating point squares were evaluated.

```
SELECT f1*f1, f2*f2, f3*f3 FROM fpn;
+----------------------+----------------------+----------+
| f1*f1                | f2*f2                | f3*f3    |
+----------------------+----------------------+----------+
|                    4 |                    4 | 4.000000 |
| 0.010000000298023226 | 0.010000000000000002 | 0.010000 |
+----------------------+----------------------+----------+
```

This page is licensed: CC BY-SA / Gnu FDL

INT

Syntax

```
INT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
INTEGER[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

A normal-size integer. When marked `UNSIGNED`, it ranges from 0 to 4294967295, otherwise its range is -2147483648 to 2147483647 (`SIGNED` is the default). If a column has been set to `ZEROFILL`, all values will be prepended by zeros so that the `INT` value contains a number of M digits. `INTEGER` is a synonym for `INT`.

> **Note:**
>
> **Note:** If the `ZEROFILL` attribute has been specified, the column will automatically become `UNSIGNED`.

`INT4` is a synonym for `INT`.

For details on the attributes, see Numeric Data Type Overview.

Examples

With strict_mode set

```
CREATE TABLE ints (a INT,b INT UNSIGNED,c INT ZEROFILL);
```

```
INSERT INTO ints VALUES (-10,-10,-10);
ERROR 1264 (22003): Out of range value for column 'b' at row 1

INSERT INTO ints VALUES (-10,10,-10);
ERROR 1264 (22003): Out of range value for column 'c' at row 1

INSERT INTO ints VALUES (-10,10,10);

INSERT INTO ints VALUES (2147483648,2147483648,2147483648);
ERROR 1264 (22003): Out of range value for column 'a' at row 1

INSERT INTO ints VALUES (2147483647,2147483648,2147483648);

SELECT * FROM ints;
+------------+------------+------------+
| a          | b          | c          |
+------------+------------+------------+
|        -10 |         10 | 0000000010 |
| 2147483647 | 2147483648 | 2147483648 |
+------------+------------+------------+
```

SIGNED and UNSIGNED

The `INT` data type may be `SIGNED` (allowing negative values) or `UNSIGNED` (not allowing negative values).

Example of `INT SIGNED` (the default):

```
CREATE TABLE int_signed_example (
   description VARCHAR(20),
   example INT SIGNED
);
```

```
INSERT INTO int_signed_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', -2147483648),
   ('Maximum', 2147483647);
```

Example of INT UNSIGNED:

```
CREATE TABLE int_unsigned_example (
   description VARCHAR(20),
   example INT UNSIGNED
);
```

```
INSERT INTO int_unsigned_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 4294967295);
```

Out-of-Range

A value is considered “out-of-range” when it is too small or too large to be stored in a data type. When sql_mode=`STRICT_TRANS_TABLES` (the default) is set, an out-of-range value generates an error. If strict mode is not in effect, the value is rounded to the nearest valid value and a warning is generated (which might be hidden, depending on your warning settings).

An example of non-strict out-of-range behavior:

```
TRUNCATE int_signed_example;

-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO int_signed_example VALUES
   ('Underflow', -2147483649),
   ('Overflow', 2147483648);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 1
Warning (sql 1264): Out of range value for column 'example' at row 2
```

```
SELECT * FROM int_signed_example;

+-------------+-------------+
| description | example     |
+-------------+-------------+
| Underflow   | -2147483648 |
| Overflow    |  2147483647 |
+-------------+-------------+
```

```
TRUNCATE int_unsigned_example;

-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO int_unsigned_example VALUES
   ('Underflow', -1),
   ('Overflow', 4294967296);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 1
Warning (sql 1264): Out of range value for column 'example' at row 2
```

```
SELECT * FROM int_unsigned_example;

+-------------+------------+
| description | example    |
+-------------+------------+
| Underflow   |          0 |
| Overflow    | 4294967295 |
+-------------+------------+
```

INT ZEROFILL

A special type of `INT UNSIGNED` is `INT ZEROFILL`, which pads out the values with leading zeros in `SELECT` results. The number of leading zeros are just enough to pad the field out to the length of the type's maximum unsigned value, but the zeros are not included in an expression result or in a `UNION SELECT` column.

Using `INT ZEROFILL` works the same way as `INT UNSIGNED` for most operations except a simple `SELECT`. For example, with the following test table setup:

```
CREATE TABLE int_zerofill_example (
   description VARCHAR(20),
   example INT ZEROFILL
);
```

```
INSERT INTO int_zerofill_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 4294967295);

-- Turn off strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO int_zerofill_example VALUES
   ('Underflow', -1),
   ('Overflow', 4294967296);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 1
Warning (sql 1264): Out of range value for column 'example' at row 2
```

The resulting data would look like this:

```
SELECT *, example + 0 FROM int_zerofill_example;

+-------------+------------+-------------+
| description | example    | example + 0 |
+-------------+------------+-------------+
| Zero        | 0000000000 |           0 |
| Forty-Two   | 0000000042 |          42 |
| Minimum     | 0000000000 |           0 |
| Maximum     | 4294967295 |  4294967295 |
| Underflow   | 0000000000 |           0 |
| Overflow    | 4294967295 |  4294967295 |
+-------------+------------+-------------+
```

See Also

- Numeric Data Type Overview
- TINYINT
- SMALLINT
- MEDIUMINT
- BIGINT

This page is licensed: CC BY-SA / Gnu FDL

INT1

`INT1` is a synonym for TINYINT.

```
CREATE TABLE t1 (x INT1);

DESC t1;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| x     | tinyint(4) | YES  |     | NULL    |       |
+-------+------------+------+-----+---------+-------+
```

EXAMPLES

```
CREATE TABLE int1_example (
  example INT1
);
```

```
SHOW CREATE TABLE int1_example\G
```

```
*************************** 1. row ***************************
       Table: int1_example
Create Table: CREATE TABLE `int1_example` (
  `example` tinyint(4) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is licensed: CC BY-SA / Gnu FDL

INT2

`INT2` is a synonym for SMALLINT.

```
CREATE TABLE t1 (x INT2);

DESC t1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| x     | smallint(6) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

EXAMPLES

```
CREATE TABLE int2_example (
  example INT2
);
```

```
SHOW CREATE TABLE int2_example\G

*************************** 1. row ***************************
       Table: int2_example
Create Table: CREATE TABLE `int2_example` (
  `example` smallint(6) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is licensed: CC BY-SA / Gnu FDL

INT3

`INT3` is a synonym for MEDIUMINT.

```
CREATE TABLE t1 (x INT3);

DESC t1;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| x     | mediumint(9) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

EXAMPLES

```
CREATE TABLE int3_example (
  example INT3
);
```

```
SHOW CREATE TABLE int3_example\G

*************************** 1. row ***************************
       Table: int3_example
Create Table: CREATE TABLE `int3_example` (
  `example` mediumint(9) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is licensed: CC BY-SA / Gnu FDL

INT4

`INT4` is a synonym for INT.

```
CREATE TABLE t1 (x INT4);

DESC t1;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| x     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
```

EXAMPLES

```
CREATE TABLE int4_example (
  example INT4
);
```

```
SHOW CREATE TABLE int4_example\G

*************************** 1. row ***************************
       Table: int4_example
Create Table: CREATE TABLE `int4_example` (
  `example` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is licensed: CC BY-SA / Gnu FDL

INT8

`INT8` is a synonym for BIGINT.

```
CREATE TABLE t1 (x INT8);

DESC t1;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| x     | bigint(20) | YES  |     | NULL    |       |
+-------+------------+------+-----+---------+-------+
```

EXAMPLES

```
CREATE TABLE int8_example (
  example INT8
);
```

```
SHOW CREATE TABLE int8_example\G

*************************** 1. row ***************************
       Table: int8_example
Create Table: CREATE TABLE `int8_example` (
  `example` bigint(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is licensed: CC BY-SA / Gnu FDL

INTEGER

Syntax

```
INTEGER[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

This type is a synonym for INT.

EXAMPLES

```
CREATE TABLE integer_example (
  example INTEGER
);
```

```
SHOW CREATE TABLE integer_example\G

*************************** 1. row ***************************
       Table: integer_example
Create Table: CREATE TABLE `integer_example` (
  `example` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

MEDIUMINT

Syntax

```
MEDIUMINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

A medium-sized integer. The signed range is -8388608 to 8388607. The unsigned range is 0 to 16777215.

`ZEROFILL` pads the integer with zeroes and assumes `UNSIGNED` (even if `UNSIGNED` is not specified).

`INT3` is a synonym for `MEDIUMINT`.

For details on the attributes, see Numeric Data Type Overview.

Examples

With strict_mode set

```
CREATE TABLE mediumints (a MEDIUMINT,b MEDIUMINT UNSIGNED,c MEDIUMINT ZEROFILL);

DESCRIBE mediumints;
+-------+--------------------------------+------+-----+---------+-------+
| Field | Type                           | Null | Key | Default | Extra |
+-------+--------------------------------+------+-----+---------+-------+
| a     | mediumint(9)                   | YES  |     | NULL    |       |
| b     | mediumint(8) unsigned          | YES  |     | NULL    |       |
| c     | mediumint(8) unsigned zerofill | YES  |     | NULL    |       |
+-------+--------------------------------+------+-----+---------+-------+
```

```
INSERT INTO mediumints VALUES (-10,-10,-10);
ERROR 1264 (22003): Out of range value for column 'b' at row 1

INSERT INTO mediumints VALUES (-10,10,-10);
ERROR 1264 (22003): Out of range value for column 'c' at row 1

INSERT INTO mediumints VALUES (-10,10,10);

INSERT INTO mediumints VALUES (8388608,8388608,8388608);
ERROR 1264 (22003): Out of range value for column 'a' at row 1

INSERT INTO mediumints VALUES (8388607,8388608,8388608);

SELECT * FROM mediumints;
+---------+---------+----------+
| a       | b       | c        |
+---------+---------+----------+
|     -10 |      10 | 00000010 |
| 8388607 | 8388608 | 08388608 |
+---------+---------+----------+
```

SIGNED and UNSIGNED

The `MEDIUMINT` data type may be `SIGNED` (allowing negative values) or `UNSIGNED` (not allowing negative values).

Example of `MEDIUMINT SIGNED` (the default):

```
CREATE TABLE mediumint_signed_example (
   description VARCHAR(20),
   example MEDIUMINT SIGNED
);
```

```
INSERT INTO mediumint_signed_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', -8388608),
   ('Maximum', 8388607);
```

Example of `MEDIUMINT UNSIGNED`:

```
CREATE TABLE mediumint_unsigned_example (
   description VARCHAR(20),
   example MEDIUMINT UNSIGNED
);
```

```
INSERT INTO mediumint_unsigned_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 16777215);
```

Out of Range

A value is considered “out of range” when it is too small or too large to be stored in a data type. When `sql_mode=STRICT_TRANS_TABLES` (the default) is set, an out-of-range value generates an error. If strict mode is not in effect, the value is rounded to the nearest valid value and a warning is generated (which might be hidden, depending on your warning settings).

An example of non-strict out-of-range behavior:

```
TRUNCATE mediumint_signed_example;
```

```
-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));
```

```
INSERT INTO mediumint_signed_example VALUES
   ('Underflow', -8388609),
   ('Overflow', 8388608);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 1
Warning (sql 1264): Out of range value for column 'example' at row 2
```

```
SELECT * FROM mediumint_signed_example;
```

```
+-------------+----------+
| description | example  |
+-------------+----------+
| Underflow   | -8388608 |
| Overflow    |  8388607 |
+-------------+----------+
```

```
TRUNCATE mediumint_unsigned_example;
```

```
-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));
```

```
INSERT INTO mediumint_unsigned_example VALUES
   ('Underflow', -1),
   ('Overflow', 16777216);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 1
Warning (sql 1264): Out of range value for column 'example' at row 2
```

```
SELECT * FROM mediumint_unsigned_example;
```

```
+-------------+----------+
| description | example  |
+-------------+----------+
| Underflow   |        0 |
| Overflow    | 16777215 |
+-------------+----------+
```

MEDIUMINT ZEROFILL

A special type of `MEDIUMINT UNSIGNED` is `MEDIUMINT ZEROFILL`, which pads out the values with leading zeros in `SELECT` results. The number of leading zeros are just enough to pad the field out to the length of the type's maximum unsigned value, but the zeros are not included in an expression result or in a `UNION SELECT` column.

Using `MEDIUMINT ZEROFILL` works the same way as `MEDIUMINT UNSIGNED` for most operations except a simple `SELECT`. For example, with the following test table setup:

```
CREATE TABLE mediumint_zerofill_example (
   description VARCHAR(20),
   example MEDIUMINT ZEROFILL
);
```

```
INSERT INTO mediumint_zerofill_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 16777215);
```

```
-- Turn off strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));
```

```
INSERT INTO mediumint_zerofill_example VALUES
   ('Underflow', -1),
   ('Overflow', 16777216);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 1
Warning (sql 1264): Out of range value for column 'example' at row 2
```

The resulting data would look like this:

```
SELECT *, example + 0 FROM mediumint_zerofill_example;
```

```
+-------------+----------+-------------+
| description | example  | example + 0 |
+-------------+----------+-------------+
| Zero        | 00000000 |           0 |
| Forty-Two   | 00000042 |          42 |
| Minimum     | 00000000 |           0 |
| Maximum     | 16777215 |    16777215 |
| Underflow   | 00000000 |           0 |
| Overflow    | 16777215 |    16777215 |
+-------------+----------+-------------+
```

See Also

- Numeric Data Type Overview
- TINYINT
- SMALLINT
- INTEGER

- BIGINT

This page is licensed: GPLv2, originally from fill_help_tables.sql

MIDDLEINT

Overview

See MEDIUMINT.

EXAMPLES

```
CREATE TABLE middleint_example (
  example MIDDLEINT
);
```

```
SHOW CREATE TABLE middleint_example\G
```

```
*************************** 1. row ***************************
       Table: middleint_example
Create Table: CREATE TABLE `middleint_example` (
  `example` mediumint(9) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

NUMBER

```
NUMBER[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]
```

In Oracle mode, `NUMBER` is a synonym for DECIMAL.

EXAMPLES

Number_Example

```
SET sql_mode='oracle';
```

```
CREATE TABLE number_example (
  example NUMBER
);
```

```
SHOW CREATE TABLE number_example\G
```

```
*************************** 1. row ***************************
       Table: number_example
Create Table: CREATE TABLE "number_example" (
  "example" double DEFAULT NULL
)
```

This page is licensed: CC BY-SA / Gnu FDL

NUMERIC

Overview

See DECIMAL.

EXAMPLES

```
CREATE TABLE numeric_example (
  example NUMERIC
);
```

```
SHOW CREATE TABLE numeric_example\G
```

```
*************************** 1. row ***************************
       Table: numeric_example
Create Table: CREATE TABLE `numeric_example` (
  `example` decimal(10,0) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

REAL

Overview

See DOUBLE.

EXAMPLES

```
CREATE TABLE real_example (
  example REAL
);
```

```
SHOW CREATE TABLE real_example\G
```

```
*************************** 1. row ***************************
       Table: real_example
Create Table: CREATE TABLE `real_example` (
  `example` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

SMALLINT

Syntax

```
SMALLINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

A small integer. The signed range is -32768 to 32767. The unsigned range is 0 to 65535.

If a column has been set to `ZEROFILL`, all values will be prepended by zeros so that the `SMALLINT` value contains a number of M digits.

Note:

If the `ZEROFILL` attribute has been specified, the column will automatically become `UNSIGNED`.

`INT2` is a synonym for `SMALLINT`.

For more details on the attributes, see Numeric Data Type Overview.

Examples

With strict_mode set

```
CREATE TABLE smallints (a SMALLINT,b SMALLINT UNSIGNED,c SMALLINT ZEROFILL);
```

```
INSERT INTO smallints VALUES (-10,-10,-10);
ERROR 1264 (22003): Out of range value for column 'b' at row 1

INSERT INTO smallints VALUES (-10,10,-10);
ERROR 1264 (22003): Out of range value for column 'c' at row 1

INSERT INTO smallints VALUES (-10,10,10);

INSERT INTO smallints VALUES (32768,32768,32768);
ERROR 1264 (22003): Out of range value for column 'a' at row 1

INSERT INTO smallints VALUES (32767,32768,32768);

SELECT * FROM smallints;
+-------+-------+-------+
| a     | b     | c     |
+-------+-------+-------+
|   -10 |    10 | 00010 |
| 32767 | 32768 | 32768 |
+-------+-------+-------+
```

SIGNED and UNSIGNED

The `SMALLINT` data type may be `SIGNED` (allowing negative values) or `UNSIGNED` (not allowing negative values).

Example of `SMALLINT SIGNED` (the default):

```
CREATE TABLE smallint_signed_example (
   description VARCHAR(20),
   example SMALLINT SIGNED
);
```

```
INSERT INTO smallint_signed_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', -32768),
   ('Maximum', 32767);
```

Example of SMALLINT UNSIGNED:

```
CREATE TABLE smallint_unsigned_example (
   description VARCHAR(20),
   example SMALLINT UNSIGNED
);
```

```
INSERT INTO smallint_unsigned_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 65535);
```

Out of Range

A value is considered “out of range” when it is too small or too large to be stored in a data type. When `sql_mode=STRICT_TRANS_TABLES` (the default) is set, an out-of-range value generates an error. If strict mode is not

in effect, the value is rounded to the nearest valid value and a warning is generated (which might be hidden, depending on your warning settings).

An example of non-strict out-of-range behavior:

```
TRUNCATE smallint_signed_example;

-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO smallint_signed_example VALUES
   ('Underflow', -32769),
   ('Overflow', 32768);
```

```
Warning (Code 1264): Out of range value for column 'example' at row 1
Warning (Code 1264): Out of range value for column 'example' at row 2
```

```
SELECT * FROM smallint_signed_example;

+-------------+---------+
| description | example |
+-------------+---------+
| Underflow   |  -32768 |
| Overflow    |   32767 |
+-------------+---------+
```

```
TRUNCATE smallint_unsigned_example;

-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO smallint_unsigned_example VALUES
   ('Underflow', -1),
   ('Overflow', 65536);
```

```
Warning (Code 1264): Out of range value for column 'example' at row 1
Warning (Code 1264): Out of range value for column 'example' at row 2
```

```
SELECT * FROM smallint_unsigned_example;
```

```
+-------------+---------+
| description | example |
+-------------+---------+
| Underflow   |       0 |
| Overflow    |   65535 |
+-------------+---------+
```

SMALLINT ZEROFILL

A special type of `SMALLINT UNSIGNED` is `SMALLINT ZEROFILL`, which pads out the values with leading zeros in `SELECT` results. The number of leading zeros are just enough to pad the field out to the length of the type's maximum unsigned value, but the zeros are not included in an expression result or in a `UNION SELECT` column.

Using `SMALLINT ZEROFILL` works the same way as `SMALLINT UNSIGNED` for most operations except a simple `SELECT`. For example, with the following test table setup:

```
CREATE TABLE smallint_zerofill_example (
   description VARCHAR(20),
   example SMALLINT ZEROFILL
);
```

```
INSERT INTO smallint_zerofill_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 65535);

-- Turn off strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO smallint_zerofill_example VALUES
   ('Underflow', -1),
   ('Overflow', 65536);
```

```
Warning (Code 1264): Out of range value for column 'example' at row 1
Warning (Code 1264): Out of range value for column 'example' at row 2
```

The resulting data would look like this:

```
SELECT *, example + 0 FROM smallint_zerofill_example;
```

```
+-------------+---------+-------------+
| description | example | example + 0 |
+-------------+---------+-------------+
| Zero        |   00000 |           0 |
| Forty-Two   |   00042 |          42 |
| Minimum     |   00000 |           0 |
| Maximum     |   65535 |       65535 |
| Underflow   |   00000 |           0 |
| Overflow    |   65535 |       65535 |
+-------------+---------+-------------+
```

See Also

- Numeric Data Type Overview
- TINYINT
- MEDIUMINT
- INTEGER
- BIGINT

This page is licensed: GPLv2, originally from fill_help_tables.sql

TINYINT

Syntax

```
TINYINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

Description

A very small integer. The signed range is -128 to 127. The unsigned range is 0 to 255. For details on the attributes, see Numeric Data Type Overview.

INT1, BOOL, and BOOLEAN are synonyms for `TINYINT`.

Examples

```
CREATE TABLE tinyints (a TINYINT,b TINYINT UNSIGNED,c TINYINT ZEROFILL);
```

With strict_mode set

```
INSERT INTO tinyints VALUES (-10,-10,-10);
ERROR 1264 (22003): Out of range value for column 'b' at row 1

INSERT INTO tinyints VALUES (-10,10,-10);
ERROR 1264 (22003): Out of range value for column 'c' at row 1

INSERT INTO tinyints VALUES (-10,10,10);

SELECT * FROM tinyints;
+------+------+------+
| a    | b    | c    |
+------+------+------+
|  -10 |   10 |  010 |
+------+------+------+

INSERT INTO tinyints VALUES (128,128,128);
ERROR 1264 (22003): Out of range value for column 'a' at row 1

INSERT INTO tinyints VALUES (127,128,128);

SELECT * FROM tinyints;
+------+------+------+
| a    | b    | c    |
+------+------+------+
|  -10 |   10 |  010 |
|  127 |  128 |  128 |
+------+------+------+
```

SIGNED and UNSIGNED

The `TINYINT` data type may be `SIGNED` (allowing negative values) or `UNSIGNED` (not allowing negative values).

Example of `TINYINT SIGNED` (the default):

```
CREATE TABLE tinyint_signed_example (
   description VARCHAR(20),
   example TINYINT SIGNED
);
```

```
INSERT INTO tinyint_signed_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', -128),
   ('Maximum', 127);
```

Example of `TINYINT UNSIGNED`:

```
CREATE TABLE tinyint_unsigned_example (
   description VARCHAR(20),
   example TINYINT UNSIGNED
);
```

```
INSERT INTO tinyint_unsigned_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 255);
```

Out of Range

A value is considered “out of range” when it is too small or too large to be stored in a data type. When `sql_mode=STRICT_TRANS_TABLES` (the default) is set, an out-of-range value generates an error. If strict mode is not in effect, the value is rounded to the nearest valid value and a warning is generated (which might be hidden, depending on your warning settings).

An example of non-strict out-of-range behavior:

```
TRUNCATE tinyint_signed_example;

-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO tinyint_signed_example VALUES
   ('Underflow', -129),
   ('Overflow', 128);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 1
Warning (sql 1264): Out of range value for column 'example' at row 2
```

```
SELECT * FROM tinyint_signed_example;
```

```
+-------------+---------+
| description | example |
+-------------+---------+
| Underflow   |    -128 |
| Overflow    |     127 |
+-------------+---------+
```

```
TRUNCATE tinyint_unsigned_example;

-- Disable strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));

INSERT INTO tinyint_unsigned_example VALUES
   ('Underflow', -1),
   ('Overflow', 256);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 1
Warning (sql 1264): Out of range value for column 'example' at row 2
```

```
SELECT * FROM tinyint_unsigned_example;
```

```
+-------------+---------+
| description | example |
+-------------+---------+
| Underflow   |       0 |
| Overflow    |     255 |
+-------------+---------+
```

TINYINT ZEROFILL

A special type of `TINYINT UNSIGNED` is `TINYINT ZEROFILL`, which pads out the values with leading zeros in SELECT results. The number of leading zeros are just enough to pad the field out to the length of the type’s maximum unsigned value, but the zeros are not included in an expression result or in a UNION `SELECT` column.

Using `TINYINT ZEROFILL` works the same way as `TINYINT UNSIGNED` for most operations except a simple `SELECT`. For example, with the following test table setup:

```
CREATE TABLE tinyint_zerofill_example (
   description VARCHAR(20),
   example TINYINT ZEROFILL
);
INSERT INTO tinyint_zerofill_example VALUES
   ('Zero', 0),
   ('Forty-Two', 42),
   ('Minimum', 0),
   ('Maximum', 255);
```

```
-- Turn off strict mode or the inserts will fail
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));
```

```
INSERT INTO tinyint_zerofill_example VALUES
   ('Underflow', -1),
   ('Overflow', 256);
```

```
Warning (sql 1264): Out of range value for column 'example' at row 1
Warning (sql 1264): Out of range value for column 'example' at row 2
```

The resulting data would look like this:

```
SELECT *, example + 0 FROM tinyint_zerofill_example;
```

```
+-------------+---------+-------------+
| description | example | example + 0 |
+-------------+---------+-------------+
| Zero        |     000 |           0 |
| Forty-Two   |     042 |          42 |
| Minimum     |     000 |           0 |
| Maximum     |     255 |         255 |
| Underflow   |     000 |           0 |
| Overflow    |     255 |         255 |
+-------------+---------+-------------+
```

See Also

- Numeric Data Type Overview
- SMALLINT
- MEDIUMINT
- INTEGER
- BIGINT
- BOOLEAN

This page is licensed: GPLv2, originally from fill_help_tables.sql

Date and Time Data Types

See DATE.

Complete DATE type reference: YYYY-MM-DD format, YYMMDD input literals, date range 1000-01-01 to 9999-12-31, and zero-date SQL_MODE handling.

See DATETIME.

Complete DATETIME data type guide for MariaDB. Complete reference for syntax, valid values, storage requirements, and range limits for production use.

See SQL_TSI_YEAR.

Synonym for YEAR. This keyword is an alias used for declaring a column to store year values.

See TIME.

Store time values. This type handles time durations or time of day, ranging from '-838:59:59' to '838:59:59' with optional microsecond precision.

See TIMESTAMP.

Complete TIMESTAMP data type guide for MariaDB. Complete reference for syntax, valid values, storage requirements, and range limits for production use.

See YEAR Data Type.

Store year values. This type stores a year in 2-digit or 4-digit format, supporting values from 1901 to 2155, and 0000.

This page is licensed: CC BY-SA / Gnu FDL

DATE

Syntax

```
DATE
```

Description

A date. The supported range is '`1000-01-01`' to '`9999-12-31`'. MariaDB displays `DATE` values in '`YYYY-MM-DD`' format, but can be assigned dates in looser formats, including strings or numbers, as long as they make sense. These include a short year, `YY-MM-DD`, no delimiters, `YYMMDD`, or any other acceptable delimiter, for example `YYYY/MM/DD`. For details, see date and time literals.

'`0000-00-00`' is a permitted special value (zero-date), unless the NO_ZERO_DATE SQL_MODE is used. Also, individual components of a date can be set to 0 (for example: '`2015-00-12`'), unless the NO_ZERO_IN_DATE SQL_MODE is used. In many cases, the result of en expression involving a zero-date, or a date with zero-parts, is `NULL`. If the ALLOW_INVALID_DATES `SQL_MODE` is enabled, if the day part is in the range between 1 and 31, the date does not produce any error, even for months that have less than 31 days.

Oracle Mode

In Oracle mode, `DATE` with a time portion is a synonym for DATETIME. See also mariadb_schema.

Examples

```
CREATE TABLE t1 (d DATE);

INSERT INTO t1 VALUES ("2010-01-12"), ("2011-2-28"), ('120314'),('13*04*21');

SELECT * FROM t1;
+------------+
| d          |
+------------+
| 2010-01-12 |
| 2011-02-28 |
| 2012-03-14 |
| 2013-04-21 |
+------------+
```

DATE Format

```
CREATE TABLE date_formats_example (
   description VARCHAR(30),
   example DATE
);
```

```
INSERT INTO date_formats_example VALUES
   ('Full year', '2022-12-30'),
   ('Short year', '22-12-30'),
   ('Short year no delimiters', '221230'),
   ('No delimiters', '20221230'),
   ('Pipe delimiters', '22|2|3'),
   ('Forward slash delimiter', '22/12/30'),
   ('Backward slash delimiter', '22\12\30'),
   ('Asterisk delimiter', '22*12*30'),
   ('Comma delimiter', '22,2,3');
```

The resulting data look like this:

```
SELECT * FROM date_formats_example;
```

```
+--------------------------+------------+
| description              | example    |
+--------------------------+------------+
| Full year                | 2022-12-30 |
| Short year               | 2022-12-30 |
| Short year no delimiters | 2022-12-30 |
| No delimiters            | 2022-12-30 |
| Pipe delimiters          | 2022-02-03 |
| Forward slash delimiter  | 2022-12-30 |
| Backward slash delimiter | 2022-12-30 |
| Asterisk delimiter       | 2022-12-30 |
| Comma delimiter          | 2022-02-03 |
+--------------------------+------------+
```

DATE Range

```
CREATE TABLE date_range_example (
   description VARCHAR(30),
   example DATE
);
```

```
INSERT INTO date_range_example VALUES
   ('Minimum date', '0001-01-01'),
   ('Maximum date', '9999-12-31'),
   ('Below minimum range', '0000*1*1'),
   ('Above maximum range', '10000,12,31');
```

If `SQL_MODE` is strict (the default), the example above generates the following error and no values are inserted:

```
ERROR 1292 (22007): Incorrect date value: '10000,12,31' for column `test`.`date_range_example`.`example`
at row 4
```

If `SQL_MODE` is not strict, the example above generates a warning and (possibly modified) values are inserted. The Below minimum range value is accepted because it contains a zero component. The Above maximum range value is truncated since it is an unacceptable date.

```
Warning (Code 1265): Data truncated for column 'example' at row 4
```

The resulting data would look like this:

```
SELECT * FROM date_range_example;
```

```
+--------------------+------------+
| description        | example    |
+--------------------+------------+
| Minimum date       | 0001-01-01 |
| Maximum date       | 9999-12-31 |
| Below minimum range | 0000-01-01 |
| Above maximum range | 0000-00-00 |
+--------------------+------------+
```

Date Expressions

When using a date value in an expression, such as `DATE_ADD()`, the following illustrates that a `NULL` is generated when a date value is not a real date and when a real date overflows:

```
SELECT example, DATE_ADD(example, INTERVAL 1 DAY)
   FROM date_range_example;
```

```
+------------+-----------------------------------+
| example    | DATE_ADD(example, INTERVAL 1 DAY) |
+------------+-----------------------------------+
| 0001-01-01 | 0001-01-02                        |
| 9999-12-31 | NULL                              |
| 0000-01-01 | NULL                              |
| 0000-00-00 | NULL                              |
+------------+-----------------------------------+
```

```
Warning (Code 1441): Datetime function: datetime field overflow
Warning (Code 1441): Datetime function: datetime field overflow
Warning (Code 1292): Incorrect datetime value: '0000-00-00'
```

Invalid Dates

The following example enhances the `SQL_MODE` to ensure that `ALLOW_INVALID_DATES` is set and illustrates the difference between a day that is outside the range of 1 to 31 and one that is just too large for its month:

```
-- Disable STRICT_TRANS_TABLES and enable ALLOW_INVALID_DATES
SET sql_mode=(SELECT REPLACE(@@sql_mode, 'STRICT_TRANS_TABLES', ''));
SET sql_mode=(SELECT CONCAT(@@sql_mode, ',ALLOW_INVALID_DATES'));
<</code>>

<<sql>>
INSERT INTO date_range_example VALUES
  ('day is invalid for all months', '2019-12-32'),
  ('day is just large for February', '2019-02-31');
```

```
Warning (Code 1265): Data truncated for column 'example' at row 1
```

The resulting data would look like this:

```
SELECT * FROM date_range_example;
```

```
+-------------------------------+------------+
| description                   | example    |
+-------------------------------+------------+
| day is invalid for all months  | 0000-00-00 |
| day is just large for February | 2019-02-31 |
+-------------------------------+------------+
```

See Also

- mariadb_schema data type qualifier

This page is licensed: GPLv2, originally from fill_help_tables.sql

DATETIME

Syntax

```
DATETIME [(microsecond precision)]
```

Description

A date and time combination.

MariaDB displays `DATETIME` values in '`YYYY-MM-DD HH:MM:SS.ffffff`' format, but allows assignment of values to `DATETIME` columns using either strings or numbers. For details, see date and time literals.

`DATETIME` columns also accept CURRENT_TIMESTAMP as the default value.

The –mysql56-temporal-format option, on by default, allows MariaDB to store `DATETME` values using the same low-level format MySQL 5.6 uses. For more information, see Internal Format, below.

For storage requirements, see Data Type Storage Requirements.

Supported Values

MariaDB stores values that use the `DATETIME` data type in a format that supports values between `1000-01-01 00:00:00.000000` and `9999-12-31 23:59:59.999999`.

MariaDB can also store microseconds with a precision between 0 and 6. If no microsecond precision is specified, then 0 is used by default.

MariaDB also supports '`0000-00-00`' as a special *zero-date* value, unless NO_ZERO_DATE is specified in the SQL_MODE. Similarly, individual components of a date can be set to `0` (for example: '`2015-00-12`'), unless NO_ZERO_IN_DATE is specified in the SQL_MODE. In many cases, the result of en expression involving a zero-date, or a date with zero-parts, is `NULL`. If the ALLOW_INVALID_DATES SQL_MODE is enabled, if the day part is in the range between 1 and 31, the date does not produce any error, even for months that have less than 31 days.

Oracle Mode

In Oracle mode, `DATE` with a time portion is a synonym for `DATETIME`. See also mariadb_schema.

Internal Format

A new temporal format was introduced from MySQL 5.6 that alters how the `TIME`, `DATETIME` and `TIMESTAMP` columns operate at lower levels. These changes allow these temporal data types to have fractional parts and negative values. You can disable this feature using the mysql56_temporal_format system variable.

Tables that include `TIMESTAMP` values that were created on an older version of MariaDB or that were created while the mysql56_temporal_format system variable was disabled continue to store data using the older data type format.

In order to update table columns from the older format to the newer format, execute an ALTER TABLE… MODIFY COLUMN statement that changes the column to the *same* data type. This change may be needed if you want to export the table's tablespace and import it onto a server that has `mysql56_temporal_format=ON` set (see MDEV-15225).

For instance, if you have a `DATETIME` column in your table:

```
SHOW VARIABLES LIKE 'mysql56_temporal_format';

+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| mysql56_temporal_format | ON    |
+-------------------------+-------+

ALTER TABLE example_table MODIFY ts_col DATETIME;
```

When MariaDB executes the ALTER TABLE statement, it converts the data from the older temporal format to the newer one.

In the event that you have several tables and columns using temporal data types that you want to switch over to the new format, make sure the system variable is enabled, then perform a dump and restore using `mysqldump`. The columns using relevant temporal data types are restored using the new temporal format.

Columns with old temporal formats are marked with a `/* mariadb-5.3 */` comment in the output of SHOW CREATE TABLE, SHOW COLUMNS, DESCRIBE statements, as well as in the `COLUMN_TYPE` column of the INFORMATION_SCHEMA.COLUMNS Table.

```
SHOW CREATE TABLE mariadb5312_datetime\G
*************************** 1. row ***************************
       Table: mariadb5312_datetime
Create Table: CREATE TABLE `mariadb5312_datetime` (
  `dt0` datetime /* mariadb-5.3 */ DEFAULT NULL,
  `dt6` datetime(6) /* mariadb-5.3 */ DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

Examples

```
CREATE TABLE t1 (d DATETIME);

INSERT INTO t1 VALUES ("2011-03-11"), ("2012-04-19 13:08:22"),
 ("2013-07-18 13:44:22.123456");

SELECT * FROM t1;
+---------------------+
| d                   |
+---------------------+
| 2011-03-11 00:00:00 |
| 2012-04-19 13:08:22 |
| 2013-07-18 13:44:22 |
+---------------------+
```

```
CREATE TABLE t2 (d DATETIME(6));

INSERT INTO t2 VALUES ("2011-03-11"), ("2012-04-19 13:08:22"),
 ("2013-07-18 13:44:22.123456");

SELECT * FROM t2;
+----------------------------+
| d                          |
+----------------------------+
| 2011-03-11 00:00:00.000000 |
| 2012-04-19 13:08:22.000000 |
| 2013-07-18 13:44:22.123456 |
+----------------------------++
```

Strings used in datetime context are automatically converted to datetime(6). If you want to have a datetime without seconds, you should use CONVERT(..,datetime).

```
SELECT CONVERT('2007-11-30 10:30:19',datetime);
+----------------------------------------+
| CONVERT('2007-11-30 10:30:19',datetime) |
+----------------------------------------+
| 2007-11-30 10:30:19                    |
+----------------------------------------+

SELECT CONVERT('2007-11-30 10:30:19',datetime(6));
+-----------------------------------------+
| CONVERT('2007-11-30 10:30:19',datetime(6)) |
+-----------------------------------------+
| 2007-11-30 10:30:19.000000                |
+-----------------------------------------+
```

DATETIME Format

```
CREATE TABLE datetime_formats_example (
  description VARCHAR(30),
  example DATETIME(6)
);
```

```
-- The time zone has no effect on the values
SET @@time_zone = '+00:00';

INSERT INTO datetime_formats_example VALUES
  ('Date without time', '2019-12-30'),
  ('Full year', '2019-12-30 00:00:00'),
  ('Short year', '19-12-30 00:00:00.000'),
  ('Pipe delimiters', '19|2|3 19|00|00.123456'),
  ('Forward slash delimiter', '19/12/30 00/00/00.0');

SET @@time_zone = '-07:00';

INSERT INTO datetime_formats_example VALUES
  ('Asterisk delimiter', '19*12*30 8*35*00'),
  ('Comma delimiter', '19,2,3 12,34,56.123');
```

The resulting data would look like this:

```
SELECT * FROM datetime_formats_example;
```

```
+------------------------+----------------------------+
| description            | example                    |
+------------------------+----------------------------+
| Date without time      | 2019-12-30 00:00:00.000000 |
| Full year              | 2019-12-30 00:00:00.000000 |
| Short year             | 2019-12-30 00:00:00.000000 |
| Pipe delimiters        | 2019-02-03 19:00:00.123456 |
| Forward slash delimiter | 2019-12-30 00:00:00.000000 |
| Asterisk delimiter     | 2019-12-30 08:35:00.000000 |
| Comma delimiter        | 2019-02-03 12:34:56.123000 |
+------------------------+----------------------------+
```

The default microsecond precision when unspecified is 0, and you can use that in a cast in order to trim off stored microseconds:

```
SELECT description, CONVERT(example, DATETIME) AS example
  FROM datetime_formats_example;

+-------------------------+---------------------+
| description             | example             |
+-------------------------+---------------------+
| Date without time       | 2019-12-30 00:00:00 |
| Full year               | 2019-12-30 00:00:00 |
| Short year              | 2019-12-30 00:00:00 |
| Pipe delimiters         | 2019-02-03 19:00:00 |
| Forward slash delimiter | 2019-12-30 00:00:00 |
| Asterisk delimiter      | 2019-12-30 08:35:00 |
| Comma delimiter         | 2019-02-03 12:34:56 |
+-------------------------+---------------------+
```

See Also

- Data Type Storage Requirements
- CONVERT()
- Oracle mode
- mariadb_schema data type qualifier

This page is licensed: GPLv2, originally from fill_help_tables.sql

SQL_TSI_YEAR

Overview

See YEAR.

EXAMPLES

```
CREATE TABLE sql_tsi_year_example (
  example SQL_TSI_YEAR
);
```

```
SHOW CREATE TABLE sql_tsi_year_example\G
```

```
*************************** 1. row ***************************
       Table: sql_tsi_year_example
Create Table: CREATE TABLE `sql_tsi_year_example` (
  `example` year(4) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

TIME

Syntax

```
TIME [(<microsecond precision>)]
```

Description

A time. The range is `'-838:59:59.999999'` to `'838:59:59.999999'`. Microsecond precision can be from 0-6; if not specified 0 is used.

MariaDB displays `TIME` values in `'HH:MM:SS.ssssss'` format, but allows assignment of times in looser formats, including 'D HH:MM:SS', 'HH:MM:SS', 'HH:MM', 'D HH:MM', 'D HH', 'SS', or 'HHMMSS', as well as permitting dropping of any leading zeros when a delimiter is provided, for example '3:9:10'. For details, see date and time literals.

The –mysql56-temporal-format option, on by default, allows MariaDB to store `TIME` values using the same low-level format MySQL 5.6 uses.

Internal Format

A new temporal format was introduced from MySQL 5.6 that alters how the `TIME`, `DATETIME` and `TIMESTAMP` columns operate at lower levels. These changes allow these temporal data types to have fractional parts and negative values. You can disable this feature using the mysql56_temporal_format system variable.

Tables that include `TIMESTAMP` values that were created on an older version of MariaDB or that were created while the mysql56_temporal_format system variable was disabled continue to store data using the older data type format.

In order to update table columns from the older format to the newer format, execute an ALTER TABLE… MODIFY COLUMN statement that changes the column to the *same* data type. This change may be needed if you want to export the table's tablespace and import it onto a server that has `mysql56_temporal_format=ON` set (see MDEV-15225).

For instance, if you have a `TIME` column in your table:

```
SHOW VARIABLES LIKE 'mysql56_temporal_format';

+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| mysql56_temporal_format | ON    |
+-------------------------+-------+

ALTER TABLE example_table MODIFY ts_col TIME;
```

When MariaDB executes the ALTER TABLE statement, it converts the data from the older temporal format to the newer one.

In the event that you have several tables and columns using temporal data types that you want to switch over to the new format, make sure the system variable is enabled, then perform a dump and restore using `mariadb-dump`. The columns using relevant temporal data types are restored using the new temporal format.

Columns with old temporal formats are marked with a `/* mariadb-5.3 */` comment in the output of SHOW CREATE TABLE, SHOW COLUMNS, DESCRIBE statements, as well as in the `COLUMN_TYPE` column of the INFORMATION_SCHEMA.COLUMNS Table.

```
SHOW CREATE TABLE mariadb5312_time\G
*************************** 1. row ***************************
       Table: mariadb5312_time
Create Table: CREATE TABLE `mariadb5312_time` (
  `t0` time /* mariadb-5.3 */ DEFAULT NULL,
  `t6` time(6) /* mariadb-5.3 */ DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

> **Warning:**
>
> Columns with the current format are not marked with a comment.

Examples

```
INSERT INTO time VALUES ('90:00:00'), ('800:00:00'), (800), (22), (151413), ('9:6:3'), ('12 09');

SELECT * FROM time;
+-----------+
| t         |
+-----------+
| 90:00:00  |
| 800:00:00 |
| 00:08:00  |
| 00:00:22  |
| 15:14:13  |
| 09:06:03  |
| 297:00:00 |
+-----------+
```

Time example:

```
CREATE TABLE time_example (
  description VARCHAR(30),
  example TIME(6)
);
```

```
INSERT INTO time_example VALUES
  ('HH:MM:SS', '12:34:56'),
  ('HHMMSS', '123456'),
  ('SS.microsec', '42.123456');
```

The resulting data look like this:

```
SELECT * FROM time_example;
```

```
+-------------+-----------------+
| description | example         |
+-------------+-----------------+
| HH:MM:SS    | 12:34:56.000000 |
| HHMMSS      | 12:34:56.000000 |
| SS.microsec | 00:00:42.123456 |
+-------------+-----------------+
```

See Also

- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

TIMESTAMP

This page is about the `TIMESTAMP` **data type**. For the timestamp function, see TIMESTAMP FUNCTION.

Syntax

```
TIMESTAMP [(<microsecond precision)]
```

Description

A timestamp in the format `YYYY-MM-DD HH:MM:SS.ffffff`.

The timestamp field is generally used to define at which moment in time a row was added or updated and by default will automatically be assigned the current datetime when a record is inserted or updated. The automatic properties

only apply to the first `TIMESTAMP` in the record; subsequent `TIMESTAMP` columns will not be changed.

MariaDB includes the –mysql56-temporal-format option, on by default, which allows MariaDB to store `TIMESTAMP` values using the same low-level format MySQL 5.6 uses.

For more information, see Internal Format.

Supported Values

> **CURRENT**
>
> MariaDB stores values that use the `TIMESTAMP` data type as the number of seconds since '1970-01-01 00:00:00' (UTC). This means that the `TIMESTAMP` data type can hold values between '1970-01-01 00:00:01' (UTC) and '2106-02-07 06:28:15 UTC'.
>
> **< 11.5**
>
> MariaDB stores values that use the `TIMESTAMP` data type as the number of seconds since '1970-01-01 00:00:00' (UTC). This means that the `TIMESTAMP` data type can hold values between '1970-01-01 00:00:01' (UTC) and '2038-01-19 03:14:07' (UTC).

MariaDB can also store microseconds with a precision between 0 and 6. If no microsecond precision is specified, then 0 is used by default.

Automatic Values

MariaDB has special behavior for the first column that uses the `TIMESTAMP` data type in a specific table when the system variable explicit_defaults_for_timestamp is not set (which was the default until MariaDB 10.10). For the first column that uses the `TIMESTAMP` data type in a specific table, MariaDB automatically assigns the following properties to the column:

- `DEFAULT CURRENT_TIMESTAMP`
- `ON UPDATE CURRENT_TIMESTAMP`

This means that if the column is not explicitly assigned a value in an `INSERT` or `UPDATE` query, then MariaDB will automatically initialize the column's value with the current date and time.

This automatic initialization for `INSERT` and `UPDATE` queries can also be **explicitly enabled** for a column that uses the `TIMESTAMP` data type by specifying the `DEFAULT CURRENT_TIMESTAMP` and `ON UPDATE CURRENT_TIMESTAMP` clauses for the column. In these clauses, any synonym of CURRENT_TIMESTAMP is accepted, including `CURRENT_TIMESTAMP()`, `NOW()`, `LOCALTIME`, `LOCALTIME()`, `LOCALTIMESTAMP`, and `LOCALTIMESTAMP()`.

This automatic initialization for `INSERT` queries can also be **explicitly disabled** for a column that uses the `TIMESTAMP` data type by specifying a constant `DEFAULT` value. For example, `DEFAULT 0`.

This automatic initialization for `UPDATE` queries can also be **explicitly disabled** for a column that uses the `TIMESTAMP` data type by specifying a `DEFAULT` clause for the column, but no `ON UPDATE` clause. If a `DEFAULT` clause is explicitly specified for a column that uses the `TIMESTAMP` data type, but an `ON UPDATE` clause is not specified for the column, then the timestamp value will not automatically change when an `UPDATE` statement is executed.

MariaDB also has special behavior if `NULL` is assigned to column that uses the `TIMESTAMP` data type. If the column is assigned the `NULL` value in an `INSERT` or `UPDATE` query, then MariaDB will automatically initialize the column's value with the current date and time. For details, see NULL values in MariaDB.

This automatic initialization for `NULL` values can also be **explicitly disabled** for a column that uses the `TIMESTAMP` data type by specifying the `NULL` attribute for the column. In this case, if the column's value is set to `NULL`, then the

column's value will actually be set to `NULL`.

Time Zones

If a column uses the `TIMESTAMP` data type, then any inserted values are converted from the session's time zone to Coordinated Universal Time (UTC) when stored, and converted back to the session's time zone when retrieved.

MariaDB validates `TIMESTAMP` literals against the session's time zone. For example, if a specific time range never occurred in a specific time zone due to daylight savings time, then `TIMESTAMP` values within that range would be invalid for that time zone.

MariaDB does not store any time zone identifier with the value of the `TIMESTAMP` data type. See MDEV-10018 for more information.

MariaDB does not support time zone literals that contain time zone identifiers. See MDEV-11829 for more information.

Limitations

- Because the `TIMESTAMP` value is stored as Epoch Seconds, the timestamp value '1970-01-01 00:00:00' (UTC) is reserved since the second #0 is used to represent '0000-00-00 00:00:00'.

SQL_MODE=MAXDB

If the SQL_MODE is set to `MAXDB`, TIMESTAMP fields will be silently converted to DATETIME.

Internal Format

A temporal format was introduced from MySQL 5.6 that alters how the `TIME`, `DATETIME` and `TIMESTAMP` columns operate at lower levels. These changes allow these temporal data types to have fractional parts and negative values. You can disable this feature using the mysql56_temporal_format system variable.

Tables that include `TIMESTAMP` values that were created on an older version of MariaDB or that were created while the mysql56_temporal_format system variable was disabled continue to store data using the older data type format.

In order to update table columns from the older format to the newer format, execute an ALTER TABLE… MODIFY COLUMN statement that changes the column to the *same* data type. This change may be needed if you want to export the table's tablespace and import it onto a server that has `mysql56_temporal_format=ON` set (see MDEV-15225).

For instance, if you have a `TIMESTAMP` column in your table:

```
SHOW VARIABLES LIKE 'mysql56_temporal_format';

+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| mysql56_temporal_format | ON    |
+-------------------------+-------+

ALTER TABLE example_table MODIFY ts_col TIMESTAMP;
```

When MariaDB executes the ALTER TABLE statement, it converts the data from the older temporal format to the newer one.

In the event that you have several tables and columns using temporal data types that you want to switch over to the new format, make sure the system variable is enabled, then perform a dump and restore using mariadb-dump. The columns using relevant temporal data types are restored using the new temporal format.

> **CURRENT**

Columns with old temporal formats are marked with a `/* mariadb-5.3 */` comment in the output of SHOW CREATE TABLE, SHOW COLUMNS, DESCRIBE statements, as well as in the `COLUMN_TYPE` column of the INFORMATION_SCHEMA.COLUMNS Table.

< 10.5.1

Columns with old temporal formats are **not** marked with a `/* mariadb-5.3 */` comment in the output of SHOW CREATE TABLE, SHOW COLUMNS, DESCRIBE statements, as well as in the `COLUMN_TYPE` column of the INFORMATION_SCHEMA.COLUMNS Table.

```
SHOW CREATE TABLE mariadb5312_timestamp\G
*************************** 1. row ***************************
       Table: mariadb5312_timestamp
Create Table: CREATE TABLE `mariadb5312_timestamp` (
  `ts0` timestamp /* mariadb-5.3 */ NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `ts6` timestamp(6) /* mariadb-5.3 */ NOT NULL DEFAULT '0000-00-00 00:00:00.000000'
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

Examples

```
CREATE TABLE t (id INT, ts TIMESTAMP);

DESC t;
+-------+-----------+------+-----+-------------------+-----------------------------+
| Field | Type      | Null | Key | Default           | Extra                       |
+-------+-----------+------+-----+-------------------+-----------------------------+
| id    | int(11)   | YES  |     | NULL              |                             |
| ts    | timestamp | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-------+-----------+------+-----+-------------------+-----------------------------+

INSERT INTO t(id)  VALUES (1),(2);

SELECT * FROM t;
+------+---------------------+
| id   | ts                  |
+------+---------------------+
|    1 | 2013-07-22 12:50:05 |
|    2 | 2013-07-22 12:50:05 |
+------+---------------------+

INSERT INTO t  VALUES (3,NULL),(4,'2001-07-22 12:12:12');

SELECT * FROM t;
+------+---------------------+
| id   | ts                  |
+------+---------------------+
|    1 | 2013-07-22 12:50:05 |
|    2 | 2013-07-22 12:50:05 |
|    3 | 2013-07-22 12:51:56 |
|    4 | 2001-07-22 12:12:12 |
+------+---------------------+
```

Converting to Unix epoch:

```
SELECT ts, UNIX_TIMESTAMP(ts) FROM t;
+---------------------+--------------------+
| ts                  | UNIX_TIMESTAMP(ts) |
+---------------------+--------------------+
| 2013-07-22 12:50:05 |         1374490205 |
| 2013-07-22 12:50:05 |         1374490205 |
| 2013-07-22 12:51:56 |         1374490316 |
| 2001-07-22 12:12:12 |          995796732 |
+---------------------+--------------------+
```

Update also changes the timestamp:

```
UPDATE t set id=5 WHERE id=1;

SELECT * FROM t;
+------+---------------------+
| id   | ts                  |
+------+---------------------+
|    5 | 2013-07-22 14:52:33 |
|    2 | 2013-07-22 12:50:05 |
|    3 | 2013-07-22 12:51:56 |
|    4 | 2001-07-22 12:12:12 |
+------+---------------------+
```

Default `NULL` :

```
CREATE TABLE t2 (id INT, ts TIMESTAMP NULL ON UPDATE CURRENT_TIMESTAMP);

INSERT INTO t(id)  VALUES (1),(2);

SELECT * FROM t2;

INSERT INTO t2(id)  VALUES (1),(2);

SELECT * FROM t2;
+------+------+
| id   | ts   |
+------+------+
|    1 | NULL |
|    2 | NULL |
+------+------+

UPDATE t2 SET id=3 WHERE id=1;

SELECT * FROM t2;
+------+---------------------+
| id   | ts                  |
+------+---------------------+
|    3 | 2013-07-22 15:32:22 |
|    2 | NULL                |
+------+---------------------+
```

Only the first timestamp is automatically inserted and updated:

```
CREATE TABLE t3 (id INT, ts1 TIMESTAMP, ts2 TIMESTAMP);

INSERT INTO t3(id)  VALUES (1),(2);

SELECT * FROM t3;
+------+---------------------+---------------------+
| id   | ts1                 | ts2                 |
+------+---------------------+---------------------+
|    1 | 2013-07-22 15:35:07 | 0000-00-00 00:00:00 |
|    2 | 2013-07-22 15:35:07 | 0000-00-00 00:00:00 |
+------+---------------------+---------------------+

DESC t3;
+-------+-----------+------+-----+---------------------+-----------------------------+
| Field | Type      | Null | Key | Default             | Extra                       |
+-------+-----------+------+-----+---------------------+-----------------------------+
| id    | int(11)   | YES  |     | NULL                |                             |
| ts1   | timestamp | NO   |     | CURRENT_TIMESTAMP   | on update CURRENT_TIMESTAMP |
| ts2   | timestamp | NO   |     | 0000-00-00 00:00:00 |                             |
+-------+-----------+------+-----+---------------------+-----------------------------+
```

Explicitly setting a timestamp with the CURRENT_TIMESTAMP function:

```
INSERT INTO t3(id,ts2)  VALUES (3,CURRENT_TIMESTAMP());

SELECT * FROM t3;
+------+---------------------+---------------------+
| id   | ts1                 | ts2                 |
+------+---------------------+---------------------+
|    1 | 2013-07-22 15:35:07 | 0000-00-00 00:00:00 |
|    2 | 2013-07-22 15:35:07 | 0000-00-00 00:00:00 |
|    3 | 2013-07-22 15:38:52 | 2013-07-22 15:38:52 |
+------+---------------------+---------------------+
```

Specifying the timestamp as `NOT NULL`:

```
CREATE TABLE t4 (id INT, ts TIMESTAMP NOT NULL);

INSERT INTO t4(id)  VALUES (1);
SELECT SLEEP(1);
INSERT INTO t4(id,ts) VALUES (2,NULL);

SELECT * FROM t4;
```

See Also

- Data Type Storage Requirements

This page is licensed: CC BY-SA / Gnu FDL

YEAR Data Type

> **Note:**
>
> This page is about the `YEAR` **data type**, not the `YEAR` function.

Syntax

```
YEAR[(4)]
```

Description

A year in four-digit format. The two-digit format (`YEAR(2)`), while supported, is error-prone and has been deprecated since 2012. As of MariaDB 13.0, `YEAR(2)` is no longer accepted by default; set `old_mode=2_DIGIT_YEAR` to allow it again. That setting is itself deprecated and is intended only as a temporary migration aid.

In four-digit format, the allowable values are 1901 to 2155, and 0000. In two-digit format, the allowable values are 70 to 69, representing years from 1970 to 2069. MariaDB displays YEAR values in YYYY format, but allows you to assign values to `YEAR` columns using either strings or numbers.

Inserting numeric zero has a different result for `YEAR(4)` and `YEAR(2)`. For `YEAR(2)`, the value `00` reflects the year 2000. For `YEAR(4)`, the value `0000` reflects the year zero. This only applies to numeric zero. String zero always reflects the year 2000.

Examples

Accepting a string or a number:

```
CREATE TABLE y(y YEAR);

INSERT INTO y VALUES (1990),('2012');

SELECT * FROM y;
+------+
| y    |
+------+
| 1990 |
| 2012 |
+------+
```

With strict_mode set — values out of range:

```
INSERT INTO y VALUES (1005),('3080');
ERROR 1264 (22003): Out of range value for column 'y' at row 1

INSERT INTO y VALUES ('2013-12-12');
ERROR 1265 (01000): Data truncated for column 'y' at row 1

SELECT * FROM y;
+------+
| y    |
+------+
| 1990 |
| 2012 |
+------+
```

With strict_mode unset — values out of range:

```
INSERT INTO y VALUES (1005),('3080');
Query OK, 2 rows affected, 2 warnings (0.05 sec)
Records: 2  Duplicates: 0  Warnings: 2

SHOW WARNINGS;
+---------+------+-------------------------------------------+
| Level   | Code | Message                                   |
+---------+------+-------------------------------------------+
| Warning | 1264 | Out of range value for column 'y' at row 1 |
| Warning | 1264 | Out of range value for column 'y' at row 2 |
+---------+------+-------------------------------------------+

SELECT * FROM y;
+------+
| y    |
+------+
| 1990 |
| 2012 |
| 0000 |
| 0000 |
+------+
```

Truncating:

```
INSERT INTO y VALUES ('2013-12-12');
Query OK, 1 row affected, 1 warning (0.05 sec)

SHOW WARNINGS;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1265 | Data truncated for column 'y' at row 1 |
+---------+------+----------------------------------------+

SELECT * FROM y;
+------+
| y    |
+------+
| 1990 |
| 2012 |
| 0000 |
| 0000 |
| 2013 |
+------+
```

Difference between `YEAR(2)` and `YEAR(4)`, and string and numeric zero:

```
CREATE TABLE y2(y YEAR(4), y2 YEAR(2));
Query OK, 0 rows affected, 1 warning (0.40 sec)

Note (Code 1287): 'YEAR(2)' is deprecated and will be removed in a future release. Please use YEAR(4)
instead

INSERT INTO y2 VALUES(0,0),('0','0');

SELECT YEAR(y),YEAR(y2) FROM y2;
+---------+----------+
| YEAR(y) | YEAR(y2) |
+---------+----------+
|       0 |     2000 |
|    2000 |     2000 |
+---------+----------+
```

See Also

- YEAR() function

This page is licensed: GPLv2, originally from fill_help_tables.sql

String Data Types

See BLOB and TEXT Data Types.

Overview of large object types. This page compares BLOB (binary) and TEXT (character) types, explaining their storage and usage differences.

See BINARY.

Fixed-length binary string type. This type stores a fixed number of bytes, padding with zero bytes if the data is shorter.

See BLOB.

Variable-length binary large object. BLOB columns can store binary data up to 65,535 bytes, suitable for images or other non-text files.

See CHAR.

Fixed-length character string type. CHAR columns store strings of a specified length (0 to 255), padding with spaces if necessary.

See CHARACTER.

Fixed-length character string type. CHARACTER columns store strings of a specified length (0 to 255), padding with spaces if necessary.

See CHAR BYTE.

Fixed-length binary string type. This type stores a fixed number of bytes, padding with zero bytes if the data is shorter.

See CHAR VARYING.

CHAR VARYING is a synonym for the VARCHAR string data type.

See CLOB.

In Oracle mode, CLOB is an alias for the LONGTEXT data type used to store large text objects.

See ENUM.

Complete ENUM type reference: ENUM('v1','v2') syntax, CHARACTER SET/COLLATE options, NULL/empty string defaults, and numeric index sorting guidelines.

See INET4.

IPv4 address data type. Stores IPv4 addresses as 4-byte binary strings for efficient storage and retrieval.

See INET6.

IPv6 address data type. Stores IPv6 addresses as 16-byte binary strings, also supporting IPv4 addresses via mapping.

See JSON Data Type.

Complete JSON Data Type data type guide for MariaDB. Complete reference for syntax, valid values, storage requirements, and range limits for production use.

See LONG and LONG VARCHAR.

LONG and LONG VARCHAR are compatibility synonyms for the MEDIUMTEXT string data type.

See LONG CHAR VARYING.

LONG CHAR VARYING is a compatibility synonym for the MEDIUMTEXT string data type.

See LONG CHARACTER VARYING.

LONG CHARACTER VARYING is a compatibility synonym for the MEDIUMTEXT string data type.

See LONG VARBINARY.

LONG VARBINARY is a compatibility synonym for the MEDIUMBLOB binary data type.

See LONG VARCHAR.

LONG VARCHAR is a compatibility synonym for the MEDIUMTEXT string data type.

See LONG VARCHARACTER.

LONG VARCHARACTER is a compatibility synonym for the MEDIUMTEXT string data type.

See LONGBLOB.

Very large binary object. A LONGBLOB column can store up to 4GB of binary data.

See LONGTEXT.

Complete LONGTEXT type reference: 4GB (2^32-1) storage limit, CHARACTER SET/COLLATE syntax, max_allowed_packet constraints, and JSON/CLOB aliases.

See MEDIUMBLOB.

Medium-sized binary object. A MEDIUMBLOB column can store up to 16MB of binary data.

See MEDIUMTEXT.

Medium-sized character string. A MEDIUMTEXT column can store up to 16MB of text data.

See NATIONAL CHAR.

NATIONAL CHAR is a synonym for the CHAR data type that uses the predefined utf8mb3 character set.

See NATIONAL CHAR VARYING.

NATIONAL CHAR VARYING is a synonym for VARCHAR that uses the predefined utf8 character set.

See NATIONAL CHARACTER VARYING.

NATIONAL CHARACTER VARYING is a synonym for VARCHAR using the predefined utf8 character set.

See NATIONAL CHARACTER.

NATIONAL CHARACTER is a synonym for the CHAR data type using the predefined utf8 character set.

See NATIONAL VARCHAR.

NATIONAL VARCHAR is a synonym for the VARCHAR data type using the predefined utf8 character set.

See NATIONAL VARCHARACTER.

NATIONAL VARCHARACTER is a synonym for VARCHAR using the predefined utf8 character set.

See NCHAR.

NCHAR is a synonym for the fixed-length CHAR string data type using the utf8mb3 character set.

See NCHAR VARCHAR.

NCHAR VARCHAR is a synonym for the VARCHAR string data type using the utf8 character set.

See NCHAR VARCHARACTER.

NCHAR VARCHARACTER is a synonym for VARCHAR using the utf8 character set.

See NCHAR VARYING.

NCHAR VARYING is a synonym for the VARCHAR string data type using the utf8 character set.

See RAW.

In Oracle mode, RAW is a variable-length binary data type synonymous with VARBINARY.

See ROW.

ROW is a data type used in stored programs to store a complete row of data from a cursor or table.

See SET Data Type.

String object with zero or more values from a predefined list. A SET column can store multiple values selected from a list of permitted strings.

See TEXT.

Complete TEXT type reference: TEXT(M) syntax, 65,535 byte maximum, 2-byte length prefix, DEFAULT value support, and indexing constraints rules.

See TINYBLOB.

Very small binary object. A TINYBLOB column can store up to 255 bytes of binary data.

See TINYTEXT.

Very small character string. A TINYTEXT column can store up to 255 characters.

See UUID Data Type.

Official UUID data type reference: 128-bit storage optimization, CAST from CHAR/VARCHAR/BINARY types, RFC4122 string format, and UUIDv6/v7 support.

See VARBINARY.

Variable-length binary string type. VARBINARY columns store binary strings of variable length up to a specified maximum.

See VARCHAR.

Complete VARCHAR reference: VARCHAR(M) syntax, length limits (0-65532 per row), CHARACTER SET/COLLATE options, indexing rules, and trailing spaces.

See VARCHAR2.

Variable-length character string type. VARCHAR2 columns store strings of variable length up to a specified maximum (up to 65,535).

See VARCHARACTER.

Variable-length character string type. VARCHARACTER columns store strings of variable length up to a specified maximum (up to 65,535).

See XMLTYPE.

The XMLTYPE data type, available from MariaDB 12.3, for storing XML data.

See Character Sets and Collations.

Learn about character sets in MariaDB Server. This section details how different character sets and collations impact string storage, comparison, and sorting within your database.

This page is licensed: CC BY-SA / Gnu FDL

BLOB and TEXT Data Types

Description

A `BLOB` is a binary large object that can hold a variable amount of data. The four `BLOB` types are

- TINYBLOB,
- BLOB,
- MEDIUMBLOB, and
- LONGBLOB.

These differ only in the maximum length of the values they can hold.

The `TEXT` types are

- TINYTEXT,

- TEXT,
- MEDIUMTEXT, and
- LONGTEXT.
- JSON (alias for LONGTEXT)

These correspond to the four `BLOB` types and have the same maximum lengths and storage requirements.

`BLOB` and `TEXT` columns can have a `DEFAULT` value.

> **Note:**
>
> It is possible to set a unique index on columns that use the `BLOB` or `TEXT` data types.

Handling Large Data via APIs

When working with very large `BLOB` or `TEXT` values, the data may exceed the limit set by the max_allowed_packet system variable. To avoid this—and to reduce memory consumption on the client—most MariaDB connectors allow you to "stream" data in chunks.

- Connector/C: Use mysql_stmt_send_long_data() to send parameter data in pieces before calling mysql_stmt_execute(). This bypasses `max_allowed_packet` limits and reduces the peak memory footprint (RSS) of the application.
- Connector/J: Use `PreparedStatement.setBinaryStream()` (for `BLOB`) or `PreparedStatement.setCharacterStream()` (for `TEXT`).
- Protocol: These APIs utilize the COM_STMT_SEND_LONG_DATA command, which appends data to a parameter on the server side.

Technical Rules for C/C++

- 0-Based Indexing: Parameter numbering starts at `0`.
- The `is_null` Flag: This must be `0`; if set to nonzero, the server may discard the streamed data.
- Resetting: Use mysql_stmt_reset() to clear all accumulated long data on the server if you need to abort or retry.
- Chunk Size: A practical performance "sweet spot" is between `64` KB and `1` MB per chunk.

See Also

- Store a column as compressed

This page is licensed: GPLv2, originally from fill_help_tables.sql

BINARY

This page describes the BINARY data type. For details about the operator, see Binary Operator.

Syntax

```
BINARY(M)
```

Description

The `BINARY` type is similar to the CHAR type, but stores binary byte strings rather than non-binary character strings. `M` represents the column length in bytes.

It contains no character set, and comparison and sorting are based on the numeric value of the bytes.

If the maximum length is exceeded, and SQL strict mode is not enabled , the extra characters will be dropped with a warning. If strict mode is enabled, an error will occur.

`BINARY` values are right-padded with `0x00` (the zero byte) to the specified length when inserted. The padding is *not* removed on select, so this needs to be taken into account when sorting and comparing, where all bytes are significant. The zero byte, `0x00` is less than a space for comparison purposes.

Use Cases for Zero Length

A `BINARY(0)` or `VARBINARY(0)` column is restricted to an empty byte string or `NULL`.

- **Schema Preservation**: Use these columns when a system expects a specific column to exist, but no data storage is required for your current application.
- **Space-Efficient Indicators**: These columns can act as a two-state indicator where the presence of an empty byte string represents one state and `NULL` represents another.

If you attempt to insert a value other than an empty string, MariaDB returns an `ERROR 1406 (22001)` indicating the data is too long for the column.

Examples

Inserting too many characters, first with strict mode off, then with it on:

```
CREATE TABLE bins (a BINARY(10));

INSERT INTO bins VALUES('12345678901');
Query OK, 1 row affected, 1 warning (0.04 sec)

SELECT * FROM bins;
+------------+
| a          |
+------------+
| 1234567890 |
+------------+

SET sql_mode='STRICT_ALL_TABLES';

INSERT INTO bins VALUES('12345678901');
ERROR 1406 (22001): Data too long for column 'a' at row 1
```

Sorting is performed with the byte value:

```
TRUNCATE bins;

INSERT INTO bins VALUES('A'),('B'),('a'),('b');

SELECT * FROM bins ORDER BY a;
+------+
| a    |
+------+
| A    |
| B    |
| a    |
| b    |
+------+
```

Using CAST to sort as a CHAR instead:

```
SELECT * FROM bins ORDER BY CAST(a AS CHAR);
+------+
| a    |
+------+
| a    |
| A    |
| b    |
| B    |
+------+
```

The field is a `BINARY(10)`, so padding of two '\0's are inserted, causing comparisons that don't take this into account to fail:

```
TRUNCATE bins;

INSERT INTO bins VALUES('12345678');

SELECT a = '12345678', a = '12345678\0\0' from bins;
+----------------+--------------------+
| a = '12345678' | a = '12345678\0\0' |
+----------------+--------------------+
|              0 |                  1 |
+----------------+--------------------+
```

Example of `BINARY`:

```
CREATE TABLE binary_example (
   description VARCHAR(20),
   example BINARY(255)
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer

INSERT INTO binary_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 255, CHAR(7)));

SELECT description, LENGTH(example) AS length
   FROM binary_example;

+---------------------+--------+
| description         | length |
+---------------------+--------+
| Normal foo          |    255 |
| Trailing spaces foo |    255 |
| NULLed              |   NULL |
| Empty               |    255 |
| Maximum             |    255 |
+---------------------+--------+
```

Data Too Long

When `SQL_MODE` is strict (the default), a value is considered "too long" when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `BINARY`:

```
TRUNCATE binary_example;

INSERT INTO binary_example VALUES
   ('Overflow', RPAD('', 256, CHAR(7)));

ERROR 1406 (22001): Data too long for column 'example' at row 1
```

See Also

- CHAR
- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

BLOB

Syntax

```
BLOB[(M)]
```

Description

> **Note:**
>
> If you are handling large binary data that exceeds the `max_allowed_packet` limit, you can stream the data in chunks using specialized API functions like `mysql_stmt_send_long_data()` or `setBinaryStream()`. See Handling Large Data via APIs for more details.

A `BLOB` column with a maximum length of 65,535 (2^{16} - 1) bytes. Each `BLOB` value is stored using a two-byte length prefix that indicates the number of bytes in the value.

An optional length `M` can be given for this type. If this is done, MariaDB creates the column as the smallest `BLOB` type large enough to hold values *`M`* bytes long.

`BLOB` values can also be used to store dynamic columns.

`BLOB` and `TEXT` columns can both be assigned a DEFAULT value.

Indexing

On a column that uses the `BLOB` data type, setting a unique index is now possible.

Note

In previous releases, setting a unique index on a column that uses the `BLOB` data type was not possible. Index would only guarantee the uniqueness of a fixed number of characters.

Oracle Mode

In Oracle mode, `BLOB` is a synonym for `LONGBLOB`.

EXAMPLES

BLOB

Example of `BLOB`:

```
CREATE TABLE blob_example (
   description VARCHAR(20),
   example BLOB
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

```
INSERT INTO blob_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 65535, CHAR(7)));
```

```
SELECT description, LENGTH(example) AS length
   FROM blob_example;

+---------------------+--------+
| description         | length |
+---------------------+--------+
| Normal foo          |      3 |
| Trailing spaces foo |      9 |
| NULLed              |   NULL |
| Empty               |      0 |
| Maximum             |  65535 |
+---------------------+--------+
```

Data Too Long

When `SQL_MODE` is strict (the default) a value is considered "too long" when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `BLOB`:

```
TRUNCATE blob_example;

INSERT INTO blob_example VALUES
   ('Overflow', RPAD('', 65536, CHAR(7)));

ERROR 1406 (22001): Data too long for column 'example' at row 1
```

See Also

- BLOB and TEXT Data Types
- Data Type Storage Requirements
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

CHAR

> **Note:**
>
> This page covers the `CHAR` data type. See CHAR Function for the function.

Syntax

```
[NATIONAL] CHAR[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]
```

Description

A fixed-length string that is always right-padded with spaces to the specified length when stored. `M` represents the column length in characters. The range of `M` is `0` to `255`. If `M` is omitted, the length is `1`.

> **Note:**
>
> Trailing spaces are removed when `CHAR` values are retrieved unless the `PAD_CHAR_TO_FULL_LENGTH` SQL mode is enabled.

If a unique index consists of a column where trailing pad characters are stripped or ignored, inserts into that column where values differ only by the number of trailing pad characters will result in a duplicate-key error.

`CHAR(0)` columns can contain 2 values: an empty string or `NULL`. Such columns cannot be part of an index. The CONNECT storage engine does not support `CHAR(0)`.

Use Cases for Zero Length

A `CHAR(0)` or `VARCHAR(0)` column occupies minimal space and is restricted to two possible values: an empty string (`''`) or `NULL`. You can use these columns for the following purposes:

- **Legacy Compatibility**: Include these columns to maintain compatibility with older applications that require a specific table schema, even if the data is no longer collected.
- **Two-State Flags**: A `CHAR(0) NULL` column can function as a boolean indicator. It uses only one bit of storage to distinguish between a "set" state (the empty string) and an "unset" state (`NULL`).
- **Row Marking**: You can use a `CHAR(0)` column to mark a specific row in a table. For example, if you require only one "active" row, set that row to an empty string while keeping all other rows `NULL`.

The following error occurs if you attempt to insert any character data into a 0-length column: `ERROR 1406 (22001): Data too long for column`.

Examples

Trailing spaces:

```
CREATE TABLE strtest (c CHAR(10));
INSERT INTO strtest VALUES('Maria   ');

SELECT c='Maria',c='Maria   ' FROM strtest;
+-----------+--------------+
| c='Maria' | c='Maria   ' |
+-----------+--------------+
|         1 |            1 |
+-----------+--------------+

SELECT c LIKE 'Maria',c LIKE 'Maria   ' FROM strtest;
+----------------+-------------------+
| c LIKE 'Maria' | c LIKE 'Maria   ' |
+----------------+-------------------+
|              1 |                 0 |
+----------------+-------------------+
```

Example of `CHAR`:

```
CREATE TABLE char_example (
   description VARCHAR(20),
   example CHAR(255)
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

```
INSERT INTO char_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 255, 'x'));
```

```
SELECT description, LENGTH(example) AS length FROM char_example;

+---------------------+--------+
| description         | length |
+---------------------+--------+
| Normal foo          |      3 |
| Trailing spaces foo |      3 |
| NULLed              |   NULL |
| Empty               |      0 |
| Maximum             |    255 |
+---------------------+--------+
```

Data Too Long

When `SQL_MODE` is strict (the default), a value is considered "too long" when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `CHAR`:

```
TRUNCATE char_example;

INSERT INTO char_example VALUES
   ('Overflow', RPAD('', 256, 'x'));

ERROR 1406 (22001): Data too long for column 'example' at row 1
```

NO PAD Collations

`NO PAD` collations regard trailing spaces as normal characters. You can get a list of all `NO PAD` collations by querying the Information Schema Collations table, for example:

```
SELECT collation_name FROM information_schema.collations
  WHERE collation_name LIKE "%nopad%";
+------------------------------+
| collation_name               |
+------------------------------+
| big5_chinese_nopad_ci        |
| big5_nopad_bin               |
...
```

See Also

- CHAR Function
- VARCHAR
- BINARY
- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

CHARACTER

Overview

This is a synonym for CHAR.

EXAMPLES

```
CREATE TABLE character_example (
  example CHARACTER
);
```

```
SHOW CREATE TABLE character_example\G
```

```
*************************** 1. row ***************************
       Table: character_example
Create Table: CREATE TABLE `character_example` (
  `example` char(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

CHAR BYTE

Description

The `CHAR BYTE` data type is an alias for the BINARY data type. This is a compatibility feature.

EXAMPLES

```
CREATE TABLE char_byte_example (
  example CHAR BYTE
);
```

```
SHOW CREATE TABLE char_byte_example\G
```

```
*************************** 1. row ***************************
       Table: char_byte_example
Create Table: CREATE TABLE `char_byte_example` (
  `example` binary(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is licensed: GPLv2, originally from fill_help_tables.sql

CHAR VARYING

Overview

This is a synonym for VARCHAR.

EXAMPLES

```
CREATE TABLE char_varying_example (
  example CHAR VARYING(32)
);
```

```
SHOW CREATE TABLE char_varying_example\G
```

```
*************************** 1. row ***************************
       Table: char_varying_example
Create Table: CREATE TABLE `char_varying_example` (
  `example` varchar(32) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

CLOB

Overview

This is a synonym for LONGTEXT.

EXAMPLES

```
SET sql_mode='oracle';

CREATE TABLE clob_example (
  example CLOB
);
```

```
SHOW CREATE TABLE clob_example\G
```

```
*************************** 1. row ***************************
       Table: clob_example
Create Table: CREATE TABLE "clob_example" (
  "example" longtext DEFAULT NULL
)
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

ENUM

Syntax

```
ENUM('value1','value2',...) [CHARACTER SET charset_name] [COLLATE collation_name]
```

Description

An enumeration. A string object that can have only one value, chosen from the list of values 'value1', 'value2', …, `NULL` or the special '' error value. In theory, an `ENUM` column can have a maximum of 65,535 distinct values; in practice, the real maximum depends on many factors. `ENUM` values are represented internally as integers.

Trailing spaces are automatically stripped from `ENUM` values on table creation.

`ENUM` values require relatively little storage space compared to strings, either one or two bytes depending on the number of enumeration values.

NULL and empty values

An `ENUM` can also contain `NULL` and empty values. If the `ENUM` column is declared to permit `NULL` values, `NULL` becomes a valid value, as well as the default value (see below). If strict SQL Mode is not enabled, and an invalid value is inserted into an `ENUM`, a special empty string, with an index value of zero (see Numeric index, below), is inserted, with a warning. This may be confusing, because the empty string is also a possible value, and the only difference if that is this case its index is not 0. Inserting will fail with an error if strict mode is active.

If a `DEFAULT` clause is missing, the default value will be:

- `NULL` if the column is nullable;
- otherwise, the first value in the enumeration.

Numeric index

`ENUM` values are indexed numerically in the order they are defined, and sorting will be performed in this numeric order. We suggest not using `ENUM` to store numerals, as there is little to no storage space benefit, and it is easy to

confuse the enum integer with the enum numeral value by leaving out the quotes.

An `ENUM` defined as `ENUM('apple','orange','pear')` would have the following index values:

Index	Value
NULL	NULL
0	''
1	'apple'
2	'orange'
3	'pear'

Examples

```
CREATE TABLE fruits (
  id INT NOT NULL auto_increment PRIMARY KEY,
  fruit ENUM('apple','orange','pear'),
  bushels INT);

DESCRIBE fruits;
+---------+-------------------------------+------+-----+---------+----------------+
| Field   | Type                          | Null | Key | Default | Extra          |
+---------+-------------------------------+------+-----+---------+----------------+
| id      | int(11)                       | NO   | PRI | NULL    | auto_increment |
| fruit   | enum('apple','orange','pear') | YES  |     | NULL    |                |
| bushels | int(11)                       | YES  |     | NULL    |                |
+---------+-------------------------------+------+-----+---------+----------------+

INSERT INTO fruits
    (fruit,bushels) VALUES
    ('pear',20),
    ('apple',100),
    ('orange',25);

INSERT INTO fruits
    (fruit,bushels) VALUES
    ('avocado',10);
ERROR 1265 (01000): Data truncated for column 'fruit' at row 1

SELECT * FROM fruits;
+----+--------+---------+
| id | fruit  | bushels |
+----+--------+---------+
|  1 | pear   |      20 |
|  2 | apple  |     100 |
|  3 | orange |      25 |
+----+--------+---------+
```

Selecting by numeric index:

```
SELECT * FROM fruits WHERE fruit=2;
+----+--------+---------+
| id | fruit  | bushels |
+----+--------+---------+
|  3 | orange |      25 |
+----+--------+---------+
```

Sorting is according to the index value:

```
CREATE TABLE enums (a ENUM('2','1'));

INSERT INTO enums VALUES ('1'),('2');

SELECT * FROM enums ORDER BY a ASC;
+------+
| a    |
+------+
| 2    |
| 1    |
+------+
```

It's easy to get confused between returning the enum integer with the stored value, so we don't suggest using `ENUM` to store numerals. The first example returns the 1st indexed field ('2' has an index value of 1, as it's defined first), while the second example returns the string value '1'.

```
SELECT * FROM enums WHERE a=1;
+------+
| a    |
+------+
| 2    |
+------+

SELECT * FROM enums WHERE a='1';
+------+
| a    |
+------+
| 1    |
+------+
```

Example of `ENUM`:

```
CREATE TABLE enum_example (
  description VARCHAR(20),
  example ENUM('Alpha', 'Beta', 'Gamma', 'RC', 'Stable')
);
```

```
INSERT INTO enum_example VALUES
  ('foo', 'Beta'),
  ('bar', 'RC'),
  ('baz', 'Alpha'),
  ('bob', 5);
```

```
SELECT * FROM enum_example;

+-------------+---------+
| description | example |
+-------------+---------+
| foo         | Beta    |
| bar         | RC      |
| baz         | Alpha   |
| bob         | Stable  |
+-------------+---------+
```

See Also

- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

INET4

> **Note:**
>
> INET4 is available from MariaDB 10.10.

Syntax

```
INET4
```

Description

`INET4` is a data type to store IPv4 addresses, as 4-byte binary strings.

> **CURRENT**
>
> Casting from INET4 data types to `INET6` is permitted, allowing for example comparisons between the two data types, and for `INET4` values to be inserted into `INET6` columns.
>
> **< 11.3**
>
> Casting from INET4 data types to `INET6` is **not** permitted.

Examples

```
CREATE OR REPLACE TABLE t1 (a INET4);

INSERT INTO t1 VALUES('0.0.0.0'), ('255.10.0.0'), ('255.255.255.255');

INSERT INTO t1 VALUES (0xa0000001);
INSERT INTO t1 VALUES (0xf0000000);
INSERT INTO t1 VALUES (0xff000001);

SELECT HEX(a), a FROM t1 ORDER BY a;
+----------+-----------------+
| HEX(a)   | a               |
+----------+-----------------+
| 00000000 | 0.0.0.0         |
| A0000001 | 160.0.0.1       |
| F0000000 | 240.0.0.0       |
| FF000001 | 255.0.0.1       |
| FF0A0000 | 255.10.0.0      |
| FFFFFFFF | 255.255.255.255 |
+----------+-----------------+
```

Casting from `INET4` to INET6 is permitted, allowing direct inserts.

> **CURRENT**
>
> ```
> CREATE TABLE t1 (a INET6);
>
> INSERT INTO t1 VALUES('0.0.0.0'), ('255.10.0.0'), ('255.255.255.255');
> Query OK, 3 rows affected (0.027 sec)
> ```
>
> Comparisons are also permitted:

```
CREATE OR REPLACE TABLE t1 (i4 INET4, i6 INET6);
INSERT INTO t1 VALUES('10.10.10.10','::ffff:192.168.0.1');

SELECT LEAST(i4,i6) FROM t1;
+--------------------+
| LEAST(i4,i6)       |
+--------------------+
| ::ffff:10.10.10.10 |
+--------------------+
```

< 11.3

```
CREATE TABLE t1 (a INET6);

INSERT INTO t1 VALUES('0.0.0.0'), ('255.10.0.0'), ('255.255.255.255');
ERROR 1292 (22007): Incorrect inet6 value: '0.0.0.0' for column `test`.`t1`.`a` at row 1
```

This page is licensed: CC BY-SA / Gnu FDL

INET6

> **Note:**
>
> `INET6` is available from MariaDB 10.5.

Syntax

```
INET6
```

Description

The `INET6` data type is intended for storage of IPv6 addresses, as well as IPv4 addresses assuming conventional mapping of IPv4 addresses into IPv6 addresses.

Both short and long IPv6 notation are permitted, according to RFC-5952.

- Values are stored as a 16-byte fixed length binary string, with most significant byte first.
- Storage engines see `INET6` as `BINARY(16)`.
- Clients see `INET6` as `CHAR(39)` and get text representation on retrieval.

The IPv4-compatible notation is considered as deprecated. It is supported for compatibility with the INET6_ATON function, which also understands this format. It's recommended to use the mapped format to store IPv4 addresses in `INET6`.

When an IPv4 mapped (or compatible) value is stored in `INET6`, it still occupies 16 bytes:

Retrieval

On retrieval, in the client-server text protocol, `INET6` values are converted to the short text representation, according to RFC-5952, that is with all leading zeroes in each group removed and with consequent zero groups compressed.

Besides creating one's own stored function, there is no a way to retrieve an `INET6` value using long text representation.

Casting

- CAST from a character string to `INET6` understands addresses in short or long text notation (including IPv4 mapped and compatible addresses). `NULL` is returned if the format is not understood.
- `CAST` from a binary string to `INET6` requires a 16-byte string as an argument. `NULL` is returned if the argument length is not equal to 16.
- `CAST` from other data types to `INET6` first converts data to a character string, then `CAST` from character string to `INET6` is applied.
- `CAST` from `INET6` to CHAR returns short text address notation.
- `CAST` from `INET6` to BINARY returns its 16-byte binary string representation.
- `CAST` from `INET6` to data types other than CHAR (e.g. `SIGNED`, `UNSIGNED`, `TIME`, etc) returns an error.

Comparisons

An `INET6` expression can be compared to:

- another `INET6` expression
- a character string expression with a text (short or long) address representation:
- a 16-byte binary string expression.

> **Warning:**
>
> Attempting to compare `INET6` to an expression of any other data type returns an error.

Mixing INET6 Values for Result

An `INET6` expression can be mixed for result (i.e. UNION, CASE..THEN, COALESCE etc) with:

- another `INET6` expression. The resulting data type is `INET6`.
- a character string in text (short or long) address representation. The result data type is `INET6`. The character string counterpart is automatically converted to `INET6`. If the string format is not understood, it's converted with a warning to either `NULL` or to '::', depending on the `NULL`-ability of the result.
- a 16-byte binary string. The resulting data type is `INET6`. The binary string counterpart is automatically converted to `INET6`. If the length of the binary string is not equal to 16, it's converted with a warning to `NULL` or to '::' depending on the `NULL`-ability of the result.

Attempts to mix `INET6` for result with other data types will return an error.

Mixing `INET6` with other data types for LEAST and GREATEST, when mixing for comparison and mixing for result are involved at the same time, uses the same rules with mixing for result, described in the previous paragraphs.

Functions and Operators

- HEX() with an INET6 argument returns a hexadecimal representation of the underlying 16-byte binary string
- Arithmetic operators (+,-,*,/,MOD,DIV) are not supported for INET6. This may change in the future.
- The INET6_ATON function now understands INET6 values as an argument
- The prototypes of the IS_IPV4_COMPAT and I S_IPV4_MAPPED functions have changed from `a BINARY(16)` to `a INET6`,

> **CURRENT**
>
> When the argument for the aforementioned two functions is not `INET6`, automatic implicit `CAST` to `INET6` is applied. As a consequence, both functions understand arguments in both text representation and binary(16) representation.

> **< 10.5**
>
> When the argument for the aforementioned two functions is not `INET6`, automatic implicit `CAST` to `INET6` is **not** applied.

Prepared Statement Parameters

INET6 understands both text and binary(16) address representation in prepared statement parameters (PREPARE..EXECUTE and EXECUTE IMMEDIATE statements).

Migration between BINARY(16) and INET6

> **CURRENT**
>
> You may have used BINARY(16) as a storage for IPv6 internet addresses, in combination with INET6_ATON and INET6_NTOA to respectively insert and retrieve data.
>
> However, you can ALTER `BINARY(16)` columns storing IPv6 addresses to `INET6`. After such an alter, there is no a need to use `INET6_ATON()` and `INET6_NTOA()`. Addresses can be inserted and retrieved directly.
>
> **< 10.5**
>
> You may use BINARY(16) as a storage for IPv6 internet addresses, in combination with INET6_ATON and INET6_NTOA to respectively insert and retrieve data.

It is also possible to convert `INET6` columns to `BINARY(16)` and continue using the data in combination with `INET6_NTOA()` and `INET6_ATON()`.

Examples

```
CREATE TABLE t1 (a INET6);
```

Inserting using short text address notation:

```
INSERT INTO t1 VALUES ('2001:db8::ff00:42:8329');
```

Long text address notation:

```
INSERT INTO t1 VALUES ('2001:0db8:0000:0000:0000:ff00:0042:8329');
```

16-byte binary string notation:

```
INSERT INTO t1 VALUES (0x20010DB8000000000000FF0000428329);
INSERT INTO t1 VALUES (UNHEX('20010DB8000000000000FF0000428329'));
```

IPv4 addresses, using IPv4-mapped and IPv4-compatible notations:

```
INSERT INTO t1 VALUES ('::ffff:192.0.2.128'); -- mapped
INSERT INTO t1 VALUES ('::192.0.2.128'); -- compatible
```

```
SELECT * FROM t1;
+------------------------+
| a                      |
+------------------------+
| 2001:db8::ff00:42:8329 |
| 2001:db8::ff00:42:8329 |
| 2001:db8::ff00:42:8329 |
| 2001:db8::ff00:42:8329 |
| ::ffff:192.0.2.128     |
| ::192.0.2.128          |
+------------------------+
```

IPv4 mapped (or compatible) values still occupy 16 bytes:

```
CREATE OR REPLACE TABLE t1 (a INET6);

INSERT INTO t1 VALUES ('::ffff:192.0.2.128');

SELECT * FROM t1;
+--------------------+
| a                  |
+--------------------+
| ::ffff:192.0.2.128 |
+--------------------+

SELECT HEX(a) FROM t1;
+----------------------------------+
| HEX(a)                           |
+----------------------------------+
| 00000000000000000000FFFFC0000280 |
+----------------------------------+
```

Casting from `INET6` to anything other than CHAR returns an error:

```
SELECT CAST(a AS DECIMAL) FROM t1;

ERROR 4079 (HY000): Illegal parameter data type inet6 for operation 'decimal_typecast'
```

Comparison Examples

Comparison with another `INET6` expression:

```
CREATE OR REPLACE TABLE t1 (a INET6);
    CREATE OR REPLACE TABLE t2 (a INET6);

    INSERT INTO t1 VALUES ('2001:db8::ff00:42:8328'),('2001:db8::ff00:42:8329');
    INSERT INTO t2 VALUES ('2001:db8::ff00:42:832a'),('2001:db8::ff00:42:8329');

    SELECT t1.* FROM t1,t2 WHERE t1.a=t2.a;
    +------------------------+
    | a                      |
    +------------------------+
    | 2001:db8::ff00:42:8329 |
    +------------------------+
```

With a character string expression with a text (short or long) address representation:

```
CREATE OR REPLACE TABLE t1 (a INET6);

    INSERT INTO t1 VALUES ('2001:db8::ff00:42:8329');

    SELECT * FROM t1 WHERE a='2001:db8::ff00:42:8329';
    +------------------------+
    | a                      |
    +------------------------+
    | 2001:db8::ff00:42:8329 |
    +------------------------+
```

With a 16-byte binary string expression:

```
CREATE OR REPLACE TABLE t1 (a INET6);

    INSERT INTO t1 VALUES ('2001:db8::ff00:42:8329');

    SELECT * FROM t1 WHERE a=X'20010DB8000000000000FF0000428329';
    +------------------------+
    | a                      |
    +------------------------+
    | 2001:db8::ff00:42:8329 |
    +------------------------+
```

With an expression of another data type:

```
SELECT * FROM t1 WHERE a=1;
ERROR 4078 (HY000): Illegal parameter data types inet6 and int for operation '='
```

Mixing for Result Examples

Mixed with another `INET6` expression, returning an `INET6` data type:

```
CREATE OR REPLACE TABLE t1 (a INET6, b INET6);

    INSERT INTO t1 VALUES (NULL,'2001:db8::ff00:42:8329');

    SELECT a FROM t1 UNION SELECT b FROM t1;
    +------------------------+
    | a                      |
    +------------------------+
    | NULL                   |
    | 2001:db8::ff00:42:8329 |
    +------------------------+

    SELECT COALESCE(a, b) FROM t1;
    +------------------------+
    | COALESCE(a, b)         |
    +------------------------+
    | 2001:db8::ff00:42:8329 |
    +------------------------+
```

Mixed with a character string in text (short or long) address representation:

```
CREATE OR REPLACE TABLE t1 (a INET6, b VARCHAR(64));

    INSERT INTO t1 VALUES (NULL,'2001:db8::ff00:42:8328');

    INSERT INTO t1 VALUES (NULL,'2001:db8::ff00:42:832a garbage');

    SELECT COALESCE(a,b) FROM t1;
    +------------------------+
    | COALESCE(a,b)          |
    +------------------------+
    | 2001:db8::ff00:42:8328 |
    | NULL                   |
    +------------------------+
    2 rows in set, 1 warning (0.001 sec)

    SHOW WARNINGS;
    +---------+------+-------------------------------------------------------------+
    | Level   | Code | Message                                                     |
    +---------+------+-------------------------------------------------------------+
    | Warning | 1292 | Incorrect inet6 value: '2001:db8::ff00:42:832a garbage' |
    +---------+------+-------------------------------------------------------------+
```

Mixed with a 16-byte binary string:

```
CREATE OR REPLACE TABLE t1 (a INET6, b VARBINARY(16));

    INSERT INTO t1 VALUES (NULL,CONCAT(0xFFFF,REPEAT(0x0000,6),0xFFFF));

    INSERT INTO t1 VALUES (NULL,0x00/*garbage*/);

    SELECT COALESCE(a,b) FROM t1;
    +---------------+
    | COALESCE(a,b) |
    +---------------+
    | ffff::ffff    |
    | NULL          |
    +---------------+
    2 rows in set, 1 warning (0.001 sec)

    SHOW WARNINGS;
    +---------+------+------------------------------+
    | Level   | Code | Message                      |
    +---------+------+------------------------------+
    | Warning | 1292 | Incorrect inet6 value: '\x00' |
    +---------+------+------------------------------+
```

Mixing with other data types:

```
SELECT CAST('ffff::ffff' AS INET6) UNION SELECT 1;
ERROR 4078 (HY000): Illegal parameter data types inet6 and int for operation 'UNION'
```

Functions and Operators Examples

HEX with an `INET6` argument returning a hexadecimal representation:

```
SELECT HEX(CAST('2001:db8::ff00:42:8329' AS INET6));
    +---------------------------------------------+
    | HEX(CAST('2001:db8::ff00:42:8329' AS INET6)) |
    +---------------------------------------------+
    | 20010DB8000000000000FF0000428329             |
    +---------------------------------------------+
```

INET6_ATON now understands `INET6` values as an argument:

```
CREATE OR REPLACE TABLE t1 (a INET6);

    INSERT INTO t1 VALUES ('2001:db8::ff00:42:8329');

    SELECT a, HEX(INET6_ATON(a)) FROM t1;
    +-----------------------+----------------------------------+
    | a                     | HEX(INET6_ATON(a))               |
    +-----------------------+----------------------------------+
    | 2001:db8::ff00:42:8329 | 20010DB8000000000000FF0000428329 |
    +-----------------------+----------------------------------+
```

IS_IPV4_COMPAT and IS_IPV4_MAPPED prototype now `a BINARY(16))`:

```
CREATE OR REPLACE TABLE t1 (a INET6);

    INSERT INTO t1 VALUES ('2001:db8::ff00:42:8329');
    INSERT INTO t1 VALUES ('::ffff:192.168.0.1');
    INSERT INTO t1 VALUES ('::192.168.0.1');

    SELECT a, IS_IPV4_MAPPED(a), IS_IPV4_COMPAT(a) FROM t1;
    +-----------------------+------------------+------------------+
    | a                     | IS_IPV4_MAPPED(a) | IS_IPV4_COMPAT(a) |
    +-----------------------+------------------+------------------+
    | 2001:db8::ff00:42:8329 |                0 |                0 |
    | ::ffff:192.168.0.1     |                1 |                0 |
    | ::192.168.0.1          |                0 |                1 |
    +-----------------------+------------------+------------------+
```

Automatic implicit `CAST` to `INET6`:

```
CREATE OR REPLACE TABLE t1 (
      a INET6,
      b VARCHAR(39) DEFAULT a
    );

    INSERT INTO t1 (a) VALUES ('ffff::ffff'),('::ffff:192.168.0.1');

    SELECT a, IS_IPV4_MAPPED(a), b, IS_IPV4_MAPPED(b) FROM t1;
    +-------------------+------------------+-------------------+------------------+
    | a                 | IS_IPV4_MAPPED(a) | b                 | IS_IPV4_MAPPED(b) |
    +-------------------+------------------+-------------------+------------------+
    | ffff::ffff        |                0 | ffff::ffff        |                0 |
    | ::ffff:192.168.0.1 |                1 | ::ffff:192.168.0.1 |                1 |
    +-------------------+------------------+-------------------+------------------+

    CREATE OR REPLACE TABLE t1 (
      a INET6,
      b BINARY(16) DEFAULT UNHEX(HEX(a))
    );

    INSERT INTO t1 (a) VALUES ('ffff::ffff'),('::ffff:192.168.0.1');

    SELECT a, IS_IPV4_MAPPED(a), HEX(b), IS_IPV4_MAPPED(b) FROM t1;
    +-------------------+------------------+----------------------------------+------------------+
    | a                 | IS_IPV4_MAPPED(a) | HEX(b)                           | IS_IPV4_MAPPED(b) |
    +-------------------+------------------+----------------------------------+------------------+
    | ffff::ffff        |                0 | FFFF00000000000000000000FFFF |                0 |
    | ::ffff:192.168.0.1 |                1 | 00000000000000000000FFFFC0A80001 |                1 |
    +-------------------+------------------+----------------------------------+------------------+
```

Prepared Statement Parameters Examples

```
CREATE OR REPLACE TABLE t1 (a INET6);

EXECUTE IMMEDIATE 'INSERT INTO t1 VALUES (?)' USING 'ffff::fffe';
EXECUTE IMMEDIATE 'INSERT INTO t1 VALUES (?)' USING X'FFFF00000000000000000000FFFF';

SELECT * FROM t1;
+------------+
| a          |
+------------+
| ffff::fffe |
| ffff::ffff |
+------------+

EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING 'ffff::fffe';
+------------+
| a          |
+------------+
| ffff::fffe |
+------------+

EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING X'FFFF00000000000000000000FFFF';
+------------+
| a          |
+------------+
| ffff::ffff |
+------------+
```

Migration between BINARY(16) and INET6 Examples

CURRENT

```
CREATE OR REPLACE TABLE t1 (a BINARY(16));

INSERT INTO t1 VALUES (INET6_ATON('ffff::ffff'));

SELECT INET6_NTOA(a) FROM t1;
+---------------+
| INET6_NTOA(a) |
+---------------+
| ffff::ffff    |
+---------------+
```

Migrating to `INET6`:

```
ALTER TABLE t1 MODIFY a INET6;

INSERT INTO t1 VALUES ('ffff::fffe');

SELECT * FROM t1;
+------------+
| a          |
+------------+
| ffff::ffff |
| ffff::fffe |
+------------+
```

< 10.5

There's no conversion you can use:

```
CREATE OR REPLACE TABLE t1 (a BINARY(16));

INSERT INTO t1 VALUES (INET6_ATON('ffff::ffff'));

SELECT INET6_NTOA(a) FROM t1;
+---------------+
| INET6_NTOA(a) |
+---------------+
| ffff::ffff    |
+---------------+
```

Migration from `INET6` to `BINARY(16)`:

```
CREATE OR REPLACE TABLE t1 (a INET6);

INSERT INTO t1 VALUES ('2001:db8::ff00:42:8329');
INSERT INTO t1 VALUES ('::ffff:192.168.0.1');
INSERT INTO t1 VALUES ('::192.168.0.1');

ALTER TABLE t1 MODIFY a BINARY(16);

SELECT INET6_NTOA(a) FROM t1;
+------------------------+
| INET6_NTOA(a)          |
+------------------------+
| 2001:db8::ff00:42:8329 |
| ::ffff:192.168.0.1     |
| ::192.168.0.1          |
+------------------------+
```

Casting from INET4 to INET6

CURRENT

Casting from INET4 data types to `INET6` is permitted, allowing `INET4` values to be inserted into `INET6` columns.

```
CREATE TABLE t1 (a INET6);

INSERT INTO t1 VALUES('0.0.0.0'), ('255.10.0.0'), ('255.255.255.255');
Query OK, 3 rows affected (0.027 sec)
```

< 11.3

Casting from INET4 data types to `INET6` is **not** permitted. You get an error if you try:

```
CREATE TABLE t1 (a INET6);

INSERT INTO t1 VALUES('0.0.0.0'), ('255.10.0.0'), ('255.255.255.255');
ERROR 1292 (22007): Incorrect inet6 value: '0.0.0.0' for column `test`.`t1`.`a` at row 1
```

See Also

- IS_IPV6
- INET6_ATON
- INET6_NTOA
- Working with IPv6 in MariaDB - the INET6 datatype (video)

This page is licensed: CC BY-SA / Gnu FDL

JSON Data Type

The `JSON` alias was added to make it possible to use JSON columns in statement based replication from MySQL to MariaDB and to make it possible for MariaDB to read mysqldumps from MySQL.

`JSON` is an alias for `LONGTEXT COLLATE utf8mb4_bin` introduced for compatibility reasons with MySQL's JSON data type. MariaDB implements this as a LONGTEXT rather, as the JSON data type contradicts the SQL:2016 standard, and MariaDB's benchmarks indicate that performance is at least equivalent.

In order to ensure that a valid json document is inserted, the JSON_VALID function can be used as a CHECK constraint. This constraint is automatically included for types using the `JSON` alias.

The assigned text value is retained verbatim. If a value fails `JSON_VALID()`, an error is raised. This `CHECK` constraint can also be manually added to any `LONGTEXT` field. When a JSON object contains duplicate keys, only the first key-value pair is accessible via functions like `JSON_EXTRACT()`.

Examples

```
CREATE TABLE t (j JSON);

DESC t;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| j     | longtext | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
```

With validation:

```
CREATE TABLE t2 (
  j JSON
  CHECK (JSON_VALID(j))
);

INSERT INTO t2 VALUES ('invalid');
ERROR 4025 (23000): CONSTRAINT `j` failed for `test`.`t2`

INSERT INTO t2 VALUES ('{"id": 1, "name": "Monty"}');
Query OK, 1 row affected (0.13 sec)
```

JSON example:

```
CREATE TABLE json_example (
  description VARCHAR(20),
  example JSON
);
```

```
SHOW CREATE TABLE json_example\G
```

```
*************************** 1. row ***************************
       Table: json_example
Create Table: CREATE TABLE `json_example` (
  `description` varchar(20) DEFAULT NULL,
  `example` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL CHECK (json_valid(`example`))
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

```
INSERT INTO json_example VALUES
  ('Array', '[ 1, 2, 3 ]'),
  ('Dictionary', '{ "a": 1, "b": 2 }'),
  ('Duplicates', '{ "a":1,"b":2, "a":3,"b": 4,"b":5}');
```

```
SELECT * FROM json_example;
```

```
+-------------+-----------------------------------+
| description | example                           |
+-------------+-----------------------------------+
| Array       | [ 1, 2, 3 ]                       |
| Dictionary  | { "a": 1, "b": 2 }                |
| Duplicates  | { "a":1,"b":2, "a":3,"b": 4,"b":5} |
+-------------+-----------------------------------+
```

```
SELECT description, JSON_EXTRACT(example, '$.b')
FROM json_example;
```

```
+-------------+-----------------------------+
| description | JSON_EXTRACT(example, '$.b') |
+-------------+-----------------------------+
| Array       | NULL                        |
| Dictionary  | 2                           |
| Duplicates  | 2                           |
+-------------+-----------------------------+
```

```
INSERT INTO json_example VALUES
  ('Invalid', '{');
```

```
ERROR 4025 (23000): CONSTRAINT `json_example.example` failed for `test`.`json_example`
```

Replicating JSON Data Between MySQL and MariaDB

The `JSON` type in MySQL stores the JSON object in a compact form, not as LONGTEXT as in MariaDB.
This means that row-based replication will not work for `JSON` types from MySQL to MariaDB.

There are a few different ways to solve this:

- Use statement-based replication.
- Change the `JSON` column to type `TEXT` in MySQL.
- If you must use row-based replication and cannot change the MySQL master from `JSON` to `TEXT`, you can try to introduce an intermediate MySQL replica, and change the column type from `JSON` to `TEXT` on it. Then you replicate from this intermediate slave to MariaDB.

Converting a MySQL TABLE with JSON Fields to MariaDB

MariaDB can't directly access MySQL's `JSON` format.

There are a few different ways to move the table to MariaDB:

- From MariaDB 10.5.7, you can use the mysql_json plugin. See Making MariaDB understand MySQL JSON (blog post).
- Change the JSON column to type `TEXT` in MySQL. After this, MariaDB can directly use the table without any need for a dump and restore.
- Use mysqldump to copy the table.

Differences Between MySQL JSON Strings and MariaDB JSON Strings

- In MySQL, JSON is an object and is compared according to json values. In MariaDB JSON strings are normal strings and compared as strings. One exception is when using JSON_EXTRACT() in which case strings are unescaped before comparison.

See Also

- JSON Functions
- CONNECT JSON Table Type
- MDEV-9144

This page is licensed: CC BY-SA / Gnu FDL

LONG and LONG VARCHAR

`LONG` and `LONG VARCHAR` are synonyms for MEDIUMTEXT.

```
CREATE TABLE t1 (a LONG, b LONG VARCHAR);

DESC t1;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| a     | mediumtext | YES  |     | NULL    |       |
| b     | mediumtext | YES  |     | NULL    |       |
+-------+------------+------+-----+---------+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

LONG CHAR VARYING

Overview

See MEDIUMTEXT.

EXAMPLES

```
CREATE TABLE long_char_varying_example (
  example LONG CHAR VARYING
);
```

```
SHOW CREATE TABLE long_char_varying_example\G
```

```
*************************** 1. row ***************************
       Table: long_char_varying_example
Create Table: CREATE TABLE `long_char_varying_example` (
  `example` mediumtext DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

LONG CHARACTER VARYING

Overview

See MEDIUMTEXT.

EXAMPLES

```
CREATE TABLE long_character_varying_example (
  example LONG CHARACTER VARYING
);
```

```
SHOW CREATE TABLE long_character_varying_example\G
```

```
*************************** 1. row ***************************
       Table: long_character_varying_example
Create Table: CREATE TABLE `long_character_varying_example` (
  `example` mediumtext DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

LONG VARBINARY

Overview

See MEDIUMBLOB.

EXAMPLES

```
CREATE TABLE long_varbinary_example (
  example LONG VARBINARY
);
```

```
SHOW CREATE TABLE long_varbinary_example\G
```

```
*************************** 1. row ***************************
       Table: long_varbinary_example
Create Table: CREATE TABLE `long_varbinary_example` (
  `example` mediumblob DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

LONG VARCHAR

Overview

See MEDIUMTEXT.

EXAMPLES

```
CREATE TABLE long_varchar_example (
  example LONG VARCHAR
);
```

```
SHOW CREATE TABLE long_varchar_example\G
```

```
*************************** 1. row ***************************
       Table: long_varchar_example
Create Table: CREATE TABLE `long_varchar_example` (
  `example` mediumtext DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

LONG VARCHARACTER

Overview

See MEDIUMTEXT.

EXAMPLES

```
CREATE TABLE long_varcharacter_example (
  example LONG VARCHARACTER
);
```

```
SHOW CREATE TABLE long_varcharacter_example\G
```

```
*************************** 1. row ***************************
       Table: long_varcharacter_example
Create Table: CREATE TABLE `long_varcharacter_example` (
  `example` mediumtext DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

LONGBLOB

Syntax

```
LONGBLOB
```

Description

A BLOB column with a maximum length of 4,294,967,295 bytes (2^{32} - 1), or 4GB. The effective maximum length of `LONGBLOB` columns depends on the configured maximum packet size in the client/server protocol and available memory. Each `LONGBLOB` value is stored using a four-byte length prefix that indicates the number of bytes in the value.

Oracle Mode

In Oracle mode, `BLOB` is a synonym for `LONGBLOB`.

Examples

```
CREATE TABLE longblob_example (
   description VARCHAR(20),
   example LONGBLOB
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

Note that the maximum size of a `LONGBLOB` is so large that it cannot be sent to the server without breaking the value up into chunks (something that the command-line client cannot do). For values larger than 16M you can increase the max_allowed_packet size up to a maximum of 1024M to increase the allowed size of non-chunked values.

```
INSERT INTO longblob_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 4294967295, CHAR(7)));
```

```
ERROR 1301 (HY000): Result of rpad() was larger than max_allowed_packet (16777216) - truncated
```

Data Too Long

When `SQL_MODE` is strict (the default) a value is considered “too long” when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `LONGBLOB`:

```
TRUNCATE longblob_example;

INSERT INTO longblob_example VALUES
   ('Overflow', RPAD('', 4294967296, CHAR(7)));
```

```
ERROR 1301 (HY000): Result of rpad() was larger than max_allowed_packet (16777216) - truncated
```

See Also

- BLOB
- BLOB and TEXT Data Types
- Data Type Storage Requirements
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

LONGTEXT

Syntax

```
LONGTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

Description

A TEXT column with a maximum length of 4,294,967,295 or 4GB (2^{32} - 1) characters. The effective maximum length is less if the value contains multi-byte characters. The effective maximum length of `LONGTEXT` columns also depends on the configured maximum packet size in the client/server protocol and available memory. Each `LONGTEXT` value is stored using a four-byte length prefix that indicates the number of bytes in the value.

`JSON` is an alias for `LONGTEXT`. See JSON Data Type for details.

Oracle Mode

In Oracle mode, CLOB is a synonym for `LONGTEXT`.

EXAMPLES

LONGTEXT

Example of `LONGTEXT`:

```
CREATE TABLE longtext_example (
   description VARCHAR(20),
   example LONGTEXT
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

Warning:

The maximum size of a `LONGTEXT` is so large that it cannot be sent to the server without breaking the value up into chunks (something that the command-line client cannot do). For values larger than `16M`, you can increase the `max_allowed_packet` size up to a maximum of `1024M` to increase the allowed size of non-chunked values.

```
INSERT INTO longtext_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 4294967295, 'x'));
```

```
ERROR 1301 (HY000): Result of rpad() was larger than max_allowed_packet (16777216) - truncated
```

Data Too Long

When `SQL_MODE` is strict (the default) a value is considered "too long" when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `LONGTEXT`:

```
TRUNCATE longtext_example;

INSERT INTO longtext_example VALUES
   ('Overflow', RPAD('', 4294967296, 'x'));
```

```
ERROR 1301 (HY000): Result of rpad() was larger than max_allowed_packet (16777216) - truncated
```

See Also

- TEXT
- BLOB and TEXT Data Types
- Data Type Storage Requirements
- JSON Data Type
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

MEDIUMBLOB

Syntax

```
MEDIUMBLOB
```

Description

A BLOB column with a maximum length of 16,777,215 (2^{24} - 1) bytes. Each `MEDIUMBLOB` value is stored using a three-byte length prefix that indicates the number of bytes in the value.

`LONG VARBINARY` is a synonym for `MEDIUMBLOB`.

EXAMPLES

MEDIUMBLOB

Example of `MEDIUMBLOB`:

```
CREATE TABLE mediumblob_example (
   description VARCHAR(20),
   example MEDIUMBLOB
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

```
INSERT INTO mediumblob_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 16777215, CHAR(7)));
```

```
SELECT description, LENGTH(example) AS length
   FROM mediumblob_example;
```

```
+--------------------+----------+
| description        | length   |
+--------------------+----------+
| Normal foo         |        3 |
| Trailing spaces foo |        9 |
| NULLed             |     NULL |
| Empty              |        0 |
| Maximum            | 16777215 |
+--------------------+----------+
```

Data Too Long

When `SQL_MODE` is strict (the default), a value is considered "too long" when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `MEDIUMBLOB`:

```
TRUNCATE mediumblob_example;
```

```
INSERT INTO mediumblob_example VALUES
   ('Overflow', RPAD('', 16777216, CHAR(7)));
```

```
ERROR 1406 (22001): Data too long for column 'example' at row 1
```

See Also

- BLOB
- BLOB and TEXT Data Types
- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

MEDIUMTEXT

Syntax

```
MEDIUMTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

Description

A TEXT column with a maximum length of 16,777,215 (2^{24} - 1) characters. The effective maximum length is less if the value contains multi-byte characters. Each `MEDIUMTEXT` value is stored using a three-byte length prefix that indicates the number of bytes in the value.

SYNONYMS

The following are synonyms for `MEDIUMTEXT`:

- LONG
- LONG CHAR VARYING
- LONG CHARACTER VARYING
- LONG VARCHAR
- LONG VARCHARACTER

EXAMPLES

MEDIUMTEXT

Example of `MEDIUMTEXT`:

```
CREATE TABLE mediumtext_example (
   description VARCHAR(20),
   example MEDIUMTEXT
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

```
INSERT INTO mediumtext_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 16777215, 'x'));
```

```
SELECT description, LENGTH(example) AS length
   FROM mediumtext_example;
```

```
+--------------------+----------+
| description        | length   |
+--------------------+----------+
| Normal foo         |        3 |
| Trailing spaces foo |        9 |
| NULLed             |     NULL |
| Empty              |        0 |
| Maximum            | 16777215 |
+--------------------+----------+
```

Data Too Long

When `SQL_MODE` is strict (the default) a value is considered “too long” when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `MEDIUMTEXT`:

```
TRUNCATE mediumtext_example;

INSERT INTO mediumtext_example VALUES
   ('Overflow', RPAD('', 16777216, 'x'));
```

```
ERROR 1406 (22001): Data too long for column 'example' at row 1
```

See Also

- TEXT
- BLOB and TEXT Data Types
- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

NATIONAL CHAR

Overview

Fixed-length string of specific character set with limit up to 255 bytes.

Examples

```
CREATE TABLE national_char_example (
  example NATIONAL CHAR(32)
);
```

```
SHOW CREATE TABLE national_char_example\G
```

```
*************************** 1. row ***************************
       Table: national_char_example
Create Table: CREATE TABLE `national_char_example` (
  `example` char(32) CHARACTER SET utf8mb3 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

External References

Additional information is available here.

This page is: Copyright © 2026 MariaDB. All rights reserved.

NATIONAL CHAR VARYING

Overview

See NATIONAL VARCHAR.

EXAMPLES

```
CREATE TABLE national_char_varying_example (
  example NATIONAL CHAR VARYING(32)
);
```

```
SHOW CREATE TABLE national_char_varying_example\G
```

```
*************************** 1. row ***************************
       Table: national_char_varying_example
Create Table: CREATE TABLE `national_char_varying_example` (
  `example` varchar(32) CHARACTER SET utf8 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

NATIONAL CHARACTER VARYING

Overview

See NATIONAL VARCHAR.

EXAMPLES

```
CREATE TABLE national_character_varying_example (
  example NATIONAL CHARACTER VARYING(32)
);
```

```
SHOW CREATE TABLE national_character_varying_example\G
```

```
*************************** 1. row ***************************
       Table: national_character_varying_example
Create Table: CREATE TABLE `national_character_varying_example` (
  `example` varchar(32) CHARACTER SET utf8 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

NATIONAL CHARACTER

Overview

See NATIONAL CHAR.

EXAMPLES

```
CREATE TABLE national_character_example (
  example NATIONAL CHARACTER(32)
);
```

```
SHOW CREATE TABLE national_character_example\G
```

```
*************************** 1. row ***************************
       Table: national_character_example
Create Table: CREATE TABLE `national_character_example` (
  `example` char(32) CHARACTER SET utf8 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

NATIONAL VARCHAR

Overview

Variable-length string of specific character set with limit up to 65,535 bytes.

EXAMPLES

```
CREATE TABLE national_varchar_example (
  example NATIONAL VARCHAR(32)
);
```

```
SHOW CREATE TABLE national_varchar_example\G
```

```
*************************** 1. row ***************************
       Table: national_varchar_example
Create Table: CREATE TABLE `national_varchar_example` (
  `example` varchar(32) CHARACTER SET utf8 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

NATIONAL VARCHARACTER

Overview

See NATIONAL VARCHAR.

EXAMPLES

```
CREATE TABLE national_varcharacter_example (
  example NATIONAL VARCHARACTER(32)
);
```

```
SHOW CREATE TABLE national_varcharacter_example\G
```

```
*************************** 1. row ***************************
       Table: national_varcharacter_example
Create Table: CREATE TABLE `national_varcharacter_example` (
  `example` varchar(32) CHARACTER SET utf8 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

NCHAR

Overview

See NATIONAL VARCHAR.

EXAMPLES

```
CREATE TABLE nchar_example (
  example NCHAR(32)
);
```

```
SHOW CREATE TABLE nchar_example\G
```

```
*************************** 1. row ***************************
       Table: nchar_example
Create Table: CREATE TABLE `nchar_example` (
  `example` char(32) CHARACTER SET utf8mb3 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

NCHAR VARCHAR

Overview

See NATIONAL VARCHAR.

EXAMPLES

```
CREATE TABLE nchar_varchar_example (
  example NCHAR VARCHAR(32)
);
```

```
SHOW CREATE TABLE nchar_varchar_example\G
```

```
*************************** 1. row ***************************
       Table: nchar_varchar_example
Create Table: CREATE TABLE `nchar_varchar_example` (
  `example` varchar(32) CHARACTER SET utf8 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

NCHAR VARCHARACTER

Overview

See NATIONAL VARCHAR.

EXAMPLES

```
CREATE TABLE nchar_varcharacter_example (
  example NCHAR VARCHARACTER(32)
);
```

```
SHOW CREATE TABLE nchar_varcharacter_example\G
```

```
*************************** 1. row ***************************
       Table: nchar_varcharacter_example
Create Table: CREATE TABLE `nchar_varcharacter_example` (
  `example` varchar(32) CHARACTER SET utf8 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

NCHAR VARYING

Overview

See NATIONAL VARCHAR.

EXAMPLES

```
CREATE TABLE nchar_varying_example (
  example NCHAR VARYING(32)
);
```

```
SHOW CREATE TABLE nchar_varying_example\G
```

```
*************************** 1. row ***************************
       Table: nchar_varying_example
Create Table: CREATE TABLE `nchar_varying_example` (
  `example` varchar(32) CHARACTER SET utf8 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

RAW

Overview

See VARBINARY.

EXAMPLES

```
SET sql_mode='oracle';
```

```
CREATE TABLE raw_example (
  example RAW(32)
);
```

```
SHOW CREATE TABLE raw_example\G
```

```
*************************** 1. row ***************************
       Table: raw_example
Create Table: CREATE TABLE "raw_example" (
  "example" varbinary(32) DEFAULT NULL
)
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

ROW

Syntax

```
ROW (<field name> <data type> [{, <field name> <data type>}... ])
```

Description

`ROW` is a data type for stored procedure variables.

Features

ROW fields as normal variables

`ROW` fields (members) act as normal variables, and are able to appear in all query parts where a stored procedure variable is allowed:

- Assignment is using the `:=` operator and the SET command:

```
a.x:= 10;
a.x:= b.x;
SET a.x= 10, a.y=20, a.z= b.z;
```

- Passing to functions and operators:

```
SELECT f1(rec.a), rec.a<10;
```

- Clauses (select list, WHERE, HAVING, LIMIT, etc…,):

```
SELECT var.a, t1.b FROM t1 WHERE t1.b=var.b LIMIT var.c;
```

- `INSERT` values:

```
INSERT INTO t1 VALUES (rec.a, rec.b, rec.c);
```

- `SELECT .. INTO` targets:

```
SELECT a,b INTO rec.a, rec.b FROM t1 WHERE t1.id=10;
```

- Dynamic SQL out parameters (EXECUTE and EXECUTE IMMEDIATE)

```
EXECUTE IMMEDIATE 'CALL proc_with_out_param(?)' USING rec.a;
```

ROW type variables as FETCH targets

`ROW` type variables are allowed as FETCH targets:

```
FETCH cur INTO rec;
```

where `cur` is a `CURSOR` and `rec` is a `ROW` type stored procedure variable.

Note, currently an attempt to use `FETCH` for a `ROW` type variable returns this error:

```
ERROR 1328 (HY000): Incorrect number of FETCH variables
```

`FETCH` from a cursor `cur` into a `ROW` variable `rec` works as follows:

- The number of fields in `cur` must match the number of fields in `rec`. Otherwise, an error is reported.
- Assignment is done from left to right. The first cursor field is assigned to the first variable field, the second cursor field is assigned to the second variable field, etc.
- Field names in `rec` are not important and can differ from field names in `cur`.

See FETCH Examples (below) for examples of using this with `sql_mode=ORACLE` and `sql_mode=DEFAULT`.

ROW type variables as `SELECT...INTO` targets

`ROW` type variables are allowed as `SELECT..INTO` targets with some differences depending on which `sql_mode` is in use.

- When using `sql_mode=ORACLE`, `table%ROWTYPE` and `cursor%ROWTYPE` variables can be used as `SELECT...INTO` targets.
- Using multiple `ROW` variables in the `SELECT..INTO` list will report an error.
- Using `ROW` variables with a different column count than in the `SELECT..INTO` list will report an error.

See SELECT…INTO Examples (below) for examples of using this with `sql_mode=ORACLE` and `sql_mode=DEFAULT`.

Features not implemented

The following features are planned, but not implemented yet:

- Returning a `ROW` type expression from a stored function (see MDEV-12252). This will need some grammar change to support field names after parentheses:

```
SELECT f1().x FROM DUAL;
```

- Returning a `ROW` type expression from a built-in hybrid type function, such as `CASE`, `IF`, etc.
- `ROW` of `ROW` values

Examples

Declaring a ROW in a stored procedure

```
DELIMITER $$
CREATE PROCEDURE p1()
BEGIN
  DECLARE r ROW (c1 INT, c2 VARCHAR(10));
  SET r.c1= 10;
  SET r.c2= 'test';
  INSERT INTO t1 VALUES (r.c1, r.c2);
END;
$$
DELIMITER ;
CALL p1();
```

FETCH Examples

A complete `FETCH` example for `sql_mode=ORACLE`:

```
DROP TABLE IF EXISTS t1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
INSERT INTO t1 VALUES (20,'b20');
INSERT INTO t1 VALUES (30,'b30');

SET sql_mode=oracle;
DROP PROCEDURE IF EXISTS p1;
DELIMITER $$
CREATE PROCEDURE p1 AS
  rec ROW(a INT, b VARCHAR(32));
  CURSOR c IS SELECT a,b FROM t1;
BEGIN
  OPEN c;
  LOOP
    FETCH c INTO rec;
    EXIT WHEN c%NOTFOUND;
    SELECT ('rec=(' || rec.a ||','|| rec.b||')');
  END LOOP;
  CLOSE c;
END;
$$
DELIMITER ;
CALL p1();
```

A complete `FETCH` example for `sql_mode=DEFAULT`:

```
DROP TABLE IF EXISTS t1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
INSERT INTO t1 VALUES (20,'b20');
INSERT INTO t1 VALUES (30,'b30');

SET sql_mode=DEFAULT;
DROP PROCEDURE IF EXISTS p1;
DELIMITER $$
CREATE PROCEDURE p1()
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE rec ROW(a INT, b VARCHAR(32));
  DECLARE c CURSOR FOR SELECT a,b FROM t1;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
  OPEN c;
read_loop:
  LOOP
    FETCH c INTO rec;
    IF done THEN
      LEAVE read_loop;
    END IF;
    SELECT CONCAT('rec=(',rec.a,',',rec.b,')');
  END LOOP;
  CLOSE c;
END;
$$
DELIMITER ;
CALL p1();
```

SELECT…INTO Examples

A `SELECT...INTO` example for `sql_mode=DEFAULT`:

```
SET sql_mode=DEFAULT;
DROP TABLE IF EXISTS t1;
DROP PROCEDURE IF EXISTS p1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
DELIMITER $$
CREATE PROCEDURE p1()
BEGIN
  DECLARE rec1 ROW(a INT, b VARCHAR(32));
  SELECT * FROM t1 INTO rec1;
  SELECT rec1.a, rec1.b;
END;
$$
DELIMITER ;
CALL p1();
```

The above example returns:

```
+--------+--------+
| rec1.a | rec1.b |
+--------+--------+
|     10 | b10    |
+--------+--------+
```

A `SELECT...INTO` example for `sql_mode=ORACLE`:

```
SET sql_mode=ORACLE;
DROP TABLE IF EXISTS t1;
DROP PROCEDURE IF EXISTS p1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
DELIMITER $$
CREATE PROCEDURE p1 AS
  rec1 ROW(a INT, b VARCHAR(32));
BEGIN
  SELECT * FROM t1 INTO rec1;
  SELECT rec1.a, rec1.b;
END;
$$
DELIMITER ;
CALL p1();
```

The above example returns:

```
+--------+--------+
| rec1.a | rec1.b |
+--------+--------+
|     10 | b10    |
+--------+--------+
```

An example for `sql_mode=ORACLE` using `table%ROWTYPE` variables as `SELECT..INTO` targets:

```
SET sql_mode=ORACLE;
DROP TABLE IF EXISTS t1;
DROP PROCEDURE IF EXISTS p1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
DELIMITER $$
CREATE PROCEDURE p1 AS
  rec1 t1%ROWTYPE;
BEGIN
  SELECT * FROM t1 INTO rec1;
  SELECT rec1.a, rec1.b;
END;
$$
DELIMITER ;
CALL p1();
```

The above example returns:

```
+--------+--------+
| rec1.a | rec1.b |
+--------+--------+
|     10 | b10    |
+--------+--------+
```

An example for `sql_mode=ORACLE` using `cursor%ROWTYPE` variables as `SELECT..INTO` targets:

```
SET sql_mode=ORACLE;
DROP TABLE IF EXISTS t1;
DROP PROCEDURE IF EXISTS p1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
DELIMITER $$
CREATE PROCEDURE p1 AS
  CURSOR cur1 IS SELECT * FROM t1;
  rec1 cur1%ROWTYPE;
BEGIN
  SELECT * FROM t1 INTO rec1;
  SELECT rec1.a, rec1.b;
END;
$$
DELIMITER ;
CALL p1();
```

The above example returns:

```
+--------+--------+
| rec1.a | rec1.b |
+--------+--------+
|     10 | b10    |
+--------+--------+
```

Row Table example:

```
CREATE TABLE row_table(
  descr VARCHAR(20),
  val INT
);
```

```
INSERT INTO row_table VALUES ('Life', 42);
```

```
DELIMITER $$
CREATE PROCEDURE row_proc()
BEGIN
  DECLARE rec1 ROW(descr VARCHAR(20), val INT);
  SELECT * INTO rec1 FROM row_table;
  SELECT rec1.descr, rec1.val;
END;
$$
DELIMITER ;
```

```
CALL row_proc();

+------------+----------+
| rec1.descr | rec1.val |
+------------+----------+
| Life       |       42 |
+------------+----------+
```

See Also

- STORED PROCEDURES
- DECLARE Variable

This page is licensed: CC BY-SA / Gnu FDL

SET Data Type

Syntax

```
SET('value1','value2',...) [CHARACTER SET charset_name] [COLLATE collation_name]
```

Description

A set. A string object that can have zero or more values, each of which must be chosen from the list of values 'value1', 'value2', … A `SET` column can have a maximum of 64 members. `SET` values are represented internally as integers.

`SET` values cannot contain commas.

If a `SET` contains duplicate values, an error will be returned if strict mode is enabled, or a warning if strict mode is not enabled.

EXAMPLES

```
CREATE TABLE set_example (
  description VARCHAR(20),
  example SET('Foo', 'Bar', 'Baz', 'Bob')
);
```

```
INSERT INTO set_example VALUES
  ('1 val', 'Foo'),
  ('2 vals', 'Baz,Foo'),
  ('4 vals', 'Bob,Foo,Bar,Foo,Baz,Bob');
```

```
SELECT * FROM set_example;

+-------------+-----------------+
| description | example         |
+-------------+-----------------+
| 1 val       | Foo             |
| 2 vals      | Foo,Baz         |
| 4 vals      | Foo,Bar,Baz,Bob |
+-------------+-----------------+
```

See Also

- Character Sets and Collations
- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

TEXT

Syntax

```
TEXT[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]
```

Description

> **Note:**
>
> If you are handling large text data that exceeds the `max_allowed_packet` limit, you can stream the data in chunks using specialized API functions like `mysql_stmt_send_long_data()` or `setCharacterStream()`. See Handling Large Data via APIs for more details.

A `TEXT` column with a maximum length of 65,535 (2^{16} - 1) characters. The effective maximum length is less if the value contains multi-byte characters. Each `TEXT` value is stored using a two-byte length prefix that indicates the number of bytes in the value. If you need a bigger storage, consider using MEDIUMTEXT instead.

An optional length `M` can be given for this type. If this is done, MariaDB creates the column as the smallest `TEXT` type large enough to hold values `M` characters long.

`BLOB` and `TEXT` columns can be assigned a DEFAULT value.

Examples

Trailing spaces:

```
CREATE TABLE strtest (d TEXT(10));
INSERT INTO strtest VALUES('Maria   ');

SELECT d='Maria',d='Maria   ' FROM strtest;
+-----------+--------------+
| d='Maria' | d='Maria   ' |
+-----------+--------------+
|         1 |            1 |
+-----------+--------------+

SELECT d LIKE 'Maria',d LIKE 'Maria   ' FROM strtest;
+---------------+-------------------+
| d LIKE 'Maria' | d LIKE 'Maria   ' |
+---------------+-------------------+
|              0 |                 1 |
+---------------+-------------------+
```

Example of TEXT

```
CREATE TABLE text_example (
   description VARCHAR(20),
   example TEXT
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

```
INSERT INTO text_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 65535, 'x'));
```

```
SELECT description, LENGTH(example) AS length
   FROM text_example;
```

```
+---------------------+--------+
| description         | length |
+---------------------+--------+
| Normal foo          |      3 |
| Trailing spaces foo |      9 |
| NULLed              |   NULL |
| Empty               |      0 |
| Maximum             |  65535 |
+---------------------+--------+
```

Data too Long

When `SQL_MODE` is strict (the default) a value is considered "too long" when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `TEXT`:

```
TRUNCATE text_example;

INSERT INTO text_example VALUES
   ('Overflow', RPAD('', 65536, 'x'));
```

```
ERROR 1406 (22001): Data too long for column 'example' at row 1
```

Indexing

A unique index can be created on a `TEXT` column.

Internally, this uses hash indexing to quickly check the values and if a hash collision is found, the actual stored values are compared in order to retain the uniqueness.

Difference between VARCHAR and TEXT

- VARCHAR columns can be fully indexed. `TEXT` columns can only be indexed over a specified length.
- Using `TEXT` or BLOB in a SELECT query that uses temporary tables for storing intermediate results will force the temporary table to be disk based (using the Aria storage engine instead of the memory storage engine, which is a bit slower. This is not that bad as the Aria storage engine caches the rows in memory. To get the benefit of this, one should ensure that the aria_pagecache_buffer_size variable is big enough to hold most of the row and index data for temporary tables.

For Storage Engine Developers

- Internally the full length of the VARCHAR column is allocated inside each TABLE objects record[] structure. As there are three such buffers, each open table will allocate 3 times max-length-to-store-varchar bytes of memory.
- `TEXT` and `BLOB` columns are stored with a pointer (4 or 8 bytes) + a 1-4 bytes length. The `TEXT` data is only stored once. This means that internally `TEXT` uses less memory for each open table but instead has the additional overhead that each `TEXT` object needs to be allocated and freed for each row access (with some caching in between).

See Also

- BLOB and TEXT Data Types
- MEDIUMTEXT
- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

TINYBLOB

Syntax

```
TINYBLOB
```

Description

A BLOB column with a maximum length of 255 (2^8 - 1) bytes. Each `TINYBLOB` value is stored using a one-byte length prefix that indicates the number of bytes in the value.

EXAMPLES

TINYBLOB

Example of TINYBLOB:

```
CREATE TABLE tinyblob_example (
   description VARCHAR(20),
   example TINYBLOB
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

```
INSERT INTO tinyblob_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 255, CHAR(7)));
```

```
SELECT description, LENGTH(example) AS length
   FROM tinyblob_example;
```

```
+---------------------+--------+
| description         | length |
+---------------------+--------+
| Normal foo          |      3 |
| Trailing spaces foo |      9 |
| NULLed              |   NULL |
| Empty               |      0 |
| Maximum             |    255 |
+---------------------+--------+
```

Data too Long

When `SQL_MODE` is strict (the default), a value is considered “too long” when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `TINYBLOB`:

```
TRUNCATE tinyblob_example;

INSERT INTO tinyblob_example VALUES
   ('Overflow', RPAD('', 256, CHAR(7)));
```

```
ERROR 1406 (22001): Data too long for column 'example' at row 1
```

See Also

- BLOB

- BLOB and TEXT Data Types
- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

TINYTEXT

Syntax

```
TINYTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

Description

A TEXT column with a maximum length of 255 (2^8 - 1) characters. The effective maximum length is less if the value contains multi-byte characters. Each `TINYTEXT` value is stored using a one-byte length prefix that indicates the number of bytes in the value.

EXAMPLES

TINYTEXT

Example of `TINYTEXT`:

```
CREATE TABLE tinytext_example (
   description VARCHAR(20),
   example TINYTEXT
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

```
INSERT INTO tinytext_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 255, 'x'));
```

```
SELECT description, LENGTH(example) AS length FROM tinytext_example;
```

```
+--------------------+--------+
| description        | length |
+--------------------+--------+
| Normal foo         |      3 |
| Trailing spaces foo |      9 |
| NULLed             |   NULL |
| Empty              |      0 |
| Maximum            |    255 |
+--------------------+--------+
```

Data too Long

When `SQL_MODE` is strict (the default) a value is considered “too long” when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `TINYTEXT`:

```
TRUNCATE tinytext_example;

INSERT INTO tinytext_example VALUES
   ('Overflow', RPAD('', 256, 'x'));
```

```
ERROR 1406 (22001): Data too long for column 'example' at row 1
```

See Also

- TEXT
- BLOB and TEXT Data Types
- Data Type Storage Requirements

This page is licensed: GPLv2, originally from fill_help_tables.sql

UUID Data Type

> **Note:**
>
> `UUID` is available from MariaDB 10.7.

Syntax

```
UUID
```

Description

The `UUID` data type is intended for the storage of 128-bit UUID (Universally Unique Identifier) data. See the UUID function page for more details on UUIDs themselves.

Retrieval

Data retrieved by this data type is in the string representation defined in RFC4122.

Casting

String literals of hexadecimal characters and CHAR/VARCHAR/TEXT can be cast to the `UUID` data type. Likewise hexadecimal literals, binary-literals, and BINARY/VARBINARY/BLOB types can also be cast to `UUID`.

The data type will not accept a short `UUID` generated with the UUID_SHORT function, but will accept a value without the `-` character generated by the SYS_GUID function (or inserted directly). Hyphens can be partially omitted as well, or included after any group of two digits.

The type does not accept `UUID` values in braces, permitted by some implementations.

Storage

`UUID` values are stored in an index-friendly manner; the order of a UUID of llllllll-mmmm-Vhhh-vsss-nnnnnnnnnnnn is stored as:

```
nnnnnnnnnnnn-vsss-Vhhh-mmmm-llllllll
```

This provides a sorting order, assuming a UUIDv1 (node and timestamp) is used, of the node, followed by the timestamp. The key aspect is the timestamps are sequential.

MariaDB starting with 10.10

> **CURRENT**
>
> Taking into account that UUIDv7 and other versions are designed around time ordering, `UUID` values version >= 6 are stored without byte-swapping, and `UUID` values with version >=8 and variant=0 are now considered invalid (as the SQL standard suggests).

> **< 10.11.5 / 10.10.6**
>
> `UUID` values version >= 6 are **not** stored without byte-swapping, and `UUID` values with version >=8 and variant=0 are **not** considered invalid.

Examples

```
CREATE TABLE t1 (id UUID);
```

Directly inserting via string literals:

```
INSERT INTO t1 VALUES('123e4567-e89b-12d3-a456-426655440000');
```

Directly Inserting via hexadecimal literals:

```
INSERT INTO t1 VALUES (x'fffffffffffffffffffffffffffffffe');
```

Generating and inserting via the UUID function.

```
INSERT INTO t1 VALUES (UUID());
```

Retrieval:

```
SELECT * FROM t1;
+--------------------------------------+
| id                                   |
+--------------------------------------+
| 123e4567-e89b-12d3-a456-426655440000 |
| ffffffff-ffff-ffff-ffff-fffffffffffe |
| 93aac041-1a14-11ec-ab4e-f859713e4be4 |
+--------------------------------------+
```

The UUID_SHORT function does not generate valid full-length `UUID`:

```
INSERT INTO t1 VALUES (UUID_SHORT());
ERROR 1292 (22007): Incorrect uuid value: '99440417627439104'
  for column `test`.`t1`.`id` at row 1
```

Accepting a value without the `-` character, either directly or generated by the SYS_GUID function:

```
INSERT INTO t1 VALUES (SYS_GUID());

SELECT * FROM t1;
+--------------------------------------+
| id                                   |
+--------------------------------------+
| 123e4567-e89b-12d3-a456-426655440000 |
| ffffffff-ffff-ffff-ffff-fffffffffffe |
| 93aac041-1a14-11ec-ab4e-f859713e4be4 |
| ea0368d3-1a14-11ec-ab4e-f859713e4be4 |
+--------------------------------------+

SELECT SYS_GUID();
+----------------------------------+
| SYS_GUID()                       |
+----------------------------------+
| ff5b6bcc1a1411ecab4ef859713e4be4 |
+----------------------------------+

INSERT INTO t1 VALUES ('ff5b6bcc1a1411ecab4ef859713e4be4');

SELECT * FROM t1;
+--------------------------------------+
| id                                   |
+--------------------------------------+
| 123e4567-e89b-12d3-a456-426655440000 |
| ffffffff-ffff-ffff-ffff-fffffffffffe |
| 93aac041-1a14-11ec-ab4e-f859713e4be4 |
| ea0368d3-1a14-11ec-ab4e-f859713e4be4 |
| ff5b6bcc-1a14-11ec-ab4e-f859713e4be4 |
+--------------------------------------+
```

Valid and invalid hyphen and brace usage:

```
TRUNCATE t1;

INSERT INTO t1 VALUES ('f8aa-ed66-1a1b-11ec-ab4e-f859-713e-4be4');

INSERT INTO t1 VALUES ('1b80667f1a1c-11ecab4ef859713e4be4');

INSERT INTO t1 VALUES ('2fd6c945-1a-1c-11ec-ab4e-f859713e4be4');

INSERT INTO t1 VALUES ('49-c9-f9-59-1a-1c-11ec-ab4e-f859713e4be4');

INSERT INTO t1 VALUES ('57-96-da-c1-1a-1c-11-ec-ab-4e-f8-59-71-3e-4b-e4');

INSERT INTO t1 VALUES ('6-eb74f8f-1a1c-11ec-ab4e-f859713e4be4');

INSERT INTO t1 VALUES ('{29bad136-1a1d-11ec-ab4e-f859713e4be4}');
ERROR 1292 (22007): Incorrect uuid value: '{29bad136-1a1d-11ec-ab4e-f859713e4be4}'
  for column `test`.`t1`.`id` at row 1

SELECT * FROM t1;
+--------------------------------------+
| id                                   |
+--------------------------------------+
| f8aaed66-1a1b-11ec-ab4e-f859713e4be4 |
| 1b80667f-1a1c-11ec-ab4e-f859713e4be4 |
| 2fd6c945-1a1c-11ec-ab4e-f859713e4be4 |
| 49c9f959-1a1c-11ec-ab4e-f859713e4be4 |
| 5796dac1-1a1c-11ec-ab4e-f859713e4be4 |
| 6eb74f8f-1a1c-11ec-ab4e-f859713e4be4 |
+--------------------------------------+
```

See Also

- 10.7 preview feature: UUID Data Type (mariadb.org blog post)

- UUID function for generating v1 UUIDs
- UUID function for generating v4 UUIDs
- UUID function for generating v7 UUIDs
- UUID_SHORT function
- SYS_GUID - UUID without the `-` character for Oracle compatibility

This page is licensed: CC BY-SA / Gnu FDL

VARBINARY

Syntax

```
VARBINARY(M)
```

Description

The `VARBINARY` type is similar to the VARCHAR type, but stores binary byte strings rather than non-binary character strings. `M` represents the maximum column length in bytes.

It contains no character set, and comparison and sorting are based on the numeric value of the bytes.

If the maximum length is exceeded, and SQL strict mode is not enabled , the extra characters will be dropped with a warning. If strict mode is enabled, an error will occur.

Unlike BINARY values, `VARBINARY` values are not right-padded when inserting.

Use Cases for Zero Length

A `BINARY(0)` or `VARBINARY(0)` column is restricted to an empty byte string or `NULL`.

- **Schema Preservation**: Use these columns when a system expects a specific column to exist, but no data storage is required for your current application.
- **Space-Efficient Indicators**: These columns can act as a two-state indicator where the presence of an empty byte string represents one state and `NULL` represents another.

If you attempt to insert a value other than an empty string, MariaDB returns an `ERROR 1406 (22001)` indicating the data is too long for the column.

Oracle Mode

In Oracle mode, `RAW` is a synonym for `VARBINARY`.

Examples

Inserting too many characters, first with strict mode off, then with it on:

```
CREATE TABLE varbins (a VARBINARY(10));

INSERT INTO varbins VALUES('12345678901');
Query OK, 1 row affected, 1 warning (0.04 sec)

SELECT * FROM varbins;
+------------+
| a          |
+------------+
| 1234567890 |
+------------+

SET sql_mode='STRICT_ALL_TABLES';

INSERT INTO varbins VALUES('12345678901');
ERROR 1406 (22001): Data too long for column 'a' at row 1
```

Sorting is performed with the byte value:

```
TRUNCATE varbins;

INSERT INTO varbins VALUES('A'),('B'),('a'),('b');

SELECT * FROM varbins ORDER BY a;
+------+
| a    |
+------+
| A    |
| B    |
| a    |
| b    |
+------+
```

Using CAST to sort as a CHAR instead:

```
SELECT * FROM varbins ORDER BY CAST(a AS CHAR);
+------+
| a    |
+------+
| a    |
| A    |
| b    |
| B    |
+------+
```

VARBINARY

For our example of `VARBINARY`, we picked a maximum size that avoids overflowing the maximum row size (65535). Keep in mind that a multi-byte character set would need more space in the row than a single-byte character set. We also avoid the auto-conversion of a `VARBINARY` into a `BLOB`, `MEDIUMBLOB`, or `LONGBLOB` that can happen when `STRICT_TRANS_TABLES` is not set in the `SQL_MODE`.

The example:

```
CREATE TABLE varbinary_example (
   description VARCHAR(20),
   example VARBINARY(65511)
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

```
INSERT INTO varbinary_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 65511, CHAR(7)));
```

```
SELECT description, LENGTH(example) AS length
   FROM varbinary_example;
```

```
+--------------------+--------+
| description        | length |
+--------------------+--------+
| Normal foo         |      3 |
| Trailing spaces foo |      9 |
| NULLed             |   NULL |
| Empty              |      0 |
| Maximum            |  65511 |
+--------------------+--------+
```

Data too Long

When `SQL_MODE` is strict (the default), a value is considered “too long” when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `VARBINARY`:

```
TRUNCATE varbinary_example;

INSERT INTO varbinary_example VALUES
   ('Overflow', RPAD('', 65512, CHAR(7)));
```

```
ERROR 1406 (22001): Data too long for column 'example' at row 1
```

See Also

- VARCHAR
- Data Type Storage Requirements
- Oracle mode

This page is licensed: GPLv2, originally from fill_help_tables.sql

VARCHAR

Syntax

```
[NATIONAL] VARCHAR(M) [CHARACTER SET charset_name] [COLLATE collation_name]
```

Description

A variable-length string. M represents the maximum column length in characters. The range of M is 0 to 65,532. The effective maximum length of a `VARCHAR` is subject to the maximum row size and the character set used. For example, utf-8 characters can require up to three bytes per character, so a `VARCHAR` column that uses the utf-8 character set can be declared to be a maximum of 21,844 characters.

`VARCHAR` is shorthand for `CHARACTER VARYING`. `NATIONAL VARCHAR` is the standard SQL way to define that a `VARCHAR` column should use some predefined character set. MariaDB uses utf-8 as its predefined character set, as does MySQL. `NVARCHAR` is shorthand for `NATIONAL VARCHAR`.

MariaDB stores `VARCHAR` values as a one-byte or two-byte length prefix plus data. The length prefix indicates the number of bytes in the value. A `VARCHAR` column uses one length byte if values require no more than 255 bytes, two length bytes if values may require more than 255 bytes.

MariaDB follows the standard SQL specification and does not remove trailing spaces from `VARCHAR` values.

If a unique index consists of a column where trailing pad characters are stripped or ignored, inserts into that column where values differ only by the number of trailing pad characters will result in a duplicate-key error.

For the ColumnStore engine, M represents the maximum column length in bytes.

For MariaDB, a number of NO PAD collations are available.

`VARCHAR(0)` columns can contain 2 values: an empty string or `NULL`. Such columns cannot be part of an index. The CONNECT storage engine does not support `VARCHAR(0)`.

Use Cases for Zero Length

A `CHAR(0)` or `VARCHAR(0)` column occupies minimal space and is restricted to two possible values: an empty string (`''`) or `NULL`. You can use these columns for the following purposes:

- **Legacy Compatibility**: Include these columns to maintain compatibility with older applications that require a specific table schema, even if the data is no longer collected.
- **Two-State Flags**: A `CHAR(0) NULL` column can function as a boolean indicator. It uses only one bit of storage to distinguish between a “set” state (the empty string) and an “unset” state (`NULL`).
- **Row Marking**: You can use a `CHAR(0)` column to mark a specific row in a table. For example, if you require only one “active” row, set that row to an empty string while keeping all other rows `NULL`.

The following error occurs if you attempt to insert any character data into a 0-length column: `ERROR 1406 (22001): Data too long for column`.

SYNONYMS

The following are synonyms for `VARCHAR`:

- CHAR VARYING
- CHARACTER VARYING
- VARCHAR2
- VARCHARACTER

Examples

The following are equivalent:

```
VARCHAR(30) CHARACTER SET utf8
NATIONAL VARCHAR(30)
NVARCHAR(30)
NCHAR VARCHAR(30)
NATIONAL CHARACTER VARYING(30)
NATIONAL CHAR VARYING(30)
```

Trailing spaces

```
CREATE TABLE strtest (v VARCHAR(10));
INSERT INTO strtest VALUES('Maria   ');

SELECT v='Maria',v='Maria   ' FROM strtest;
+-----------+--------------+
| v='Maria' | v='Maria   ' |
+-----------+--------------+
|         1 |            1 |
+-----------+--------------+

SELECT v LIKE 'Maria',v LIKE 'Maria   ' FROM strtest;
+----------------+-------------------+
| v LIKE 'Maria' | v LIKE 'Maria   ' |
+----------------+-------------------+
|              0 |                 1 |
+----------------+-------------------+
```

VARCHAR

For our example of `VARCHAR`, we picked a maximum size that avoids overflowing the maximum row size (65535). Keep in mind that a multi-byte character set would need more space in the row than a single-byte character set. We also avoid the auto-conversion of a `VARCHAR` into a `TEXT`, `MEDIUMTEXT`, or `LONGTEXT` that can happen when `STRICT_TRANS_TABLES` is not set in the `SQL_MODE`.

The example:

```
CREATE TABLE varchar_example (
   description VARCHAR(20),
   example VARCHAR(65511)
) DEFAULT CHARSET=latin1; -- One byte per char makes the examples clearer
```

```
INSERT INTO varchar_example VALUES
   ('Normal foo', 'foo'),
   ('Trailing spaces foo', 'foo      '),
   ('NULLed', NULL),
   ('Empty', ''),
   ('Maximum', RPAD('', 65511, 'x'));
```

```
SELECT description, LENGTH(example) AS length
   FROM varchar_example;
```

```
+---------------------+--------+
| description         | length |
+---------------------+--------+
| Normal foo          |      3 |
| Trailing spaces foo |      9 |
| NULLed              |   NULL |
| Empty               |      0 |
| Maximum             |  65511 |
+---------------------+--------+
```

Data too Long

When `SQL_MODE` is strict (the default), a value is considered “too long” when its length exceeds the size of the data type, and an error is generated.

Example of data too long behavior for `VARCHAR`:

```
TRUNCATE varchar_example;

INSERT INTO varchar_example VALUES
   ('Overflow', RPAD('', 65512, 'x'));
```

```
ERROR 1406 (22001): Data too long for column 'example' at row 1
```

Storage on Disk

`VARCHAR` is a standard variable-length data type. `VARCHAR` values are stored **in-table (in-row)**, where the actual string data is part of the table record.

Each value is stored as a one-byte or two-byte length prefix followed by the data. The length prefix specifies the number of bytes contained in the value.

This differs from `TEXT` and `BLOB` data types, which are typically stored **off-row**, with only a pointer stored in the table record.

> **Note:** References to multiple buffers for `VARCHAR` specify internal memory usage for table operations. This does not indicate that `VARCHAR` values are stored in memory rather than on disk.

The exact storage behavior may vary depending on the storage engine. For the specific byte-calculation formulas and limits, see Data Type Storage Requirements.

Truncation

- Depending on whether or not strict sql mode is set, you will either get a warning or an error if you try to insert a string that is too long into a `VARCHAR` column. If the extra characters are spaces, the spaces that can't fit will be removed, and you will always get a warning, regardless of the sql mode setting.

Difference Between VARCHAR and TEXT

- **Indexing differences**: `VARCHAR` columns can be fully indexed. TEXT columns can only be indexed over a specified length.
- **Temporary table behavior**: Using TEXT or BLOB in a SELECT query that uses temporary tables for storing intermediate results will force the temporary table to be disk-based (using the Aria storage engine instead of the memory storage engine, which is a bit slower. This is not that bad, as the Aria storage engine caches the rows in memory. To get the benefit of this, one should ensure that the aria_pagecache_buffer_size variable is big enough to hold most of the row and index data for temporary tables.
- **Storage characteristics**: Although both can overflow, `TEXT` is primarily designed for off-page storage, beginning with a pointer in the main row. In contrast, `VARCHAR` is optimized to remain in-page and only overflows when row size limits necessitate it.

Oracle Mode

In Oracle mode, `VARCHAR2` is a synonym.

For Storage Engine Developers

- Internally, the full length of the `VARCHAR` column is allocated inside each `TABLE` objects record[] structure. As there are three such buffers, each open table will allocate 3 times the max-length-to-store-varchar bytes of memory.
- TEXT and BLOB columns are stored with a pointer (4 or 8 bytes) + a 1-4 byte length. The TEXT data is only stored once. This means that internally `TEXT` uses less memory for each open table but instead has the additional overhead that each `TEXT` object needs to be allocated and freed for each row access (with some caching in between).

See Also

- VARBINARY
- TEXT
- CHAR
- Character Sets and Collations
- Data Type Storage Requirements
- Oracle mode from MariaDB 10.3

This page is licensed: GPLv2, originally from fill_help_tables.sql

VARCHAR2

Overview

This is a synonym for VARCHAR.

EXAMPLES

```
SET sql_mode='oracle';

CREATE TABLE varchar2_example (
  example VARCHAR2(32)
);
```

```
SHOW CREATE TABLE varchar2_example\G
```

```
*************************** 1. row ***************************
       Table: varchar2_example
Create Table: CREATE TABLE "varchar2_example" (
  "example" varchar(32) DEFAULT NULL
)
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

VARCHARACTER

Overview

See VARCHAR.

EXAMPLES

```
CREATE TABLE varcharacter_example (
  example VARCHARACTER(32)
);
```

```
SHOW CREATE TABLE varcharacter_example\G

*************************** 1. row ***************************
       Table: varcharacter_example
Create Table: CREATE TABLE `varcharacter_example` (
  `example` varchar(32) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

XMLTYPE

Introduced

MariaDB 12.3

Syntax

```
XMLTYPE
```

- No parameters accepted
- Length specification for `XMLTYPE` (e.g., `XMLTYPE(100)`) results in an error

Description

`XMLTYPE` is a data type introduced in MariaDB 12.3 for storing XML data. It is designed to:

- Provide convenient storage of XML data
- Ensure compatibility with Oracle databases
- Support future XML validation and processing capabilities

In its initial implementation, `XMLTYPE` provides basic XML storage capabilities only, without validation or specialized XML-specific functionality. When using string functions, the data type is effectively converted to strings and is maintained in temporary tables.

Characteristics

- Maximum storage capacity: 4GB (same as `LONGBLOB`)
- Compatibility: Designed to be compatible with Oracle's `XMLTYPE`
- Validation: XML validation or schema enforcement is not supported
- Length restriction: Length cannot be specified.

Example (invalid usage):

```
CREATE TABLE t1 (a XMLTYPE(6));
```

`XMLTYPE` does not accept length parameters, unlike data types such as `VARCHAR(255)` or `DECIMAL(10,2)`.

Related Functions

With MariaDB 12.3, the following functions return the `XMLTYPE` data type:

- `UPDATEXML`
- `CAST`

Examples

Basic Tables Creation

```
CREATE TABLE t1(id INT, x xmltype);
SHOW CREATE TABLE t1;
```

Output

```
Table   Create Table
t1  CREATE TABLE `t1` (
  `id` int(11) DEFAULT NULL,
  `x` xmltype DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci
```

Character Set Specification

```
CREATE TABLE t1(id INT, x xmltype  CHARACTER SET utf8mb3) CHARACTER SET utf8mb4;
SHOW CREATE TABLE t1;
```

Output

```
Table   Create Table
t1  CREATE TABLE `t1` (
  `id` int(11) DEFAULT NULL,
  `x` xmltype CHARACTER SET utf8mb3 COLLATE utf8mb3_uca1400_ai_ci DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci
```

Binary Attribute

```
CREATE TABLE t1 (a xmltype binary) CHARACTER SET utf8mb4;
SHOW CREATE TABLE t1;
```

Output

```
Table   Create Table
t1  CREATE TABLE `t1` (
  `a` xmltype CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci
```

Basic Data Insertion and Selection

```
CREATE TABLE t1(id INT, x xmltype);
INSERT INTO t1 VALUES (1, 'one'), (2, 'two'), (3, 'three');
SELECT * FROM t1;
```

Output

```
id  x
1    one
2    two
3    three
```

Restrictions and Error Cases

Length Specification Not Permitted

If a length parameter is provided for `XMLTYPE`, the system returns an error:

```
CREATE TABLE t1 (a XMLTYPE(6));
```

Output

```
ERROR HY000: Data type 'XMLTYPE' doesn't support LENGTH attribute.
```

REF_SYSTEM_ID Attribute Not Supported

```
CREATE TABLE t1 (a XMLTYPE REF_SYSTEM_ID=4);
```

Output

```
ERROR HY000: Data type 'XMLTYPE' doesn't support REF_SYSTEM_ID attribute.
```

Length Parameters in Complex Column Definitions

```
CREATE TABLE t1(id INT, x xmltype(10, 2));
```

Output

```
ERROR HY000: Data type 'xmltype' doesn't support LENGTH attribute.
```

See Also

- CAST
- Data Types
- String Functions
- LONGBLOB
- Data Type Storage Requirements

This page is licensed: CC BY-SA / Gnu FDL

Character Sets and Collations

See Character Set and Collation Overview.

Learn the core concepts of character sets and collations in MariaDB, including how they define string storage and sorting rules.

See Setting Character Sets and Collations.

Complete Setting Character Sets and Collations data type guide for MariaDB. Complete reference for syntax, valid values, storage requirements, and range.

See Supported Character Sets and Collations.

Complete character sets reference: SHOW CHARACTER SET command, information_schema.CHARACTER_SETS/COLLATIONS tables, NO PAD queries, and ci/cs flags.

See Unicode.

Explore MariaDB's support for Unicode, covering the differences between the utf8mb3 and utf8mb4 character sets for multi-byte storage.

See Internationalization and Localization.

Discover how MariaDB supports internationalization and localization, enabling databases to store and process data in multiple languages.

This page is licensed: CC BY-SA / Gnu FDL

Character Set and Collation Overview

What are Character Sets and Collations?

A character set is a set of characters, while a collation comprises the rules for comparing and sorting a particular character set.

For example, a subset of a character set could consist of the letters `A`, `B` and `C`. A default collation could define these as appearing in an ascending order of `A, B, C`.

If we consider different case characters, more complexity is added. A binary collation would evaluate the characters `A` and `a` differently, ordering them in a particular way. A case-insensitive collation would evaluate `A` and `a` equivalently, while the German phone book collation evaluates the characters `ue` and `ü` equivalently.

A character set can have many collations associated with it, while each collation is only associated with one character set. In MariaDB, the character set name is always part of the collation name. For example, the `latin1_german1_ci` collation applies only to the `latin1` character set. Each character set also has one default collation. The `latin1` default collation is `latin1_swedish_ci`.

As an example, by default, the character `y` comes between `x` and `z`, while in Lithuanian, it's sorted between `i` and `k`. Similarly, the German phone book order is different to the German dictionary order, so while they share the same character set, the collation is different.

Viewing Character Sets and Collations

> **CURRENT**
>
> The default character set is `utf8mb4` and the default collation is `utf8mb4_uca1400_ai_ci`.
> This may differ in some distros, see for example Differences in MariaDB in Debian.
>
> **< 11.6**
>
> The default character set is `latin1` and the default collation is `latin1_swedish_ci`.
> This may differ in some distros, see for example Differences in MariaDB in Debian.

You can view a full list of character sets and collations supported by MariaDB at Supported Character Sets and Collations, or see what's supported on your server with the SHOW CHARACTER SET and SHOW COLLATION commands.

By default, `A` comes before `Z`, so the following evaluates to true:

```
SELECT "A" < "Z";
+-----------+
| "A" < "Z" |
+-----------+
|         1 |
+-----------+
```

By default, comparisons are case-insensitive:

```
SELECT "A" < "a", "A" = "a";
+-----------+-----------+
| "A" < "a" | "A" = "a" |
+-----------+-----------+
|         0 |         1 |
+-----------+-----------+
```

Changing Character Sets and Collations

Character sets and collations can be set from the server level right down to the column level, as well as for client-server communication.

For example, `ue` and `ü` are by default evaluated differently.

```
SELECT 'Mueller' = 'Müller';
+----------------------+
| 'Müller' = 'Mueller' |
+----------------------+
|                    0 |
+----------------------+
```

By using the collation_connection system variable to change the connection character set to `latin1_german2_ci`, or German phone book, the same two characters will evaluate as equivalent.

```
SET collation_connection = latin1_german2_ci;

SELECT 'Mueller' = 'Müller';
+-----------------------+
| 'Mueller' = 'Müller'  |
+-----------------------+
|                     1 |
+-----------------------+
```

See Setting Character Sets and Collations for more.

This page is licensed: CC BY-SA / Gnu FDL

Setting Character Sets and Collations

> **CURRENT**
>
> The default character set is `utf8mb4` and the default collation is `utf8mb4_uca1400_ai_ci`.
> This may differ in some distros, see for example Differences in MariaDB in Debian.
>
> **< 11.6**
>
> The default character set is `latin1` and the default collation is `latin1_swedish_ci`.
> This may differ in some distros, see for example Differences in MariaDB in Debian.

> **Warning:**
>
> In MariaDB 11.6, the default character set changed from `latin1` to `utf8mb4`.
>
> When upgrading to 11.6 or above from a previous release series, this can lead to behavior different from what you've been seeing in the old version.
>
> See this section for details, including the impact on replicating to older MariaDB (or MySQL) replicas.

The character sets and the collations can be specified from the server right down to the column level, as well as for client-server connections. When changing a character set and not specifying a collation, the default collation for the new character set is always used.

Character sets and collations always cascade down, so a column without a specified collation will look for the table default, the table for the database, and the database for the server. It's therefore possible to have extremely fine-grained control over all the character sets and collations used in your data.

Default collations for each character set can be viewed with the SHOW COLLATION statement, for example, to find the default collation for the latin2 character set:

```
SHOW COLLATION LIKE 'latin2%';
+--------------------+---------+----+---------+----------+---------+
| Collation          | Charset | Id | Default | Compiled | Sortlen |
+--------------------+---------+----+---------+----------+---------+
| latin2_czech_cs    | latin2  |  2 |         | Yes      |       4 |
| latin2_general_ci  | latin2  |  9 | Yes     | Yes      |       1 |
| latin2_hungarian_ci | latin2  | 21 |         | Yes      |       1 |
| latin2_croatian_ci | latin2  | 27 |         | Yes      |       1 |
| latin2_bin         | latin2  | 77 |         | Yes      |       1 |
+--------------------+---------+----+---------+----------+---------+
```

Server Level

The character_set_server system variable can be used to change the default server character set. It can be set both on startup or dynamically, with the SET command:

```
SET character_set_server = 'latin2';
```

Similarly, the collation_server variable is used for setting the default server collation.

```
SET collation_server = 'latin2_czech_cs';
```

Database Level

The CREATE DATABASE and ALTER DATABASE statements have optional character set and collation clauses. If these are left out, the server defaults are used.

```
CREATE DATABASE czech_slovak_names
  CHARACTER SET = 'keybcs2'
  COLLATE = 'keybcs2_bin';
```

```
ALTER DATABASE czech_slovak_names COLLATE = 'keybcs2_general_ci';
```

To determine the default character set used by a database, use:

```
SHOW CREATE DATABASE czech_slovak_names;
+--------------------+-----------------------------------------------------------------------------------+
| Database           | Create Database                                                                   |
+--------------------+-----------------------------------------------------------------------------------+
| czech_slovak_names | CREATE DATABASE `czech_slovak_names` /*!40100 DEFAULT CHARACTER SET keybcs2 */ |
+--------------------+-----------------------------------------------------------------------------------+
```

Alternatively, for the character set and collation:

```
SELECT * FROM INFORMATION_SCHEMA.SCHEMATA;
+--------------+--------------------+----------------------------+------------------------+----------+
| CATALOG_NAME | SCHEMA_NAME        | DEFAULT_CHARACTER_SET_NAME | DEFAULT_COLLATION_NAME | SQL_PATH |
+--------------+--------------------+----------------------------+------------------------+----------+
| def          | czech_slovak_names | keybcs2                    | keybcs2_general_ci     | NULL     |
| def          | information_schema | utf8                       | utf8_general_ci        | NULL     |
| def          | mysql              | latin1                     | latin1_swedish_ci      | NULL     |
| def          | performance_schema | utf8                       | utf8_general_ci        | NULL     |
| def          | test               | latin1                     | latin1_swedish_ci      | NULL     |
+--------------+--------------------+----------------------------+------------------------+----------+
```

It is also possible to specify only the collation, and, since each collation only applies to one character set, the associated character set will automatically be specified.

```
CREATE DATABASE danish_names COLLATE 'utf8_danish_ci';

SHOW CREATE DATABASE danish_names;
+--------------+-----------------------------------------------------------------------------------------------
-----+
| Database     | Create Database
|
+--------------+-----------------------------------------------------------------------------------------------
-----+
| danish_names | CREATE DATABASE `danish_names` /*!40100 DEFAULT CHARACTER SET utf8 COLLATE
utf8_danish_ci */ |
+--------------+-----------------------------------------------------------------------------------------------
-----+
```

Although there are character_set_database and collation_database system variables which can be set dynamically, these are used for determining the character set and collation for the default database, and should only be set by the server.

A database's collation is inherited only by statements that don't name a character set of their own. A `CREATE TABLE` that names one, such as `CREATE TABLE ... DEFAULT CHARACTER SET utf8mb4`, takes its collation from the character_set_collations map instead. See Table Level and Changing Default Collation.

Table Level

The CREATE TABLE and ALTER TABLE statements support optional character set and collation clauses, a MariaDB and MySQL extension to standard SQL.

```
CREATE TABLE english_names (id INT, name VARCHAR(40))
  CHARACTER SET 'utf8'
  COLLATE 'utf8_icelandic_ci';
```

If neither character set nor collation is provided, the database default will be used. If only the character set is provided, the default collation for that character set will be used. If only the collation is provided, the associated character set will be used. See Supported Character Sets and Collations.

Warning:

Naming a character set without a `COLLATE` clause does **not** inherit the database's collation. The collation comes from the character_set_collations map instead, which has a non-empty default value since MariaDB 11.5. So these two statements can produce different collations in the same database:

```
CREATE TABLE t1 (id VARCHAR(24));                               -- inherits the database collation
CREATE TABLE t2 (id VARCHAR(24)) DEFAULT CHARACTER SET utf8mb4;  -- uses character_set_collations
```

In a database whose default collation is `utf8mb4_general_ci`, `t1` is `utf8mb4_general_ci` while `t2` is `utf8mb4_uca1400_ai_ci`. To get the same result from both, either add an explicit `COLLATE` clause or change the map — see Changing Default Collation.

```
ALTER TABLE table_name
 CONVERT TO CHARACTER SET charset_name [COLLATE collation_name];
```

If no collation is provided, the collation will be set to the default collation for that character set. See Supported Character Sets and Collations.

For VARCHAR or TEXT columns, `CONVERT TO CHARACTER SET` changes the data type if needed to ensure the new column is long enough to store as many characters as the original column.

For example, an ascii `TEXT` column requires a single byte per character, so the column can hold up to 65,535 characters. If the column is converted to `utf8mb4`, 4 bytes can be required for each character, so the column will be

converted to MEDIUMTEXT to be able to hold the same number of characters.

`CONVERT TO CHARACTER SET binary` will convert CHAR, VARCHAR and TEXT columns to BINARY, VARBINARY and BLOB respectively, and from that point will no longer have a character set, or be affected by future `CONVERT TO CHARACTER SET` statements.

To avoid data type changes resulting from `CONVERT TO CHARACTER SET`, use `MODIFY` on the individual columns instead. For example:

```
ALTER TABLE table_name MODIFY ascii_text_column TEXT CHARACTER SET utf8;
ALTER TABLE table_name MODIFY ascii_varchar_column VARCHAR(M) CHARACTER SET utf8;
```

Column Level

Character sets and collations can also be specified for columns that are character types `CHAR`, `TEXT` or `VARCHAR`. The CREATE TABLE and ALTER TABLE statements support optional character set and collation clauses for this purpose - unlike those at the table level, the column level definitions are standard SQL.

```
CREATE TABLE european_names (
  croatian_names VARCHAR(40) COLLATE 'cp1250_croatian_ci',
  greek_names VARCHAR(40) CHARACTER SET 'greek');
```

If neither collation nor character set is provided, the table default is used. If only the character set is specified, that character set's default collation is used, while if only the collation is specified, the associated character set is used.

When using ALTER TABLE to change a column's character set, you need to ensure the character sets are compatible with your data. MariaDB will map the data as best it can, but it's possible to lose data if care is not taken.

The SHOW CREATE TABLE statement or `INFORMATION SCHEMA` database can be used to determine column character sets and collations.

```
SHOW CREATE TABLE european_names\G
*************************** 1. row ***************************
       Table: european_names
Create Table: CREATE TABLE `european_names` (
  `croatian_names` varchar(40) CHARACTER SET cp1250 COLLATE cp1250_croatian_ci DEFAULT NULL,
  `greek_names` varchar(40) CHARACTER SET greek DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_danish_ci
```

```
SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME LIKE 'european%'\G
*************************** 1. row ***************************
           TABLE_CATALOG: def
            TABLE_SCHEMA: danish_names
              TABLE_NAME: european_names
             COLUMN_NAME: croatian_names
        ORDINAL_POSITION: 1
          COLUMN_DEFAULT: NULL
             IS_NULLABLE: YES
               DATA_TYPE: varchar
CHARACTER_MAXIMUM_LENGTH: 40
  CHARACTER_OCTET_LENGTH: 40
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: cp1250
          COLLATION_NAME: cp1250_croatian_ci
             COLUMN_TYPE: varchar(40)
              COLUMN_KEY:
                   EXTRA:
              PRIVILEGES: select,insert,update,references
          COLUMN_COMMENT:
*************************** 2. row ***************************
           TABLE_CATALOG: def
            TABLE_SCHEMA: danish_names
              TABLE_NAME: european_names
             COLUMN_NAME: greek_names
        ORDINAL_POSITION: 2
          COLUMN_DEFAULT: NULL
             IS_NULLABLE: YES
               DATA_TYPE: varchar
CHARACTER_MAXIMUM_LENGTH: 40
  CHARACTER_OCTET_LENGTH: 40
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: greek
          COLLATION_NAME: greek_general_ci
             COLUMN_TYPE: varchar(40)
              COLUMN_KEY:
                   EXTRA:
              PRIVILEGES: select,insert,update,references
          COLUMN_COMMENT:
```

Filenames

The character_set_filesystem system variable has controlled interpretation of file names that are given as literal strings. This affects the following statements and functions:

- SELECT INTO DUMPFILE
- SELECT INTO OUTFILE
- LOAD DATA INFILE
- LOAD XML
- LOAD_FILE()

Literals

By default, the character set and collation used for literals is determined by the character_set_connection and collation_connection system variables. However, they can also be specified explicitly:

```
[_charset_name]'string' [COLLATE collation_name]
```

The character set of string literals that do not have a character set introducer is determined by the character_set_connection system variable.

This query always returns the same character set name in both columns.:

```
SELECT CHARSET('a'), @@character_set_connection;
```

character_set_client and character_set_connection are normally (e.g. during handshake, or after a `SET NAMES` query) are set to equal values. However, it's possible to set to different values.

Examples

Examples when setting `@@character_set_client` and `@@character_set_connection` to different values can be useful:

Example 1:

Suppose, we have a `utf8` database with this table:

```
CREATE TABLE t1 (a VARCHAR(10)) CHARACTER SET utf8 COLLATE utf8_general_ci;
INSERT INTO t1 VALUES ('oe'),('ö');
```

Now we connect to it using a client which uses the DOS character set (cp850 on a West European machine), and want to fetch all records that are equal to 'ö' according to the German phonebook rules.

It's possible with the following:

```
SET @@character_set_client=cp850, @@character_set_connection=utf8;
SELECT a FROM t1 WHERE a='ö' COLLATE utf8_german2_ci;
```

This will return:

```
+------+
| a    |
+------+
| oe   |
| ö    |
+------+
```

It works as follows:

1. The client sends the query using `cp850`.
2. The server, when parsing the query, creates a utf8 string literal by converting 'ö' from `@@character_set_client` (cp850) to `@@character_set_connection` (utf8).
3. The server applies the collation `utf8_german2_ci` to this string literal.
4. The server uses `utf8_german2_ci` for comparison.

Note, if we rewrite the script like this:

```
SET NAMES cp850;
SELECT a FROM t1 WHERE a='ö' COLLATE utf8_german2_ci;
```

We get an error:

```
ERROR 1253 (42000): COLLATION 'utf8_german2_ci' is not valid for CHARACTER SET 'cp850'
```

Reason:

- In step #2, the literal is not converted to `utf8` any more and is created using `cp850`.

- In step #3, the server fails to apply `utf8_german2_ci` to an `cp850` string literal.

Example 2:

Suppose we have a utf8 database and use “mysql.exe” from a West European machine again.

We can do this:

```
SET @@character_set_client=cp850, @@character_set_connection=utf8;
CREATE TABLE t2 AS SELECT 'ö';
```

It creates a table with a column of the type `VARCHAR(1) CHARACTER SET utf8`.

Note, if we rewrite the query like this:

```
SET NAMES cp850;
CREATE TABLE t2 AS SELECT 'ö';
```

It creates a table with a column of the type `VARCHAR(1) CHARACTER SET cp850`, which is probably not a good idea.

N

Also, `N` or `n` can be used as prefix to convert a literal into the National Character set (which in MariaDB is always `utf8`).

For example:

```
SELECT _latin2 'Müller';
+-----------+
| MÄźller   |
+-----------+
| MÄźller   |
+-----------+
```

CURRENT

```
SELECT CHARSET(N'a string');
+----------------------+
| CHARSET(N'a string') |
+----------------------+
| utf8mb3              |
+----------------------+
```

```
SELECT 'Mueller' = 'Müller' COLLATE 'latin1_german2_ci';
+-----------------------------------------------------+
| 'Mueller' = 'Müller' COLLATE 'latin1_german2_ci'  |
+-----------------------------------------------------+
|                                                   1 |
+-----------------------------------------------------+
```

< 10.6

```
SELECT CHARSET(N'a string');
+----------------------+
| CHARSET(N'a string') |
+----------------------+
| utf8                 |
+----------------------+
```

Stored Programs and Views

The literals which occur in stored programs and views, by default, use the character set and collation which was specified by the character_set_connection and collation_connection system variables when the stored program was created. These values can be seen using the `SHOW CREATE` statements. To change the character sets used for literals in an existing stored program, it is necessary to drop and recreate the stored program.

For stored routines parameters and return values, a character set and a collation can be specified via the `CHARACTER SET` and `COLLATE` clauses.

The following example shows that the character set and collation are determined at the time of creation:

```
SET @@local.character_set_connection='latin1';

DELIMITER ||
CREATE PROCEDURE `test`.`x`()
BEGIN
    SELECT CHARSET('x');
END;
||
Query OK, 0 rows affected (0.00 sec)

DELIMITER ;
SET @@local.character_set_connection='utf8';

CALL `test`.`x`();
+--------------+
| CHARSET('x') |
+--------------+
| latin1       |
+--------------+
```

The following example shows how to specify a function parameters character set and collation:

```
CREATE FUNCTION `test`.`y`(`str` TEXT CHARACTER SET utf8 COLLATE utf8_bin)
    RETURNS TEXT CHARACTER SET latin1 COLLATE latin1_bin
BEGIN
    SET @param_coll = COLLATION(`str`);
    RETURN `str`;
END;

-- return value's collation:
SELECT COLLATION(`test`.`y`('Hello, planet!'));
+----------------------------------------+
| COLLATION(`test`.`y`('Hello, planet!')) |
+----------------------------------------+
| latin1_bin                             |
+----------------------------------------+

-- parameter's collation:
SELECT @param_coll;
+-------------+
| @param_coll |
+-------------+
| utf8_bin    |
+-------------+
```

Changing Default Collation

> **CURRENT**
>
> The default collation associated with a particular character set is determined by the character_set_collations system variable, which accepts a comma-delimited list of character sets and their default collations, for example:

```
SET @@character_set_collations = 'utf8mb4=uca1400_ai_ci, latin2=latin2_hungarian_ci';
```

The variable is **not** empty by default. Since MariaDB 11.5, its default value maps every Unicode character set to a UCA 14.0.0 collation, so the map applies even on a server where nobody has ever set the variable. To see the value in effect:

```
SELECT @@character_set_collations;
+-----------------------------------------------------------------------------------------------------------
----------------------------------+
| @@character_set_collations
|
+-----------------------------------------------------------------------------------------------------------
----------------------------------+
|
utf8mb3=utf8mb3_uca1400_ai_ci,ucs2=ucs2_uca1400_ai_ci,utf8mb4=utf8mb4_uca1400_ai_ci,utf16=utf16_uca1400_
ai_ci,utf32=utf32_uca1400_ai_ci |
+-----------------------------------------------------------------------------------------------------------
----------------------------------+
```

A collation can be given either by its full name, such as `utf8mb4_uca1400_ai_ci`, or by the shorter family name that applies to any character set, such as `uca1400_ai_ci`. The server stores the full name, so the value it reports back is always the expanded form.

< 11.5

The variable exists but is empty by default, so each character set uses its compiled-in default collation, such as `utf8mb4_general_ci` for `utf8mb4`.

< 11.2.1

It is **not** possible to change the default collation associated with a particular character set.

The variable takes effect in all cases where a character set is explicitly or implicitly specified without an explicit `COLLATE` clause, including but not limited to:

- Column collation
- Table collation
- Database collation
- `CHAR` (expr `USING` csname)
- `CONVERT` (expr `USING` csname)
- `CAST` (expr `AS CHAR CHARACTER SET` csname)
- '' - character string literal
- `_utf8mb3'text'` - a character string literal with an introducer
- `_utf8mb3 X'61'` - a character string literal with an introducer with hex notation
- `_utf8mb3 0x61` - a character string literal with an introducer with hex hybrid notation
- `@@collation_connection` after a `SET NAMES` statement without `COLLATE`

Overriding the Map for One Character Set

Assigning to the variable **replaces the entire map**; it does not merge with the value already in effect. Setting only `utf8mb4` therefore returns the other Unicode character sets to their compiled-in defaults:

```
SET GLOBAL character_set_collations = 'utf8mb4=utf8mb4_general_ci';
```

To change `utf8mb4` while keeping the UCA 14.0.0 defaults for the rest, list them all:

```
SET GLOBAL character_set_collations =

'utf8mb3=uca1400_ai_ci,ucs2=uca1400_ai_ci,utf16=uca1400_ai_ci,utf32=uca1400_ai_ci,utf8mb4=utf8mb4_general
_ci';
```

To make either change permanent, set it in a configuration file:

```
[mariadbd]
character-set-collations = utf8mb4=utf8mb4_general_ci
```

Note:

`SET GLOBAL` affects new connections only. Existing connections keep the session value they inherited when they connected, so reconnect the application before relying on the new map.

Default Character Set and Collation Changes

Note:

The default character set and collation changed in MariaDB 11.6. Separately, the default collation applied to an explicitly named Unicode character set changed in MariaDB 11.5 — see Changing Default Collation.

MariaDB 11.8 is the first long-term support release containing either change, so both take effect at once when upgrading from an earlier long-term support release such as MariaDB 10.6 or 11.4.

The default character set has changed from `latin1` to `utf8mb4`, and the default collation has changed from `latin1_swedish_ci` to `utf8mb4_uca1400_ai_ci`. This update improves global compatibility and supports modern data requirements, such as emojis.

Why the Defaults Changed

The shift to `utf8mb4` and UCA-based collations provides several benefits:

- Global Compatibility: The new defaults support users worldwide without requiring additional configuration, whereas the previous `latin1` defaults were primarily suited for West European languages.
- Supplementary Character Support: You can now store supplementary characters, such as emojis, which were not supported by `latin1`.
- Accurate Sorting and Comparison: The `utf8mb4_uca1400_ai_ci` collation correctly handles supplementary characters and supports expansions and contractions from the Default Unicode Collation Element Table (DUCET). For example, the German character “ß” is correctly compared as equal to “ss”.

Important Considerations

Before upgrading, please be aware of the following technical implications:

- The storage overhead for `CHAR(N)` indeed significantly increases. The server reserved `N` bytes for a `CHAR(N) CHARACTER SET latin1` column in every record. Now it must reserve `N*4` bytes for a `CHAR(N) CHARACTER SET utf8mb4` column, and fill the unused bytes with trailing spacing.
- The storage overhead for a `VARCHAR(N)` is not really palpable because the server only stores actual strings without padding. West European languages mostly use basic ASCII letters, only rarely accented letters. For example, for German, the letter use statistics is here: https://www.sttmedia.com/characterfrequency-german. After switching from latin1 to utf8mb4, only accented letters need more storage, but since they’re making up only for a small portion, the overall growth is insignificant.
- The previous two points indicate that it might be worthwhile to consider switching from `CHAR` to `VARCHAR` columns.

- Replication Impact: Because `utf8mb4_uca1400_ai_ci` was not available in earlier versions, replication from MariaDB 11.8 to MariaDB 10.6 will fail unless the server is configured to use the old defaults.
- Storage Overhead: Using `utf8mb4` may increase storage requirements for `CHAR` and `VARCHAR` columns, particularly for data containing non-ASCII characters. This is particularly true when table columns contain `latin1` code points outside of the ASCII range.
- Performance: Some comparison operations may be slower with the new collation compared to the fixed-width `latin1_swedish_ci` collation.
- Application Compatibility: Some applications that rely on `latin1` behavior may require updates to function correctly with the new defaults. Problems can occur because a `latin1` column can always compare to a binary string, while a `utf8mb4` column cannot (at least not always), because not every binary string is a well-formed `utf8mb4` string. A workaround is to cast the `utf8mb4` column to `BINARY` before doing a comparison.

Restoring Old Defaults

If you need to maintain compatibility with MariaDB 10.6 replicas or require the specific performance characteristics of the previous defaults, you can configure the server to use the old settings.

To return to the `latin1` defaults, add the following lines to your `my.cnf` configuration file:

```
[mariadb]
character-set-server=latin1
collation-server=latin1_swedish_ci
character-set-collations=''
```

> Note: These settings make data files compatible with older versions like MariaDB 10.6, which lack the newer UCA collation support.

Example: Changing the Default Character Set To UTF-8

To change the default character set from latin1 to UTF-8, the following settings should be specified in the my.cnf configuration file.

```
[mysql]
...
default-character-set=utf8mb4
...
[mysqld]
...
collation-server = utf8mb4_unicode_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
...
```

Note that the `default-character-set` option is a client option, not a server option.

See Also

- String literals
- CAST()
- CONVERT()
- Operator Precedence

This page is licensed: CC BY-SA / Gnu FDL

Supported Character Sets and Collations

Character Sets

You can see which character sets are available in a particular version by running the SHOW CHARACTER SET statement or by querying the Information Schema CHARACTER_SETS Table.

> **CURRENT**
>
> It is possible to change the default collation associated with a character set. See Changing Default Collation.
>
> **< 11.2**
>
> It is **not** possible to change the default collation associated with a character set. See Changing Default Collation

MariaDB supports the following character sets:

Charset	Description	Default collation	Maxlen
armscii8	ARMSCII-8 Armenian	armscii8_general_ci	1
ascii	US ASCII	ascii_general_ci	1
big5	Big5 Traditional Chinese	big5_chinese_ci	2
binary	Binary pseudo charset	binary	1
cp1250	Windows Central European	cp1250_general_ci	1
cp1251	Windows Cyrillic	cp1251_general_ci	1
cp1256	Windows Arabic	cp1256_general_ci	1
cp1257	Windows Baltic	cp1257_general_ci	1
cp850	DOS West European	cp850_general_ci	1
cp852	DOS Central European	cp852_general_ci	1
cp866	DOS Russian	cp866_general_ci	1
cp932	SJIS for Windows Japanese	cp932_japanese_ci	2
dec8	DEC West European	dec8_swedish_ci	1
eucjpms	UJIS for Windows Japanese	eucjpms_japanese_ci	3
euckr	EUC-KR Korean	euckr_korean_ci	2
gb2312	GB2312 Simplified Chinese	gb2312_chinese_ci	2
gbk	GBK Simplified Chinese	gbk_chinese_ci	2
geostd8	GEOSTD8 Georgian	geostd8_general_ci	1
greek	ISO 8859-7 Greek	greek_general_ci	1
hebrew	ISO 8859-8 Hebrew	hebrew_general_ci	1
hp8	HP West European	hp8_english_ci	1
keybcs2	DOS Kamenicky Czech-Slovak	keybcs2_general_ci	1
koi8r	KOI8-R Relcom Russian	koi8r_general_ci	1
koi8u	KOI8-U Ukrainian	koi8u_general_ci	1
latin1	cp1252 West European	latin1_swedish_ci	1
latin2	ISO 8859-2 Central European	latin2_general_ci	1

Charset	Description	Default collation	Maxlen
latin5	ISO 8859-9 Turkish	latin5_turkish_ci	1
latin7	ISO 8859-13 Baltic	latin7_general_ci	1
macce	Mac Central European	macce_general_ci	1
macroman	Mac West European	macroman_general_ci	1
sjis	Shift-JIS Japanese	sjis_japanese_ci	2
swe7	7bit Swedish	swe7_swedish_ci	1
tis620	TIS620 Thai	tis620_thai_ci	1
ucs2	UCS-2 Unicode		2
ujis	EUC-JP Japanese	ujis_japanese_ci	3
utf8	UTF-8 Unicode	utf8_general_ci	3/4 (see OLD_MODE)
utf16	UTF-16 Unicode		4
utf16le	UTF-16LE Unicode	utf16le_general_ci	4
utf32	UTF-32 Unicode	utf32_general_ci (<= MariaDB 11.4) utf32_uca1400_ai_ci (>=MariaDB 11.5)	4
utf8mb3	UTF-8 Unicode	utf8mb3_general_ci (<= MariaDB 11.4) utf8mb3_uca1400_ai_ci (>=MariaDB 11.5)	3
utf8mb4	UTF-8 Unicode	utf8mb4_general_ci (<= MariaDB 11.4) utf8mb4_uca1400_ai_ci (>=MariaDB 11.5)	4

Warning:

Note that the Mroonga Storage Engine only supports a limited number of character sets. See Mroonga's available character sets.

Collations

MariaDB supports the following collations (from MariaDB 11.4.5):

```
SELECT collation_name, character_set_name as cs_name, id, is_default AS def, is_compiled AS com,
  sortlen, comment FROM information_schema.collations ORDER BY collation_name;
+--------------------------------+----------+------+------+-----+---------+-----------------------------
--------------------+
| collation_name                 | cs_name  | id   | def  | com | sortlen | comment
|
+--------------------------------+----------+------+------+-----+---------+-----------------------------
--------------------+
| armscii8_bin                   | armscii8 |   64 |      | Yes |       1 | ARMSCII-8 Armenian
|
| armscii8_general_ci            | armscii8 |   32 | Yes  | Yes |       1 | ARMSCII-8 Armenian
|
| armscii8_general_nopad_ci      | armscii8 | 1056 |      | Yes |       1 | ARMSCII-8 Armenian
|
| armscii8_nopad_bin             | armscii8 | 1088 |      | Yes |       1 | ARMSCII-8 Armenian
|
| ascii_bin                      | ascii    |   65 |      | Yes |       1 | US ASCII
|
| ascii_general_ci               | ascii    |   11 | Yes  | Yes |       1 | US ASCII
|
| ascii_general_nopad_ci         | ascii    | 1035 |      | Yes |       1 | US ASCII
|
| ascii_nopad_bin                | ascii    | 1089 |      | Yes |       1 | US ASCII
|
| big5_bin                       | big5     |   84 |      | Yes |       1 | Big5 Traditional Chinese
|
| big5_chinese_ci                | big5     |    1 | Yes  | Yes |       1 | Big5 Traditional Chinese
|
| big5_chinese_nopad_ci          | big5     | 1025 |      | Yes |       1 |
|
| big5_nopad_bin                 | big5     | 1108 |      | Yes |       1 |
|
| binary                         | binary   |   63 | Yes  | Yes |       1 | Binary pseudo charset
|
| cp1250_bin                     | cp1250   |   66 |      | Yes |       1 | Windows Central European
|
| cp1250_croatian_ci             | cp1250   |   44 |      | Yes |       1 | Windows Central European
|
| cp1250_czech_cs                | cp1250   |   34 |      | Yes |       2 | Windows Central European
|
| cp1250_general_ci              | cp1250   |   26 | Yes  | Yes |       1 | Windows Central European
|
| cp1250_general_nopad_ci        | cp1250   | 1050 |      | Yes |       1 | Windows Central European
|
| cp1250_nopad_bin               | cp1250   | 1090 |      | Yes |       1 | Windows Central European
|
| cp1250_polish_ci               | cp1250   |   99 |      | Yes |       1 | Windows Central European
|
| cp1251_bin                     | cp1251   |   50 |      | Yes |       1 | Windows Cyrillic
|
| cp1251_bulgarian_ci            | cp1251   |   14 |      | Yes |       1 | Windows Cyrillic
|
| cp1251_general_ci              | cp1251   |   51 | Yes  | Yes |       1 | Windows Cyrillic
|
| cp1251_general_cs              | cp1251   |   52 |      | Yes |       1 | Windows Cyrillic
|
| cp1251_general_nopad_ci        | cp1251   | 1075 |      | Yes |       1 | Windows Cyrillic
|
| cp1251_nopad_bin               | cp1251   | 1074 |      | Yes |       1 | Windows Cyrillic
|
| cp1251_ukrainian_ci            | cp1251   |   23 |      | Yes |       1 | Windows Cyrillic
|
| cp1256_bin                     | cp1256   |   67 |      | Yes |       1 | Windows Arabic
|
| cp1256_general_ci              | cp1256   |   57 | Yes  | Yes |       1 | Windows Arabic
|
| cp1256_general_nopad_ci        | cp1256   | 1081 |      | Yes |       1 | Windows Arabic
|
```

```
| cp1256_nopad_bin               | cp1256   | 1091 |      | Yes |       1 | Windows Arabic
|
| cp1257_bin                     | cp1257   |   58 |      | Yes |       1 | Windows Baltic
|
| cp1257_general_ci              | cp1257   |   59 | Yes  | Yes |       1 | Windows Baltic
|
| cp1257_general_nopad_ci        | cp1257   | 1083 |      | Yes |       1 | Windows Baltic
|
| cp1257_lithuanian_ci           | cp1257   |   29 |      | Yes |       1 | Windows Baltic
|
| cp1257_nopad_bin               | cp1257   | 1082 |      | Yes |       1 | Windows Baltic
|
| cp850_bin                      | cp850    |   80 |      | Yes |       1 | DOS West European
|
| cp850_general_ci               | cp850    |    4 | Yes  | Yes |       1 | DOS West European
|
| cp850_general_nopad_ci         | cp850    | 1028 |      | Yes |       1 | DOS West European
|
| cp850_nopad_bin                | cp850    | 1104 |      | Yes |       1 | DOS West European
|
| cp852_bin                      | cp852    |   81 |      | Yes |       1 | DOS Central European
|
| cp852_general_ci               | cp852    |   40 | Yes  | Yes |       1 | DOS Central European
|
| cp852_general_nopad_ci         | cp852    | 1064 |      | Yes |       1 | DOS Central European
|
| cp852_nopad_bin                | cp852    | 1105 |      | Yes |       1 | DOS Central European
|
| cp866_bin                      | cp866    |   68 |      | Yes |       1 | DOS Russian
|
| cp866_general_ci               | cp866    |   36 | Yes  | Yes |       1 | DOS Russian
|
| cp866_general_nopad_ci         | cp866    | 1060 |      | Yes |       1 | DOS Russian
|
| cp866_nopad_bin                | cp866    | 1092 |      | Yes |       1 | DOS Russian
|
| cp932_bin                      | cp932    |   96 |      | Yes |       1 | SJIS for Windows Japanese
|
| cp932_japanese_ci              | cp932    |   95 | Yes  | Yes |       1 | SJIS for Windows Japanese
|
| cp932_japanese_nopad_ci        | cp932    | 1119 |      | Yes |       1 |
|
| cp932_nopad_bin                | cp932    | 1120 |      | Yes |       1 |
|
| dec8_bin                       | dec8     |   69 |      | Yes |       1 | DEC West European
|
| dec8_nopad_bin                 | dec8     | 1093 |      | Yes |       1 | DEC West European
|
| dec8_swedish_ci                | dec8     |    3 | Yes  | Yes |       1 | DEC West European
|
| dec8_swedish_nopad_ci          | dec8     | 1027 |      | Yes |       1 | DEC West European
|
| eucjpms_bin                    | eucjpms  |   98 |      | Yes |       1 | UJIS for Windows Japanese
|
| eucjpms_japanese_ci            | eucjpms  |   97 | Yes  | Yes |       1 | UJIS for Windows Japanese
|
| eucjpms_japanese_nopad_ci      | eucjpms  | 1121 |      | Yes |       1 |
|
| eucjpms_nopad_bin              | eucjpms  | 1122 |      | Yes |       1 |
|
| euckr_bin                      | euckr    |   85 |      | Yes |       1 | EUC-KR Korean
|
| euckr_korean_ci                | euckr    |   19 | Yes  | Yes |       1 | EUC-KR Korean
|
| euckr_korean_nopad_ci          | euckr    | 1043 |      | Yes |       1 |
|
| euckr_nopad_bin                | euckr    | 1109 |      | Yes |       1 |
|
| gb2312_bin                     | gb2312   |   86 |      | Yes |       1 | GB2312 Simplified Chinese
```

```
|
| gb2312_chinese_ci              | gb2312   |   24 | Yes  | Yes |       1 | GB2312 Simplified Chinese
|
| gb2312_chinese_nopad_ci        | gb2312   | 1048 |      | Yes |       1 |
|
| gb2312_nopad_bin               | gb2312   | 1110 |      | Yes |       1 |
|
| gbk_bin                        | gbk      |   87 |      | Yes |       1 | GBK Simplified Chinese
|
| gbk_chinese_ci                 | gbk      |   28 | Yes  | Yes |       1 | GBK Simplified Chinese
|
| gbk_chinese_nopad_ci           | gbk      | 1052 |      | Yes |       1 |
|
| gbk_nopad_bin                  | gbk      | 1111 |      | Yes |       1 |
|
| geostd8_bin                    | geostd8  |   93 |      | Yes |       1 | GEOSTD8 Georgian
|
| geostd8_general_ci             | geostd8  |   92 | Yes  | Yes |       1 | GEOSTD8 Georgian
|
| geostd8_general_nopad_ci       | geostd8  | 1116 |      | Yes |       1 | GEOSTD8 Georgian
|
| geostd8_nopad_bin              | geostd8  | 1117 |      | Yes |       1 | GEOSTD8 Georgian
|
| greek_bin                      | greek    |   70 |      | Yes |       1 | ISO 8859-7 Greek
|
| greek_general_ci               | greek    |   25 | Yes  | Yes |       1 | ISO 8859-7 Greek
|
| greek_general_nopad_ci         | greek    | 1049 |      | Yes |       1 | ISO 8859-7 Greek
|
| greek_nopad_bin                | greek    | 1094 |      | Yes |       1 | ISO 8859-7 Greek
|
| hebrew_bin                     | hebrew   |   71 |      | Yes |       1 | ISO 8859-8 Hebrew
|
| hebrew_general_ci              | hebrew   |   16 | Yes  | Yes |       1 | ISO 8859-8 Hebrew
|
| hebrew_general_nopad_ci        | hebrew   | 1040 |      | Yes |       1 | ISO 8859-8 Hebrew
|
| hebrew_nopad_bin               | hebrew   | 1095 |      | Yes |       1 | ISO 8859-8 Hebrew
|
| hp8_bin                        | hp8      |   72 |      | Yes |       1 | HP West European
|
| hp8_english_ci                 | hp8      |    6 | Yes  | Yes |       1 | HP West European
|
| hp8_english_nopad_ci           | hp8      | 1030 |      | Yes |       1 | HP West European
|
| hp8_nopad_bin                  | hp8      | 1096 |      | Yes |       1 | HP West European
|
| keybcs2_bin                    | keybcs2  |   73 |      | Yes |       1 | DOS Kamenicky Czech-Slovak
|
| keybcs2_general_ci             | keybcs2  |   37 | Yes  | Yes |       1 | DOS Kamenicky Czech-Slovak
|
| keybcs2_general_nopad_ci       | keybcs2  | 1061 |      | Yes |       1 | DOS Kamenicky Czech-Slovak
|
| keybcs2_nopad_bin              | keybcs2  | 1097 |      | Yes |       1 | DOS Kamenicky Czech-Slovak
|
| koi8r_bin                      | koi8r    |   74 |      | Yes |       1 | KOI8-R Relcom Russian
|
| koi8r_general_ci               | koi8r    |    7 | Yes  | Yes |       1 | KOI8-R Relcom Russian
|
| koi8r_general_nopad_ci         | koi8r    | 1031 |      | Yes |       1 | KOI8-R Relcom Russian
|
| koi8r_nopad_bin                | koi8r    | 1098 |      | Yes |       1 | KOI8-R Relcom Russian
|
| koi8u_bin                      | koi8u    |   75 |      | Yes |       1 | KOI8-U Ukrainian
|
| koi8u_general_ci               | koi8u    |   22 | Yes  | Yes |       1 | KOI8-U Ukrainian
|
| koi8u_general_nopad_ci         | koi8u    | 1046 |      | Yes |       1 | KOI8-U Ukrainian
|
```

```
| koi8u_nopad_bin                | koi8u    | 1099 |      | Yes |       1 | KOI8-U Ukrainian
|
| latin1_bin                     | latin1   |   47 |      | Yes |       1 | cp1252 West European
|
| latin1_danish_ci               | latin1   |   15 |      | Yes |       1 | cp1252 West European
|
| latin1_general_ci              | latin1   |   48 |      | Yes |       1 | cp1252 West European
|
| latin1_general_cs              | latin1   |   49 |      | Yes |       1 | cp1252 West European
|
| latin1_german1_ci              | latin1   |    5 |      | Yes |       1 | cp1252 West European
|
| latin1_german2_ci              | latin1   |   31 |      | Yes |       2 | cp1252 West European
|
| latin1_nopad_bin               | latin1   | 1071 |      | Yes |       1 |
|
| latin1_spanish_ci              | latin1   |   94 |      | Yes |       1 | cp1252 West European
|
| latin1_swedish_ci              | latin1   |    8 | Yes  | Yes |       1 | cp1252 West European
|
| latin1_swedish_nopad_ci        | latin1   | 1032 |      | Yes |       1 |
|
| latin2_bin                     | latin2   |   77 |      | Yes |       1 | ISO 8859-2 Central European
|
| latin2_croatian_ci             | latin2   |   27 |      | Yes |       1 | ISO 8859-2 Central European
|
| latin2_czech_cs                | latin2   |    2 |      | Yes |       4 | ISO 8859-2 Central European
|
| latin2_general_ci              | latin2   |    9 | Yes  | Yes |       1 | ISO 8859-2 Central European
|
| latin2_general_nopad_ci        | latin2   | 1033 |      | Yes |       1 | ISO 8859-2 Central European
|
| latin2_hungarian_ci            | latin2   |   21 |      | Yes |       1 | ISO 8859-2 Central European
|
| latin2_nopad_bin               | latin2   | 1101 |      | Yes |       1 | ISO 8859-2 Central European
|
| latin5_bin                     | latin5   |   78 |      | Yes |       1 | ISO 8859-9 Turkish
|
| latin5_nopad_bin               | latin5   | 1102 |      | Yes |       1 | ISO 8859-9 Turkish
|
| latin5_turkish_ci              | latin5   |   30 | Yes  | Yes |       1 | ISO 8859-9 Turkish
|
| latin5_turkish_nopad_ci        | latin5   | 1054 |      | Yes |       1 | ISO 8859-9 Turkish
|
| latin7_bin                     | latin7   |   79 |      | Yes |       1 | ISO 8859-13 Baltic
|
| latin7_estonian_cs             | latin7   |   20 |      | Yes |       1 | ISO 8859-13 Baltic
|
| latin7_general_ci              | latin7   |   41 | Yes  | Yes |       1 | ISO 8859-13 Baltic
|
| latin7_general_cs              | latin7   |   42 |      | Yes |       1 | ISO 8859-13 Baltic
|
| latin7_general_nopad_ci        | latin7   | 1065 |      | Yes |       1 | ISO 8859-13 Baltic
|
| latin7_nopad_bin               | latin7   | 1103 |      | Yes |       1 | ISO 8859-13 Baltic
|
| macce_bin                      | macce    |   43 |      | Yes |       1 | Mac Central European
|
| macce_general_ci               | macce    |   38 | Yes  | Yes |       1 | Mac Central European
|
| macce_general_nopad_ci         | macce    | 1062 |      | Yes |       1 | Mac Central European
|
| macce_nopad_bin                | macce    | 1067 |      | Yes |       1 | Mac Central European
|
| macroman_bin                   | macroman |   53 |      | Yes |       1 | Mac West European
|
| macroman_general_ci            | macroman |   39 | Yes  | Yes |       1 | Mac West European
|
| macroman_general_nopad_ci      | macroman | 1063 |      | Yes |       1 | Mac West European
```

```
|
| macroman_nopad_bin             | macroman | 1077 |      | Yes |       1 | Mac West European
|
| sjis_bin                       | sjis     |   88 |      | Yes |       1 | Shift-JIS Japanese
|
| sjis_japanese_ci               | sjis     |   13 | Yes  | Yes |       1 | Shift-JIS Japanese
|
| sjis_japanese_nopad_ci         | sjis     | 1037 |      | Yes |       1 |
|
| sjis_nopad_bin                 | sjis     | 1112 |      | Yes |       1 |
|
| swe7_bin                       | swe7     |   82 |      | Yes |       1 | 7bit Swedish
|
| swe7_nopad_bin                 | swe7     | 1106 |      | Yes |       1 | 7bit Swedish
|
| swe7_swedish_ci                | swe7     |   10 | Yes  | Yes |       1 | 7bit Swedish
|
| swe7_swedish_nopad_ci          | swe7     | 1034 |      | Yes |       1 | 7bit Swedish
|
| tis620_bin                     | tis620   |   89 |      | Yes |       1 | TIS620 Thai
|
| tis620_nopad_bin               | tis620   | 1113 |      | Yes |       1 |
|
| tis620_thai_ci                 | tis620   |   18 | Yes  | Yes |       4 | TIS620 Thai
|
| tis620_thai_nopad_ci           | tis620   | 1042 |      | Yes |       4 |
|
| uca1400_ai_ci                  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_ai_cs                  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_as_ci                  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_as_cs                  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_croatian_ai_ci         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_croatian_ai_cs         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_croatian_as_ci         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_croatian_as_cs         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_croatian_nopad_ai_ci   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_croatian_nopad_ai_cs   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_croatian_nopad_as_ci   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_croatian_nopad_as_cs   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_czech_ai_ci            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_czech_ai_cs            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_czech_as_ci            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_czech_as_cs            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_czech_nopad_ai_ci      | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_czech_nopad_ai_cs      | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_czech_nopad_as_ci      | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_czech_nopad_as_cs      | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_danish_ai_ci           | NULL     | NULL | NULL | Yes |       8 |
|
```

```
| uca1400_danish_ai_cs           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_danish_as_ci           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_danish_as_cs           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_danish_nopad_ai_ci     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_danish_nopad_ai_cs     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_danish_nopad_as_ci     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_danish_nopad_as_cs     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_esperanto_ai_ci        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_esperanto_ai_cs        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_esperanto_as_ci        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_esperanto_as_cs        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_esperanto_nopad_ai_ci  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_esperanto_nopad_ai_cs  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_esperanto_nopad_as_ci  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_esperanto_nopad_as_cs  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_estonian_ai_ci         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_estonian_ai_cs         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_estonian_as_ci         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_estonian_as_cs         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_estonian_nopad_ai_ci   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_estonian_nopad_ai_cs   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_estonian_nopad_as_ci   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_estonian_nopad_as_cs   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_german2_ai_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_german2_ai_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_german2_as_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_german2_as_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_german2_nopad_ai_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_german2_nopad_ai_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_german2_nopad_as_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_german2_nopad_as_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_hungarian_ai_ci        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_hungarian_ai_cs        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_hungarian_as_ci        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_hungarian_as_cs        | NULL     | NULL | NULL | Yes |       8 |
```

```
|
| uca1400_hungarian_nopad_ai_ci  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_hungarian_nopad_ai_cs  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_hungarian_nopad_as_ci  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_hungarian_nopad_as_cs  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_icelandic_ai_ci        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_icelandic_ai_cs        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_icelandic_as_ci        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_icelandic_as_cs        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_icelandic_nopad_ai_ci  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_icelandic_nopad_ai_cs  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_icelandic_nopad_as_ci  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_icelandic_nopad_as_cs  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_latvian_ai_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_latvian_ai_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_latvian_as_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_latvian_as_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_latvian_nopad_ai_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_latvian_nopad_ai_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_latvian_nopad_as_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_latvian_nopad_as_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_lithuanian_ai_ci       | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_lithuanian_ai_cs       | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_lithuanian_as_ci       | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_lithuanian_as_cs       | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_lithuanian_nopad_ai_ci | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_lithuanian_nopad_ai_cs | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_lithuanian_nopad_as_ci | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_lithuanian_nopad_as_cs | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_nopad_ai_ci            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_nopad_ai_cs            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_nopad_as_ci            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_nopad_as_cs            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_persian_ai_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_persian_ai_cs          | NULL     | NULL | NULL | Yes |       8 |
|
```

```
| uca1400_persian_as_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_persian_as_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_persian_nopad_ai_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_persian_nopad_ai_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_persian_nopad_as_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_persian_nopad_as_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_polish_ai_ci           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_polish_ai_cs           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_polish_as_ci           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_polish_as_cs           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_polish_nopad_ai_ci     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_polish_nopad_ai_cs     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_polish_nopad_as_ci     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_polish_nopad_as_cs     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_romanian_ai_ci         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_romanian_ai_cs         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_romanian_as_ci         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_romanian_as_cs         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_romanian_nopad_ai_ci   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_romanian_nopad_ai_cs   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_romanian_nopad_as_ci   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_romanian_nopad_as_cs   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_roman_ai_ci            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_roman_ai_cs            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_roman_as_ci            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_roman_as_cs            | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_roman_nopad_ai_ci      | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_roman_nopad_ai_cs      | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_roman_nopad_as_ci      | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_roman_nopad_as_cs      | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_sinhala_ai_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_sinhala_ai_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_sinhala_as_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_sinhala_as_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_sinhala_nopad_ai_ci    | NULL     | NULL | NULL | Yes |       8 |
```

```
|
| uca1400_sinhala_nopad_ai_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_sinhala_nopad_as_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_sinhala_nopad_as_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovak_ai_ci           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovak_ai_cs           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovak_as_ci           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovak_as_cs           | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovak_nopad_ai_ci     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovak_nopad_ai_cs     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovak_nopad_as_ci     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovak_nopad_as_cs     | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovenian_ai_ci        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovenian_ai_cs        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovenian_as_ci        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovenian_as_cs        | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovenian_nopad_ai_ci  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovenian_nopad_ai_cs  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovenian_nopad_as_ci  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_slovenian_nopad_as_cs  | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish2_ai_ci         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish2_ai_cs         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish2_as_ci         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish2_as_cs         | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish2_nopad_ai_ci   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish2_nopad_ai_cs   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish2_nopad_as_ci   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish2_nopad_as_cs   | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish_ai_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish_ai_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish_as_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish_as_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish_nopad_ai_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish_nopad_ai_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_spanish_nopad_as_ci    | NULL     | NULL | NULL | Yes |       8 |
|
```

```
| uca1400_spanish_nopad_as_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_swedish_ai_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_swedish_ai_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_swedish_as_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_swedish_as_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_swedish_nopad_ai_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_swedish_nopad_ai_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_swedish_nopad_as_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_swedish_nopad_as_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_turkish_ai_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_turkish_ai_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_turkish_as_ci          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_turkish_as_cs          | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_turkish_nopad_ai_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_turkish_nopad_ai_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_turkish_nopad_as_ci    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_turkish_nopad_as_cs    | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_vietnamese_ai_ci       | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_vietnamese_ai_cs       | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_vietnamese_as_ci       | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_vietnamese_as_cs       | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_vietnamese_nopad_ai_ci | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_vietnamese_nopad_ai_cs | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_vietnamese_nopad_as_ci | NULL     | NULL | NULL | Yes |       8 |
|
| uca1400_vietnamese_nopad_as_cs | NULL     | NULL | NULL | Yes |       8 |
|
| ucs2_bin                       | ucs2     |   90 |      | Yes |       1 | UCS-2 Unicode
|
| ucs2_croatian_ci               | ucs2     |  640 |      | Yes |       8 |
|
| ucs2_croatian_mysql561_ci      | ucs2     |  149 |      | Yes |       8 |
|
| ucs2_czech_ci                  | ucs2     |  138 |      | Yes |       8 |
|
| ucs2_danish_ci                 | ucs2     |  139 |      | Yes |       8 |
|
| ucs2_esperanto_ci              | ucs2     |  145 |      | Yes |       8 |
|
| ucs2_estonian_ci               | ucs2     |  134 |      | Yes |       8 |
|
| ucs2_general_ci                | ucs2     |   35 | Yes  | Yes |       1 | UCS-2 Unicode
|
| ucs2_general_mysql500_ci       | ucs2     |  159 |      | Yes |       1 |
|
| ucs2_general_nopad_ci          | ucs2     | 1059 |      | Yes |       1 |
```

```
|
| ucs2_german2_ci               | ucs2     |  148 |      | Yes |       8 |
|
| ucs2_hungarian_ci             | ucs2     |  146 |      | Yes |       8 |
|
| ucs2_icelandic_ci             | ucs2     |  129 |      | Yes |       8 |
|
| ucs2_latvian_ci               | ucs2     |  130 |      | Yes |       8 |
|
| ucs2_lithuanian_ci            | ucs2     |  140 |      | Yes |       8 |
|
| ucs2_myanmar_ci               | ucs2     |  641 |      | Yes |       8 |
|
| ucs2_nopad_bin                | ucs2     | 1114 |      | Yes |       1 |
|
| ucs2_persian_ci               | ucs2     |  144 |      | Yes |       8 |
|
| ucs2_polish_ci                | ucs2     |  133 |      | Yes |       8 |
|
| ucs2_romanian_ci              | ucs2     |  131 |      | Yes |       8 |
|
| ucs2_roman_ci                 | ucs2     |  143 |      | Yes |       8 |
|
| ucs2_sinhala_ci               | ucs2     |  147 |      | Yes |       8 |
|
| ucs2_slovak_ci                | ucs2     |  141 |      | Yes |       8 |
|
| ucs2_slovenian_ci             | ucs2     |  132 |      | Yes |       8 |
|
| ucs2_spanish2_ci              | ucs2     |  142 |      | Yes |       8 |
|
| ucs2_spanish_ci               | ucs2     |  135 |      | Yes |       8 |
|
| ucs2_swedish_ci               | ucs2     |  136 |      | Yes |       8 |
|
| ucs2_thai_520_w2              | ucs2     |  642 |      | Yes |       4 |
|
| ucs2_turkish_ci               | ucs2     |  137 |      | Yes |       8 |
|
| ucs2_unicode_520_ci           | ucs2     |  150 |      | Yes |       8 |
|
| ucs2_unicode_520_nopad_ci     | ucs2     | 1174 |      | Yes |       8 |
|
| ucs2_unicode_ci               | ucs2     |  128 |      | Yes |       8 |
|
| ucs2_unicode_nopad_ci         | ucs2     | 1152 |      | Yes |       8 |
|
| ucs2_vietnamese_ci            | ucs2     |  151 |      | Yes |       8 |
|
| ujis_bin                      | ujis     |   91 |      | Yes |       1 | EUC-JP Japanese
|
| ujis_japanese_ci              | ujis     |   12 | Yes  | Yes |       1 | EUC-JP Japanese
|
| ujis_japanese_nopad_ci        | ujis     | 1036 |      | Yes |       1 |
|
| ujis_nopad_bin                | ujis     | 1115 |      | Yes |       1 |
|
| utf16le_bin                   | utf16le  |   62 |      | Yes |       1 | UTF-16LE Unicode
|
| utf16le_general_ci            | utf16le  |   56 | Yes  | Yes |       1 | UTF-16LE Unicode
|
| utf16le_general_nopad_ci      | utf16le  | 1080 |      | Yes |       1 | UTF-16LE Unicode
|
| utf16le_nopad_bin             | utf16le  | 1086 |      | Yes |       1 | UTF-16LE Unicode
|
| utf16_bin                     | utf16    |   55 |      | Yes |       1 | UTF-16 Unicode
|
| utf16_croatian_ci             | utf16    |  672 |      | Yes |       8 |
|
```

```
| utf16_croatian_mysql561_ci     | utf16    |  122 |      | Yes |       8 |
|
| utf16_czech_ci                 | utf16    |  111 |      | Yes |       8 |
|
| utf16_danish_ci                | utf16    |  112 |      | Yes |       8 |
|
| utf16_esperanto_ci             | utf16    |  118 |      | Yes |       8 |
|
| utf16_estonian_ci              | utf16    |  107 |      | Yes |       8 |
|
| utf16_general_ci               | utf16    |   54 | Yes  | Yes |       1 | UTF-16 Unicode
|
| utf16_general_nopad_ci         | utf16    | 1078 |      | Yes |       1 | UTF-16 Unicode
|
| utf16_german2_ci               | utf16    |  121 |      | Yes |       8 |
|
| utf16_hungarian_ci             | utf16    |  119 |      | Yes |       8 |
|
| utf16_icelandic_ci             | utf16    |  102 |      | Yes |       8 |
|
| utf16_latvian_ci               | utf16    |  103 |      | Yes |       8 |
|
| utf16_lithuanian_ci            | utf16    |  113 |      | Yes |       8 |
|
| utf16_myanmar_ci               | utf16    |  673 |      | Yes |       8 |
|
| utf16_nopad_bin                | utf16    | 1079 |      | Yes |       1 | UTF-16 Unicode
|
| utf16_persian_ci               | utf16    |  117 |      | Yes |       8 |
|
| utf16_polish_ci                | utf16    |  106 |      | Yes |       8 |
|
| utf16_romanian_ci              | utf16    |  104 |      | Yes |       8 |
|
| utf16_roman_ci                 | utf16    |  116 |      | Yes |       8 |
|
| utf16_sinhala_ci               | utf16    |  120 |      | Yes |       8 |
|
| utf16_slovak_ci                | utf16    |  114 |      | Yes |       8 |
|
| utf16_slovenian_ci             | utf16    |  105 |      | Yes |       8 |
|
| utf16_spanish2_ci              | utf16    |  115 |      | Yes |       8 |
|
| utf16_spanish_ci               | utf16    |  108 |      | Yes |       8 |
|
| utf16_swedish_ci               | utf16    |  109 |      | Yes |       8 |
|
| utf16_thai_520_w2              | utf16    |  674 |      | Yes |       4 |
|
| utf16_turkish_ci               | utf16    |  110 |      | Yes |       8 |
|
| utf16_unicode_520_ci           | utf16    |  123 |      | Yes |       8 |
|
| utf16_unicode_520_nopad_ci     | utf16    | 1147 |      | Yes |       8 |
|
| utf16_unicode_ci               | utf16    |  101 |      | Yes |       8 |
|
| utf16_unicode_nopad_ci         | utf16    | 1125 |      | Yes |       8 |
|
| utf16_vietnamese_ci            | utf16    |  124 |      | Yes |       8 |
|
| utf32_bin                      | utf32    |   61 |      | Yes |       1 | UTF-32 Unicode
|
| utf32_croatian_ci              | utf32    |  736 |      | Yes |       8 |
|
| utf32_croatian_mysql561_ci     | utf32    |  181 |      | Yes |       8 |
|
| utf32_czech_ci                 | utf32    |  170 |      | Yes |       8 |
```

```
|
| utf32_danish_ci                | utf32    |  171 |      | Yes |       8 |
|
| utf32_esperanto_ci             | utf32    |  177 |      | Yes |       8 |
|
| utf32_estonian_ci              | utf32    |  166 |      | Yes |       8 |
|
| utf32_general_ci               | utf32    |   60 | Yes  | Yes |       1 | UTF-32 Unicode
|
| utf32_general_nopad_ci         | utf32    | 1084 |      | Yes |       1 | UTF-32 Unicode
|
| utf32_german2_ci               | utf32    |  180 |      | Yes |       8 |
|
| utf32_hungarian_ci             | utf32    |  178 |      | Yes |       8 |
|
| utf32_icelandic_ci             | utf32    |  161 |      | Yes |       8 |
|
| utf32_latvian_ci               | utf32    |  162 |      | Yes |       8 |
|
| utf32_lithuanian_ci            | utf32    |  172 |      | Yes |       8 |
|
| utf32_myanmar_ci               | utf32    |  737 |      | Yes |       8 |
|
| utf32_nopad_bin                | utf32    | 1085 |      | Yes |       1 | UTF-32 Unicode
|
| utf32_persian_ci               | utf32    |  176 |      | Yes |       8 |
|
| utf32_polish_ci                | utf32    |  165 |      | Yes |       8 |
|
| utf32_romanian_ci              | utf32    |  163 |      | Yes |       8 |
|
| utf32_roman_ci                 | utf32    |  175 |      | Yes |       8 |
|
| utf32_sinhala_ci               | utf32    |  179 |      | Yes |       8 |
|
| utf32_slovak_ci                | utf32    |  173 |      | Yes |       8 |
|
| utf32_slovenian_ci             | utf32    |  164 |      | Yes |       8 |
|
| utf32_spanish2_ci              | utf32    |  174 |      | Yes |       8 |
|
| utf32_spanish_ci               | utf32    |  167 |      | Yes |       8 |
|
| utf32_swedish_ci               | utf32    |  168 |      | Yes |       8 |
|
| utf32_thai_520_w2              | utf32    |  738 |      | Yes |       4 |
|
| utf32_turkish_ci               | utf32    |  169 |      | Yes |       8 |
|
| utf32_unicode_520_ci           | utf32    |  182 |      | Yes |       8 |
|
| utf32_unicode_520_nopad_ci     | utf32    | 1206 |      | Yes |       8 |
|
| utf32_unicode_ci               | utf32    |  160 |      | Yes |       8 |
|
| utf32_unicode_nopad_ci         | utf32    | 1184 |      | Yes |       8 |
|
| utf32_vietnamese_ci            | utf32    |  183 |      | Yes |       8 |
|
| utf8mb3_bin                    | utf8mb3  |   83 |      | Yes |       1 | UTF-8 Unicode
|
| utf8mb3_croatian_ci            | utf8mb3  |  576 |      | Yes |       8 |
|
| utf8mb3_croatian_mysql561_ci   | utf8mb3  |  213 |      | Yes |       8 |
|
| utf8mb3_czech_ci               | utf8mb3  |  202 |      | Yes |       8 |
|
| utf8mb3_danish_ci              | utf8mb3  |  203 |      | Yes |       8 |
|
```

```
| utf8mb3_esperanto_ci           | utf8mb3  |  209 |      | Yes |       8 |
|
| utf8mb3_estonian_ci            | utf8mb3  |  198 |      | Yes |       8 |
|
| utf8mb3_general_ci             | utf8mb3  |   33 | Yes  | Yes |       1 | UTF-8 Unicode
|
| utf8mb3_general_mysql500_ci    | utf8mb3  |  223 |      | Yes |       1 |
|
| utf8mb3_general_nopad_ci       | utf8mb3  | 1057 |      | Yes |       1 |
|
| utf8mb3_german2_ci             | utf8mb3  |  212 |      | Yes |       8 |
|
| utf8mb3_hungarian_ci           | utf8mb3  |  210 |      | Yes |       8 |
|
| utf8mb3_icelandic_ci           | utf8mb3  |  193 |      | Yes |       8 |
|
| utf8mb3_latvian_ci             | utf8mb3  |  194 |      | Yes |       8 |
|
| utf8mb3_lithuanian_ci          | utf8mb3  |  204 |      | Yes |       8 |
|
| utf8mb3_myanmar_ci             | utf8mb3  |  577 |      | Yes |       8 |
|
| utf8mb3_nopad_bin              | utf8mb3  | 1107 |      | Yes |       1 |
|
| utf8mb3_persian_ci             | utf8mb3  |  208 |      | Yes |       8 |
|
| utf8mb3_polish_ci              | utf8mb3  |  197 |      | Yes |       8 |
|
| utf8mb3_romanian_ci            | utf8mb3  |  195 |      | Yes |       8 |
|
| utf8mb3_roman_ci               | utf8mb3  |  207 |      | Yes |       8 |
|
| utf8mb3_sinhala_ci             | utf8mb3  |  211 |      | Yes |       8 |
|
| utf8mb3_slovak_ci              | utf8mb3  |  205 |      | Yes |       8 |
|
| utf8mb3_slovenian_ci           | utf8mb3  |  196 |      | Yes |       8 |
|
| utf8mb3_spanish2_ci            | utf8mb3  |  206 |      | Yes |       8 |
|
| utf8mb3_spanish_ci             | utf8mb3  |  199 |      | Yes |       8 |
|
| utf8mb3_swedish_ci             | utf8mb3  |  200 |      | Yes |       8 |
|
| utf8mb3_thai_520_w2            | utf8mb3  |  578 |      | Yes |       4 |
|
| utf8mb3_turkish_ci             | utf8mb3  |  201 |      | Yes |       8 |
|
| utf8mb3_unicode_520_ci         | utf8mb3  |  214 |      | Yes |       8 |
|
| utf8mb3_unicode_520_nopad_ci   | utf8mb3  | 1238 |      | Yes |       8 |
|
| utf8mb3_unicode_ci             | utf8mb3  |  192 |      | Yes |       8 |
|
| utf8mb3_unicode_nopad_ci       | utf8mb3  | 1216 |      | Yes |       8 |
|
| utf8mb3_vietnamese_ci          | utf8mb3  |  215 |      | Yes |       8 |
|
| utf8mb4_0900_ai_ci             | utf8mb4  |  255 |      | Yes |       8 | Alias for
utf8mb4_uca1400_nopad_ai_ci            |
| utf8mb4_0900_as_ci             | utf8mb4  |  305 |      | Yes |       8 | Alias for
utf8mb4_uca1400_nopad_as_ci            |
| utf8mb4_0900_as_cs             | utf8mb4  |  278 |      | Yes |       8 | Alias for
utf8mb4_uca1400_nopad_as_cs            |
| utf8mb4_0900_bin               | utf8mb4  |  309 |      | Yes |       1 | Alias for utf8mb4_bin
|
| utf8mb4_bin                    | utf8mb4  |   46 |      | Yes |       1 | UTF-8 Unicode
|
| utf8mb4_croatian_ci            | utf8mb4  |  608 |      | Yes |       8 |
```

```
|
| utf8mb4_croatian_mysql561_ci   | utf8mb4  |  245 |      | Yes |       8 |
|
| utf8mb4_cs_0900_ai_ci          | utf8mb4  |  266 |      | Yes |       8 | Alias for
utf8mb4_uca1400_czech_nopad_ai_ci      |
| utf8mb4_cs_0900_as_cs          | utf8mb4  |  289 |      | Yes |       8 | Alias for
utf8mb4_uca1400_czech_nopad_as_cs      |
| utf8mb4_czech_ci               | utf8mb4  |  234 |      | Yes |       8 |
|
| utf8mb4_danish_ci              | utf8mb4  |  235 |      | Yes |       8 |
|
| utf8mb4_da_0900_ai_ci          | utf8mb4  |  267 |      | Yes |       8 | Alias for
utf8mb4_uca1400_danish_nopad_ai_ci     |
| utf8mb4_da_0900_as_cs          | utf8mb4  |  290 |      | Yes |       8 | Alias for
utf8mb4_uca1400_danish_nopad_as_cs     |
| utf8mb4_de_pb_0900_ai_ci       | utf8mb4  |  256 |      | Yes |       8 | Alias for
utf8mb4_uca1400_german2_nopad_ai_ci    |
| utf8mb4_de_pb_0900_as_cs       | utf8mb4  |  279 |      | Yes |       8 | Alias for
utf8mb4_uca1400_german2_nopad_as_cs    |
| utf8mb4_eo_0900_ai_ci          | utf8mb4  |  273 |      | Yes |       8 | Alias for
utf8mb4_uca1400_esperanto_nopad_ai_ci  |
| utf8mb4_eo_0900_as_cs          | utf8mb4  |  296 |      | Yes |       8 | Alias for
utf8mb4_uca1400_esperanto_nopad_as_cs  |
| utf8mb4_esperanto_ci           | utf8mb4  |  241 |      | Yes |       8 |
|
| utf8mb4_estonian_ci            | utf8mb4  |  230 |      | Yes |       8 |
|
| utf8mb4_es_0900_ai_ci          | utf8mb4  |  263 |      | Yes |       8 | Alias for
utf8mb4_uca1400_spanish_nopad_ai_ci    |
| utf8mb4_es_0900_as_cs          | utf8mb4  |  286 |      | Yes |       8 | Alias for
utf8mb4_uca1400_spanish_nopad_as_cs    |
| utf8mb4_es_trad_0900_ai_ci     | utf8mb4  |  270 |      | Yes |       8 | Alias for
utf8mb4_uca1400_spanish2_nopad_ai_ci   |
| utf8mb4_es_trad_0900_as_cs     | utf8mb4  |  293 |      | Yes |       8 | Alias for
utf8mb4_uca1400_spanish2_nopad_as_cs   |
| utf8mb4_et_0900_ai_ci          | utf8mb4  |  262 |      | Yes |       8 | Alias for
utf8mb4_uca1400_estonian_nopad_ai_ci   |
| utf8mb4_et_0900_as_cs          | utf8mb4  |  285 |      | Yes |       8 | Alias for
utf8mb4_uca1400_estonian_nopad_as_cs   |
| utf8mb4_general_ci             | utf8mb4  |   45 | Yes  | Yes |       1 | UTF-8 Unicode
|
| utf8mb4_general_nopad_ci       | utf8mb4  | 1069 |      | Yes |       1 | UTF-8 Unicode
|
| utf8mb4_german2_ci             | utf8mb4  |  244 |      | Yes |       8 |
|
| utf8mb4_hr_0900_ai_ci          | utf8mb4  |  275 |      | Yes |       8 | Alias for
utf8mb4_uca1400_croatian_nopad_ai_ci   |
| utf8mb4_hr_0900_as_cs          | utf8mb4  |  298 |      | Yes |       8 | Alias for
utf8mb4_uca1400_croatian_nopad_as_cs   |
| utf8mb4_hungarian_ci           | utf8mb4  |  242 |      | Yes |       8 |
|
| utf8mb4_hu_0900_ai_ci          | utf8mb4  |  274 |      | Yes |       8 | Alias for
utf8mb4_uca1400_hungarian_nopad_ai_ci  |
| utf8mb4_hu_0900_as_cs          | utf8mb4  |  297 |      | Yes |       8 | Alias for
utf8mb4_uca1400_hungarian_nopad_as_cs  |
| utf8mb4_icelandic_ci           | utf8mb4  |  225 |      | Yes |       8 |
|
| utf8mb4_is_0900_ai_ci          | utf8mb4  |  257 |      | Yes |       8 | Alias for
utf8mb4_uca1400_icelandic_nopad_ai_ci  |
| utf8mb4_is_0900_as_cs          | utf8mb4  |  280 |      | Yes |       8 | Alias for
utf8mb4_uca1400_icelandic_nopad_as_cs  |
| utf8mb4_latvian_ci             | utf8mb4  |  226 |      | Yes |       8 |
|
| utf8mb4_la_0900_ai_ci          | utf8mb4  |  271 |      | Yes |       8 | Alias for
utf8mb4_uca1400_roman_nopad_ai_ci      |
| utf8mb4_la_0900_as_cs          | utf8mb4  |  294 |      | Yes |       8 | Alias for
utf8mb4_uca1400_roman_nopad_as_cs      |
| utf8mb4_lithuanian_ci          | utf8mb4  |  236 |      | Yes |       8 |
|
```

```
| utf8mb4_lt_0900_ai_ci          | utf8mb4  |  268 |      | Yes |       8 | Alias for
utf8mb4_uca1400_lithuanian_nopad_ai_ci |
| utf8mb4_lt_0900_as_cs          | utf8mb4  |  291 |      | Yes |       8 | Alias for
utf8mb4_uca1400_lithuanian_nopad_as_cs |
| utf8mb4_lv_0900_ai_ci          | utf8mb4  |  258 |      | Yes |       8 | Alias for
utf8mb4_uca1400_latvian_nopad_ai_ci    |
| utf8mb4_lv_0900_as_cs          | utf8mb4  |  281 |      | Yes |       8 | Alias for
utf8mb4_uca1400_latvian_nopad_as_cs    |
| utf8mb4_myanmar_ci             | utf8mb4  |  609 |      | Yes |       8 |
|
| utf8mb4_nopad_bin              | utf8mb4  | 1070 |      | Yes |       1 | UTF-8 Unicode
|
| utf8mb4_persian_ci             | utf8mb4  |  240 |      | Yes |       8 |
|
| utf8mb4_pl_0900_ai_ci          | utf8mb4  |  261 |      | Yes |       8 | Alias for
utf8mb4_uca1400_polish_nopad_ai_ci     |
| utf8mb4_pl_0900_as_cs          | utf8mb4  |  284 |      | Yes |       8 | Alias for
utf8mb4_uca1400_polish_nopad_as_cs     |
| utf8mb4_polish_ci              | utf8mb4  |  229 |      | Yes |       8 |
|
| utf8mb4_romanian_ci            | utf8mb4  |  227 |      | Yes |       8 |
|
| utf8mb4_roman_ci               | utf8mb4  |  239 |      | Yes |       8 |
|
| utf8mb4_ro_0900_ai_ci          | utf8mb4  |  259 |      | Yes |       8 | Alias for
utf8mb4_uca1400_romanian_nopad_ai_ci   |
| utf8mb4_ro_0900_as_cs          | utf8mb4  |  282 |      | Yes |       8 | Alias for
utf8mb4_uca1400_romanian_nopad_as_cs   |
| utf8mb4_sinhala_ci             | utf8mb4  |  243 |      | Yes |       8 |
|
| utf8mb4_sk_0900_ai_ci          | utf8mb4  |  269 |      | Yes |       8 | Alias for
utf8mb4_uca1400_slovak_nopad_ai_ci     |
| utf8mb4_sk_0900_as_cs          | utf8mb4  |  292 |      | Yes |       8 | Alias for
utf8mb4_uca1400_slovak_nopad_as_cs     |
| utf8mb4_slovak_ci              | utf8mb4  |  237 |      | Yes |       8 |
|
| utf8mb4_slovenian_ci           | utf8mb4  |  228 |      | Yes |       8 |
|
| utf8mb4_sl_0900_ai_ci          | utf8mb4  |  260 |      | Yes |       8 | Alias for
utf8mb4_uca1400_slovenian_nopad_ai_ci  |
| utf8mb4_sl_0900_as_cs          | utf8mb4  |  283 |      | Yes |       8 | Alias for
utf8mb4_uca1400_slovenian_nopad_as_cs  |
| utf8mb4_spanish2_ci            | utf8mb4  |  238 |      | Yes |       8 |
|
| utf8mb4_spanish_ci             | utf8mb4  |  231 |      | Yes |       8 |
|
| utf8mb4_sv_0900_ai_ci          | utf8mb4  |  264 |      | Yes |       8 | Alias for
utf8mb4_uca1400_swedish_nopad_ai_ci    |
| utf8mb4_sv_0900_as_cs          | utf8mb4  |  287 |      | Yes |       8 | Alias for
utf8mb4_uca1400_swedish_nopad_as_cs    |
| utf8mb4_swedish_ci             | utf8mb4  |  232 |      | Yes |       8 |
|
| utf8mb4_thai_520_w2            | utf8mb4  |  610 |      | Yes |       4 |
|
| utf8mb4_tr_0900_ai_ci          | utf8mb4  |  265 |      | Yes |       8 | Alias for
utf8mb4_uca1400_turkish_nopad_ai_ci    |
| utf8mb4_tr_0900_as_cs          | utf8mb4  |  288 |      | Yes |       8 | Alias for
utf8mb4_uca1400_turkish_nopad_as_cs    |
| utf8mb4_turkish_ci             | utf8mb4  |  233 |      | Yes |       8 |
|
| utf8mb4_unicode_520_ci         | utf8mb4  |  246 |      | Yes |       8 |
|
| utf8mb4_unicode_520_nopad_ci   | utf8mb4  | 1270 |      | Yes |       8 |
|
| utf8mb4_unicode_ci             | utf8mb4  |  224 |      | Yes |       8 |
|
| utf8mb4_unicode_nopad_ci       | utf8mb4  | 1248 |      | Yes |       8 |
|
| utf8mb4_vietnamese_ci          | utf8mb4  |  247 |      | Yes |       8 |
```

```
|
| utf8mb4_vi_0900_ai_ci          | utf8mb4  |  277 |      | Yes |       8 | Alias for
utf8mb4_uca1400_vietnamese_nopad_ai_ci |
| utf8mb4_vi_0900_as_cs          | utf8mb4  |  300 |      | Yes |       8 | Alias for
utf8mb4_uca1400_vietnamese_nopad_as_cs |
+--------------------------------+----------+------+------+-----+---------+-----------------------------
--------------------+
550 rows in set (0.005 sec)
```

:information_source: The UCA-14.0.0 collations were added in MariaDB 10.10.1.

:information_source: Before MariaDB 10.6.1, the `utf8mb3*` collations listed above were named `utf8*`.

:information_source: MariaDB 11.4.5 added UCA-9.0.0 collations, as well as a Comment column to `information_schema.collations`, to make it clear that the UCA-9.0.0 collations are mapped to the UCA-14.0.0 collations. The UCA-9.0.0 collations have mainly been added to make it easy to replicate from MySQL 8.0 to MariaDB 11.4.5 and newer.

> **Note:**
>
> Note that some of the collations are used with several different character sets. In this case, the `Charset` and `Id` columns are `NULL`.

You can find all combinations of the supported character set and collation in the information_schema.COLLATION_CHARACTER_SET_APPLICABILITY table.

Case Sensitivity

A '`ci`' at the end of a collation name indicates the collation is case-insensitive. A '`cs`' at the end of a collation name indicates the collation is case sensitive.

NO PAD Collations

`NO PAD` collations regard trailing spaces as normal characters. You can get a list of all of these by querying the Information Schema COLLATIONS Table as follows:

```
SELECT collation_name FROM information_schema.COLLATIONS
WHERE collation_name LIKE "%nopad%";
+------------------------------+
| collation_name               |
+------------------------------+
| big5_chinese_nopad_ci        |
| big5_nopad_bin               |
...
```

Accent Insensitivity

An accent insensitive collation is one where the accented and unaccented versions of a letter are considered to be identical for sorting purposes.

:information_source: MariaDB 10.10 added the accent insensitivity flag, and new collations are marked with '_ai' or '_as' in the name to indicate this, for example:

```
...
| uca1400_spanish2_ai_ci         |
| uca1400_spanish2_ai_cs         |
| uca1400_spanish2_as_ci         |
| uca1400_spanish2_as_cs         |
...
```

Changes

- MariaDB 10.10 added 184 UCA-14.0.0 collations. Unicode-14.0.0 was released in September 2021. They contain 939 built-in contractions. Old collations based on UCA-4.0.0 and UCA-5.2.0 did not support built-in contractions. This is a step towards better Unicode Collation Algorithm compliance. With built-in contractions, some languages (e.g., Thai) won't need specific collations and will just work with the default "root" collation.
- MariaDB 10.6.1 changed the `utf8` character set by default to be an alias for utf8mb3 rather than the other way around. It can be set to imply `utf8mb4` by changing the value of the old_mode system variable.
- MariaDB 10.2.2 added 88 `NO PAD` collations.
- MariaDB 10.1.15 added the `utf8_thai_520_w2`, `utf8mb4_thai_520_w2`, `ucs2_thai_520_w2`, `utf16_thai_520_w2` and `utf32_thai_520_w2` collations.
- MariaDB 10.0.7 added the `utf8_myanmar_ci`, `ucs2_myanmar_ci`, `utf8mb4_myanmar_ci`, `utf16_myanmar_ci` and `utf32_myanmar_ci` collations.
- MariaDB 10.0.5 added the `utf8_german2_ci`, `utf8mb4_german2_ci`, `ucs2_german2_ci`, `utf16_german2_ci` and `utf32_german2_ci` collations.
- MariaDB 5.1.41 added a Croatian collation patch from Alexander Barkov to fix some problems with the Croatian character set and `LIKE` queries. This patch added `utf8_croatian_ci` and `ucs2_croatian_ci` collations to MariaDB.

See Also

- Information Schema CHARACTER_SETS Table
- Information Schema COLLATIONS Table

This page is licensed: CC BY-SA / Gnu FDL

Unicode

Unicode is a standard for encoding text across multiple writing systems. MariaDB supports a number of character sets for storing Unicode data:

Character Set	Description
`ucs2`	UCS-2, each character is represented by a 2-byte code with the most significant byte first. Fixed-length 16-bit encoding.
`utf8`	`utf8` is an alias for `utf8mb3`, but this can changed to `ut8mb4` by changing the default value of the old_mode system variable.
`utf8mb3`	UTF-8 encoding using one to three bytes per character. Basic Latin letters, numbers and punctuation use one byte. European and Middle East letters mostly fit into 2 bytes. Korean, Chinese, and Japanese ideographs use 3-bytes. No supplementary characters are stored. Until MariaDB 10.5, this was an alias for utf8. From MariaDB 10.6, utf8 is by default an alias for utf8mb3, but this can changed to ut8mb4 by changing the default value of the old_mode system variable.
`utf8mb4`	UTF-8 encoding the same as `utf8mb3` but which stores supplementary characters in four bytes.
`utf16`	UTF-16, same as `ucs2`, but stores supplementary characters in 32 bits. 16 or 32-bits.
`utf32`	UTF-32, fixed-length 32-bit encoding.

Support for the UCA-14.0.0 collations was added in MariaDB 10.10 (MDEV-27009).

Support for the MySQL 8.0 UCA-9-0-0 (utf8mb4_0900_…) collations will be added to MariaDB 11.4.5.

This page is licensed: CC BY-SA / Gnu FDL

Internationalization and Localization

See Coordinated Universal Time.

Coordinated Universal Time (UTC) is the primary time standard by which the world regulates clocks and time, and is the internal storage format for MariaDB timestamp values.

See Locales Plugin.

The LOCALES plugin enables the INFORMATION_SCHEMA.LOCALES table and SHOW LOCALES statement, allowing users to view all locales compiled into the server.

See Server Locale.

Server locale settings control the language for date and time functions via lc_time_names and the language for error messages via lc_messages.

See Setting the Language for Error Messages.

Learn how to configure the lc_messages and lc_messages_dir system variables to display server error messages in a supported local language.

See Time Zones.

How MariaDB tracks and converts time zone settings, including named time zones, the time-zone system tables, and per-session configuration.

This page is licensed: CC BY-SA / Gnu FDL

Coordinated Universal Time

UTC stands for Coordinated Universal Time. It is the world standard for regulating time.

MariaDB stores values internally in UTC, converting them to the required time zone as required.

In general terms it is equivalent to Greenwich Mean Time (GMT), but UTC is used in technical contexts, as it is precisely defined at the subsecond level.

Time zones are offset relative to UTC. For example, time in Tonga is UTC + 13, so 03h00 UTC is 16h00 in Tonga.

See Also

- UTC_DATE
- UTC_TIME
- UTC_TIMESTAMP

This page is licensed: CC BY-SA / Gnu FDL

Locales Plugin

The `LOCALES` plugin creates the LOCALES table in the INFORMATION_SCHEMA database. The plugin also adds the SHOW LOCALES statement.The table and statement can be queried to see all locales that are compiled into the server.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN. For example:

```
INSTALL SONAME 'locales';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
...
plugin_load_add = locales
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN. For example:

```
UNINSTALL SONAME 'locales';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Example

```
SELECT * FROM INFORMATION_SCHEMA.LOCALES;
+-----+-------+-----------------------------------+----------------------+--------------------+------
---------+--------------+------------------------+
| ID  | NAME  | DESCRIPTION                       | MAX_MONTH_NAME_LENGTH | MAX_DAY_NAME_LENGTH |
DECIMAL_POINT | THOUSAND_SEP | ERROR_MESSAGE_LANGUAGE |
+-----+-------+-----------------------------------+----------------------+--------------------+------
---------+--------------+------------------------+
|   0 | en_US | English - United States           |                    9 |                  9 | .
| ,            | english                |
|   1 | en_GB | English - United Kingdom          |                    9 |                  9 | .
| ,            | english                |
|   2 | ja_JP | Japanese - Japan                  |                    3 |                  3 | .
| ,            | japanese               |
|   3 | sv_SE | Swedish - Sweden                  |                    9 |                  7 | ,
|              | swedish                |
|   4 | de_DE | German - Germany                  |                    9 |                 10 | ,
| .            | german                 |
|   5 | fr_FR | French - France                   |                    9 |                  8 | ,
|              | french                 |
|   6 | ar_AE | Arabic - United Arab Emirates     |                    6 |                  8 | .
| ,            | english                |
|   7 | ar_BH | Arabic - Bahrain                  |                    6 |                  8 | .
| ,            | english                |
|   8 | ar_JO | Arabic - Jordan                   |                   12 |                  8 | .
| ,            | english                |
...
| 106 | no_NO | Norwegian - Norway                |                    9 |                  7 | ,
| .            | norwegian              |
| 107 | sv_FI | Swedish - Finland                 |                    9 |                  7 | ,
|              | swedish                |
| 108 | zh_HK | Chinese - Hong Kong SAR           |                    3 |                  3 | .
| ,            | english                |
| 109 | el_GR | Greek - Greece                    |                   11 |                  9 | ,
| .            | greek                  |
| 110 | rm_CH | Romansh - Switzerland             |                    9 |                  9 | ,
| .            | english                |
+-----+-------+-----------------------------------+----------------------+--------------------+------
---------+--------------+------------------------+
```

Options

`locales`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server still continues starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server fails to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server fails to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--locales=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

Server Locale

The lc_time_names server system variable sets the language used by the date and time functions DAYNAME(), MONTHNAME(), and DATE_FORMAT(), and the lc_messages sets the language for error messages.

The list of the locales supported by the current MariaDB installation can be obtained via the LOCALES plugin.

MariaDB supports the following locale values:

Locale	Language	Territory
ar_AE	Arabic	United Arab Emirates
ar_BH	Arabic	Bahrain
ar_DZ	Arabic	Algeria
ar_EG	Arabic	Egypt
ar_IN	Arabic	Iran
ar_IQ	Arabic	Iraq
ar_JO	Arabic	Jordan
ar_KW	Arabic	Kuwait
ar_LB	Arabic	Lebanon
ar_LY	Arabic	Libya
ar_MA	Arabic	Morocco
ar_OM	Arabic	Oman
ar_QA	Arabic	Qatar
ar_SA	Arabic	Saudi Arabia
ar_SD	Arabic	Sudan
ar_SY	Arabic	Syria
ar_TN	Arabic	Tunisia

Locale	Language	Territory
ar_YE	Arabic	Yemen
be_BY	Belarusian	Belarus
bg_BG	Bulgarian	Bulgaria
ca_ES	Catalan	Catalan
cs_CZ	Czech	Czech Republic
da_DK	Danish	Denmark
de_AT	German	Austria
de_BE	German	Belgium
de_CH	German	Switzerland
de_DE	German	Germany
de_LU	German	Luxembourg
el_GR	Greek	Greece
en_AU	English	Australia
en_CA	English	Canada
en_GB	English	United Kingdom
en_IN	English	India
en_NZ	English	New Zealand
en_PH	English	Philippines
en_US	English	United States
en_ZA	English	South Africa
en_ZW	English	Zimbabwe
es_AR	Spanish	Argentina
es_BO	Spanish	Bolivia
es_CL	Spanish	Chile
es_CO	Spanish	Columbia
es_CR	Spanish	Costa Rica
es_DO	Spanish	Dominican Republic
es_EC	Spanish	Ecuador
es_ES	Spanish	Spain
es_GT	Spanish	Guatemala
es_HN	Spanish	Honduras
es_MX	Spanish	Mexico
es_NI	Spanish	Nicaragua
es_PA	Spanish	Panama
es_PE	Spanish	Peru
es_PR	Spanish	Puerto Rico
es_PY	Spanish	Paraguay
es_SV	Spanish	El Salvador
es_US	Spanish	United States

Locale	Language	Territory
es_UY	Spanish	Uruguay
es_VE	Spanish	Venezuela
et_EE	Estonian	Estonia
eu_ES	Basque	Basque
fi_FI	Finnish	Finland
fo_FO	Faroese	Faroe Islands
fr_BE	French	Belgium
fr_CA	French	Canada
fr_CH	French	Switzerland
fr_FR	French	France
fr_LU	French	Luxembourg
gl_ES	Galician	Galician
gu_IN	Gujarati	India
he_IL	Hebrew	Israel
hi_IN	Hindi	India
hr_HR	Croatian	Croatia
hu_HU	Hungarian	Hungary
id_ID	Indonesian	Indonesia
is_IS	Icelandic	Iceland
it_CH	Italian	Switzerland
it_IT	Italian	Italy
ja_JP	Japanese	Japan
ka_GE	Georgian	Georgia (From MariaDB 10.11.3)
ko_KR	Korean	Republic of Korea
lt_LT	Lithuanian	Lithuania
lv_LV	Latvian	Latvia
mk_MK	Macedonian	FYROM
mn_MN	Mongolia	Mongolian
ms_MY	Malay	Malaysia
nb_NO	Norwegian(Bokmål)	Norway
nl_BE	Dutch	Belgium
nl_NL	Dutch	The Netherlands
no_NO	Norwegian	Norway
pl_PL	Polish	Poland
pt_BR	Portuguese	Brazil
pt_PT	Portuguese	Portugal
rm_CH	Romansh	Switzerland
ro_RO	Romanian	Romania
ru_RU	Russian	Russia

Locale	Language	Territory
ru_UA	Russian	Ukraine
sk_SK	Slovak	Slovakia
sl_SI	Slovenian	Slovenia
sq_AL	Albanian	Albania
sr_YU	Serbian	Serbia (Deprecated in MariaDB 10.0.11 and removed in MariaDB 10.3.0)
sv_FI	Swedish	Finland
sv_SE	Swedish	Sweden
sw_KE	Swahili	Kenya (from MariaDB 11.1.2)
ta_IN	Tamil	India
te_IN	Telugu	India
th_TH	Thai	Thailand
tr_TR	Turkish	Turkey
uk_UA	Ukrainian	Ukraine
ur_PK	Urdu	Pakistan
vi_VN	Vietnamese	Viet Name
zh_CN	Chinese	China
zh_HK	Chinese	Hong Kong
zh_TW	Chinese	Taiwan Province of China

Examples

Setting the lc_time_names and lc_messages variables to localize the units of date and time, and the server error messages.

```
SELECT DAYNAME('2013-04-01'), MONTHNAME('2013-04-01');
+----------------------+------------------------+
| DAYNAME('2013-04-01') | MONTHNAME('2013-04-01') |
+----------------------+------------------------+
| Monday                | April                  |
+----------------------+------------------------+

SET lc_time_names = 'fr_CA';

SELECT DAYNAME('2013-04-01'), MONTHNAME('2013-04-01');
+----------------------+------------------------+
| DAYNAME('2013-04-01') | MONTHNAME('2013-04-01') |
+----------------------+------------------------+
| lundi                 | avril                  |
+----------------------+------------------------+

SELECT blah;
ERROR 1054 (42S22): Unknown column 'blah' in 'field' list'

SET lc_messages = 'nl_NL';

SELECT blah;
ERROR 1054 (42S22): Onbekende kolom 'blah' in field list
```

This page is licensed: CC BY-SA / Gnu FDL

Setting the Language for Error Messages

MariaDB server error messages are by default in English. However, MariaDB server also supports error message localization in many different languages. Each supported language has its own version of the error message file called `errmsg.sys` in a dedicated directory for that language.

Supported Languages for Error Messages

Error message localization is supported for the following languages:

- Bulgarian
- Chinese (from MariaDB 10.4.25, 10.5.16, 10.6.8, 10.7.4, 10.8.3)
- Czech
- Danish
- Dutch
- English
- Estonian
- French
- Georgian (from MariaDB 10.11.3)
- German
- Greek
- Hindi
- Hungarian
- Italian
- Japanese
- Korean
- Norwegian
- Norwegian-ny (Nynorsk)
- Polish
- Portuguese
- Romanian
- Russian
- Serbian
- Slovak
- Spanish
- Swahili (from MariaDB 11.1.2)
- Swedish
- Ukrainian

Setting the `lc_messages` and `lc_messages_dir` System Variables

The lc_messages and lc_messages_dir system variables can be used to set the server locale used for error messages.

The lc_messages system variable can be specified as a locale name. The language of the associated locale will be used for error messages. See Server Locales for a list of supported locales and their associated languages.

The lc_messages system variable is set to `en_US` by default, which means that error messages are in English by default.

If the lc_messages system variable is set to a valid locale name, but the server can't find an error message file for the language associated with the locale, then the default language will be used instead.

This system variable can be specified as command-line arguments to mariadbd or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
...
lc_messages=fr_CA
```

The lc_messages system variable can also be changed dynamically with SET GLOBAL. For example:

```
SET GLOBAL lc_messages='fr_CA';
```

If a server has the lc_messages system variable set to the `fr_CA` locale like the above example, then error messages would be in French. For example:

```
SELECT blah;
ERROR 1054 (42S22): Champ 'blah' inconnu dans field list
```

The lc_messages_dir system variable can be specified either as the path to the directory storing the server's error message files or as the path to the directory storing the specific language's error message file.

The server initially tries to interpret the value of the lc_messages_dir system variable as a path to the directory storing the server's error message files. Therefore, it constructs the path to the language's error message file by concatenating the value of the lc_messages_dir system variable with the language name of the locale specified by the lc_messages system variable .

If the server does not find the error message file for the language, then it tries to interpret the value of the lc_messages_dir system variable as a direct path to the directory storing the specific language's error message file.

This system variable can be specified as command-line arguments to mariadbd or it can be specified in a relevant server option group in an option file.

For example, to specify the path to the directory storing the server's error message files:

```
[mariadb]
...
lc_messages_dir=/usr/share/mysql/
```

Or to specify the path to the directory storing the specific language's error message file:

```
[mariadb]
...
lc_messages_dir=/usr/share/mysql/french/
```

The lc_messages_dir system variable can not be changed dynamically.

Setting the –language Option

The –language option can also be used to set the server's language for error messages, but it is deprecated. It is recommended to set the lc_messages system variable instead.

The –language option can be specified either as a language name or as the path to the directory storing the language's error message file. See Server Locales for a list of supported locales and their associated languages.

This option can be specified as command-line arguments to mariadbd or it can be specified in a relevant server option group in an option file.

For example, to specify a language name:

```
[mariadb]
...
language=french
```

Or to specify the path to the directory storing the language's error message file:

```
[mariadb]
...
language=/usr/share/mysql/french/
```

Character Set

The character set that the error messages are returned in is determined by the character_set_results variable, which defaults to UTF8.

This page is licensed: CC BY-SA / Gnu FDL

Time Zones

MariaDB keeps track of several time zone settings.

Setting the Time Zone

The time_zone system variable is the primary way to set the time zone. It can be specified in one of the following formats:

- The default value is `SYSTEM`, which indicates that the system time zone defined in the system_time_zone system variable will be used. Note that if you are using `SYSTEM` with replication in either statement or mixed mode, you MUST use the same value for `system_time_zone` on all replicas (otherwise `TIMESTAMP` columns will not replicate correctly). See System Time Zone below for more information.
- An offset from Coordinated Universal Time (UTC), such as `+5:00` or `-9:00`, can also be used.
- If the time zone tables in the mysql database were loaded, then a named time zone, such as `America/New_York`, `Africa/Johannesburg`, or `Europe/Helsinki`, is also permissible. See mysql Time Zone Tables below for more information.

There are two time zone settings that can be set within MariaDB–the global server time zone, and the time zone for your current session. There is also a third time zone setting which may be relevant–the system time zone.

Global Server Time Zone

The global server time zone can be changed at server startup by setting the `--default-time-zone` option either on the command-line or in a server option group in an option file. For example:

```
[mariadb]
...
default_time_zone = 'America/New_York'
```

The global server time zone can also be changed dynamically by setting the time_zone system variable as a user account that has the SUPER privilege. For example:

```
SET GLOBAL time_zone = 'America/New_York';
```

The current global server time zone can be viewed by looking at the global value of the time_zone system variable. For example:

```
SHOW GLOBAL VARIABLES LIKE 'time_zone';
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| time_zone     | SYSTEM |
+---------------+--------+
```

Session Time Zone

Each session that connects to the server will also have its own time zone. This time zone is initially inherited from the global value of the time_zone system variable, which sets the session value of the same variable.

A session's time zone can be changed dynamically by setting the time_zone system variable. For example:

```
SET time_zone = 'America/New_York';
```

The current session time zone can be viewed by looking at the session value of the time_zone system variable. For example:

```
SHOW SESSION VARIABLES LIKE 'time_zone';
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| time_zone     | SYSTEM |
+---------------+--------+
```

System Time Zone

The system time zone is determined when the server starts, and it sets the value of the system_time_zone system variable. The system time zone is usually read from the operating system's environment. You can change the system time zone in several different ways, such as:

- If you are starting the server with mariadbd-safe, then you can set the system time zone with the `--timezone` option either on the command-line or in the [mariadbd-safe] option group in an option file. For example:

```
[mariadbd-safe]
timezone='America/New_York'
```

- If you are using a Unix-like operating system, then you can set the system time zone by setting the `TZ` environment variable in your shell before starting the server. For example:

```
$ export TZ='America/New_York'
$ service mariadb start
```

- On some Linux operating systems, you can change the default time zone for the whole system by making the /etc/localtime symbolic link point to the desired time zone. For example:

```
$ sudo rm /etc/localtime
$ sudo ln -s /usr/share/zoneinfo/America/New_York /etc/localtime
```

- On some Debian-based Linux operating systems, you can change the default time zone for the whole system by executing the following:

```
sudo dpkg-reconfigure tzdata
```

- On Linux operating systems that use systemd, you can change the default time zone for the whole system by using the timedatectl utility. For example:

```
sudo timedatectl set-timezone America/New_York
```

Time Zone Effects

Time Zone Effects on Functions

Some functions are affected by the time zone settings. These include:

- NOW()
- SYSDATE()
- CURDATE()
- CURTIME()
- UNIX_TIMESTAMP()

Some functions are not affected. These include:

- UTC_DATE()
- UTC_TIME()
- UTC_TIMESTAMP()

Time Zone Effects on Data Types

Some data types are affected by the time zone settings.

- TIMESTAMP - See TIMESTAMP: Time Zones for information on how this data type is affected by time zones.
- DATETIME - See DATETIME: Time Zones for information on how this data type is affected by time zones.

mysql Time Zone Tables

The mysql database contains a number of time zone tables:

- time_zone
- time_zone_leap_second
- time_zone_name
- time_zone_transition
- time_zone_transition_type

By default, these time zone tables in the mysql database are created, but not populated.

If you are using a Unix-like operating system, then you can populate these tables using the mariadb-tzinfo-to-sql utility, which uses the zoneinfo data available on Linux, Mac OS X, FreeBSD and Solaris.

If you are using Windows, then you will need to import pre-populated time zone tables. These are available at MariaDB mirrors.

Time zone data needs to be updated on occasion. When that happens, the time zone tables may need to be reloaded.

See Also

- LinuxJedi in Spacetime: Properly Handling Time and Date (video)

This page is licensed: CC BY-SA / Gnu FDL

Plugins

See Plugin Overview.

MariaDB supports loading plugins at startup or runtime to extend functionality, including storage engines, security features, and logging capabilities, without rebuilding the server.

See Plugin Maturity.

This page lists the maturity level (Alpha, Beta, Gamma, Stable) of various MariaDB plugins, helping users determine which are suitable for production environments.

See Storage Engines.

Every MariaDB storage engine is implemented as a plugin. This section covers them all, from InnoDB and Aria to Spider, S3, and CONNECT.

See MariaDB Enterprise Audit.

The MariaDB Enterprise Audit plugin logs detailed data access and configuration changes, offering advanced filtering to meet security and compliance requirements.

See MariaDB Community Audit Plugin.

Complete MariaDB Audit Plugin reference: server_audit activity logging, connection/query event tracking, file/syslog output, and compliance configuration.

See Authentication Plugins.

Explore the authentication plugins available in MariaDB, such as ed25519, GSSAPI, and PAM, which provide flexible and secure methods for user verification.

See MariaDB Replication & Cluster Plugins.

This section covers plugins specifically designed for high availability and clustering, including the wsrep_provider plugin used for Galera Cluster integration.

See Other Plugins.

Discover additional plugins that extend MariaDB Server functionality, such as the Disks, Feedback, and Query Response Time plugins, for specialized use cases.

See Password Validation Plugins.

Password validation plugins, like simple_password_check and cracklib, enforce strong password policies by checking new passwords against defined complexity rules.

This page is licensed: CC BY-SA / Gnu FDL

Plugin Overview

MariaDB supports the use of plugins, software components that may be added to the core software without having to rebuild the MariaDB server from source code. Plugins can be loaded at startup, or loaded and unloaded while the server is running, without interruption. Plugins are commonly used for adding desired storage engines, additional security requirements, logging special information about the server, or even small enhancements, such as a plugin to get a timestamp as an integer.

Querying Plugin Information

There are a number of ways to see which plugins are currently active.

A server almost always has a large number of active plugins, because the server contains a large number of built-in plugins, which are active by default and cannot be uninstalled.

Querying Plugin Information with `SHOW PLUGINS`

The SHOW PLUGINS statement can be used to query information about all active plugins.

For example:

```
SHOW PLUGINS\G;
*********************** 1. row ***********************
   Name: binlog
 Status: ACTIVE
   Type: STORAGE ENGINE
Library: NULL
License: GPL
*********************** 2. row ***********************
   Name: mysql_native_password
 Status: ACTIVE
   Type: AUTHENTICATION
Library: NULL
License: GPL
*********************** 3. row ***********************
   Name: mysql_old_password
 Status: ACTIVE
   Type: AUTHENTICATION
Library: NULL
License: GPL
...
```

If a plugin's `Library` column has a `NULL` value, then the plugin is built-in, and it cannot be uninstalled.

Querying Plugin Information with information_schema.PLUGINS

The information_schema.PLUGINS table can be queried to get more detailed information about plugins.

For example:

```
SELECT * FROM information_schema.PLUGINS\G
...
*************************** 65. row ***************************
           PLUGIN_NAME: user_variables
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: ACTIVE
           PLUGIN_TYPE: INFORMATION SCHEMA
   PLUGIN_TYPE_VERSION: 110600.0
        PLUGIN_LIBRARY: NULL
PLUGIN_LIBRARY_VERSION: NULL
         PLUGIN_AUTHOR: Sergey Vojtovich
    PLUGIN_DESCRIPTION: User-defined variables
        PLUGIN_LICENSE: GPL
           LOAD_OPTION: ON
       PLUGIN_MATURITY: Stable
   PLUGIN_AUTH_VERSION: 1.0
*************************** 66. row ***************************
           PLUGIN_NAME: wsrep_provider
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: ACTIVE
           PLUGIN_TYPE: REPLICATION
   PLUGIN_TYPE_VERSION: 2.0
        PLUGIN_LIBRARY: NULL
PLUGIN_LIBRARY_VERSION: NULL
         PLUGIN_AUTHOR: Codership Oy
    PLUGIN_DESCRIPTION: Wsrep provider plugin
        PLUGIN_LICENSE: GPL
           LOAD_OPTION: ON
       PLUGIN_MATURITY: Alpha
   PLUGIN_AUTH_VERSION: 1.0
*************************** 67. row ***************************
           PLUGIN_NAME: THREAD_POOL_GROUPS
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: ACTIVE
           PLUGIN_TYPE: INFORMATION SCHEMA
   PLUGIN_TYPE_VERSION: 110600.0
        PLUGIN_LIBRARY: NULL
PLUGIN_LIBRARY_VERSION: NULL
         PLUGIN_AUTHOR: Vladislav Vaintroub
    PLUGIN_DESCRIPTION: Provides information about threadpool groups.
        PLUGIN_LICENSE: GPL
           LOAD_OPTION: ON
       PLUGIN_MATURITY: Stable
   PLUGIN_AUTH_VERSION: 1.0
...
```

If a plugin's `PLUGIN_LIBRARY` column has the `NULL` value, the plugin is built-in and cannot be uninstalled.

Querying Plugin Information with `mysql.plugin`

The mysql.plugin table can be queried to get information about installed plugins.

This table only contains information about plugins that have been installed via the following methods:

- The INSTALL SONAME statement.
- The INSTALL PLUGIN statement.
- The mariadb-plugin utility.

This table does not contain information about:

- Built-in plugins.
- Plugins loaded with the –plugin-load-add option.
- Plugins loaded with the –plugin-load option.

This table only contains enough information to reload the plugin when the server is restarted, which means it only contains the plugin name and the plugin library.

For example:

```
SELECT * FROM mysql.plugin;

+------+------------+
| name | dl         |
+------+------------+
| PBXT | libpbxt.so |
+------+------------+
```

Installing a Plugin

There are three primary ways to install a plugin:

- A plugin can be installed dynamically with an SQL statement.
- A plugin can be installed with a mariadbd option, but it requires a server restart.
- A plugin can be installed with the mariadb-plugin utility, while the server is completely offline.

When you are installing a plugin, you also have to ensure that:

- The server's plugin directory is properly configured, and the plugin's library is in the plugin directory.
- The server's minimum plugin maturity is properly configured, and the plugin is mature enough to be installed.

Installing a Plugin Dynamically

A plugin can be installed dynamically by executing either the INSTALL SONAME or the INSTALL PLUGIN statement.

If a plugin is installed with one of these statements, a record will be added to the mysql.plugins table for the plugin. This means that the plugin will automatically be loaded every time the server restarts, unless specifically uninstalled or deactivated.

Installing a Plugin with `INSTALL SONAME`

You can install a plugin dynamically by executing the INSTALL SONAME statement. INSTALL SONAME installs all plugins from the given plugin library. This could be required for some plugin libraries.

For example, to install all plugins in the `server_audit` plugin library (which is currently only the server_audit audit plugin), you could execute the following:

```
INSTALL SONAME 'server_audit';
```

Installing a Plugin with `INSTALL PLUGIN`

You can install a plugin dynamically by executing the INSTALL PLUGIN statement. INSTALL PLUGIN installs a single plugin from the given plugin library.

For example, to install the server_audit audit plugin from the `server_audit` plugin library, you could execute the following:

```
INSTALL PLUGIN server_audit SONAME 'server_audit';
```

Installing a Plugin with Plugin Load Options

A plugin can be installed with a mariadbd option by providing either the –plugin-load-add or the –plugin-load option.

If a plugin is installed with one of these options, then a record will **not** be added to the mysql.plugins table for the plugin. This means that if the server is restarted without the same option set, then the plugin will **not** automatically be loaded.

Installing a Plugin with `--plugin-load-add`

You can install a plugin with the –plugin-load-add option by specifying the option as a command-line argument to mariadbd or by specifying the option in a relevant server option group in an option file.

The `--plugin-load-add` option uses the following format:

- Plugins can be specified in the format `name=library`, where `name` is the plugin name and `library` is the plugin library. This format installs a single plugin from the given plugin library.
- Plugins can also be specified in the format `library`, where `library` is the plugin library. This format installs all plugins from the given plugin library.
- Multiple plugins can be specified by separating them with semicolons.

For example, to install all plugins in the `server_audit` plugin library (which is currently only the server_audit audit plugin) and also the ed25519 authentication plugin from the `auth_ed25519` plugin library, you could set the option to the following values on the command-line:

```
$ mariadbd --user=mysql --plugin-load-add='server_audit' --plugin-load-add='ed25519=auth_ed25519'
```

You could also set the option to the same values in an option file:

```
[mariadb]
...
plugin_load_add = server_audit
plugin_load_add = ed25519=auth_ed25519
```

Special care must be taken when specifying both the –plugin-load option and the –plugin-load-add option together. The `--plugin-load` option resets the plugin load list, and this can cause unexpected problems if you are not aware. The `--plugin-load-add` option does **not** reset the plugin load list, so it is much safer to use. See Specifying Multiple Plugin Load Options for more information.

Installing a Plugin with `--plugin-load`

You can install a plugin with the –plugin-load option by specifying the option as a command-line argument to mariadbd or by specifying the option in a relevant server option group in an option file.

The `--plugin-load` option uses the following format:

- Plugins can be specified in the format `name=library`, where `name` is the plugin name and `library` is the plugin library. This format installs a single plugin from the given plugin library.
- Plugins can also be specified in the format `library`, where `library` is the plugin library. This format installs all plugins from the given plugin library.
- Multiple plugins can be specified by separating them with semicolons.

For example, to install all plugins in the `server_audit` plugin library (which is currently only the server_audit audit plugin) and also the ed25519 authentication plugin from the `auth_ed25519` plugin library, you could set the option to the following values on the command-line:

```
$ mariadbd --user=mysql --plugin-load='server_audit;ed25519=auth_ed25519'
```

You could also set the option to the same values in an option file:

```
[mariadb]
...
plugin_load = server_audit;ed25519=auth_ed25519
```

Special care must be taken when specifying the `--plugin-load` option multiple times, or when specifying both the –plugin-load option and the `--plugin-load-add` option together. The `--plugin-load` option resets the plugin load list, and this can cause unexpected problems if you are not aware. The `--plugin-load-add` option does **not** reset the plugin load list, so it is much safer to use. See Specifying Multiple Plugin Load Options for more information.

Specifying Multiple Plugin Load Options

Special care must be taken when specifying the `--plugin-load` option multiple times, or when specifying both the `--plugin-load` option and the -`-plugin-load-add` option. The `--plugin-load` option resets the plugin load list, and this can cause unexpected problems if you are not aware. The `--plugin-load-add` option does **not** reset the plugin load list, so it is much safer to use.

This can have the following consequences:

- If the `--plugin-load` option is specified multiple times, then only the last instance will have any effect. For example, in the following case, the first instance of the option is reset:

```
[mariadb]
...
plugin_load = server_audit
plugin_load = ed25519=auth_ed25519
```

- If the –plugin-load option is specified after the `--plugin-load-add` option, then it will also reset the changes made by that option. For example, in the following case, the `--plugin-load-add` option does not do anything, because the subsequent `--plugin-load` option resets the plugin load list:

```
[mariadb]
...
plugin_load_add = server_audit
plugin_load = ed25519=auth_ed25519
```

- In contrast, if the `--plugin-load` option is specified before the `--plugin-load-add` option, then it will work fine, because the `--plugin-load-add` option does not reset the plugin load list. For example, in the following case, both plugins are properly loaded:

```
[mariadb]
...
plugin_load = server_audit
plugin_load_add = ed25519=auth_ed25519
```

Installing a Plugin with mariadb-plugin

A plugin can be installed with the mariadb-plugin utility if the server is completely offline.

The syntax is:

```
mariadb-plugin [options] <plugin> ENABLE|DISABLE
```

For example, to install the server_audit audit plugin, you could execute the following:

```
mariadb-plugin server_audit ENABLE
```

If a plugin is installed with this utility, a record will be added to the mysql.plugins table for the plugin. This means that the plugin will automatically be loaded every time the server restarts, unless specifically uninstalled or deactivated.

Configuring the Plugin Directory

When a plugin is being installed, the server looks for the plugin's library in the server's plugin directory. This directory is configured by the `plugin_dir` system variable. This can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_dir = /usr/lib64/mysql/plugin
```

Configuring the Minimum Plugin Maturity

When a plugin is being installed, the server compares the plugin's maturity level against the server's minimum allowed plugin maturity. This can help prevent users from using unstable plugins on production servers. This minimum plugin maturity is configured by the plugin_maturity system variable. This can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_maturity = stable
```

Configuring Plugin Activation at Server Startup

A plugin will be loaded by default when the server starts if:

- The plugin was installed with the INSTALL SONAME statement.
- The plugin was installed with the INSTALL PLUGIN statement.
- The plugin was installed with the mariadb-plugin utility.
- The server is configured to load the plugin with the –plugin-load-add option.
- The server is configured to load the plugin with the –plugin-load option.

This behavior can be changed with special options that take the form `--plugin-name`. For example, for the server_audit audit plugin, the special option is called –server-audit.

The possible values for these special options are:

Option Value	Description
OFF	Disables the plugin without removing it from the mysql.plugins table.
ON	Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
FORCE	Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
FORCE_PLUS_PERMANENT	Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.

A plugin's status can be found by looking at the `PLUGIN_STATUS` column of the information_schema.PLUGINS table.

Uninstalling Plugins

Plugins that are found in the mysql.plugin table, that is those that were installed with INSTALL SONAME, INSTALL PLUGIN or mariadb-plugin can be uninstalled in one of two ways:

- The UNINSTALL SONAME or the UNINSTALL PLUGIN statement while the server is running
- With mariadb-plugin while the server is offline.

Plugins that were enabled as a `--plugin-load` option do not need to be uninstalled. If `--plugin-load` is omitted the next time the server starts, or the plugin is not listed as one of the `--plugin-load` entries, the plugin will not be

loaded.

UNINSTALL PLUGIN uninstalls a single installed plugin, while UNINSTALL SONAME uninstalls all plugins belonging to a given library.

See Also

- List of Plugins
- INSTALL PLUGIN
- INSTALL SONAME
- UNINSTALL PLUGIN
- UNINSTALL SONAME
- SHOW PLUGINS
- INFORMATION_SCHEMA.PLUGINS Table
- mariadb-plugin

This page is licensed: CC BY-SA / Gnu FDL

Plugin Maturity

The following table lists the various plugins included in MariaDB ordered by their maturity. Note that maturity will differ across MariaDB versions - see below for an easy way to get a complete list of plugins and their maturity in your version of MariaDB:

Plugin	Version	Maturity
Archive	3.0	Stable
Aria	1.6	Stable
Audit Plugin	1.4	Stable
aws_key_management	1.0	Stable
binlog	2.0	Stable
Blackhole	1.0	Stable
Connect	1.7	Stable
CLIENT_STATISTICS	2.0	Stable
cracklib_password_check	1.0	Stable
CSV	1.0	Stable
DECLARE BY / associative array	1.0	Gamma
DISKS	1.1	Stable
ed25519	1.1	Stable
FederatedX	2.1	Stable
Feedback	1.1	Stable
file_key_management	2.0	Gamma
gssapi	1.0	Stable
INDEX_STATISTICS	2.0	Stable
INET4	1.0	Gamma
INET6	1.0	Stable
InnoDB	10.*	Stable
LOCALES	1.0	Stable

Plugin	Version	Maturity
Memory	1.0	Stable
METADATA_LOCK_INFO	0.1	Stable
mhnsw	1.0	Stable
MRG_MyISAM	1.0	Stable
Mroonga	7.7	Stable
MyISAM	1.0	Stable
MyRocks	1.0	Stable
mysql_json	0.1	Stable
mysql_native_password	1.0	Stable
mysql_old_password	1.0	Stable
named_pipe	1.0	Stable
online_alter_log	2.0	Stable
pam	1.0	Stable
password_reuse_check	2.0	Stable (introduced in 10.10.2, 10.9.5, and 10.8.7)
`partition`	1.0	Stable
Performance_Schema	0.1	Stable
QUERY_CACHE_INFO	1.1	Stable
query_response_time	1.0	Stable
S3	1.0	Stable
semisync	1.0	Stable
Sequence	1.0	Stable
SERVER_AUDIT	2.6	Stable
simple_password_check	1.0	Stable
Spider	3.3	Stable
SQL_ERROR_LOG	1.0	Stable
SYS_REFCURSOR	1.0	Gamma
TABLE_STATISTICS	2.0	Stable
USER_STATISTICS	2.0	Stable
user_variables	1.0	Stable
TokuDB	4.0	Stable (removed in 10.6)
unix_socket	1.1	Stable
UUID	1.0	Stable (introduced in 10.9.1)
uuid_v4	1.0	Stable
uuid_v7	1.0	Stable
wsrep	1.0	Stable
wsrep_connections	1.0	Stable
WSREP_INFO	1.0	Stable

Plugin	Version	Maturity
wsrep_provider	1.0	Stable
wsrep_thd_info	1.0	Stable
OQGraph	3.0	Gamma
Sphinx	2.0	Gamma
Columnstore	1.0	Beta
handlersocket	1.0	Beta
Cassandra	0.1	Experimental (removed in 10.6)
debug_key_management	1.0	Experimental
example_key_management	1.0	Experimental

This query shows a complete list of plugins and their maturity level:

```
SELECT plugin_name, plugin_version, plugin_maturity
FROM information_schema.plugins
ORDER BY plugin_name;
```

Plugin Support and Maintenance

Plugin maturity is not the same as plugin support and maintenance. When a plugin has a maturity level of *Stable*, that doesn't necessarily imply that it's actively maintained or supported via MariaDB Support.

Currently, the plugins maintained and supported by MariaDB plc for a given release series are the ones included in the corresponding MariaDB Enterprise Server release. Plugins that are not supported by MariaDB plc are either placed in a separate **unsupported repository** (reached by passing the `--include-unsupported` flag to the `mariadb_es_repo_setup` script) or omitted from the build entirely.

See Also

- Plugin Overview
- INSTALL PLUGIN
- INFORMATION_SCHEMA.PLUGINS Table
- mysql_plugin
- SHOW PLUGINS
- INSTALL SONAME
- UNINSTALL PLUGIN
- UNINSTALL SONAME

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Enterprise Audit

Overview

While application design specifications and database configurations may intend specific limitations to be placed around access to data, audit mechanisms help confirm the effectiveness of these controls.

MariaDB Enterprise Audit is a plugin that logs data access and database operations.

Audit mechanisms are most effective when they produce a manageable quantity of output. MariaDB Enterprise Audit includes advanced filtering features to enable narrowly defining which information is logged.

Where audit mechanisms may only be effective when control parameters can also be audited, MariaDB Enterprise Audit implements logging of configuration changes.

> **Important:**
>
> **Plugin Conflict (**MENT-316**)**
>
> The MariaDB Enterprise Audit plugin (`server_audit2.so`) is incompatible with the older server_audit.so (`v1`) plugin. Running both can cause server instability or deadlocks.
>
> *If you are a new user: You can continue reading.*
>
> *If you are migrating: You must remove the old `v1` plugin. Please go directly to the Upgrades section for complete instructions.*

Configuration Overview

MariaDB Enterprise Audit is installed and loaded by default. If you are unsure whether it is loaded on your system, you can confirm that the plugin is loaded.

To use MariaDB Enterprise Audit, the plugin must be configured: Administrators must define Audit Filters to configure what MariaDB Enterprise Audit writes to the audit log.

MariaDB Enterprise Audit supports two types of Audit Filters:

Audit Filter Type	Used For
Default Audit Filter	The Default Audit Filter is used for any user account that is not assigned a Named Audit Filter.
Named Audit Filters	Named Audit Filters are assigned to specific user accounts.

Administrators can define Audit Filters to audit log activity using multiple types of filters:

Filter Type	Used For
Event Filters	Event Filters are used to enable or disable audit logging for specific types of operations performed by the user accounts assigned to the Audit Filter.
Logging Filters	Logging Filters are used to enable or disable audit logging for the user accounts assigned to the Audit Filter.
Object Filters	Object Filters are used to enable or disable audit logging for specific databases or tables accessed by the user accounts assigned to the Audit Filter. Support for Object Filters was added in MariaDB Enterprise Server 10.6. Support for Object Filters was backported to ES 10.4.21-13 and ES 10.5.12-8.

Administrators must start audit logging.

General Operation

MariaDB Enterprise Audit performs audit logging during typical operations of MariaDB Enterprise Server:

- When a user connects, fails to connect, or disconnects, MariaDB Enterprise Audit will write a message to the audit log if the Audit Filter specifies the specific type of Connect Event.
- When a user executes a query, MariaDB Enterprise Audit will write a message to the audit log if the Audit Filter specifies the specific type of Query Event and if the queried objects are not excluded by Object Filters.
- When a query accesses a table, MariaDB Enterprise Audit will write a message to the audit log if the Audit Filter specifies the specific type of Table Event and if the table is not excluded by Object Filters.
- When a query modifies the MariaDB Enterprise Audit configuration, MariaDB Enterprise Audit will write a message to the audit log if the Audit Filter specifies the Audit Config Event.

- When a user's Audit Filter contains a Logging Filter that disables audit logging, all audit logging for the user's activity will be skipped.

MariaDB Enterprise Audit writes audit log messages either to a dedicated audit log file or to the system log (syslog), depending on configuration.

Audit log messages that correspond to a query are logged when the query completes. If a query is executed in a transaction, the audit log messages that correspond to the query are logged when the individual query completes, not when the transaction completes. If the query fails, the audit log message is logged at time of failure.

Audit Log Considerations

Care must be taken when logging data for audit to maintain alignment to business requirements. Concerns include:

- Queries should not be logged for tables containing Personally Identifiable Information (PII) or Sensitive PII (SPII) such as passwords, since audit log data is written unencrypted. Consider using Object Filters to exclude sensitive information from the audit log.
- Backup of audit data should be performed at least as frequently as database backups.
- Audit log data on database servers could be tampered with if the database server is compromised. Consider secure transmission of log data to a hardened and remote logging server.
- Where audit data is mission-critical, it should be subject to controls, data protection, data retention, and highly-available storage as are used for other mission-critical data.

Install the Audit Plugin

MariaDB Enterprise Audit comes preinstalled with MariaDB Enterprise Server, so no manual installation is required.

To verify that the plugin is installed:

1. Locate your server's plugin directory.
   - When MariaDB Enterprise Server is running, you can find the directory by checking the value of the `plugin_dir` system variable.

```
SHOW GLOBAL VARIABLES
   LIKE 'plugin_dir';
```

```
+---------------+--------------------------+
| Variable_name | Value                    |
+---------------+--------------------------+
| plugin_dir    | /usr/lib64/mysql/plugin/ |
+---------------+--------------------------+
```

2. Verify the `server_audit2.so` file—the shared library used by MariaDB Enterprise Audit—is present in your server's plugin directory.

```
$ ls -l /usr/lib64/mysql/plugin/server_audit2.so
```

```
-rwxr-xr-x. 1 root root 70432 Jul 15 19:03 /usr/lib64/mysql/plugin/server_audit2.so
```

MariaDB Enterprise Audit is bundled with all MariaDB Enterprise Server distributions (binary tarball, DEB/RPM package tarball, and DEB/RPM packages). If you do not see the `server_audit2.so` file, verify that MariaDB Enterprise Server has been installed correctly.

Load the Audit Plugin

MariaDB Enterprise Audit is enabled through the `mariadb-enterprise.cnf` configuration file, which is included by default with MariaDB Enterprise Server. This means manual loading is usually not required.

The `mariadb-enterprise.cnf` file activates MariaDB Enterprise Audit by configuring the `plugin-load-add` and `server-audit` options.

```
# -- Auditing - pre-load Plugin
plugin-load-add=server_audit
server_audit=FORCE_PLUS_PERMANENT
```

If your environment does not use `mariadb-enterprise.cnf`, you can enable MariaDB Enterprise Audit by adding the same options to your own configuration file.

Confirm the Audit Plugin is Loaded

To verify that MariaDB Enterprise Audit is installed, check the `information_schema.PLUGINS` table.

```
SELECT PLUGIN_STATUS, PLUGIN_LIBRARY, PLUGIN_DESCRIPTION, LOAD_OPTION
FROM information_schema.PLUGINS
WHERE PLUGIN_NAME='SERVER_AUDIT'\G
```

```
*************************** 1. row ***************************
     PLUGIN_STATUS: ACTIVE
    PLUGIN_LIBRARY: server_audit2.so
PLUGIN_DESCRIPTION: MariaDB Enterprise Audit
       LOAD_OPTION: FORCE_PLUS_PERMANENT
```

MariaDB Enterprise Audit is enabled through the `mariadb-enterprise.cnf` configuration file, which is included by default in MariaDB Enterprise Server. If your results differ from the example output above, verify that the `mariadb-enterprise.cnf` file specifies the `plugin-load-add` and `server-audit` options.

Start Audit Logging

When MariaDB Enterprise Audit is installed and loaded, audit logging does not begin automatically. You must explicitly start it, either from the shell or through SQL.

Interface	Method	Benefits
Shell	Configuration File	SQL access is not required SUPER privilege is not required Configuration file can be version controlled.
SQL	SET GLOBAL Statement	Server restart is not required.

Start Audit Logging in Configuration File

Enable audit logging with MariaDB Enterprise Audit by setting the `server_audit_logging` system variable in a configuration file. Alternatively, enable it dynamically with `SET GLOBAL`, which does not require a server restart.

To configure in a file:

1. Set the `server_audit_logging` system variable in the configuration file.

```
[mariadb]
server_audit_logging = ON
```

2. Restart MariaDB Enterprise Server:

```
$ sudo systemctl restart mariadb
```

If the server does not start, review the error log for details.

3. To confirm that audit logging is running, check the value of the `Server_audit_active` status variable using the `SHOW GLOBAL STATUS` statement.

```
SHOW GLOBAL STATUS
   LIKE 'Server_audit_active';
```

```
+---------------------+-------+
| Variable_name       | Value |
+---------------------+-------+
| Server_audit_active | ON    |
+---------------------+-------+
```

Start Audit Logging with SET GLOBAL

Audit logging with MariaDB Enterprise Audit can be started by setting the server_audit_logging system variable with the SET GLOBAL statement, which requires the SUPER privilege.

1. Set the server_audit_logging system variable with the SET GLOBAL statement:

```
SET GLOBAL server_audit_logging=ON;
```

2. Confirm that audit logging is started by querying the Server_audit_active status variable with the SHOW GLOBAL STATUS statement:

```
SHOW GLOBAL STATUS
   LIKE 'Server_audit_active';
```

```
+---------------------+-------+
| Variable_name       | Value |
+---------------------+-------+
| Server_audit_active | ON    |
+---------------------+-------+
```

When you modify a system variable dynamically using the `SET GLOBAL` statement, the change is not preserved after a server restart. To ensure audit logging automatically starts with the server, also configure the `server_audit_logging` system variable in a configuration file.

```
[mariadb]
server_audit_logging = ON
```

Confirm Audit Logging is Started

Confirm that audit logging is started by querying the Server_audit_active status variable with the SHOW GLOBAL STATUS statement:

```
SHOW GLOBAL STATUS
   LIKE 'Server_audit_active';
```

```
+---------------------+-------+
| Variable_name       | Value |
+---------------------+-------+
| Server_audit_active | ON    |
+---------------------+-------+
```

Audit Logging Buffer Writes

> **Note:**
>
> Buffer writes are available from MariaDB Enterprise Server 11.8.

Audit log buffering is controlled by these variables:

- `server_audit_file_buffer_size` — This defines the size of the buffer. The default value is `0`, meaning there's no buffering at all. Setting non-zero value enables the buffering with the buffer of the specified size aligned by `8192`. The maximum value is `65536`.
- `server_audit_sync_log_file` — This flushes the buffer to the log file. While the log record is in the buffer, it cannot be seen in the log file. If there aren't many events to log, the time before records can be observed can be significant. You can issue this statement to force writing the buffer to the file, making sure not to miss recent records:

```
SET GLOBAL server_audit_log_file=1
```

Forbid Uninstallation

In a secure environment, MariaDB Enterprise Audit provides administrators with an audit trail of actions performed by users on the MariaDB Enterprise Server node. To protect the integrity of the audit trail, users should not be able to uninstall MariaDB Enterprise Audit. If the server-audit option is set to `FORCE_PLUS_PERMANENT`, MariaDB Enterprise Server will prevent MariaDB Enterprise Audit from being uninstalled:

```
server_audit=FORCE_PLUS_PERMANENT
```

When a user tries to uninstall MariaDB Enterprise Audit with the server-audit option set to `FORCE_PLUS_PERMANENT`, the operation fails with the `ER_PLUGIN_IS_PERMANENT` error code:

```
UNINSTALL SONAME 'server_audit2';
```

```
ERROR 1702 (HY000): Plugin 'SERVER_AUDIT' is force_plus_permanent and can not be unloaded
```

Note:

The `mariadb-enterprise.cnf` configuration file included by default in MariaDB Enterprise Server sets the server-audit option to `FORCE_PLUS_PERMANENT`. As a consequence, MariaDB Enterprise Server forbids MariaDB Enterprise Audit from being uninstalled by default.

If you do not use `mariadb-enterprise.cnf` in your environment, you can configure MariaDB Enterprise Audit to forbid uninstallation by setting the server-audit option in your configuration file.

Confirm that Uninstallation is Forbidden

To confirm that MariaDB Enterprise Audit is configured to forbid uninstallation, query the information_schema.PLUGINS table:

```
SELECT PLUGIN_STATUS, PLUGIN_LIBRARY, PLUGIN_DESCRIPTION, LOAD_OPTION
FROM information_schema.PLUGINS
WHERE PLUGIN_NAME='SERVER_AUDIT'\G
```

```
*************************** 1. row ***************************
     PLUGIN_STATUS: ACTIVE
    PLUGIN_LIBRARY: server_audit2.so
PLUGIN_DESCRIPTION: MariaDB Enterprise Audit
       LOAD_OPTION: FORCE_PLUS_PERMANENT
```

If your output does not match the example output shown above, confirm that the `mariadb-enterprise.cnf` configuration file sets the server-audit option to `FORCE_PLUS_PERMANENT`.

Server Startup Behavior

Some specific server startup behavior is described in the sections below.

For examples of error messages that can appear in the MariaDB error log during server startup, see Messages in MariaDB Error Log.

Enterprise Audit Not Loaded

MariaDB Enterprise Server can startup and handle traffic even if MariaDB Enterprise Audit is not installed or loaded. However, the specific behavior can be configured.

When the server-audit option is set to `FORCE` or `FORCE_PLUS_PERMANENT`, MariaDB Enterprise Server will fail to start if MariaDB Enterprise Audit can't be loaded.

The mariadb-enterprise.cnf configuration file included by default in MariaDB Enterprise Server sets the server-audit option to `FORCE_PLUS_PERMANENT`. As a consequence, MariaDB Enterprise Server forbids MariaDB Enterprise Audit from being uninstalled by default.

Invalid Filter Definitions

MariaDB Enterprise Audit attempts to load the Audit Filters during startup.

When MariaDB Enterprise Audit encounters errors in the Audit Filter definitions, MariaDB Enterprise Server will still start and load MariaDB Enterprise Audit. In this situation, an error will be written to the MariaDB error log and all Events for all objects will be audit logged.

Additional Points of Control

When enabled, MariaDB Enterprise Audit logs designated Events that occur on a running instance of MariaDB Enterprise Server.

Additional audit practices should be established to cover:

- Data backup controls.
- System-level controls, including authentication, file system, and process execution.
- Network-level controls.
- Monitoring systems, including monitoring for audit logging.
- Changes to user accounts (and the mysql.global_priv system table), which can necessitate changes to Audit Filters.

Audit Filters

Filters are JSON objects that specify what you want MariaDB Enterprise Audit to monitor.

There are two types of filters:

Audit Filter Type	Description
Default Audit Filter	The Default Audit Filter is used for all user accounts that are not assigned a Named Audit Filter. Only a single Default Audit Filter can be defined in the `mysql.server_audit_filters` system table, and it must be defined with the name default.
Named Audit Filters	Named Audit Filters must be assigned to specific user accounts. Many Named Audit Filters can be defined in the `mysql.server_audit_filters` system table, and they must be defined with unique names. A Named Audit Filter can be assigned to a user account by inserting the user account details and the filter name into the `mysql.server_audit_users` system table.

Default Audit Filter

The Default Audit Filter applies to all user accounts that do not have a specific Named Audit Filter assigned. Only one Default Audit Filter is allowed in the `mysql.server_audit_filters` system table, and it must be named

`default`.

Create a Default Audit Filter

If you want to create a Default Audit Filter, you need to insert the details into the `mysql.server_audit_filters` system table. The name for the Default Audit Filter must be default and the rule should be designed to meet your audit logging requirements.

To create a Default Audit Filter:

1. Confirm that a Default Audit Filter does not already exist:

```
SELECT * FROM mysql.server_audit_filters
    WHERE filtername = 'default';
```

2. If a Default Audit Filter already exists, remove it:

```
DELETE FROM mysql.server_audit_filters
    WHERE filtername = 'default';
```

3. Insert the details for the new Default Audit Filter into the mysql.server_audit_filters system table:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES ('default',
      JSON_COMPACT(
         '{
            "connect_event":"ALL",
            "table_event":"WRITE"
         }'
      ));
```

This example Audit Filter configures audit logging for all Connection Events and Write Table Events.

The example passes the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

4. Reload the Audit Filters by setting the server_audit_reload_filters system variable to `ON`:

```
SET GLOBAL server_audit_reload_filters=ON;
```

Disable Logging with the Default Filter

It is recommended to use the default filter to assure that any user is audited, also if not defined in the `mysql.server_audit_users` system table.

In some special cases you might want audits only to be enabled for the users in `mysql.server_audit_users`. In this case you should use the following default filter to disable logging for all other users.

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES ('default',
      JSON_COMPACT(
         '{
    “logging”: “OFF”
          }'
      ));
```

Named Audit Filters

Named Audit Filters must be assigned to specific user accounts.

Multiple Named Audit Filters can be defined in the mysql.server_audit_filters system table, and they must be defined with unique names. A Named Audit Filter can be assigned to a user account by inserting the user account details and the filter name into the mysql.server_audit_users system table.

Create a Named Audit Filter

To create a Named Audit Filter, insert a row with the Audit Filter's name and the rule into the `mysql.server_audit_filters` system table:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES ('reporting',
      JSON_COMPACT(
         '{
            "connect_event": [
               "CONNECT",
               "DISCONNECT"
            ],
            "table_event":[
               "WRITE",
               "CREATE",
               "DROP",
               "RENAME",
               "ALTER"
            ]
         }'
      ));
```

This example Audit Filter is defined with the name reporting, and it configures audit logging for Connection Events and Disconnection Events and for several types of Table Events. This example Audit Filter can be useful for read-only users since it does not log table reads, but it does log table writes and schema changes which are illegitimate activities for a read-only user.

The example passes the JSON object to the `JSON_COMPACT()` function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

For additional information on how to assign the Audit Filter to a user account, see "Assign a Named Audit Filter".

Assign a Named Audit Filter

Named Audit Filters are only active when they are assigned to a specific user account.

To assign a Named Audit Filter to a user account:

1. Insert a row with the user account details and the Audit Filter's name into the `mysql.server_audit_users` system table:

```
INSERT INTO mysql.server_audit_users (host, user, filtername)
   VALUES ("%", "reader", "reporting");
```

This example statement assigns the reporting Audit Filter created in "Create a Named Audit Filter" to the `reader@%` user account.

2. Reload the Audit Filters by setting the server_audit_reload_filters system variable to `ON`:

```
SET GLOBAL server_audit_reload_filters=ON;
```

MariaDB Enterprise Audit does not track changes to user accounts. If you delete or rename a user account, the change doesn't cascade to the Audit Filter. The Audit Filters must be updated manually.

System Tables for Audit Filters

There are two system tables for Audit Filters:

System Table	Description
mysql.server_audit_filters	Audit Filter definitions with MariaDB Enterprise Audit
mysql.server_audit_users	Audit Filter assignments for user accounts with MariaDB Enterprise Audit.

Query Audit Filters

You can query Audit Filters by querying the `mysql.server_audit_filters` system table.

The JSON objects can be made more human-readable by passing them to the JSON_DETAILED() or JSON_LOOSE() functions:

```
SELECT filtername,
   JSON_DETAILED(rule)
FROM mysql.server_audit_filters\G
```

```
*************************** 1. row ***************************
         filtername: reporting
JSON_DETAILED(rule): {
    "table_event":
    [
        "WRITE",
        "CREATE",
        "DROP",
        "RENAME",
        "ALTER",

        {
            "log_databases":
            [
                "production",
                "reporting"
            ]
        }
    ]
}
```

Query User Assignments for Named Audit Filters

You can query user assignments for Named Audit Filters by joining the `mysql.server_audit_filters` and `mysql.server_audit_users` system tables.

The JSON objects can be made more human-readable by passing them to the JSON_DETAILED() or JSON_LOOSE() functions:

```
SELECT sau.host, sau.user, saf.filtername,
   JSON_DETAILED(saf.rule)
FROM mysql.server_audit_filters saf
JOIN mysql.server_audit_users sau
   ON saf.filtername = sau.filtername
WHERE saf.filtername != 'default'\G
```

```
*************************** 1. row ***************************
                   host: %
                   user: reader
             filtername: reporting
JSON_DETAILED(saf.rule): {
    "table_event":
    [
        "WRITE",
        "CREATE",
        "DROP",
        "RENAME",
        "ALTER",

        {
            "log_databases":
            [
                "production",
                "reporting"
            ]
        }
    ]
}
*************************** 2. row ***************************
                   host: %
                   user: writer
             filtername: reporting
JSON_DETAILED(saf.rule): {
    "table_event":
    [
        "WRITE",
        "CREATE",
        "DROP",
        "RENAME",
        "ALTER",

        {
            "log_databases":
            [
                "production",
                "reporting"
            ]
        }
    ]
}
```

Reload Audit Filters and Assignments

MariaDB Enterprise Audit caches its Audit Filters to improve performance. When you change an Audit Filter or an Audit Filter assignment in the system tables, you need to reload the Audit Filters for the changes to take effect.

To reload Audit Filters, set the server_audit_reload_filters system variable to ON with the SET GLOBAL statement, which requires the SUPER privilege:

```
SET GLOBAL server_audit_reload_filters=ON;
```

When you set the server_audit_reload_filters system variable to ON, MariaDB Enterprise Audit reloads all Audit Filters and assignments to ensure that it is using the latest definitions. Afterward, it sets `server_audit_reload_filters` back to `OFF`.

For additional information on how to use the server_audit_reload_filters system variable, see “Default Audit Filter” and “Named Audit Filters”.

Event Filters

MariaDB Enterprise Audit supports Event Filters in Audit Filters. Event Filters can be used to configure which Events you want to audit. Each Event refers to a different class of operation.

MariaDB Enterprise Audit defines multiple Event classes that can be used in Event Filters:

- Audit Config Events
- Connect Events
- Query Events
- Table Events

The Event classes are described in the sections below. Example audit logs for each Event class are shown in the sections below. For details about the audit log format, see Audit Log Format.

Audit Config Events

MariaDB Enterprise Audit implements Audit Config Events to help keep track of changes to the audit log configuration.

MariaDB Enterprise Audit logs Audit Config (`AUDIT_CONFIG`) Events in the following situations:

- When one of MariaDB Enterprise Audit's system variables is changed with the SET GLOBAL statement, the change is logged.
- When the audit log file is rotated, it is logged.

In the following output, three audit configuration events (`AUDIT_CONFIG`) and one query event (`QUERY`) are shown:

```
20190622 02:10:21,localhost.localdomain,,,0,0,AUDIT_CONFIG,,file_path=server_audit.log,0
20190622 02:10:21,localhost.localdomain,,,0,0,AUDIT_CONFIG,,rotate_size=1000000,0
20190622 02:10:21,localhost.localdomain,,,0,0,AUDIT_CONFIG,,file_rotations=9,0
20190622 02:10:21,localhost.localdomain,root,localhost,8,7,QUERY,mysql,'set global server_audit_logging =
on',0
```

> **Tip:**
>
> `AUDIT_CONFIG` events are always logged, so no configuration is required when audit logging is started.

Connection Events

MariaDB Enterprise Audit implements Connection Events to audit connection attempts, authentication failures, and user account changes that occur due to certain authentication plugins, such as pam.

MariaDB Enterprise Audit logs Connection Events in the following situations:

- When a user successfully connects, it is logged with the `CONNECT` Event sub-class.
- When a user disconnects, it is logged with the `DISCONNECT` Event sub-class.
- When a user fails to connect, it is logged with the `FAILED_CONNECT` Event sub-class.
- When an existing connection authenticates as a different user, it is logged with the `CHANGE_USER` Event sub-class.
- When an authentication plugin changes to a proxy user, it is logged with the `PROXY_CONNECT` Event sub-class.

In the following output, multiple sub-classes of connection events (`CONNECT` , `DISCONNECT` , `FAILED_CONNECT`) are shown:

```
20190710 00:05:30,localhost.localdomain,root, localhost,2,0,CONNECT,,,0
20190710 00:05:53,localhost.localdomain,root, localhost,2,0,DISCONNECT,,,0
20190710 00:06:28,localhost.localdomain,unknownuser,localhost,3,0,FAILED_CONNECT,,,1045
20190710 00:06:28,localhost.localdomain,unknownuser, localhost,3,0,DISCONNECT,,,0
```

Starting with MariaDB Enterprise Server 12.3, connection events record the client port alongside the host and append the negotiated TLS version:

```
20260731 09:14:22,mdbe123,root,192.168.1.24:54312,7,0,CONNECT,mysql,,0,TLSv1.3
20260731 09:14:59,mdbe123,root,192.168.1.24:54312,7,0,DISCONNECT,mysql,,0,TLSv1.3
20260731 09:15:03,mdbe123,app,localhost:unavailable,8,0,CONNECT,,,0,
```

The third record is an unencrypted Unix socket connection, so it has no client port and no TLS version.

> **Note:**
>
> `PROXY_CONNECT` records do not include the TLS version.

An **event filter for connection events** can be added to an Audit filter with the `connect_event` key, which supports the following values:

Value	Description
CONNECT	Records when the user connects to MariaDB Enterprise Server
DISCONNECT	Records when the user disconnects from MariaDB Enterprise Server
FAILED_CONNECT	Records when a user attempts to connect to MariaDB Enterprise Server, but the connection attempt fails due to authentication or similar issues
CHANGE_USER	Records when a user switches to a different user account
PROXY_CONNECT	Records proxy user connections.
ALL	Records all connection Events

This query defines a Named Audit Filter that specifies connection events:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES ('connections',
      JSON_COMPACT(
         '{
            "connect_event": [
               "CONNECT",
               "DISCONNECT"
            ]
         }'
      ));
```

The example passes the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

Query Events

MariaDB Enterprise Audit implements Query Events to audit the execution of a specific subset of SQL statements.

MariaDB Enterprise Audit logs Query Events in the following situations:

- When a SQL statement is directly executed, it is logged with the `QUERY` event.

MariaDB Enterprise Audit does not log query events for SQL statements that are indirectly executed. For example, if an SQL statement is executed as part of a view, stored procedure, stored function, or trigger, the query will not be logged. If you want to audit all table accesses, including indirect table accesses, it is recommended to enable audit logging for Table Events in addition to query events.

This query shows one query event:

```
20190710 02:21:07,localhost.localdomain,John,localhost,3,30,QUERY,db1,'SELECT * FROM services WHERE
typeid IN (SELECT id FROM services_types WHERE name="consulting")',0
```

An **event filter for query events** can be added to an Audit Filter with the `query_event` key, which supports the following values:

Value	Description
DML	Records any SQL statements in the Data Manipulation Language subset, including SELECT, INSERT, UPDATE, and DELETE statements.
DDL	Records any SQL statements in the Data Definition Language subset, including CREATE TABLE and ALTER TABLE, as well as DROP TABLE and TRUNCATE operations.
DCL	Records any SQL statements in the Data Control Language subset, including GRANT and REVOKE.
DML_READ	Records SELECT statements in the Data Manipulation Language subset.
DML_WRITE	Records any SQL statements for writes in the Data Manipulation Language subset, including INSERT, UPDATE, and DELETE statements.
DML_NO_SELECT	Alias for DML_WRITE
ALL	Records any SQL statements run by the user.

For example, the following query defines a Named Audit Filter that specifies Query Events:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES (
       'queries',
       JSON_COMPACT(
          '{
              "query_event": [
                  "DML",
                  "DDL",
          "DCL",
          "DML_NO_SELECT",
          "DML_WRITE",
          "DML_READ",
          "ALL"
              ]
          }'
       )
    );
```

The example passes the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

> **CURRENT**
>
> MariaDB Enterprise Audit supports Object Filters for Query Events.
>
> **< 10.6 / 10.5.12 / 10.4.21**
>
> MariaDB Enterprise Audit does **not** support Object Filters for Query Events.

Table Events

MariaDB Enterprise Audit implements Table Events to audit when a table is accessed or modified.

MariaDB Enterprise Audit logs Table Events in the following situations:

- When an operation reads from a table, it is logged with the `READ` Event sub-class.
- When an operation writes to a table, it is logged with the `WRITE` Event sub-class.

- When an operation creates a table, it is logged with the `CREATE` Event sub-class.
- When an operation drops a table, it is logged with the `DROP` Event sub-class.
- When an operation alters a table, it is logged with the `ALTER` Event sub-class.
- When an operation renames a table, it is logged with the `RENAME` Event sub-class.

Table Events are logged when a table is accessed or modified directly or indirectly by a query. They complement Query Events very well, because the Query Event causes the raw query to be audit logged, and the Table Event causes all table operations to be audit logged. Both Query Events and Table Events are logged with the query ID, so each Table Event can easily be mapped to its corresponding Query Event. The combination of Query Events and Table Events can be useful when table operations can be hidden by views or triggers.

In the following example output, 6 sub-classes of Table Events are shown:

```
20190710 02:21:06,localhost.localdomain,John,localhost,3,25,CREATE,db1,services,
20190710 02:21:06,localhost.localdomain,John,localhost,3,27,READ,db1,services,
20190710 02:21:07,localhost.localdomain,John,localhost,3,29,WRITE,db1,services,
20190710 02:21:27,localhost.localdomain,John,localhost,3,35,ALTER,db1,services,
20190710 02:21:27,localhost.localdomain,John,localhost,3,36,RENAME,db1,services,db1,services_new
20190710 02:21:45,localhost.localdomain,John,localhost,3,38,DROP,db1,services_new,
```

When a query uses the `DELAYED` keyword, it is executed by a system user. In this case, any Table Event associated with the query is written to the audit log with the user set to `DELAYED`. However, the Query Event associated with the query is written to the audit log with the original user:

```
20190622 02:10:21,localhost.localdomain,root,localhost,8,5,QUERY,test,'INSERT DELAYED INTO t1 VALUES(1),
(2),(3);',0
20190622 02:10:25,localhost.localdomain,root,localhost,2,2,WRITE,test,t1,
20190622 02:10:25,localhost,DELAYED,localhost,2,2,WRITE,test,t1,
```

When your application executes queries with the `DELAYED` keyword, it is recommended to enable audit logging for Query Events in addition to Table Events to ensure that the full details are logged.

If the query cache is enabled, `READ` Table Events may not be audit logged. If MariaDB Enterprise Audit detects that the query cache is enabled during startup, MariaDB Enterprise Audit writes the following message to the MariaDB error log:

```
2021-07-23  0:11:26 server_audit: Query cache is enabled with the TABLE events. Some table reads can be
veiled.
```

An Event Filter for Table Events can be added to an Audit Filter with the `table_event` key, which supports the following values:

Value	Description
READ	Records read operations run on table objects, such as from a `SELECT` statement or an `INSERT SELECT`
WRITE	Records write operations run on table objects, such as `INSERT` or `UPDATE` statements
CREATE	Records any creation operations run on table objects, such as from a `CREATE TABLE` or `CREATE SERVER`.
DROP	Records any deletion operations run on table objects, such as `DELETE` or `DROP TABLE`.
ALTER	Records any modifications made on table objects, such as `ALTER TABLE` or `ALTER USER`.
RENAME	Records any renaming operations run on table objects, such as `RENAME TABLE`
ALL	Records all operations run on table objects

For example, the following query defines a Named Audit Filter that specifies Table Events:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES (
       'tables',
       JSON_COMPACT(
           '{
              "table_event": [
                  "WRITE",
                  "CREATE",
                  "DROP",
                  "RENAME",
                  "ALTER"
              ]
           }'
       )
    );
```

The example passes the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

> **CURRENT**
>
> MariaDB Enterprise Audit supports Object Filters for Table Events.
>
> **< 10.6 / 10.5.12 / 10.4.21**
>
> MariaDB Enterprise Audit does **not** support Object Filters for Table Events.

Logging Filter

MariaDB Enterprise Audit supports a special Logging Filter that can be used in Audit Filters to enable or disable audit logging for any users that are assigned the Audit Filter.

The Logging Filter can be added to an Audit Filter with the logging key, which supports the following values:

Value	Description
ON	Enables audit logging for this Audit Filter
OFF	Disables audit logging for this Audit Filter

For example, the following query defines a Default Audit Filter that enables logging for all Events for any user account without a Named Audit Filter:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES (
       'default',
       JSON_COMPACT(
           '{
              "logging": "ON"
           }'
       )
    );
```

The example passes the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

Object Filters

> **Note:**
>
> Object Filters are available from MariaDB 10.6, 10.5.12, and 10.4.21.

Object Filters allow Audit Filters to be limited to specific databases and/or tables. Object Filters can be specified at different scopes:

- An Object Filter can be specified for an entire Audit Filter
- An Object Filter can be specified for a single Event for the Audit Filter

Object Filter Format

Object Filters are formatted as JSON objects, which are key-value pairs.

For Object Filters, the key in the key-value pair refers to the specific type of Object Filter. The following types of Object Filters are supported:

Audit Log?	Object Filter Key	Description
No	ignore_databases	When one or more databases are specified with the `ignore_databases` Object Filter key, the specified databases are not audit-logged. The `ignore_databases` Object Filter key is an alias for the `ignore_tables` Object Filter key, with the table identifier set to the wildcard character (`*`). The ignore_databases Object Filter key cannot be specified in the same Object Filter as the `log_databases` and `log_tables` Object Filter keys.
No	ignore_tables	When one or more tables are specified with the `ignore_tables` Object Filter key, the specified tables will not be audit logged. Table names must be provided in the form *`database.table`*. Wildcard characters (`*`) are allowed. The `ignore_tables` Object Filter key cannot be specified in the same Object Filter as the `log_databases` and `log_tables` Object Filter keys.
Yes	log_databases	When one or more databases are specified with the `log_databases` Object Filter key, the specified databases will be audit logged, and all other databases will not be audit logged. The `log_databases` Object Filter key is an alias for the `log_tables` Object Filter key, with the table identifier set to the wildcard character (`*`).
Yes	log_tables	When one or more databases are specified with the `log_tables` Object Filter key, the specified tables will be audit logged, and all other tables will not be audit logged. Table names must be provided in the form *`database.table`*. Wildcard characters (`*`) are allowed.

The values in the key-value pair refer to object names.

When the Object Filter only applies to one object, the object name can be specified as a string scalar value in the JSON object:

```
{"object_filter_key": "object"}
```

When the Object Filter applies to multiple objects, the object names can be specified in a JSON array in the JSON object:

```
{"object_filter_key": [ "object", "object" ]}
```

In the following example, the Object Filter is designed to audit log the production and reporting databases and to skip audit logging for all other databases and tables:

```
{"log_tables": ["production.*", "reporting.*"]}
```

And in the following example, the Object Filter is designed to skip audit logging for the production.app_log table:

```
{"ignore_tables": "production.app_log"}
```

The Object Filter keys must be specified in different locations in the Audit Filter's JSON object, depending on the desired scope of the Object Filter. The following sections describe how to set Object Filters in more detail.

Set Object Filters at Event Scope

An Object Filter can be specified for a single Event Filter within an Audit Filter.

Object Filters are supported by the following Events:

- Query Events
- Table Events

To create an Object Filter for a single Event Filter, specify the Object Filter's JSON object as part of the JSON object for the Event Filter.

The examples below pass the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

In the following example, the reporting Audit Filter specifies an Event Filter for Table Events with an embedded Object Filter that includes all tables in the production and reporting databases:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES (
       'reporting',
       JSON_COMPACT(
           '{
              "table_event": [
                  "WRITE",
                  "CREATE",
                  "DROP",
                  "RENAME",
                  "ALTER",
                  {
                     "log_tables": [
                        "production.*",
                        "reporting.*"
                     ]
                  }
              ]
           }'
       )
    );
```

Set Object Filters at Audit Filter Scope

An Object Filter can be specified for an entire Audit Filter.

To create an Object Filter at Audit Filter scope, specify the Object Filter's JSON object as part of the root JSON object for the Audit Filter.

The examples below pass the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

In the following example, the reporting Audit Filter specifies an Object Filter that includes all tables in the production and reporting databases:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES (
       'reporting',
       JSON_COMPACT(
          '{
              "log_tables": [
                  "production.*",
                  "reporting.*"
              ]
          }'
       )
    );
```

When an Object Filter is specified at Audit Filter scope, it can contain embedded Event Filters. The following Event types support Object Filters:

- Query Events
- Table Events

In the following example, the reporting Audit Filter has been modified to include Event Filters on specific Query Event sub-classes and Table Event sub-classes:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES (
       'reporting',
       JSON_COMPACT(
          '{
              "log_tables": [
                  "production.*",
                  "reporting.*",
                  {
                     "query_event": [
                         "DML",
                         "DDL"
                     ],
                     "table_event": [
                         "WRITE",
                         "CREATE",
                         "DROP",
                         "RENAME",
                         "ALTER"
                     ]
                  }
              ]
          }'
       )
    );
```

Combine Object Filters at Audit Filter and Event Scope

When an Object Filter is specified at Audit Filter scope with embedded Event Filters, the embedded Event Filters can contain additional Object Filters. When you combine Object Filters at different levels, it is possible to combine Object Filters that are normally incompatible, like tables and ignore_tables.

To create Object Filters at Audit Filter and Event Type scope, specify the JSON object for the Object Filter at Audit Filter scope as part of the root JSON object for the Audit Filter and specify the JSON object for the Object Filter at Event Type scope as part of the JSON object for the Event Filter.

The examples below pass the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

In the following example, the reporting Audit Filter specifies an Event Filter for Table Events with an embedded Object Filter that includes all tables in the production and reporting databases, but it excludes Query Events that

target specific tables that store Personally Identifiable Information (PII), so that the sensitive information does not appear in the audit log:

```
INSERT INTO mysql.server_audit_filters (filtername, rule)
   VALUES (
       'reporting',
       JSON_COMPACT(
          '{
              "log_tables": [
                  "production.*",
                  "reporting.*",
                  {
                     "table_event": [
                         "WRITE",
                         "CREATE",
                         "DROP",
                         "RENAME",
                         "ALTER"
                     ],
                     "query_event": [
                         "DML",
                         "DDL",
                         {
                             "ignore_tables": [
                                 "production.customer_profiles",
                                 "production.customer_addresses"
                             ]
                         }
                     ]
                  }
              ]
          }'
       )
    );
```

Audit Log Destinations

MariaDB Enterprise Audit writes audit log messages either to a dedicated audit log file or to the system log (syslog), depending on configuration.

The audit log destination is configured with the dit_output_type|server_audit_output_type system variable:

Value	Description
FILE	Audit log messages are written to a dedicated file.
SYSLOG	Audit log messages are written to the system log (syslog).

Audit Logging to File

MariaDB Enterprise Audit writes audit log messages to a dedicated audit log file when the dit_output_type|server_audit_output_type system variable is set to FILE.

Audit Log Path

The path to the dedicated audit log file is configured with the server_audit_file_path system variable. The path can be a relative or absolute path. If it is a relative path, then it will be relative to the datadir. For example, to set the path to mariadb-enterprise-audit.log with the SET GLOBAL statement:

```
SET GLOBAL server_audit_file_path = 'mariadb-enterprise-audit.log'
```

When a system variable is dynamically changed with the SET GLOBAL statement, the change does not survive server restarts. To ensure that the new path is used when the server restarts, set the server_audit_file_path system

variable in a configuration file:

```
[mariadb]
server_audit_file_path=mariadb-enterprise-audit.log
```

Audit Log Rotation

When MariaDB Enterprise Audit is configured to use the dedicated audit log file, it can rotate the file.

The file is rotated when its size exceeds the size specified by the server_audit_file_rotate_size system variable. For example, to set the maximum log size to 2 GB with the SET GLOBAL statement:

```
SET GLOBAL server_audit_file_rotate_size = 2 * (1024 * 1024 * 1024);
```

When a system variable is dynamically changed with the SET GLOBAL statement, the change does not survive server restarts. To ensure that the new file rotation size is used when the server restarts, set the server_audit_file_rotate_size system variable in a configuration file:

```
[mariadb]
server_audit_file_rotate_size=2147483648
```

The file can also be rotated manually by setting the server_audit_file_rotate_now system variable to `ON`. For example, to rotate the log with the SET GLOBAL statement:

```
SET GLOBAL server_audit_file_rotate_now = ON;
```

Audit Logging to System Log

MariaDB Enterprise Audit writes audit log messages to the system log (syslog) when the server_audit_output_type system variable is set to SYSLOG.

It can be very useful to configure audit logging to the syslog when your system's syslog is configured to securely transmit logs to a remote syslog server.

System Log Parameters

Several syslog parameters can be changed for MariaDB Enterprise Audit by setting the following system variables:

- server_audit_syslog_facility
- server_audit_syslog_ident
- server_audit_syslog_info
- server_audit_syslog_priority

Audit Log Format

The audit log format for MariaDB Enterprise Audit depends on the audit log destination.

Formal Specification

When MariaDB Enterprise Audit is configured to use a dedicated file, it records events in a comma-separated (CSV) format. For tool developers, it is critical to use standardized field mappings to correlate audit data with other server logs.

Template: `<timestamp>,<serverhost>,<username>,<host>:<port>,<connectionid>,<queryid>,<operation>,<database>,<object>,<retcode>,<tlsversion>`

Field	Component	Data Type	Standardized Name / Description
1	`timestamp`	`DateTime`	Formatted as `%Y%m%d %H:%i:%s` (the default), or as described below.
2	`serverhost`	`String`	The hostname of the server instance.
3	`username`	`String`	The MariaDB user account triggering the event.
4	`host:port`	`String`	The client host from which the user connected, followed by a colon and the client's TCP port. The port is written as `unavailable` when the client did not connect over TCP/IP, such as over a Unix socket or a named pipe. The port was added in MariaDB Enterprise Server 12.3.
5	`connectionid`	`Unsigned Int`	Standardized: Thread ID. Matches the `Thread ID` in Error, General, and Slow logs.
6	`queryid`	`Unsigned Int`	A unique identifier for the specific query. Used to link Query and Table events.
7	`operation`	`String`	The type of action (e.g., `CONNECT`, `QUERY`, `WRITE`).
8	`database`	`String`	The name of the database being accessed.
9	`object`	`String`	The specific table or object involved in the operation.
10	`retcode`	`Integer`	The return code; `0` indicates success, non-zero indicates an error code.
11	`tlsversion`	`String`	The TLS version negotiated for the connection, such as `TLSv1.3`. Present only on connection events, and empty when the connection is not encrypted. Added in MariaDB Enterprise Server 12.3.

Warning:

Changed in MariaDB Enterprise Server 12.3

Two changes to the log format require updates to tools that parse the audit log:

- The `host` field now contains a colon and the client port.
- Connection event records carry an eleventh field. Records for all other operations still end at `retcode`.

Milliseconds Precision Timestamps

Auditing regulations and standards require auditing logs with timestamps recording fractions of a second and timezone information. To fulfill this requirement, MariaDB Enterprise Audit uses the system log (syslog) via the server_audit_output_type option, where the timestamp format is controlled by the syslog setting.

To define the format of the timestamp:

- Configure `server_audit_output_type=file`.
- Change the default setting of `%Y%m%d %H:%i:%s` (which assures compatibility) to something including milliseconds (`%f`) and time zone information (`%z`), like `%Y%m%dT%H:%i:%s.%f%z`.

This results in audit log entries like these:

```
# for server_audit_timestamp_format=%Y%m%dT%H:%i:%s.%f%z
20231207T13:14:28.000000+0100,mdbe106,root,localhost,4,0,CONNECT,,,0
20231207T13:14:32.481161+0100,mdbe106,root,localhost,4,0,DISCONNECT,,,0
```

Value Mapping for Operations

To build accurate filters and parsers, the `<operation>` field corresponds to the following standardized event types:

Operation	Triggered By
`CONNECT`	Successful user authentication and connection.
`FAILED_CONNECT`	Authentication failures or denied access.
`DISCONNECT`	Session termination.
`QUERY`	Direct execution of a SQL statement.
`READ`	Table read access (e.g., `SELECT`).
`WRITE`	Table modification (e.g., `INSERT`, `UPDATE`, `DELETE`).
`AUDIT_CONFIG`	Changes to audit settings via `SET GLOBAL` or log rotation.

Audit Log Format with Syslog

When configured to use `SYSLOG`, the standard CSV line is prefixed with syslog metadata: `<timestamp> <syslog_host> <syslog_ident>: <syslog_info> [Standard CSV Fields]`

Messages in MariaDB Error Log

MariaDB Enterprise Audit writes messages to the MariaDB Error Log in various scenarios. Some of the scenarios and log messages are described below.

Load Plugin

When MariaDB Enterprise Server loads the plugin for MariaDB Enterprise Audit, MariaDB Enterprise Audit writes the following message in the MariaDB error log:

```
2021-08-03 21:07:03 server_audit: MariaDB Audit Plugin version 2.0.3 STARTED.
```

For additional information, see “Load the Audit Plugin”.

Unload Plugin

When MariaDB Enterprise Server unloads the plugin for MariaDB Enterprise Audit, MariaDB Enterprise Audit writes the following message in the MariaDB error log:

```
2021-08-02 23:54:59 server_audit: STOPPED
```

The plugin is unloaded when MariaDB Enterprise Server is shutdown, so this message is most commonly written to the MariaDB error log during the shutdown process.

Start Audit Logging to File

When audit logging is started and it is directed to a file, MariaDB Enterprise Audit writes the following message in the MariaDB error log:

```
2021-08-03 21:39:42 server_audit: logging started to the file server_audit.log.
```

If a custom audit log path is configured, then the message will refer to the custom path.

For additional information, see “Start Audit Logging” and “Audit Logging to File”.

Start Audit Logging to Syslog

When audit logging is started and it is directed to syslog, MariaDB Enterprise Audit writes the following message in the MariaDB error log:

For additional information, see “Start Audit Logging” and “Audit Logging to Syslog”.

```
2021-08-03 22:02:45 server_audit: logging started to the syslog.
```

Stop Audit Logging

When audit logging is stopped, MariaDB Enterprise Audit writes the following message in the MariaDB error log:

```
2021-08-03 21:39:50 server_audit: logging was stopped.
```

Change Audit Logging to File

When audit logging is changed to a file, MariaDB Enterprise Audit writes the following message in the MariaDB error log:

```
2021-08-03 22:03:31 server_audit: Output was redirected to 'file'
```

For additional information, see “Audit Logging to File”.

Change Audit Logging to Syslog

When audit logging is changed to syslog, MariaDB Enterprise Audit writes the following message in the MariaDB error log:

```
2021-08-03 22:01:22 server_audit: Output was redirected to 'syslog'
```

For additional information, see “|mariadb-enterprise-audit/#audit-logging-to-system-logAudit Logging to Syslog”.

Change File Name for Audit Logging

When the file name for audit logging is changed, MariaDB Enterprise Audit writes the following message in the MariaDB error log:

```
2021-08-03 22:05:17 server_audit: Log file name was changed to 'mariadb-enterprise-audit.log'.
```

For additional information, see “Audit Log Path”.

Reload Invalid Audit Filters

When the Audit Filters are reloaded and one or more of the Audit Filters are invalid, MariaDB Enterprise Audit writes the following message in the MariaDB error log:

```
2021-08-03 21:51:55 server_audit: Unknown filter function tables.
2021-08-03 21:51:55 server_audit: Can't parse filter's 'production' definition { "tables": "production.*"
}.
2021-08-03 21:51:55 server_audit: can't load filters - old filters are saved.
```

For additional information, see “Reload Audit Filters and Assignments”.

Conflict with the Query Cache

If the query cache is enabled, `READ` Table Events may not be audit logged. If MariaDB Enterprise Audit detects during startup that the query cache is enabled, MariaDB Enterprise Audit writes the following message to the MariaDB error log:

```
2021-08-03 21:07:03 server_audit: Query cache is enabled with the TABLE events. Some table reads can be
veiled.
```

Upgrades

MariaDB Enterprise Audit is included with MariaDB Enterprise Server. Special consideration is needed when upgrading from MariaDB releases that include the MariaDB Audit Plugin, including MariaDB Community.

For details on how to upgrade from the MariaDB Audit Plugin to MariaDB Enterprise Audit, see the sections below.

Warning:

Before upgrading the package: Remove the v1 plugin **and** every `server_audit_*` variable setting from your configuration files *before* running `apt upgrade` (or your platform's equivalent). The new Enterprise Server does not recognize the v1 variables and will fail to start if they remain. Follow the pre-upgrade cleanup steps below.

Important:

Migrating from MariaDB Audit Plugin (v1)

MariaDB Enterprise Audit (`server_audit2.so`) is a different and incompatible plugin from the older MariaDB Audit Plugin (`v1` , `server_audit.so`) found in Community Server or older Enterprise Server versions.

The `v1` plugin is configured with server variables (e.g., `server_audit_events`).

The Enterprise (`v2`) plugin is configured with system tables (e.g., `mysql.server_audit_filters`).

You must uninstall the `v1` plugin before using the `v2` plugin, as running both can cause server instability or deadlocks (MENT-545). The following sections guide you through the full migration process, starting with checking for and removing the old plugin.

Check for and Uninstall the v1 Plugin

Confirm the MariaDB Audit Plugin is Loaded

To confirm that the MariaDB Audit plugin is loaded, query the information_schema.PLUGINS table:

```
SELECT PLUGIN_STATUS, PLUGIN_LIBRARY, PLUGIN_DESCRIPTION
FROM information_schema.PLUGINS
WHERE PLUGIN_NAME='SERVER_AUDIT'\G
```

```
*************************** 1. row ***************************
     PLUGIN_STATUS: ACTIVE
    PLUGIN_LIBRARY: server_audit.so
PLUGIN_DESCRIPTION: Audit the server activity
```

If you see the output shown above, then the MariaDB Audit plugin is installed, and it must be uninstalled prior to performing the upgrade or migration. Follow the instructions in the section below.

If you do not see any rows in the output, then the MariaDB Audit plugin is not installed.

Determine Uninstallation Method

The MariaDB Audit plugin has multiple uninstallation methods. You must choose the uninstallation method that corresponds to how the plugin was installed on your system.

To determine the uninstallation method, query the mysql.plugin system table:

```
SELECT *
FROM mysql.plugin
WHERE name = 'SERVER_AUDIT'\G
```

```
*************************** 1. row ***************************
name: SERVER_AUDIT
  dl: server_audit.so
```

If you see the output shown above, then the MariaDB Audit plugin can be uninstalled with UNINSTALL SONAME.

If you do not see any rows in the output, then the MariaDB Audit plugin call be uninstalled by editing the configuration file.

Uninstall with UNINSTALL SONAME

To uninstall the MariaDB Audit Plugin with UNINSTALL SONAME:

Step 1

Check the plugin load option by querying the information_schema.PLUGINS table:

```
SELECT PLUGIN_STATUS, PLUGIN_LIBRARY, PLUGIN_DESCRIPTION, LOAD_OPTION
FROM information_schema.PLUGINS
WHERE PLUGIN_NAME='SERVER_AUDIT'\G
```

```
*************************** 1. row ***************************
     PLUGIN_STATUS: ACTIVE
    PLUGIN_LIBRARY: server_audit.so
PLUGIN_DESCRIPTION: Audit the server activity
       LOAD_OPTION: FORCE_PLUS_PERMANENT
```

Step 2

Check `LOAD_OPTION` column for `FORCE_PLUS_PERMANENT`

> **Note:**
>
> If the `LOAD_OPTION` column does not contain the value `FORCE_PLUS_PERMANENT`, then you can skip to step 5, which executes the UNINSTALL SONAME statement.

If the `LOAD_OPTION` column contains the value `FORCE_PLUS_PERMANENT`, then check your configuration files for the server-audit option:

```
$ grep --extended-regexp --with-filename \
   'server[-_]audit[[:blank:]]*=' \
   /etc/mysql/my.cnf \
   /etc/mysql/mariadb.conf.d/*
```

```
/etc/mysql/mariadb.conf.d/enable-audit.cnf:server_audit=FORCE_PLUS_PERMANENT
```

Step 3

If the server-audit option was found in a configuration file, then remove or comment the option out:

```
[mariadb]
# server_audit=FORCE_PLUS_PERMANENT
```

Step 4

If the configuration file was changed, then restart the server:

```
$ sudo systemctl restart mariadb
```

Step 5

Uninstall the plugin by executing the UNINSTALL SONAME statement:

```
UNINSTALL SONAME 'server_audit';
```

Step 6

Confirm the plugin is uninstalled by querying the mysql.plugin system table:

```
SELECT *
FROM mysql.plugin
WHERE name = 'SERVER_AUDIT'\G
```

If the query returns no results, then the plugin has been uninstalled.

Uninstall with Configuration File

To uninstall the MariaDB Audit plugin with a configuration file:

Step 1

Check your configuration files for the `plugin_load_add` option:

```
$ grep --extended-regexp --with-filename \
   'plugin[-_]load[-_]add[[:blank:]]*=[[:blank:]]*server_audit' \
   /etc/mysql/my.cnf \
   /etc/mysql/mariadb.conf.d/*
```

```
/etc/mysql/mariadb.conf.d/enable-audit.cnf:plugin_load_add=server_audit
```

Step 2

Remove or comment out the plugin_load_add option from the configuration file:

```
[mariadb]
# plugin_load_add=server_audit
```

Remove v1 Variable Settings

After uninstalling the plugin, also remove every `server_audit_*` system variable from your configuration files. If any of these settings remain, the new Enterprise Server will fail to start because it does not recognize them.

```
$ grep --extended-regexp --with-filename \
   'server_audit[_a-z]*[[:blank:]]*=' \
   /etc/mysql/my.cnf \
   /etc/mysql/mariadb.conf.d/*
```

Comment out or delete every matching line. Common variables include `server_audit_events`, `server_audit_logging`, `server_audit_incl_users`, `server_audit_excl_users`, `server_audit_file_path`, `server_audit_output_type`, `server_audit_query_log_limit`, and `server_audit_file_rotate_size`.

Migrate v1 Settings to Enterprise Audit (v2)

Update System Tables

After upgrading to MariaDB Enterprise Server, execute `mariadb-upgrade` to create the System Tables for Audit Filters.

> **Note:**
>
> If `mariadb-upgrade` reports that the installation is already up to date — for example, when upgrading from Community Server 10.6 to Enterprise Server 10.6 — the audit-filter system tables will not be created. In that case, run `mariadb-upgrade --force` to force the upgrade scripts to run anyway. See mariadb-upgrade for the full behaviour.

Migrate Audit Filters

The MariaDB Audit Plugin defines Audit Filters using the server_audit_events system system variable, but MariaDB Enterprise Audit defines Audit Filters using the mysql.server_audit_filters system table.

If you are upgrading from the MariaDB Audit Plugin to MariaDB Enterprise Audit, perform the following procedure:

Step 1

Remove or comment out lines involving the server_audit_events system variable from the configuration file:

```
[mariadb]
...
# server_audit_events=CONNECT,QUERY
```

Step 2

Insert a replacement Audit Filter into the `mysql.server_audit_filters` system table:

```
INSERT INTO mysql.server_audit_filters
   VALUES ('default',
      JSON_COMPACT(
         '{
            "connect_event": "ALL",
            "query_event": "ALL"
         }'
      ));
```

The example passes the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

Migrate Users

The MariaDB Audit Plugin enables or disable audit logging for specific user accounts using the server_audit_incl_users and server_audit_excl_users system variables, but MariaDB Enterprise Audit uses the mysql.server_audit_users system table.

If you are upgrading from the MariaDB Audit Plugin to MariaDB Enterprise Audit, perform the following procedure:

Step 1

Edit configuration file

Remove or comment out lines involving the server_audit_incl_users and server_audit_excl_users system variables from the configuration file:

```
[mariadb]
...
# server_audit_incl_users = root,app
# server_audit_excl_users = backup_user,monitor_user
```

Step 2

For the server_audit_incl_users system variable

For any user account previously mentioned in the server_audit_incl_users system variable, determine if the user account can use the Default Audit Filter or if the user account requires a Named Audit Filter.

Insert the relevant Audit Filters into the `mysql.server_audit_filters` system table, and insert the user assignments into the `mysql.server_audit_users` system table:

```
INSERT INTO mysql.server_audit_filters
   VALUES
   ('default',
      JSON_COMPACT(
         '{
            "connect_event": [
               "CONNECT",
               "DISCONNECT"
            ],
            "query_event": [
               "DML",
               "DDL"
            ]
         }'
   )),
   ('root_filter',
      JSON_COMPACT(
         '{
            "logging": "ON"
         }'
   ));
INSERT INTO mysql.server_audit_users (host, user, filtername)
   VALUES ('%', 'root', 'root_filter');
```

The example passes the JSON object to the JSON_COMPACT() function, so that the JSON object is compacted prior to being inserted into the system table. This step is recommended, but not required.

Step 3

For the server_audit_excl_users system variable

For any user account previously mentioned in the server_audit_excl_users system variable, create a Named Audit Filter that acts as an exclusion filter.

Insert the relevant Audit Filters into the `mysql.server_audit_filters` system table, and insert the user assignments into the `mysql.server_audit_users` system table:

```
INSERT INTO mysql.server_audit_filters
   VALUES ('exclusion_filter',
      JSON_COMPACT(
         '{
            "logging": "OFF"
         }'
      ));

INSERT INTO mysql.server_audit_users (host, user, filtername)
   VALUES
   ('%', 'backup_user', 'exclusion_filter'),
   ('%', 'monitor_user', 'exclusion_filter');
```

This page is: Copyright © 2026 MariaDB. All rights reserved.

MariaDB Community Audit Plugin

See Audit Plugin Overview.

The MariaDB Audit Plugin records server activity, including connections, queries, and table access, to help meet organizational auditing and compliance regulations.

See Audit Plugin Installation.

Follow this guide to install the Audit Plugin on your MariaDB server. Learn how to verify the plugin file's location, load it dynamically, or configure it to load automatically at startup.

See Audit Plugin Configuration.

Configure the Audit Plugin to suit your monitoring requirements. Learn how to enable logging, select specific event types to record, and exclude specific users from the audit trail.

See Audit Plugin Log Settings.

Control where and how the audit data is stored. This section explains how to direct output to a file or the system syslog, and how to configure logging parameters for different environments.

See Audit Plugin Location and Rotation of Logs.

Manage your audit log files effectively. Learn how to define the log file path, set size limits, and configure rotation strategies to prevent log files from consuming all available disk space.

See Audit Plugin Log Format.

Understand the structure of audit log entries. This guide breaks down the fields in the log records, including timestamps, server IDs, user details, and the specific operations performed.

See Audit Plugin Options and System Variables.

Browse the complete reference of system variables for the Audit Plugin. Use these settings to fine-tune logging behavior, control performance impact, and manage log file handling.

See Audit Plugin Status Variables.

Monitor the performance and status of the Audit Plugin. View variables that track the number of logged events and current settings to ensure the auditing system is functioning correctly.

See Audit Plugin Versions.

Review the version history of the MariaDB Audit Plugin. Check compatibility with different MariaDB Server releases and identify which features or bug fixes are included in each version.

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Overview

MariaDB and MySQL are used in a broad range of environments, but if you needed to record user access to be in compliance with auditing regulations for your organization, you would previously have had to use other database solutions. To meet this need, though, MariaDB has developed the MariaDB Audit Plugin. Although the MariaDB Audit Plugin has some unique features available only for MariaDB, it can be used also with MySQL.

Basically, the purpose of the MariaDB Audit Plugin is to log the server's activity. For each client session, it records who connected to the server (i.e., user name and host), what queries were executed, and which tables were accessed and server variables that were changed. This information is stored in a rotating log file or it may be sent to the local `syslogd`.

Review these pages for detailed documentation:

- Installation
- Configuration
- Log Settings
- Log Location & Rotation
- Log Format
- Status Variables
- System Variables

Tutorials

- Activating MariaDB Audit Log by Jaykishan Mutkawoa, May 30, 2016
- Installing MariaDB Audit Plugin on Amazon RDS
  Amazon RDS supports using the MariaDB Audit Plugin on MySQL and MariaDB database instances.

Blog Posts

- MySQL Auditing with MariaDB Auditing Plugin by Peter Zaitsev, February 15, 2016

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Installation

The `server_audit` plugin logs the server's activity. For each client session, it records who connected to the server (i.e., user name and host), what queries were executed, and which tables were accessed and server variables that were changed. This information is stored in a rotating log file or it may be sent to the local syslogd.

Locating the Plugin

The `server_audit` plugin's shared library is included in MariaDB packages as the `server_audit.so` or `server_audit.dll` shared library on systems where it can be built.

The plugin must be located in the plugin directory, the directory containing all plugin libraries for MariaDB. The path to this directory is configured by the plugin_dir system variable. To see the value of this variable and thereby

determine the file path of the plugin library, execute the following SQL statement:

```
SHOW GLOBAL VARIABLES LIKE 'plugin_dir';

+---------------+--------------------------+
| Variable_name | Value                    |
+---------------+--------------------------+
| plugin_dir    | /usr/lib64/mysql/plugin/ |
+---------------+--------------------------+
```

Check the directory returned at the filesystem level to make sure that you have a copy of the plugin library, `server_audit.so` or `server_audit.dll`, depending on your system. It's included in recent installations of MariaDB. If you do not have it, you should upgrade MariaDB.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'server_audit';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to `mariadbd`, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = server_audit
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'server_audit';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Prohibiting Uninstallation

The UNINSTALL SONAME or UNINSTALL PLUGIN statements may be used to uninstall plugins. For the `server_audit` plugin, you might want to disable this capability. To prevent the plugin from being uninstalled, you could set the server_audit option to `FORCE_PLUS_PERMANENT` in a relevant server option group in an option file after the plugin is loaded once:

```
[mariadb]
...
plugin_load_add = server_audit
server_audit=FORCE_PLUS_PERMANENT
```

Once you've added the option to the server's option file and restarted the server, the plugin can't be uninstalled. If someone tries to uninstall the audit plugin, then an error message will be returned. Below is an example of the error message:

```
UNINSTALL PLUGIN server_audit;

ERROR 1702 (HY000):
Plugin 'server_audit' is force_plus_permanent and can not be unloaded
```

For more information on `FORCE_PLUS_PERMANENT` and other option values for the server_audit option, see Plugin Overview: Configuring Plugin Activation at Server Startup for more information.

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Configuration

After the audit plugin has been installed and loaded, there will be some new global variables within MariaDB. These can be used to configure many components, limits, and methods related to auditing the server. You may set these variables related to the logs, such as their location, size limits, rotation parameters, and method of logging information. You may also set what information is logged, such connects, disconnects, and failed attempts to connect. You can also have the audit plugin log queries, read and write access to tables. So as not to overload your logs, the audit plugin can be configured based on lists of users. You can include or exclude the activities of specific users in the logs.

To see a list of audit plugin-related variables on the server and their values, execute the follow while connected to the server:

CURRENT

```
SHOW GLOBAL VARIABLES LIKE 'server_audit%';
+-------------------------------+-----------------------+
| Variable_name                 | Value                 |
+-------------------------------+-----------------------+
| server_audit_events           |                       |
| server_audit_excl_users       |                       |
| server_audit_file_buffer_size | 0                     |
| server_audit_file_path        | server_audit.log      |
| server_audit_file_rotate_now  | OFF                   |
| server_audit_file_rotate_size | 1000000               |
| server_audit_file_rotations   | 9                     |
| server_audit_incl_users       |                       |
| server_audit_logging          | OFF                   |
| server_audit_mode             | 0                     |
| server_audit_output_type      | file                  |
| server_audit_query_log_limit  | 1024                  |
| server_audit_sync_log_file    | OFF                   |
| server_audit_syslog_facility  | LOG_USER              |
| server_audit_syslog_ident     | mysql-server_auditing |
| server_audit_syslog_info      |                       |
| server_audit_syslog_priority  | LOG_INFO              |
+-------------------------------+-----------------------+
17 rows in set (0.003 sec)
```

< 12.1

```
SHOW GLOBAL VARIABLES LIKE 'server_audit%';

+-------------------------------+-----------------------+
| Variable_name                 | Value                 |
+-------------------------------+-----------------------+
| server_audit_events           | CONNECT,QUERY,TABLE   |
| server_audit_excl_users       |                       |
| server_audit_file_path        | server_audit.log      |
| server_audit_file_rotate_now  | OFF                   |
| server_audit_file_rotate_size | 1000000               |
| server_audit_file_rotations   | 9                     |
| server_audit_incl_users       |                       |
| server_audit_logging          | ON                    |
| server_audit_mode             | 0                     |
| server_audit_output_type      | file                  |
| server_audit_query_log_limit  | 1024                  |
| server_audit_syslog_facility  | LOG_USER              |
| server_audit_syslog_ident     | mysql-server_auditing |
| server_audit_syslog_info      |                       |
| server_audit_syslog_priority  | LOG_INFO              |
+-------------------------------+-----------------------+
```

The values of these variables can be changed by an administrator with the `SUPER` privilege, using the SET statement. Below is an example of how to disable audit logging:

```
SET GLOBAL server_audit_logging=OFF;
```

Although it is possible to change all of the variables shown above, some of them may be reset when the server restarts. Therefore, you may want set them in the configuration file (e.g., `/etc/my.cnf.d/server.cnf`) to ensure the values are the same after a restart:

```
[server]
...
server_audit_logging=OFF
…
```

For the reason given in the paragraph above, you would not generally set variables related to the auditing plugin using the `SET` statement. However, you might do so to test settings before making them more permanent. Since one cannot always restart the server, you would use the `SET` statement to change immediately the variables and then include the same settings in the configuration file so that the variables are set again as you prefer when the server is restarted.

Configuring Logs and Setting Other Variables

Of all of the server variables you can set, you may want to set initially the server_audit_events variable to tell the Audit Plugin which events to log. The Log Settings documentation page describes in detail the choices you have and provides examples of log entries related to them.

You can see a detailed list of system variables related to the MariaDB Audit Plugin on the System Variables documentation page. Status variables related to the Audit Plugin are listed and explained on the Status Variables documentation page.

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Log Settings

Events that are logged by the MariaDB Audit Plugin are grouped generally into different types: connect, query, and table events. To log based on these types of events, set the variable, server_audit_events to `CONNECT`, `QUERY`, or `TABLE`. To have the Audit Plugin log more than one type of event, put them in a comma-separated list like so:

```
SET GLOBAL server_audit_events = 'CONNECT,QUERY,TABLE';
```

You can put the equivalent of this in the configuration file like so:

```
[mysqld]
...
server_audit_events=connect,query
```

By default, logging is set to `OFF`. To enable it, set the server_audit_logging variable to `ON`. Note that if the query cache is enabled, and a query is returned from the query cache, no `TABLE` records will appear in the log since the server didn't open or access any tables and instead relied on the cached results. So you may want to disable query caching.

There are a few types of events that may be logged, not just the three common ones mentioned above. A full list of related system variables is detailed on the Server_Audit System Variables page, and status variables on the Server_Audit System Variables page of this documentation. Some of the major ones are highlighted below:

Type	Description
CONNECT	Connects, disconnects and failed connects—including the error code.
QUERY	Queries executed and their results in plain text, including failed queries due to syntax or permission errors.
TABLE	Tables affected by query execution.
QUERY_DDL	Similar to `QUERY`, but filters only DDL-type queries (`CREATE`, `ALTER`, `DROP`, `RENAME` and `TRUNCATE`). There are some exceptions however. `RENAME USER` is not logged, while `CREATE/DROP [PROCEDURE / FUNCTION / USER]` are logged.
QUERY_DML	Similar to `QUERY`, but filters only DML-type queries (`DO`, `CALL`, `LOAD DATA/XML`, `DELETE`, `INSERT`, `SELECT`, `UPDATE`, `HANDLER`, and `REPLACE` statements).
QUERY_DML_NO_SELECT	Similar to `QUERY_DML`, but doesn't log `SELECT` queries.
QUERY_DCL	Similar to `QUERY`, but filters only DCL-type queries (`CREATE USER`, `DROP USER`, `RENAME USER`, `GRANT`, `REVOKE` and `SET PASSWORD` statements).

Since there are other types of queries besides DDL and DML, using the `QUERY_DDL` and `QUERY_DML` options together is not equivalent to using `QUERY`. There is the `QUERY_DCL` option for logging DCL types of queries (e.g., and `GRANT``REVOKE` statements). In the same version, the server_audit_query_log_limit variable was added to be able to set the length of a log record. Previously, a log entry would be truncated due to long query strings.

Logging Connect Events

If the Audit Plugin has been configured to log connect events, it will log connects, disconnects, and failed connects. For a failed connection, the log includes the error code.

It's possible to define a list of users for which events can be excluded or included for tracing their database activities. This list will be ignored, though, for the logging of connect events. This is because auditing standards distinguish between technical and physical users. Connects need to be logged for all types of users; access to objects needs to be logged only for physical users.

Logging Query Events

If `QUERY`, `QUERY_DDL`, `QUERY_DML`, `QUERY_DML_NO_SELECT`, and/or `QUERY_DCL` event types are enabled, then the corresponding types of queries that are executed will be logged for defined users. The queries will be logged exactly as they are executed, in plain text. This is a security vulnerability: anyone who has access to the log files will be able to read the queries. So make sure that only trusted users have access to the log files and that the files are in a protected location. An alternative is to use the `TABLE` event type instead of the query-related event types.

Queries are also logged if they cannot be executed or if they're unsuccessful. For example, a query will be logged because of a syntax error or because the user doesn't have the privileges necessary to access an object. These queries can be parsed by the error code that's provided in the log.

You may find failed queries to be more interesting: They can reveal problems with applications (e.g., an SQL statement in an application that doesn't match the current schema). They can also reveal if a malicious user is guessing at the names of tables and columns to try to get access to data.

Below is an example in which a user attempts to execute an `UPDATE` statement on a table for which he does not have permission:

```
UPDATE employees
SET salary = salary * 1.2
WHERE emp_id = 18236;

ERROR 1142 (42000):
UPDATE command denied to user 'bob'@'localhost' for table 'employees'
```

Looking in the Audit Plugin log (`server_audit.log`) for this entry, you can see the following entry:

```
20170817 11:07:18,ip-172-30-0-38,bob,localhost,15,46,QUERY,company,'UPDATE employees SET salary = salary
* 1.2 WHERE emp_id = 18236',1142
```

This log entry contains the date and time of the query, followed by the server host, and the user and host for the account.

> **Note:**
>
> From MariaDB 12.0, in addition to the host, the audit log also contains the port, where applicable (for instance, connecting via a Unix socket doesn't use a port).

Next is the connection and query identification numbers (i.e., `15` and `46`). After the log event type (i.e., `QUERY`), the database name (i.e., `company`), the query, and the error number are recorded.

Notice that the last value in the log entry is `1142` . That's the error number for the query. To find failed queries, you would look for two elements: the notation indicating that it's an entry `QUERY` and the last value for the entry. If the query is successful, the value will be `0` .

Queries Not Included in Subordinate Query Event Types

Note that the `QUERY` event type will log queries that are not included in any of the subordinate `QUERY_*` event types, such as:

- CREATE FUNCTION
- DROP FUNCTION
- CREATE PROCEDURE
- DROP PROCEDURE
- SET
- CHANGE MASTER TO
- FLUSH
- KILL
- CHECK
- OPTIMIZE
- LOCK
- UNLOCK

- ANALYZE
- INSTALL PLUGIN
- UNINSTALL PLUGIN
- INSTALL SONAME
- UNINSTALL SONAME
- EXPLAIN

Logging Table Events

MariaDB has the ability to record table events in the logs—this is not a feature of MySQL. This feature is the only way to log which tables have been accessed through a view, a stored procedure, a stored function, or a trigger. Without this feature, a log entry for a query shows only the view, stored procedure or function used, not the underlying tables. Of course, you could create a custom application to parse each query executed to find the SQL statements used and the tables accessed, but that would be a drain on system resources. Table event logging is much simpler: it adds a line to the log for each table accessed, without any parsing. It includes notes as to whether it was a read or a write.

If you want to monitor user access to specific databases or tables (e.g., `mysql.user`), you can search the log for them. Then if you want to see a query which accessed a certain table, the audit log entry will include the query identification number. You can use it to search the same log for the query entry. This can be useful when searching a log containing tens of thousands of entries.

Because of the `TABLE` option, you may disable query logging and still know who accessed which tables. You might want to disable `QUERY` event logging to prevent sensitive data from being logged. Since *table* event logging will log who accessed which table, you can still watch for malicious activities with the log. This is often enough to fulfill auditing requirements.

Below is an example with both `TABLE` and `QUERY` events logging. For this scenario, suppose there is a VIEW in which columns are selected from a few tables in a `company` database. The underlying tables are related to sensitive employee information, in particular salaries. Although we may have taken precautions to ensure that only certain user accounts have access to those tables, we will monitor the Audit Plugin logs for anyone who queries them—directly or indirectly through a view.

```
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,READ,company,employees,
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,READ,company,employees_salaries,
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,READ,company,ref_job_titles,
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,READ,company,org_departments,
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,QUERY,company,
'SELECT * FROM employee_pay WHERE title LIKE \'%Executive%\'  OR title LIKE \'%Manager%\'',0
```

Although the user executed only one SELECT statement, there are multiple entries to the log: one for each table accessed and one entry for the query on the view, (i.e., `employee_pay`). We know primarily this is all for one query because they all have the same connection and query identification numbers (i.e., `29` and `913`).

Logging User Activities

The Audit Plugin will log the database activities of all users, or only the users that you specify. A database activity is defined as a *query* event or a *table* event. *Connect* events are logged for all users.

You may specify users to include in the log with the `server_audit_incl_users` variable or exclude users with the `server_audit_excl_users` variable. This can be useful if you would like to log entries, but are not interested in entries from trusted applications and would like to exclude them from the logs.

You would typically use either the `server_audit_incl_users` variable or the `server_audit_excl_users` variable. You may, though, use both variables. If a username is inadvertently listed in both variables, database activities for that user will be logged because `server_audit_incl_users` takes priority.

Although MariaDB considers a user as the combination of the username and hostname, the Audit Plugin logs only based on the username. MariaDB uses both the username and hostname so as to grant privileges relevant to the location of the user. Privileges are not relevant though for tracing the access to database objects. The host name is still recorded in the log, but logging is not determined based on that information.

The following example shows how to add a new username to the `server_audit_incl_users` variable without removing previous usernames:

```
SET GLOBAL server_audit_incl_users = CONCAT(@@global.server_audit_incl_users, ',Maria');
```

Remember to add also any new users to be included in the logs to the same variable in MariaDB configuration file. Otherwise, when the server restarts it will discard the setting.

Excluding or Including Users

By default events from all users are logged, but certain users can be excluded from logging by using the `server_audit_excl_users` variable. For example, to exclude users *valerianus* and *rocky* from having their events logged:

```
server_audit_excl_users=valerianus,rocky
```

This option is primarily used to exclude the activities of trusted applications.

Alternatively, `server_audit_incl_users` can be used to specifically include users. Both variables can be used, but if a user appears on both lists, `server_audit_incl_users` has a higher priority, and their activities will be logged.

> **Note:**
>
> Note that `CONNECT` events are always logged for all users, regardless of these two settings. Logging is also based on username only, not the username and hostname combination that MariaDB uses to determine privileges.

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Location and Rotation of Logs

Logs can be written to a separate file or to the system logs. If you prefer to have the logging separated from other system information, the value of the variable server_audit_output_type should be set to `file`. Incidentally, `file` is the only option on Windows systems.

You can force a rotation by enabling the server_audit_file_rotate_now variable like so:

```
SET GLOBAL server_audit_file_rotate_now = ON;
```

Separate log files

In addition to setting `server_audit_output_type`, you will have to provide the file path and name of the audit file. This is set in the variable server_audit_file_path. You can set the file size limit of the log file with the variable server_audit_file_rotate_size.

So, if rotation is on and the log file has reached the size limit you set, a copy is created with a consecutive number as extension the original file will be truncated to be used for the auditing again. To limit the number of log files created, set the variable server_audit_file_rotations. You can force log file rotation by setting the variable server_audit_file_rotate_now to a value of `ON`. When the number of files permitted is reached, the oldest file will be overwritten. Here is how those variables might be set in a server configuration file:

```
[mysqld]
...
server_audit_file_rotate_now=ON
server_audit_file_rotate_size=1000000
server_audit_file_rotations=5
...
```

System logs

For security reasons, it's better sometimes to use the system logs instead of a local file owned by the `mysql` user. To do this, the value of server_audit_output_type needs to be set to `syslog`. Advanced configurations, such as using a remote `syslogd` service, are part of the `syslogd` configuration.

The variables,server_audit_syslog_ident and server_audit_syslog_info can be used to identify a system log entry made by the audit plugin. If a remote `syslogd` service is used for several MariaDB servers, these same variables are also used to identify the MariaDB server.

Here is a system log entry taken from a server which had server_audit_syslog_ident set to the default value of `mysql-server_auditing`, and server_audit_syslog_info set to `<prod1>`.

```
Aug 717:19:58localhostmysql-server_auditing:
<prod1> localhost.localdomain,root,localhost,1,7,
QUERY, mysql, 'SELECT * FROM user',0
```

Although the default values for server_audit_syslog_facility and server_audit_syslog_priority should be sufficient in most cases. They can be changed based on the definition in `syslog.h` for the functions `openlog()` and `syslog()`.

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Log Format

The audit plugin logs user access to MariaDB and its objects. The audit trail (that is, the audit log) is a set of records, written as a list of fields to a file in a plain-text format. The fields in the log are separated by commas. The format used for the plugin's own log file is slightly different from the format used if it logs to the system log because it has its own standard format.

Formal Specification

When the MariaDB Audit Plugin (v1) writes to a dedicated file, it uses a comma-separated (CSV) format. For tool developers, it is essential to map the `connectionid` to the server's global connection identifier to enable cross-log analysis.

Template: `<timestamp>,<serverhost>,<username>,<host>,<connectionid>,<queryid>,<operation>,<database>,<object>,<retcode>`

Field	Component	Data Type	Standardized Name / Description
1	`timestamp`	`DateTime`	Formatted as `%Y%m%d %H:%i:%s` (the default), or as specified in server_audit_timestamp_format.
2	`serverhost`	`String`	Hostname of the originating server.
5	`connectionid`	`Unsigned Int`	Standardized: Thread ID. Matches the `Thread ID` in Error, General, and Slow logs.
6	`queryid`	`Unsigned Int`	A unique identifier for the specific query, used to correlate `QUERY` and `TABLE` events.
7	`operation`	`String`	Action type (e.g., `CONNECT`, `QUERY`, `READ`, `WRITE`).

Field	Component	Data Type	Standardized Name / Description
10	`retcode`	`Integer`	Result code (`0` for success).

Audit Log Format with Syslog

If `server_audit_output_type` is set to `SYSLOG`, the standard CSV line is prefixed with syslog metadata: `<timestamp> <syslog_host> <syslog_ident>: <syslog_info> [Standard CSV Fields]`

Various events result in different audit records. Some events do not return a value for some fields (for instance, when the active database is not set when connecting to the server).

Below is a generic example of the output for connect events, with placeholders representing data. These are events in which a user connected, disconnected, or tried unsuccessfully to connect to the server.

```
[timestamp],[serverhost],[username],[host],[connectionid],0,CONNECT,[database],,0
[timestamp],[serverhost],[username],[host],[connectionid],0,DISCONNECT,,,0
[timestamp],[serverhost],[username],[host],[connectionid],0,FAILED_CONNECT,,,[retcode]
```

Here is the one audit record generated for each query event:

```
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],QUERY,[database],[object], [retcode]
```

Below are generic examples of records that are entered in the audit log for each type of table event:

```
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],CREATE,[database],[object],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],READ,[database],[object],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],WRITE,[database],[object],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],ALTER,[database],[object],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],RENAME,[database],
[object_old]|[database_new].[object_new],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],DROP,[database],[object],
```

Passwords are hidden in the log for certain types of queries. They are replaced with asterisks for `GRANT`, `CREATE USER`, `CREATE MASTER`, `CREATE SERVER`, and `ALTER SERVER` statements. Passwords, however, are not replaced for the `PASSWORD()` and `OLD_PASSWORD()` functions when they are used inside other SQL statements (i.e., `SET PASSWORD`).

> **CURRENT**
>
> For Galera Cluster replication applier operations, audit log plugin logs events with a generic name of `<wsrep_applier>`.
>
> This addresses an issue where the user was logged on the primary node, but stripped from other cluster nodes. See MDEV-35511 for details.
>
> **< 10.11.16**
>
> For Galera Cluster replication applier operations, audit log plugin logs events without indicating what user initiates them.

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Options and System Variables

Overview

There are a several options and system variables related to the MariaDB Audit Plugin, once it has been installed. System variables can be displayed using the SHOW VARIABLES statement like so:

```
SHOW GLOBAL VARIABLES LIKE 'server_audit%';
+-------------------------------+-----------------------+
| Variable_name                 | Value                 |
+-------------------------------+-----------------------+
| server_audit_events           |                       |
| server_audit_excl_users       |                       |
| server_audit_file_buffer_size | 0                     |
| ...                           | ...                   |
| server_audit_syslog_priority  | LOG_INFO              |
| server_audit_timestamp_format | %Y%m%d %H:%i:%s       |
+-------------------------------+-----------------------+
```

To change the value of one of these variables, you can use the `SET` statement, or set them at the command-line when starting MariaDB. It's recommended that you set them in the MariaDB configuration for the server like so:

```
[mariadb]
...
server_audit_excl_users='bob,ted'
...
```

System Variables

Below is a list of all system variables related to the Audit Plugin. See Server System Variables for a complete list of system variables and instructions on setting them. See also the full list of MariaDB options, system and status variables.

`server_audit_events`

- Description: If set, this restricts audit logging to certain event types. If not set, every event type is logged to the audit log.
- Command line: `--server-audit-events=value`
- Scope: Global
- Dynamic: Yes
- Data type: `string`
- Default value: Empty string
- Valid values:
  - `CONNECT`, `QUERY`, `TABLE` (MariaDB Audit Plugin < 1.2.0)
  - `CONNECT`, `QUERY`, `TABLE`, `QUERY_DDL`, `QUERY_DML` (MariaDB Audit Plugin >= 1.2.0)
  - `CONNECT`, `QUERY`, `TABLE`, `QUERY_DDL`, `QUERY_DML`, `QUERY_DCL` (MariaDB Audit Plugin >=1.3.0)
  - `CONNECT`, `QUERY`, `TABLE`, `QUERY_DDL`, `QUERY_DML`, `QUERY_DCL`, `QUERY_DML_NO_SELECT` (MariaDB Audit Plugin >= 1.4.4)
  - Consult MariaDB Audit Plugin - Versions for a list of MariaDB releases and their corresponding Audit Plugin versions.

`server_audit_excl_users`

- Description: If not empty, it contains the list of users whose activity will NOT be logged: `SET GLOBAL server_audit_excl_users='user_foo, user_bar'`. CONNECT records aren't affected by this variable - they are always logged. The user is still logged if it's specified in server_audit_incl_users.

- Command line: `--server-audit-excl-users=value`
- Scope: Global
- Dynamic: Yes
- Data type: `string`
- Default value: Empty string
- Size limit: 1024 characters

`server_audit_file_buffer_size`

- Description: Size (in bytes) of file buffer to make logging faster. Values > `0` are adjusted in increments of `8192`. (For instance, a value of `100` would be adjusted to `8192`.)
- Command line: `--server-audit-file-buffer-size= #`
- Scope: Global
- Dynamic: Yes
- Data type: `numeric`
- Value range: `0` to `65536`
- Default value: `0` (no buffering)
- Introduced: MariaDB 12.1
- Usage: See description

`server_audit_file_path`

- Description: When server_audit_output_type=file, sets the path and the filename to the log file. If the specified path exists as a directory, then the log will be created inside that directory with the name 'server_audit.log'. Otherwise the value is treated as a filename. The default value is 'server_audit.log', which means this file will be created in the database directory.
- Command line: `--server-audit-file-path=value`
- Scope: Global
- Dynamic: Yes
- Data type: `string`
- Default value: `server_audit.log`

`server_audit_file_rotate_now`

- Description: When server_audit_output_type=file, the user can force the log file rotation by setting this variable to ON or 1.
- Command line: `--server-audit-rotate-now[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data type: `boolean`
- Default value: `OFF`

`server_audit_file_rotate_size`

- Description: When server_audit_output_type=file, it limits the size of the log file to the given amount of bytes. Reaching that limit turns on the rotation - the current log file is renamed as 'file_path.1'. The empty log file is created as 'file_path' to log into it. The default value is 1000000.
- Command line: `--server-audit-rotate-size=#`
- Scope: Global
- Dynamic: Yes

- Data Type: `numeric`
- Default Value: `1000000`
- Range: `100` to `9223372036854775807`

`server_audit_file_rotations`

- Description: When server_audit_output_type=file', this specifies the number of rotations to save. If set to 0 then the log never rotates. The default value is 9.
- Command line: `--server-audit-rotations=#`
- Scope: Global
- Dynamic: Yes
- Data type: `numeric`
- Default value: `9`
- Range: `0` to `999`

`server_audit_incl_users`

- Description: If not empty, it contains a comma-delimited list of users whose activity will be logged: `SET GLOBAL server_audit_incl_users='user_foo, user_bar'`. CONNECT records aren't affected by this variable - they are always logged. This setting has higher priority than server_audit_excl_users. So if the same user is specified both in incl_ and excl_ lists, they will still be logged.
- Command line: `--server-audit-incl-users=value`
- Scope: Global
- Dynamic: Yes
- Data type: `string`
- Default value: Empty string
- Size limit: 1024 characters

`server_audit_loc_info`

- Description: Used by plugin internals. It has no useful meaning to users.
  - In earlier versions, users see it as a read-only variable.
  - In later versions, it is hidden from the user.
- Command line: N/A
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: Empty string

`server_audit_logging`

- Description: Enables/disables the logging. Expected values are ON/OFF: `SET GLOBAL server_audit_logging=on` If the server_audit_output_type is FILE, this will actually create/open the logfile so the server_audit_file_path should be properly specified beforehand. Same about the `SYSLOG`-related parameters. The logging is turned off by default.
- Command line: `--server-audit-logging[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data type: `boolean`

- Default value: `OFF`

`server_audit_mode`

- Description: This variable doesn't have any distinctive meaning for a user. Its value mostly reflects the server version with which the plugin was started and is intended to be used by developers for testing.
- Command line: `--server-audit-mode[=#]`
- Scope: Global
- Dynamic: Yes
- Data type: `numeric`
- Default value: `0`
- Range: `0` to `1`

`server_audit_output_type`

- Description: Specifies the desired output type. Can be SYSLOG, FILE or null as no output: `SET GLOBAL server_audit_output_type=file` file: log records will be saved into the rotating log file. The name of the file set by server_audit_file_path variable. syslog: log records will be sent to the local syslogd daemon with the standard <syslog.h> API. The default value is 'file'.
- Command line: `--server-audit-output-type=value`
- Scope: Global
- Dynamic: Yes
- Data type: `enum`
- Default value: `file`
- Valid values: `SYSLOG`, `FILE`

`server_audit_query_log_limit`

- Description: Limit on the length of the query string in a record.
- Command line: `--server-audit-query-log-limit=#`
- Scope: Global
- Dynamic: Yes
- Data type: `numeric`
- Default value: `1024`
- Range: `0` to `2147483647`

`server_audit_sync_log_file`

- Description: Flushes the buffer to the log file.
  While log records are in the buffer, they don't appear in the log file. To write them out from the buffer, issue this statement:
  `SET GLOBAL server_audit_sync_log_file=1`
- Command line: `--server-audit-sync-log-file`
- Scope: Global
- Dynamic: Yes
- Data type: N/A
- Default value: `OFF`
- Valid values: `ON` (or `1`), `OFF` (or `0`)
- Introduced: MariaDB 12.1
- Usage: See description

`server_audit_syslog_facility`

- Description: SYSLOG-mode variable. It defines the ‘facility’ of the records that will be sent to the syslog. Later the log can be filtered by this parameter.
- Command line: `--server-audit-syslog-facility=value`
- Scope: Global
- Dynamic: Yes
- Data type: `enum`
- Default value: `LOG_USER`
- Valid values: `LOG_USER`, `LOG_MAIL`, `LOG_DAEMON`, `LOG_AUTH`, `LOG_SYSLOG`, `LOG_LPR`, `LOG_NEWS`, `LOG_UUCP`, `LOG_CRON`, `LOG_AUTHPRIV`, `LOG_FTP`, and `LOG_LOCAL0` – `LOG_LOCAL7`.

`server_audit_syslog_ident`

- Description: SYSLOG-mode variable. String value for the ‘ident’ part of each syslog record. Default value is ‘mysql-server_auditing’. New value becomes effective only after restarting the logging.
- Command line: `--server-audit-syslog-ident=value`
- Scope: Global
- Dynamic: Yes
- Data type: `string`
- Default value: `mysql-server_auditing`

`server_audit_syslog_info`

- Description: SYSLOG-mode variable. The ‘info’ string to be added to the syslog records. Can be changed any time.
- Command line: `--server-audit-syslog-info=value`
- Scope: Global
- Dynamic: Yes
- Data type: `string`
- Default value: Empty string

`server_audit_syslog_priority`

- Description: SYSLOG-mode variable. Defines the priority of the log records for the syslogd.
- Command line: `--server-audit-syslog-priority=value`
- Scope: Global
- Dynamic: Yes
- Data type: `enum`
- Default value: `LOG_INFO`
- Valid values: `LOG_EMERG`, `LOG_ALERT`, `LOG_CRIT`, `LOG_ERR`, `LOG_WARNING`, `LOG_NOTICE`, `LOG_INFO`, `LOG_DEBUG`

`server_audit_timestamp_format`

- Description: A format string used to print the timestamp into the audit log messages. The format used is the same as DATE_FORMAT.
- Command line: `--server-audit-timestamp-format=value`
- Scope: Global
- Dynamic: Yes

- Data type: `string`
- Default value: `%Y%m%d %H:%i:%s`

Notes on System Variables

audit_file_buffer_size and server_audit_sync_log_file

The server audit plugin typically employs synchronous, per-event logging, causing performance bottlenecks. Individual file writes for each log entry can result in significant I/O overhead, especially in large database environments. As of MariaDB 12.1, two new variables were introduced to allow asynchronous logging, and more fine grained control over how audit log writes are handled. Using the `server_audit_file_buffer_size` setting (buffer size in bytes), you can configure an additional in-memory audit log buffer. When the size of the buffer exceeds the `server_audit_file_buffer_size` setting, the audit log is written to disk. Additionally, a manual on-demand audit log disk sync can be triggered by setting `server_audit_sync_log_file` to `ON` or `1`.

Options

`server_audit`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGINwhile the server is running.
  - See MariaDB Audit Plugin - Installation: Prohibiting Uninstallation for more information.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--server-audit=val`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Status Variables

There are a few status variables related to the MariaDB Audit Plugin, once it has been installed. These variables can be displayed using the SHOW STATUS statement like so:

```
SHOW STATUS LIKE 'server_audit%';

+----------------------------+------------------+
| Variable_name              | Value            |
+----------------------------+------------------+
| Server_audit_active        | ON               |
| Server_audit_current_log   | server_audit.log |
| Server_audit_last_error    |                  |
| Server_audit_writes_failed | 0                |
+----------------------------+------------------+
```

Status Variables

Below is a list of all status variables related to the Audit Plugin. These cannot be set: These are not to be confused with system variables, which can be set. See Server Status Variables for a complete list of status variables that can be viewed with the SHOW STATUS statement. See also the Full list of MariaDB options, system and status variables.

`Server_audit_active`

- Description: If the auditing is actually working. It gets the ON value when the logging is successfully started. Then it can get the OFF value if the logging was stopped or log records can't be properly stored due to file or syslog errors.
- Data Type: `boolean`

`Server_audit_current_log`

- Description: The name of the logfile or the SYSLOG parameters that are in current use.
- Data Type: `string`

`Server_audit_last_error`

- Description: If something went wrong with the logging here you can see the message.
- Data Type: `string`

`Server_audit_writes_failed`

- Description: The number of log records since last logging-start that weren't properly stored because of errors of any kind. The global value can be flushed by FLUSH STATUS.
- Data Type: `numeric`
- Default Value: `0`

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Versions

Below is a list of the releases of the MariaDB Audit Plugin, the most recent version first, and in which versions of MariaDB each plugin version was included.

MariaDB Community Audit Plugin Version	Introduced in MariaDB Community Server
1.5.0	MariaDB 12.0.1
1.4.14	MariaDB 10.3.35, MariaDB 10.4.25, MariaDB 10.5.16, MariaDB 10.6.8, MariaDB 10.7.4
1.4.13	MariaDB 10.2.38, MariaDB 10.3.29, MariaDB 10.4.19, MariaDB 10.5.10
1.4.10	MariaDB 10.2.35, MariaDB 10.3.26, MariaDB 10.5.7
1.4.7	MariaDB 10.1.41, MariaDB 10.2.26, MariaDB 10.3.17, MariaDB 10.4.7
1.4.5	MariaDB 10.2.24, MariaDB 10.3.15, MariaDB 10.4.5
1.4.4	MariaDB 5.5.61, MariaDB 10.0.36, MariaDB 10.1.34, MariaDB 10.2.15, MariaDB 10.3.7, MariaDB 10.4.0
1.4.0	MariaDB 5.5.48, MariaDB 10.0.24, MariaDB 10.1.11
1.3.0	MariaDB 5.5.43, MariaDB 10.0.18, MariaDB 10.1.5
1.2.0	MariaDB 5.5.42, MariaDB 10.0.17, MariaDB 10.1.4
1.1.7	MariaDB 5.5.38, MariaDB 10.0.11, MariaDB 10.1.0

MariaDB Community Audit Plugin Version	Introduced in MariaDB Community Server
1.1.6	MariaDB 5.5.37, MariaDB 10.0.10
1.1.5	MariaDB 10.0.09
1.1.4	MariaDB 5.5.36
1.1.3	MariaDB 5.5.34, MariaDB 10.0.7

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugins

See Pluggable Authentication Overview.

Pluggable authentication allows MariaDB to use various authentication methods, enabling external validation, different hashing algorithms, and role-based access control.

See Authentication Plugin - caching_sha2_password.

MySQL-compatible caching_sha2_password authentication plugin, for migrating MySQL users to MariaDB without changing their passwords (migration use; PARSEC is recommended otherwise).

See Authentication Plugin - ed25519.

The ed25519 authentication plugin provides high-security password authentication using the Elliptic Curve Digital Signature Algorithm, a modern alternative to SHA-1.

See Authentication Plugin - GSSAPI.

Complete GSSAPI authentication setup: Kerberos/SSPI single sign-on, INSTALL SONAME ‘auth_gssapi’, gssapi_keytab_path/principal_name, CREATE USER syntax.

See Authentication Plugin - mysql_native_password.

Complete Authentication Plugin - mysql_native_password guide for MariaDB. Complete reference documentation for implementation, configuration, and usage.

See Authentication Plugin - mysql_old_password.

This plugin provides backward compatibility for pre-4.1 clients using an older, insecure password hashing algorithm and should not be used for new installations.

See Authentication Plugin - Named Pipe.

The named_pipe authentication plugin allows Windows users connecting via named pipes to authenticate using their operating system credentials without a password.

See Authentication Plugin - PARSEC.

PARSEC is a modern, secure authentication plugin that uses salted passwords and elliptic curve cryptography to prevent replay attacks and secure user credentials.

See Authentication Plugin - SHA-256.

The SHA-256 authentication plugin uses the SHA-256 hashing algorithm for password storage, offering stronger security than the default SHA-1 method.

See Authentication Plugin - Unix Socket.

Official Unix socket authentication: OS user login via SO_PEERCRED/uid matching, CREATE USER IDENTIFIED VIA unix_socket, and unix_socket force modes.

See Authentication with Pluggable Authentication Modules (PAM).

Learn about authentication with Pluggable Authentication Modules (PAM) in MariaDB Server. This section details how to integrate MariaDB with PAM for centralized and flexible user authentication.

This page is licensed: CC BY-SA / Gnu FDL

Pluggable Authentication Overview

When a user attempts to log in, the authentication plugin controls how MariaDB Server determines whether the connection is from a legitimate user.

When creating or altering a user account with the GRANT, CREATE USER or ALTER USER statements, you can specify the authentication plugin you want the user account to use, by providing the `IDENTIFIED VIA` clause. By default, when you create a user account without specifying an authentication plugin, MariaDB uses the mysql_native_password plugin.

> **Tip:**
>
> You can specify multiple authentication plugins for each user account.

The `root@localhost` user created by mariadb-install-db has the ability to use two authentication plugins:

1. It is configured to try to use the unix_socket authentication plugin. This allows the `root@localhost` user to log in without a password via the local Unix socket file defined by the socket system variable, as long as the login is attempted from a process owned by the operating system `root` user account.
2. If authentication fails with the unix_socket authentication plugin, it is configured to try to use the mysql_native_password authentication plugin. However, an invalid password is initially set, so in order to authenticate this way, a password must be set with SET PASSWORD.

Supported Authentication Plugins

The authentication process is a conversation between the server and a client. MariaDB implements both server-side and client-side authentication plugins.

Supported Server Authentication Plugins

MariaDB provides nine server-side authentication plugins:

- mysql_native_password
- mysql_old_password
- ed25519
- gssapi
- pam (Unix only)
- unix_socket (Unix only)
- named_pipe (Windows only)
- PARSEC (from MariaDB Community Server 11.6 and MariaDB Enterprise Server 11.8)
- caching_sha2_password (from MariaDB Community Server 12.1 and MariaDB Enterprise Server 11.8)

Supported Client Authentication Plugins

MariaDB provides eight client-side authentication plugins:

- mysql_native_password
- mysql_old_password
- client_ed25519

- auth_gssapi_client
- dialog
- mysql_clear_password
- sha256_password
- caching_sha2_password

Options Related to Authentication Plugins

Server Options Related to Authentication Plugins

MariaDB supports the following server options related to authentication plugins:

Server Option	Description
old_passwords={1 | 0}	If set to `1` (`0` is default), MariaDB reverts to using the mysql_old_password authentication plugin by default for newly created users and passwords, instead of the mysql_native_password authentication plugin.
plugin_dir=path	Path to the plugin directory. For security reasons, either make sure this directory can only be read by the server, or set secure_file_priv.
plugin_maturity=level	The lowest acceptable plugin maturity. MariaDB will not load plugins less mature than the specified level.
secure_auth	Connections will be blocked if they use the mysql_old_password authentication plugin.

Client Options Related to Authentication Plugins

Most clients and utilities support command-line arguments related to client authentication plugins:

Client Option	Description
–connect-expired-password	Notify the server that this client is prepared to handle expired password sandbox mode even if –batch was specified.
–default-auth=name	Default authentication client-side plugin to use.
–plugin-dir=path	Directory for client-side plugins.
–secure-auth	Refuse to connect to the server if the server uses the mysql_old_password authentication plugin. This mode is off by default, which is different from MySQL.

Developers who are using MariaDB Connector/C can implement similar functionality in their application by setting the following options with the mysql_optionsv function:

- `MYSQL_OPT_CAN_HANDLE_EXPIRED_PASSWORDS`
- `MYSQL_PLUGIN_DIR`
- `MYSQL_DEFAULT_AUTH`
- `MYSQL_SECURE_AUTH`

For example:

```
mysql_optionsv(mysql, MYSQL_OPT_CAN_HANDLE_EXPIRED_PASSWORDS, 1);
mysql_optionsv(mysql, MYSQL_DEFAULT_AUTH, "name");
mysql_optionsv(mysql, MYSQL_PLUGIN_DIR, "path");
mysql_optionsv(mysql, MYSQL_SECURE_AUTH, 1);
```

Installation Options Related to Authentication Plugins

mariadb-install-db supports the following installation options related to authentication plugins:

Installation Option	Description
–auth-root-authentication-method={normal | socket}	If set to `normal` (the default), it creates a `root@localhost` account that authenticates with the mysql_native_password authentication plugin and that has no initial password set, which can be insecure. If set to `socket`, it creates a `root@localhost` account that authenticates with the unix_socket authentication plugin. Set to `normal` by default.
–auth-root-socket-user=USER	Used with `--auth-root-authentication-method=socket`. It specifies the name of the second account to create with SUPER privileges in addition to root, as well as of the system account allowed to access it. Defaults to the value of `--user`.

Extended SQL Syntax

MariaDB has extended the SQL standard GRANT, CREATE USER, and ALTER USER statements, so that they support specifying different authentication plugins for specific users. An authentication plugin can be specified with these statements by providing the `IDENTIFIED VIA` clause. Examples:

```
GRANT <privileges> ON <level> TO <user>
   IDENTIFIED VIA <plugin> [ USING <string> ]
```

```
CREATE USER <user>
   IDENTIFIED VIA <plugin> [ USING <string> ]
```

```
ALTER USER <user>
   IDENTIFIED VIA <plugin> [ USING <string> ]
```

The optional `USING` clause allows users to provide an authentication string to a plugin. The authentication string's format and meaning is completely defined by the plugin.

For example, for the mysql_native_password authentication plugin, the authentication string should be a password hash:

```
CREATE USER mysqltest_up1
   IDENTIFIED VIA mysql_native_password USING '*E8D46CE25265E545D225A8A6F1BAF642FEBEE5CB';
```

Since mysql_native_password is the default authentication plugin, the above is just another way of saying the following:

```
CREATE USER mysqltest_up1
   IDENTIFIED BY PASSWORD '*E8D46CE25265E545D225A8A6F1BAF642FEBEE5CB';
```

In contrast, for the pam authentication plugin, the authentication string should refer to a PAM service name:

```
CREATE USER mysqltest_up1
   IDENTIFIED VIA pam USING 'mariadb';
```

A user account can be associated with multiple authentication plugins.

To configure the `root@localhost` user account to try the unix_socket authentication plugin, followed by the mysql_native_password authentication plugin as a backup, execute the following query:

```
CREATE USER root@localhost
   IDENTIFIED VIA unix_socket
   OR mysql_native_password USING PASSWORD("verysecret");
```

See Authentication for more information.

Authentication Plugins Installed by Default

Server Authentication Plugins Installed by Default

Not all server-side authentication plugins are installed by default. If a specific server-side authentication plugin is not installed by default, then you can find the installation procedure on the documentation page for the specific authentication plugin.

The following server-side authentication plugins are installed by default:

- The mysql_native_password and mysql_old_password authentication plugins authentication plugins are installed by default in all builds.
- The unix_socket authentication plugin is installed by default in all builds on Unix and Linux.
- The named_pipe authentication plugin is installed by default in all builds on Windows.

The following server-side authentication plugins are installed by default:

- The mysql_native_password and mysql_old_password authentication plugins are installed by default in all builds.
- The unix_socket authentication plugin is installed by default in new installations that use the .deb packages provided by Debian's default repositories and Ubuntu's default repositories in Ubuntu. See Differences in MariaDB in Debian (and Ubuntu) for more information.
- The named_pipe authentication plugin is installed by default in all builds on Windows.

Client Authentication Plugins Installed by Default

Client-side authentication plugins do not need to be *installed* in the same way that server-side authentication plugins do. If the client uses either the `libmysqlclient` or MariaDB Connector/C library, then the library automatically loads client-side authentication plugins from the library's plugin directory whenever they are needed.

Most clients and utilities support the `--plugin-dir` command line argument that can be used to set the path to the library's plugin directory:

Client Option	Description
–plugin-dir=path	Directory for client-side plugins.

Developers who are using MariaDB Connector/C can implement similar functionality in their application by setting the `MYSQL_PLUGIN_DIR` option with the mysql_optionsv function. Example:

```
mysql_optionsv(mysql, MYSQL_PLUGIN_DIR, "path");
```

If your client encounters errors similar to the following error, you may need to set the path to the library's plugin directory:

```
ERROR 2059 (HY000): Authentication plugin 'dialog' cannot be loaded: /usr/lib/mysql/plugin/dialog.so:
cannot open shared object file: No such file or directory
```

If the client does **not** use either the `libmysqlclient` or MariaDB Connector/C library, then you will have to determine which authentication plugins are supported by the specific client library used by the client.

If the client uses either the `libmysqlclient` or MariaDB Connector/C library, but the client is not bundled with either library's *optional* client authentication plugins, then you can only use the conventional authentication plugins (like mysql_native_password and mysql_old_password) and the non-conventional authentication plugins that don't require special client-side authentication plugins (like unix_socket and named_pipe).

Default Authentication Plugin

Default Server Authentication Plugin

The mysql_native_password authentication plugin is currently the default authentication plugin in all versions of MariaDB if the old_passwords system variable is set to `0`, which is the default.

On a system with the old_passwords system variable set to `0`, this means that if you create a user account with either the GRANT or CREATE USER `statements, and if you do not specify an authentication plugin with the` IDENTIFIED VIA `clause, then MariaDB will use the [mysql_native_password](#pg-tication-plugins-authentication-plugin-mysql-native-password-0a775c07) authentication plugin for the user account.`

Creating a user account like this, it uses the mysql_native_password authentication plugin:

```
CREATE USER username@hostname;
```

The same is true for this user account:

```
CREATE USER username@hostname IDENTIFIED BY 'notagoodpassword';
```

The mysql_old_password authentication plugin becomes the default authentication plugin in all versions of MariaDB, if the old_passwords system variable is explicitly set to `1`.

Important:

The mysql_old_password authentication plugin is not considered secure. It is recommended to avoid using this authentication plugin. To help prevent undesired use of the mysql_old_password authentication plugin, the server supports the secure_auth system variable that configures the server to refuse connections trying to use the mysql_old_password authentication plugin.

Most clients and utilities support `secure_auth`.

Server Option	Description
old_passwords={1 | 0}	If set to `1` (`0` is default), MariaDB reverts to using the mysql_old_password authentication plugin by default for newly created users and passwords, instead of the mysql_native_password authentication plugin.
secure_auth	Connections are blocked if they use the mysql_old_password authentication plugin.

Developers using MariaDB Connector/C can implement similar functionality in their application by setting the `MYSQL_SECURE_AUTH` option with the mysql_optionsv function:

```
mysql_optionsv(mysql, MYSQL_SECURE_AUTH, 1);
```

Default Client Authentication Plugin

The default client-side authentication plugin depends on these factors:

- If a client doesn't explicitly set the default client-side authentication plugin, the client determines which authentication plugin to use, by checking the length of the scramble in the server's handshake packet.
- If the server's handshake packet contains a 9-byte scramble, the client defaults to the mysql_old_password authentication plugin.
- If the server's handshake packet contains a 20-byte scramble, the client defaults to the mysql_native_password authentication plugin.

Setting the Default Client Authentication Plugin

Most clients and utilities support the `--default-auth` command-line argument that sets the default client-side authentication plugin:

Client Option	Description
`--default-auth=name`	Default authentication client-side plugin to use.

Developers using MariaDB Connector/C can implement similar functionality in applications, by setting the `MYSQL_DEFAULT_AUTH` option with the mysql_optionsv function:

```
mysql_optionsv(mysql, MYSQL_DEFAULT_AUTH, "name");
```

If you know that your user account is configured to require a client-side authentication plugin other than mysql_old_password or mysql_native_password, this helps speed up your connection process when you explicitly set the default client-side authentication plugin.

According to the client-server protocol, the server first sends the handshake packet to the client, then the client replies with a packet containing the user name of the user account that is requesting access. The server handshake packet initially tells the client to use the default server authentication plugin, and the client reply initially tells the server that it will use the default client authentication plugin.

However, the server-side and client-side authentication plugins mentioned in these initial packets may not be the correct ones for this specific user account. The server only knows what authentication plugin to use for this specific user account after reading the user name from the client reply packet and finding the appropriate row for the user account in either the mysql.user table or the mysql.global_priv table, depending on the MariaDB version.

If the server finds that either the server-side or client-side default authentication plugin does not match the actual authentication plugin that should be used for the given user account, then the server restarts the authentication on either the server side or the client side.

This means that, if you know what client authentication plugin your user account requires, then you can avoid an unnecessary authentication restart and you can save two packets and two round-trips.between the client and server by configuring your client to use the correct authentication plugin by default.

Available Authentication Plugins

Server Authentication Plugins

`mysql_native_password`

The mysql_native_password authentication plugin uses the same password hashing algorithm used by the PASSWORD() function when old_passwords is set. This hashing algorithm is based on SHA-1.

`mysql_old_password`

The mysql_old_password authentication plugin uses the same hashing algorithm used by the OLD_PASSWORD() function and by the PASSWORD() function when old_passwords=1 is set.

`ed25519`

The ed25519 authentication plugin uses the Elliptic Curve Digital Signature Algorithm to securely store users' passwords and to authenticate users. The ed25519 algorithm is the same one that is used by OpenSSH. It is based on the elliptic curve and code created by Daniel J. Bernstein.

`gssapi`

The gssapi authentication plugin allows authenticating with services that use the Generic Security Services Application Program Interface (GSSAPI). Windows has a slightly different but very similar API called Security Support Provider Interface (SSPI).

On Windows, this authentication plugin supports Kerberos and NTLM authentication. Windows authentication is supported regardless of whether a domain is used in the environment.

On Unix systems, the most dominant GSSAPI service is Kerberos. However, it is less commonly used on Unix systems than it is on Windows. Regardless, this authentication plugin also supports Kerberos authentication on Unix.

The gssapi authentication plugin is most often used for authenticating with Microsoft Active Directory.

`pam`

The pam authentication plugin allows MariaDB to offload user authentication to the system's Pluggable Authentication Module (PAM) framework. PAM is an authentication framework used by Linux, FreeBSD, Solaris, and other Unix-like operating systems.

`unix_socket`

The unix_socket authentication plugin allows the user to use operating system credentials when connecting to MariaDB via the local Unix socket file. This Unix socket file is defined by the socket system variable.

The unix_socket authentication plugin works by calling the getsockopt system call with the `SO_PEERCRED` socket option, which allows it to retrieve the `uid` of the process that is connected to the socket. It is then able to get the user name associated with that `uid`. Once it has the user name, it will authenticate the connecting user as the MariaDB account that has the same user name.

For example:

```
$ mysql -uroot
MariaDB []> CREATE USER serg IDENTIFIED VIA unix_socket;
MariaDB []> CREATE USER monty IDENTIFIED VIA unix_socket;
MariaDB []> quit
Bye
$ whoami
serg
$ mysql --user=serg
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.2.0-MariaDB-alpha-debug Source distribution
MariaDB []> quit
Bye
$ mysql --user=monty
ERROR 1045 (28000): Access denied for user 'monty'@'localhost' (using password: NO)
```

In this example, a user `serg` is already logged into the operating system and has full shell access. The user has already authenticated with the operating system and the MariaDB account is configured to use the unix_socket authentication plugin, so the user does not need to authenticate again for the database. MariaDB accepts the user's operating system credentials and allows connecting. However, any attempt to connect to the database as another operating system user will be denied.

`named_pipe`

The named_pipe authentication plugin allows the user to use operating system credentials when connecting to MariaDB via named pipe on Windows. Named pipe connections are enabled by the named_pipe system variable.

The named_pipe authentication plugin works by using named pipe impersonation and calling `GetUserName()` to retrieve the user name of the process that is connected to the named pipe. Once it has the user name, it authenticates the connecting user as the MariaDB account that has the same user name:

```
CREATE USER wlad IDENTIFIED VIA named_pipe;
CREATE USER monty IDENTIFIED VIA named_pipe;
quit

C:\>echo %USERNAME%
wlad

C:\> mysql --user=wlad --protocol=PIPE
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.1.12-MariaDB-debug Source distribution

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> quit
Bye

C:\> mysql --user=monty  --protocol=PIPE
ERROR 1698 (28000): Access denied for user 'monty'@'localhost'
```

`PARSEC`

The PARSEC (Password Authentication using Response Signed with Elliptic Curve) authentication plugin uses salted passwords, key derivation, an extensible password storage format, and both server- and client-side scrambles. It signs the authentication response with stock unmodified `ed25519` (as provided by OpenSSL, WolfSSL, or GnuTLS). PARSEC is intended to become the default authentication plugin in a future release.

This plugin is available from MariaDB Community Server 11.6 and MariaDB Enterprise Server 11.8.

`caching_sha2_password`

The caching_sha2_password authentication plugin provides MySQL-compatible authentication and allows users to be moved from MySQL to MariaDB without changing their passwords. It is intended primarily as a migration aid; for new accounts the more secure PARSEC authentication plugin is recommended.

This plugin is available from MariaDB Community Server 12.1 and MariaDB Enterprise Server 11.8, and is not installed by default — see Authentication Plugin - caching_sha2_password for installation instructions.

See Also

- GRANT
- CREATE USER
- ALTER USER
- Authentication from MariaDB 10.4
- Who are you? The history of MySQL and MariaDB authentication protocols from 1997 to 2017
- MySQL 5.6 Reference Manual: Pluggable Authentication
- MySQL 5.6 Reference Manual: Writing Authentication Plugins

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - caching_sha2_password

> **Note:**
>
> This plugin is available from these MariaDB Server 11.4 versions: MariaDB Community Server 11.4.9 / 11.8.4, MariaDB Enterprise Server 11.4.10-7 / 11.8.6-3.

Overview

To aid migrations MariaDB provides MySQL compatible `caching_sha2_password` authentication plugin. It allows to move users from MySQL to MariaDB without requiring them to change their passwords. It should be only used for migration, otherwise a more secure and convenient PARSEC plugin is recommended.

Installing the Plugin

The `caching_sha2_password` authentication plugin's shared library is included in MariaDB packages as the `auth_mysql_sha2.so` or `auth_mysql_sha2.dll` shared library on systems where it can be built.

> **Warning:**
>
> Some builds also contain a file named `caching_sha2_password.so` or `caching_sha2_password.dll`. That one *cannot* be loaded by the server, though – it's the client plugin implementing the client side of this authentication protocol.

Although the plugin's shared library is distributed with MariaDB, the plugin is not actually installed into MariaDB by default. There are two methods that can be used to install the plugin into MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'auth_mysql_sha2';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = auth_mysql_sha2
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'auth_mysql_sha2';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options must be removed to prevent the plugin from being loaded the next time the server is restarted.

Using the Plugin

To create a user in MariaDB that was using `caching_sha2_password` plugin in MySQL, issue this statement:

```
CREATE USER user@host IDENTIFIED WITH caching_sha2_password USING 'authentication_string';
```

Here, `authentication_string` is taken from the `mysql.user` table for the corresponding user in MySQL installation. Beware that the authentication string for `caching_sha2_password` in MySQL can contain non-printable

characters and copying it from the terminal window will likely not work.

System Variables

`caching_sha2_password_private_key_path`

- Description: A path to the private RSA key used for authentication.
- Command line: `--caching-sha2-password-private-key-path`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `private_key.pem`
- Introduced: MariaDB 11.4.9, MariaDB 11.8.4

`caching_sha2_password_public_key_path`

- Description: A path to the public RSA key used for authentication.
- Command line: `--caching-sha2-password-public-key-path`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: `public_key.pem`
- Introduced: MariaDB 11.4.9, MariaDB 11.8.4

`caching_sha2_password_auto_generate_rsa_keys`

- Description: Auto generate RSA keys at server startup if key paths are not explicitly set and key files are not present at their default locations.
- Command line: `--caching-sha2-password-auto-generate-rsa-keys`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `ON`
- Introduced: MariaDB 11.4.9, MariaDB 11.8.4

`caching_sha2_password_digest_rounds`

- Description: Number of SHA2 rounds to be performed when computing a password hash.
- Command line: `--caching-sha2-password-digest-rounds`
- Scope: Global
- Dynamic: No
- Data Type: `integer`
- Default Value: `5000`
- Introduced: MariaDB 11.4.9, MariaDB 11.8.4

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - ed25519

MySQL has used SHA-1 based authentication since version 4.1. The authentication plugin is called mysql_native_password. Over the years as computers became faster, new attacks on SHA-1 were being developed.

Nowadays SHA-1 is no longer considered as secure as it was in 2001. That's why the `ed25519` authentication plugin was created.

The `ed25519` authentication plugin uses Elliptic Curve Digital Signature Algorithm (ECDSA) to securely store users' passwords and to authenticate users. The ed25519 algorithm is the same one that is used by OpenSSH. It is based on the elliptic curve and code created by Daniel J. Bernstein.

Installing the Plugin

Although the plugin's shared library is distributed by default with MariaDB, with a file name of `auth_ed25519.so` (Unix) or `auth_ed25519.dll` (Windows), the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'auth_ed25519';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = auth_ed25519
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'auth_ed25519';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, those options must be removed to prevent the plugin from being loaded the next time the server is restarted.

Creating Users

You can create a user account by executing the CREATE USER statement and providing the IDENTIFIED VIA clause followied by the name of the plugin, `ed25519`, and providing the `USING` clause followed by the PASSWORD() function, with the plain-text password as an argument:

```
CREATE USER username@hostname IDENTIFIED VIA ed25519 USING PASSWORD('secret');
```

If SQL_MODE does not have `NO_AUTO_CREATE_USER` set, then you can also create the user account via GRANT:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA ed25519 USING PASSWORD('secret');
```

The PASSWORD() function and SET PASSWORD statements don't work with the `ed25519` authentication plugin. Instead, you have to use the UDF that comes with the authentication plugin to calculate the password hash:

```
CREATE FUNCTION ed25519_password RETURNS STRING SONAME "auth_ed25519.so";
```

Now you can calculate a password hash by executing this query:

```
SELECT ed25519_password("secret");
+--------------------------------------------+
| SELECT ed25519_password("secret");         |
+--------------------------------------------+
| ZIgUREUg5PVgQ6LskhXmO+eZLS0nC8be6HPjYWR4YJY |
+--------------------------------------------+
```

Now you can use it to create the user account using the new password hash. As with any password, you should always use a complex password that isn't easy to guess. If you don't, if anyone gets access to the stored passwords in the `mysql.user` table, they could use rainbow tables to figure out the original password.

To create a user account via CREATE USER, specify the name of the plugin in the IDENTIFIED VIA clause while providing the password hash as the `USING` clause:

```
CREATE USER username@hostname IDENTIFIED VIA ed25519
  USING 'ZIgUREUg5PVgQ6LskhXmO+eZLS0nC8be6HPjYWR4YJY';
```

If SQL_MODE does not have `NO_AUTO_CREATE_USER` set, you can also create the user account via GRANT:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA ed25519
  USING 'ZIgUREUg5PVgQ6LskhXmO+eZLS0nC8be6HPjYWR4YJY';
```

Warning:

Note that users require a password in order to be able to connect. It is possible to create users without specifying a password, but they will be unable to connect.

Changing User Passwords

You can change a user account's password by executing the SET PASSWORD statement followed by the PASSWORD() function and providing the plain-text password as an argument:

```
SET PASSWORD =  PASSWORD('new_secret')
```

You can also change the user account's password with the ALTER USER statement. You would have to specify the name of the plugin in the IDENTIFIED VIA clause while providing the plain-text password as an argument to the `PASSWORD()` function in the `USING` clause:

```
ALTER USER username@hostname IDENTIFIED VIA ed25519 USING PASSWORD('new_secret');
```

The `PASSWORD()` function and SET PASSWORD statement did not work with the `ed25519` authentication plugin. Instead, you would have to use the UDF that comes with the authentication plugin to calculate the password hash:

```
CREATE FUNCTION ed25519_password RETURNS STRING SONAME "auth_ed25519.so";
```

Now you can calculate a password hash by executing this query:

```
SELECT ed25519_password("secret");
+--------------------------------------------+
| SELECT ed25519_password("secret");         |
+--------------------------------------------+
| ZIgUREUg5PVgQ6LskhXmO+eZLS0nC8be6HPjYWR4YJY |
+--------------------------------------------+
```

Now you can change the user account's password using the new password hash.

You can change the user account's password with the ALTER USER statement. You have to specify the name of the plugin in the IDENTIFIED VIA clause, while providing the password hash as the `USING` clause:

```
ALTER USER username@hostname IDENTIFIED VIA ed25519
  USING 'ZIgUREUg5PVgQ6LskhXmO+eZLS0nC8be6HPjYWR4YJY';
```

Client Authentication Plugins

For clients that use the `libmysqlclient` or MariaDB Connector/C libraries, MariaDB provides one client authentication plugin that is compatible with the `ed25519` authentication plugin:

- `client_ed25519`

When connecting with a client or utility to a server as a user account that authenticates with the `ed25519` authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the `--plugin-dir` option:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

`client_ed25519`

The `client_ed25519` client authentication plugin hashes and signs the password using the Elliptic Curve Digital Signature Algorithm (ECDSA) before sending it to the server.

Support in Client Libraries

Using the Plugin with MariaDB Connector/C

MariaDB Connector/C supports `ed25519` authentication using the client authentication plugins mentioned in the previous section.

Using the Plugin with MariaDB Connector/ODBC

MariaDB Connector/ODBC supports `ed25519` authentication using the client authentication plugins mentioned in the previous section.

Using the Plugin with MariaDB Connector/J

MariaDB Connector/J supports `ed25519` authentication.

Using the Plugin with MariaDB Connector/Node.js

MariaDB Connector/Node.js supports `ed25519` authentication.

Using the Plugin with MySqlConnector for .NET

MySqlConnector for ADO.NET supports `ed25519` authentication.

The connector implemented support for this authentication plugin in a separate NuGet package called MySqlConnector.Authentication.Ed25519. After the package is installed, your application must call `Ed25519AuthenticationPlugin.Install` to enable it.

Options

`ed25519`

- Description: Controls how the server should treat the plugin when it starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.

        - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
        - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
    - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--ed25519=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - GSSAPI

The `gssapi` authentication plugin allows the user to authenticate with services that use the Generic Security Services Application Program Interface (GSSAPI). Windows has a slightly different but very similar API called Security Support Provider Interface (SSPI). The GSSAPI is a standardized API described in RFC2743 and RFC2744. The client and server negotiate using a standardized protocol described in RFC7546.

On Windows, this authentication plugin supports Kerberos and NTLM authentication. Windows authentication is supported regardless of whether a domain is used in the environment.

On Unix systems, the most dominant GSSAPI service is Kerberos. However, it is less commonly used on Unix systems than it is on Windows. Regardless, this authentication plugin also supports Kerberos authentication on Unix.

The `gssapi` authentication plugin is most often used for authenticating with Microsoft Active Directory.

This article gives instructions on configuring the `gssapi` authentication plugin for MariaDB for passwordless login.

Installing the Plugin's Package

> **Note:**
>
> Before MariaDB 10.11, the GSSAPI server plugin was distributed on Windows as a dynamic plugin (`auth_gssapi.dll`). Since MariaDB 10.11, the plugin is statically compiled into the server and enabled by default. As a result, `auth_gssapi.dll` is no longer produced or required.

The `gssapi` authentication plugin's shared library is included in MariaDB packages as the `auth_gssapi.so` or `auth_gssapi.dll` shared library on systems where it can be built.

Installing on Linux

The `gssapi` authentication plugin is included in binary tarballs on Linux.

Installing with a Package Manager

The `gssapi` authentication plugin can also be installed via a package manager on Linux. In order to do so, your system needs to be configured to install from one of the MariaDB repositories.

You can configure your package manager to install it from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script.

You can also configure your package manager to install it from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool.

Installing with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using yum or dnf. Starting with RHEL 8 and Fedora 22, `yum` has been replaced by `dnf`, which is the next major version of `yum`. However, `yum` commands still work on many systems that use `dnf`:

```
sudo yum install MariaDB-gssapi-server
```

Installing with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant DEB package from MariaDB's repository using apt-get:

```
sudo apt-get install mariadb-plugin-gssapi-server
```

Installing with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using zypper:

```
sudo zypper install MariaDB-gssapi-server
```

Installing on Windows

> **Note:**
>
> Before MariaDB 10.11, the `gssapi` authentication plugin is included in MSI and ZIP packages on Windows.

Installing the Plugin

> **Note:**
>
> Since MariaDB 10.11, on Windows, the GSSAPI server plugin is statically compiled into the server and enabled by default. No separate installation is required.

On Windows, and on other operating systems, although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'auth_gssapi';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = auth_gssapi
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'auth_gssapi';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options must be removed to prevent the plugin from being loaded the next time the server is restarted.

Configuring the Plugin

If the MariaDB server is running on Unix, then some additional configuration steps will need to be implemented in order to use the plugin.

If the MariaDB server is running on Windows, then no special configuration steps will need to be implemented in order to use the plugin, as long as the following is true:

- The Windows server is joined to a domain.
- The MariaDB server process is running as either a NetworkService Account or a Domain User Account.

Creating a Keytab File on Unix

If the MariaDB server is running on Unix, then the KDC server will need to create a keytab file for the MariaDB server. The keytab file contains the service principal name, which is the identity that the MariaDB server will use to communicate with the KDC server. The keytab will need to be transferred to the MariaDB server, and the `mysqld` server process will need read access to this keytab file.

How this keytab file is generated depends on whether the KDC server is Microsoft Active Directory KDC or MIT Kerberos KDC.

Creating a Keytab File with Microsoft Active Directory

If you are using Microsoft Active Directory KDC, you may need to create a keytab using the ktpass.exe utility on a Windows host. The service principal will need to be mapped to an existing domain user. To do so, follow the steps listed below.

Be sure to replace the following items in the step below:

- Replace `${HOST}` with the fully qualified DNS name for the MariaDB server host.
- Replace `${DOMAIN}` with the Active Directory domain.
- Replace `${AD_USER}` with the existing domain user.
- Replace `${PASSWORD}` with the password for the service principal.

To create the service principal, execute the following command:

```
ktpass.exe /princ mariadb/${HOST}@${DOMAIN} /mapuser ${AD_USER} /pass ${PASSWORD} /out mariadb.keytab
/crypto all /ptype KRB5_NT_PRINCIPAL /mapop set
```

Creating a Keytab File with MIT Kerberos

If you are using MIT Kerberos KDC, then you can create a keytab file using the kadmin utility. To do so, follow the steps listed below.

In the following steps, be sure to replace `${HOST}` with the fully qualified DNS name for the MariaDB server host.

First, create the service principal using the kadmin utility:

```
kadmin -q "addprinc -randkey mariadb/${HOST}"
```

Then, export the newly created user to the keytab file using the kadmin utility:

```
kadmin -q "ktadd -k /path/to/mariadb.keytab mariadb/${HOST}"
```

More details can be found at the following links:

- MIT Kerberos Documentation: Database administration
- MIT Kerberos Documentation: Application servers

Configuring the Path to the Keytab File on Unix

If the MariaDB server is running on Unix, then the path to the keytab file that was previously created can be set by configuring the gssapi_keytab_path system variable. This can be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
gssapi_keytab_path=/path/to/mariadb.keytab
```

Configuring the Service Principal Name

The service principal name can be set by configuring the gssapi_principal_name system variable. This can be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
gssapi_principal_name=service_principal_name/host.domain.com@REALM
```

If a service principal name is not provided, the plugin tries to use `mariadb/host.domain.com@REALM` by default.

If the MariaDB server is running on Unix, the plugin needs a service principal name in order to function.

If the MariaDB server is running on Windows, the plugin does not usually need a service principal in order to function. However, if you want to use one, anyway, it can be created with the `setspn` utility.

Different KDC implementations may use different canonical forms to identify principals. See RFC2744: Section 3.10 to learn what the standard says about principal names.

More details can be found at the following links:

- Active Directory Domain Services: Service Principal Names
- MIT Kerberos Documentation: Realm configuration decisions
- MIT Kerberos Documentation: Principal names and DNS

Creating Users

To create a user account via CREATE USER, specify the name of the plugin in the IDENTIFIED VIA clause:

```
CREATE USER username@hostname IDENTIFIED VIA gssapi;
```

If SQL_MODE does not have `NO_AUTO_CREATE_USER` set, then you can also create the user account via GRANT:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA gssapi;
```

You can also specify the user's realm for MariaDB with the `USING` clause:

```
CREATE USER username@hostname IDENTIFIED VIA gssapi USING 'username@EXAMPLE.COM';
```

The format of the realm depends on the specific authentication mechanism that is used. For example, the format would need to be `machine\\username` for Windows users authenticating with NTLM.

If the realm is not provided in the user account's definition, then the realm is **not** used for comparison. Therefore, 'usr1@EXAMPLE.COM', 'usr1@EXAMPLE.CO.UK' and 'mymachine\usr1' would all identify as the following user account:

```
CREATE USER usr1@hostname IDENTIFIED VIA gssapi;
```

Creating Users Identified Via Group Membership or SID (Windows-specific)

On Windows only, it is possible to login using a AD or local group-membership. This is achieved by using the `GROUP` prefix in `IDENTIFIED ... AS`:

```
CREATE USER root IDENTIFIED VIA gssapi AS 'GROUP:Administrators'
CREATE USER root IDENTIFIED VIA gssapi AS 'GROUP:BUILTIN\\Administrators'
```

The effect of the above definition is that every user that identifies as a member of group `Administrators` can log in using the user name `root` without a password.

User can also login using own or group SID (Security Identifier):

```
CREATE USER root IDENTIFIED VIA gssapi AS 'SID:S-1-5-32-544'
```

Using SIDs will perform slightly faster than using name (since it will spare translation between SID and name which is otherwise done). SIDs are immune against user or group renaming.

Passwordless login on Windows

> **Note:**
>
> This feature is available from MariaDB 10.11.

On Windows, in addition to the usual authentication with a password, passwordless authentication is permitted when creating the `root` user during installation. This works in a similar manner to Unix socket authentication. However, since auth_gssapi, unlike unix_socket, requires client support, to avoid failures when MariaDB is used with third-party drivers, authentication on Windows first attempts password-based `native_authentication`, and only if it fails, falls back to passwordless `auth_gssapi`.

Client Authentication Plugins

For clients that use the `libmysqlclient` or MariaDB Connector/C libraries, MariaDB provides one client authentication plugin that is compatible with the `gssapi` authentication plugin:

- `auth_gssapi_client`

When connecting with a client or utility to a server as a user account that authenticates with the `gssapi` authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the `--plugin-dir` option:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

`auth_gssapi_client`

The `auth_gssapi_client` client authentication plugin receives the principal name from the server, and then uses either the gss_init_sec_context function (on Unix) or the InitializeSecurityContext function (on Windows) to establish a security context on the client.

Support in Client Libraries

Using the Plugin with MariaDB Connector/C

MariaDB Connector/C supports `gssapi` authentication using the client authentication plugins mentioned in the previous section.

Using the Plugin with MariaDB Connector/ODBC

MariaDB Connector/ODBC supports `gssapi` authentication using the client authentication plugins mentioned in the previous section.

Using the Plugin with MariaDB Connector/J

MariaDB Connector/J supports `gssapi` authentication. Current documentation can be found here.

Using the Plugin with MariaDB Connector/Node.js

MariaDB Connector/Node.js does not yet support `gssapi` authentication. See CONJS-72 for more information.

Using the Plugin with MySqlConnector for .NET

MySqlConnector for ADO.NET supports `gssapi` authentication.

The support is transparent. Normally, the connector only needs to be provided the correct user name, and no other parameters are required.

However, this connector also supports the ServerSPN connection string parameter, which can be used for mutual authentication.

.NET specific problems/workarounds

When connecting from Unix client to Windows server with ADO.NET, in an Active Directory domain environment, be aware that .NET Core on Unix does not support principal names in UPN(User Principal Name) form, which is default on Windows (e.g machine$@domain.com) . Thus, upon encountering an authentication exception with “server not found in Kerberos database”, use one of workarounds below

- Force host-based SPN on server side.
  - For example, this can be done by setting the gssapi_principal_name system variable to `HOST/machine` in a server option group in an option file.
- Pass host-based SPN on client side.
  - For example, this can be done by setting the connector’s ServerSPN connection string parameter to `HOST/machine`.

System Variables

`gssapi_keytab_path`

- Description: Defines the path to the server’s keytab file.
  - This system variable is only meaningful on Unix.
  - See Creating a Keytab File on Unix and Configuring the Path to the Keytab File on Unix for more information.
- Command line: `--gssapi-keytab-path`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: ’’

- Introduced: MariaDB 10.1.11

`gssapi_principal_name`

- Description: Name of the service principal.
  - See Configuring the Service Principal Name for more information.
- Command line: `--gssapi-principal-name`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: ''
- Introduced: MariaDB 10.1.11

`gssapi_mech_name`

- Description: Name of the SSPI package used by server. Can be either 'Kerberos' or 'Negotiate'. Set it to 'Kerberos', to prevent less secure NTLM in domain environments, but leave it as default (Negotiate) to allow non-domain environments (e.g if server does not run in a domain environment).
  - This system variable is only meaningful on Windows.
- Command line: `--gssapi-mech-name`
- Scope: Global
- Dynamic: No
- Data Type: `enumerated`
- Default Value: `Negotiate`
- Valid Values: `Kerberos`, `Negotiate`
- Introduced: MariaDB 10.1.11

Options

`gssapi`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--gssapi=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`
- Introduced: MariaDB 10.1.11

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - mysql_native_password

The `mysql_native_password` authentication plugin is the default authentication plugin that will be used for an account created when no authentication plugin is explicitly mentioned and old_passwords=0 is set. It uses the password hashing algorithm introduced in MySQL 4.1, which is also used by the PASSWORD() function when old_passwords=0 is set. This hashing algorithm is based on SHA-1.

It is not recommended to use the `mysql_native_password` authentication plugin for new installations that require **high password security**. If someone is able to both listen to the connection protocol and get a copy of the mysql.user table, then the person would be able to use this information to connect to the MariaDB server. The ed25519 authentication plugin is a more modern authentication plugin that provides simple password authentication using a more secure algorithm.

Installing the Plugin

The `mysql_native_password` authentication plugin is statically linked into the server, so no installation is necessary.

Creating Users

The easiest way to create a user account with the `mysql_native_password` authentication plugin is to make sure that old_passwords=0 is set, and then create a user account via CREATE USER that does not specify an authentication plugin, but does specify a password via the IDENTIFIED BY clause:

```
SET old_passwords=0;
CREATE USER username@hostname IDENTIFIED BY 'mariadb';
```

If SQL_MODE does not have `NO_AUTO_CREATE_USER` set, then you can also create the user account via GRANT:

```
SET old_passwords=0;
GRANT SELECT ON db.* TO username@hostname IDENTIFIED BY 'mariadb';
```

You can also create the user account by providing a password hash via the IDENTIFIED BY PASSWORD clause, and MariaDB will validate whether the password hash is one that is compatible with `mysql_native_password`:

```
SET old_passwords=0;

SELECT PASSWORD('mariadb');
+-------------------------------------------+
| PASSWORD('mariadb')                       |
+-------------------------------------------+
| *54958E764CE10E50764C2EECBB71D01F08549980 |
+-------------------------------------------+

CREATE USER username@hostname
  IDENTIFIED BY PASSWORD '*54958E764CE10E50764C2EECBB71D01F08549980';
```

Similar to all other authentication plugins, you could also specify the name of the plugin in the IDENTIFIED VIA clause while providing the password hash as the `USING` clause:

```
CREATE USER username@hostname
  IDENTIFIED VIA mysql_native_password USING '*54958E764CE10E50764C2EECBB71D01F08549980';
```

Changing User Passwords

You can change a user account's password with the SET PASSWORD statement while providing the plain-text password as an argument to the PASSWORD() function:

```
SET PASSWORD =  PASSWORD('new_secret')
```

You can also change the user account's password with the ALTER USER statement. You would have to make sure that old_passwords=0 is set, and then you would have to specify a password via the IDENTIFIED BY clause:

```
SET old_passwords=0;
ALTER USER username@hostname IDENTIFIED BY 'new_secret';
```

Client Authentication Plugins

For clients that use the `libmysqlclient` or MariaDB Connector/C libraries, MariaDB provides one client authentication plugin that is compatible with the `mysql_native_password` authentication plugin:

- `mysql_native_password`

When connecting with a client or utility to a server as a user account that authenticates with the `mysql_native_password` authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the `--plugin-dir` option:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

However, the `mysql_native_password` client authentication plugin is generally statically linked into client libraries like `libmysqlclient` or MariaDB Connector/C, so this is not usually necessary.

`mysql_native_password`

The `mysql_native_password` client authentication plugin hashes the password before sending it to the server.

Support in Client Libraries

The `mysql_native_password` authentication plugin is one of the conventional authentication plugins, so all client libraries should support it.

Known Old Issues (Only Relevant for Old Installations)

Mismatches Between Password and authentication_string Columns

For compatibility reasons, the `mysql_native_password` authentication plugin tries to read the password hash from both the `Password` and `authentication_string` columns in the mysql.user table. This has caused issues in the past if one of the columns had a different value than the other.

> **CURRENT**
>
> CREATE USER, ALTER USER, GRANT, and SET PASSWORD set the `Password` and `authentication_string` columns in the mysql.user table whenever an account's password is changed.
>
> **< 10.3.11 / 10.2.19**
>
> CREATE USER, ALTER USER, GRANT, and SET PASSWORD do **not** set the `Password` and `authentication_string` columns in the mysql.user table whenever an account's password is changed.

See Also

- ed25519 secure connection plugin
- History of MySQL and MariaDB authentication protocols

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - mysql_old_password

The `mysql_old_password` authentication plugin is the default authentication plugin that is used for an account created when no authentication plugin is explicitly mentioned and old_passwords=1 is set. It uses the pre-MySQL 4.1 password hashing algorithm, which is also used by the OLD_PASSWORD() function and by the PASSWORD() function when `old_passwords=1` is set.

> **Important:**
>
> It is not recommended to use the `mysql_old_password` authentication plugin for new installations. The password hashing algorithm is no longer secure, and the plugin is primarily provided for backward compatibility. The ed25519 authentication plugin is a more modern authentication plugin that provides simple password authentication.

Installing the Plugin

The `mysql_old_password` authentication plugin is statically linked into the server, so no installation is necessary.

Creating Users

The easiest way to create a user account with the `mysql_old_password` authentication plugin is to make sure that old_passwords=1 is set, and create a user account via CREATE USER that does not specify an authentication plugin, but instead specifies a password via the IDENTIFIED BY clause:

```
SET old_passwords=1;
CREATE USER username@hostname IDENTIFIED BY 'mariadb';
```

If SQL_MODE does not have `NO_AUTO_CREATE_USER` set, then you can also create the user via GRANT:

```
SET old_passwords=1;
GRANT SELECT ON db.* TO username@hostname IDENTIFIED BY 'mariadb';
```

You can also create the user account by providing a password hash via the IDENTIFIED BY PASSWORD clause, and MariaDB validates whether the password hash is one that is compatible with `mysql_old_password`:

```
SET old_passwords=1;
Query OK, 0 rows affected (0.000 sec)

SELECT PASSWORD('mariadb');
+--------------------+
| PASSWORD('mariadb') |
+--------------------+
| 021bec665bf663f1    |
+--------------------+
1 row in set (0.000 sec)

CREATE USER username@hostname IDENTIFIED BY PASSWORD '021bec665bf663f1';
Query OK, 0 rows affected (0.000 sec)
```

Similar to all other authentication plugins, you could also specify the name of the plugin in the `IDENTIFIED VIA` clause, while providing the password hash as the `USING` clause:

```
CREATE USER username@hostname IDENTIFIED VIA mysql_old_password USING '021bec665bf663f1';
Query OK, 0 rows affected (0.000 sec)
```

Changing User Passwords

You can change a user account's password with the SET PASSWORD statement, while providing the plain-text password as an argument to the PASSWORD() function:

```
SET PASSWORD =  PASSWORD('new_secret')
```

You can also change the user account's password with the ALTER USER statement. You have to make sure that old_passwords=1 is set, and you have to specify a password via the IDENTIFIED BY clause:

```
SET old_passwords=1;
ALTER USER username@hostname IDENTIFIED BY 'new_secret';
```

Client Authentication Plugins

For clients that use the `libmysqlclient` or MariaDB Connector/C libraries, MariaDB provides one client authentication plugin that is compatible with the `mysql_old_password` authentication plugin:

- `mysql_old_password`

When connecting with a client or utility to a server as a user account that authenticates with the `mysql_old_password` authentication plugin. You may need to tell the client where to find the relevant client authentication plugin by specifying the `--plugin-dir` option:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

However, the `mysql_old_password` client authentication plugin is generally statically linked into client libraries like `libmysqlclient` or MariaDB Connector/C, so this is not usually necessary.

`mysql_old_password`

The `mysql_old_password` client authentication plugin hashes the password before sending it to the server.

Support in Client Libraries

The `mysql_old_password` authentication plugin is one of the conventional authentication plugins, so all client libraries should support it.

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - Named Pipe

The `named_pipe` authentication plugin allows the user to use operating system credentials when connecting to MariaDB via named pipe on Windows. Named pipe connections are enabled by the named_pipe system variable.

The `named_pipe` authentication plugin works by using named pipe impersonation and calling `GetUserName()` to retrieve the user name of the process that is connected to the named pipe. Once it has the user name, it authenticates the connecting user as the MariaDB account that has the same user name.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'auth_named_pipe';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = auth_named_pipe
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'auth_named_pipe';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Creating Users

To create a user account via CREATE USER, specify the name of the plugin in the IDENTIFIED VIA clause:

```
CREATE USER username@hostname IDENTIFIED VIA named_pipe;
```

If SQL_MODE does not have `NO_AUTO_CREATE_USER` set, then you can also create the user account via GRANT:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA named_pipe;
```

Client Authentication Plugins

The `named_pipe` authentication plugin does not require any specific client authentication plugins. It should work with all clients.

Support in Client Libraries

The `named_pipe` authentication plugin does not require any special support in client libraries. It should work with all client libraries.

Example

```
CREATE USER wlad IDENTIFIED VIA named_pipe;
CREATE USER monty IDENTIFIED VIA named_pipe;
quit

C:\>echo %USERNAME%
wlad

C:\> mysql --user=wlad --protocol=PIPE
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.1.12-MariaDB-debug Source distribution

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> quit
Bye

C:\> mysql --user=monty  --protocol=PIPE
ERROR 1698 (28000): Access denied for user 'monty'@'localhost'
```

In this example, a user `wlad` is already logged into the system. Because he has identified himself to the operating system, he does not need to do it again for the database — MariaDB trusts the operating system credentials.

However, he cannot connect to the database as another user.

Options

`named_pipe`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
  - There may be ambiguity between this option and the named_pipe system variable. See MDEV-19625 about that.
- Command line: `--named-pipe=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`
- Introduced: MariaDB 10.1.11

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - PARSEC

> **Note:**
>
> This plugin is available from MariaDB 11.6.

The `PARSEC` Authentication Plugin is intended to be the default in a future release.

The `PARSEC` (Password Authentication using Response Signed with Elliptic Curve) authentication plugin uses salted passwords, key derivation, extensible password storage format, and both server- and client-side scrambles.

It signs the response with `ed25519`, but it uses stock unmodified `ed25519` as provided by OpenSSL/WolfSSL/GnuTLS.

Description

- The KDF function is pbkdf2 (supported by everything, including windows native, Java, javascript, PHP, .NET.
- Parameters to the pbkdf2 are stored in with authentication plugin data : hash function (SHA512,SHA256), iteration count, salt, key_length, together with derived key = `PBKDF2(func, password, salt, iteration_count, key_length).`
- The number of iterations is a power of two, at least 1024. From MariaDB 13.1, it is configurable through the `parsec_iterations` system variable.
- The algorithm is ed25519, “hash” is the public key generated using ed25519 from the PBKDF2(password).

The authentication string, stored by the server, is this:

```
concat('P', conv(log2(iterations)-10, 10, 62), ':', base64(salt), ':', base64(hash))
```

For example, it looks like this: `P0:WW9sXaaL/o:vubFBzIrapbfHct1/J72dnUryz5VS7lA6XHH8sIx4TI`

- It consists of colon-separated fields.
- The first field is 'P' (denotes KDF algorithm = PBKDF2) and the number of iterations, '0' means 1024, '1' means 2048, etc.
- This is followed by the salt.
- This is followed by the password hash.

The first two fields together are called *ext-salt*, extended salt.

Login Process, Packet Exchange

1. The server sends an Authentication Switch Request with a 32-byte random scramble.
2. The client sends an empty packet to the server to request the ext-salt.
3. The server sends the ext-salt to the client.
4. The client sends the random 32-byte scramble, and the `concat(server scramble, client scramble)` ed25519-signed by a secret key generated from the function `PBKDF2(password, ext-salt)`.
5. The server replies with "ok" or "access denied".

Installing

If you run into the error `ERROR 1524 (HY000): Plugin 'parsec' is not loaded` it means you need to install the authentication plugin first. You can do it on a running server with:

```
INSTALL SONAME 'auth_parsec';
```

There is no need to pass additional command-line options or have config files to keep the PARSEC authentication method available. Running the `INSTALL SONAME` once is enough and the MariaDB Server will remember it even if server is restarted or upgraded.

Example

```
CREATE USER test1@'%' IDENTIFIED VIA parsec USING PASSWORD('pwd');
```

Configuring the Number of Iterations

> **Note:**
>
> The `parsec_iterations` system variable is available from MariaDB 13.1.

The number of PBKDF2 iterations used to derive the key from a password is controlled by the `parsec_iterations` system variable. A higher value increases the cost of deriving the key, which strengthens resistance to brute-force attacks at the expense of more work per authentication.

The iteration count is baked into an account's stored credentials when its password is set. The value in effect at that time — the session value if set, otherwise the global value — is the one recorded for the account. Changing `parsec_iterations` afterwards does not alter existing accounts; they keep the iteration count they were created with until their password is set again.

```
SET SESSION parsec_iterations = 262144;
CREATE USER test1@'%' IDENTIFIED VIA parsec USING PASSWORD('pwd');
```

The value must be a power of two. If a value that is not a power of two is supplied, it is rounded up to the next power of two and a warning is issued:

```
SET GLOBAL parsec_iterations = 5000;
```

```
Warning (1231): parsec_iterations rounded up to 8192
```

`parsec_iterations`

- Description: Number of iterations used when generating the key corresponding to the password (PBKDF2-HMAC-SHA512). Rounded up to a power of two. This variable is only available when the PARSEC plugin is loaded.
- Commandline: `--parsec-iterations=#`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `INT UNSIGNED`
- Default Value: `1024`
- Minimum Value: `1024`
- Maximum Value: `1073741824`
- Introduced: MariaDB 13.1

Future

PARSEC is currently available in latest MariaDB versions, but not installed or used by default yet. Once MDEV-12320 is implemented, MariaDB plans to start using PARSEC as the default password authentication method.

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - SHA-256

Background

MySQL 5.6 added support for the sha256_password authentication plugin, and MySQL 8.0 also added support for the caching_sha2_password authentication plugin.

The `caching_sha2_password` plugin is now the default authentication plugin in MySQL 8.0.4 and above, based on the value of the default_authentication_plugin system variable.

Support in MariaDB Server

> **Warning:**
>
> MariaDB **Server** does not support the sha256_password plugin. A `caching_sha2_password` authentication plugin was added in MariaDB Community Server 12.1 and Enterprise Server 11.8. See MDEV-9804 for more information.

Reasons for not supporting the SHA-256 plugin:

- To use the protocol, you have to distribute the server's public key to all MariaDB users, which can be cumbersome and impractical.
- The server receives the password in clear text, which can cause problems if the user connects to a malicious server.

If you are migrating from a MySQL instance that is using SHA-256 authentication, you have to change the SHA-256 authentication to `mysql_native_authentication` :

```
ALTER USER user_name IDENTIFIED WITH mysql_native_password BY 'new_password'
```

Support in Client Libraries

Client Authentication Plugins

For clients that use the MariaDB Connector/C library, MariaDB provides client authentication plugins that are compatible with MySQL's SHA-256 authentication plugins:

- `sha256_password`
- `caching_sha256_password`

When connecting with a client or utility to a server, using a user account that authenticates with the `sha256_password` or `caching_sha256_password` authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the `--plugin-dir` option:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

> **Warning:**
>
> For clients that use MariaDB's `libmysqlclient` library instead of MariaDB Connector/C, those authentication plugins are not supported.

`sha256_password`

The `sha256_password` client authentication plugin is compatible with MySQL's sha256_password authentication plugin, which was added in MySQL 5.6.

`caching_sha256_password`

The `caching_sha256_password` client authentication plugin is compatible with MySQL's caching_sha2_password authentication plugin, which was added in MySQL 8.0.

The `caching_sha2_password` plugin is now the default authentication plugin in MySQL 8.0.4 and above, based on the value of the default_authentication_plugin system variable.

Using the Plugin with MariaDB Connector/C

MariaDB Connector/C supports `sha256_password` and `caching_sha2_password` authentication using the client authentication plugins mentioned in the previous section.

It has supported the `sha256_password` client authentication plugin since MariaDB Connector/C 3.0.2. See CONC-229 for more information.

It has supported the `caching_sha256_password` client authentication plugin since MariaDB Connector/C 3.0.8 and MariaDB Connector/C 3.1.0. See CONC-312 for more information.

Using Plugins with MariaDB Connector/ODBC

MariaDB Connector/ODBC supports `sha256_password` and `caching_sha2_password` authentication using the client authentication plugins mentioned in the previous section.

It has supported `sha256_password` and `caching_sha2_password` authentication since MariaDB Connector/ODBC 3.1.4. See ODBC-241 for more information.

Using Plugins with MariaDB Connector/J

MariaDB Connector/J supports `sha256_password` and `caching_sha2_password` authentication since MariaDB Connector/J 2.5.0. See CONJ-327 and CONJ-663 for more information.

note: The version 3.x being a rewrite of the connector, only caching_sha2_password is implemented, since sha256_password is only implemented on EOL version.

Using Plugins with MariaDB Connector/Node.js

MariaDB Connector/Node.js supports `sha256_password` and `caching_sha2_password` authentication since MariaDB Connector/Node.js 2.5.0. See CONJS-76 and CONJS-77 for more information.

See Also

- MDEV-9804 contains the plans to use if we ever decide to support these protocols.
- History of MySQL and MariaDB authentication protocols

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - Unix Socket

The `unix_socket` authentication plugin is installed by default, and it is used by the `'root'@'localhost'` user account by default. See Authentication for more information.

The `unix_socket` authentication plugin allows the user to use operating system credentials when connecting to MariaDB via the local Unix socket file. This Unix socket file is defined by the socket system variable.

The `unix_socket` authentication plugin works by calling the getsockopt system call with the `SO_PEERCRED` socket option, which allows it to retrieve the `uid` of the process that is connected to the socket. It is then able to get the user name associated with that `uid`. Once it has the user name, it will authenticate the connecting user as the MariaDB account that has the same user name.

The `unix_socket` authentication plugin is not suited to multiple Unix users accessing a single MariaDB user account.

Security

A `unix_socket` authentication plugin is a passwordless security mechanism. Its security lies in the strength of the access to the Unix user, rather than the complexity and the secrecy of the password.

> **Warning:**
>
> As security differs from password security, the strengths and weaknesses need to be considered, and those can differ depending on the specific installation.

Strengths

- Access is limited to the Unix user so, for example, a `www-data` user cannot access `root` with the `unix_socket` authentication plugin.
- There is no password which can be cracked by brute force.
- There is no password that can be accidentally exposed by user accident, poor security on backups, or poor security on passwords in configuration files.
- Default Unix user security is usually strong on preventing remote access and password brute force attempts.

Weaknesses

The strength of a `unix_socket` authentication plugin is effectively the strength of the security of the Unix users on the system. In most cases, the Unix user default installation is sufficiently secure. However, the following is a non-exhaustive list of potential Unix user security issues that may arise.

- Common access areas without screen locks, where an unauthorized user accesses the logged in Unix user of an authorized user.
- Extensive sudo access grants that provide users with access to execute commands of a different Unix user.
- Scripts writable by Unix users other than the Unix user that are executed (via cron or directly) by the Unix user.
- Web pages that are susceptible to command injection, where the Unix user running the web page has elevated privileges in the database that weren't intended to be used.
- Poor Unix user password practices including weak user passwords, password exposure and password reuse accompanied by an access vulnerability/mechanism of an unauthorized user to exploit this weakness.
- Weak remote access mechanisms and network file system privileges.
- Poor user security behavior including running untrusted scripts and software.

In some of these scenarios a database password may prevent these security exploits, however it will remove all the strengths of the `unix_socket` authentication plugin previously mentioned.

Disabling the Plugin

The `unix_socket` authentication plugin is installed by default.

> **Warning:**
>
> If you do not want it to be available by default, you must disable it.

The `unix_socket` authentication plugin is also installed by default in new installations that use the .deb packages provided by Debian's default repositories and Ubuntu's default repositories. See Differences in MariaDB in Debian (and Ubuntu) for more information.

The `unix_socket` authentication plugin can be disabled by starting the server with the unix_socket option set to `OFF`. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
unix_socket=OFF
```

As an alternative, the unix_socket option can also be set to `OFF` by pairing the option with the `disable` option prefix:

```
[mariadb]
...
disable_unix_socket
```

Installing the Plugin

The `unix_socket` authentication plugin is installed by default in almost all MariaDB server versions. If you work with a version that doesn't have the plugin installed, you can install it as described in one of the following ways.

- Install the plugin without restarting the server. You can install the plugin dynamically, by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'auth_socket';
```

- Instruct the server to load the plugin at startup. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = auth_socket
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'auth_socket';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a server option group in an option file, those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Creating Users

To create a user account via CREATE USER, specify the name of the plugin in the IDENTIFIED VIA clause:

```
CREATE USER username@hostname IDENTIFIED VIA unix_socket;
```

If SQL_MODE does not have `NO_AUTO_CREATE_USER` set, then you can also create the user account via GRANT:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA unix_socket;
```

CURRENT

The authentication string (if present) is compared with the socket's user name. Authentication proceeds if there's a match. In this case, the external_user system variable contains the OS user.

Consider an OS user named 'bob' that has been created like this:

```
CREATE USER A identified via unix_socket as 'bob';
```

That user can connect like this:

```
mariadb -uA
```

Alternatively, accessing the sock file directly, the user can connect like this:

```
mariadb -uA -S /var/run/mysqld/mysqld.sock
```

Once connected, you can view that user like this:

```
SELECT USER(),@@external_user;
+-------------+-----------------+
| user()      | @@external_user |
+-------------+-----------------+
| A@localhost | bob             |
+-------------+-----------------+
```

< 11.6

The plugin only checks whether the OS socket user id matches the MariaDB user name. It ignores the authentication string.

Switching to Password-Based Authentication

If Unix socket authentication does not meet your needs, you can switch a user account back to password-based authentication, by telling MariaDB to use a different authentication plugin for the account. The specific authentication plugin is specified with the IDENTIFIED VIA clause. To switch to the mysql_native_password authentication plugin, you need to do this:

```
ALTER USER root@localhost IDENTIFIED VIA mysql_native_password;
SET PASSWORD = PASSWORD('foo');
```

Warning:

If you use scripts that require passwordless access to MariaDB, this would cause them to break. You may be able to fix that by setting a password in the `[client]` option group in your /root/.my.cnf option file.

```
[client]
password=foo
```

Client Authentication Plugins

The `unix_socket` authentication plugin does not require any specific client authentication plugins. It should work with all clients.

Support in Client Libraries

The `unix_socket` authentication plugin does not require any special support in client libraries. It should work with all client libraries.

Example

```
$ mysql -uroot
MariaDB []> CREATE USER serg IDENTIFIED VIA unix_socket;
MariaDB []> CREATE USER monty IDENTIFIED VIA unix_socket;
MariaDB []> quit
Bye
$ whoami
serg
$ mysql --user=serg
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.2.0-MariaDB-alpha-debug Source distribution
MariaDB []> quit
Bye
$ mysql --user=monty
ERROR 1045 (28000): Access denied for user 'monty'@'localhost' (using password: NO)
```

In this example, user `serg` is already logged into the operating system and has full shell access. The user has already authenticated with the operating system and the MariaDB account is configured to use the `unix_socket` authentication plugin, so there is no need to authenticate again for the database. MariaDB accepts the operating system credentials and allows the user to connect. However, any attempt to connect to the database as another operating system is denied.

Options

`unix_socket`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugin table.

    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--unix-socket=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

See Also

- Differences in MariaDB in Debian (and Ubuntu)
- Authentication
- Authentication from MariaDB 10 4 video tutorial

This page is licensed: CC BY-SA / Gnu FDL

Authentication with Pluggable Authentication Modules (PAM)

See Authentication Plugin - PAM.

The PAM authentication plugin delegates password validation to the operating system's PAM framework, enabling integration with LDAP, Kerberos, and other services.

See Configuring PAM Authentication and User Mapping with LDAP Authentication.

Learn to configure the PAM plugin to authenticate users via LDAP and map LDAP groups to MariaDB accounts using the pam_user_map module.

See Configuring PAM Authentication and User Mapping with Unix Authentication.

This guide shows how to authenticate database users using local Unix accounts and map Unix groups to MariaDB users with the PAM plugin.

See User and Group Mapping with PAM.

The pam_user_map PAM module allows administrators to map external PAM users and groups to specific MariaDB accounts for flexible authorization management.

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin - PAM

Overview

The `pam` authentication plugin allows MariaDB to offload user authentication to the system's Pluggable Authentication Module (PAM) framework. PAM is an authentication framework used by Linux, FreeBSD, Solaris, and other Unix-like operating systems.

Note: Windows does not support PAM, so the `pam` authentication plugin does not support Windows. However, one can use a MariaDB client on Windows to connect to MariaDB server that is installed on a Unix-like operating system and that is configured to use the `pam` authentication plugin. For an example of how to do this, see the blog post: MariaDB: Improve Security with Two-Step Verification.

Use Cases

PAM makes it possible to implement various authentication scenarios of different complexity:

- Authentication using passwords from `/etc/shadow` (this is what a default PAM configuration usually does). See the pam_unix PAM module.
- Authentication using LDAP. See the pam_ldap PAM module.
- Authentication using Microsoft's Active Directory. See the pam_lsass, pam_winbind, and pam_centrifydc PAM modules.
- Authentication using one-time passwords (even with SMS confirmation!). See the pam_google_authenticator and pam_securid PAM modules.
- Authentication using SSH keys. See the pam_ssh PAM module.
- User and group mapping. See the pam_user_map PAM module.
- Combining different authentication modules in interesting ways in a PAM service.
- Password expiration.
- Limiting access by time, date, day of the week, etc. See the pam_time PAM module.
- Logging of every login attempt.

Installing the Plugin

The `pam` authentication plugin's library is provided in binary packages in all releases on Linux.

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'auth_pam';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = auth_pam
```

Installing the v1 Plugin

The `auth_pam` shared library actually refers to version `2.0` of the `pam` authentication plugin. Version `1.0` of the plugin as the `auth_pam_v1` shared library is also available.

If you need to install version `1.0` of the authentication plugin instead of version `2.0`, install it with INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'auth_pam_v1';
```

Alternatively, specify it in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = auth_pam_v1
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'auth_pam';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, those options must be removed to prevent the plugin from being loaded the next time the server is restarted.

Uninstalling the v1 Plugin

If you installed version `1.0` of the authentication plugin, you can uninstall it by executing a similar statement for `auth_pam_v1`:

```
UNINSTALL SONAME 'auth_pam_v1';
```

Configuring PAM

The `pam` authentication plugin tells MariaDB to delegate the authentication to the PAM authentication framework. How exactly that authentication is performed depends on how PAM is configured.

Configuring the PAM Service

PAM is divided into `services`. PAM services are configured by PAM configuration files. Typically, the global PAM configuration file is located at `/etc/pam.conf` and PAM directory-based configuration files for individual services are located in `/etc/pam.d/`.

If you want to use a PAM service called `mariadb` for your MariaDB PAM authentication, then the PAM configuration file for that service would also be called `mariadb`, and it would typically be located at `/etc/pam.d/mariadb`.

For example, here is a minimal PAM service configuration file that performs simple password authentication with UNIX passwords:

```
auth required pam_unix.so audit
account required pam_unix.so audit
```

Let's breakdown this relatively simple PAM service configuration file.

Each line of a PAM service configuration file has the following general format:

```
type control module-path module-arguments
```

It instructs the PAM authentication framework that for successful **authentication** (i.e. type=auth), it is **required** that the `pam_unix.so` PAM module returns a success. It also instructs the PAM authentication framework that for an **account** (i.e. type=account) to be valid, it is **required** that the `pam_unix.so` PAM module returns a success.

PAM also supports session and password types, but MariaDB's `pam` authentication plugin does not support those.

The above PAM service configuration file also provides the `audit` module argument to the `pam_unix` PAM module. The pam_unix manual says that this module argument enables extreme debug logging to the syslog.

On most systems, you can find many other examples of PAM service configuration files in your `/etc/pam.d/` directory.

Configuring the pam_unix PAM Module

If you configure PAM to use the pam_unix PAM module (as in the above example), then you might notice on some systems that this will fail by default with errors like the following:

```
Apr 14 12:56:23 localhost unix_chkpwd[3332]: check pass; user unknown
Apr 14 12:56:23 localhost unix_chkpwd[3332]: password check failed for user (alice)
Apr 14 12:56:23 localhost mysqld: pam_unix(mysql:auth): authentication failure; logname= uid=991 euid=991
tty= ruser= rhost=  user=alice
```

The problem is that on some systems, the `pam_unix` PAM module needs access to `/etc/shadow` in order to function, and most systems only allow `root` to access that file by default.

Newer versions of PAM do not have this limitation, so you may want to try upgrading your version of PAM to see if that fixes the issue.

If that does not work, then you can work around this problem by giving the user that runs mysqld access to `/etc/shadow`. For example, if the `mysql` user runs mysqld, then you could do the following:

```
sudo groupadd shadow
sudo usermod -a -G shadow mysql
sudo chown root:shadow /etc/shadow
sudo chmod g+r /etc/shadow
```

After configuring, you have to restart the server. The server should now be able to read `/etc/shadow`.

The `pam` authentication plugin uses a setuid wrapper to perform its PAM checks, so it should not need any special workarounds to perform privileged operations, such as reading `/etc/shadow` when using the `pam_unix` PAM module. See MDEV-7032 for more information.

Creating Users

To create a user in MariaDB which uses the `pam` authentication plugin, execute CREATE USER while specifying the name of the plugin in the IDENTIFIED VIA clause:

```
CREATE USER username@hostname IDENTIFIED VIA pam;
```

If SQL_MODE does not have `NO_AUTO_CREATE_USER` set, then you can also create the user this way with GRANT:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA pam;
```

You can also specify a PAM service name for MariaDB to use by providing it with the `USING` clause:

```
CREATE USER username@hostname IDENTIFIED VIA pam USING 'mariadb';
```

This line creates a user that needs to be authenticated via the `pam` authentication plugin using the PAM service name `mariadb`. As mentioned in a previous section, this service's configuration file will typically be present in `/etc/pam.d/mariadb`.

If no service name is specified, then the plugin will use `mysql` as the default PAM service name.

Client Authentication Plugins

For clients that use the `libmysqlclient` or MariaDB Connector/C libraries, MariaDB provides two client authentication plugins that are compatible with the `pam` authentication plugin:

- `dialog`
- `mysql_clear_password`

When connecting with a client or utility to a server as a user account that authenticates with the `pam` authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the `--plugin-dir` option:

```
mariadb --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

Both the `dialog` and the `mysql_clear_password` client authentication plugins transmit the password to the server in clear text. Therefore, when you use the `pam` authentication plugin, it is very important to encrypt client connections using TLS to prevent the clear-text passwords from being seen by unauthorized users.

`dialog`

Usually the `pam` authentication plugin uses the `dialog` client authentication plugin to communicate with the user. This client authentication plugin allows MariaDB to support arbitrarily complex PAM configurations with regular or one-time passwords, challenge-response, multiple questions, or just about anything else. When using a MariaDB client library, there is no need to install or enable anything — the `dialog` client authentication plugin is loaded by the client library completely automatically and transparently for the application.

The `dialog` client authentication plugin was developed by MariaDB, so MySQL's clients and client libraries as well as third party applications that bundle MySQL's client libraries do not support the `dialog` client authentication plugin out of the box. If the server tells an unsupported client to use the `dialog` client authentication plugin, then the client is likely to throw an error like the following:

```
ERROR 2059 (HY000): Authentication plugin 'dialog' cannot be loaded: /usr/lib/mysql/plugin/dialog.so:
cannot open shared object file: No such file or directory
```

For some libraries or applications, this problem can be fixed by copying `dialog.so` or `dialog.dll` from a MariaDB client installation that is compatible with the system into the system's MySQL client authentication plugin directory. However, not all clients are compatible with the `dialog` client authentication plugin, so this may not work for every client.

If your client does not support the `dialog` client authentication plugin, then you may need to use the mysql_clear_password client authentication plugin instead.

The `dialog` client authentication plugin transmits the password to the server in clear text. Therefore, when you use the `pam` authentication plugin, it is incredibly important to encrypt client connections using TLS to prevent the clear-text passwords from being seen by unauthorized users.

`mysql_clear_password`

Users can instruct the `pam` authentication plugin to use the `mysql_clear_password` client authentication plugin instead of the dialog client authentication plugin by configuring the pam_use_cleartext_plugin system variable on the server. It can be set in a relevant server option group in an option file:

```
[mariadb]
...
pam_use_cleartext_plugin
```

It is important to note that the `mysql_clear_password` plugin has very limited functionality.

- The `mysql_clear_password` client authentication plugin only supports PAM services that require password-based authentication.
- The `mysql_clear_password` client authentication plugin also only supports PAM services that ask the user a single question.
- If the PAM service requires challenge-responses, multiple questions, or other similar complicated authentication schemes, then the PAM service is not compatible with `mysql_clear_password` client authentication plugin. In that case, the dialog client authentication plugin will have to be used instead.

The `mysql_clear_password` client authentication plugin transmits the password to the server in clear text. Therefore, when you use the `pam` authentication plugin, it is incredibly important to encrypt client connections using

TLS to prevent the clear-text passwords from being seen by unauthorized users.

Compatibility with MySQL Clients and Client Libraries

The `mysql_clear_password` client authentication plugin is similar to MySQL's mysql_clear_password client authentication plugin.

The `mysql_clear_password` client authentication plugin is compatible with MySQL clients and most MySQL client libraries, while the dialog client authentication plugin is not always compatible with them. Therefore, the `mysql_clear_password` client authentication plugin is most useful if you need some kind of MySQL compatibility in your environment, but you still want to use the `pam` authentication plugin.

Even though the `mysql_clear_password` client authentication plugin is compatible with MySQL clients and most MySQL client libraries, the `mysql_clear_password` client authentication plugin may be disabled by default by these clients and client libraries. For example, MySQL's version of the mysql command-line client has the –enable-cleartext-plugin option that must be set in order to use the `mysql_clear_password` client authentication plugin:

```
mysql --enable-cleartext-plugin --user=alice -p
```

Other clients may require other methods to enable the authentication plugin. For example, MySQL Workbench has a checkbox titled **Enable Cleartext Authentication Plugin** under the **Advanced** tab on the connection configuration screen.

For applications that use MySQL's `libmysqlclient`, the authentication plugin can be enabled by setting the `MYSQL_ENABLE_CLEARTEXT_PLUGIN` option with the `mysql_options()` function:

```
mysql_options(mysql, MYSQL_ENABLE_CLEARTEXT_PLUGIN, 1);
```

For MySQL compatibility, MariaDB Connector/C also allows applications to set the `MYSQL_ENABLE_CLEARTEXT_PLUGIN` option with the mysql_optionsv function. However, this option does not actually do anything in MariaDB Connector/C, because the `mysql_clear_password` client authentication plugin is always enabled for MariaDB clients and client libraries.

Support in Client Libraries

Using the Plugin with MariaDB Connector/C

MariaDB Connector/C supports `pam` authentication using the client authentication plugins, regardless of the value of the pam_use_cleartext_plugin system variable.

Using the Plugin with MariaDB Connector/ODBC

MariaDB Connector/ODBC supports `pam` authentication using the client authentication plugins, regardless of the value of the pam_use_cleartext_plugin system variable.

Using the Plugin with MariaDB Connector/J

MariaDB Connector/J supports `pam v1` authentication, regardless of the value of the pam_use_cleartext_plugin system variable.

MariaDB Connector/J supports `pam v2` authentication, regardless of the value of the pam_use_cleartext_plugin system variable.

Using the Plugin with MariaDB Connector/Node.js

MariaDB Connector/Node.js supports `pam` authentication, regardless of the value of the pam_use_cleartext_plugin system variable.

Using the Plugin with MySqlConnector for .NET

MySqlConnector for ADO.NET supports `pam` authentication, but only if the pam_use_cleartext_plugin system variable is enabled on the server.

Logging

MariaDB PAM authentication can be collected from a variety of sources in the authentication framework, including the underlying PAM modules, the PAM authentication plugin, and custom scripts like pam_exec. The destination of these logs is decided by the layer that generated message and the configuration of your system.

PAM Module Logging

Errors and messages from PAM modules are usually logged using the syslog daemon with the `authpriv` facility. To determine the specific log file where the `authpriv` facility is logged, you can check rsyslog.conf.

On RHEL, CentOS, Fedora, and other similar Linux distributions, the default location for these messages is usually `/var/log/secure`.

On Debian, Ubuntu, and other similar Linux distributions, the default location for these messages is usually `/var/log/auth.log`.

For example, the syslog can contain messages like the following when MariaDB's `pam` authentication plugin is configured to use the pam_unix PAM module, and the user enters an incorrect password:

```
Jan  9 05:35:41 ip-172-30-0-198 unix_chkpwd[1205]: password check failed for user (foo)
Jan  9 05:35:41 ip-172-30-0-198 mysqld: pam_unix(mariadb:auth): authentication failure; logname= uid=997
euid=997 tty= ruser= rhost=  user=foo
```

PAM Authentication Plugin's Debug Logging

MariaDB's `pam` authentication plugin can also log additional verbose debug logging to the error log. This is only done if the plugin is a debug build and if pam_debug is set.

The output looks like this:

```
PAM: pam_start(mariadb, alice)
PAM: pam_authenticate(0)
PAM: conv: send(Enter PASSCODE:)
PAM: conv: recv(123456789)
PAM: pam_acct_mgmt(0)
PAM: pam_get_item(PAM_USER)
PAM: status = 0 user = ��\>
```

Custom Logging with pam_exec

The pam_exec PAM module can be used to implement some custom logging. This can be very useful when debugging certain kinds of issues.

For example, first, create a script that writes the log output:

```
tee /tmp/pam_log_script.sh <<EOF
#!/bin/bash
echo "\${PAM_SERVICE}:\${PAM_TYPE} - \${PAM_RUSER}@\${PAM_RHOST} is authenticating as \${PAM_USER}"
EOF
chmod 0775 /tmp/pam_log_script.sh
```

And, then, change the PAM service configuration to execute the script using the pam_exec PAM module. For example:

```
auth optional pam_exec.so log=/tmp/pam_output.txt /tmp/pam_log_script.sh
auth required pam_unix.so audit
account optional pam_exec.so log=/tmp/pam_output.txt /tmp/pam_log_script.sh
account required pam_unix.so audit
```

Whenever the above PAM service is used, the output of the script is written to `/tmp/pam_output.txt`. It looks similar to this output:

```
*** Tue May 14 14:53:23 2019
mariadb:auth - @ is authenticating as alice
*** Tue May 14 14:53:25 2019
mariadb:account - @ is authenticating as alice
*** Tue May 14 14:53:28 2019
mariadb:auth - @ is authenticating as alice
*** Tue May 14 14:53:31 2019
mariadb:account - @ is authenticating as alice
```

User and Group Mapping

Even when using the `pam` authentication plugin, the authenticating PAM user account still needs to exist in MariaDB, and the account needs to have privileges in the database. Creating these MariaDB accounts and making sure the privileges are correct can be a lot of work. To decrease the amount of work involved, some users would like to be able to map a PAM user to a different MariaDB user. For example, let's say that `alice` and `bob` are both DBAs. It would be nice if each of them could log into MariaDB with their own PAM username and password, while MariaDB sees both of them as the same `dba` user. That way, there is only one MariaDB account to keep track of. See User and Group Mapping with PAM for more information on how to do this.

PAM Modules

There are many PAM modules. The ones described below are the ones that have been seen most often by MariaDB. Several of them — including pam_sss, pam_winbind, pam_lsass, pam_centrifydc, and pam_krb5 — can authenticate against Microsoft Active Directory.

> **Note:**
>
> MariaDB supports the `pam` authentication plugin itself, not the individual PAM modules described below. These modules are third-party software, and the links to their documentation are provided for convenience only. MariaDB doesn't test, validate, or maintain them and can't list specific modules as officially supported. Consult each module's own documentation, and validate the configuration in your environment before relying on it in production.

pam_unix

The pam_unix PAM module provides support for Unix password authentication. It is the default PAM module on most systems.

For a tutorial on setting up PAM authentication and user or group mapping with Unix authentication, see Configuring PAM Authentication and User Mapping with Unix Authentication.

pam_user_map

The pam_user_map PAM module was developed by MariaDB to support user and group mapping.

pam_ldap

The pam_ldap PAM module provides support for LDAP authentication.

For a tutorial on setting up PAM authentication and user or group mapping with LDAP authentication, see Configuring PAM Authentication and User Mapping with LDAP Authentication.

This can also be configured for Active Directory authentication.

pam_sss

The pam_sss PAM module provides support for authentication with System Security Services Daemon (SSSD).

This can be configured for Active Directory authentication.

pam_lsass

The `pam_lsass` PAM module provides support for Active Directory authentication. It is provided by PowerBroker Identity Services – Open Edition.

pam_winbind

The pam_winbind PAM module provides support for Active Directory authentication. It is provided by winbindd from the samba suite.

This PAM module converts all provided user names to lowercase. There is no way to disable this functionality. If you do not want to be forced to use all lowercase user names, then you may need to configure the pam_winbind_workaround system variable. See MDEV-18686 for more information.

pam_centrifydc

The pam_centrifydc PAM module provides support for Active Directory authentication. It integrates with the commercial Active Directory Bridge from Centrify.

pam_krb5

The pam_krb5 PAM module provides support for Kerberos authentication.

This can be configured for Active Directory authentication.

pam_google_authenticator

The `pam_google_authenticator` PAM module provides two-factor identification with Google Authenticator. It is from Google's google-authenticator-libpam open-source project. The PAM module should work with the open-source mobile apps built by Google's google-authenticator and google-authenticator-android projects as well as the closed source Google Authenticator mobile apps that are present in each mobile app store.

For an example of how to use this PAM module, see the blog post: MariaDB: Improve Security with Two-Step Verification.

pam_securid

The `pam_securid` PAM module provides support for multi-factor authentication. It is part of the commercial RSA SecurID Suite.

Note that current versions of this module are not safe for multi-threaded environments, and the vendor does not officially support the product on MariaDB. See MDEV-10361 about that. However, the module may work with a current version of MariaDB.

pam_ssh

The pam_ssh PAM module provides authentication using SSH keys.

pam_time

The pam_time PAM module provides time-controlled access.

Known Issues

Multi-Threaded Issues

MariaDB is a multi-threaded program, which means that different connections concurrently run in different threads. Current versions of MariaDB's `pam` authentication plugin execute PAM module code in the server address space. This means that any PAM modules used with MariaDB must be safe for multi-threaded environments. Otherwise, if multiple clients try to authenticate with the same PAM module in parallel, undefined behavior can occur. For example, the `pam_fprintd` PAM module is not safe for multi-threaded environments, and if you use it with MariaDB, you may experience server crashes.

The `pam` authentication plugin isolates PAM module code from the server address space, so even PAM modules that are known to be unsafe for multi-threaded environments should not cause issues with MariaDB. See MDEV-15473 for more information.

Conflicts with Password Validation

When a password validation plugin is enabled, MariaDB won't allow an account to be created if the password validation plugin says that the account's password is too weak. This creates a problem for accounts that authenticate with the `pam` authentication plugin, since MariaDB has no knowledge of the user's password. When a user tries to create an account that authenticates with the `pam` authentication plugin, the password validation plugin would throw an error, even with strict_password_validation=OFF set.

The workaround is to uninstall the password validation plugin with UNINSTALL PLUGIN, and then create the account, and then reinstall the password validation plugin with INSTALL PLUGIN:

```
INSTALL PLUGIN simple_password_check SONAME 'simple_password_check';
Query OK, 0 rows affected (0.002 sec)

SET GLOBAL strict_password_validation=OFF;
Query OK, 0 rows affected (0.000 sec)

CREATE USER ''@'%' IDENTIFIED VIA pam USING 'mariadb';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
UNINSTALL PLUGIN simple_password_check;
Query OK, 0 rows affected (0.000 sec)

CREATE USER ''@'%' IDENTIFIED VIA pam USING 'mariadb';
Query OK, 0 rows affected (0.000 sec)

INSTALL PLUGIN simple_password_check SONAME 'simple_password_check';
Query OK, 0 rows affected (0.001 sec)
```

> **Warning:**
>
> Accounts that authenticate with the `pam` authentication plugin should be exempt from password validation checks. See MDEV-12321 and MDEV-10457 for more information.

SELinux

SELinux may cause issues when using the `pam` authentication plugin. For example, using pam_unix with the `pam` authentication plugin while SELinux is enabled can sometimes lead to SELinux errors involving unix_chkpwd, such as the following::

```
Apr 14 12:37:59 localhost setroubleshoot: Plugin Exception restorecon_source
Apr 14 12:37:59 localhost setroubleshoot: SELinux is preventing /usr/sbin/unix_chkpwd from execute access
on the file . For complete SELinux messages. run sealert -l c56fe6e0-c78c-4bdb-a80f-27ef86a1ea85
Apr 14 12:37:59 localhost python: SELinux is preventing /usr/sbin/unix_chkpwd from execute access on the
file .

*****  Plugin catchall (100. confidence) suggests   *************************

If you believe that unix_chkpwd should be allowed execute access on the  file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# grep unix_chkpwd /var/log/audit/audit.log | audit2allow -M mypol
# semodule -i mypol.pp
```

Sometimes issues like this can be fixed by updating the system's SELinux policies. You may be able to update the policies using audit2allow. See SELinux: Generating SELinux Policies with audit2allow for more information.

If you can't get the `pam` authentication plugin to work with SELinux at all, then it can help to disable SELinux entirely. See SELinux: Changing SELinux's Mode for information on how to do this.

Memory Overcommit

You may run into authentication failures with the following log message in the MariaDB error log:

```
pam: cannot exec /usr/lib64/mysql/plugin/auth_pam_tool_dir/auth_pam_tool (errno: 12 "Cannot allocate
memory")
```

This can happen on operating system setups that are configured to prevent memory overcommit. When the MariaDB server process spawns the `auth_pam_tool` helper process there's a brief period where the new process inherits the memory of the MariaDB process before releasing that memory and executing the new command. When having a MariaDB server configured to use more than 50% of the server machine's RAM — which is common for dedicated database servers — this duplication would lead to an over-commit situation.

> **CURRENT**
>
> Current MariaDB version use `posix_spawn()` instead of the classic `fork();exec()` to prevent memory overcommit issues. However, operating systems with older glibc versions (< 2.26) still use `fork();exec()` to implement `posix_spawn()` internally. Therefore, they are still affected; this is for example still the case on RedHat Enterprise Linux 7.
>
> **< 10.7.4 / 10.6.8 / 10.5.16**
>
> Older MariaDB versions still use `fork();exec()` , which means they are affected by the memory overcommit issue.

To solve this you can do this:

- Change the `vm.overcommit_memory` kernel setting to allow memory overcommit.
- Alternatively, install the older `auth_pam_v1` plugin version that does not spawn a helper process (but may run into problems with file permissions or multi threading with some PAM modules).

See also MDEV-26212 and MDEV-3073.

Tutorials

- Configuring PAM Authentication and User Mapping with Unix Authentication
- Configuring PAM Authentication and User Mapping with LDAP Authentication

System Variables

`pam_debug`

- Description: Enables verbose debug logging to the error log for all authentication handled by the plugin.
  - This system variable is only available when the plugin is a debug build.
- Command line: `--pam-debug`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.2.2, MariaDB 10.1.17

`pam_use_cleartext_plugin`

- Description: Use the mysql_clear_password client authentication plugin instead of the dialog client authentication plugin. This may be needed for compatibility reasons, but it only supports simple PAM configurations that don't require any input besides a password.
- Command line: `--pam-use-cleartext-plugin`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.1.1, MariaDB 5.5.32

`pam_winbind_workaround`

- Description: Configures the authentication plugin to compare the user name provided by the client with the user name returned by the PAM module in a case insensitive manner. This may be needed if you use the pam_winbind PAM module, which is known to convert all user names to lowercase, and which does not allow this behavior to be disabled.
- Command line: `--pam-winbind-workaround`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`
- Introduced: MariaDB 10.4.5, MariaDB 10.3.15, MariaDB 10.2.24, MariaDB 10.1.39

Options

`pam`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL

PLUGIN while the server is running.

- See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.

- Command line: `--pam=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

See Also

- Writing a MariaDB PAM Authentication Plugin
- MariaDB: Improve Security with Two-Step Verification

This page is licensed: CC BY-SA / Gnu FDL

Configuring PAM Authentication and User Mapping with LDAP Authentication

In this article, we will walk through the configuration of PAM authentication using the pam authentication plugin and user and group mapping with the pam_user_map PAM module. The primary authentication will be handled by the pam_ldap PAM module, which performs LDAP authentication. We will also set up an OpenLDAP server.

Hypothetical Requirements

In this walkthrough, we are going to assume the following hypothetical requirements:

- The LDAP user `foo` should be mapped to the MariaDB user `bar`. (`foo: bar`)
- Any LDAP user in the LDAP group `dba` should be mapped to the MariaDB user `dba`. (`@dba: dba`)

Setting up the OpenLDAP Server

Before we can use LDAP authentication, we first need to set up our OpenLDAP Server. This is usually done on a server that is completely separate from the database server.

Installing the OpenLDAP Server and Client Components

On the server acting as the OpenLDAP Server, first, we need to install the OpenLDAP components.

On RHEL, CentOS, and other similar Linux distributions that use RPM packages, that would go like this:

```
sudo yum install openldap openldap-servers openldap-clients nss-pam-ldapd
```

Configuring the OpenLDAP Server

Next, let's to configure the OpenLDAP Server. The easiest way to do that is to copy the template configuration file that is included with the installation. In many installations, that will be at `/usr/share/openldap-servers/DB_CONFIG.example`:

```
sudo cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
sudo chown ldap. /var/lib/ldap/DB_CONFIG
```

Configuring the OpenLDAP Port

Sometimes it is useful to change the port used by OpenLDAP. For example, some cloud environments block well-known authentication services, so they block the default LDAP port.

On some systems, the port can be changed by setting `SLAPD_URLS` in `/etc/sysconfig/slapd`:

```
SLAPD_URLS="ldapi:/// ldap://0.0.0.0:3306/"
```

I used `3306` because that is the port that is usually used by `mysqld`, so I know that it is not blocked.

Starting the OpenLDAP Server

Next, let's start the OpenLDAP Server and configure it to start on reboot. On systemd systems, that would go like this:

```
sudo systemctl start slapd
sudo systemctl enable slapd
```

Installing the Standard LDAP objectClasses

In order to use LDAP for authentication, we also need to install some standard `objectClasses`, such as `posixAccount` and `posixGroup`. In LDAP, things like `objectClasses` are defined in LDIF files. In many installations, these specific `objectClasses` are defined in `/etc/openldap/schema/nis.ldif`. `nis.ldif` also depends on `core.ldif` and `cosine.ldif`. However, `core.ldif` is usually installed by default.

We can install them with ldapmodify:

```
sudo ldapmodify -a -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif
sudo ldapmodify -a -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif
```

Creating the LDAP Directory Manager User

Next, let's create a directory manager user. We can do this by using OpenLDAP's olc configuration system to change the olcRootDN directive to the DN of the directory manager user, which means that the user will be a privileged LDAP user that is not subject to access controls. We will also set the root password for the user by changing the olcRootPW directive.

We will also set the DN suffix for our backend LDAP database by changing the olcSuffix directive.

Let's use the slappasswd utility to generate a password hash from a clear-text password. Simply execute:

```
slappasswd
```

This utility provides a password hash that looks like this: `{SSHA}AwT4jrvmokeCkbDrFAnGvzzjCMb7bvEl`

OpenLDAP's olc configuration system also uses LDIF files. Now that we have the password hash, let's create an `LDIF` file to create the directory manager user:

```
tee ~/setupDirectoryManager.ldif <<EOF
dn: olcDatabase={1}monitor,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to *
    by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" read
    by dn.base="cn=Manager,dc=support,dc=mariadb,dc=com" read
    by * none

dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=support,dc=mariadb,dc=com

dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=Manager,dc=support,dc=mariadb,dc=com

dn: olcDatabase={2}hdb,cn=config
changetype: modify
add: olcRootPW
olcRootPW: {SSHA}AwT4jrvmokeCkbDrFAnGvzzjCMb7bvEl

dn: olcDatabase={2}hdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {0}to attrs=userPassword,shadowLastChange
    by   dn="cn=Manager,dc=support,dc=mariadb,dc=com" write
    by anonymous auth
    by self write
    by * none
olcAccess: {1}to dn.base=""
    by * read
olcAccess: {2}to *
    by dn="cn=Manager,dc=support,dc=mariadb,dc=com" write
    by * read
EOF
```

Note that this is using the `dc=support,dc=mariadb,dc=com` domain for the directory. You can change it to whatever is relevant to you.

Now let's run the `ldif` file with ldapmodify:

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f ~/setupDirectoryManager.ldif
```

We will use the new directory manager user to make changes to the LDAP directory after this step.

Creating the Structure of the Directory

Next, let's create the structure of the directory by creating parts of our tree.

```
tee ~/setupDirectoryStructure.ldif <<EOF
dn: dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: dcObject
objectclass: organization
o: MariaDB Support Team
dc: support

dn: cn=Manager,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: organizationalRole
cn: Manager
description: Directory Manager

dn: ou=People,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: organizationalUnit
ou: People

dn: ou=Groups,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Groups

dn: ou=System Users,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: organizationalUnit
ou: System Users
EOF
```

Now, let's use our new directory manager user and run the LDIF file with ldapmodify:

```
ldapmodify -a -x -D cn=Manager,dc=support,dc=mariadb,dc=com -W -f ~/setupDirectoryStructure.ldif
```

Creating the LDAP Users and Groups

Let's go ahead and create the LDAP users and groups that we are using for this scenario.

First, let's create the `foo` user:

```
tee ~/createFooUser.ldif <<EOF
dn: uid=foo,ou=People,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: foo
uid: foo
uidNumber: 16859
gidNumber: 100
homeDirectory: /home/foo
loginShell: /bin/bash
gecos: foo
userPassword: {crypt}x
shadowLastChange: -1
shadowMax: -1
shadowWarning: 0
EOF
ldapmodify -a -x -D cn=Manager,dc=support,dc=mariadb,dc=com -W -f ~/createFooUser.ldif
```

Next, let's create a couple of users to go into the `dba` group:

```
tee ~/createDbaUsers.ldif <<EOF
dn: uid=gmontee,ou=People,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: gmontee
uid: gmontee
uidNumber: 16860
gidNumber: 100
homeDirectory: /home/gmontee
loginShell: /bin/bash
gecos: gmontee
userPassword: {crypt}x
shadowLastChange: -1
shadowMax: -1
shadowWarning: 0

dn: uid=bstillman,ou=People,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: bstillman
uid: bstillman
uidNumber: 16861
gidNumber: 100
homeDirectory: /home/bstillman
loginShell: /bin/bash
gecos: bstillman
userPassword: {crypt}x
shadowLastChange: -1
shadowMax: -1
shadowWarning: 0
EOF
ldapmodify -a -x -D cn=Manager,dc=support,dc=mariadb,dc=com -W -f ~/createDbaUsers.ldif
```

Note that each of these users needs a password, so we can set it for each user with ldappasswd:

```
ldappasswd -x -D cn=Manager,dc=support,dc=mariadb,dc=com -W -S
uid=foo,ou=People,dc=support,dc=mariadb,dc=com
ldappasswd -x -D cn=Manager,dc=support,dc=mariadb,dc=com -W -S
uid=gmontee,ou=People,dc=support,dc=mariadb,dc=com
ldappasswd -x -D cn=Manager,dc=support,dc=mariadb,dc=com -W -S
uid=bstillman,ou=People,dc=support,dc=mariadb,dc=com
```

Next, let's create our `dba` group:

```
tee ~/createDbaGroup.ldif <<EOF
dn: cn=dba,ou=Groups,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: posixGroup
gidNumber: 678
EOF
ldapmodify -a -x -D cn=Manager,dc=support,dc=mariadb,dc=com -W -f ~/createDbaGroup.ldif
```

Next, let's add our two users to it:

```
tee ~/addUsersToDbaGroup.ldif <<EOF
dn: cn=dba,ou=Groups,dc=support,dc=mariadb,dc=com
changetype: modify
add: memberuid
memberuid: gmontee

dn: cn=dba,ou=Groups,dc=support,dc=mariadb,dc=com
changetype: modify
add: memberuid
memberuid: bstillman
EOF
ldapmodify -a -x -D cn=Manager,dc=support,dc=mariadb,dc=com -W -f ~/addUsersToDbaGroup.ldif
```

We also need to create LDAP users with the same name as the `bar` and `dba` MariaDB users. See here to read more about the reasons to do so. No one will be logging in as these users, so they do not need passwords. Instead of the `People` `organizationalUnit`, we create them in the `System Users` `organizationalUnit`.

```
tee ~/createSystemUsers.ldif <<EOF
dn: uid=bar,ou=System Users,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: bar
uid: bar
uidNumber: 16862
gidNumber: 100
homeDirectory: /home/bar
loginShell: /bin/bash
gecos: bar
userPassword: {crypt}x
shadowLastChange: -1
shadowMax: -1
shadowWarning: 0

dn: uid=dba,ou=System Users,dc=support,dc=mariadb,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: dba
uid: dba
uidNumber: 16863
gidNumber: 100
homeDirectory: /home/dba
loginShell: /bin/bash
gecos: dba
userPassword: {crypt}x
shadowLastChange: -1
shadowMax: -1
shadowWarning: 0
EOF
ldapmodify -a -x -D cn=Manager,dc=support,dc=mariadb,dc=com -W -f ~/createSystemUsers.ldif
```

Setting up the MariaDB Server

At this point, we can move on to setting up the MariaDB Server.

Installing LDAP and PAM Libraries

First, we need to make sure that the LDAP and PAM libraries are installed.

On RHEL, CentOS, and other similar Linux distributions that use RPM packages, we need to install the following packages:

```
sudo yum install openldap-clients nss-pam-ldapd pam pam-devel
```

Configuring LDAP

Next, let's configure LDAP on the system. We can use authconfig for this:

```
sudo authconfig --enableldap \
   --enableldapauth \
   --ldapserver="ldap://172.30.0.238:3306" \
   --ldapbasedn="dc=support,dc=mariadb,dc=com" \
   --enablemkhomedir \
   --update
```

Warning:

Be sure to replace `--ldapserver` and `--ldapbasedn` with values that are relevant for your environment.

Installing the pam_user_map PAM Module

CURRENT

The `pam_user_map` PAM module is included in the base install. No installation is needed.

< 10.5.2 / 10.4.13 / 10.3.23 / 10.2.32.7

Next, let's install the pam_user_map PAM module.

Before the module can be compiled from source, we may need to install some dependencies.

On RHEL, CentOS, and other similar Linux distributions that use RPM packages, we need to install `gcc` and `pam-devel`:

```
sudo yum install gcc pam-devel
```

On Debian, Ubuntu, and other similar Linux distributions that use DEB packages, we need to install `gcc` and `libpam0g-dev`:

```
sudo apt-get install gcc libpam0g-dev
```

And then we can build and install the library with the following:

```
wget https://raw.githubusercontent.com/MariaDB/server/10.4/plugin/auth_pam/mapper/pam_user_map.c
gcc pam_user_map.c -shared -lpam -fPIC -o pam_user_map.so
sudo install --mode=0755 pam_user_map.so /lib64/security/
```

Configuring the pam_user_map PAM Module

Next, let's configure the pam_user_map PAM module based on our hypothetical requirements.

The configuration file for the `pam_user_map` PAM module is `/etc/security/user_map.conf`. Based on our requirements, ours would look like:

```
foo: bar
@dba:dba
```

Installing the PAM Authentication Plugin

Next, let's install the pam authentication plugin.

Log into the MariaDB Server and execute the following:

```
INSTALL SONAME 'auth_pam';
```

Configuring the PAM Service

> **Note:**
>
> For modern Linux distributions (like RHEL 8 and newer) that use SSSD (System Security Services Daemon) to connect to an LDAP provider, the `pam_sss.so` module is the modern equivalent of `pam_ldap.so`. In such a configuration, `pam_sss.so` replaces `pam_ldap.so`. For more information, please see the Red Hat Enterprise Linux documentation.

Next, let's configure the PAM service. We will call our service `mariadb`, so our PAM service configuration file will be located at `/etc/pam.d/mariadb` on most systems.

Configuring PAM to Allow Only LDAP Authentication

Since we are only doing LDAP authentication with the pam_ldap PAM module and group mapping with the `pam_user_map` PAM module, our configuration file would look like this:

```
auth required pam_ldap.so
auth required pam_user_map.so
account required pam_ldap.so
```

Configuring PAM to Allow LDAP and Local Unix Authentication

If we want to allow authentication from LDAP users **and** from local Unix users through pam_unix, while giving priority to the local users, then we could do this instead:

```
auth [success=1 new_authtok_reqd=1 default=ignore] pam_unix.so audit
auth required pam_ldap.so try_first_pass
auth required pam_user_map.so
account sufficient pam_unix.so audit
account required pam_ldap.so
```

Configuring the pam_unix PAM Module

If you also want to allow authentication from local Unix users, the `pam_unix` PAM module adds some additional configuration steps on a lot of systems. We basically have to give the user that runs `mysqld` access to `/etc/shadow`.

If the `mysql` user is running `mysqld`, then we can do that by executing the following:

```
sudo groupadd shadow
sudo usermod -a -G shadow mysql
sudo chown root:shadow /etc/shadow
sudo chmod g+r /etc/shadow
```

The server needs to be restarted for this change to take affect.

Creating MariaDB Users

Next, let's create the MariaDB users. Remember that our PAM service is called `mariadb`.

First, let's create the MariaDB user for the user mapping: `foo: bar`

That means that we need to create a `bar` user:

```
CREATE USER 'bar'@'%' IDENTIFIED BY 'strongpassword';
GRANT ALL PRIVILEGES ON *.* TO 'bar'@'%' ;
```

And then let's create the MariaDB user for the group mapping: `@dba: dba`

That means that we need to create a `dba` user:

```
CREATE USER 'dba'@'%' IDENTIFIED BY 'strongpassword';
GRANT ALL PRIVILEGES ON *.* TO 'dba'@'%' ;
```

And then to allow for the user and group mapping, we need to create an anonymous user that authenticates with the pam authentication plugin that is also able to `PROXY` as the `bar` and `dba` users. Before we can create the proxy user, we might need to clean up some defaults:

```
DELETE FROM mysql.db WHERE User='' AND Host='%';
FLUSH PRIVILEGES;
```

And then let's create the anonymous proxy user:

```
CREATE USER ''@'%' IDENTIFIED VIA pam USING 'mariadb';
GRANT PROXY ON 'bar'@'%' TO ''@'%';
GRANT PROXY ON 'dba'@'%' TO ''@'%';
```

Testing our Configuration

Next, let's test our configuration by verifying that mapping is occurring. We can verify this by logging in as each of our users and comparing the return value of USER(), which is the original user name and the return value of CURRENT_USER(), which is the authenticated user name.

Testing LDAP Authentication

First, let's test our `foo` user:

```
$ mysql -u foo -h 172.30.0.198
[mariadb] Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 134
Server version: 10.3.10-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT USER(), CURRENT_USER();
+-----------------------------------------------+----------------+
| USER()                                        | CURRENT_USER() |
+-----------------------------------------------+----------------+
| foo@ip-172-30-0-198.us-west-2.compute.internal | bar@%          |
+-----------------------------------------------+----------------+
1 row in set (0.000 sec)
```

We can verify that our `foo` LDAP user was properly mapped to the `bar` MariaDB user by looking at the return value of CURRENT_USER().

Then let's test our `gmontee` user in the `dba` group:

```
$ mysql -u gmontee -h 172.30.0.198
[mariadb] Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 135
Server version: 10.3.10-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT USER(), CURRENT_USER();
+---------------------------------------------------+----------------+
| USER()                                            | CURRENT_USER() |
+---------------------------------------------------+----------------+
| gmontee@ip-172-30-0-198.us-west-2.compute.internal | dba@%          |
+---------------------------------------------------+----------------+
1 row in set (0.000 sec)
```

And then let's test our `bstillman` user in the `dba` group:

```
$ mysql -u bstillman -h 172.30.0.198
[mariadb] Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 136
Server version: 10.3.10-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT USER(), CURRENT_USER();
+-----------------------------------------------------+----------------+
| USER()                                              | CURRENT_USER() |
+-----------------------------------------------------+----------------+
| bstillman@ip-172-30-0-198.us-west-2.compute.internal | dba@%          |
+-----------------------------------------------------+----------------+
1 row in set (0.000 sec)
```

We can verify that our `gmontee` and `bstillman` LDAP users in the `dba` LDAP group were properly mapped to the `dba` MariaDB user by looking at the return values of `CURRENT_USER()`.

Testing Local Unix Authentication

If you chose the option that also allowed local Unix authentication, then let's test that out. Let's create a Unix user and give the user a password real quick:

```
sudo useradd alice
sudo passwd alice
```

And let's also map this user to `dba`:

```
@dba:dba
foo: bar
alice: dba
```

And we know that the existing anonymous user already has the `PROXY` privilege granted to the `dba` user, so this should just work without any other configuration. Let's test it out:

```
$ mysql -u alice -h 172.30.0.198
[mariadb] Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 141
Server version: 10.3.10-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT USER(), CURRENT_USER();
+-------------------------------------------------+----------------+
| USER()                                          | CURRENT_USER() |
+-------------------------------------------------+----------------+
| alice@ip-172-30-0-198.us-west-2.compute.internal | dba@%          |
+-------------------------------------------------+----------------+
1 row in set (0.000 sec)
```

We can verify that our `alice` Unix user was properly mapped to the `dba` MariaDB user by looking at the return values of CURRENT_USER().

Integrating with MariaDB MaxScale

If you are connecting to MariaDB Server through MariaDB MaxScale, it is also recommended to configure the proxy to authenticate users via MaxScale PAM Authenticator.

This page is licensed: CC BY-SA / Gnu FDL

Configuring PAM Authentication and User Mapping with Unix Authentication

We walk through the configuration of PAM authentication using the pam authentication plugin and user and group mapping with the pam_user_map PAM module. The primary authentication will be handled by the pam_unix PAM module, which performs standard Unix password authentication.

Hypothetical Requirements

In this walkthrough, we are going to assume the following hypothetical requirements:

- The Unix user `foo` should be mapped to the MariaDB user `bar`. (`foo: bar`)
- Any Unix user in the Unix group `dba` should be mapped to the MariaDB user `dba`. (`@dba: dba`)

Creating Our Unix Users and Groups

Let's go ahead and create the Unix users and groups that we are using for this hypothetical scenario.

First, let's create the `foo` user and a couple users to go into the `dba` group. Note that each of these users needs a password.

```
sudo useradd foo
sudo passwd foo
sudo useradd alice
sudo passwd alice
sudo useradd bob
sudo passwd bob
```

Next, let's create the `dba` group and add our two users to it:

```
sudo groupadd dba
sudo usermod -a -G dba alice
sudo usermod -a -G dba bob
```

We also need to create Unix users with the same name as the `bar` and `dba` MariaDB users. See here to read more about why. No one will be logging in as these users, so they do not need passwords.

```
sudo useradd bar
sudo useradd dba -g dba
```

Installing the pam_user_map PAM Module

Next, let's install the pam_user_map PAM module.

Before the module can be compiled from source, we may need to install some dependencies.

On RHEL, CentOS, and other similar Linux distributions that use RPM packages, we need to install `gcc` and `pam-devel`:

```
sudo yum install gcc pam-devel
```

On Debian, Ubuntu, and other similar Linux distributions that use DEB packages, we need to install `gcc` and `libpam0g-dev`:

```
sudo apt-get install gcc libpam0g-dev
```

And then we can build and install the library with the following:

```
wget https://raw.githubusercontent.com/MariaDB/server/10.4/plugin/auth_pam/mapper/pam_user_map.c
gcc pam_user_map.c -shared -lpam -fPIC -o pam_user_map.so
sudo install --mode=0755 pam_user_map.so /lib64/security/
```

Configuring the pam_user_map PAM Module

Next, let's configure the pam_user_map PAM module based on our hypothetical requirements.

The configuration file for the `pam_user_map` PAM module is `/etc/security/user_map.conf`. Based on our hypothetical requirements, ours would look like:

```
foo: bar
@dba:dba
```

Installing the PAM Authentication Plugin

Next, let's install the pam authentication plugin.

Log into the MariaDB Server and execute the following:

```
INSTALL SONAME 'auth_pam';
```

Configuring the PAM Service

Next, let's configure the PAM service. We will call our service `mariadb`, so our PAM service configuration file will be located at `/etc/pam.d/mariadb` on most systems.

Since we are only doing Unix authentication with the `pam_unix` PAM module and group mapping with the `pam_user_map` PAM module, our configuration file would look like this:

```
auth required pam_unix.so audit
auth required pam_user_map.so
account required pam_unix.so audit
```

Configuring the pam_unix PAM Module

The `pam_unix` PAM module adds some additional configuration steps on a lot of systems. We basically have to give the user that runs `mysqld` access to `/etc/shadow`.

If the `mysql` user is running `mysqld`, then we can do that by executing the following:

```
sudo groupadd shadow
sudo usermod -a -G shadow mysql
sudo chown root:shadow /etc/shadow
sudo chmod g+r /etc/shadow
```

The server needs to be restarted for this change to take affect.

Creating MariaDB Users

Next, let's create the MariaDB users. Remember that our PAM service is called `mariadb`.

First, let's create the MariaDB user for the user mapping: `foo: bar` . This means we need to create a `bar` user:

```
CREATE USER 'bar'@'%' IDENTIFIED BY 'strongpassword';
GRANT ALL PRIVILEGES ON *.* TO 'bar'@'%' ;
```

Next, let's create the MariaDB user for the group mapping: `@dba: dba` . This means that we need to create a `dba` user:

```
CREATE USER 'dba'@'%' IDENTIFIED BY 'strongpassword';
GRANT ALL PRIVILEGES ON *.* TO 'dba'@'%' ;
```

Next, to allow for the user and group mapping, we need to create an anonymous user that authenticates with the pam authentication plugin that is also able to `PROXY` as the `bar` and `dba` users. Before we can create the proxy user, we might need to clean up some defaults:

```
DELETE FROM mysql.db WHERE User='' AND Host='%';
FLUSH PRIVILEGES;
```

Finally, let's create the anonymous proxy user:

```
CREATE USER ''@'%' IDENTIFIED VIA pam USING 'mariadb';
GRANT PROXY ON 'bar'@'%' TO ''@'%';
GRANT PROXY ON 'dba'@'%' TO ''@'%';
```

Testing our Configuration

Next, let's test out our configuration by verifying that mapping is occurring. We can verify this by logging in as each of our users and comparing the return value of USER(), which is the original user name and the return value of CURRENT_USER(), which is the authenticated user name.

First, let's test our `foo` user:

```
$ mysql -u foo -h 172.30.0.198
[mariadb] Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 22
Server version: 10.3.10-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT USER(), CURRENT_USER();
+-----------------------------------------------+----------------+
| USER()                                        | CURRENT_USER() |
+-----------------------------------------------+----------------+
| foo@ip-172-30-0-198.us-west-2.compute.internal | bar@%          |
+-----------------------------------------------+----------------+
1 row in set (0.000 sec)
```

We can verify that our `foo` Unix user was properly mapped to the `bar` MariaDB user by looking at the return value of `CURRENT_USER()`.

Next, let's test our `alice` user in the `dba` group:

```
$ mysql -u alice -h 172.30.0.198
[mariadb] Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 19
Server version: 10.3.10-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT USER(), CURRENT_USER();
+-------------------------------------------------+----------------+
| USER()                                          | CURRENT_USER() |
+-------------------------------------------------+----------------+
| alice@ip-172-30-0-198.us-west-2.compute.internal | dba@%          |
+-------------------------------------------------+----------------+
1 row in set (0.000 sec)
```

Finally, let's test our `bob` user in the `dba` group:

```
$ mysql -u bob -h 172.30.0.198
[mariadb] Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 20
Server version: 10.3.10-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT USER(), CURRENT_USER();
+-----------------------------------------------+----------------+
| USER()                                        | CURRENT_USER() |
+-----------------------------------------------+----------------+
| bob@ip-172-30-0-198.us-west-2.compute.internal | dba@%          |
+-----------------------------------------------+----------------+
1 row in set (0.000 sec)
```

We can verify that our `alice` and `bob` Unix users in the `dba` Unix group were properly mapped to the `dba` MariaDB user by looking at the return values of `CURRENT_USER()`.

This page is licensed: CC BY-SA / Gnu FDL

User and Group Mapping with PAM

Overview

Even when using the pam authentication plugin, the authenticating PAM user account still needs to exist in MariaDB, and the account needs to have privileges in the database. Creating these MariaDB accounts and making sure the privileges are correct can be a lot of work. To decrease the amount of work involved, some users would like to be able to map a PAM user to a different MariaDB user. For example, let's say that `alice` and `bob` are both DBAs. It would be nice if each of them could log into MariaDB with their own PAM username and password, while MariaDB sees both of them as the same `dba` user. That way, there is only one MariaDB account to keep track of.

Although most PAM modules usually do not do things like this, PAM supports the ability to change the user name in the process of authentication.The MariaDB `pam` authentication plugin fully supports this feature of PAM.

The pam_user_map PAM Module

Rather than building user and group mapping into the `pam` authentication plugin, MariaDB thought that it would cover the most use cases and offer the most flexibility to offload this functionality to an external PAM module. The `pam_user_map` PAM module was implemented by MariaDB to facilitate this. This PAM module can be configured in the PAM service used by the `pam` authentication plugin, just like other PAM modules.

Lack of Support for MySQL/Percona Group Mapping Syntax

Unlike MariaDB, MySQL and Percona implemented group mapping in their PAM authentication plugins. If you've read through MySQL's PAM authentication documentation on group mapping or Percona's PAM authentication documentation on group mapping, you've probably seen syntax where the group mappings are provided in the CREATE USER statement like this:

```
CREATE USER ''@''
  IDENTIFIED WITH authentication_pam
  AS 'mysql, root=developer, users=data_entry';
```

Since MariaDB's user and group mapping is performed by an external PAM module, MariaDB's `pam` authentication plugin does not support this syntax. Instead, the user and group mappings for the `pam_user_map` PAM module are configured in an external configuration file. This is discussed in a later section.

Installing the pam_user_map PAM Module

The `pam_user_map` PAM module gets installed as part of all our MariaDB server packages.

Some Linux distributions have not picked up this change in their own packages yet. When using such an installation, it may be necessary to compile the PAM module from source as described in the next section, or to manually extract it from one of our server packages and copy it to the target system.

Installing the pam_user_map PAM Module from Source

Installing Compilation Dependencies

Before the module can be compiled from source, you may need to install some dependencies.

On RHEL, CentOS, and other similar Linux distributions that use RPM packages, you need to install `gcc`, `pam-devel` and `MariaDB-devel`:

```
sudo yum install gcc pam-devel MariaDB-devel
```

On Debian, Ubuntu, and other similar Linux distributions that use DEB packages, you need to install `gcc`, `libpam0g-dev`:

```
sudo apt-get install gcc libpam0g-dev libmariadb-dev
```

Compiling and Installing the pam_user_map PAM Module

The `pam_user_map` PAM module can be built by downloading `plugin/auth_pam/mapper/pam_user_map.c` file from the MariaDB source tree and compiling it after minor adjustments. Once it is built, it can be installed to the system's PAM module directory, which is typically `/lib64/security/`:

```
wget https://raw.githubusercontent.com/MariaDB/server/10.4/plugin/auth_pam/mapper/pam_user_map.c
sed -ie 's/config_auth_pam/plugin_auth_common/' pam_user_map.c
gcc -I/usr/include/mysql/ pam_user_map.c -shared -lpam -fPIC -o pam_user_map.so
sudo install --mode=0755 pam_user_map.so /lib64/security/
```

You also need to adjust the major version number in the URL on the first line to match your installed MariaDB version, and the `#-I` include path argument on the `gcc` line, as depending on operating system and MariaDB server version, the `plugin_auth_common.h` file may be installed in a directory other than `/usr/include/mysql/` .

Configuring the pam_user_map PAM Module

The `pam_user_map` PAM module uses the configuration file at the path `/etc/security/user_map.conf` to determine its user and group mappings. The file's format is described below.

To map a specific PAM user to a specific MariaDB user:

```
orig_pam_user_name: mapped_mariadb_user_name
```

Or to map any PAM user in a specific PAM group to a specific MariaDB user, the group name is prefixed with `@`:

```
@orig_pam_group_name: mapped_mariadb_user_name
```

For example, here is an example `/etc/security/user_map.conf`:

```
============================================================
#comments and empty lines are ignored
john: jack
bob:  admin
top:  accounting
@group_ro: readonly
```

Configuring PAM

With user and group mapping, configuring PAM is done similar to how it is normally done with the pam authentication plugin. However, when configuring the PAM service, you will have to add an `auth` line for the `pam_user_map` PAM module to the service's PAM configuration file:

```
auth required pam_unix.so audit
auth required pam_user_map.so
account required pam_unix.so audit
```

Creating Users

With user and group mapping, creating users is done similar to how it is normally done with the pam authentication plugin. However, one major difference is that you will need to GRANT the PROXY privilege on the mapped user to the original user.

Consider having the following configured in `/etc/security/user_map.conf`:

```
foo: bar
@dba:dba
```

Then you could execute the following to grant the relevant privileges:

```
CREATE USER 'bar'@'%' IDENTIFIED BY 'strongpassword';
GRANT ALL PRIVILEGES ON *.* TO 'bar'@'%' ;

CREATE USER 'dba'@'%' IDENTIFIED BY 'strongpassword';
GRANT ALL PRIVILEGES ON *.* TO 'dba'@'%' ;

CREATE USER ''@'%' IDENTIFIED VIA pam USING 'mariadb';
GRANT PROXY ON 'bar'@'%' TO ''@'%';
GRANT PROXY ON 'dba'@'%' TO ''@'%';
```

Note that the `''@'%'` account is a special catch-all anonymous account. Any login by a user that has no more specific account match in the system will be matched by this anonymous account.

Also note that you might not be able to create the `''@'%'` anonymous account by default on some systems without doing some extra steps first. See Fixing a Legacy Default Anonymous Account for more information.

Verifying that Mapping is Occurring

In case any user mapping is performed, the original user name is returned by the SQL function USER(), while the authenticated user name is returned by the SQL function CURRENT_USER(). The latter actually defines what privileges are available to a connected user.

Consider having the following configured:

```
foo: bar
```

Then the following output would verify that it is working properly:

```
$ mysql -u foo -h 172.30.0.198
[mariadb] Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 22
Server version: 10.3.10-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT USER(), CURRENT_USER();
+-----------------------------------------------+----------------+
| USER()                                        | CURRENT_USER() |
+-----------------------------------------------+----------------+
| foo@ip-172-30-0-198.us-west-2.compute.internal | bar@%          |
+-----------------------------------------------+----------------+
1 row in set (0.000 sec)
```

We can verify that our `foo` PAM user was properly mapped to the `bar` MariaDB user by looking at the return value of `CURRENT_USER()`.

Logging

By default, the `pam_user_map` PAM module does not perform any logging. However, if you want to enable debug logging, then you can add the `debug` module argument to the service's PAM configuration file:

```
auth required pam_unix.so audit
auth required pam_user_map.so debug
account required pam_unix.so audit
```

When debug logging is enabled, the `pam_user_map` PAM module will write log entries to the same syslog location as other PAM modules, which is typically `/var/log/secure` on many systems.

For example, this debug log output can look like the following:

```
Jan  9 05:42:13 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Opening file
'/etc/security/user_map.conf'.
Jan  9 05:42:13 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Incoming username 'alice'.
Jan  9 05:42:13 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): User belongs to 2 groups [alice,dba].
Jan  9 05:42:13 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Check if user is in group 'dba': YES
Jan  9 05:42:13 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): User mapped as 'dba'
Jan  9 05:43:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Opening file
'/etc/security/user_map.conf'.
Jan  9 05:43:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Incoming username 'bob'.
Jan  9 05:43:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): User belongs to 2 groups [bob,dba].
Jan  9 05:43:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Check if user is in group 'dba': YES
Jan  9 05:43:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): User mapped as 'dba'
Jan  9 06:08:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Opening file
'/etc/security/user_map.conf'.
Jan  9 06:08:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Incoming username 'foo'.
Jan  9 06:08:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): User belongs to 1 group [foo].
Jan  9 06:08:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Check if user is in group 'dba': NO
Jan  9 06:08:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): Check if username 'foo': YES
Jan  9 06:08:36 ip-172-30-0-198 mysqld: pam_user_map(mariadb:auth): User mapped as 'bar'
```

Known Issues

PAM User with Same Name as Mapped MariaDB User Must Exist

With user and group mapping, any PAM user or any PAM user in a given PAM group can be mapped to a specific MariaDB user account. However, due to the way PAM works, a PAM user with the same name as the mapped MariaDB user account must exist.

Consider the configuration file for the PAM service file containing the following:

```
auth required pam_sss.so
auth required pam_user_map.so debug
account sufficient pam_unix.so
account sufficient pam_sss.so
```

Consider `etc/security/user_map.conf` containing the following:

```
@dba: dba
```

In that case, any PAM user in the PAM group `dba` are mapped to the MariaDB user account `dba`. But if a PAM user with the name `dba` doesn't exist, the `pam_user_map` PAM module's debug logging writes errors to the syslog, like the following:

```
Sep 27 17:17:05 dbserver1 mysqld: pam_user_map(mysql:auth): Opening file '/etc/security/user_map.conf'.
Sep 27 17:17:05 dbserver1 mysqld: pam_user_map(mysql:auth): Incoming username 'alice'.
Sep 27 17:17:05 dbserver1 mysqld: pam_user_map(mysql:auth): User belongs to 4 groups
[dba,mongod,mongodba,mysql].
Sep 27 17:17:05 dbserver1 mysqld: pam_user_map(mysql:auth): Check if user is in group 'mysql': YES
Sep 27 17:17:05 dbserver1 mysqld: pam_user_map(mysql:auth): User mapped as 'dba'
Sep 27 17:17:05 dbserver1 mysqld: pam_unix(mysql:account): could not identify user (from getpwnam(dba))
Sep 27 17:17:05 dbserver1 mysqld: pam_sss(mysql:account): Access denied for user dba: 10 (User not known
to the underlying authentication module)
Sep 27 17:17:05 dbserver1 mysqld: 2018-09-27 17:17:05 72 [Warning] Access denied for user
'alice'@'localhost' (using password: NO)
```

In the above log snippet, notice that both the `pam_unix` and the `pam_sss` PAM modules are complaining that the `dba` PAM user does not appear to exist, and that these complaints cause the PAM authentication process to fail, which causes the MariaDB authentication process to fail as well.

This can be fixed by creating a PAM user with the same name as the mapped MariaDB user account, which is `dba` in this case.

You may also be able to work around this problem by essentially disabling PAM's account verification for the service with the pam_permit PAM module. For example, in the above case, that would be:

```
auth required pam_sss.so
auth required pam_user_map.so debug
account required pam_permit.so
```

See MDEV-17315 for more information.

Tutorials

You may find the following PAM and user mapping-related tutorials helpful:

- Configuring PAM Authentication and User Mapping with Unix Authentication
- Configuring PAM Authentication and User Mapping with LDAP Authentication

See Also

- Configuring PAM Authentication and User Mapping with MariaDB
- Configuring PAM Group Mapping with MariaDB
- Configuring LDAP Authentication and Group Mapping With MariaDB

This page is licensed: CC BY-SA / Gnu FDL

Authentication with JWT, OIDC, and OAuth 2.0 via PAM

> **Important:**
>
> This draft, intended for technical review under DOCS-5715, requires validation for technical accuracy. It includes generated text and code blocks and needs evaluation by a subject matter expert (SME) in this format. Please refrain from relying on any information on the page.

MariaDB Server supports modern authentication standards—including JSON Web Tokens (JWT), OpenID Connect (OIDC), and OAuth 2.0—through its pluggable authentication architecture.

By leveraging the operating system's PAM (Pluggable Authentication Modules) subsystem, MariaDB can integrate with external Identity Providers (IdPs) like AWS Cognito, Google Cloud Identity, and Azure AD without requiring proprietary database plugins. The database delegates the validation of tokens to the underlying OS security layer.

Prerequisites

Before proceeding, ensure the following components are available:

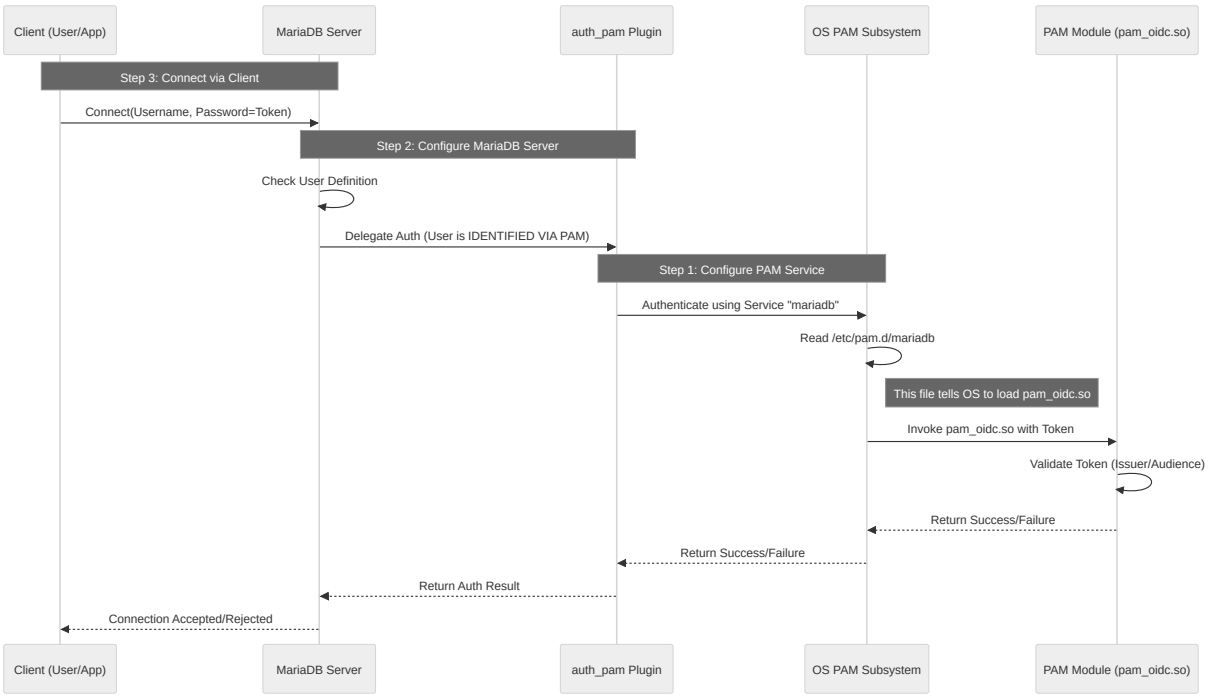
- MariaDB Server (Active and running).
- PAM Module: You must install a third-party PAM module corresponding to your desired authentication method (e.g., `pam_oidc.so`, `pam_oauth2.so`, or `pam_jwt.so`).
  - *Note: These are system-level libraries installed into `/lib/security` or `/lib64/security`. Ensure the module you choose is actively maintained and compatible with your OS.*
- Identity Provider (IdP): You must have your `Issuer URL`, `Client ID`, and `Audience` values ready from your provider (e.g., Google, Okta, Azure).

Configuration and Connection Workflow

How to bridge the operating system's authentication layer with MariaDB user management.

How Authentication Works

Before configuring the system, it is helpful to understand the handshake between the Client, the Database, and the Operating System.



Step 1

Configure the PAM Service

You must define a PAM service that loads your chosen module. This file tells the operating system how to validate the credentials passed by MariaDB.

File: `/etc/pam.d/mariadb`

The specific flags below (like `issuer` or `aud`) are standard OIDC parameters. Consult your specific PAM module's documentation for exact flag syntax.

```
# /etc/pam.d/mariadb
# Generic OIDC Configuration Example

# 1. 'auth' checks the validity of the token
auth required pam_oidc.so issuer=https://accounts.google.com aud=YOUR_CLIENT_ID

# 2. 'account' checks if the user is allowed to login (often required by PAM)
account required pam_oidc.so
```

Important:

Ensure strict validation of the `aud` (Audience) parameter to confirm that tokens are issued exclusively for your database application, preventing the use of tokens intended for different applications.

Step 2

Configure MariaDB Server

Once the OS is configured, you must enable the PAM plugin in MariaDB and create a user that utilizes it.

1. Log in to MariaDB as root.
2. Install the PAM Plugin:

   ```
   INSTALL SONAME 'auth_pam';
   ```

3. Create the User: The `USING` clause specifies the service name (filename) in `/etc/pam.d/`.

   ```
   -- Create a user that maps to the 'sub' (Subject) claim in the token
   CREATE USER 'jdoe'@'%' IDENTIFIED VIA pam USING 'mariadb';
   ```

Step 3

Connect via Client

When connecting, the "Password" field is used to transmit the JWT or OIDC Token.

```
# 1. Obtain your token (e.g., via gcloud or your IdP's CLI)
export DB_TOKEN="eyJhbGciOiJSUzI1NiIsImtpZ..."

# 2. Connect to MariaDB, passing the token as the password
mariadb --user="jdoe" --password="$DB_TOKEN"
```

Troubleshooting

- Authentication Failure: Check the system authentication logs.

  ```
  sudo tail -f /var/log/auth.log
  ```

  Look for errors regarding "invalid audience," "expired token," or "issuer mismatch."

- SELinux/AppArmor: If the database cannot access the PAM configuration, ensure your security context allows `mysqld` to read from `/etc/pam.d/`.
- Username Mapping: By default, the MariaDB username must match the identity in the token (often the `sub` or `email` claim). Some PAM modules allow you to map these (e.g., mapping `user@example.com` to `user`); check your module's `user_template` or mapping documentation.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Replication & Cluster Plugins

See WSREP_INFO Plugin.

The WSREP_INFO plugin adds the WSREP_MEMBERSHIP and WSREP_STATUS tables to the Information Schema, providing detailed insights into Galera Cluster membership and status.

See wsrep_provider.

The wsrep_provider plugin exposes Galera Cluster provider options as individual system variables, allowing for easier configuration and validation of cluster settings.

This page is licensed: CC BY-SA / Gnu FDL

WSREP_INFO Plugin

The `WSREP_INFO` plugin library contains the following plugins:

- `WSREP_MEMBERSHIP`
- `WSREP_STATUS`

The `WSREP_MEMBERSHIP` plugin creates the WSREP_MEMBERSHIP table in the INFORMATION_SCHEMA database. The plugin also adds the SHOW WSREP_MEMBERSHIP statement.

The `WSREP_STATUS` plugin creates the WSREP_STATUS table in the INFORMATION_SCHEMA database. The plugin also adds the SHOW WSREP_STATUS statement.

These tables and statements provide information about Galera. Only users with the SUPER privilege can access this information.

Installing the Plugin

iAlthough the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'wsrep_info';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = wsrep_info
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'wsrep_info';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Example

```
SHOW TABLES FROM information_schema LIKE 'WSREP%';
+---------------------------------------+
| Tables_in_information_schema (WSREP%) |
+---------------------------------------+
| WSREP_STATUS                          |
| WSREP_MEMBERSHIP                      |
+---------------------------------------+
```

Options

`wsrep_membership`

- Description: Controls how the server treats the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--wsrep-membership=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

`wsrep_status`

- Description: Controls how the server treats the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--wsrep-status=value`
- Data Type: `enumerated`
- Default Value: `ON`

- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

wsrep_provider

This plugin is for Galera Cluster. It splits up the `wsrep_provider_options` setting into individual configuration variables.

> **Note:**
>
> The plugin is available from MariaDB 11.4, and built in to the server, but not enabled by default.

Without that plugin, options are grouped together, like this:

```
wsrep_provider_options="base_dir = /var/lib/mysql/; base_host = node-1;..."
```

With this plugin loaded, you can configure individual variables, like this:

```
wsrep_provider_base_dir  = /var/lib/mysql/
wsrep_provider_base_host = node-1
...
```

This makes managing provider options easier, and helps avoid the problem of wsrep_provider_options exceeding the maximum length of 2048 characters for an individual variable.

To enable the plugin, add the following line to the `[mariadbd]`, `[server]`, or `[galera]` sections of your server option file:

```
plugin-wsrep-provider=ON
```

Alternatively, start the server with the `--plugin-wsrep-provider` option.

See the wsrep_provider_options page for what you can configure for Galera Cluster.

For plugin version and maturity level, see this page.

This page is licensed: CC BY-SA / Gnu FDL

Other Plugins

See Disks Plugin.

The Disks plugin adds the DISKS table to the Information Schema, providing metadata about the system's disk storage and usage.

See Feedback Plugin.

The Feedback plugin collects and sends anonymous server usage and configuration data to MariaDB to help improve the software.

See inet4.

The inet4 plugin provides the INET4 data type, allowing for efficient native storage and manipulation of IPv4 addresses as 4-byte binary strings.

See METADATA_LOCK_INFO Plugin.

This plugin creates the METADATA_LOCK_INFO table in the Information Schema, allowing users to view active metadata locks and their owners.

See MYSQL_JSON.

The MYSQL_JSON plugin provides a JSON data type alias for compatibility, ensuring that tables created with the MySQL JSON type can be read by MariaDB.

See mhnsw.

The mhnsw plugin implements the Hierarchical Navigable Small World algorithm, enabling high-performance approximate nearest neighbor search for vector data.

See online_alter_log.

The online_alter_log plugin provides logging capabilities for online ALTER TABLE operations, helping administrators monitor and debug schema changes.

See Query Cache Information Plugin.

This plugin exposes the contents of the query cache via the QUERY_CACHE_INFO table in the Information Schema, aiding in performance analysis.

See Query Response Time Plugin.

The Query Response Time plugin collects and displays the distribution of query execution times, helping to identify performance bottlenecks.

See User Variables Plugin.

The User Variables plugin adds the USER_VARIABLES table to the Information Schema, allowing users to inspect defined user variables and their values.

This page is licensed: CC BY-SA / Gnu FDL

Disks Plugin

The `DISKS` plugin creates the DISKS table in the INFORMATION_SCHEMA database. This table shows metadata about disks on the system, enabling monitoring of the disk space status. Accessing the `INFORMATION_SCHEMA.DISKS` table requires the FILE privilege.

> **Note:**
>
> The plugin is supported on Linux systems only.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB automatically. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'disks';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be provided as a command-line argument to mysqld or specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = disks
```

Example

After the plugin is installed, the `INFORMATION_SCHEMA.DISKS` table displays disk utilization information.

To view disk usage information, run the following query:

```
SELECT * FROM information_schema.DISKS;
```

Example Output

```
+-----------+----------------------+-----------+----------+-----------+
| Disk      | Path                 | Total     | Used     | Available |
+-----------+----------------------+-----------+----------+-----------+
| /dev/sda3 | /                    |  47929956 | 30666304 |  14805864 |
| /dev/sda1 | /boot/efi            |    191551 |     3461 |    188090 |
| /dev/sda4 | /home                | 174679768 | 80335392 |  85448120 |
| /dev/sdb1 | /mnt/hdd             | 961301832 |    83764 | 912363644 |
| /dev/sdb1 | /home/wikman/Music   | 961301832 |    83764 | 912363644 |
| /dev/sdb1 | /home/wikman/Videos  | 961301832 |    83764 | 912363644 |
| /dev/sdb1 | /home/wikman/hdd     | 961301832 |    83764 | 912363644 |
| /dev/sdb1 | /home/wikman/Pictures| 961301832 |    83764 | 912363644 |
| /dev/sda3 | /var/lib/docker/aufs |  47929956 | 30666304 |  14805864 |
+-----------+----------------------+-----------+----------+-----------+
9 rows in set (0.00 sec)
```

Column Descriptions

The `INFORMATION_SCHEMA.DISKS` table contains the following columns:

- **Disk**: The name of the disk device or partition.
- **Path**: The mount point where the disk is mounted.
- **Total**: The total disk space available on the disk (in KiB).
- **Used**: The amount of disk space currently used (in KiB).
- **Available**: The amount of disk space available to non-root users (in KiB).

> Note: As root users may have more available space than non-root users, `Available` + `Used` may be less than `Total`.
>
> Additionally, all paths to which a specific disk is mounted are reported, so a disk may appear multiple times in the output.

Alternative: Mounts Table

As an alternative option, the plugin includes the `INFORMATION_SCHEMA.MOUNTS` table, which contains mount points and their respective disks.

For example:

```
SELECT * FROM information_schema.MOUNTS;
```

Expected Output

```
+----------------------+-----------+
| Path                 | Disk      |
+----------------------+-----------+
| /                    | /dev/sda3 |
| /boot/efi            | /dev/sda1 |
| /home                | /dev/sda4 |
| /mnt/hdd             | /dev/sdb1 |
| /home/wikman/Music   | /dev/sdb1 |
+----------------------+-----------+
```

Versions

Version	Status	Introduced
1.1	Stable	MariaDB 10.4.7, MariaDB 10.3.17, MariaDB 10.2.26, MariaDB 10.1.41
1.0	Beta	MariaDB 10.3.6, MariaDB 10.2.14, MariaDB 10.1.32

Options

`disks`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--disks=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'disks';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

See Also

- Information Schema
- Plugin Overview

This page is licensed: CC BY-SA / Gnu FDL

Feedback Plugin

The `feedback` plugin is designed to collect and, optionally, upload configuration and usage information to MariaDB.org or to any other configured URL.

See the MariaDB User Feedback page on MariaDB.org to see collected MariaDB usage statistics.

MariaDB is usually distributed with this plugin included, but it is not enabled by default.
On Windows, this plugin is part of the server and has a special checkbox in the installer window. Either way, you need to explicitly install and enable it in order for feedback data to be sent.

Verifying the Plugin's Status

To verify whether the `feedback` plugin is installed and enabled, execute the SHOW PLUGINS statement or query the information_schema.plugins table:

```
SELECT plugin_status FROM information_schema.plugins
  WHERE plugin_name = 'feedback';
+---------------+
| plugin_status |
+---------------+
| DISABLED      |
+---------------+
```

If that `SELECT` returns no rows, then you still need to install the plugin.

When the plugin is installed and enabled, you will see:

```
SELECT plugin_status FROM information_schema.plugins
  WHERE plugin_name = 'feedback';
+---------------+
| plugin_status |
+---------------+
| ACTIVE        |
+---------------+
```

Installing the Plugin

In some releases, the plugin's shared library is distributed with MariaDB by default, but the plugin is not actually installed by MariaDB. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'feedback';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = feedback
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'feedback';
```

If you installed the plugin by providing the `--plugin-load` or the `--plugin-load-add` options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Enabling the Plugin

You can enable the plugin by setting the `feedback` option to `ON` in a relevant server option group in an option file:

```
[mariadb]
...
feedback=ON
```

In Windows, the plugin can also be enabled during a new MSI installation. The MSI GUI installation provides the "Enable feedback plugin" checkbox to enable the plugin. The MSI command-line installation provides the `FEEDBACK=1` command-line option to enable the plugin.

See the next section for how to verify the plugin is installed and active and (if needed) install the plugin.

Collecting Data

The `feedback` plugin collects the following data:

- Certain rows from SHOW STATUS and SHOW VARIABLES.
- All installed plugins and their versions.
- System information such as CPU count, memory, architecture, and OS/linux distribution.
- The feedback_server_uid, which is a SHA1 hash of the MAC address of the first network interface and the TCP port that the server listens on.

The `feedback` plugin creates the FEEDBACK table in the INFORMATION_SCHEMA database. To see the data that has been collected by the plugin, you can execute:

```
SELECT * FROM information_schema.feedback;
```

Only the contents of this table are sent to the feedback_url.

MariaDB stores collation usage statistics. Each collation that has been used by the server will have a record in output of `SELECT * FROM information_schema.feedback` , for example:

```
+---------------------------------------+---------------------+
| VARIABLE_NAME                         | VARIABLE_VALUE      |
+---------------------------------------+---------------------+
| Collation used utf8_unicode_ci        | 10                  |
| Collation used latin1_general_ci      | 20                  |
+---------------------------------------+---------------------+
```

Collations that have not been used will not be included in the result.

Sending Data

The `feedback` plugin sends the data using a `POST` request to any URL or a list of URLs that you specify by setting the feedback_url system variable. By default, this is set to the following URL:

- https://feedback.mariadb.org/rest/v1/post

Both HTTP and HTTPS protocols are supported.

If HTTP traffic requires a proxy in your environment, then you can specify the proxy by setting the feedback_http_proxy system variable.

If the feedback_url system variable is not set to an empty string, then the plugin will automatically send a report to all URLs in the list a few minutes after the server starts up and then once a week after that.

If the feedback_url system variable is set to an empty string, then the plugin will **not** automatically send any data. This may be necessary if outbound HTTP communication from your database server is not permitted. In this case, you can still upload the data manually, if you'd like.

First, generate the report file with the MariaDB command-line mariadb client:

```
$ mariadb -e 'select * from information_schema.feedback' > report.txt
```

Then, you can upload the generated `report.txt` here from the command line with tools such as curl:

```
$ curl -F data=@report.txt https://feedback.mariadb.org/rest/v1/post
```

Manual uploading allows you to be absolutely sure that we receive only the data shown in the INFORMATION_SCHEMA.FEEDBACK table and that no private or sensitive information is being sent.

System Variables

`feedback_http_proxy`

- Description: Proxy server for use when http calls cannot be made, such as in a firewall environment. The format is `host:port`.
- Command line: `--feedback-http=proxy=value`
- Read-only: Yes
- Data Type: string
- Default Value: `''` (empty)

`feedback_send_retry_wait`

- Description: Time in seconds before retrying if the plugin failed to send the data for any reason.
- Command line: `--feedback-send-retry-wait=#`
- Scope: Global
- Dynamic: Yes
- Data Type: numeric
- Default Value: `60`
- Valid Values: `1` to `86400`

`feedback_send_timeout`

- Description: An attempt to send the data times out and fails after this many seconds.
- Command line: `--feedback-send-timeout=#`
- Scope: Global
- Dynamic: Yes
- Data Type: numeric
- Default Value: `60`
- Valid Values: `1` to `86400`

`feedback_server_uid`

- Description: Automatically calculated server unique id hash.
- Scope: Global
- Dynamic: No
- Data Type: string

`feedback_url`

- Description: URL to which the data is sent. More than one URL, separated by spaces, can be specified. Set it to an empty string to disable data sending.
- Command line: `--feedback-url=url`
- Scope: Global
- Dynamic: No
- Data Type: string

- Default Value: https://feedback.mariadb.org/rest/v1/post

`feedback_user_info`

- Description: The value of this option is not used by the plugin, but it is included in the feedback data. It can be used to add any user-specified string to the report. This could be used to help to identify it. For example, a support contract number, or a computer name (if you collect reports internally by specifying your own `feedback-url`).
- Command line: `--feedback-user-info=string`
- Scope: Global
- Dynamic: No
- Data Type: string
- Default Value: Empty string

Options

`feedback`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--feedback=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

inet4

This plugin implements the inet4 data type and functions used by this data type, like conversions.

> **Note:**
>
> For plugin version and maturity level, see this page.

This plugin is built in to MariaDB Server. It cannot be disabled.

`inet4` is a data type plugin. It can be used in stored procedures as a variable type and as a data type for table columns.

This page is licensed: CC BY-SA / Gnu FDL

METADATA_LOCK_INFO Plugin

The `METADATA_LOCK_INFO` plugin creates the METADATA_LOCK_INFO table in the INFORMATION_SCHEMA database. This table shows active metadata locks. The table is empty if there are no active metadata locks.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'metadata_lock_info';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = metadata_lock_info
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'metadata_lock_info';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Examples

Viewing all Metadata Locks

```
SELECT * FROM information_schema.metadata_lock_info;
+-----------+-------------------------+---------------+---------------------+-----------------+--------
-----+
| THREAD_ID | LOCK_MODE               | LOCK_DURATION | LOCK_TYPE           | TABLE_SCHEMA    |
TABLE_NAME  |
+-----------+-------------------------+---------------+---------------------+-----------------+--------
-----+
|        31 | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT  | Global read lock    |                 |
|
|        31 | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT  | Commit lock         |                 |
|
|        31 | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT  | Schema metadata lock | dbname          |
|
|        31 | MDL_SHARED_NO_READ_WRITE | MDL_EXPLICIT  | Table metadata lock  | dbname          | exotics
|
+-----------+-------------------------+---------------+---------------------+-----------------+--------
-----+
4 rows in set (0.00 sec)
```

Matching Metadata Locks with Threads and Queries

```
SELECT
CONCAT('Thread ',P.ID,' executing "',P.INFO,'" IS LOCKED BY Thread ',
M.THREAD_ID) WhoLocksWho
FROM INFORMATION_SCHEMA.PROCESSLIST P,
INFORMATION_SCHEMA.METADATA_LOCK_INFO M
WHERE LOCATE(lcase(LOCK_TYPE), lcase(STATE))>0;
+-------------------------------------------------------------------------------------+
| WhoLocksWho                                                                         |
+-------------------------------------------------------------------------------------+
| Thread 3 executing "INSERT INTO foo ( b ) VALUES ( 'FOO' )" IS LOCKED BY Thread 2 |
+-------------------------------------------------------------------------------------+
1 row in set (0.00 sec)

SHOW PROCESSLIST;
+----+------+-----------+------+---------+------+-----------------------------+-----------------------
---------------+----------+
| Id | User | Host      | db   | Command | Time | State                       | Info
| Progress |
+----+------+-----------+------+---------+------+-----------------------------+-----------------------
---------------+----------+
|  2 | root | localhost | test | Sleep   |  123 |                             | NULL
|    0.000 |
|  3 | root | localhost | test | Query   |  103 | Waiting for global read lock | INSERT INTO foo ( b )
VALUES ( 'FOO' ) |    0.000 |
|  4 | root | localhost | test | Query   |    0 | init                        | SHOW PROCESSLIST
|    0.000 |
+----+------+-----------+------+---------+------+-----------------------------+-----------------------
---------------+----------+
3 rows in set (0.00 sec)
```

Options

`metadata_lock_info`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--metadata-lock-info=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

MYSQL_JSON

> **Note:**
>
> The `TYPE_MYSQL_JSON` plugin is available from MariaDB 10.5.7.

The JSON type in MySQL stores the JSON object in its own native form, while, in MariaDB, the JSON type is a LONGTEXT. Opening a table with a JSON type created in MySQL results in an error:

```
SELECT * FROM mysql_json_table;
ERROR 4161 (HY000): Unknown data type: 'MYSQL_JSON'
```

The `mysql_json` plugin is used to make it easier to upgrade to MariaDB.

Installing

Installing can be done in a number of ways, for example:

```
INSTALL SONAME 'type_mysql_json';
```

See Making MariaDB understand MySQL JSON for a full description.

This page is licensed: CC BY-SA / Gnu FDL

mhnsw

> **Note:**
>
> This plugin is for internal use only.

This plugin implements the `nhnsw` vector index algorithm. It is used to create vector indexes.

See Vector Overview for the functionality, and Vector System Variables for what you can configure.

It is built in the server, and is always enabled.

For plugin version and maturity level, see this page.

This page is licensed: CC BY-SA / Gnu FDL

online_alter_log

> **Note:**
>
> This plugin is for internal use only.

This plugin represents the online alter log in a transaction. It is used to commit transactions for tables while an online `ALTER TABLE` query is running.

It is built in the server, and is always enabled.

See the Online Schema Change page for functionality details.

For plugin version and maturity level, see this page.

This page is licensed: CC BY-SA / Gnu FDL

Query Cache Information Plugin

The `QUERY_CACHE_INFO` plugin creates the QUERY_CACHE_INFO table in the INFORMATION_SCHEMA database. This table shows all queries in the query cache. Querying this table acquires the query cache lock and will result in lock waits for queries that are using or expiring from the query cache. You must have the PROCESS privilege to query this table.

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'query_cache_info';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = query_cache_info
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'query_cache_info';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Example

```
SELECT statement_schema, statement_text, result_blocks_count,
  result_blocks_size FROM information_schema.query_cache_info;
+------------------+------------------+--------------------+--------------------+
| statement_schema | statement_text   | result_blocks_count | result_blocks_size |
+------------------+------------------+--------------------+--------------------+
| test             | SELECT * FROM t1 |                  1 |                512 |
+------------------+------------------+--------------------+--------------------+
```

Options

`query_cache_info`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--query-cache-info=value`
- Data Type: `enumerated`
- Default Value: `ON`

- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

Query Response Time Plugin

The `query_response_time` plugin creates the QUERY_RESPONSE_TIME table in the INFORMATION_SCHEMA database. The plugin also adds the SHOW QUERY_RESPONSE_TIME and FLUSH QUERY_RESPONSE_TIME statements.

The slow query log provides exact information about queries that take a long time to execute. However, sometimes there are a large number of queries that each take a very short amount of time to execute. This feature provides a tool for analyzing that information by counting and displaying the number of queries according to the length of time they took to execute.

This feature is based on Percona's Response Time Distribution.

Installing the Plugin

> **CURRENT**
>
> This shared library actually consists of a number of different plugins:
>
> - `QUERY_RESPONSE_TIME` - An INFORMATION_SCHEMA plugin that exposes statistics.
> - `QUERY_RESPONSE_TIME_AUDIT` - audit plugin, collects statistics.
>
> Both plugins need to be installed to get meaningful statistics.
>
> In addition, these additional plugins are available:
>
> - `QUERY_RESPONSE_TIME_READ`
> - `QUERY_RESPONSE_TIME_READ_WRITE`
> - `QUERY_RESPONSE_TIME_WRITE`
>
> **< 11.5**
>
> This shared library actually consists of a number of different plugins:
>
> - `QUERY_RESPONSE_TIME` - An INFORMATION_SCHEMA plugin that exposes statistics.
> - `QUERY_RESPONSE_TIME_AUDIT` - audit plugin, collects statistics.
>
> Both plugins need to be installed to get meaningful statistics.

Although the plugin's shared library is distributed with MariaDB by default, the plugins are not actually installed by MariaDB by default. There are two methods that can be used to install the plugins with MariaDB.

The first method can be used to install the plugin library without restarting the server. You can install the plugins dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'query_response_time';
```

The second method can be used to tell the server to load the plugin library when it starts up. The plugins can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = query_response_time
```

Note that in both cases you have to activate data collection by changing the query_response_time_stats setting to ON, it is OFF by default even when the plugin library is loaded.

You can change the setting at runtime with

```
SET GLOBAL query_response_time_stats=ON;
```

or in the options file after the plugin has been loaded:

```
[mariadb]
...
plugin_load_add = query_response_time
query_response_time_stats=ON;
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'query_response_time';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Response Time Distribution

The user can define time intervals that divide the range 0 to positive infinity into smaller intervals and then collect the number of commands whose execution times fall into each of those intervals.

Each interval is described as:

```
(range_base ^ n; range_base ^ (n+1)]
```

The range_base is some positive number (see Limitations). The interval is defined as the difference between two nearby powers of the range base.

If the range base equals 10, we have the following intervals:

```
(0; 10 ^ -6], (10 ^ -6; 10 ^ -5], (10 ^ -5; 10 ^ -4], ...,
  (10 ^ -1; 10 ^1], (10^1; 10^2]...(10^7; positive infinity]
```

or

```
(0; 0.000001], (0.000001; 0.000010], (0.000010; 0.000100], ...,
  (0.100000; 1.0]; (1.0; 10.0]...(1000000; positive infinity]
```

For each interval, a count is made of the queries with execution times that fell into that interval.

You can select the range of the intervals by changing the range base. For example, for base range=2 we have the following intervals:

```
(0; 2 ^ -19], (2 ^ -19; 2 ^ -18], (2 ^ -18; 2 ^ -17], ...,
  (2 ^ -1; 2 ^1], (2 ^ 1; 2 ^ 2]...(2 ^ 25; positive infinity]
```

or

```
(0; 0.000001], (0.000001, 0.000003], ...,
  (0.25; 0.5], (0.5; 2], (2; 4]...(8388608; positive infinity]
```

Small numbers look strange (i.e., don't look like powers of 2), because we lose precision on division when the ranges are calculated at runtime. In the resulting table, you look at the high boundary of the range:

```
SELECT * FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME;
+----------------+-------+----------------+
| TIME           | COUNT | TOTAL          |
+----------------+-------+----------------+
|       0.000001 |     0 |       0.000000 |
|       0.000010 |    17 |       0.000094 |
|       0.000100 |  4301 |       0.236555 |
|       0.001000 |  1499 |       0.824450 |
|       0.010000 | 14851 |      81.680502 |
|       0.100000 |  8066 |     443.635693 |
|       1.000000 |     0 |       0.000000 |
|      10.000000 |     0 |       0.000000 |
|     100.000000 |     1 |      55.937094 |
|    1000.000000 |     0 |       0.000000 |
|   10000.000000 |     0 |       0.000000 |
|  100000.000000 |     0 |       0.000000 |
| 1000000.000000 |     0 |       0.000000 |
| TOO LONG       |     0 | TOO LONG       |
+----------------+-------+----------------+
```

This means there were:

```
* 17 queries with 0.000001 < query execution time < = 0.000010 seconds; total execution time of the 17
queries = 0.000094 seconds

* 4301 queries with 0.000010 < query execution time < = 0.000100 seconds; total execution time of the
4301 queries = 0.236555 seconds

* 1499 queries with 0.000100 < query execution time < = 0.001000 seconds; total execution time of the
1499 queries = 0.824450 seconds

* 14851 queries with 0.001000 < query execution time < = 0.010000 seconds; total execution time of the
14851 queries = 81.680502 seconds

* 8066 queries with 0.010000 < query execution time < = 0.100000 seconds; total execution time of the
8066 queries = 443.635693 seconds

* 1 query with 10.000000 < query execution time < = 100.0000 seconds; total execution time of the 1 query
= 55.937094 seconds
```

Using the Plugin

Using the Information Schema Table

You can get the distribution by querying the QUERY_RESPONSE_TIME table in the INFORMATION_SCHEMA database:

```
SELECT * FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME;
```

You can also write more complex queries:

```
SELECT c.count, c.time,
(SELECT SUM(a.count) FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME AS a
   WHERE a.count != 0) AS query_count,
(SELECT COUNT(*)     FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME AS b
  WHERE b.count != 0) AS not_zero_region_count,
(SELECT COUNT(*)     FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME) AS region_count
FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME AS c
  WHERE c.count > 0;
```

Note: If query_response_time_stats is set to `ON`, then the execution times for these two `SELECT` queries will also be collected.

Using the SHOW Statement

As an alternative to the QUERY_RESPONSE_TIME table in the INFORMATION_SCHEMA database, you can also use the SHOW QUERY_RESPONSE_TIME statement:

```
SHOW QUERY_RESPONSE_TIME;
```

Flushing Plugin Data

Flushing the plugin data does two things:

- Clears the collected times from the QUERY_RESPONSE_TIME table in the INFORMATION_SCHEMA database.
- Reads the value of query_response_time_range_base and uses it to set the range base for the table.

Plugin data can be flushed with the FLUSH QUERY_RESPONSE_TIME statement:

```
FLUSH QUERY_RESPONSE_TIME;
```

Setting the query_response_time_flush system variable has the same effect:

```
SET GLOBAL query_response_time_flush=1;
```

> **CURRENT**
>
> It is possible to specify flushing read and/or write statements with the `FLUSH QUERY_RESPONSE_TIME_READ`, `FLUSH QUERY_RESPONSE_TIME_WRITE` and `FLUSH QUERY_RESPONSE_TIME_READ_WRITE` statements.
>
> **< 11.5**
>
> It is **not** possible to specify flushing read and/or write statements with the `FLUSH QUERY_RESPONSE_TIME_READ`, `FLUSH QUERY_RESPONSE_TIME_WRITE` and `FLUSH QUERY_RESPONSE_TIME_READ_WRITE` statements.

System Variables

`query_response_time_flush`

- Description: Updating this variable flushes the statistics and re-reads query_response_time_range_base.
- Command line: None
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

`query_response_time_range_base`

- Description: Select base of log for `QUERY_RESPONSE_TIME` ranges. WARNING: variable change takes affect only after flush.
- Command line: `--query-response-time-range-base=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`

- Default Value: `10`
- Range: `2` to `1000`

`query_response_time_exec_time_debug`

- Description: Pretend queries take this many microseconds. When 0 (the default) use the actual execution time.
  - This system variable is only available when the plugin is a debug build.
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `31536000`

`query_response_time_session_stats`

- Description: Controls query response time statistics collection for the current session: `ON` - enable, `OFF` - disable, `GLOBAL` (default) - use query_response_time_stats value.
- Command line: `query-response-time-session-stats=val]`
- Scope: Global, Session
- Dynamic: Yes
- Data Type: `enum`
- Default Value: `GLOBAL`
- Valid Values: `GLOBAL`, `ON`, `OFF`
- Introduced: MariaDB 11.5

`query_response_time_stats`

- Description: Enable or disable query response time statistics collecting.
- Command line: `query-response-time-stats[={0|1}]`
- Scope: Global
- Dynamic: Yes
- Data Type: `boolean`
- Default Value: `OFF`

Options

`query_response_time`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--query-response-time=value`

- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

`query_response_time_audit`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--query-response-time-audit=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

User Variables Plugin

The `user_variables` plugin creates the USER_VARIABLES table in the INFORMATION_SCHEMA database. This table contains information about user-defined variables.

Viewing

User-defined variables can be viewed by either querying the USER_VARIABLES, or by running `SHOW USER_VARIABLES`.

Flushing User-Defined Variables

User-defined variables are reset and the Information Schema table emptied with the FLUSH USER_VARIABLES statement.

Examples

```
SET @v1 = 0;
SET @v2 = 'abc';
SET @v3 = CAST(123 AS CHAR(5));

SHOW USER_VARIABLES;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| v3            | 123   |
| v2            | abc   |
| v1            | 0     |
+---------------+-------+

SELECT * FROM information_schema.USER_VARIABLES ORDER BY VARIABLE_NAME;
+---------------+----------------+---------------+--------------------+
| VARIABLE_NAME | VARIABLE_VALUE | VARIABLE_TYPE | CHARACTER_SET_NAME |
+---------------+----------------+---------------+--------------------+
| v1            | 0              | INT           | latin1             |
| v2            | abc            | VARCHAR       | utf8               |
| v3            | 123            | VARCHAR       | utf8               |
+---------------+----------------+---------------+--------------------+

FLUSH USER_VARIABLES;

SELECT * FROM information_schema.USER_VARIABLES ORDER BY VARIABLE_NAME;
Empty set (0.000 sec)
```

Installing the Plugin

> **Note:**
>
> In current versions, the `user_variables` plugin is statically linked into the server by default, so it does not need to be installed.

If it is not installed, however, here are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'user_variables';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = user_variables
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'user_variables';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Options

`user_variables`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--user-variables=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

This page is licensed: CC BY-SA / Gnu FDL

Password Validation Plugins

See Password Validation Plugin Overview.

General introduction into plugins that enforce specific security policies and complexity rules for user passwords.

See Cracklib Password Check Plugin.

The Cracklib Password Check Plugin enforces password strength by validating new passwords against the CrackLib library and its dictionary.

See Password Reuse Check Plugin.

The Password Reuse Check Plugin prevents users from reusing previous passwords, with a retention policy controlled by the password_reuse_check_interval variable.

See password_reuse_check_interval Variable.

This system variable defines the retention period in days for the password history used by the Password Reuse Check Plugin to prevent reuse.

See Simple Password Check Plugin.

The Simple Password Check Plugin enforces basic password complexity rules, such as minimum length and required numbers of digits, letters, and special characters.

This page is licensed: CC BY-SA / Gnu FDL

Password Validation Plugin Overview

Password validation means ensuring that user passwords meet certain minimal security requirements. A dedicated plugin API allows the creation of password validation plugins that will check user passwords as they are set (in SET PASSWORD and GRANT statements) and either allow or reject them.

SQL-Level Extensions

MariaDB comes with three password validation plugins — the simple_password_check plugin, the cracklib_password_check plugin and the password_reuse_check plugin. They are not enabled by default – use INSTALL SONAME (or INSTALL PLUGIN) statement to install them.

When at least one password plugin is loaded, all new passwords will be validated and password-changing statements will fail if the password will not pass validation checks. Several password validation plugin can be loaded at the same time — in this case a password must pass **all** validation checks by **all** plugins.

Password-Changing Statements

One can use various SQL statements to change a user password:

With Plain Text Password

```
SET PASSWORD = PASSWORD('plain-text password');
SET PASSWORD FOR `user`@`host` = PASSWORD('plain-text password');
SET PASSWORD = OLD_PASSWORD('plain-text password');
SET PASSWORD FOR `user`@`host` = OLD_PASSWORD('plain-text password');
CREATE USER `user`@`host` IDENTIFIED BY 'plain-text password';
GRANT PRIVILEGES TO `user`@`host` IDENTIFIED BY 'plain-text password';
```

These statements are subject to password validation. If at least one password validation plugin is loaded, plain-text passwords specified in these statements will be validated.

With Password Hash

```
SET PASSWORD = 'password hash';
SET PASSWORD FOR `user`@`host` = 'password hash';
CREATE USER `user`@`host` IDENTIFIED BY PASSWORD 'password hash';
CREATE USER `user`@`host` IDENTIFIED VIA mysql_native_password USING 'password hash';
CREATE USER `user`@`host` IDENTIFIED VIA mysql_old_password USING 'password hash';
GRANT PRIVILEGES TO `user`@`host` IDENTIFIED BY PASSWORD 'password hash';
GRANT PRIVILEGES TO `user`@`host` IDENTIFIED VIA mysql_native_password USING 'password hash';
GRANT PRIVILEGES TO `user`@`host` IDENTIFIED VIA mysql_old_password USING 'password hash';
```

These statements can not possibly use password validation — there is nothing to validate, the original plain-text password is not available.

MariaDB introduces a **strict password validation** mode — controlled by a strict_password_validation global server variable.

If the strict password validation is enabled and at least one password validation plugin is loaded, passwords that cannot be validated are rejected; otherwise, they're accepted. By default, a strict password validation is enabled (but note that it has no effect if no password validation plugin is loaded).

Examples

Failed password validation:

```
GRANT SELECT ON *.* to foobar IDENTIFIED BY 'raboof';
ERROR HY000: Your password does not satisfy the current policy requirements

SHOW WARNINGS;
+---------+------+--------------------------------------------------------------+
| Level   | Code | Message                                                      |
+---------+------+--------------------------------------------------------------+
| Warning | 1819 | cracklib: it is based on your username                       |
| Error   | 1819 | Your password does not satisfy the current policy requirements |
+---------+------+--------------------------------------------------------------+
```

Strict password validation:

```
GRANT SELECT ON *.* TO foo IDENTIFIED BY PASSWORD '2222222222222222';
ERROR HY000: The MariaDB server is running with the --strict-password-validation option so it cannot
execute this statement
```

This page is licensed: CC BY-SA / Gnu FDL

Cracklib Password Check Plugin

`cracklib_password_check` is a password validation plugin. It uses the CrackLib library to check the strength of new passwords. CrackLib is installed by default in many Linux distributions, since the system's Pluggable Authentication Module (PAM) authentication framework is usually configured to check the strength of new passwords with the pam_cracklib PAM module.

Note that passwords can be directly set as a hash, bypassing the password validation, if the strict_password_validation variable is `OFF` (it is `ON` by default).

The plugin requires at least cracklib 2.9.0, so it is not available on Debian/Ubuntu builds before Debian 8 Jessie/Ubuntu 14.04 Trusty, RedHat Enterprise Linux / CentOS 6.

Installing the Plugin's Package

The `cracklib_password_check` plugin's shared library is included in MariaDB packages as the `cracklib_password_check.so` or `cracklib_password_check.dll` shared library on systems where it can be built.

Installing on Linux

The `cracklib_password_check` plugin is included in `systemd` binary tarballs on Linux, but not in the older generic and `glibc_214` tarballs.

Installing with a Package Manager

The `cracklib_password_check` plugin can also be installed via a package manager on Linux. In order to do so, your system needs to be configured to install from one of the MariaDB repositories.

You can configure your package manager to install it from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script.

You can also configure your package manager to install it from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool.

Installing with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using yum or dnf. Starting with RHEL 8 and Fedora 22, `yum` has been replaced by `dnf`, which is the next major version of `yum`. However, `yum` commands still work on many systems that use `dnf`:

```
sudo yum install MariaDB-cracklib-password-check
```

Installing with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant DEB package from MariaDB's repository using apt-get:

```
sudo apt-get install mariadb-plugin-cracklib-password-check
```

Installing with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using zypper:

```
sudo zypper install MariaDB-cracklib-password-check
```

Installing the Plugin

Once the shared library is in place, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'cracklib_password_check';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = cracklib_password_check
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'cracklib_password_check';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Viewing CrackLib Errors

If password validation fails, then the original CrackLib error message can be viewed by executing SHOW WARNINGS.

Example

When creating a new password, if the criteria are not met, the following error is returned:

```
SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('abc');
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```

Known Issues

SELinux

When using the standard SELinux policy with the mode set to `enforcing`, `mariadbd` does not have access to `/usr/share/cracklib`, and you may see the following error when attempting to use the `cracklib_password_check` plugin:

```
CREATE USER `user`@`hostname` IDENTIFIED BY 's0mePwd123.';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements

SHOW WARNINGS;
+---------+------+--------------------------------------------------------------+
| Level   | Code | Message                                                      |
+---------+------+--------------------------------------------------------------+
| Warning | 1819 | cracklib: error loading dictionary                           |
| Error   | 1819 | Your password does not satisfy the current policy requirements |
| Error   | 1396 | Operation CREATE USER failed for 'user'@'hostname'            |
+---------+------+--------------------------------------------------------------+
```

And the SELinux `audit.log` contains errors like these:

```
type=AVC msg=audit(1548371977.821:66): avc:  denied  { read } for  pid=3537 comm="mysqld"
name="pw_dict.pwd" dev="xvda2" ino=564747 scontext=system_u:system_r:mysqld_t:s0
tcontext=system_u:object_r:crack_db_t:s0 tclass=file
type=SYSCALL msg=audit(1548371977.821:66): arch=c000003e syscall=2 success=no exit=-13 a0=7fdd2a674580
a1=0 a2=1b6 a3=1b items=0 ppid=1 pid=3537 auid=4294967295 uid=995 gid=992 euid=995 suid=995 fsuid=995
egid=992 sgid=992 fsgid=992 tty=(none) ses=4294967295 comm="mysqld" exe="/usr/sbin/mysqld"
subj=system_u:system_r:mysqld_t:s0 key=(null)
```

This can be fixed by creating an SELinux policy that allows `mysqld` to load the CrackLib dictionary:

```
cd /usr/share/mysql/policy/selinux/
tee ./mariadb-plugin-cracklib-password-check.te <<EOF

module mariadb-plugin-cracklib-password-check 1.0;

require {
        type mysqld_t;
        type crack_db_t;
        class file { execute setattr read create getattr execute_no_trans write ioctl open append unlink
};
        class dir { write search getattr add_name read remove_name open };
}

allow mysqld_t crack_db_t:dir { search read open };
allow mysqld_t crack_db_t:file { getattr read open };
EOF
sudo yum install selinux-policy-devel
make -f /usr/share/selinux/devel/Makefile mariadb-plugin-cracklib-password-check.pp
sudo semodule -i mariadb-plugin-cracklib-password-check.pp
```

See MDEV-18374 for more information.

System Variables

`cracklib_password_check_dictionary`

- Description: Sets the path to the CrackLib dictionary. If not set, the default CrackLib dictionary path is used. The parameter expects the base name of a cracklib dictionary (a set of three files with endings `.hwm`, `.pwd`, `.pwi`), not a directory path.
- Command line: `--cracklib-password-check-dictionary=value`
- Scope: Global
- Dynamic: No
- Data Type: `string`
- Default Value: Depends on the system. Often `/usr/share/cracklib/pw_dict`

Options

`cracklib_password_check`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.
    - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
  - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--cracklib-password-check=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

See Also

- Password Validation
- simple_password_check plugin - permits the setting of basic criteria for passwords

This page is licensed: CC BY-SA / Gnu FDL

Password Reuse Check Plugin

> **Note:**
>
> `password_reuse_check` is available from MariaDB 10.7.0.

Description

The plugin is used to prevent a user from reusing a password, which can be a requirement in some security policies. The password_reuse_check_interval system variable determines the retention period, in days, for a password. By default, this is zero, meaning unlimited retention. Old passwords are stored in the mysql.password_reuse_check_history table.

Note that passwords can be directly set as a hash, bypassing the password validation, if the strict_password_validation variable is `OFF` (it is `ON` by default).

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default.

You can install the plugin dynamically, without restarting the server, by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'password_reuse_check';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line

argument to mysqld or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = password_reuse_check
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'password_reuse_check';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Example

```
INSTALL SONAME 'password_reuse_check';

GRANT SELECT ON *.* TO user1@localhost identified by 'pwd1';
Query OK, 0 rows affected (0.038 sec)

GRANT SELECT ON *.* TO user1@localhost identified by 'pwd1';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements

GRANT SELECT ON *.* TO user1@localhost identified by 'pwd2';
Query OK, 0 rows affected (0.003 sec)

GRANT SELECT ON *.* TO user1@localhost identified by 'pwd1';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```

Versions

Version	Status	Introduced
1.0	Alpha	MariaDB 10.7.0
1.0	Beta	MariaDB 10.7.2
1.0	Gamma	MariaDB 10.7.4
2.0	Stable	MariaDB 10.7.7, MariaDB 10.8.7, MariaDB 10.9.5, MariaDB 10.10.2

Warning:

The bump to version 2.0 required the change of the stored format to mitigate an implementation weakness (MDEV-28838) and as such the bump from 1.0 to 2.0 will invalidate previously saved password reuse protections.

See Also

- Password Validation
- 10.7 preview feature: Password Reuse Check plugin (MariaDB Foundation blog post)

This page is licensed: CC BY-SA / Gnu FDL

password_reuse_check_interval Variable

The `password_reuse_check_interval` system variable is available when the password_reuse_check plugin is installed. It determines the retention period for the password history in days. Zero, the default, means that passwords are never discarded.

- Command line: `--password_reuse_check_interval=#`
- Scope: Global
- Read-only: No
- Data Type: `numeric`
- Default Value: `0`
- Range: `0` to `36500`

This page is licensed: CC BY-SA / Gnu FDL

Simple Password Check Plugin

`simple_password_check` is a password validation plugin. It can check whether a password contains at least a certain number of characters of a specific type. When first installed, a password is required to be at least eight characters, and requires at least one digit, one uppercase character, one lowercase character, and one character that is neither a digit nor a letter.

Note that passwords can be directly set as a hash, bypassing the password validation, if the strict_password_validation variable is `OFF` (it is `ON` by default).

Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing INSTALL SONAME or INSTALL PLUGIN:

```
INSTALL SONAME 'simple_password_check';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the –plugin-load or the –plugin-load-add options. This can be specified as a command-line argument to mariadbd, or it can be specified in a relevant server option group in an option file:

```
[mariadb]
...
plugin_load_add = simple_password_check
```

Uninstalling the Plugin

You can uninstall the plugin dynamically by executing UNINSTALL SONAME or UNINSTALL PLUGIN:

```
UNINSTALL SONAME 'simple_password_check';
```

If you installed the plugin by providing the –plugin-load or the –plugin-load-add options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

Example

When creating a new password, if the criteria are not met, the following error is returned:

```
SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('abc');
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```

Known Issues

System Variables

`simple_password_check_digits`

- Description: A password must contain at least this many digits.
- Command line: `--simple-password-check-digits=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `1000`

`simple_password_check_letters_same_case`

- Description: A password must contain at least this many upper-case and this many lower-case letters.
- Command line: `--simple-password-check-letters-same-case=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `1000`

`simple_password_check_minimal_length`

- Description: A password must contain at least this many characters.
- Command line: `--simple-password-check-minimal-length=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `8`
- Range: `0` to `1000`

`simple_password_check_other_characters`

- Description: A password must contain at least this many characters that are neither digits nor letters.
- Command line: `--simple-password-check-other-characters=#`
- Scope: Global
- Dynamic: Yes
- Data Type: `numeric`
- Default Value: `1`
- Range: `0` to `1000`

Options

`simple_password_check`

- Description: Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - `OFF` - Disables the plugin without removing it from the mysql.plugins table.

        - `ON` - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
        - `FORCE` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
        - `FORCE_PLUS_PERMANENT` - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with UNINSTALL SONAME or UNINSTALL PLUGIN while the server is running.
    - See Plugin Overview: Configuring Plugin Activation at Server Startup for more information.
- Command line: `--simple-password-check=value`
- Data Type: `enumerated`
- Default Value: `ON`
- Valid Values: `OFF`, `ON`, `FORCE`, `FORCE_PLUS_PERMANENT`

See Also

- Password Validation
- cracklib_password_check plugin - use the Cracklib password-strength checking library

This page is licensed: CC BY-SA / Gnu FDL

System Tables

The MariaDB Server includes several system tables that store metadata about database objects, user privileges, and server status.

See Information Schema.

The `information_schema` is a standard-compliant database that provides information about all other databases, tables, and columns on the server.

See Performance Schema.

The `performance_schema` allows you to monitor server execution at a low level with minimal impact on performance.

See Sys Schema.

The `sys` schema contains a collection of views, procedures, and functions that simplify the interpretation of data from the `performance_schema`.

See The mysql Database Tables.

Contains the core system tables in the `mysql` database, including privilege tables, logging tables, and status variables.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema

The `information_schema` is a standard-compliant database available in every MariaDB instance, providing metadata about all other databases, tables, columns, and server-level objects.

See Information Schema Tables.

Provides a comprehensive list of all tables available within the `information_schema`, including their purpose and the metadata they provide.

See TIME_MS column in INFORMATION_SCHEMA.PROCESSLIST.

Details the `TIME_MS` column in the `INFORMATION_SCHEMA.PROCESSLIST` table, which provides high-resolution timing for active server processes.

This page is licensed: CC BY-SA / Gnu FDL

TIME_MS column in INFORMATION_SCHEMA.PROCESSLIST

In MariaDB, an extra column `TIME_MS` was added to the INFORMATION_SCHEMA.PROCESSLIST table. This column shows the same information as the column '`TIME`', but in units of milliseconds with microsecond precision (the unit and precision of the `TIME` column is one second).

For details about microseconds support in MariaDB, see microseconds in MariaDB.

The value displayed in the `TIME` and `TIME_MS` columns is the period of time that the given thread has been in its current state. Thus, it can be used to check for example how long a thread has been executing the current query, or for how long it has been idle.

```
SELECT id, TIME, time_ms, command, state FROM
   information_schema.processlist, (SELECT sleep(2)) t;
+----+------+----------+---------+-----------+
| id | time | time_ms  | command | state     |
+----+------+----------+---------+-----------+
| 37 |    2 | 2000.493 | Query   | executing |
+----+------+----------+---------+-----------+
```

Note that as a difference to MySQL, in MariaDB the `TIME` column (and also the `TIME_MS` column) are not affected by any setting of @TIMESTAMP. This means that it can be reliably used also for threads that change `@TIMESTAMP` (such as the replication SQL thread). See also MySQL Bug #22047.

As a consequence of this, the `TIME` column of `SHOW FULL PROCESSLIST` and `INFORMATION_SCHEMA.PROCESSLIST` cannot be used to determine if a slave is lagging behind. For this, use instead the `Seconds_Behind_Master` column in the output of SHOW SLAVE STATUS.

The addition of the `TIME_MS` column is based on the microsec_process patch, developed by Percona.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema Tables

- Information Schema ALL_PLUGINS Table
- Information Schema APPLICABLE_ROLES Table
- Information Schema CATALOG Table
- Information Schema CHARACTER_SETS Table
- Information Schema CHECK_CONSTRAINTS Table
- Information Schema CLIENT_STATISTICS Table
- Information Schema COLLATION_CHARACTER_SET_APPLICABILITY Table
- Information Schema COLLATIONS Table
- Information Schema COLUMN_PRIVILEGES Table
- Information Schema COLUMNS Table
- Information Schema DISKS Table
- Information Schema ENABLED_ROLES Table
- Information Schema ENGINES Table
- Information Schema EVENTS Table
- Information Schema FEEDBACK Table
- Information Schema FILE_KEY_MANAGEMENT_KEYS Table

- Information Schema HASHICORP_KEY_MANAGEMENT_CACHE Table
- Information Schema GEOMETRY_COLUMNS Table
- Information Schema GLOBAL_STATUS and SESSION_STATUS Tables
- Information Schema GLOBAL_VARIABLES and SESSION_VARIABLES Tables
- Information Schema INDEX_STATISTICS Table
- Information Schema KEY_CACHES Table
- Information Schema KEY_COLUMN_USAGE Table
- Information Schema KEY_PERIOD_USAGE Table
- Information Schema KEYWORDS Table
- Information Schema LOCALES Table
- Information Schema METADATA_LOCK_INFO Table
- Information Schema MROONGA_STATS Table
- Information Schema OPTIMIZER_TRACE Table
- Information Schema PARAMETERS Table
- Information Schema PARTITIONS Table
- Information Schema PERIODS Table
- Information Schema PLUGINS Table
- Information Schema PROCESSLIST Table
- Information Schema PROFILING Table
- Information Schema QUERY_CACHE_INFO Table
- Information Schema QUERY_RESPONSE_TIME Table
- Information Schema REFERENTIAL_CONSTRAINTS Table
- Information Schema ROUTINES Table
- Information Schema SCHEMA_PRIVILEGES Table
- Information Schema SCHEMATA Table
- Information Schema SEQUENCES Table
- Information Schema SLAVE_STATUS Table
- Information Schema SPATIAL_REF_SYS Table
- Information Schema SPIDER_ALLOC_MEM Table
- Information Schema SQL_FUNCTIONS Table
- Information Schema STATISTICS Table
- Information Schema SYSTEM_VARIABLES Table
- Information Schema TABLE_CONSTRAINTS Table
- Information Schema TABLE_PRIVILEGES Table
- Information Schema TABLE_STATISTICS Table
- Information Schema TABLES Table
- Information Schema TABLESPACES Table
- Information Schema THREAD_POOL_GROUPS Table
- Information Schema THREAD_POOL_QUEUES Table
- Information Schema THREAD_POOL_STATS Table
- Information Schema THREAD_POOL_WAITS Table
- Information Schema TRIGGERED_UPDATE_COLUMNS
- Information Schema TRIGGERS Table
- Information Schema USER_PRIVILEGES Table
- Information Schema USER_STATISTICS Table

- Information Schema USER_VARIABLES Table
- Information Schema USERS Table
- Information Schema WSREP_BF_ABORTS
- Information Schema WSREP_CERT_KEYS
- Information Schema WSREP_CERT_KEYS_HISTORY
- Information Schema WSREP_CONNECTIONS
- Information Schema WSREP_MEMBERSHIP Table
- Information Schema WSREP_STATUS Table
- Information Schema WSREP_THD_STATE
- Information Schema WSREP_THD_STATE_HISTORY
- Information Schema InnoDB Tables
- Information Schema MyRocks Tables
- Information Schema XtraDB Tables

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ALL_PLUGINS Table

Description

The Information Schema `ALL_PLUGINS` table contains information about server plugins, whether installed or not.

It contains the following columns:

Column	Description
PLUGIN_NAME	Name of the plugin.
PLUGIN_VERSION	Version from the plugin's general type descriptor.
PLUGIN_STATUS	Plugin status, one of ACTIVE, INACTIVE, DISABLED, DELETED or NOT INSTALLED.
PLUGIN_TYPE	Plugin type; STORAGE ENGINE, INFORMATION_SCHEMA, AUTHENTICATION, REPLICATION, DAEMON or AUDIT.
PLUGIN_TYPE_VERSION	Version from the plugin's type-specific descriptor.
PLUGIN_LIBRARY	Plugin's shared object file name, located in the directory specified by the plugin_dir system variable, and used by the INSTALL PLUGIN and UNINSTALL PLUGIN statements. NULL if the plugin is complied in and cannot be uninstalled.
PLUGIN_LIBRARY_VERSION	Version from the plugin's API interface.
PLUGIN_AUTHOR	Author of the plugin.
PLUGIN_DESCRIPTION	Description.
PLUGIN_LICENSE	Plugin's licence.
LOAD_OPTION	How the plugin was loaded; one of OFF, ON, FORCE or FORCE_PLUS_PERMANENT. See Installing Plugins.
PLUGIN_MATURITY	Plugin's maturity level; one of Unknown, Experimental, Alpha, Beta,'Gamma, and Stable.
PLUGIN_AUTH_VERSION	Plugin's version as determined by the plugin author. An example would be '0.99 beta 1'.

It provides a superset of the information shown by the SHOW PLUGINS SONAME statement, as well as the information_schema.PLUGINS table. For specific information about storage engines (a particular type of plugin), see the Information Schema ENGINES table and the SHOW ENGINES statement.

The table is not a standard Information Schema table, and is a MariaDB extension.

Example

```
SELECT * FROM information_schema.all_plugins\G
*************************** 1. row ***************************
           PLUGIN_NAME: binlog
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: ACTIVE
           PLUGIN_TYPE: STORAGE ENGINE
   PLUGIN_TYPE_VERSION: 100314.0
        PLUGIN_LIBRARY: NULL
PLUGIN_LIBRARY_VERSION: NULL
         PLUGIN_AUTHOR: MySQL AB
    PLUGIN_DESCRIPTION: This is a pseudo storage engine to represent the binlog in a transaction
        PLUGIN_LICENSE: GPL
           LOAD_OPTION: FORCE
       PLUGIN_MATURITY: Stable
   PLUGIN_AUTH_VERSION: 1.0
*************************** 2. row ***************************
           PLUGIN_NAME: mysql_native_password
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: ACTIVE
           PLUGIN_TYPE: AUTHENTICATION
   PLUGIN_TYPE_VERSION: 2.1
        PLUGIN_LIBRARY: NULL
PLUGIN_LIBRARY_VERSION: NULL
         PLUGIN_AUTHOR: R.J.Silk, Sergei Golubchik
    PLUGIN_DESCRIPTION: Native MySQL authentication
        PLUGIN_LICENSE: GPL
           LOAD_OPTION: FORCE
       PLUGIN_MATURITY: Stable
   PLUGIN_AUTH_VERSION: 1.0
*************************** 3. row ***************************
           PLUGIN_NAME: mysql_old_password
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: ACTIVE
           PLUGIN_TYPE: AUTHENTICATION
   PLUGIN_TYPE_VERSION: 2.1
        PLUGIN_LIBRARY: NULL
PLUGIN_LIBRARY_VERSION: NULL
         PLUGIN_AUTHOR: R.J.Silk, Sergei Golubchik
    PLUGIN_DESCRIPTION: Old MySQL-4.0 authentication
        PLUGIN_LICENSE: GPL
           LOAD_OPTION: FORCE
       PLUGIN_MATURITY: Stable
   PLUGIN_AUTH_VERSION: 1.0
...
*************************** 104. row ***************************
           PLUGIN_NAME: WSREP_MEMBERSHIP
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: NOT INSTALLED
           PLUGIN_TYPE: INFORMATION SCHEMA
   PLUGIN_TYPE_VERSION: 100314.0
        PLUGIN_LIBRARY: wsrep_info.so
PLUGIN_LIBRARY_VERSION: 1.13
         PLUGIN_AUTHOR: Nirbhay Choubey
    PLUGIN_DESCRIPTION: Information about group members
        PLUGIN_LICENSE: GPL
           LOAD_OPTION: OFF
       PLUGIN_MATURITY: Stable
   PLUGIN_AUTH_VERSION: 1.0
*************************** 105. row ***************************
           PLUGIN_NAME: WSREP_STATUS
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: NOT INSTALLED
           PLUGIN_TYPE: INFORMATION SCHEMA
   PLUGIN_TYPE_VERSION: 100314.0
        PLUGIN_LIBRARY: wsrep_info.so
PLUGIN_LIBRARY_VERSION: 1.13
         PLUGIN_AUTHOR: Nirbhay Choubey
```

```
    PLUGIN_DESCRIPTION: Group view information
        PLUGIN_LICENSE: GPL
           LOAD_OPTION: OFF
       PLUGIN_MATURITY: Stable
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema APPLICABLE_ROLES Table

The Information Schema `APPLICABLE_ROLES` table shows the role authorizations that the current user may use.

It contains the following columns:

Column	Description
GRANTEE	Account that the role was granted to.
ROLE_NAME	Name of the role.
IS_GRANTABLE	Whether the role can be granted or not.
IS_DEFAULT	Whether the role is the user's default role or not

The current role is in the ENABLED_ROLES Information Schema table.

Example

```
SELECT * FROM information_schema.APPLICABLE_ROLES;
+----------------+-------------+--------------+------------+
| GRANTEE        | ROLE_NAME   | IS_GRANTABLE | IS_DEFAULT |
+----------------+-------------+--------------+------------+
| root@localhost | journalist  | YES          | NO         |
| root@localhost | staff       | YES          | NO         |
| root@localhost | dd          | YES          | NO         |
| root@localhost | dog         | YES          | NO         |
+----------------+-------------+--------------+------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema CATALOG Table

> **Note:**
>
> This table is available as of MariaDB 12.0.

The Information Schema `CATALOG` table stores information about catalogs on the server.

It contains the following columns:

Column	Description
CATALOG_NAME	Catalog name.
DEFAULT_CHARACTER_SET_NAME	Default character set for the database.
DEFAULT_COLLATION_NAME	Default collation.
SCHEMA_COMMENT	Catalog comment

Example

```
MariaDB [def.test]> SELECT * FROM INFORMATION_SCHEMA.CATALOGS\G
*************************** 1. row ***************************
              CATALOG_NAME: c1
DEFAULT_CHARACTER_SET_NAME: latin1
    DEFAULT_COLLATION_NAME: latin1_swedish_ci
           CATALOG_COMMENT: This is catalog c1
*************************** 2. row ***************************
              CATALOG_NAME: cat2
DEFAULT_CHARACTER_SET_NAME: latin1
    DEFAULT_COLLATION_NAME: latin1_swedish_ci
           CATALOG_COMMENT:
*************************** 3. row ***************************
              CATALOG_NAME: def
DEFAULT_CHARACTER_SET_NAME: latin1
    DEFAULT_COLLATION_NAME: latin1_swedish_ci
           CATALOG_COMMENT: default catalog
...
```

See Also

- SHOW CATALOGS
- SHOW CREATE CATALOG
- CREATE CATALOG
- DROP CATALOG
- Character Sets and Collations

This page is licensed: CC BY-SA / Gnu FDL

Information Schema CHARACTER_SETS Table

The Information Schema `CHARACTER_SETS` table contains a list of supported character sets, their default collations and maximum lengths.

It contains the following columns:

Column	Description
CHARACTER_SET_NAME	Name of the character set.
DEFAULT_COLLATE_NAME	Default collation used.
DESCRIPTION	Character set description.
MAXLEN	Maximum length.

The SHOW CHARACTER SET statement returns the same results (although in a different order), and both can be refined in the same way. For example, the following two statements return the same results:

```
SHOW CHARACTER SET WHERE Maxlen LIKE '2';
```

and

```
SELECT * FROM information_schema.CHARACTER_SETS
WHERE MAXLEN LIKE '2';
```

See Setting Character Sets and Collations for details on specifying the character set at the server, database, table and column levels, and Supported Character Sets and Collations for a full list of supported characters sets and collations.

Example

```
SELECT CHARACTER_SET_NAME FROM information_schema.CHARACTER_SETS
WHERE DEFAULT_COLLATE_NAME LIKE '%chinese%';
+--------------------+
| CHARACTER_SET_NAME |
+--------------------+
| big5               |
| gb2312             |
| gbk                |
+--------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema CHECK_CONSTRAINTS Table

The Information Schema `CHECK_CONSTRAINTS` table stores metadata about the constraints defined for tables in all databases.

It contains the following columns:

Column	Description
CONSTRAINT_CATALOG	Always contains the string 'def'.
CONSTRAINT_SCHEMA	Database name.
CONSTRAINT_NAME	Constraint name.
TABLE_NAME	Table name.
LEVEL	Type of the constraint ('Column' or 'Table'). From MariaDB 10.5.10.
CHECK_CLAUSE	Constraint clause.

Example

A table with a numeric table check constraint and with a default check constraint name:

```
CREATE TABLE t ( a INT, CHECK (a>10));
```

To see check constraint call `check_constraints` table from information schema.

```
SELECT * FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS\G
```

```
*************************** 1. row ***************************
CONSTRAINT_CATALOG: def
 CONSTRAINT_SCHEMA: test
   CONSTRAINT_NAME: CONSTRAINT_1
        TABLE_NAME: t
      CHECK_CLAUSE: `a` > 10
```

A new table check constraint called `a_upper`:

```
ALTER TABLE t ADD CONSTRAINT a_upper CHECK (a<100);
```

```
SELECT * FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS\G
```

```
*************************** 1. row ***************************
CONSTRAINT_CATALOG: def
 CONSTRAINT_SCHEMA: test
   CONSTRAINT_NAME: CONSTRAINT_1
        TABLE_NAME: t
     CHECK_CLAUSE: `a` > 10
*************************** 2. row ***************************
CONSTRAINT_CATALOG: def
 CONSTRAINT_SCHEMA: test
   CONSTRAINT_NAME: a_upper
        TABLE_NAME: t
     CHECK_CLAUSE: `a` < 100
```

A new table `tt` with a field check constraint called `b` , as well as a table check constraint called `b_upper`:

```
CREATE TABLE tt(b INT CHECK(b>0),CONSTRAINT b_upper CHECK(b<50));

SELECT * FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS;
+--------------------+-------------------+-----------------+------------+--------------+
| CONSTRAINT_CATALOG | CONSTRAINT_SCHEMA | CONSTRAINT_NAME | TABLE_NAME | CHECK_CLAUSE |
+--------------------+-------------------+-----------------+------------+--------------+
| def                | test              | b               | tt         | `b` > 0      |
| def                | test              | b_upper         | tt         | `b` < 50     |
| def                | test              | CONSTRAINT_1    | t          | `a` > 10     |
| def                | test              | a_upper         | t          | `a` < 100    |
+--------------------+-------------------+-----------------+------------+--------------+
```

Note: The name of the field constraint is the same as the field name.

After dropping the default table constraint called `CONSTRAINT_1`:

```
ALTER TABLE t DROP CONSTRAINT CONSTRAINT_1;

SELECT * FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS;
+--------------------+-------------------+-----------------+------------+--------------+
| CONSTRAINT_CATALOG | CONSTRAINT_SCHEMA | CONSTRAINT_NAME | TABLE_NAME | CHECK_CLAUSE |
+--------------------+-------------------+-----------------+------------+--------------+
| def                | test              | b               | tt         | `b` > 0      |
| def                | test              | b_upper         | tt         | `b` < 50     |
| def                | test              | a_upper         | t          | `a` < 100    |
+--------------------+-------------------+-----------------+------------+--------------+
```

Trying to insert invalid arguments into table `t` and `tt` generates an error.

```
INSERT INTO t VALUES (10),(20),(100);
ERROR 4025 (23000): CONSTRAINT `a_upper` failed for `test`.`t`

INSERT INTO tt VALUES (10),(-10),(100);
ERROR 4025 (23000): CONSTRAINT `b` failed for `test`.`tt`

INSERT INTO tt VALUES (10),(20),(100);
ERROR 4025 (23000): CONSTRAINT `b_upper` failed for `test`.`tt`
```

Note:

The following statement can be performed from MariaDB 10.5.10.

```
CREATE TABLE majra(CHECK(x>0), x INT, y INT CHECK(y < 0), z INT,
                            CONSTRAINT z CHECK(z>0), CONSTRAINT xyz CHECK(x<10 AND y<10 AND z<10));
Query OK, 0 rows affected (0.036 sec)

SHOW CREATE TABLE majra;
+-------+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+
| Table | Create Table
|
+-------+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+
| majra | CREATE TABLE `majra` (
  `x` int(11) DEFAULT NULL,
  `y` int(11) DEFAULT NULL CHECK (`y` < 0),
  `z` int(11) DEFAULT NULL,
  CONSTRAINT `CONSTRAINT_1` CHECK (`x` > 0),
  CONSTRAINT `z` CHECK (`z` > 0),
  CONSTRAINT `xyz` CHECK (`x` < 10 AND `y` < 10 AND `z` < 10)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-------+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+
1 row in set (0.000 sec)

SELECT * FROM information_schema.check_constraints WHERE table_name='majra';
+--------------------+-------------------+------------+-----------------+--------+-----------------------
-------------+
| CONSTRAINT_CATALOG | CONSTRAINT_SCHEMA | TABLE_NAME | CONSTRAINT_NAME | LEVEL  | CHECK_CLAUSE
|
+--------------------+-------------------+------------+-----------------+--------+-----------------------
-------------+
| def                | test              | majra      | y               | Column | `y` < 0
|
| def                | test              | majra      | CONSTRAINT_1    | Table  | `x` > 0
|
| def                | test              | majra      | z               | Table  | `z` > 0
|
| def                | test              | majra      | xyz             | Table  | `x` < 10 and `y` < 10
and `z` < 10 |
+--------------------+-------------------+------------+-----------------+--------+-----------------------
-------------+
4 rows in set (0.001 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema CLIENT_STATISTICS Table

The Information Schema `CLIENT_STATISTICS` table holds statistics about client connections. This is part of the User Statistics feature, which is not enabled by default.

It contains the following columns:

Field	Type	Notes
CLIENT	VARCHAR(64)	The IP address or hostname the connection originated from.
TOTAL_CONNECTIONS	BIGINT(21)	The number of connections created for this client.
CONCURRENT_CONNECTIONS	BIGINT(21)	The number of concurrent connections for this client.
CONNECTED_TIME	BIGINT(21)	The cumulative number of seconds elapsed while there were connections from this client.

Field	Type	Notes
BUSY_TIME	DOUBLE	The cumulative number of seconds there was activity on connections from this client.
CPU_TIME	DOUBLE	The cumulative CPU time elapsed while servicing this client's connections. Note that this number may be wrong on SMP system if there was a CPU migration for the thread during the execution of the query.
BYTES_RECEIVED	BIGINT(21)	The number of bytes received from this client's connections.
BYTES_SENT	BIGINT(21)	The number of bytes sent to this client's connections.
BINLOG_BYTES_WRITTEN	BIGINT(21)	The number of bytes written to the binary log from this client's connections.
ROWS_READ	BIGINT(21)	The number of rows read by this client's connections.
ROWS_SENT	BIGINT(21)	The number of rows sent by this client's connections.
ROWS_DELETED	BIGINT(21)	The number of rows deleted by this client's connections.
ROWS_INSERTED	BIGINT(21)	The number of rows inserted by this client's connections.
ROWS_UPDATED	BIGINT(21)	The number of rows updated by this client's connections.
KEY_READ_HITS	BIGINT(21)	From MariaDB 11.5
KEY_READ_MISSES	BIGINT(21)	From MariaDB 11.5
SELECT_COMMANDS	BIGINT(21)	The number of SELECT commands executed from this client's connections.
UPDATE_COMMANDS	BIGINT(21)	The number of UPDATE commands executed from this client's connections.
OTHER_COMMANDS	BIGINT(21)	The number of other commands executed from this client's connections.
COMMIT_TRANSACTIONS	BIGINT(21)	The number of COMMIT commands issued by this client's connections.
ROLLBACK_TRANSACTIONS	BIGINT(21)	The number of ROLLBACK commands issued by this client's connections.
DENIED_CONNECTIONS	BIGINT(21)	The number of connections denied to this client.
LOST_CONNECTIONS	BIGINT(21)	The number of this client's connections that were terminated uncleanly.
ACCESS_DENIED	BIGINT(21)	The number of times this client's connections issued commands that were denied.
EMPTY_QUERIES	BIGINT(21)	The number of times this client's connections sent queries that returned no results to the server.
TOTAL_SSL_CONNECTIONS	BIGINT(21)	The number of TLS connections created for this client.
MAX_STATEMENT_TIME_EXCEEDED	BIGINT(21)	The number of times a statement was aborted, because it was executed longer than its MAX_STATEMENT_TIME threshold.

Example

```
SELECT * FROM information_schema.CLIENT_STATISTICS\G
*************************** 1. row ***************************
                CLIENT: localhost
     TOTAL_CONNECTIONS: 3
CONCURRENT_CONNECTIONS: 0
        CONNECTED_TIME: 4883
             BUSY_TIME: 0.009722
              CPU_TIME: 0.0102131
        BYTES_RECEIVED: 841
            BYTES_SENT: 13897
  BINLOG_BYTES_WRITTEN: 0
             ROWS_READ: 0
             ROWS_SENT: 214
          ROWS_DELETED: 0
         ROWS_INSERTED: 207
          ROWS_UPDATED: 0
       SELECT_COMMANDS: 10
       UPDATE_COMMANDS: 0
        OTHER_COMMANDS: 13
   COMMIT_TRANSACTIONS: 0
 ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
      LOST_CONNECTIONS: 0
         ACCESS_DENIED: 0
         EMPTY_QUERIES: 1
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema COLLATION_CHARACTER_SET_APPLICABILITY Table

The Information Schema `COLLATION_CHARACTER_SET_APPLICABILITY` table shows which character sets are associated with which collations.

It contains the following columns:

Column	Description
COLLATION_NAME	Collation name.
CHARACTER_SET_NAME	Name of the associated character set.
FULL_COLLATION_NAME	Name of the associated character set/collation combination.
ID	The unique identifier of this character set/collation combination.
IS_DEFAULT	If the collation is the default for this character set.

The COLLATIONS table is table of the base `COLLATION_NAMES` in the same way that CHARACTER_SETS table is table of the base `CHARACTER_SET_NAMES`. The `COLLATION_CHARACTER_SET_APPLICABILITY` table is the mapping between collations and character sets.

When joining the information_schema.TABLES table with its field `TABLE_COLLATIONS`, this should be joined to `FULL_COLLATION_NAME` in the `COLLATION_CHARACTER_SET_APPLICABILITY` table.

See Setting Character Sets and Collations for details on specifying the character set at the server, database, table and column levels.

Example

```
SELECT * FROM information_schema.COLLATION_CHARACTER_SET_APPLICABILITY  WHERE
  CHARACTER_SET_NAME='utf32' ORDER BY IS_DEFAULT DESC, ID LIMIT 10;
+--------------------+--------------------+--------------------+-----+------------+
| COLLATION_NAME     | CHARACTER_SET_NAME | FULL_COLLATION_NAME | ID  | IS_DEFAULT |
+--------------------+--------------------+--------------------+-----+------------+
| utf32_general_ci   | utf32              | utf32_general_ci    |  60 | Yes        |
| utf32_bin          | utf32              | utf32_bin           |  61 |            |
| utf32_unicode_ci   | utf32              | utf32_unicode_ci    | 160 |            |
| utf32_icelandic_ci | utf32              | utf32_icelandic_ci  | 161 |            |
| utf32_latvian_ci   | utf32              | utf32_latvian_ci    | 162 |            |
| utf32_romanian_ci  | utf32              | utf32_romanian_ci   | 163 |            |
| utf32_slovenian_ci | utf32              | utf32_slovenian_ci  | 164 |            |
| utf32_polish_ci    | utf32              | utf32_polish_ci     | 165 |            |
| utf32_estonian_ci  | utf32              | utf32_estonian_ci   | 166 |            |
| utf32_spanish_ci   | utf32              | utf32_spanish_ci    | 167 |            |
+--------------------+--------------------+--------------------+-----+------------+
10 rows in set (0.004 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema COLLATIONS Table

The Information Schema `COLLATIONS` table contains a list of supported collations.

It contains the following columns:

Column	Description
COLLATION_NAME	Name of the collation.
CHARACTER_SET_NAME	Associated character set.
ID	Collation id.
IS_DEFAULT	Whether the collation is the character set's default.
IS_COMPILED	Whether the collation is compiled into the server.
SORTLEN	Sort length, used for determining the memory used to sort strings in this collation.
PAD_ATTRIBUTE	
COMMENT	

The SHOW COLLATION statement returns the same results and both can be reduced in a similar way.

CURRENT

The following two statements return the same results:

```
SHOW COLLATION WHERE Charset LIKE 'utf8mb3';
```

```
SELECT * FROM information_schema.COLLATIONS
WHERE CHARACTER_SET_NAME LIKE 'utf8mb3';
```

< 10.5

The following two statements return the same results:

```
SHOW COLLATION WHERE Charset LIKE 'utf8';
```

```
SELECT * FROM information_schema.COLLATIONS
WHERE CHARACTER_SET_NAME LIKE 'utf8';
```

NO PAD Collations

`NO PAD` collations regard trailing spaces as normal characters. You can get a list of all `NO PAD` collations as follows:

CURRENT

```
SELECT collation_name FROM information_schema.COLLATIONS
WHERE pad_attribute = "NO PAD";
+------------------------------+
| collation_name               |
+------------------------------+
| big5_chinese_nopad_ci        |
| big5_nopad_bin               |
...
```

< 12.1

```
SELECT collation_name FROM information_schema.COLLATIONS
WHERE collation_name LIKE "%nopad%";
+------------------------------+
| collation_name               |
+------------------------------+
| big5_chinese_nopad_ci        |
| big5_nopad_bin               |
...
```

In comparisons, `NO PAD` collations evaluate to `0` (`FALSE`). Example:

```
SELECT 'a ' = 'a';
+------------+
| 'a ' = 'a' |
+------------+
|          0 |
+------------+
```

PAD SPACE Collations

`PAD SPACE` collations pad strings to equal lengths in comparisons, so that the comparison evaluates to `1` (`TRUE`). Example:

```
SELECT 'a ' = 'a';
+------------+
| 'a ' = 'a' |
+------------+
|          1 |
+------------+
```

Example

```
SELECT * FROM information_schema.COLLATIONS;
+-----------------------------+--------------------+------+------------+-------------+---------+
| COLLATION_NAME              | CHARACTER_SET_NAME | ID   | IS_DEFAULT | IS_COMPILED | SORTLEN |
+-----------------------------+--------------------+------+------------+-------------+---------+
| big5_chinese_ci             | big5               |    1 | Yes        | Yes         |       1 |
| big5_bin                    | big5               |   84 |            | Yes         |       1 |
| big5_chinese_nopad_ci       | big5               | 1025 |            | Yes         |       1 |
| big5_nopad_bin              | big5               | 1108 |            | Yes         |       1 |
| dec8_swedish_ci             | dec8               |    3 | Yes        | Yes         |       1 |
| dec8_bin                    | dec8               |   69 |            | Yes         |       1 |
| dec8_swedish_nopad_ci       | dec8               | 1027 |            | Yes         |       1 |
| dec8_nopad_bin              | dec8               | 1093 |            | Yes         |       1 |
| cp850_general_ci            | cp850              |    4 | Yes        | Yes         |       1 |
| cp850_bin                   | cp850              |   80 |            | Yes         |       1 |
...
```

See Also

- Setting Character Sets and Collations - specifying the character set at the server, database, table and column levels
- Supported Character Sets and Collations - full list of supported characters sets and collations.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema COLUMN_PRIVILEGES Table

The Information Schema `COLUMN_PRIVILEGES` table contains column privilege information derived from the mysql.columns_priv grant table.

It has the following columns:

Column	Description
GRANTEE	In the format user_name@host_name.
TABLE_CATALOG	Always def.
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
COLUMN_NAME	Column name.
PRIVILEGE_TYPE	One of SELECT, INSERT, UPDATE or REFERENCES.
IS_GRANTABLE	Whether the user has the GRANT OPTION for this privilege.

Similar information can be accessed with the SHOW FULL COLUMNS and SHOW GRANTS statements. See the GRANT statement for details about privileges.

This information is also stored in the columns_priv table, in the `mysql` system database.

For a description of the privileges that are shown in this table, see column privileges.

Example

In the following example, no column-level privilege has been explicitly assigned:

```
SELECT * FROM information_schema.COLUMN_PRIVILEGES;
Empty SET
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema COLUMNS Table

The Information Schema `COLUMNS` table provides information about columns in each table on the server.

It contains the following columns:

Column	Description
TABLE_CATALOG	Always contains the string `'def'`.
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
COLUMN_NAME	Column name.
ORDINAL_POSITION	Column position in the table. Can be used for ordering.
COLUMN_DEFAULT	Default value for the column. Literals are quoted to distinguish them from expressions. `NULL` means that the column has no default.
IS_NULLABLE	Whether the column can contain `NULL` values.
DATA_TYPE	The column's data type.
CHARACTER_MAXIMUM_LENGTH	Maximum length.
CHARACTER_OCTET_LENGTH	Same as the `CHARACTER_MAXIMUM_LENGTH` except for multi-byte character sets.
NUMERIC_PRECISION	For numeric types, the precision (number of significant digits) for the column. `NULL` if not a numeric field.
NUMERIC_SCALE	For numeric types, the scale (significant digits to the right of the decimal point). `NULL` if not a numeric field.
DATETIME_PRECISION	Fractional-seconds precision, or `NULL` if not a time data type.
CHARACTER_SET_NAME	Character set if a non-binary string data type, otherwise `NULL`.
COLLATION_NAME	Collation if a non-binary string data type, otherwise `NULL`.
COLUMN_TYPE	Column definition, a MySQL and MariaDB extension.
COLUMN_KEY	Index type. `PRI` for primary key, `UNI` for unique index, `MUL` for multiple index. A MySQL and MariaDB extension.
CREATE_OPTIONS	Extra CREATE TABLE options.This column is available from MariaDB 13.0.
EXTRA	Additional information about a column, for example whether the column is an invisible column, or `WITHOUT SYSTEM VERSIONING` if the table is not a system-versioned table. A MySQL and MariaDB extension.
PRIVILEGES	Which privileges you have for the column. A MySQL and MariaDB extension.
COLUMN_COMMENT	Column comments.
IS_GENERATED	Indicates whether the column value is generated (virtual, or computed). Can be `ALWAYS` or `NEVER`.
GENERATION_EXPRESSION	The expression used for computing the column value in a generated (virtual, or computed) column.
IS_SYSTEM_TIME_PERIOD_START	From MariaDB 11.4.1.
IS_SYSTEM_TIME_PERIOD_END	From MariaDB 11.4.1.

It provides information similar to, but more complete, than SHOW COLUMNS or output from mariadb-show.

Examples

```
SELECT * FROM information_schema.COLUMNS\G
...
*************************** 9. row ***************************
           TABLE_CATALOG: def
            TABLE_SCHEMA: test
              TABLE_NAME: t2
             COLUMN_NAME: j
        ORDINAL_POSITION: 1
          COLUMN_DEFAULT: NULL
             IS_NULLABLE: YES
               DATA_TYPE: longtext
CHARACTER_MAXIMUM_LENGTH: 4294967295
  CHARACTER_OCTET_LENGTH: 4294967295
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: utf8mb4
          COLLATION_NAME: utf8mb4_bin
             COLUMN_TYPE: longtext
              COLUMN_KEY:
                   EXTRA:
              PRIVILEGES: select,insert,update,references
          COLUMN_COMMENT:
            IS_GENERATED: NEVER
   GENERATION_EXPRESSION: NULL
...
```

```
CREATE TABLE t (
  s1 VARCHAR(20) DEFAULT 'ABC',
  s2 VARCHAR(20) DEFAULT (concat('A','B')),
  s3 VARCHAR(20) DEFAULT ("concat('A','B')"),
  s4 VARCHAR(20),
  s5 VARCHAR(20) DEFAULT NULL,
  s6 VARCHAR(20) NOT NULL,
  s7 VARCHAR(20) DEFAULT 'NULL' NULL,
  s8 VARCHAR(20) DEFAULT 'NULL' NOT NULL
);

SELECT
  table_name,
  column_name,
  ordinal_position,
  column_default,
  column_default IS NULL
FROM information_schema.COLUMNS
WHERE table_schema=DATABASE()
AND TABLE_NAME='t';
+------------+-------------+------------------+----------------------+------------------------+
| table_name | column_name | ordinal_position | column_default       | column_default IS NULL |
+------------+-------------+------------------+----------------------+------------------------+
| t          | s1          |                1 | 'ABC'                |                      0 |
| t          | s2          |                2 | concat('A','B')      |                      0 |
| t          | s3          |                3 | 'concat(''A'',''B'')' |                      0 |
| t          | s4          |                4 | NULL                 |                      0 |
| t          | s5          |                5 | NULL                 |                      0 |
| t          | s6          |                6 | NULL                 |                      1 |
| t          | s7          |                7 | 'NULL'               |                      0 |
| t          | s8          |                8 | 'NULL'               |                      0 |
+------------+-------------+------------------+----------------------+------------------------+
```

In the results above, the two single quotes in `concat(''A'',''B'')` indicate an escaped single quote - see string-literals. Note that while mariadb client appears to show the same default value for columns `s5` and `s6`, the first is a 4-character string “NULL”, while the second is the SQL `NULL` value.

Note:

The following statement is available from MariaDB 11.3.

```
CREATE TABLE t(
     x INT,
     start_timestamp TIMESTAMP(6) GENERATED ALWAYS AS ROW START,
     end_timestamp TIMESTAMP(6) GENERATED ALWAYS AS ROW END,
     PERIOD FOR SYSTEM_TIME(start_timestamp, end_timestamp)
) WITH SYSTEM VERSIONING;

SELECT TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION,
  IS_SYSTEM_TIME_PERIOD_START, IS_SYSTEM_TIME_PERIOD_END
  FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='t'\G
*************************** 1. row ***************************
                TABLE_NAME: t
               COLUMN_NAME: x
          ORDINAL_POSITION: 1
IS_SYSTEM_TIME_PERIOD_START: NO
  IS_SYSTEM_TIME_PERIOD_END: NO
*************************** 2. row ***************************
                TABLE_NAME: t
               COLUMN_NAME: start_timestamp
          ORDINAL_POSITION: 2
IS_SYSTEM_TIME_PERIOD_START: YES
  IS_SYSTEM_TIME_PERIOD_END: NO
*************************** 3. row ***************************
                TABLE_NAME: t
               COLUMN_NAME: end_timestamp
          ORDINAL_POSITION: 3
IS_SYSTEM_TIME_PERIOD_START: NO
  IS_SYSTEM_TIME_PERIOD_END: YES
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema DISKS Table

Warning:

The plugin only works on Linux.

Description

The `DISKS` table is created when the DISKS plugin is enabled, and shows metadata about disks on the system.

CURRENT

This plugin requires the FILE privilege.

< 10.4.7 / 10.3.17 / 10.2.26 / 10.1.41

Important:

This plugin does **not** check user privileges. When it is enabled, **any** user can query the `INFORMATION_SCHEMA.DISKS` table and see all the information it provides.

The table contains the following columns:

Column	Description
DISK	Name of the disk itself.
PATH	Mount point of the disk.

Column	Description
TOTAL	Total space in KiB.
USED	Used amount of space in KiB.
AVAILABLE	Amount of space in KiB available to non-root users.

Note that as the amount of space available to root (OS user) may be more that what is available to non-root users, ‘available’ + ‘used’ may be less than ‘total’.

All paths to which a particular disk has been mounted are reported. The rationale is that someone might want to take different action e.g. depending on which disk is relevant for a particular path. This leads to the same disk being reported multiple times.

Example

```
SELECT * FROM information_schema.DISKS;

+-----------+-------+----------+---------+-----------+
| Disk      | Path  | Total    | Used    | Available |
+-----------+-------+----------+---------+-----------+
| /dev/vda1 | /     | 26203116 | 2178424 |  24024692 |
| /dev/vda1 | /boot | 26203116 | 2178424 |  24024692 |
| /dev/vda1 | /etc  | 26203116 | 2178424 |  24024692 |
+-----------+-------+----------+---------+-----------+
```

See Also

- Disks Plugin for details on installing, options
- Plugin Overview for details on managing plugins.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ENABLED_ROLES Table

The Information Schema `ENABLED_ROLES` table shows the enabled roles for the current session.

It contains the following column:

Column	Description
ROLE_NAME	The enabled role name, or NULL.

This table lists all roles that are currently enabled, one role per row — the current role, roles granted to the current role, roles granted to these roles and so on. If no role is set, the row contains a `NULL` value.

The roles that the current user can enable are listed in the APPLICABLE_ROLES Information Schema table.

See also CURRENT_ROLE().

Examples

```
SELECT * FROM information_schema.ENABLED_ROLES;
+-----------+
| ROLE_NAME |
+-----------+
| NULL      |
+-----------+

SET ROLE staff;

SELECT * FROM information_schema.ENABLED_ROLES;
+-----------+
| ROLE_NAME |
+-----------+
| staff     |
+-----------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ENGINES Table

The Information Schema `ENGINES` table displays status information about the server's storage engines.

It contains the following columns:

Column	Description
ENGINE	Name of the storage engine.
SUPPORT	Whether the engine is the default, or is supported or not.
COMMENT	Storage engine comments.
TRANSACTIONS	Whether or not the engine supports transactions.
XA	Whether or not the engine supports XA transactions.
SAVEPOINTS	Whether or not savepoints are supported.

It provides identical information to the SHOW ENGINES statement. Since storage engines are plugins, different information about them is also shown in the information_schema.PLUGINS table and by the SHOW PLUGINS statement.

The table is not a standard Information Schema table, and is a MySQL and MariaDB extension.

Note that both MySQL's InnoDB and Percona's XtraDB replacement are labeled as `InnoDB`. However, if XtraDB is in use, it will be specified in the `COMMENT` field. See XtraDB and InnoDB. The same applies to FederatedX.

Example

```
SELECT * FROM information_schema.ENGINES\G;
*************************** 1. row ***************************
      ENGINE: InnoDB
     SUPPORT: DEFAULT
     COMMENT: Supports transactions, row-level locking, and foreign keys
TRANSACTIONS: YES
          XA: YES
  SAVEPOINTS: YES
*************************** 2. row ***************************
      ENGINE: CSV
     SUPPORT: YES
     COMMENT: CSV storage engine
TRANSACTIONS: NO
          XA: NO
  SAVEPOINTS: NO
*************************** 3. row ***************************
      ENGINE: MyISAM
     SUPPORT: YES
     COMMENT: MyISAM storage engine
TRANSACTIONS: NO
          XA: NO
  SAVEPOINTS: NO
*************************** 4. row ***************************
      ENGINE: BLACKHOLE
     SUPPORT: YES
     COMMENT: /dev/null storage engine (anything you write to it disappears)
TRANSACTIONS: NO
          XA: NO
  SAVEPOINTS: NO
*************************** 5. row ***************************
      ENGINE: FEDERATED
     SUPPORT: YES
     COMMENT: FederatedX pluggable storage engine
TRANSACTIONS: YES
          XA: NO
  SAVEPOINTS: YES
*************************** 6. row ***************************
      ENGINE: MRG_MyISAM
     SUPPORT: YES
     COMMENT: Collection of identical MyISAM tables
TRANSACTIONS: NO
          XA: NO
  SAVEPOINTS: NO
*************************** 7. row ***************************
      ENGINE: ARCHIVE
     SUPPORT: YES
     COMMENT: Archive storage engine
TRANSACTIONS: NO
          XA: NO
  SAVEPOINTS: NO
*************************** 8. row ***************************
      ENGINE: MEMORY
     SUPPORT: YES
     COMMENT: Hash based, stored in memory, useful for temporary tables
TRANSACTIONS: NO
          XA: NO
  SAVEPOINTS: NO
*************************** 9. row ***************************
      ENGINE: PERFORMANCE_SCHEMA
     SUPPORT: YES
     COMMENT: Performance Schema
TRANSACTIONS: NO
          XA: NO
  SAVEPOINTS: NO
*************************** 10. row ***************************
      ENGINE: Aria
     SUPPORT: YES
```

```
     COMMENT: Crash-safe tables with MyISAM heritage
TRANSACTIONS: NO
          XA: NO
  SAVEPOINTS: NO
10 rows in set (0.00 sec)
```

Check which storage engine supports XA transactions:

```
SELECT ENGINE FROM information_schema.ENGINES WHERE XA = 'YES';
+--------+
| ENGINE |
+--------+
| InnoDB |
+--------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema EVENTS Table

The Information Schema `EVENTS` table stores information about Events on the server.

It contains the following columns:

Column	Description
EVENT_CATALOG	Always def.
EVENT_SCHEMA	Database where the event was defined.
EVENT_NAME	Event name.
DEFINER	Event definer.
TIME_ZONE	Time zone used for the event's scheduling and execution, by default SYSTEM.
EVENT_BODY	SQL.
EVENT_DEFINITION	The SQL defining the event.
EVENT_TYPE	Either ONE TIME or RECURRING.
EXECUTE_AT	DATETIME when the event is set to execute, or NULL if recurring.
INTERVAL_VALUE	Numeric interval between event executions for a recurring event, or NULL if not recurring.
INTERVAL_FIELD	Interval unit (e.g., HOUR)
SQL_MODE	The SQL_MODE at the time the event was created.
STARTS	Start DATETIME for a recurring event, NULL if not defined or not recurring.
ENDS	End DATETIME for a recurring event, NULL if not defined or not recurring.
STATUS	One of ENABLED, DISABLED or /SLAVESIDE_DISABLED.
ON_COMPLETION	The ON COMPLETION clause, either PRESERVE or NOT PRESERVE .
CREATED	When the event was created.
LAST_ALTERED	When the event was last changed.
LAST_EXECUTED	When the event was last run.
EVENT_COMMENT	The comment provided in the CREATE EVENT statement, or an empty string if none.
ORIGINATOR	MariaDB server ID on which the event was created.
CHARACTER_SET_CLIENT	character_set_client system variable session value at the time the event was created.
COLLATION_CONNECTION	collation_connection system variable session value at the time the event was created.
DATABASE_COLLATION	Database collation with which the event is linked.

The SHOW EVENTS and SHOW CREATE EVENT statements provide similar information.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema FEEDBACK Table

The Information Schema `FEEDBACK` table is created when the Feedback Plugin is enabled, and contains the complete contents submitted by the plugin.

It contains two columns:

Column	Description
VARIABLE_NAME	Name of the item of information being collected.
VARIABLE_VALUE	Contents of the item of information being collected.

It is possible to disable automatic collection, by setting the feedback_url variable to an empty string, and to submit the contents manually, as follows:

```
$ mysql -e 'SELECT * FROM information_schema.FEEDBACK' > report.txt
```

Then you can send it by opening post in your browser, and uploading your generated `report.txt`. Or you can do it from the command line with (for example):

```
$ curl -F data=@report.txt https://mariadb.org/feedback_plugin/post
```

Manual uploading allows you to be absolutely sure that we receive only the data shown in the `information_schema.FEEDBACK` table and that no private or sensitive information is being sent.

Example

```
SELECT * FROM information_schema.FEEDBACK\G
...
*************************** 906. row ***************************
 VARIABLE_NAME: Uname_sysname
VARIABLE_VALUE: Linux
*************************** 907. row ***************************
 VARIABLE_NAME: Uname_release
VARIABLE_VALUE: 3.13.0-53-generic
*************************** 908. row ***************************
 VARIABLE_NAME: Uname_version
VARIABLE_VALUE: #89-Ubuntu SMP Wed May 20 10:34:39 UTC 2015
*************************** 909. row ***************************
 VARIABLE_NAME: Uname_machine
VARIABLE_VALUE: x86_64
*************************** 910. row ***************************
 VARIABLE_NAME: Uname_distribution
VARIABLE_VALUE: lsb: Ubuntu 14.04.2 LTS
*************************** 911. row ***************************
 VARIABLE_NAME: Collation used latin1_german1_ci
VARIABLE_VALUE: 1
*************************** 912. row ***************************
 VARIABLE_NAME: Collation used latin1_swedish_ci
VARIABLE_VALUE: 18
*************************** 913. row ***************************
 VARIABLE_NAME: Collation used utf8_general_ci
VARIABLE_VALUE: 567
*************************** 914. row ***************************
 VARIABLE_NAME: Collation used latin1_bin
VARIABLE_VALUE: 1
*************************** 915. row ***************************
 VARIABLE_NAME: Collation used binary
VARIABLE_VALUE: 16
*************************** 916. row ***************************
 VARIABLE_NAME: Collation used utf8_bin
VARIABLE_VALUE: 4044
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema FILE_KEY_MANAGEMENT_KEYS Table

> **Note:**
>
> This table is available from MariaDB Enterprise Server 11.8.

The `INFORMATION_SCHEMA.FILE_KEY_MANAGEMENT_KEYS` table lists key ID and latest key version for keys managed by the file_key_management plugin. The table's contents can be queried directly, or viewed using the `SHOW FILE_KEY_MANAGEMENT_KEYS` statement.

You can also can reload the data from the key file into the cache by issuing this statement:

```
FLUSH FILE_KEY_MANAGEMENT_KEYS
```

This functionality is provided by the `file_key_management` plugin.

This Information Schema table allows to query the key numbers and key versions which are currently in the cache (which can be different to the file). It also allows to verify which keys are in use when joining the table with INNODB_TABLESPACES_ENCRYPTION.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema HASHICORP_KEY_MANAGEMENT_CACHE Table

> **Note:**
>
> This table is available from MariaDB 12.3.

The `INFORMATION_SCHEMA.HASHICORP_KEY_MANAGEMENT_CACHE` table lists the Key ID and the latest Key Version for keys managed by the HashiCorp Key Management plugin. The table's contents can be queried directly, or viewed using the `SHOW HASHICORP_KEY_MANAGEMENT_CACHE` statement.

You can reload the data from the Vault server into the cache by issuing this statement:

```
FLUSH HASHICORP_KEY_MANAGEMENT_CACHE
```

This functionality is provided by the HashiCorp Key Management plugin.

This Information Schema table allows you to query the key numbers and key versions which are currently in the cache.

Columns

The table contains the following columns:

Column Name	Description
`KEY_ID`	The Key ID used by the plugin.
`KEY_VERSION`	The latest version of the key currently in the cache.

Required Privileges

Accessing the `INFORMATION_SCHEMA.HASHICORP_KEY_MANAGEMENT_CACHE` table requires the `PROCESS` privilege.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema GEOMETRY_COLUMNS Table

Description

The Information Schema `GEOMETRY_COLUMNS` table provides support for Spatial Reference systems for GIS data.

It contains the following columns:

Column	Type	Null	Description
F_TABLE_CATALOG	VARCHAR(512)	NO	Together with F_TABLE_SCHEMA and F_TABLE_NAME, the fully qualified name of the featured table containing the geometry column.
F_TABLE_SCHEMA	VARCHAR(64)	NO	Together with F_TABLE_CATALOG and F_TABLE_NAME, the fully qualified name of the featured table containing the geometry column.
F_TABLE_NAME	VARCHAR(64)	NO	Together with F_TABLE_CATALOG and F_TABLE_SCHEMA, the fully qualified name of the featured table containing the geometry column.
F_GEOMETRY_COLUMN	VARCHAR(64)	NO	Name of the column in the featured table that is the geometry golumn.
G_TABLE_CATALOG	VARCHAR(512)	NO	
G_TABLE_SCHEMA	VARCHAR(64)	NO	Database name of the table implementing the geometry column.

Column	Type	Null	Description
G_TABLE_NAME	VARCHAR(64)	NO	Table name that is implementing the geometry column.
G_GEOMETRY_COLUMN	VARCHAR(64)	NO	
STORAGE_TYPE	TINYINT(2)	NO	Binary geometry implementation. Always 1 in MariaDB.
GEOMETRY_TYPE	INT(7)	NO	Integer reflecting the type of geometry stored in this column (see table below).
COORD_DIMENSION	TINYINT(2)	NO	Number of dimensions in the spatial reference system. Always 2 in MariaDB.
MAX_PPR	TINYINT(2)	NO	Always 0 in MariaDB.
SRID	SMALLINT(5)	NO	ID of the Spatial Reference System used for the coordinate geometry in this table. It is a foreign key reference to the SPATIAL_REF_SYS table.

Storage_type

The integers in the `storage_type` field match the geometry types as follows:

Integer	Type
0	GEOMETRY
1	POINT
3	LINESTRING
5	POLYGON
7	MULTIPOINT
9	MULTILINESTRING
11	MULTIPOLYGON

Example

```
CREATE TABLE g1(g GEOMETRY(9,4) REF_SYSTEM_ID=101);

SELECT * FROM information_schema.GEOMETRY_COLUMNS\G
*************************** 1. row ***************************
  F_TABLE_CATALOG: def
   F_TABLE_SCHEMA: test
     F_TABLE_NAME: g1
F_GEOMETRY_COLUMN:
  G_TABLE_CATALOG: def
   G_TABLE_SCHEMA: test
     G_TABLE_NAME: g1
G_GEOMETRY_COLUMN: g
     STORAGE_TYPE: 1
    GEOMETRY_TYPE: 0
  COORD_DIMENSION: 2
          MAX_PPR: 0
             SRID: 101
```

See also

- The SPATIAL_REF_SYS table.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema GLOBAL_STATUS and SESSION_STATUS Tables

The Information Schema `GLOBAL_STATUS` and `SESSION_STATUS` tables store a record of all status variables and their global and session values respectively. This is the same information as displayed by the SHOW STATUS commands `SHOW GLOBAL STATUS` and `SHOW SESSION STATUS`.

They contain the following columns:

Column	Description
VARIABLE_NAME	Status variable name.
VARIABLE_VALUE	Global or session value.

Example

```
SELECT * FROM information_schema.GLOBAL_STATUS;
+-----------------------------------------------+--------------------+
| VARIABLE_NAME                                 | VARIABLE_VALUE     |
+-----------------------------------------------+--------------------+
...
| BINLOG_SNAPSHOT_FILE                          | mariadb-bin.000208 |
| BINLOG_SNAPSHOT_POSITION                      | 369                |
...
| THREADS_CONNECTED                             | 1                  |
| THREADS_CREATED                               | 1                  |
| THREADS_RUNNING                               | 1                  |
| UPTIME                                        | 57358              |
| UPTIME_SINCE_FLUSH_STATUS                     | 57358              |
+-----------------------------------------------+--------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema GLOBAL_VARIABLES and SESSION_VARIABLES Tables

The Information Schema `GLOBAL_VARIABLES` and `SESSION_VARIABLES` tables stores a record of all system variables and their global and session values respectively. This is the same information as displayed by the SHOW VARIABLES commands `SHOW GLOBAL VARIABLES` and `SHOW SESSION VARIABLES`.

It contains the following columns:

Column	Description
VARIABLE_NAME	System variable name.
VARIABLE_VALUE	Global or session value.

Example

```
SELECT * FROM information_schema.GLOBAL_VARIABLES ORDER BY VARIABLE_NAME\G
*************************** 1. row ***************************
 VARIABLE_NAME: ARIA_BLOCK_SIZE
VARIABLE_VALUE: 8192
*************************** 2. row ***************************
 VARIABLE_NAME: ARIA_CHECKPOINT_LOG_ACTIVITY
VARIABLE_VALUE: 1048576
*************************** 3. row ***************************
 VARIABLE_NAME: ARIA_CHECKPOINT_INTERVAL
VARIABLE_VALUE: 30
...
*************************** 455. row ***************************
 VARIABLE_NAME: VERSION_COMPILE_MACHINE
VARIABLE_VALUE: x86_64
*************************** 456. row ***************************
 VARIABLE_NAME: VERSION_COMPILE_OS
VARIABLE_VALUE: debian-linux-gnu
*************************** 457. row ***************************
 VARIABLE_NAME: WAIT_TIMEOUT
VARIABLE_VALUE: 600
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INDEX_STATISTICS Table

The Information Schema `INDEX_STATISTICS` table shows statistics on index usage and makes it possible to do such things as locating unused indexes and generating the commands to remove them.

This is part of the User Statistics feature, which is not enabled by default.

It contains the following columns:

Field	Type	Notes
TABLE_SCHEMA	VARCHAR(192)	The schema (database) name.
TABLE_NAME	VARCHAR(192)	The table name.
INDEX_NAME	VARCHAR(192)	The index name (as visible in SHOW CREATE TABLE).
ROWS_READ	BIGINT(21)	The number of rows read from this index.
QUERIES	BIGINT(21)	Incremented for each index the query is part of. This assists one to see how effective the index is. From MariaDB 11.5

Example

```
SELECT * FROM information_schema.INDEX_STATISTICS
WHERE TABLE_NAME = "author";
+--------------+------------+------------+-----------+
| TABLE_SCHEMA | TABLE_NAME | INDEX_NAME | ROWS_READ |
+--------------+------------+------------+-----------+
| books        | author     | by_name    |        15 |
+--------------+------------+------------+-----------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema KEY_CACHES Table

The Information Schema `KEY_CACHES` table shows statistics about the segmented key cache.

It contains the following columns:

Column Name	Description
KEY_CACHE_NAME	The name of the key cache
SEGMENTS	total number of segments (set to NULL for regular key caches)
SEGMENT_NUMBER	segment number (set to NULL for any regular key caches and for rows containing aggregation statistics for segmented key caches)
FULL_SIZE	memory for cache buffers/auxiliary structures
BLOCK_SIZE	size of the blocks
USED_BLOCKS	number of currently used blocks
UNUSED_BLOCKS	number of currently unused blocks
DIRTY_BLOCKS	number of currently dirty blocks
READ_REQUESTS	number of read requests
READS	number of actual reads from files into buffers
WRITE_REQUESTS	number of write requests
WRITES	number of actual writes from buffers into files

Example

```
SELECT * FROM information_schema.KEY_CACHES \G
*********************** 1. row ***********************
KEY_CACHE_NAME: default
SEGMENTS: NULL
SEGMENT_NUMBER: NULL
     FULL_SIZE: 134217728
    BLOCK_SIZE: 1024
   USED_BLOCKS: 36
 UNUSED_BLOCKS: 107146
  DIRTY_BLOCKS: 0
 READ_REQUESTS: 40305
         READS: 21
WRITE_REQUESTS: 19239
        WRITES: 358
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema KEY_COLUMN_USAGE Table

The Information Schema `KEY_COLUMN_USAGE` table shows which key columns have constraints.

It contains the following columns:

Column	Description
CONSTRAINT_CATALOG	Always def.
CONSTRAINT_SCHEMA	Database name of the constraint.
CONSTRAINT_NAME	Name of the constraint (PRIMARY for the primary key).
TABLE_CATALOG	Always #def.
TABLE_SCHEMA	Database name of the column constraint.
TABLE_NAME	Table name of the column constraint.
COLUMN_NAME	Column name of the constraint.

Column	Description
ORDINAL_POSITION	Position of the column within the constraint.
POSITION_IN_UNIQUE_CONSTRAINT	For foreign keys, the position in the unique constraint.
REFERENCED_TABLE_SCHEMA	For foreign keys, the referenced database name.
REFERENCED_TABLE_NAME	For foreign keys, the referenced table name.
REFERENCED_COLUMN_NAME	For foreign keys, the referenced column name.

Example

```
SELECT * FROM information_schema.KEY_COLUMN_USAGE LIMIT 1 \G
*********************** 1. row ***********************
           CONSTRAINT_CATALOG: def
            CONSTRAINT_SCHEMA: my_website
              CONSTRAINT_NAME: PRIMARY
                TABLE_CATALOG: def
                 TABLE_SCHEMA: users
                  COLUMN_NAME: user_id
             ORDINAL_POSITION: 1
POSITION_IN_UNIQUE_CONSTRAINT: NULL
      REFERENCED_TABLE_SCHEMA: NULL
        REFERENCED_TABLE_NAME: NULL
       REFERENCED_COLUMN_NAME: NULL
```

See Also

- Finding Tables Without Primary Keys

This page is licensed: CC BY-SA / Gnu FDL

Information Schema KEY_PERIOD_USAGE Table

> **Note:**
>
> This table is available as of MariaDB 11.4.

The Information Schema `KEY_PERIOD_USAGE` table shows information about Application-Time Periods.

It contains the following columns:

Column	Description
CONSTRAINT_CATALOG	
CONSTRAINT_SCHEMA	
CONSTRAINT_NAME	
TABLE_CATALOG	
TABLE_SCHEMA	
TABLE_NAME	
PERIOD_NAME	

Example

This page is licensed: CC BY-SA / Gnu FDL

Information Schema KEYWORDS Table

> **Note:**
>
> This table is available as of MariaDB 10.6.3.

Description

The Information Schema `KEYWORDS` table contains the list of MariaDB keywords.

It contains a single column:

Column	Description
WORD	Keyword

The table is not a standard Information Schema table, but a MariaDB extension.

Example

```
SELECT * FROM INFORMATION_SCHEMA.KEYWORDS;
+-------------------------------+
| WORD                          |
+-------------------------------+
| &&                            |
| <=                            |
| <>                            |
| !=                            |
| >=                            |
| <<                            |
| >>                            |
| <=>                           |
| ACCESSIBLE                    |
| ACCOUNT                       |
| ACTION                        |
| ADD                           |
| ADMIN                         |
| AFTER                         |
| AGAINST                       |
| AGGREGATE                     |
| ALL                           |
| ALGORITHM                     |
| ALTER                         |
| ALWAYS                        |
| ANALYZE                       |
| AND                           |
| ANY                           |
| AS                            |
| ASC                           |
| ASCII                         |
| ASENSITIVE                    |
| AT                            |
| ATOMIC                        |
| AUTHORS                       |
| AUTO_INCREMENT                |
| AUTOEXTEND_SIZE               |
| AUTO                          |
| AVG                           |
| AVG_ROW_LENGTH                |
| BACKUP                        |
| BEFORE                        |
| BEGIN                         |
| BETWEEN                       |
| BIGINT                        |
| BINARY                        |
| BINLOG                        |
| BIT                           |
| BLOB                          |
| BLOCK                         |
| BODY                          |
| BOOL                          |
| BOOLEAN                       |
| BOTH                          |
| BTREE                         |
| BY                            |
| BYTE                          |
| CACHE                         |
| CALL                          |
| CASCADE                       |
| CASCADED                      |
| CASE                          |
| CATALOG_NAME                  |
| CHAIN                         |
| CHANGE                        |
| CHANGED                       |
| CHAR                          |
| CHARACTER                     |
```

```
| CHARSET                       |
| CHECK                         |
| CHECKPOINT                    |
| CHECKSUM                      |
| CIPHER                        |
| CLASS_ORIGIN                  |
| CLIENT                        |
| CLOB                          |
| CLOSE                         |
| COALESCE                      |
| CODE                          |
| COLLATE                       |
| COLLATION                     |
| COLUMN                        |
| COLUMN_NAME                   |
| COLUMNS                       |
| COLUMN_ADD                    |
| COLUMN_CHECK                  |
| COLUMN_CREATE                 |
| COLUMN_DELETE                 |
| COLUMN_GET                    |
| COMMENT                       |
| COMMIT                        |
| COMMITTED                     |
| COMPACT                       |
| COMPLETION                    |
| COMPRESSED                    |
| CONCURRENT                    |
| CONDITION                     |
| CONNECTION                    |
| CONSISTENT                    |
| CONSTRAINT                    |
| CONSTRAINT_CATALOG            |
| CONSTRAINT_NAME               |
| CONSTRAINT_SCHEMA             |
| CONTAINS                      |
| CONTEXT                       |
| CONTINUE                      |
| CONTRIBUTORS                  |
| CONVERT                       |
| CPU                           |
| CREATE                        |
| CROSS                         |
| CUBE                          |
| CURRENT                       |
| CURRENT_DATE                  |
| CURRENT_POS                   |
| CURRENT_ROLE                  |
| CURRENT_TIME                  |
| CURRENT_TIMESTAMP             |
| CURRENT_USER                  |
| CURSOR                        |
| CURSOR_NAME                   |
| CYCLE                         |
| DATA                          |
| DATABASE                      |
| DATABASES                     |
| DATAFILE                      |
| DATE                          |
| DATETIME                      |
| DAY                           |
| DAY_HOUR                      |
| DAY_MICROSECOND               |
| DAY_MINUTE                    |
| DAY_SECOND                    |
| DEALLOCATE                    |
| DEC                           |
| DECIMAL                       |
| DECLARE                       |
```

```
| DEFAULT                       |
| DEFINER                       |
| DELAYED                       |
| DELAY_KEY_WRITE               |
| DELETE                        |
| DELETE_DOMAIN_ID              |
| DESC                          |
| DESCRIBE                      |
| DES_KEY_FILE                  |
| DETERMINISTIC                 |
| DIAGNOSTICS                   |
| DIRECTORY                     |
| DISABLE                       |
| DISCARD                       |
| DISK                          |
| DISTINCT                      |
| DISTINCTROW                   |
| DIV                           |
| DO                            |
| DOUBLE                        |
| DO_DOMAIN_IDS                 |
| DROP                          |
| DUAL                          |
| DUMPFILE                      |
| DUPLICATE                     |
| DYNAMIC                       |
| EACH                          |
| ELSE                          |
| ELSEIF                        |
| ELSIF                         |
| EMPTY                         |
| ENABLE                        |
| ENCLOSED                      |
| END                           |
| ENDS                          |
| ENGINE                        |
| ENGINES                       |
| ENUM                          |
| ERROR                         |
| ERRORS                        |
| ESCAPE                        |
| ESCAPED                       |
| EVENT                         |
| EVENTS                        |
| EVERY                         |
| EXAMINED                      |
| EXCEPT                        |
| EXCHANGE                      |
| EXCLUDE                       |
| EXECUTE                       |
| EXCEPTION                     |
| EXISTS                        |
| EXIT                          |
| EXPANSION                     |
| EXPIRE                        |
| EXPORT                        |
| EXPLAIN                       |
| EXTENDED                      |
| EXTENT_SIZE                   |
| FALSE                         |
| FAST                          |
| FAULTS                        |
| FEDERATED                     |
| FETCH                         |
| FIELDS                        |
| FILE                          |
| FIRST                         |
| FIXED                         |
| FLOAT                         |
```

```
| FLOAT4                        |
| FLOAT8                        |
| FLUSH                         |
| FOLLOWING                     |
| FOLLOWS                       |
| FOR                           |
| FORCE                         |
| FOREIGN                       |
| FORMAT                        |
| FOUND                         |
| FROM                          |
| FULL                          |
| FULLTEXT                      |
| FUNCTION                      |
| GENERAL                       |
| GENERATED                     |
| GET_FORMAT                    |
| GET                           |
| GLOBAL                        |
| GOTO                          |
| GRANT                         |
| GRANTS                        |
| GROUP                         |
| HANDLER                       |
| HARD                          |
| HASH                          |
| HAVING                        |
| HELP                          |
| HIGH_PRIORITY                 |
| HISTORY                       |
| HOST                          |
| HOSTS                         |
| HOUR                          |
| HOUR_MICROSECOND              |
| HOUR_MINUTE                   |
| HOUR_SECOND                   |
| ID                            |
| IDENTIFIED                    |
| IF                            |
| IGNORE                        |
| IGNORED                       |
| IGNORE_DOMAIN_IDS             |
| IGNORE_SERVER_IDS             |
| IMMEDIATE                     |
| IMPORT                        |
| INTERSECT                     |
| IN                            |
| INCREMENT                     |
| INDEX                         |
| INDEXES                       |
| INFILE                        |
| INITIAL_SIZE                  |
| INNER                         |
| INOUT                         |
| INSENSITIVE                   |
| INSERT                        |
| INSERT_METHOD                 |
| INSTALL                       |
| INT                           |
| INT1                          |
| INT2                          |
| INT3                          |
| INT4                          |
| INT8                          |
| INTEGER                       |
| INTERVAL                      |
| INVISIBLE                     |
| INTO                          |
| IO                            |
```

```
| IO_THREAD                      |
| IPC                            |
| IS                             |
| ISOLATION                      |
| ISOPEN                         |
| ISSUER                         |
| ITERATE                        |
| INVOKER                        |
| JOIN                           |
| JSON                           |
| JSON_TABLE                     |
| KEY                            |
| KEYS                           |
| KEY_BLOCK_SIZE                 |
| KILL                           |
| LANGUAGE                       |
| LAST                           |
| LAST_VALUE                     |
| LASTVAL                        |
| LEADING                        |
| LEAVE                          |
| LEAVES                         |
| LEFT                           |
| LESS                           |
| LEVEL                          |
| LIKE                           |
| LIMIT                          |
| LINEAR                         |
| LINES                          |
| LIST                           |
| LOAD                           |
| LOCAL                          |
| LOCALTIME                      |
| LOCALTIMESTAMP                 |
| LOCK                           |
| LOCKED                         |
| LOCKS                          |
| LOGFILE                        |
| LOGS                           |
| LONG                           |
| LONGBLOB                       |
| LONGTEXT                       |
| LOOP                           |
| LOW_PRIORITY                   |
| MASTER                         |
| MASTER_CONNECT_RETRY           |
| MASTER_DELAY                   |
| MASTER_GTID_POS                |
| MASTER_HOST                    |
| MASTER_LOG_FILE                |
| MASTER_LOG_POS                 |
| MASTER_PASSWORD                |
| MASTER_PORT                    |
| MASTER_SERVER_ID               |
| MASTER_SSL                     |
| MASTER_SSL_CA                  |
| MASTER_SSL_CAPATH              |
| MASTER_SSL_CERT                |
| MASTER_SSL_CIPHER              |
| MASTER_SSL_CRL                 |
| MASTER_SSL_CRLPATH             |
| MASTER_SSL_KEY                 |
| MASTER_SSL_VERIFY_SERVER_CERT  |
| MASTER_USER                    |
| MASTER_USE_GTID                |
| MASTER_HEARTBEAT_PERIOD        |
| MATCH                          |
| MAX_CONNECTIONS_PER_HOUR       |
| MAX_QUERIES_PER_HOUR           |
```

```
| MAX_ROWS                      |
| MAX_SIZE                      |
| MAX_STATEMENT_TIME            |
| MAX_UPDATES_PER_HOUR          |
| MAX_USER_CONNECTIONS          |
| MAXVALUE                      |
| MEDIUM                        |
| MEDIUMBLOB                    |
| MEDIUMINT                     |
| MEDIUMTEXT                    |
| MEMORY                        |
| MERGE                         |
| MESSAGE_TEXT                  |
| MICROSECOND                   |
| MIDDLEINT                     |
| MIGRATE                       |
| MINUS                         |
| MINUTE                        |
| MINUTE_MICROSECOND            |
| MINUTE_SECOND                 |
| MINVALUE                      |
| MIN_ROWS                      |
| MOD                           |
| MODE                          |
| MODIFIES                      |
| MODIFY                        |
| MONITOR                       |
| MONTH                         |
| MUTEX                         |
| MYSQL                         |
| MYSQL_ERRNO                   |
| NAME                          |
| NAMES                         |
| NATIONAL                      |
| NATURAL                       |
| NCHAR                         |
| NESTED                        |
| NEVER                         |
| NEW                           |
| NEXT                          |
| NEXTVAL                       |
| NO                            |
| NOMAXVALUE                    |
| NOMINVALUE                    |
| NOCACHE                       |
| NOCYCLE                       |
| NO_WAIT                       |
| NOWAIT                        |
| NODEGROUP                     |
| NONE                          |
| NOT                           |
| NOTFOUND                      |
| NO_WRITE_TO_BINLOG            |
| NULL                          |
| NUMBER                        |
| NUMERIC                       |
| NVARCHAR                      |
| OF                            |
| OFFSET                        |
| OLD_PASSWORD                  |
| ON                            |
| ONE                           |
| ONLINE                        |
| ONLY                          |
| OPEN                          |
| OPTIMIZE                      |
| OPTIONS                       |
| OPTION                        |
| OPTIONALLY                    |
```

```
| OR                          |
| ORDER                       |
| ORDINALITY                  |
| OTHERS                      |
| OUT                         |
| OUTER                       |
| OUTFILE                     |
| OVER                        |
| OVERLAPS                    |
| OWNER                       |
| PACKAGE                     |
| PACK_KEYS                   |
| PAGE                        |
| PAGE_CHECKSUM               |
| PARSER                      |
| PARSE_VCOL_EXPR             |
| PATH                        |
| PERIOD                      |
| PARTIAL                     |
| PARTITION                   |
| PARTITIONING                |
| PARTITIONS                  |
| PASSWORD                    |
| PERSISTENT                  |
| PHASE                       |
| PLUGIN                      |
| PLUGINS                     |
| PORT                        |
| PORTION                     |
| PRECEDES                    |
| PRECEDING                   |
| PRECISION                   |
| PREPARE                     |
| PRESERVE                    |
| PREV                        |
| PREVIOUS                    |
| PRIMARY                     |
| PRIVILEGES                  |
| PROCEDURE                   |
| PROCESS                     |
| PROCESSLIST                 |
| PROFILE                     |
| PROFILES                    |
| PROXY                       |
| PURGE                       |
| QUARTER                     |
| QUERY                       |
| QUICK                       |
| RAISE                       |
| RANGE                       |
| RAW                         |
| READ                        |
| READ_ONLY                   |
| READ_WRITE                  |
| READS                       |
| REAL                        |
| REBUILD                     |
| RECOVER                     |
| RECURSIVE                   |
| REDO_BUFFER_SIZE            |
| REDOFILE                    |
| REDUNDANT                   |
| REFERENCES                  |
| REGEXP                      |
| RELAY                       |
| RELAYLOG                    |
| RELAY_LOG_FILE              |
| RELAY_LOG_POS               |
| RELAY_THREAD                |
```

```
| RELEASE                       |
| RELOAD                        |
| REMOVE                        |
| RENAME                        |
| REORGANIZE                    |
| REPAIR                        |
| REPEATABLE                    |
| REPLACE                       |
| REPLAY                        |
| REPLICA                       |
| REPLICAS                      |
| REPLICA_POS                   |
| REPLICATION                   |
| REPEAT                        |
| REQUIRE                       |
| RESET                         |
| RESIGNAL                      |
| RESTART                       |
| RESTORE                       |
| RESTRICT                      |
| RESUME                        |
| RETURNED_SQLSTATE             |
| RETURN                        |
| RETURNING                     |
| RETURNS                       |
| REUSE                         |
| REVERSE                       |
| REVOKE                        |
| RIGHT                         |
| RLIKE                         |
| ROLE                          |
| ROLLBACK                      |
| ROLLUP                        |
| ROUTINE                       |
| ROW                           |
| ROWCOUNT                      |
| ROWNUM                        |
| ROWS                          |
| ROWTYPE                       |
| ROW_COUNT                     |
| ROW_FORMAT                    |
| RTREE                         |
| SAVEPOINT                     |
| SCHEDULE                      |
| SCHEMA                        |
| SCHEMA_NAME                   |
| SCHEMAS                       |
| SECOND                        |
| SECOND_MICROSECOND            |
| SECURITY                      |
| SELECT                        |
| SENSITIVE                     |
| SEPARATOR                     |
| SEQUENCE                      |
| SERIAL                        |
| SERIALIZABLE                  |
| SESSION                       |
| SERVER                        |
| SET                           |
| SETVAL                        |
| SHARE                         |
| SHOW                          |
| SHUTDOWN                      |
| SIGNAL                        |
| SIGNED                        |
| SIMPLE                        |
| SKIP                          |
| SLAVE                         |
| SLAVES                        |
```

```
| SLAVE_POS                     |
| SLOW                          |
| SNAPSHOT                      |
| SMALLINT                      |
| SOCKET                        |
| SOFT                          |
| SOME                          |
| SONAME                        |
| SOUNDS                        |
| SOURCE                        |
| STAGE                         |
| STORED                        |
| SPATIAL                       |
| SPECIFIC                      |
| REF_SYSTEM_ID                 |
| SQL                           |
| SQLEXCEPTION                  |
| SQLSTATE                      |
| SQLWARNING                    |
| SQL_BIG_RESULT                |
| SQL_BUFFER_RESULT             |
| SQL_CACHE                     |
| SQL_CALC_FOUND_ROWS           |
| SQL_NO_CACHE                  |
| SQL_SMALL_RESULT              |
| SQL_THREAD                    |
| SQL_TSI_SECOND                |
| SQL_TSI_MINUTE                |
| SQL_TSI_HOUR                  |
| SQL_TSI_DAY                   |
| SQL_TSI_WEEK                  |
| SQL_TSI_MONTH                 |
| SQL_TSI_QUARTER               |
| SQL_TSI_YEAR                  |
| SSL                           |
| START                         |
| STARTING                      |
| STARTS                        |
| STATEMENT                     |
| STATS_AUTO_RECALC             |
| STATS_PERSISTENT              |
| STATS_SAMPLE_PAGES            |
| STATUS                        |
| STOP                          |
| STORAGE                       |
| STRAIGHT_JOIN                 |
| STRING                        |
| SUBCLASS_ORIGIN               |
| SUBJECT                       |
| SUBPARTITION                  |
| SUBPARTITIONS                 |
| SUPER                         |
| SUSPEND                       |
| SWAPS                         |
| SWITCHES                      |
| SYSDATE                       |
| SYSTEM                        |
| SYSTEM_TIME                   |
| TABLE                         |
| TABLE_NAME                    |
| TABLES                        |
| TABLESPACE                    |
| TABLE_CHECKSUM                |
| TEMPORARY                     |
| TEMPTABLE                     |
| TERMINATED                    |
| TEXT                          |
| THAN                          |
| THEN                          |
```

```
| TIES                          |
| TIME                          |
| TIMESTAMP                     |
| TIMESTAMPADD                  |
| TIMESTAMPDIFF                 |
| TINYBLOB                      |
| TINYINT                       |
| TINYTEXT                      |
| TO                            |
| TRAILING                      |
| TRANSACTION                   |
| TRANSACTIONAL                 |
| THREADS                       |
| TRIGGER                       |
| TRIGGERS                      |
| TRUE                          |
| TRUNCATE                      |
| TYPE                          |
| TYPES                         |
| UNBOUNDED                     |
| UNCOMMITTED                   |
| UNDEFINED                     |
| UNDO_BUFFER_SIZE              |
| UNDOFILE                      |
| UNDO                          |
| UNICODE                       |
| UNION                         |
| UNIQUE                        |
| UNKNOWN                       |
| UNLOCK                        |
| UNINSTALL                     |
| UNSIGNED                      |
| UNTIL                         |
| UPDATE                        |
| UPGRADE                       |
| USAGE                         |
| USE                           |
| USER                          |
| USER_RESOURCES                |
| USE_FRM                       |
| USING                         |
| UTC_DATE                      |
| UTC_TIME                      |
| UTC_TIMESTAMP                 |
| VALUE                         |
| VALUES                        |
| VARBINARY                     |
| VARCHAR                       |
| VARCHARACTER                  |
| VARCHAR2                      |
| VARIABLES                     |
| VARYING                       |
| VIA                           |
| VIEW                          |
| VIRTUAL                       |
| VISIBLE                       |
| VERSIONING                    |
| WAIT                          |
| WARNINGS                      |
| WEEK                          |
| WEIGHT_STRING                 |
| WHEN                          |
| WHERE                         |
| WHILE                         |
| WINDOW                        |
| WITH                          |
| WITHIN                        |
| WITHOUT                       |
| WORK                          |
```

```
| WRAPPER                       |
| WRITE                         |
| X509                          |
| XOR                           |
| XA                            |
| XML                           |
| YEAR                          |
| YEAR_MONTH                    |
| ZEROFILL                      |
| ||                            |
+-------------------------------+
694 rows in set (0.000 sec)
```

See Also

- Reserved Words

This page is licensed: CC BY-SA / Gnu FDL

Information Schema LOCALES Table

Description

The Information Schema `LOCALES` table contains a list of all compiled-in locales. It is only available if the LOCALES plugin has been installed.

It contains the following columns:

Column	Description
ID	Row ID.
NAME	Locale name, for example en_GB.
DESCRIPTION	Locale description, for example English - United Kingdom.
MAX_MONTH_NAME_LENGTH	Numeric length of the longest month in the locale
MAX_DAY_NAME_LENGTH	Numeric length of the longest day name in the locale.
DECIMAL_POINT	Decimal point character (some locales use a comma).
THOUSAND_SEP	Thousand's character separator,
ERROR_MESSAGE_LANGUAGE	Error message language.

The table is not a standard Information Schema table, and is a MariaDB extension.

The SHOW LOCALES statement returns a subset of the information.

Example

```
SELECT * FROM information_schema.LOCALES;
+-----+-------+-------------------------------------+----------------------+--------------------+------
---------+--------------+-----------------------+
| ID  | NAME  | DESCRIPTION                         | MAX_MONTH_NAME_LENGTH | MAX_DAY_NAME_LENGTH |
DECIMAL_POINT | THOUSAND_SEP | ERROR_MESSAGE_LANGUAGE |
+-----+-------+-------------------------------------+----------------------+--------------------+------
---------+--------------+-----------------------+
|   0 | en_US | English - United States             |                    9 |                  9 | .
| ,            | english                |
|   1 | en_GB | English - United Kingdom            |                    9 |                  9 | .
| ,            | english                |
|   2 | ja_JP | Japanese - Japan                    |                    3 |                  3 | .
| ,            | japanese               |
|   3 | sv_SE | Swedish - Sweden                    |                    9 |                  7 | ,
|              | swedish                |
|   4 | de_DE | German - Germany                    |                    9 |                 10 | ,
| .            | german                 |
|   5 | fr_FR | French - France                     |                    9 |                  8 | ,
|              | french                 |
|   6 | ar_AE | Arabic - United Arab Emirates       |                    6 |                  8 | .
| ,            | english                |
|   7 | ar_BH | Arabic - Bahrain                    |                    6 |                  8 | .
| ,            | english                |
|   8 | ar_JO | Arabic - Jordan                     |                   12 |                  8 | .
| ,            | english                |
...
| 106 | no_NO | Norwegian - Norway                  |                    9 |                  7 | ,
| .            | norwegian              |
| 107 | sv_FI | Swedish - Finland                   |                    9 |                  7 | ,
|              | swedish                |
| 108 | zh_HK | Chinese - Hong Kong SAR             |                    3 |                  3 | .
| ,            | english                |
| 109 | el_GR | Greek - Greece                      |                   11 |                  9 | ,
| .            | greek                  |
+-----+-------+-------------------------------------+----------------------+--------------------+------
---------+--------------+-----------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema METADATA_LOCK_INFO Table

The Information Schema `METADATA_LOCK_INFO` table is created by the metadata_lock_info plugin. It shows active metadata locks and user locks (the locks acquired with GET_LOCK).

Description

It has the following columns:

Column	Description
THREAD_ID	
LOCK_MODE	One of MDL_INTENTION_EXCLUSIVE, MDL_SHARED, MDL_SHARED_HIGH_PRIO, MDL_SHARED_READ, MDL_SHARED_READ_ONLY, MDL_SHARED_WRITE, MDL_SHARED_NO_WRITE, MDL_SHARED_NO_READ_WRITE, MDL_SHARED_UPGRADABLE or MDL_EXCLUSIVE.
LOCK_DURATION	One of MDL_STATEMENT, MDL_TRANSACTION or MDL_EXPLICIT
LOCK_TYPE	One of Global read lock, Schema metadata lock, Table metadata lock, Stored function metadata lock, Stored procedure metadata lock, Trigger metadata lock, Event metadata lock, Commit lock or User lock.

Column	Description
TABLE_SCHEMA	
TABLE_NAME	

LOCK_MODE Descriptions

The `LOCK_MODE` column can have the following values:

Value	Description
MDL_INTENTION_EXCLUSIVE	An intention exclusive metadata lock (IX). Used only for scoped locks. Owner of this type of lock can acquire upgradable exclusive locks on individual objects. Compatible with other IX locks, but is incompatible with scoped S and X locks. IX lock is taken in SCHEMA namespace when we intend to modify object metadata. Object may refer table, stored procedure, trigger, view/etc.
MDL_SHARED	A shared metadata lock (S). To be used in cases when we are interested in object metadata only and there is no intention to access object data (e.g. for stored routines or during preparing prepared statements). We also mis-use this type of lock for open HANDLERs, since lock acquired by this statement has to be compatible with lock acquired by LOCK TABLES … WRITE statement, i.e. SNRW (We can't get by acquiring S lock at HANDLER … OPEN time and upgrading it to SR lock for HANDLER … READ as it doesn't solve problem with need to abort DML statements which wait on table level lock while having open HANDLER in the same connection). To avoid deadlock which may occur when SNRW lock is being upgraded to X lock for table on which there is an active S lock which is owned by thread which waits in its turn for table-level lock owned by thread performing upgrade we have to use thr_abort_locks_for_thread() facility in such situation. This problem does not arise for locks on stored routines as we don't use SNRW locks for them. It also does not arise when S locks are used during PREPARE calls as table-level locks are not acquired in this case. This lock is taken for global read lock, when caching a stored procedure in memory for the duration of the transaction and for tables used by prepared statements.
MDL_SHARED_HIGH_PRIO	A high priority shared metadata lock. Used for cases when there is no intention to access object data (i.e. data in the table). "High priority" means that, unlike other shared locks, it is granted ignoring pending requests for exclusive locks. Intended for use in cases when we only need to access metadata and not data, e.g. when filling an INFORMATION_SCHEMA table. Since SH lock is compatible with SNRW lock, the connection that holds SH lock should not try to acquire any kind of table-level or row-level lock, as this can lead to a deadlock. Moreover, after acquiring SH lock, the connection should not wait for any other resource, as it might cause starvation for X locks and a potential deadlock during upgrade of SNW or SNRW to X lock (e.g. if the upgrading connection holds the resource that is being waited for).
MDL_SHARED_READ	A shared metadata lock (SR) for cases when there is an intention to read data from table. A connection holding this kind of lock can read table metadata and read table data (after acquiring appropriate table and row-level locks). This means that one can only acquire TL_READ, TL_READ_NO_INSERT, and similar table-level locks on table if one holds SR MDL lock on it. To be used for tables in SELECTs, subqueries, and LOCK TABLE … READ statements.
MDL_SHARED_WRITE	A shared metadata lock (SW) for cases when there is an intention to modify (and not just read) data in the table. A connection holding SW lock can read table metadata and modify or read table data (after acquiring appropriate table and row-level locks). To be used for tables to be modified by INSERT, UPDATE, DELETE statements, but not LOCK TABLE … WRITE or DDL). Also taken by SELECT … FOR UPDATE.
MDL_SHARED_UPGRADABLE	An upgradable shared metadata lock for cases when there is an intention to modify (and not just read) data in the table. Can be upgraded to MDL_SHARED_NO_WRITE and MDL_EXCLUSIVE. A connection holding SU lock can read table metadata and modify or read table data (after acquiring appropriate table and row-level locks). To be used for the first phase of ALTER TABLE.

Value	Description
MDL_SHARED_READ_ONLY	A shared metadata lock for cases when we need to read data from table and block all concurrent modifications to it (for both data and metadata). Used by LOCK TABLES READ statement.
MDL_SHARED_NO_WRITE	An upgradable shared metadata lock which blocks all attempts to update table data, allowing reads. A connection holding this kind of lock can read table metadata and read table data. Can be upgraded to X metadata lock. Note, that since this type of lock is not compatible with SNRW or SW lock types, acquiring appropriate engine-level locks for reading (TL_READ* for MyISAM, shared row locks in InnoDB) should be contention-free. To be used for the first phase of ALTER TABLE, when copying data between tables, to allow concurrent SELECTs from the table, but not UPDATEs.
MDL_SHARED_NO_READ_WRITE	An upgradable shared metadata lock which allows other connections to access table metadata, but not data. It blocks all attempts to read or update table data, while allowing INFORMATION_SCHEMA and SHOW queries. A connection holding this kind of lock can read table metadata modify and read table data. Can be upgraded to X metadata lock. To be used for LOCK TABLES WRITE statement. Not compatible with any other lock type except S and SH.
MDL_EXCLUSIVE	An exclusive metadata lock (X). A connection holding this lock can modify both table's metadata and data. No other type of metadata lock can be granted while this lock is held. To be used for CREATE/DROP/RENAME TABLE statements and for execution of certain phases of other DDL statements.

Examples

First, install the `metadata _lock_info` plugin, if it is not already installed:

```
INSTALL plugin if not exists METADATA_LOCK_INFO soname "metadata_lock_info.so";
```

User lock:

```
SELECT GET_LOCK('abc',1000);
+----------------------+
| GET_LOCK('abc',1000) |
+----------------------+
|                    1 |
+----------------------+

SELECT * FROM information_schema.METADATA_LOCK_INFO;
+-----------+--------------------------+---------------+-----------+--------------+------------+
| THREAD_ID | LOCK_MODE                | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----------+--------------------------+---------------+-----------+--------------+------------+
|        61 | MDL_SHARED_NO_READ_WRITE | MDL_EXPLICIT  | User lock | abc          |            |
+-----------+--------------------------+---------------+-----------+--------------+------------+
```

Table metadata lock:

```
START TRANSACTION;

INSERT INTO t VALUES (1,2);

SELECT * FROM information_schema.METADATA_LOCK_INFO \G
*************************** 1. row ***************************
    THREAD_ID: 4
    LOCK_MODE: MDL_SHARED_WRITE
LOCK_DURATION: MDL_TRANSACTION
    LOCK_TYPE: Table metadata lock
 TABLE_SCHEMA: test
   TABLE_NAME: t
```

```
SELECT * FROM information_schema.METADATA_LOCK_INFO;
+-----------+---------------------------+--------------+---------------------+-----------------+--------
-----+
| THREAD_ID | LOCK_MODE | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----------+---------------------------+--------------+---------------------+-----------------+--------
-----+
| 31 | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT | Global read lock | | |
| 31 | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT | Commit lock | | |
| 31 | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT | Schema metadata lock | dbname | |
| 31 | MDL_SHARED_NO_READ_WRITE | MDL_EXPLICIT | Table metadata lock | dbname | exotics |
+-----------+---------------------------+--------------+---------------------+-----------------+--------
-----+
```

See Also

- metadata locks
- Performance Schema metadata_locks table
- GET_LOCK

This page is licensed: CC BY-SA / Gnu FDL

Information Schema MROONGA_STATS Table

The Information Schema `MROONGA_STATS` table only exists if the Mroonga storage engine is installed, and contains information about its activities.

Column	Description
VERSION	Mroonga version.
rows_written	Number of rows written into Mroonga tables.
rows_read	Number of rows read from all Mroonga tables.

This table always contains 1 row.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema OPTIMIZER_TRACE Table

Description

The Information Schema `OPTIMIZER_TRACE` table contains Optimizer Trace information.

It contains the following columns:

Column	Description
QUERY	Displays the query that was asked to be traced.
TRACE	A JSON document displaying the stats we collected when the query was run.

Column	Description
MISSING_BYTES_BEYOND_MAX_MEM_SIZE	For huge trace, where the trace is truncated due to the optimizer_trace_max_mem_size limit being reached, displays the bytes that are missing in the trace
INSUFFICIENT_PRIVILEGES	Set to 1 if the user running the trace does not have the privileges to see the trace.

Structure:

```
SHOW CREATE TABLE INFORMATION_SCHEMA.OPTIMIZER_TRACE \G
*************************** 1. row ***************************
       Table: OPTIMIZER_TRACE
Create Table: CREATE TEMPORARY TABLE `OPTIMIZER_TRACE` (
  `QUERY` longtext NOT NULL DEFAULT '',
  `TRACE` longtext NOT NULL DEFAULT '',
  `MISSING_BYTES_BEYOND_MAX_MEM_SIZE` int(20) NOT NULL DEFAULT 0,
  `INSUFFICIENT_PRIVILEGES` tinyint(1) NOT NULL DEFAULT 0
) ENGINE=Aria DEFAULT CHARSET=utf8 PAGE_CHECKSUM=0
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema PARAMETERS Table

The Information Schema `PARAMETERS` table stores information about stored procedures and stored functions parameters.

It contains the following columns:

Column	Description
SPECIFIC_CATALOG	Always def.
SPECIFIC_SCHEMA	Database name containing the stored routine parameter.
SPECIFIC_NAME	Stored routine name.
ORDINAL_POSITION	Ordinal position of the parameter, starting at 1. 0 for a function RETURNS clause.
PARAMETER_DEFAULT	Displays default values for stored routine parameters.
PARAMETER_MODE	One of IN, OUT, INOUT or NULL for RETURNS.
PARAMETER_NAME	Name of the parameter, or NULL for RETURNS.
DATA_TYPE	The column's data type.
CHARACTER_MAXIMUM_LENGTH	Maximum length.
CHARACTER_OCTET_LENGTH	Same as the CHARACTER_MAXIMUM_LENGTH except for multi-byte character sets.
NUMERIC_PRECISION	For numeric types, the precision (number of significant digits) for the column. NULL if not a numeric field.
NUMERIC_SCALE	For numeric types, the scale (significant digits to the right of the decimal point). NULL if not a numeric field.
DATETIME_PRECISION	Fractional-seconds precision, or NULL if not a time data type.
CHARACTER_SET_NAME	Character set if a non-binary string data type, otherwise NULL.
COLLATION_NAME	Collation if a non-binary string data type, otherwise NULL.
DTD_IDENTIFIER	Description of the data type.
ROUTINE_TYPE	PROCEDURE or FUNCTION.
PARAMETER_DEFAULT	Default parameter value. From MariaDB 12.2.

Information from this table is similar to that found in the `param_list` column in the mysql.proc table, and the output of the SHOW CREATE PROCEDURE and SHOW CREATE FUNCTION statements.

To obtain information about the routine itself, you can query the Information Schema ROUTINES table.

Example

```
SELECT * FROM information_schema.PARAMETERS
LIMIT 1 \G
********************* 1. row *********************
        SPECIFIC_CATALOG: def
         SPECIFIC_SCHEMA: accounts
           SPECIFIC_NAME: user_counts
        ORDINAL_POSITION: 1
          PARAMETER_MODE: IN
          PARAMETER_NAME: user_order
               DATA_TYPE: varchar
CHARACTER_MAXIMUM_LENGTH: 255
  CHARACTER_OCTET_LENGTH: 765
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: utf8
          COLLATION_NAME: utf8_general_ci
          DTD_IDENTIFIER: varchar(255)
            ROUTINE_TYPE: PROCEDURE
       PARAMETER_DEFAULT: NULL
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema PARTITIONS Table

The Information Schema `PARTITIONS` contains information about table partitions, with each record corresponding to a single partition or subpartition of a partitioned table. Each non-partitioned table also has a record in the `PARTITIONS` table, but most of the values are `NULL`.

It contains the following columns:

Column	Description
TABLE_CATALOG	Always def.
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name containing the partition.
PARTITION_NAME	Partition name.
SUBPARTITION_NAME	Subpartition name, or NULL if not a subpartition.
PARTITION_ORDINAL_POSITION	Order of the partition starting from 1.
SUBPARTITION_ORDINAL_POSITION	Order of the subpartition starting from 1.
PARTITION_METHOD	The partitioning type; one of RANGE, LIST, HASH, LINEAR HASH, KEY or LINEAR KEY.
SUBPARTITION_METHOD	Subpartition type; one of HASH, LINEAR HASH, KEY or LINEAR KEY, or NULL if not a subpartition.

Column	Description
PARTITION_EXPRESSION	Expression used to create the partition by the CREATE TABLE or ALTER TABLE statement.
SUBPARTITION_EXPRESSION	Expression used to create the subpartition by the CREATE TABLE or ALTER TABLE statement, or NULL if not a subpartition.
PARTITION_DESCRIPTION	For a RANGE partition, contains either MAXINTEGER or an integer, as set in the VALUES LESS THAN clause. For a LIST partition, contains a comma-separated list of integers, as set in the VALUES IN. For a SYSTEM_TIME INTERVAL partition, shows a defined upper boundary timestamp for historical values (the last history partition can contain values above the upper boundary). NULL if another type of partition.
TABLE_ROWS	Number of rows in the table (may be an estimate for some storage engines).
AVG_ROW_LENGTH	Average row length, that is DATA_LENGTH divided by TABLE_ROWS
DATA_LENGTH	Total number of bytes stored in all rows of the partition.
MAX_DATA_LENGTH	Maximum bytes that could be stored in the partition.
INDEX_LENGTH	Size in bytes of the partition index file.
DATA_FREE	Unused bytes allocated to the partition.
CREATE_TIME	Time the partition was created
UPDATE_TIME	Time the partition was last modified.
CHECK_TIME	Time the partition was last checked, or NULL for storage engines that don't record this information.
CHECKSUM	Checksum value, or NULL if none.
PARTITION_COMMENT	Partition comment, truncated to 80 characters, or an empty string if no comment.
NODEGROUP	Node group, only used for MySQL Cluster, defaults to 0.
TABLESPACE_NAME	Always default.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema PERIODS Table

> **Note:**
>
> This table is available as of MariaDB 11.4.

The Information Schema `PERIODS` table provides information about Application-Time Periods.

It contains the following columns:

Column	Description
TABLE_CATALOG	Always contains the string 'def'.
TABLE_SCHEMA	Database name.

Column	Description
TABLE_NAME	Table name.
PERIOD	Period name.
START_COLUMN_NAME	Name of the column that starts the period.
END_COLUMN_NAME	Name of the column that ends the period.

Example

```
CREATE OR REPLACE TABLE t1(
 name VARCHAR(50),
 date_1 DATE,
 date_2 DATE,
 PERIOD FOR date_period(date_1, date_2)
);

SELECT * FROM INFORMATION_SCHEMA.PERIODS;
+---------------+--------------+------------+-------------+-------------------+-----------------+
| TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME | PERIOD      | START_COLUMN_NAME | END_COLUMN_NAME |
+---------------+--------------+------------+-------------+-------------------+-----------------+
| def           | test         | t1         | date_period | date_1            | date_2          |
+---------------+--------------+------------+-------------+-------------------+-----------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema PLUGINS Table

The Information Schema `PLUGINS` table contains information about server plugins.

It contains the following columns:

Column	Description
PLUGIN_NAME	Name of the plugin.
PLUGIN_VERSION	Version from the plugin's general type descriptor.
PLUGIN_STATUS	Plugin status, one of ACTIVE, INACTIVE, DISABLED or DELETED.
PLUGIN_TYPE	Plugin type; STORAGE ENGINE, INFORMATION_SCHEMA, AUTHENTICATION, REPLICATION, DAEMON or AUDIT.
PLUGIN_TYPE_VERSION	Version from the plugin's type-specific descriptor.
PLUGIN_LIBRARY	Plugin's shared object file name, located in the directory specified by the plugin_dir system variable, and used by the INSTALL PLUGIN and UNINSTALL PLUGIN statements. NULL if the plugin is complied in and cannot be uninstalled.
PLUGIN_LIBRARY_VERSION	Version from the plugin's API interface.
PLUGIN_AUTHOR	Author of the plugin.
PLUGIN_DESCRIPTION	Description.
PLUGIN_LICENSE	Plugin's licence.
LOAD_OPTION	How the plugin was loaded; one of OFF, ON, FORCE or FORCE_PLUS_PERMANENT. See Installing Plugins.
PLUGIN_MATURITY	Plugin's maturity level; one of Unknown, Experimental, Alpha, Beta,'Gamma, and Stable.

Column	Description
PLUGIN_AUTH_VERSION	Plugin's version as determined by the plugin author. An example would be '0.99 beta 1'.

It provides a superset of the information shown by the SHOW PLUGINS statement. For specific information about storage engines (a particular type of plugins), see the information_schema.ENGINES table and the SHOW ENGINES statement.

This table provides a subset of the Information Schema information_schema.ALL_PLUGINS table, which contains all available plugins, installed or not.

The table is not a standard Information Schema table, and is a MariaDB extension.

Examples

The easiest way to get basic information on plugins is with SHOW PLUGINS:

```
SHOW PLUGINS;

+----------------------------+----------+--------------------+-------------+---------+
| Name                       | Status   | Type               | Library     | License |
+----------------------------+----------+--------------------+-------------+---------+
| binlog                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| mysql_native_password      | ACTIVE   | AUTHENTICATION     | NULL        | GPL     |
| mysql_old_password         | ACTIVE   | AUTHENTICATION     | NULL        | GPL     |
| MRG_MyISAM                 | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| MyISAM                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| CSV                        | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| MEMORY                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| FEDERATED                  | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| PERFORMANCE_SCHEMA         | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| Aria                       | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| InnoDB                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| INNODB_TRX                 | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_LOCKS               | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_LOCK_WAITS          | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_CMP                 | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_CMP_RESET           | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_CMPMEM              | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_CMPMEM_RESET        | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_BUFFER_PAGE         | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_BUFFER_PAGE_LRU     | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_BUFFER_POOL_STATS   | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_METRICS             | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_FT_DEFAULT_STOPWORD | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_FT_INSERTED         | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_FT_DELETED          | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_FT_BEING_DELETED    | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_FT_CONFIG           | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_FT_INDEX_CACHE      | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_FT_INDEX_TABLE      | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_SYS_TABLES          | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_SYS_TABLESTATS      | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_SYS_INDEXES         | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_SYS_COLUMNS         | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_SYS_FIELDS          | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_SYS_FOREIGN         | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| INNODB_SYS_FOREIGN_COLS    | ACTIVE   | INFORMATION SCHEMA | NULL        | GPL     |
| SPHINX                     | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| ARCHIVE                    | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| BLACKHOLE                  | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| FEEDBACK                   | DISABLED | INFORMATION SCHEMA | NULL        | GPL     |
| partition                  | ACTIVE   | STORAGE ENGINE     | NULL        | GPL     |
| pam                        | ACTIVE   | AUTHENTICATION     | auth_pam.so | GPL     |
+----------------------------+----------+--------------------+-------------+---------+
```

The equivalent SELECT query would be:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS,
PLUGIN_TYPE, PLUGIN_LIBRARY, PLUGIN_LICENSE
FROM INFORMATION_SCHEMA.PLUGINS;
```

Other SELECT queries can be used to see additional information. For example:

```
SELECT PLUGIN_NAME, PLUGIN_DESCRIPTION,
PLUGIN_MATURITY, PLUGIN_AUTH_VERSION
FROM INFORMATION_SCHEMA.PLUGINS
WHERE PLUGIN_TYPE='STORAGE ENGINE'
ORDER BY PLUGIN_MATURITY \G

*************************** 1. row ***************************
        PLUGIN_NAME: FEDERATED
 PLUGIN_DESCRIPTION: FederatedX pluggable storage engine
    PLUGIN_MATURITY: Beta
PLUGIN_AUTH_VERSION: 2.1
*************************** 2. row ***************************
        PLUGIN_NAME: Aria
 PLUGIN_DESCRIPTION: Crash-safe tables with MyISAM heritage
    PLUGIN_MATURITY: Gamma
PLUGIN_AUTH_VERSION: 1.5
*************************** 3. row ***************************
        PLUGIN_NAME: PERFORMANCE_SCHEMA
 PLUGIN_DESCRIPTION: Performance Schema
    PLUGIN_MATURITY: Gamma
PLUGIN_AUTH_VERSION: 0.1
*************************** 4. row ***************************
        PLUGIN_NAME: binlog
 PLUGIN_DESCRIPTION: This is a pseudo storage engine to represent the binlog in a transaction
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
*************************** 5. row ***************************
        PLUGIN_NAME: MEMORY
 PLUGIN_DESCRIPTION: Hash based, stored in memory, useful for temporary tables
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
*************************** 6. row ***************************
        PLUGIN_NAME: MyISAM
 PLUGIN_DESCRIPTION: MyISAM storage engine
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
*************************** 7. row ***************************
        PLUGIN_NAME: MRG_MyISAM
 PLUGIN_DESCRIPTION: Collection of identical MyISAM tables
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
*************************** 8. row ***************************
        PLUGIN_NAME: CSV
 PLUGIN_DESCRIPTION: CSV storage engine
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
*************************** 9. row ***************************
        PLUGIN_NAME: InnoDB
 PLUGIN_DESCRIPTION: Supports transactions, row-level locking, and foreign keys
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.2.5
*************************** 10. row ***************************
        PLUGIN_NAME: BLACKHOLE
 PLUGIN_DESCRIPTION: /dev/null storage engine (anything you write to it disappears)
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
*************************** 11. row ***************************
        PLUGIN_NAME: ARCHIVE
 PLUGIN_DESCRIPTION: Archive storage engine
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
*************************** 12. row ***************************
        PLUGIN_NAME: partition
 PLUGIN_DESCRIPTION: Partition Storage Engine Helper
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
```

Show authentication plugins:

```
SELECT PLUGIN_NAME, LOAD_OPTION
FROM INFORMATION_SCHEMA.PLUGINS
WHERE PLUGIN_TYPE LIKE 'authentication' \G

*************************** 1. row ***************************
PLUGIN_NAME: mysql_native_password
LOAD_OPTION: FORCE
*************************** 2. row ***************************
PLUGIN_NAME: mysql_old_password
LOAD_OPTION: FORCE
```

See Also

- List of Plugins
- Plugin Overview
- SHOW PLUGINS
- INSTALL PLUGIN
- INSTALL SONAME
- UNINSTALL PLUGIN
- UNINSTALL SONAME

This page is licensed: CC BY-SA / Gnu FDL

Information Schema PROCESSLIST Table

The Information Schema `PROCESSLIST` table contains information about running threads.

Similar information can also be returned with the SHOW [FULL] PROCESSLIST statement, or the mariadb-admin processlist command.

It contains the following columns:

Column	Description
ID	Connection identifier.
USER	MariaDB User.
HOST	The hostname from which this client is connected.For Unix socket connections, this is `localhost`. For TCP/IP connections, the TCP port is appended (for instance, `192.168.1.17:58061` or `other-host.company.com:58061`). For the system user, this column is blank (`''`).
DB	Default database, or `NULL` if none.
COMMAND	Type of command running, corresponding to the Com_ status variables. See Thread Command Values.
TIME	Seconds that the thread has spent on the current `COMMAND` so far.
STATE	Current state of the thread. See Thread States.
INFO	Statement the thread is executing, or `NULL` if none.
TIME_MS	Time in milliseconds with microsecond precision that the thread has spent on the current `COMMAND` so far (see more).
STAGE	The stage the process is currently in.
MAX_STAGE	The maximum number of stages.
PROGRESS	The progress of the process within the current stage (0-100%).
MEMORY_USED	Memory in bytes used by the thread.

Column	Description
MAX_MEMORY_USED	Maximum memory in bytes used by the thread.
EXAMINED_ROWS	Rows examined by the thread. Only updated by `UPDATE`, `DELETE`, and similar statements. For `SELECT` and other statements, the value remains zero.
SENT_ROWS	Number of rows sent by the statement being executed. From MariaDB 11.3.0.
QUERY_ID	Query ID.
INFO_BINARY	Binary data information
TID	Thread ID (MDEV-6756)
TMP_SPACE_USED	See Limiting Size of Created Disk Temporary Files and Tables Overview. From MariaDB 11.5.

Note that as a difference to MySQL, in MariaDB the `TIME` column (and also the `TIME_MS` column) are not affected by any setting of @TIMESTAMP. This means that it can be reliably used also for threads that change `@TIMESTAMP` (such as the replication SQL thread). See also MySQL Bug #22047.

As a consequence of this, the `TIME` column of `SHOW FULL PROCESSLIST` and `INFORMATION_SCHEMA.PROCESSLIST` cannot be used to determine if a slave is lagging behind. For this, use instead the `Seconds_Behind_Master` column in the output of SHOW REPLICA STATUS.

Note that the `PROGRESS` field from the information schema, and the `PROGRESS` field from `SHOW PROCESSLIST` display different results. `SHOW PROCESSLIST` shows the total progress, while the information schema shows the progress for the current stage only. To retrieve a similar “total” Progress value from `information_schema.PROCESSLIST` as the one from `SHOW PROCESSLIST`, issue this statement:

```
SELECT CASE WHEN Max_Stage < 2 THEN Progress ELSE (Stage-1)/Max_Stage*100+Progress/Max_Stage END
  AS Progress FROM INFORMATION_SCHEMA.PROCESSLIST;
```

Example

```
SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST\G
*************************** 1. row ***************************
           ID: 9
         USER: msandbox
         HOST: localhost
           DB: NULL
      COMMAND: Query
         TIME: 0
        STATE: Filling schema table
         INFO: SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST
      TIME_MS: 0.351
        STAGE: 0
    MAX_STAGE: 0
     PROGRESS: 0.000
  MEMORY_USED: 85392
EXAMINED_ROWS: 0
     QUERY_ID: 15
  INFO_BINARY: SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST
          TID: 11838
*************************** 2. row ***************************
           ID: 5
         USER: system user
         HOST:
           DB: NULL
      COMMAND: Daemon
         TIME: 0
        STATE: InnoDB shutdown handler
         INFO: NULL
      TIME_MS: 0.000
        STAGE: 0
    MAX_STAGE: 0
     PROGRESS: 0.000
  MEMORY_USED: 24160
EXAMINED_ROWS: 0
     QUERY_ID: 0
  INFO_BINARY: NULL
          TID: 3856
...
```

See Also

- TIME_MS column in Information Schema SHOW PROCESSLIST

This page is licensed: CC BY-SA / Gnu FDL

Information Schema PROFILING Table

The Information Schema `PROFILING` table contains information about statement resource usage. Profiling information is only recorded if the profiling session variable is set to `1`.

It contains the following columns:

Column Name	Description
QUERY_ID	Query_ID.
SEQ	Sequence number showing the display order for rows with the same QUERY_ID.
STATE	Profiling state.
DURATION	Time in seconds that the statement has been in the current state.
CPU_USER	User CPU usage in seconds.
CPU_SYSTEM	System CPU usage in seconds.
CONTEXT_VOLUNTARY	Number of voluntary context switches.

Column Name	Description
CONTEXT_INVOLUNTARY	Number of involuntary context switches.
BLOCK_OPS_IN	Number of block input operations.
BLOCK_OPS_OUT	Number of block output operations.
MESSAGES_SENT	Number of communications sent.
MESSAGES_RECEIVED	Number of communications received.
PAGE_FAULTS_MAJOR	Number of major page faults.
PAGE_FAULTS_MINOR	Number of minor page faults.
SWAPS	Number of swaps.
SOURCE_FUNCTION	Function in the source code executed by the profiled state.
SOURCE_FILE	File in the source code executed by the profiled state.
SOURCE_LINE	Line in the source code executed by the profiled state.

It contains similar information to the SHOW PROFILE and SHOW PROFILES statements.

Profiling is enabled per session. When a session ends, its profiling information is lost.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema QUERY_CACHE_INFO Table

Description

> **Note:**
>
> This table is not a standard Information Schema table, but a MariaDB extension.

The `QUERY_CACHE_INFO` table is created by the QUERY_CACHE_INFO plugin, and allows you to see the contents of the query cache. It creates a table in the information_schema database that shows all queries that are in the cache. You must have the `PROCESS` privilege (see GRANT) to use this table.

It contains the following columns:

Column	Description
STATEMENT_SCHEMA	Database used when query was included
STATEMENT_TEXT	Query text
RESULT_BLOCKS_COUNT	Number of result blocks
RESULT_BLOCKS_SIZE	Size in bytes of result blocks
RESULT_BLOCKS_SIZE_USED	Size in bytes of used blocks
LIMIT	
MAX_SORT_LENGTH	
GROUP_CONCAT_MAX_LENGTH	
CHARACTER_SET_CLIENT	
CHARACTER_SET_RESULT	
COLLATION	
TIMEZONE	
DEFAULT_WEEK_FORMAT	

Column	Description
DIV_PRECISION_INCREMENT	
SQL_MODE	
LC_TIME_NAMES	
CLIENT_LONG_FLAG	
CLIENT_PROTOCOL_41	
PROTOCOL_TYPE	
MORE_RESULTS_EXISTS	
IN_TRANS	
AUTOCOMMIT	
PACKET_NUMBER	
HITS	Incremented each time the query cache is hit.

For example:

```
SELECT * FROM information_schema.QUERY_CACHE_INFO;
+------------------+-----------------+---------------------+-------------------+-----------------------
-+
| STATEMENT_SCHEMA | STATEMENT_TEXT  | RESULT_BLOCKS_COUNT | RESULT_BLOCKS_SIZE | RESULT_BLOCKS_SIZE_USED
|
+------------------+-----------------+---------------------+-------------------+-----------------------
-+
...
| test             | SELECT * FROM a |                   1 |               512 |                    143
|
| test             | select * FROM a |                   1 |               512 |                    143
|
...
+------------------+-----------------+---------------------+-------------------+-----------------------
-
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema QUERY_RESPONSE_TIME Table

Description

The Information Schema `QUERY_RESPONSE_TIME` table contains information about queries that take a long time to execute . It is only available if the QUERY_RESPONSE_TIME plugin has been installed.

It contains the following columns:

Column	Description
TIME	Time interval
COUNT	Count of queries falling into the time interval
TOTAL	Total execution time of all queries for this interval

See QUERY_RESPONSE_TIME plugin for a full description.

The table is not a standard Information Schema table, and is a MariaDB extension.

SHOW QUERY_RESPONSE_TIME is available as an alternative for retrieving the data.

Example

```
SELECT * FROM information_schema.QUERY_RESPONSE_TIME;
+----------------+-------+----------------+
| TIME           | COUNT | TOTAL          |
+----------------+-------+----------------+
|       0.000001 |     0 |       0.000000 |
|       0.000010 |    17 |       0.000094 |
|       0.000100 |  4301 |       0.236555 |
|       0.001000 |  1499 |       0.824450 |
|       0.010000 | 14851 |      81.680502 |
|       0.100000 |  8066 |     443.635693 |
|       1.000000 |     0 |       0.000000 |
|      10.000000 |     0 |       0.000000 |
|     100.000000 |     1 |      55.937094 |
|    1000.000000 |     0 |       0.000000 |
|   10000.000000 |     0 |       0.000000 |
|  100000.000000 |     0 |       0.000000 |
| 1000000.000000 |     0 |       0.000000 |
| TOO LONG       |     0 | TOO LONG       |
+----------------+-------+----------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema REFERENTIAL_CONSTRAINTS Table

The Information Schema `REFERENTIAL_CONSTRAINTS` table contains information about foreign keys. The single columns are listed in the KEY_COLUMN_USAGE table.

It has the following columns:

Column	Description
CONSTRAINT_CATALOG	Always def.
CONSTRAINT_SCHEMA	Database name, together with CONSTRAINT_NAME identifies the foreign key.
CONSTRAINT_NAME	Foreign key name, together with CONSTRAINT_SCHEMA identifies the foreign key.
UNIQUE_CONSTRAINT_CATALOG	Always def.
UNIQUE_CONSTRAINT_SCHEMA	Database name, together with UNIQUE_CONSTRAINT_NAME and REFERENCED_TABLE_NAME identifies the referenced key.
UNIQUE_CONSTRAINT_NAME	Referenced key name, together with UNIQUE_CONSTRAINT_SCHEMA and REFERENCED_TABLE_NAME identifies the referenced key.
MATCH_OPTION	Always NONE.
UPDATE_RULE	The Update Rule; one of CASCADE, SET NULL, SET DEFAULT, RESTRICT, NO ACTION.
DELETE_RULE	The Delete Rule; one of CASCADE, SET NULL, SET DEFAULT, RESTRICT, NO ACTION.
TABLE_NAME	Table name from the TABLE_CONSTRAINTS table.
REFERENCED_TABLE_NAME	Referenced key table name, together with UNIQUE_CONSTRAINT_SCHEMA and UNIQUE_CONSTRAINT_NAME identifies the referenced key.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROUTINES Table

The Information Schema `ROUTINES` table stores information about stored procedures and stored functions.

It contains the following columns:

Column	Description
SPECIFIC_NAME	
ROUTINE_CATALOG	Always def.
ROUTINE_SCHEMA	Database name associated with the routine.
ROUTINE_NAME	Name of the routine.
ROUTINE_TYPE	Whether the routine is a PROCEDURE or a FUNCTION.
DATA_TYPE	The return value's data type (for stored functions).
CHARACTER_MAXIMUM_LENGTH	Maximum length.
CHARACTER_OCTET_LENGTH	Same as the CHARACTER_MAXIMUM_LENGTH except for multi-byte character sets.
NUMERIC_PRECISION	For numeric types, the precision (number of significant digits) for the column. NULL if not a numeric field.
NUMERIC_SCALE	For numeric types, the scale (significant digits to the right of the decimal point). NULL if not a numeric field.
DATETIME_PRECISION	Fractional-seconds precision, or NULL if not a time data type.
CHARACTER_SET_NAME	Character set if a non-binary string data type, otherwise NULL.
COLLATION_NAME	Collation if a non-binary string data type, otherwise NULL.
DATA_TYPE	The column's data type.
ROUTINE_BODY	Always SQL.
ROUTINE_DEFINITION	Definition of the routine.
EXTERNAL_NAME	Always NULL.
EXTERNAL_LANGUAGE	Always SQL.
PARAMETER_STYLE	Always SQL.
IS_DETERMINISTIC	Whether the routine is deterministic (can produce only one result for a given list of parameters) or not.
SQL_DATA_ACCESS	One of READS SQL DATA, MODIFIES SQL DATA, CONTAINS SQL, or NO SQL.
SQL_PATH	Always NULL.
SECURITY_TYPE	INVOKER or DEFINER. Indicates which user's privileges apply to this routine.
CREATED	Date and time the routine was created.
LAST_ALTERED	Date and time the routine was last changed.
SQL_MODE	The SQL_MODE at the time the routine was created.
ROUTINE_COMMENT	Comment associated with the routine.
DEFINER	If the SECURITY_TYPE is DEFINER, this value indicates which user defined this routine.
CHARACTER_SET_CLIENT	The character set used by the client that created the routine.
COLLATION_CONNECTION	The collation (and character set) used by the connection that created the routine.
DATABASE_COLLATION	The default collation (and character set) for the database, at the time the routine was created.

It provides information similar to, but more complete, than the SHOW PROCEDURE STATUS and SHOW FUNCTION STATUS statements.

For information about the parameters accepted by the routine, you can query the information_schema.PARAMETERS table.

See also

- Stored Function Overview

- Stored Procedure Overview

This page is licensed: CC BY-SA / Gnu FDL

Information Schema SCHEMA_PRIVILEGES Table

The Information Schema `SCHEMA_PRIVILEGES` table contains information about database privileges.

It contains the following columns:

Column	Description
GRANTEE	Account granted the privilege in the format user_name@host_name.
TABLE_CATALOG	Always def
TABLE_SCHEMA	Database name.
PRIVILEGE_TYPE	The granted privilege.
IS_GRANTABLE	Whether the privilege can be granted.

The same information in a different format can be found in the mysql.db table.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema SCHEMATA Table

The Information Schema `SCHEMATA` table stores information about databases on the server.

It contains the following columns:

Column	Description
CATALOG_NAME	Always def.
SCHEMA_NAME	Database name.
DEFAULT_CHARACTER_SET_NAME	Default character set for the database.
DEFAULT_COLLATION_NAME	Default collation.
SQL_PATH	Always NULL.
SCHEMA_COMMENT	Database comment. From MariaDB 10.5.0.

Example

CURRENT

```
SELECT * FROM INFORMATION_SCHEMA.SCHEMATA\G
...
*************************** 2. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: presentations
DEFAULT_CHARACTER_SET_NAME: latin1
    DEFAULT_COLLATION_NAME: latin1_swedish_ci
                  SQL_PATH: NULL
            SCHEMA_COMMENT: Presentations for conferences
...
```

< 10.5.0

```
SELECT * FROM INFORMATION_SCHEMA.SCHEMATA\G
*************************** 1. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: information_schema
DEFAULT_CHARACTER_SET_NAME: utf8
    DEFAULT_COLLATION_NAME: utf8_general_ci
                  SQL_PATH: NULL
*************************** 2. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: mysql
DEFAULT_CHARACTER_SET_NAME: latin1
    DEFAULT_COLLATION_NAME: latin1_swedish_ci
                  SQL_PATH: NULL
*************************** 3. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: performance_schema
DEFAULT_CHARACTER_SET_NAME: utf8
    DEFAULT_COLLATION_NAME: utf8_general_ci
                  SQL_PATH: NULL
*************************** 4. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: test
DEFAULT_CHARACTER_SET_NAME: latin1
    DEFAULT_COLLATION_NAME: latin1_swedish_ci
                  SQL_PATH: NULL
...
```

See Also

- CREATE DATABASE
- ALTER DATABASE
- DROP DATABASE
- SHOW CREATE DATABASE
- SHOW DATABASES
- Character Sets and Collations

This page is licensed: CC BY-SA / Gnu FDL

Information Schema SEQUENCES Table

> **Note:**
>
> This table is available as of MariaDB 11.5.

The Information Schema `SEQUENCES` table stores information about sequences on the server.

It contains the following columns. See CREATE SEQUENCE for details.

Column	Description
SEQUENCE_CATALOG	Catalog name
SEQUENCE_SCHEMA	Database name
SEQUENCE_NAME	Sequence name
DATA_TYPE	Data type, for example BIGINT or TINYINT UNSIGNED
NUMERIC_PRECISION	
NUMERIC_PRECISION_RADIX	
NUMERIC_SCALE	

Column	Description
START_VALUE	
MINIMUM_VALUE	
MAXIMUM_VALUE	
INCREMENT	
CYCLE_OPTION	1 if the CYCLE OPTION is set, otherwise 0

Example

```
CREATE SEQUENCE s START WITH 100 INCREMENT BY 10;

SELECT * FROM INFORMATION_SCHEMA.SEQUENCES\G
*************************** 1. row ***************************
       SEQUENCE_CATALOG: def
        SEQUENCE_SCHEMA: test
          SEQUENCE_NAME: s
              DATA_TYPE: bigint
      NUMERIC_PRECISION: 64
NUMERIC_PRECISION_RADIX: 2
          NUMERIC_SCALE: 0
            START_VALUE: 100
          MINIMUM_VALUE: 1
          MAXIMUM_VALUE: 9223372036854775806
              INCREMENT: 10
           CYCLE_OPTION: 0
```

See Also

- Sequence Overview
- CREATE SEQUENCE
- ALTER SEQUENCE
- DROP SEQUENCE
- NEXT VALUE FOR
- PREVIOUS VALUE FOR
- SETVAL(). Set next value for the sequence.
- AUTO INCREMENT
- Sequence Storage Engine

This page is licensed: CC BY-SA / Gnu FDL

Information Schema SLAVE_STATUS Table

> **Note:**
>
> This table is available as of MariaDB 11.6.

The Information Schema `SLAVE_STATUS` table contains replica status information, similar to SHOW REPLICA STATUS.

See SHOW REPLICA STATUS for column descriptions.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema SPATIAL_REF_SYS Table

Description

The Information Schema `SPATIAL_REF_SYS` table stores information on each spatial reference system used in the database.

It contains the following columns:

Column	Type	Null	Description
SRID	smallint(5)	NO	An integer value that uniquely identifies each Spatial Reference System within a database.
AUTH_NAME	varchar(512)	NO	The name of the standard or standards body that is being cited for this reference system.
AUTH_SRID	smallint(5)	NO	The numeric ID of the coordinate system in the above authority's catalog.
SRTEXT	varchar(2048)	NO	The Well-known Text Representation of the Spatial Reference System.

Note: See MDEV-7540.

See Also

- information_schema.GEOMETRY_COLUMNS table.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema SPIDER_ALLOC_MEM Table

The Information Schema `SPIDER_ALLOC_MEM` table is installed along with the Spider storage engine. It shows information about Spider's memory usage.

It contains the following columns:

Column	Description
ID	
FUNC_NAME	
FILE_NAME	
LINE_NO	
TOTAL_ALLOC_MEM	
CURRENT_ALLOC_MEM	
ALLOC_MEM_COUNT	
FREE_MEM_COUNT	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema SQL_FUNCTIONS Table

> **Note:**
>
> This table is available as of MariaDB 10.6.3 / 10.5.12 / 10.4.21 / 10.3.31 / 10.2.40.

Description

The Information Schema `SQL_FUNCTIONS` table contains the list of MariaDB functions.

It contains a single column:

Column	Description
FUNCTION	Function name

The table is not a standard Information Schema table, and is a MariaDB extension.

Example

```
SELECT * FROM INFORMATION_SCHEMA.SQL_FUNCTIONS;
+---------------------------+
| FUNCTION                  |
+---------------------------+
| ADDDATE                   |
| ADD_MONTHS                |
| BIT_AND                   |
| BIT_OR                    |
| BIT_XOR                   |
| CAST                      |
| COUNT                     |
| CUME_DIST                 |
| CURDATE                   |
| CURTIME                   |
| DATE_ADD                  |
| DATE_SUB                  |
| DATE_FORMAT               |
| DECODE                    |
| DENSE_RANK                |
| EXTRACT                   |
| FIRST_VALUE               |
| GROUP_CONCAT              |
| JSON_ARRAYAGG             |
| JSON_OBJECTAGG            |
| LAG                       |
| LEAD                      |
| MAX                       |
| MEDIAN                    |
| MID                       |
| MIN                       |
| NOW                       |
| NTH_VALUE                 |
| NTILE                     |
| POSITION                  |
| PERCENT_RANK              |
| PERCENTILE_CONT           |
| PERCENTILE_DISC           |
| RANK                      |
| ROW_NUMBER                |
| SESSION_USER              |
| STD                       |
| STDDEV                    |
| STDDEV_POP                |
| STDDEV_SAMP               |
| SUBDATE                   |
| SUBSTR                    |
| SUBSTRING                 |
| SUM                       |
| SYSTEM_USER               |
| TRIM                      |
| TRIM_ORACLE               |
| VARIANCE                  |
| VAR_POP                   |
| VAR_SAMP                  |
| ABS                       |
| ACOS                      |
| ADDTIME                   |
| AES_DECRYPT               |
| AES_ENCRYPT               |
| ASIN                      |
| ATAN                      |
| ATAN2                     |
| BENCHMARK                 |
| BIN                       |
| BINLOG_GTID_POS           |
| BIT_COUNT                 |
| BIT_LENGTH                |
```

```
| CEIL                       |
| CEILING                    |
| CHARACTER_LENGTH           |
| CHAR_LENGTH                |
| CHR                        |
| COERCIBILITY               |
| COLUMN_CHECK               |
| COLUMN_EXISTS              |
| COLUMN_LIST                |
| COLUMN_JSON                |
| COMPRESS                   |
| CONCAT                     |
| CONCAT_OPERATOR_ORACLE     |
| CONCAT_WS                  |
| CONNECTION_ID              |
| CONV                       |
| CONVERT_TZ                 |
| COS                        |
| COT                        |
| CRC32                      |
| DATEDIFF                   |
| DAYNAME                    |
| DAYOFMONTH                 |
| DAYOFWEEK                  |
| DAYOFYEAR                  |
| DEGREES                    |
| DECODE_HISTOGRAM           |
| DECODE_ORACLE              |
| DES_DECRYPT                |
| DES_ENCRYPT                |
| ELT                        |
| ENCODE                     |
| ENCRYPT                    |
| EXP                        |
| EXPORT_SET                 |
| EXTRACTVALUE               |
| FIELD                      |
| FIND_IN_SET                |
| FLOOR                      |
| FORMAT                     |
| FOUND_ROWS                 |
| FROM_BASE64                |
| FROM_DAYS                  |
| FROM_UNIXTIME              |
| GET_LOCK                   |
| GREATEST                   |
| HEX                        |
| IFNULL                     |
| INSTR                      |
| ISNULL                     |
| IS_FREE_LOCK               |
| IS_USED_LOCK               |
| JSON_ARRAY                 |
| JSON_ARRAY_APPEND          |
| JSON_ARRAY_INSERT          |
| JSON_COMPACT               |
| JSON_CONTAINS              |
| JSON_CONTAINS_PATH         |
| JSON_DEPTH                 |
| JSON_DETAILED              |
| JSON_EXISTS                |
| JSON_EXTRACT               |
| JSON_INSERT                |
| JSON_KEYS                  |
| JSON_LENGTH                |
| JSON_LOOSE                 |
| JSON_MERGE                 |
| JSON_MERGE_PATCH           |
| JSON_MERGE_PRESERVE        |
```

```
| JSON_QUERY                |
| JSON_QUOTE                |
| JSON_OBJECT               |
| JSON_REMOVE               |
| JSON_REPLACE              |
| JSON_SET                  |
| JSON_SEARCH               |
| JSON_TYPE                 |
| JSON_UNQUOTE              |
| JSON_VALID                |
| JSON_VALUE                |
| LAST_DAY                  |
| LAST_INSERT_ID            |
| LCASE                     |
| LEAST                     |
| LENGTH                    |
| LENGTHB                   |
| LN                        |
| LOAD_FILE                 |
| LOCATE                    |
| LOG                       |
| LOG10                     |
| LOG2                      |
| LOWER                     |
| LPAD                      |
| LPAD_ORACLE               |
| LTRIM                     |
| LTRIM_ORACLE              |
| MAKEDATE                  |
| MAKETIME                  |
| MAKE_SET                  |
| MASTER_GTID_WAIT          |
| MASTER_POS_WAIT           |
| MD5                       |
| MONTHNAME                 |
| NAME_CONST                |
| NVL                       |
| NVL2                      |
| NULLIF                    |
| OCT                       |
| OCTET_LENGTH              |
| ORD                       |
| PERIOD_ADD                |
| PERIOD_DIFF               |
| PI                        |
| POW                       |
| POWER                     |
| QUOTE                     |
| REGEXP_INSTR              |
| REGEXP_REPLACE            |
| REGEXP_SUBSTR             |
| RADIANS                   |
| RAND                      |
| RELEASE_ALL_LOCKS         |
| RELEASE_LOCK              |
| REPLACE_ORACLE            |
| REVERSE                   |
| ROUND                     |
| RPAD                      |
| RPAD_ORACLE               |
| RTRIM                     |
| RTRIM_ORACLE              |
| SEC_TO_TIME               |
| SHA                       |
| SHA1                      |
| SHA2                      |
| SIGN                      |
| SIN                       |
| SLEEP                     |
```

```
| SOUNDEX                   |
| SPACE                     |
| SQRT                      |
| STRCMP                    |
| STR_TO_DATE               |
| SUBSTR_ORACLE             |
| SUBSTRING_INDEX           |
| SUBTIME                   |
| SYS_GUID                  |
| TAN                       |
| TIMEDIFF                  |
| TIME_FORMAT               |
| TIME_TO_SEC               |
| TO_BASE64                 |
| TO_CHAR                   |
| TO_DAYS                   |
| TO_SECONDS                |
| UCASE                     |
| UNCOMPRESS                |
| UNCOMPRESSED_LENGTH       |
| UNHEX                     |
| UNIX_TIMESTAMP            |
| UPDATEXML                 |
| UPPER                     |
| UUID                      |
| UUID_SHORT                |
| VERSION                   |
| WEEKDAY                   |
| WEEKOFYEAR                |
| WSREP_LAST_WRITTEN_GTID   |
| WSREP_LAST_SEEN_GTID      |
| WSREP_SYNC_WAIT_UPTO_GTID |
| YEARWEEK                  |
+---------------------------+
234 rows in set (0.001 sec)
```

See Also

- Reserved Words

This page is licensed: CC BY-SA / Gnu FDL

Information Schema STATISTICS Table

The Information Schema `STATISTICS` table provides information about table indexes.

It contains the following columns:

Column	Description
TABLE_CATALOG	Always def.
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
NON_UNIQUE	1 if the index can have duplicates, 0 if not.
INDEX_SCHEMA	Database name.
INDEX_NAME	Index name. The primary key is always named PRIMARY.

Column	Description
SEQ_IN_INDEX	The column sequence number, starting at 1.
COLUMN_NAME	Column name.
COLLATION	A for sorted in ascending order, or NULL for unsorted.
CARDINALITY	Estimate of the number of unique values stored in the index based on statistics stored as integers. Higher cardinalities usually mean a greater chance of the index being used in a join. Updated by the ANALYZE TABLE statement or myisamchk -a.
SUB_PART	NULL if the whole column is indexed, or the number of indexed characters if partly indexed.
PACKED	NULL if not packed, otherwise how the index is packed.
NULLABLE	YES if the column may contain NULLs, empty string if not.
INDEX_TYPE	Index type, one of BTREE, RTREE, HASH or FULLTEXT. See Storage Engine Index Types.
COMMENT	Index comments from the CREATE INDEX statement.
IGNORED	Whether or not an index will be ignored by the optimizer. See Ignored Indexes. From MariaDB 10.6.0.
CREATE_OPTIONS	Extra CREATE TABLE options.This column is available from MariaDB 13.0.

The SHOW INDEX statement produces similar output.

Example

```
SELECT * FROM INFORMATION_SCHEMA.STATISTICS\G
...
*************************** 85. row ***************************
TABLE_CATALOG: def
 TABLE_SCHEMA: test
   TABLE_NAME: table1
   NON_UNIQUE: 1
 INDEX_SCHEMA: test
   INDEX_NAME: col2
 SEQ_IN_INDEX: 1
  COLUMN_NAME: col2
    COLLATION: A
  CARDINALITY: 6
     SUB_PART: NULL
       PACKED: NULL
     NULLABLE:
   INDEX_TYPE: BTREE
      COMMENT:
INDEX_COMMENT:
...
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema SYSTEM_VARIABLES Table

The Information Schema `SYSTEM_VARIABLES` table shows current values and various metadata of all system variables.

It contains the following columns:

Column	Description
VARIABLE_NAME	System variable name.
SESSION_VALUE	Session value of the variable or `NULL` if the variable only has a global scope.
GLOBAL_VALUE	Global value of the variable or `NULL` if the variable only has a session scope.
GLOBAL_VALUE_ORIGIN	How the global value is set — a compile-time default, auto-configured by the server, configuration file (or on a command line), with the SQL statement.
DEFAULT_VALUE	Compile-time default value of the variable.
VARIABLE_SCOPE	Global, session, or session-only.
VARIABLE_TYPE	Data type of the variable value.
VARIABLE_COMMENT	Help text, usually shown in `mariadbd --help --verbose`.
NUMERIC_MIN_VALUE	For numeric variables — minimal allowed value.
NUMERIC_MAX_VALUE	For numeric variables — maximal allowed value.
NUMERIC_BLOCK_SIZE	For numeric variables — a valid value must be a multiple of the “block size”.
ENUM_VALUE_LIST	For `ENUM`, `SET`, and `FLAGSET` variables — the list of recognized values.
READ_ONLY	Whether a variable can be set with the SQL statement. Note that many “read only” variables can still be set on the command line.
COMMAND_LINE_ARGUMENT	Whether an argument is required when setting the variable on the command line. `NULL` when a variable can not be set on the command line.
GLOBAL_VALUE_PATH	Which config file the variable got its value from. `NULL` if not set in any config file.
IS_DEPRECATED	Used to identify deprecated variables in the server configuration (in an option file or on the command line). Available from MariaDB 13.0.
DEPRECATED_REPLACEMENT	The replacement of a deprecated variable. If `NULL`, there is no replacement. Available from MariaDB 13.0.

Examples

Show Values of a Specific Variable

```
SELECT * FROM information_schema.SYSTEM_VARIABLES
  WHERE VARIABLE_NAME='JOIN_BUFFER_SIZE'\G
*************************** 1. row ***************************
        VARIABLE_NAME: JOIN_BUFFER_SIZE
        SESSION_VALUE: 131072
         GLOBAL_VALUE: 131072
  GLOBAL_VALUE_ORIGIN: COMPILE-TIME
       DEFAULT_VALUE: 131072
      VARIABLE_SCOPE: SESSION
       VARIABLE_TYPE: BIGINT UNSIGNED
    VARIABLE_COMMENT: The size of the buffer that is used for joins
    NUMERIC_MIN_VALUE: 128
    NUMERIC_MAX_VALUE: 18446744073709551615
   NUMERIC_BLOCK_SIZE: 128
      ENUM_VALUE_LIST: NULL
            READ_ONLY: NO
COMMAND_LINE_ARGUMENT: REQUIRED
        IS_DEPRECATED: NO
```

Show Deprecated Variables That are set in the Configuration File

This query shows variables used in your server deployment that are deprecated:

```
SELECT VARIABLE_NAME, GLOBAL_VALUE, DEFAULT_VALUE
 FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES
 WHERE GLOBAL_VALUE_ORIGIN='CONFIG' AND IS_DEPRECATED = 'YES';
 -- Possible output
+----------------------+-------------+---------------+
| VARIABLE_NAME         | GLOBAL_VALUE | DEFAULT_VALUE |
+----------------------+-------------+---------------+
| INNODB_LRU_FLUSH_SIZE | 0            | 32            |
+----------------------+-------------+---------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema TABLE_CONSTRAINTS Table

The Information Schema `TABLE_CONSTRAINTS` table contains information about tables that have constraints.

It has the following columns:

Column	Description
CONSTRAINT_CATALOG	Always def.
CONSTRAINT_SCHEMA	Database name containing the constraint.
CONSTRAINT_NAME	Constraint name.
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
CONSTRAINT_TYPE	Type of constraint; one of UNIQUE, PRIMARY KEY, FOREIGN KEY or CHECK.

The REFERENTIAL_CONSTRAINTS table has more information about foreign keys.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema TABLE_PRIVILEGES Table

The Information Schema `TABLE_PRIVILEGES` table contains table privilege information derived from the mysql.tables_priv grant table.

It has the following columns:

Column	Description
GRANTEE	In the format user_name@host_name.
TABLE_CATALOG	Always def.
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
PRIVILEGE_TYPE	One of SELECT, INSERT, UPDATE, REFERENCES, ALTER, INDEX, DROP or CREATE VIEW.
IS_GRANTABLE	Whether the user has the GRANT OPTION for this privilege.

Similar information can be accessed with the SHOW GRANTS statement. See the GRANT article for more about privileges.

The table only shows privileges granted on the table level. This differs from the Sys Schema privileges_by_table_by_level view, which shows all privileges, broken down by table.

For a description of the privileges that are shown in this table, see table privileges.

See Also

- sys.privileges_by_table_by_level

This page is licensed: CC BY-SA / Gnu FDL

Information Schema TABLE_STATISTICS Table

The Information Schema `TABLE_STATISTICS` table shows statistics on table usage.

This is part of the User Statistics feature, which is not enabled by default.

It contains the following columns:

Field	Type	Notes
TABLE_SCHEMA	varchar(192)	The schema (database) name.
TABLE_NAME	varchar(192)	The table name.
ROWS_READ	bigint(21)	The number of rows read from the table.
ROWS_CHANGED	bigint(21)	The number of rows changed in the table.
ROWS_CHANGED_X_INDEXES	bigint(21)	The number of rows changed in the table, multiplied by the number of indexes changed.
ROWS_INSERTED	bigint(21)	From MariaDB 11.5
ROWS_UPDATED	bigint(21)	From MariaDB 11.5
ROWS_DELETED	bigint(21)	From MariaDB 11.5
KEY_READ_HITS	bigint(21)	From MariaDB 11.5
KEY_READ_MISSES	bigint(21)	From MariaDB 11.5

Example

```
SELECT * FROM INFORMATION_SCHEMA.TABLE_STATISTICS WHERE TABLE_NAME='user';
+--------------+------------+-----------+--------------+-----------------------+
| TABLE_SCHEMA | TABLE_NAME | ROWS_READ | ROWS_CHANGED | ROWS_CHANGED_X_INDEXES |
+--------------+------------+-----------+--------------+-----------------------+
| mysql        | user       |         5 |            2 |                     2 |
+--------------+------------+-----------+--------------+-----------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema TABLES Table

CURRENT

The Information Schema table shows information about the various tables and views on the server.

< 11.2.0

The Information Schema table shows information about the various tables, excluding `TEMPORARY` tables, except for tables from the `Information Schema` database and views on the server.

It contains the following columns:

Column	Description
TABLE_CATALOG	Always `def`.
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.

Column	Description
TABLE_TYPE	One of `BASE TABLE` for a regular table, VIEW for a view, `SYSTEM VIEW` for Information Schema tables, `SYSTEM VERSIONED` for system-versioned tables, `SEQUENCE` for sequences or `TEMPORARY` for local temporary tables.
ENGINE	Storage Engine.
VERSION	Version number from the table's `.frm` file.
ROW_FORMAT	Row format (see InnoDB, Aria and MyISAM row formats).
TABLE_ROWS	Number of rows in the table. Some engines, such as XtraDB and InnoDB may store an estimate.
AVG_ROW_LENGTH	Average row length in the table.
DATA_LENGTH	For InnoDB/XtraDB, the index size, in pages, multiplied by the page size. For Aria and MyISAM, length of the data file, in bytes. For MEMORY, the approximate allocated memory.
MAX_DATA_LENGTH	Maximum length of the data file, ie the total number of bytes that could be stored in the table. Not used in XtraDB and InnoDB.
INDEX_LENGTH	Length of the index file.
DATA_FREE	Bytes allocated but unused. For InnoDB tables in a shared tablespace, the free space of the shared tablespace with small safety margin. An estimate in the case of partitioned tables - see the PARTITIONS table.
AUTO_INCREMENT	Next AUTO_INCREMENT value.
CREATE_TIME	Time the table was created. Some engines just return the ctime information from the file system layer here, in that case the value is not necessarily the table creation time but rather the time the file system metadata for it had last changed.
UPDATE_TIME	Time the table was last updated. On Windows, the timestamp is not updated on update, so MyISAM values will be inaccurate. In InnoDB, if shared tablespaces are used, will be NULL, while buffering can also delay the update, so the value will differ from the actual time of the last `UPDATE`, `INSERT` or `DELETE`.
CHECK_TIME	Time the table was last checked. Not kept by all storage engines, in which case will be `NULL`.
TABLE_COLLATION	Character set and collation.
CHECKSUM	Live checksum value, if any.
CREATE_OPTIONS	Extra CREATE TABLE options.
TABLE_COMMENT	Table comment provided when MariaDB created the table.
MAX_INDEX_LENGTH	Maximum index length (supported by MyISAM and Aria tables).
TEMPORARY	Set to `'Y'` for local temporary tables.

Although the table is standard in the Information Schema, all but `TABLE_CATALOG`, `TABLE_SCHEMA`, `TABLE_NAME`, `TABLE_TYPE`, `ENGINE` and `VERSION` are MySQL and MariaDB extensions.

SHOW TABLES lists all tables in a database.

Examples

```
SELECT * FROM information_schema.tables WHERE table_schema='test'\G
*************************** 1. row ***************************
   TABLE_CATALOG: def
    TABLE_SCHEMA: test
      TABLE_NAME: xx5
      TABLE_TYPE: BASE TABLE
          ENGINE: InnoDB
         VERSION: 10
      ROW_FORMAT: Dynamic
      TABLE_ROWS: 0
  AVG_ROW_LENGTH: 0
     DATA_LENGTH: 16384
 MAX_DATA_LENGTH: 0
    INDEX_LENGTH: 0
       DATA_FREE: 0
  AUTO_INCREMENT: NULL
     CREATE_TIME: 2020-11-18 15:57:10
     UPDATE_TIME: NULL
      CHECK_TIME: NULL
 TABLE_COLLATION: latin1_swedish_ci
        CHECKSUM: NULL
  CREATE_OPTIONS:
   TABLE_COMMENT:
MAX_INDEX_LENGTH: 0
       TEMPORARY: N
*************************** 2. row ***************************
   TABLE_CATALOG: def
    TABLE_SCHEMA: test
      TABLE_NAME: xx4
      TABLE_TYPE: BASE TABLE
          ENGINE: MyISAM
         VERSION: 10
      ROW_FORMAT: Fixed
      TABLE_ROWS: 0
  AVG_ROW_LENGTH: 0
     DATA_LENGTH: 0
 MAX_DATA_LENGTH: 1970324836974591
    INDEX_LENGTH: 1024
       DATA_FREE: 0
  AUTO_INCREMENT: NULL
     CREATE_TIME: 2020-11-18 15:56:57
     UPDATE_TIME: 2020-11-18 15:56:57
      CHECK_TIME: NULL
 TABLE_COLLATION: latin1_swedish_ci
        CHECKSUM: NULL
  CREATE_OPTIONS:
   TABLE_COMMENT:
MAX_INDEX_LENGTH: 17179868160
       TEMPORARY: N
...
```

Example with `temporary = 'y'`:

```
SELECT * FROM information_schema.tables WHERE temporary='y'\G
 *************************** 1. row ***************************
   TABLE_CATALOG: def
    TABLE_SCHEMA: information_schema
      TABLE_NAME: INNODB_FT_DELETED
      TABLE_TYPE: SYSTEM VIEW
          ENGINE: MEMORY
         VERSION: 11
      ROW_FORMAT: Fixed
      TABLE_ROWS: NULL
  AVG_ROW_LENGTH: 9
     DATA_LENGTH: 0
 MAX_DATA_LENGTH: 9437184
    INDEX_LENGTH: 0
       DATA_FREE: 0
  AUTO_INCREMENT: NULL
     CREATE_TIME: 2020-11-17 21:54:02
     UPDATE_TIME: NULL
      CHECK_TIME: NULL
 TABLE_COLLATION: utf8_general_ci
        CHECKSUM: NULL
  CREATE_OPTIONS: max_rows=1864135
   TABLE_COMMENT:
MAX_INDEX_LENGTH: 0
       TEMPORARY: Y
...
```

View Tables in Order of Size

Return a list of all tables in the database, ordered by size:

```
SELECT table_schema AS `DB`, table_name AS `TABLE`,
  ROUND(((data_length + index_length) / 1024 / 1024), 2) `Size (MB)`
  FROM information_schema.TABLES
  ORDER BY (data_length + index_length) DESC;

+--------------------+--------------------------------------+-----------+
| DB                 | Table                                | Size (MB) |
+--------------------+--------------------------------------+-----------+
| wordpress          | wp_simple_history_contexts           |      7.05 |
| wordpress          | wp_posts                             |      6.59 |
| wordpress          | wp_simple_history                    |      3.05 |
| wordpress          | wp_comments                          |      2.73 |
| wordpress          | wp_commentmeta                       |      2.47 |
| wordpress          | wp_simple_login_log                  |      2.03 |
...
```

Narrow down that information for a specific database:

```
SELECT table_schema AS `DB`, table_name AS `TABLE`,
  ROUND(((data_length + index_length) / 1024 / 1024), 2) `Size (MB)`
  FROM information_schema.TABLES
  WHERE table_schema='nation'
  ORDER BY (data_length + index_length) DESC;
+--------+------------------+-----------+
| DB     | TABLE            | Size (MB) |
+--------+------------------+-----------+
| nation | country_stats    |      0.41 |
| nation | countries        |      0.06 |
| nation | country_languages |      0.06 |
| nation | regions          |      0.03 |
| nation | guests           |      0.02 |
| nation | vips             |      0.02 |
| nation | region_areas     |      0.02 |
| nation | languages        |      0.02 |
| nation | continents       |      0.02 |
+--------+------------------+-----------+
```

Return information about a temporary table:

```
CREATE TEMPORARY TABLE foo.t1 (a INT);

SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA='foo' AND TEMPORARY='y'\G
*************************** 1. row ***************************
   TABLE_CATALOG: def
    TABLE_SCHEMA: foo
      TABLE_NAME: t1
      TABLE_TYPE: TEMPORARY
...
       TEMPORARY: Y
```

See Also

- mysqlshow
- SHOW TABLE STATUS
- Finding Tables Without Primary Keys

This page is licensed: CC BY-SA / Gnu FDL

Information Schema TABLESPACES Table

The Information Schema `TABLESPACES` table contains information about active tablespaces..

The table is a MariaDB and MySQL extension, and does not include information about InnoDB tablespaces.

Column	Description
TABLESPACE_NAME	
ENGINE	
TABLESPACE_TYPE	
LOGFILE_GROUP_NAME	
EXTENT_SIZE	
AUTOEXTEND_SIZE	
MAXIMUM_SIZE	
NODEGROUP_ID	
TABLESPACE_COMMENT	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema THREAD_POOL_GROUPS Table

> **Note:**
>
> This table is available as of MariaDB 10.5.

The table provides information about thread pool groups, and contains the following columns:

Column	Description
GROUP_ID	the thread group this row is showing data for
CONNECTIONS	the number of clients currently connected to this thread group
THREADS	total number of threads in this group (ACTIVE+STANDBY+LISTENER)
ACTIVE_THREADS	number of threads currently executing a query
STANDBY_THREADS	number of threads in reserve that do not currently execute anything
QUEUE_LENGTH	number of client requests waiting for execution
HAS_LISTENER	whether there is an active listener thread right now, always 1 if thread_pool_dedicated_listener is ON
IS_STALLED	whether there's currently an active worker thread in this group that has exceeded thread_pool_stall_limit time

Setting thread_pool_dedicated_listener will give each group its own dedicated listener, and the listener thread will not pick up work items. As a result, the actual queue size in the table will be more exact, since IO requests are immediately dequeued from poll, without delay.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema THREAD_POOL_QUEUES Table

> **Note:**
>
> This table is available from MariaDB 10.5.

The table provides information about thread pool queues, and contains the following columns:

Column	Description
GROUP_ID	the thread group this row is showing data for
POSITION	position in the groups queue
PRIORITY	request priority, see priority scheduling
CONNECTION_ID	ID of the client connection that submitted the queued request
QUEUEING_TIME_MICROSECONDS	how long the request has already been waiting in the queue in microseconds

Setting thread_poll_exact_stats will provides better queueing time statistics by using a high precision timestamp, at a small performance cost, for the time when the connection was added to the queue. This timestamp helps calculate the queuing time shown in the table.

Setting thread_pool_dedicated_listener will give each group its own dedicated listener, and the listener thread will not pick up work items. As a result, the queueing time in the table will be more exact, since IO requests are immediately dequeued from poll, without delay.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema THREAD_POOL_STATS Table

> **Note:**
>
> This table is available from MariaDB 10.5.

The table provides performance counter information for the thread pool, and contains the following columns:

Column	Description
GROUP_ID	the thread group this row is showing data for
THREAD_CREATIONS	number of threads created for this group so far
THREAD_CREATIONS_DUE_TO_STALL	number of threads created due to detected stalls
WAKES	standby thread wakeups
WAKES_DUE_TO_STALL	wakeups due to stalls
THROTTLES	how often thread creation was throttled, see also: thread-creation-throttling
STALLS	number of detected stalls
POLLS_BY_LISTENER	
POLLS_BY_WORKER	
DEQUEUES_BY_LISTENER	
DEQUEUES_BY_WORKER	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema THREAD_POOL_WAITS Table

> **Note:**
>
> This table is available from MariaDB 10.5.

The table provides wait counters for the thread pool, and contains the following columns:

Column	Description
REASON	name of the reason for waiting, e.g. ROW_LOCK, DISKIO, NET …
COUNT	how often a wait for this specific reason has happened so far

This page is licensed: CC BY-SA / Gnu FDL

Information Schema TRIGGERED_UPDATE_COLUMNS

> **Note:**
>
> This table is available from MariaDB 12.2.

The Information Schema `TRIGGERED_UPDATE_COLUMNS` table shows columns specified in triggers for update operations.

Columns are displayed only if the user has non-SELECT privileges on the columns.

It contains the following columns:

Column	Description
TRIGGER_CATALOG	Always `def` in MariaDB.
TRIGGER_SCHEMA	Name of the database containing the trigger.

Column	Description
TRIGGER_NAME	Name of the trigger.
EVENT_OBJECT_CATALOG	Always `def` in MariaDB
EVENT_OBJECT_SCHEMA	Name of the database containing the table that the trigger is defined on.
EVENT_OBJECT_TABLE	Name of the table that the trigger is defined on.
EVENT_OBJECT_COLUMN	Name of the column that the trigger is defined on.

Examples

```
CREATE TABLE t (a INT, b INT, c INT);
INSERT INTO t VALUES (1, 2, 3);
CREATE TABLE t1 (a_old INT, b_old INT, a_new INT, b_new INT);
CREATE TABLE t2 (a_old INT, b_old INT, a_new INT, b_new INT);
CREATE TRIGGER trigger_before_update BEFORE UPDATE OF a, b ON t
  FOR EACH ROW INSERT INTO t1 VALUES (OLD.a, OLD.b, NEW.a, NEW.b);

SELECT * FROM INFORMATION_SCHEMA.TRIGGERED_UPDATE_COLUMNS\G
*************************** 1. row ***************************
 TRIGGER_CATALOG: def
TRIGGER_SCHEMA: test
TRIGGER_NAME: trigger_before_update
EVENT_OBJECT_CATALOG: def
EVENT_OBJECT_SCHEMA: test
EVENT_OBJECT_TABLE: t
EVENT_OBJECT_COLUMN: a
*************************** 2. row ***************************
TRIGGER_CATALOG: def
TRIGGER_SCHEMA: test
TRIGGER_NAME: trigger_before_update
EVENT_OBJECT_CATALOG: def
EVENT_OBJECT_SCHEMA: test
EVENT_OBJECT_TABLE: t
EVENT_OBJECT_COLUMN: b
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema TRIGGERS Table

The Information Schema `TRIGGERS` table contains information about triggers.

It has the following columns:

Column	Description
TRIGGER_CATALOG	Always def.
TRIGGER_SCHEMA	Database name in which the trigger occurs.
TRIGGER_NAME	Name of the trigger.
EVENT_MANIPULATION	The event that activates the trigger. Prior to MariaDB 12.0, one of `INSERT`, `UPDATE` or `DELETE`. From MariaDB 12.0, can contain multiple values, comma-separated.
EVENT_OBJECT_CATALOG	Always def.
EVENT_OBJECT_SCHEMA	Database name on which the trigger acts.
EVENT_OBJECT_TABLE	Table name on which the trigger acts.
ACTION_ORDER	Indicates the order that the action will be performed in (of the list of a table's triggers with identical `EVENT_MANIPULATION` and `ACTION_TIMING` values). See the FOLLOWS/PRECEDES clauses.

Column	Description
ACTION_CONDITION	`NULL`
ACTION_STATEMENT	Trigger body, UTF-8 encoded.
ACTION_ORIENTATION	Always ROW.
ACTION_TIMING	Whether the trigger acts `BEFORE` or `AFTER` the event that triggers it.
ACTION_REFERENCE_OLD_TABLE	Always `NULL`.
ACTION_REFERENCE_NEW_TABLE	Always `NULL`.
ACTION_REFERENCE_OLD_ROW	Always `OLD`.
ACTION_REFERENCE_NEW_ROW	Always `NEW`.
CREATED	Always `NULL`.
SQL_MODE	The SQL_MODE when the trigger was created, and which it uses for execution.
DEFINER	The account that created the trigger, in the form user_name@host_name
CHARACTER_SET_CLIENT	The client character set when the trigger was created, from the session value of the character_set_client system variable.
COLLATION_CONNECTION	The client collation when the trigger was created, from the session value of the collation_connection system variable.
DATABASE_COLLATION	Collation of the associated database.

Queries to the `TRIGGERS` table return information even if the user doesn't have `SELECT` privileges on the columns. The columns `ACTION_CONDITION`, `ACTION_STATEMENT` and `DEFINER` are `NULL` unless the user has the `TRIGGER` privilege. Similar information is returned by the SHOW TRIGGERS statement.

See also

- Trigger Overview
- CREATE TRIGGER
- DROP TRIGGER
- SHOW TRIGGERS
- SHOW CREATE TRIGGER
- Trigger Limitations

This page is licensed: CC BY-SA / Gnu FDL

Information Schema USER_PRIVILEGES Table

The Information Schema `USER_PRIVILEGES` table contains global user privilege information derived from the mysql.global_priv grant table.

It contains the following columns:

Column	Description
GRANTEE	In the format user_name@host_name.
TABLE_CATALOG	Always def.

Column	Description
PRIVILEGE_TYPE	The specific privilege, for example CREATE USER, RELOAD, SHUTDOWN, SELECT, INSERT, UPDATE or REFERENCES.
IS_GRANTABLE	Whether the user has the GRANT OPTION for this privilege.

The database, table and column privileges returned here are the ones granted on all databases and tables, and by implication all columns.

Similar information can be accessed with the SHOW GRANTS statement. See the GRANT article for more about privileges.

This information is also stored in the mysql.global_priv table, in the `mysql` system database.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema USER_STATISTICS Table

The Information Schema `USER_STATISTICS` table holds statistics about user activity. This is part of the User Statistics feature, which is not enabled by default.

You can use this table to find out such things as which user is causing the most load and which users are being abusive. You can also use this table to measure how close to capacity the server may be.

It contains the following columns:

Field	Type	Notes
USER	varchar(48)	The username. The value '#mysql_system_user#' appears when there is no username (such as for the slave SQL thread).
TOTAL_CONNECTIONS	int(11)	The number of connections created for this user.
CONCURRENT_CONNECTIONS	int(11)	The number of concurrent connections for this user.
CONNECTED_TIME	int(11)	The cumulative number of seconds elapsed while there were connections from this user.
BUSY_TIME	double	The cumulative number of seconds there was activity on connections from this user.
CPU_TIME	double	The cumulative CPU time elapsed while servicing this user's connections.
BYTES_RECEIVED	bigint(21)	The number of bytes received from this user's connections.
BYTES_SENT	bigint(21)	The number of bytes sent to this user's connections.
BINLOG_BYTES_WRITTEN	bigint(21)	The number of bytes written to the binary log from this user's connections.
ROWS_READ	bigint(21)	The number of rows read by this user's connections.
ROWS_SENT	bigint(21)	The number of rows sent by this user's connections.
ROWS_DELETED	int(21)	The number of rows deleted by this user's connections.
ROWS_INSERTED	bigint(21)	The number of rows inserted by this user's connections.
ROWS_UPDATED	bigint(21)	The number of rows updated by this user's connections.
KEY_READ_HITS	bigint(21)	From MariaDB 11.5

Field	Type	Notes
KEY_READ_MISSES	bigint(21)	From MariaDB 11.5
SELECT_COMMANDS	bigint(21)	The number of SELECT commands executed from this user's connections.
UPDATE_COMMANDS	bigint(21)	The number of UPDATE commands executed from this user's connections.
OTHER_COMMANDS	bigint(21)	The number of other commands executed from this user's connections.
COMMIT_TRANSACTIONS	bigint(21)	The number committed transactions. Note that in autocommit mode every statement is a separate transaction.
ROLLBACK_TRANSACTIONS	bigint(21)	The number of transactions that were rolled back.
DENIED_CONNECTIONS	bigint(21)	The number of connections denied to this user.
LOST_CONNECTIONS	bigint(21)	The number of this user's connections that were terminated uncleanly.
ACCESS_DENIED	bigint(21)	The number of times this user's connections issued commands that were denied.
EMPTY_QUERIES	bigint(21)	The number of times this user's connections sent queries to the server that did not return data to the client (a per-user aggregate of the empty_queries status variable).
TOTAL_SSL_CONNECTIONS	bigint(21)	The number of TLS connections created for this user.
MAX_STATEMENT_TIME_EXCEEDED	bigint(21)	The number of times a statement was aborted, because it was executed longer than its MAX_STATEMENT_TIME threshold.

Example

```
SELECT * FROM information_schema.USER_STATISTICS\G
*************************** 1. row ***************************
                 USER: root
     TOTAL_CONNECTIONS: 1
CONCURRENT_CONNECTIONS: 0
        CONNECTED_TIME: 297
             BUSY_TIME: 0.001725
              CPU_TIME: 0.001982
        BYTES_RECEIVED: 388
            BYTES_SENT: 2327
  BINLOG_BYTES_WRITTEN: 0
            ROWS_READ: 0
            ROWS_SENT: 12
         ROWS_DELETED: 0
        ROWS_INSERTED: 13
         ROWS_UPDATED: 0
       SELECT_COMMANDS: 4
       UPDATE_COMMANDS: 0
        OTHER_COMMANDS: 3
   COMMIT_TRANSACTIONS: 0
 ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
      LOST_CONNECTIONS: 0
         ACCESS_DENIED: 0
         EMPTY_QUERIES: 1
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema USER_VARIABLES Table

Description

The `USER_VARIABLES` table is created when the user_variables plugin is enabled, and contains information about user-defined variables.

The table contains the following columns:

Column	Description
VARIABLE_NAME	Variable name.
VARIABLE_VALUE	Variable value.
VARIABLE_TYPE	Variable type.
CHARACTER_SET_NAME	Character set.

User variables are reset and the table emptied with the FLUSH USER_VARIABLES statement. `SHOW USER_VARIABLES` displays a subset of the data.

Example

```
SET @v1 = 0;
SET @v2 = 'abc';
SET @v3 = CAST(123 AS CHAR(5));

SELECT * FROM information_schema.USER_VARIABLES ORDER BY VARIABLE_NAME;
+---------------+----------------+---------------+--------------------+
| VARIABLE_NAME | VARIABLE_VALUE | VARIABLE_TYPE | CHARACTER_SET_NAME |
+---------------+----------------+---------------+--------------------+
| v1            | 0              | INT           | latin1             |
| v2            | abc            | VARCHAR       | utf8               |
| v3            | 123            | VARCHAR       | utf8               |
+---------------+----------------+---------------+--------------------+

SHOW USER_VARIABLES;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| v3            | 123   |
| v2            | abc   |
| v1            | 0     |
+---------------+-------+
```

See Also

- User-defined variables
- Performance Schema user_variables_by_thread Table

This page is licensed: CC BY-SA / Gnu FDL

Information Schema USERS Table

> **Note:**
>
> This table is available from MariaDB 11.5.

The Information Schema `USERS` table contains information about users, password expiry, and the limits set by max_password_errors. Unprivileged users can access their own data, which is different to what mysql.global_priv provides.

It contains the following columns:

Column	Description
USER	In the format user_name@host_name.
PASSWORD_ERRORS	A current accumulated value of consecutive password login failures. If password_errors is not applicable for the user (see max_password_errors), PASSWORD_ERRORS will be NULL. Only password login attempts with nonempty password are taken into account.
PASSWORD_EXPIRATION_TIME	A timestamp with the exact point in time calculated from password_last_changed and password_lifetime (i.e. days) stored for the user.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema WSREP_BF_ABORTS

> **Note:**
>
> This plugin is used in MariaDB Galera Cluster.

> **Note:**
>
> This plugin is available as of MariaDB Enterprise Server 11.8.

This table contains execution state information for Galera threads. The respective plugin is not installed by default — you have to install it.

The table displays the history of BF[44] aborts. It has these columns:

- TIME - Datetime when bf abort happened
- VICTIM_TRX_ID - Victim trx ID or NULL
- VICTIM_THREAD_ID - Victim thread
- THD ID VICTIM_QUERY_ID - Victim query id
- VICTIM_SEQNO - Victim TO seqno associated with victim transaction
- VICTIM_CLIENT_STATE - Victim thread wsrep client state at the moment when BF abort attempt happened
- VICTIM_CLIENT_MODE - Victim thread wsrep client mode
- VICTIM_TRX_STATE - Victim thread wsrep transaction state at the moment when BF abort attempt happened
- VICTIM_LOCK - Victim lock information
- BF_TRX_ID - BF thread transaction ID
- BF_THREAD_ID - BF thread THD ID
- BF_QUERY_ID - BF query id
- BF_SEQNO - TO sequence number associated with BF transaction
- BF_CLIENT_STATE - BF thread wsrep client state
- BF_CLIENT_MODE - BF thread wsrep client mode
- BF_TRX_STATE - BF thread wsrep transaction state
- BF_LOCK - Information about the lock which BF thread tried to acquire
- SPACE_ID - Lock space_id in case of record or table lock
- PAGE_NO - Lock page_no in case of record lock
- INDEX_NAME - Record lock index name
- TABLE_NAME - Lock table name

Example output:

```
SELECT * FROM INFORMATION_SCHEMA.WSREP_BF_ABORT_HISTORY LIMIT 100;
+----------------------------+---------------+------------------+-----------------+--------------+-------
--------------+--------------------+------------------+-----------------------------+-----------+---------
-----+-------------+----------+-----------------+----------------+--------------+-----------------------
----+----------+---------+------------+------------------+
| TIME                       | VICTIM_TRX_ID | VICTIM_THREAD_ID | VICTIM_QUERY_ID | VICTIM_SEQNO |
VICTIM_CLIENT_STATE | VICTIM_CLIENT_MODE | VICTIM_TRX_STATE | VICTIM_LOCK                | BF_TRX_ID |
BF_THREAD_ID | BF_QUERY_ID | BF_SEQNO | BF_CLIENT_STATE | BF_CLIENT_MODE | BF_TRX_STATE | BF_LOCK
| SPACE_ID | PAGE_NO | INDEX_NAME | TABLE_NAME       |
+----------------------------+---------------+------------------+-----------------+--------------+-------
--------------+--------------------+------------------+-----------------------------+-----------+---------
-----+-------------+----------+-----------------+----------------+--------------+-----------------------
----+----------+---------+------------+------------------+
| 2025-05-14 09:48:31.539207 |        378998 |               94 |         2365214 |         NULL | exec
| local              | executing        | REC|LOCK_X|NOT_GAP|WAITING |    379010 |            1 |
2365229 |   174271 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.597005 |        379044 |               81 |         2365344 |         NULL | exec
| local              | certifying       | REC|LOCK_X|NOT_GAP         |    379060 |            1 |
2365419 |   174284 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.609301 |        379055 |              104 |         2365412 |         NULL | exec
| local              | certifying       | REC|LOCK_X|NOT_GAP         |    379070 |            1 |
2365445 |   174289 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.725899 |        379134 |              101 |         2366437 |         NULL | exec
| local              | executing        | REC|LOCK_X|NOT_GAP         |    379164 |            1 |
2366536 |   174307 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |  147008 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.789128 |        379094 |              131 |         2366355 |         NULL | exec
| local              | certifying       | REC|LOCK_X|NOT_GAP         |    379195 |            1 |
2366599 |   174323 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.789140 |        379101 |              124 |         2366103 |         NULL | exec
| local              | executing        | REC|LOCK_X|NOT_GAP|WAITING |    379195 |            1 |
2366599 |   174323 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.789145 |        379093 |               82 |         2366113 |         NULL | exec
| local              | executing        | REC|LOCK_X|NOT_GAP|WAITING |    379195 |            1 |
2366599 |   174323 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.789148 |        379110 |              117 |         2366193 |         NULL | exec
| local              | executing        | REC|LOCK_X|NOT_GAP|WAITING |    379195 |            1 |
2366599 |   174323 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.789152 |        379126 |              111 |         2366201 |         NULL | exec
| local              | executing        | REC|LOCK_X|NOT_GAP|WAITING |    379195 |            1 |
2366599 |   174323 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.789156 |        379100 |              134 |         2366318 |         NULL | exec
| local              | executing        | REC|LOCK_X|NOT_GAP|WAITING |    379195 |            1 |
2366599 |   174323 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.789159 |        379109 |               93 |         2366334 |         NULL | exec
| local              | executing        | REC|LOCK_X|NOT_GAP|WAITING |    379195 |            1 |
2366599 |   174323 | exec            | high priority  | executing    | REC|LOCK_X|NOT_GAP|WAITING |
12 |       4 | PRIMARY    | `test`.`sbtest1` |
| 2025-05-14 09:48:31.789163 |        379119 |               90 |         2366384 |         NULL | exec
| local              | executing       +----------------------------+---------------+------------------+-
----------------+--------------+--------------------+--------------------+------------------+-----------
-----------------+-----------+--------------+-------------+----------+-----------------+----------------
+--------------+----------------------------+----------+---------+------------+------------------+
...
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema WSREP_CERT_KEYS

> **Note:**
>
> This table is used in MariaDB Galera Cluster.

> **Note:**
>
> This table is available as of MariaDB Enterprise Server 11.8.

The Wsrep certification keys, the primary keys of changed rows, are collected into a write set for Galera Cluster's certification-based replication. The write set is used to ensure data consistency by detecting conflicting transactions before they commit. The `WSREP_CERT_INFO` plugin exposes wsrep certification keys in two information schema tables: `WSREP_CERT_KEYS` (this table), and `WSREP_CERT_KEYS_HISTORY`.

There are several key types.

Exclusive

The Exclusive key type is used when a transaction inserts, updates, or deletes a row. Example:

```
CREATE TABLE parent (f1 TINYINT PRIMARY KEY, f2 TINYINT)engine=innodb;
START TRANSACTION;
INSERT INTO parent VALUES (1, 3);
SELECT * FROM INFORMATION_SCHEMA.WSREP_CERT_KEYS;
COMMIT;
+--------+-----------------------------+-----------+
| THD_ID | KEY_STRING                  | KEY_TYPE  |
+--------+-----------------------------+-----------+
|      9 | 74657374 706172656E74 0001  | exclusive |
+--------+-----------------------------+-----------+
1 row in set (0.001 sec)
```

Reference

The Reference key type is used when a `FOREIGN KEY` constraint is defined with referenced table key tpe set as `REFERENCE`. In the example, `706172656E74` is a hexadecimal value for 'parent'. Example:

```
CREATE TABLE parent (f1 TINYINT PRIMARY KEY, f2 TINYINT) engine=innodb;
CREATE TABLE child (f1 TINYINT PRIMARY KEY, f2 TINYINT, FOREIGN KEY (f2) REFERENCES parent(f1))
engine=innodb;

START TRANSACTION;
INSERT INTO parent VALUES (1, 2);
SELECT KEY_STRING, KEY_TYPE FROM INFORMATION_SCHEMA.WSREP_CERT_KEYS;
+-----------------------------+-----------+
| KEY_STRING                  | KEY_TYPE  |
+-----------------------------+-----------+
| 74657374 706172656E74 0001  | exclusive |
+-----------------------------+-----------+
1 row in set (0.001 sec)
COMMIT;

START TRANSACTION;
INSERT INTO child VALUES (10, 1);
MariaDB [test]> SELECT KEY_STRING, KEY_TYPE FROM INFORMATION_SCHEMA.WSREP_CERT_KEYS;
+-----------------------------+-----------+
| KEY_STRING                  | KEY_TYPE  |
+-----------------------------+-----------+
| 74657374 706172656E74 0001  | exclusive |
| 74657374 706172656E74 0001  | reference |
| 74657374 6368696C64 000A    | exclusive |
+-----------------------------+-----------+
3 rows in set (0.000 sec)
COMMIT;
```

Update

The Update key type is used when a transaction updates a row and `wsrep_protocol_version` is greater than 4. Otherwise, the Exclusive key type is used. Example:

```
CREATE TABLE parent (f1 TINYINT PRIMARY KEY, f2 TINYINT)engine=innodb;
update parent set f2=1;

SELECT KEY_STRING, KEY_TYPE FROM INFORMATION_SCHEMA.WSREP_CERT_KEYS;
+-----------------------------+----------+
| KEY_STRING                  | KEY_TYPE |
+-----------------------------+----------+
| 74657374 706172656E74 0001  | update   |
+-----------------------------+----------+
1 row in set (0.001 sec)
```

Shared

This key type is used when a referenced table update happens, and `wsrep_protocol_version` is smaller than 4.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema WSREP_CERT_KEYS_HISTORY

> **Note:**
>
> This table is used in MariaDB Galera Cluster.

> **Note:**
>
> This table is available as of MariaDB Enterprise Server 11.8.

This table contains certification key information history for Galera threads. See the WSREP_CERT_KEYS documentation for details.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema WSREP_CONNECTIONS

> **Note:**
>
> This table is used in MariaDB Galera Cluster.

> **Note:**
>
> This table is available as of MariaDB Enterprise Server 11.8.

This table contains active Galera connections on queried nodes in the cluster. Example output:

```
MariaDB [(none)]> SELECT * FROM information_schema.wsrep_connections;
+-----------------+-------------------+-----------------+-----------------+------------------------+-----
-----------------------------------------------+-----------------------------------------------------------+---
------------------+
| connection_id   | connection_scheme | local_address   | remote_address  | cipher                 |
certificate_subject                             | certificate_issuer
| certificate_version |
+-----------------+-------------------+-----------------+-----------------+------------------------+-----
-----------------------------------------------+-----------------------------------------------------------+---
------------------+
| 136018124822096 | ssl               | 127.0.0.1:19033 | 127.0.0.1:59010 | TLS_AES_256_GCM_SHA384 |
/C=FI/ST=Helsinki/L=Helsinki/O=Galera/CN=galera.1 | /C=FI/ST=Helsinki/L=Helsinki/O=Galera/CN=galera.root
| TLSv1.3             |
| 136018124966656 | ssl               | 127.0.0.1:49650 | 127.0.0.1:19039 | TLS_AES_256_GCM_SHA384 |
/C=FI/ST=Helsinki/L=Helsinki/O=Galera/CN=galera.1 | /C=FI/ST=Helsinki/L=Helsinki/O=Galera/CN=galera.root
| TLSv1.3             |
+-----------------+-------------------+-----------------+-----------------+------------------------+-----
-----------------------------------------------+-----------------------------------------------------------+---
------------------+
2 rows in set (0.004 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema WSREP_MEMBERSHIP Table

The `WSREP_STATUS` table makes Galera node cluster membership information available through the Information Schema. The same information can be returned using the SHOW WSREP_MEMBERSHIP statement. Only users with the SUPER can access information from this table.

The `WSREP_MEMBERSHIP` table is part of the WSREP_INFO plugin.

Example

```
SELECT * FROM information_schema.WSREP_MEMBERSHIP;
+-------+--------------------------------------+-------+-----------------+
| INDEX | UUID                                 | NAME  | ADDRESS         |
+-------+--------------------------------------+-------+-----------------+
|     0 | 46da96e3-6e9e-11e4-95a2-f609aa5444b3 | node1 | 10.0.2.15:16000 |
|     1 | 5f6bc72a-6e9e-11e4-84ed-57ec6780a3d3 | node2 | 10.0.2.15:16001 |
|     2 | 7473fd75-6e9e-11e4-91de-0b541ad91bd0 | node3 | 10.0.2.15:16002 |
+-------+--------------------------------------+-------+-----------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema WSREP_STATUS Table

The `WSREP_STATUS` table makes Galera node cluster status information available through the Information Schema. The same information can be returned using the SHOW WSREP_STATUS statement. Only users with the SUPER

privilege can access information from this table.

The `WSREP_STATUS` table is part of the WSREP_INFO plugin.

Example

```
SELECT * FROM information_schema.WSREP_STATUS\G
*************************** 1. row ***************************
         NODE_INDEX: 0
        NODE_STATUS: Synced
     CLUSTER_STATUS: Primary
       CLUSTER_SIZE: 3
 CLUSTER_STATE_UUID: 00b0fbad-6e84-11e4-8a8b-376f19ce8ee7
CLUSTER_STATE_SEQNO: 2
    CLUSTER_CONF_ID: 3
                GAP: NO
   PROTOCOL_VERSION: 3
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema WSREP_THD_STATE

> **Note:**
>
> This table is used in MariaDB Galera Cluster.

> **Note:**
>
> This table is available as of MariaDB Enterprise Server 11.8.

This table contains execution state information for Galera threads, and has these columns:

- ID - Thread ID
- OS_THREAD_ID - Os thread ID
- CLIENT_MODE - Client mode (local, high priority, toi…)
- CLIENT_STATE - Wsrep query state (idle, exec, result…)
- TRANSACTION_STATE - Wsrep transaction state or “none” if no active transaction
- SEQNO - Commit order seqno if assigned, otherwise -1
- DEPENDS_ON - Depends on seqno if assigned, otherwise -1
  The table must have low overhead (populated only when queried) and must be enabled at all times (not plugin or plugin must be loaded by default).
- GTID - transaction GTID assigned during commit phase

Example output:

```
SELECT * FROM information_schema.wsrep_thd_info;
ID  OS_THREAD_ID    MODE    STATE   TRANSACTION_ID  TRANSACTION_STATE   SEQNO   DEPENDS_ON  GTID
14  128875466565312 local   exec    64  executing   NULL    NULL    NULL
13  128875466872512 local   exec    61  committing  9   3   0-1-7
12  128875467179712 local   idle    NULL    aborted NULL    NULL    NULL
2   128875634570944 high priority   exec    NULL    executing   NULL    NULL    NULL
1   128875634878144 local   none    NULL    executing   NULL    NULL    NULL
```

There is also an `INFORMATION_SCHEMA.WSREP_THD_STATE_HISTORY` table that contains the history of execution state information for Galera threads.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema WSREP_THD_STATE_HISTORY

> **Note:**
>
> This table is used in MariaDB Galera Cluster.

> **Note:**
>
> This table is available as of MariaDB Enterprise Server 11.8.

This table contains execution state information history for Galera threads. See the WSREP_THD_STATE documentation for details.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema InnoDB Tables

- Information Schema INNODB_CMP_PER_INDEX and INNODB_CMP_PER_INDEX_RESET Tables
- Information Schema INNODB_BUFFER_PAGE Table
- Information Schema INNODB_BUFFER_PAGE_LRU Table
- Information Schema INNODB_BUFFER_POOL_PAGES Table
- Information Schema INNODB_BUFFER_POOL_PAGES_BLOB Table
- Information Schema INNODB_BUFFER_POOL_PAGES_INDEX Table
- Information Schema INNODB_BUFFER_POOL_STATS Table
- Information Schema INNODB_CHANGED_PAGES Table
- Information Schema INNODB_CMP and INNODB_CMP_RESET Tables
- Information Schema INNODB_CMPMEM and INNODB_CMPMEM_RESET Tables
- Information Schema INNODB_FT_BEING_DELETED Table
- Information Schema INNODB_FT_CONFIG Table
- Information Schema INNODB_FT_DEFAULT_STOPWORD Table
- Information Schema INNODB_FT_DELETED Table
- Information Schema INNODB_FT_INDEX_CACHE Table
- Information Schema INNODB_FT_INDEX_TABLE Table
- Information Schema INNODB_LOCK_WAITS Table
- Information Schema INNODB_LOCKS Table
- Information Schema INNODB_METRICS Table
- Information Schema INNODB_MUTEXES Table
- Information Schema INNODB_SYS_COLUMNS Table
- Information Schema INNODB_SYS_DATAFILES Table
- Information Schema INNODB_SYS_FIELDS Table
- Information Schema INNODB_SYS_FOREIGN Table
- Information Schema INNODB_SYS_FOREIGN_COLS Table
- Information Schema INNODB_SYS_INDEXES Table
- Information Schema INNODB_SYS_SEMAPHORE_WAITS Table
- Information Schema INNODB_SYS_TABLES Table
- Information Schema INNODB_SYS_TABLESPACES Table
- Information Schema INNODB_SYS_TABLESTATS Table
- Information Schema INNODB_SYS_VIRTUAL Table
- Information Schema INNODB_TABLESPACES_ENCRYPTION Table

- Information Schema INNODB_TABLESPACES_SCRUBBING Table
- Information Schema INNODB_TRX Table
- Information Schema TEMP_TABLES_INFO Table
- Information Schema INNODB_UNDO_LOGS Table

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_CMP_PER_INDEX and INNODB_CMP_PER_INDEX_RESET Tables

The `INNODB_CMP_PER_INDEX` and `INNODB_CMP_PER_INDEX_RESET` tables contain status information on compression operations related to compressed XtraDB/InnoDB tables, grouped by individual indexes. These tables are only populated if the innodb_cmp_per_index_enabled system variable is set to `ON`.

The PROCESS privilege is required to query this table.

These tables contains the following columns:

Column Name	Description
DATABASE_NAME	Database containing the index.
TABLE_NAME	Table containing the index.
INDEX_NAME	Other values are totals which refer to this index's compression.
COMPRESS_OPS	How many times a page of INDEX_NAME has been compressed. This happens when a new page is created because the compression log runs out of space. This value includes both successful operations and compression failures.
COMPRESS_OPS_OK	How many times a page of INDEX_NAME has been successfully compressed. This value should be as close as possible to COMPRESS_OPS. If it is notably lower, either avoid compressing some tables, or increase the KEY_BLOCK_SIZE for some compressed tables.
COMPRESS_TIME	Time (in seconds) spent to compress pages of the size PAGE_SIZE. This value includes time spent in compression failures.
UNCOMPRESS_OPS	How many times a page of INDEX_NAME has been uncompressed. This happens when an uncompressed version of a page is created in the buffer pool, or when a compression failure occurs.
UNCOMPRESS_TIME	Time (in seconds) spent to uncompress pages of INDEX_NAME.

These tables can be used to measure the effectiveness of XtraDB/InnoDB compression, per table or per index. The values in these tables show which tables perform better with index compression, and which tables cause too many *compression failures* or perform too many compression/uncompression operations. When compression performs badly for a table, this might mean that you should change its `KEY_BLOCK_SIZE`, or that the table should not be compressed.

`INNODB_CMP_PER_INDEX` and `INNODB_CMP_PER_INDEX_RESET` have the same columns and always contain the same values, but when `INNODB_CMP_PER_INDEX_RESET` is queried, both the tables are cleared. `INNODB_CMP_PER_INDEX_RESET` can be used, for example, if a script periodically logs the performances of compression in the last period of time. `INNODB_CMP_PER_INDEX` can be used to see the cumulated statistics.

See Also

Other tables that can be used to monitor XtraDB/InnoDB compressed tables:

- INNODB_CMP and INNODB_CMP_RESET
- INNODB_CMPMEM and INNODB_CMPMEM_RESET

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_BUFFER_PAGE Table

The Information Schema `INNODB_BUFFER_PAGE` table contains information about pages in the buffer pool.

Querying this table can have a noticeable performance impact on a production server.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
POOL_ID	Buffer Pool identifier. From MariaDB 10.5.1 returns a value of 0, since multiple InnoDB buffer pool instances has been removed.
BLOCK_ID	Buffer Pool Block identifier.
SPACE	Tablespace identifier. Matches the SPACE value in the INNODB_SYS_TABLES table.
PAGE_NUMBER	Buffer pool page number.
PAGE_TYPE	Page type; one of allocated (newly-allocated page), index (B-tree node), undo_log (undo log page), inode (index node), ibuf_free_list (insert buffer free list), ibuf_bitmap (insert buffer bitmap), system (system page), trx_system (transaction system data), file_space_header (file space header), extent_descriptor (extent descriptor page), blob (uncompressed blob page), compressed_blob (first compressed blob page), compressed_blob2 (subsequent compressed blob page) or unknown.
FLUSH_TYPE	Flush type.
FIX_COUNT	Count of the threads using this block in the buffer pool. When it is zero, the block can be evicted from the buffer pool.
IS_HASHED	Whether or not a hash index has been built on this page.
NEWEST_MODIFICATION	Most recent modification's Log Sequence Number.
OLDEST_MODIFICATION	Oldest modification's Log Sequence Number.
ACCESS_TIME	Abstract number representing the time the page was first accessed.
TABLE_NAME	Table that the page belongs to.
INDEX_NAME	Index that the page belongs to, either a clustered index or a secondary index.
NUMBER_RECORDS	Number of records the page contains.
DATA_SIZE	Size in bytes of all the records contained in the page.
COMPRESSED_SIZE	Compressed size in bytes of the page, or NULL for pages that aren't compressed.

Column	Description
PAGE_STATE	Page state; one of FILE_PAGE (page from a file) or MEMORY (page from an in-memory object) for valid data, or one of NULL, READY_FOR_USE, NOT_USED, REMOVE_HASH.
IO_FIX	Whether there is I/O pending for the page; one of IO_NONE (no pending I/O), IO_READ (read pending), IO_WRITE (write pending).
IS_OLD	Whether the page is old or not.
FREE_PAGE_CLOCK	Freed_page_clock counter, which tracks the number of blocks removed from the end of the least recently used (LRU) list, at the time the block was last placed at the head of the list.

The related INFORMATION_SCHEMA.INNODB_BUFFER_PAGE_LRU table contains the same information, but with an LRU (least recently used) position rather than block id.

Examples

```
DESC information_schema.innodb_buffer_page;
+---------------------+---------------------+------+-----+---------+-------+
| Field               | Type                | Null | Key | Default | Extra |
+---------------------+---------------------+------+-----+---------+-------+
| POOL_ID             | bigint(21) unsigned | NO   |     | 0       |       |
| BLOCK_ID            | bigint(21) unsigned | NO   |     | 0       |       |
| SPACE               | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_NUMBER         | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_TYPE           | varchar(64)         | YES  |     | NULL    |       |
| FLUSH_TYPE          | bigint(21) unsigned | NO   |     | 0       |       |
| FIX_COUNT           | bigint(21) unsigned | NO   |     | 0       |       |
| IS_HASHED           | varchar(3)          | YES  |     | NULL    |       |
| NEWEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       |
| OLDEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       |
| ACCESS_TIME         | bigint(21) unsigned | NO   |     | 0       |       |
| TABLE_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| INDEX_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| NUMBER_RECORDS      | bigint(21) unsigned | NO   |     | 0       |       |
| DATA_SIZE           | bigint(21) unsigned | NO   |     | 0       |       |
| COMPRESSED_SIZE     | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_STATE          | varchar(64)         | YES  |     | NULL    |       |
| IO_FIX              | varchar(64)         | YES  |     | NULL    |       |
| IS_OLD              | varchar(3)          | YES  |     | NULL    |       |
| FREE_PAGE_CLOCK     | bigint(21) unsigned | NO   |     | 0       |       |
+---------------------+---------------------+------+-----+---------+-------+
```

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE\G
...
*************************** 6. row ***************************
            POOL_ID: 0
          BLOCK_ID: 5
             SPACE: 0
       PAGE_NUMBER: 11
         PAGE_TYPE: INDEX
        FLUSH_TYPE: 1
         FIX_COUNT: 0
         IS_HASHED: NO
NEWEST_MODIFICATION: 2046835
OLDEST_MODIFICATION: 0
       ACCESS_TIME: 2585566280
        TABLE_NAME: `SYS_INDEXES`
        INDEX_NAME: CLUST_IND
    NUMBER_RECORDS: 57
         DATA_SIZE: 4016
   COMPRESSED_SIZE: 0
        PAGE_STATE: FILE_PAGE
            IO_FIX: IO_NONE
            IS_OLD: NO
   FREE_PAGE_CLOCK: 0
...
```

See Also

- InnoDB Buffer Pool

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_BUFFER_PAGE_LRU Table

The Information Schema `INNODB_BUFFER_PAGE_LRU` table contains information about pages in the buffer pool and how they are ordered for eviction purposes.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
POOL_ID	Buffer Pool identifier. From MariaDB 10.5.1 returns a value of 0, since multiple InnoDB buffer pool instances has been removed.
LRU_POSITION	LRU (Least recently-used), for determining eviction order from the buffer pool.
SPACE	Tablespace identifier. Matches the SPACE value on the INNODB_SYS_TABLES table.
PAGE_NUMBER	Buffer pool page number.
PAGE_TYPE	Page type; one of allocated (newly-allocated page), index (B-tree node), undo_log (undo log page), inode (index node), ibuf_free_list (insert buffer free list), ibuf_bitmap (insert buffer bitmap), system (system page), trx_system (transaction system data), file_space_header (file space header), extent_descriptor (extent descriptor page), blob (uncompressed blob page), compressed_blob (first compressed blob page), compressed_blob2 (subsequent compressed blob page) or unknown.
FLUSH_TYPE	Flush type. 0= FLUSH_KEEP, 1 =FLUSH_RELEASE, 2 = FLUSH_IGNORE_CHANGED, 3= FLUSH_FORCE_WRITE
FIX_COUNT	Count of the threads using this block in the buffer pool. When it is zero, the block can be evicted from the buffer pool.

Column	Description
IS_HASHED	Whether or not a hash index has been built on this page.
NEWEST_MODIFICATION	Most recent modification's Log Sequence Number.
OLDEST_MODIFICATION	Oldest modification's Log Sequence Number.
ACCESS_TIME	Abstract number representing the time the page was first accessed.
TABLE_NAME	Table that the page belongs to.
INDEX_NAME	Index that the page belongs to, either a clustered index or a secondary index.
NUMBER_RECORDS	Number of records the page contains.
DATA_SIZE	Size in bytes of all the records contained in the page.
COMPRESSED_SIZE	Compressed size in bytes of the page, or NULL for pages that aren't compressed.
PAGE_STATE	Page state; one of FILE_PAGE (page from a file) or MEMORY (page from an in-memory object) for valid data, or one of NULL, READY_FOR_USE, NOT_USED, REMOVE_HASH.
IO_FIX	Whether there is I/O pending for the page; one of IO_NONE (no pending I/O), IO_READ (read pending), IO_WRITE (write pending).
IS_OLD	Whether the page is old or not.
FREE_PAGE_CLOCK	Freed_page_clock counter, which tracks the number of blocks removed from the end of the LRU list, at the time the block was last placed at the head of the list.

The related INFORMATION_SCHEMA.INNODB_BUFFER_PAGE table contains the same information, but with a block id rather than LRU position.

Example

```
DESC information_schema.innodb_buffer_page_lru;
+---------------------+---------------------+------+-----+---------+-------+
| Field               | Type                | Null | Key | Default | Extra |
+---------------------+---------------------+------+-----+---------+-------+
| POOL_ID             | bigint(21) unsigned | NO   |     | 0       |       |
| LRU_POSITION        | bigint(21) unsigned | NO   |     | 0       |       |
| SPACE               | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_NUMBER         | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_TYPE           | varchar(64)         | YES  |     | NULL    |       |
| FLUSH_TYPE          | bigint(21) unsigned | NO   |     | 0       |       |
| FIX_COUNT           | bigint(21) unsigned | NO   |     | 0       |       |
| IS_HASHED           | varchar(3)          | YES  |     | NULL    |       |
| NEWEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       |
| OLDEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       |
| ACCESS_TIME         | bigint(21) unsigned | NO   |     | 0       |       |
| TABLE_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| INDEX_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| NUMBER_RECORDS      | bigint(21) unsigned | NO   |     | 0       |       |
| DATA_SIZE           | bigint(21) unsigned | NO   |     | 0       |       |
| COMPRESSED_SIZE     | bigint(21) unsigned | NO   |     | 0       |       |
| COMPRESSED          | varchar(3)          | YES  |     | NULL    |       |
| IO_FIX              | varchar(64)         | YES  |     | NULL    |       |
| IS_OLD              | varchar(3)          | YES  |     | NULL    |       |
| FREE_PAGE_CLOCK     | bigint(21) unsigned | NO   |     | 0       |       |
+---------------------+---------------------+------+-----+---------+-------+
```

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE_LRU\G
...
*************************** 6. row ***************************
            POOL_ID: 0
       LRU_POSITION: 5
              SPACE: 0
        PAGE_NUMBER: 11
          PAGE_TYPE: INDEX
         FLUSH_TYPE: 1
          FIX_COUNT: 0
          IS_HASHED: NO
NEWEST_MODIFICATION: 2046835
OLDEST_MODIFICATION: 0
        ACCESS_TIME: 2585566280
         TABLE_NAME: `SYS_INDEXES`
         INDEX_NAME: CLUST_IND
     NUMBER_RECORDS: 57
          DATA_SIZE: 4016
    COMPRESSED_SIZE: 0
         COMPRESSED: NO
             IO_FIX: IO_NONE
             IS_OLD: NO
    FREE_PAGE_CLOCK: 0
...
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_BUFFER_POOL_PAGES Table

The Information Schema `INNODB_BUFFER_POOL_PAGES` table is a Percona enhancement, and is only available for XtraDB, not InnoDB (see XtraDB and InnoDB). It contains a record for each page in the buffer pool.

It has the following columns:

Column	Description
PAGE_TYPE	Type of page; one of index, undo_log, inode, ibuf_free_list, allocated, bitmap, sys, trx_sys, fsp_hdr, xdes, blob, zblob, zblob2 and unknown.

Column	Description
SPACE_ID	Tablespace ID.
PAGE_NO	Page offset within tablespace.
LRU_POSITION	Page position in the LRU (least-recently-used) list.
FIX_COUNT	Page reference count, incremented each time the page is accessed. 0 if the page is not currently being accessed.
FLUSH_TYPE	Flush type of the most recent flush.0 (LRU), 2 (flush_list)

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_BUFFER_POOL_PAGES_BLOB Table

The Information Schema `INNODB_BUFFER_POOL_PAGES_BLOB` table is a Percona enhancement, and is only available for XtraDB, not InnoDB (see XtraDB and InnoDB). It contains information about buffer pool blob pages.

It has the following columns:

Column	Description
SPACE_ID	Tablespace ID.
PAGE_NO	Page offset within tablespace.
COMPRESSED	1 if the blob contains compressed data, 0 if not.
PART_LEN	Page data length.
NEXT_PAGE_NO	Next page number.
LRU_POSITION	Page position in the LRU (least-recently-used) list.
FIX_COUNT	Page reference count, incremented each time the page is accessed. 0 if the page is not currently being accessed.
FLUSH_TYPE	Flush type of the most recent flush.0 (LRU), 2 (flush_list)

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_BUFFER_POOL_PAGES_INDEX Table

The Information Schema `INNODB_BUFFER_POOL_PAGES` table is a Percona enhancement, and is only available for XtraDB, not InnoDB (see XtraDB and InnoDB). It contains information about buffer pool index pages.

It has the following columns:

Column	Description
INDEX_ID	Index name
SPACE_ID	Tablespace ID
PAGE_NO	Page offset within tablespace.
N_RECS	Number of user records on the page.
DATA_SIZE	Total data size in bytes of records in the page.
HASHED	1 if the block is in the adaptive hash index, 0 if not.
ACCESS_TIME	Page's last access time.
MODIFIED	1 if the page has been modified since being loaded, 0 if not.
DIRTY	1 if the page has been modified since it was last flushed, 0 if not
OLD	1 if the page in the in the old blocks of the LRU (least-recently-used) list, 0 if not.
LRU_POSITION	Position in the LRU (least-recently-used) list.

Column	Description
FIX_COUNT	Page reference count, incremented each time the page is accessed. 0 if the page is not currently being accessed.
FLUSH_TYPE	Flush type of the most recent flush.0 (LRU), 2 (flush_list)

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_BUFFER_POOL_STATS Table

The Information Schema `INNODB_BUFFER_POOL_STATS` table contains information about pages in the buffer pool, similar to what is returned with the SHOW ENGINE INNODB STATUS statement.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
POOL_ID	Buffer Pool identifier. From MariaDB 10.5.1 returns a value of 0, since multiple InnoDB buffer pool instances has been removed.
POOL_SIZE	Size in pages of the buffer pool.
FREE_BUFFERS	Number of free pages in the buffer pool.
DATABASE_PAGES	Total number of pages in the buffer pool.
OLD_DATABASE_PAGES	Number of pages in the old sublist.
MODIFIED_DATABASE_PAGES	Number of dirty pages.
PENDING_DECOMPRESS	Number of pages pending decompression.
PENDING_READS	Pending buffer pool level reads.
PENDING_FLUSH_LRU	Number of pages in the LRU pending flush.
PENDING_FLUSH_LIST	Number of pages in the flush list pending flush.
PAGES_MADE_YOUNG	Pages moved from the old sublist to the new sublist.
PAGES_NOT_MADE_YOUNG	Pages that have remained in the old sublist without moving to the new sublist.
PAGES_MADE_YOUNG_RATE	Hits that cause blocks to move to the top of the new sublist.
PAGES_MADE_NOT_YOUNG_RATE	Hits that do not cause blocks to move to the top of the new sublist due to the innodb_old_blocks delay not being met.
NUMBER_PAGES_READ	Number of pages read.
NUMBER_PAGES_CREATED	Number of pages created.
NUMBER_PAGES_WRITTEN	Number of pages written.
PAGES_READ_RATE	Number of pages read since the last printout divided by the time elapsed, giving pages read per second.
PAGES_CREATE_RATE	Number of pages created since the last printout divided by the time elapsed, giving pages created per second.
PAGES_WRITTEN_RATE	Number of pages written since the last printout divided by the time elapsed, giving pages written per second.
NUMBER_PAGES_GET	Number of logical read requests.
HIT_RATE	Buffer pool hit rate.
YOUNG_MAKE_PER_THOUSAND_GETS	For every 1000 gets, the number of pages made young.
NOT_YOUNG_MAKE_PER_THOUSAND_GETS	For every 1000 gets, the number of pages not made young.

Column	Description
NUMBER_PAGES_READ_AHEAD	Number of pages read ahead.
NUMBER_READ_AHEAD_EVICTED	Number of pages read ahead by the read-ahead thread that were later evicted without being accessed by any queries.
READ_AHEAD_RATE	Pages read ahead since the last printout divided by the time elapsed, giving read-ahead rate per second.
READ_AHEAD_EVICTED_RATE	Read-ahead pages not accessed since the last printout divided by time elapsed, giving the number of read-ahead pages evicted without access per second.
LRU_IO_TOTAL	Total least-recently used I/O.
LRU_IO_CURRENT	Least-recently used I/O for the current interval.
UNCOMPRESS_TOTAL	Total number of pages decompressed.
UNCOMPRESS_CURRENT	Number of pages decompressed in the current interval

Examples

```
DESC information_schema.innodb_buffer_pool_stats;
+---------------------------------+---------------------+------+-----+---------+-------+
| Field                           | Type                | Null | Key | Default | Extra |
+---------------------------------+---------------------+------+-----+---------+-------+
| POOL_ID                         | bigint(21) unsigned | NO   |     | 0       |       |
| POOL_SIZE                       | bigint(21) unsigned | NO   |     | 0       |       |
| FREE_BUFFERS                    | bigint(21) unsigned | NO   |     | 0       |       |
| DATABASE_PAGES                  | bigint(21) unsigned | NO   |     | 0       |       |
| OLD_DATABASE_PAGES              | bigint(21) unsigned | NO   |     | 0       |       |
| MODIFIED_DATABASE_PAGES         | bigint(21) unsigned | NO   |     | 0       |       |
| PENDING_DECOMPRESS              | bigint(21) unsigned | NO   |     | 0       |       |
| PENDING_READS                   | bigint(21) unsigned | NO   |     | 0       |       |
| PENDING_FLUSH_LRU               | bigint(21) unsigned | NO   |     | 0       |       |
| PENDING_FLUSH_LIST              | bigint(21) unsigned | NO   |     | 0       |       |
| PAGES_MADE_YOUNG                | bigint(21) unsigned | NO   |     | 0       |       |
| PAGES_NOT_MADE_YOUNG            | bigint(21) unsigned | NO   |     | 0       |       |
| PAGES_MADE_YOUNG_RATE           | double              | NO   |     | 0       |       |
| PAGES_MADE_NOT_YOUNG_RATE       | double              | NO   |     | 0       |       |
| NUMBER_PAGES_READ               | bigint(21) unsigned | NO   |     | 0       |       |
| NUMBER_PAGES_CREATED            | bigint(21) unsigned | NO   |     | 0       |       |
| NUMBER_PAGES_WRITTEN            | bigint(21) unsigned | NO   |     | 0       |       |
| PAGES_READ_RATE                 | double              | NO   |     | 0       |       |
| PAGES_CREATE_RATE               | double              | NO   |     | 0       |       |
| PAGES_WRITTEN_RATE              | double              | NO   |     | 0       |       |
| NUMBER_PAGES_GET                | bigint(21) unsigned | NO   |     | 0       |       |
| HIT_RATE                        | bigint(21) unsigned | NO   |     | 0       |       |
| YOUNG_MAKE_PER_THOUSAND_GETS    | bigint(21) unsigned | NO   |     | 0       |       |
| NOT_YOUNG_MAKE_PER_THOUSAND_GETS | bigint(21) unsigned | NO   |     | 0       |       |
| NUMBER_PAGES_READ_AHEAD         | bigint(21) unsigned | NO   |     | 0       |       |
| NUMBER_READ_AHEAD_EVICTED       | bigint(21) unsigned | NO   |     | 0       |       |
| READ_AHEAD_RATE                 | double              | NO   |     | 0       |       |
| READ_AHEAD_EVICTED_RATE         | double              | NO   |     | 0       |       |
| LRU_IO_TOTAL                    | bigint(21) unsigned | NO   |     | 0       |       |
| LRU_IO_CURRENT                  | bigint(21) unsigned | NO   |     | 0       |       |
| UNCOMPRESS_TOTAL                | bigint(21) unsigned | NO   |     | 0       |       |
| UNCOMPRESS_CURRENT              | bigint(21) unsigned | NO   |     | 0       |       |
+---------------------------------+---------------------+------+-----+---------+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_CHANGED_PAGES Table

The Information Schema `INNODB_CHANGED_PAGES` Table contains data about modified pages from the bitmap file. It is updated at checkpoints by the log tracking thread parsing the log, so does not contain real-time data.

The number of records is limited by the value of the innodb_max_changed_pages system variable.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
SPACE_ID	Modified page space id
PAGE_ID	Modified page id
START_LSN	Interval start after which page was changed (equal to checkpoint LSN)
END_LSN	Interval end before which page was changed (equal to checkpoint LSN)

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_CMP and INNODB_CMP_RESET Tables

The `INNODB_CMP` and `INNODB_CMP_RESET` tables contain status information on compression operations related to compressed XtraDB/InnoDB tables.

The PROCESS privilege is required to query this table.

These tables contain the following columns:

Column Name	Description
PAGE_SIZE	Compressed page size, in bytes. This value is unique in the table; other values are totals which refer to pages of this size.
COMPRESS_OPS	How many times a page of the size PAGE_SIZE has been compressed. This happens when a new page is created because the compression log runs out of space. This value includes both successful operations and compression failures.
COMPRESS_OPS_OK	How many times a page of the size PAGE_SIZE has been successfully compressed. This value should be as close as possible to COMPRESS_OPS. If it is notably lower, either avoid compressing some tables, or increase the KEY_BLOCK_SIZE for some compressed tables.
COMPRESS_TIME	Time (in seconds) spent to compress pages of the size PAGE_SIZE. This value includes time spent in compression failures.
UNCOMPRESS_OPS	How many times a page of the size PAGE_SIZE has been uncompressed. This happens when an uncompressed version of a page is created in the buffer pool, or when a compression failure occurs.
UNCOMPRESS_TIME	Time (in seconds) spent to uncompress pages of the size PAGE_SIZE.

These tables can be used to measure the effectiveness of XtraDB/InnoDB table compression. When you have to decide a value for `KEY_BLOCK_SIZE`, you can create more than one version of the table (one for each candidate value) and run a realistic workload on them. Then, these tables can be used to see how the operations performed with different page sizes.

`INNODB_CMP` and `INNODB_CMP_RESET` have the same columns and always contain the same values, but when `INNODB_CMP_RESET` is queried, both the tables are cleared. `INNODB_CMP_RESET` can be used, for example, if a script periodically logs the performances of compression in the last period of time. `INNODB_CMP` can be used to see the cumulated statistics.

Examples

```
SELECT * FROM information_schema.INNODB_CMP\G
*************************** 1. row ***************************
      page_size: 1024
   compress_ops: 0
compress_ops_ok: 0
  compress_time: 0
 uncompress_ops: 0
uncompress_time: 0
...
```

See Also

Other tables that can be used to monitor XtraDB/InnoDB compressed tables:

- INNODB_CMP_PER_INDEX and INNODB_CMP_PER_INDEX_RESET
- INNODB_CMPMEM and INNODB_CMPMEM_RESET

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_CMPMEM and INNODB_CMPMEM_RESET Tables

The `INNODB_CMPMEM` and `INNODB_CMPMEM_RESET` tables contain status information on compressed pages in the buffer pool (see InnoDB COMPRESSED format).

The PROCESS privilege is required to query this table.

These tables contain the following columns:

Column Name	Description
PAGE_SIZE	Compressed page size, in bytes. This value is unique in the table; other values are totals which refer to pages of this size.
BUFFER_POOL_INSTANCE	Buffer Pool identifier. From MariaDB 10.5.1 returns a value of 0, since multiple InnoDB buffer pool instances has been removed.
PAGES_USED	Number of pages of the size PAGE_SIZE which are currently in the buffer pool.
PAGES_FREE	Number of pages of the size PAGE_SIZE which are currently free, and thus are available for allocation. This value represents the buffer pool's fragmentation. A totally unfragmented buffer pool has at most 1 free page.
RELOCATION_OPS	How many times a page of the size PAGE_SIZE has been relocated. This happens when data exceeds a page (because a row must be copied into a new page) and when two pages are merged (because their data shrunk and can now be contained in one page).
RELOCATION_TIME	Time (in seconds) spent in relocation operations for pages of the size PAGE_SIZE. This column is reset when the INNODB_CMPMEM_RESET table is queried.

These tables can be used to measure the effectiveness of InnoDB table compression. When you have to decide a value for `KEY_BLOCK_SIZE`, you can create more than one version of the table (one for each candidate value) and run a realistic workload on them. Then, these tables can be used to see how the operations performed with different page sizes.

`INNODB_CMPMEM` and `INNODB_CMPMEM_RESET` have the same columns and always contain the same values, but when `INNODB_CMPMEM_RESET` is queried, the `RELOCATION_TIME` column from both the tables are cleared. `INNODB_CMPMEM_RESET` can be used, for example, if a script periodically logs the performances of compression in the last period of time. `INNODB_CMPMEM` can be used to see the cumulated statistics.

Example

```
SELECT * FROM information_schema.INNODB_CMPMEM\G
*********************** 1. row ***********************
            page_size: 1024
 buffer_pool_instance: 0
           pages_used: 0
           pages_free: 0
      reloacation_ops: 0
      relocation_time: 0
```

See Also

Other tables that can be used to monitor InnoDB compressed tables:

- INNODB_CMP and INNODB_CMP_RESET
- INNODB_CMP_PER_INDEX and INNODB_CMP_PER_INDEX_RESET

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_FT_BEING_DELETED Table

The Information Schema `INNODB_FT_BEING_DELETED` table is only used while document ID's in the related INNODB_FT_DELETED are being removed from an InnoDB fulltext index while an OPTIMIZE TABLE is underway. At all other times the table will be empty.

The `SUPER` privilege is required to view the table, and it also requires the innodb_ft_aux_table system variable to be set.

It has the following column:

Column	Description
DOC_ID	Document ID of the row being deleted. Either an underlying ID value, or a sequence value generated by InnoDB if no usable option exists.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_FT_CONFIG Table

The Information Schema `INNODB_FT_CONFIG` table contains InnoDB fulltext index metadata.

The `SUPER` privilege is required to view the table, and it also requires the innodb_ft_aux_table system variable to be set.

It has the following columns:

Column	Description
KEY	Metadata item name.
VALUE	Associated value.

Example

```
SELECT * FROM INNODB_FT_CONFIG;
+---------------------------+-------+
| KEY                       | VALUE |
+---------------------------+-------+
| optimize_checkpoint_limit | 180   |
| synced_doc_id             | 6     |
| last_optimized_word       |       |
| deleted_doc_count         | 0     |
| total_word_count          |       |
| optimize_start_time       |       |
| optimize_end_time         |       |
| stopword_table_name       |       |
| use_stopword              | 1     |
| table_state               | 0     |
+---------------------------+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_FT_DEFAULT_STOPWORD Table

The Information Schema `INNODB_FT_DEFAULT_STOPWORD` table contains a list of default stopwords used when creating an InnoDB fulltext index.

The PROCESS privilege is required to view the table.

It has the following column:

Column	Description
VALUE	Default stopword for an InnoDB fulltext index. Setting either the innodb_ft_server_stopword_table or the innodb_ft_user_stopword_table system variable will override this.

Example

```
SELECT * FROM information_schema.INNODB_FT_DEFAULT_STOPWORD\G
*************************** 1. row ***************************
value: a
*************************** 2. row ***************************
value: about
*************************** 3. row ***************************
value: an
*************************** 4. row ***************************
value: are
...
*************************** 36. row ***************************
value: www
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_FT_DELETED Table

The Information Schema `INNODB_FT_DELETED` table contains rows that have been deleted from an InnoDB fulltext index. This information is then used to filter results on subsequent searches, removing the need to expensively reorganise the index each time a row is deleted.

The fulltext index is then only reorganized when an OPTIMIZE TABLE statement is underway. The related INNODB_FT_BEING_DELETED table contains rows being deleted while an `OPTIMIZE TABLE` is in the process of running.

The `SUPER` privilege is required to view the table, and it also requires the innodb_ft_aux_table system variable to be set.

It has the following column:

Column	Description
DOC_ID	Document ID of the deleted row deleted. Either an underlying ID value, or a sequence value generated by InnoDB if no usable option exists.

Example

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DELETED;
+--------+
| DOC_ID |
+--------+
|      2 |
+--------+

DELETE FROM test.ft_innodb LIMIT 1;

SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DELETED;
+--------+
| DOC_ID |
+--------+
|      2 |
|      3 |
+--------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_FT_INDEX_CACHE Table

The Information Schema `INNODB_FT_INDEX_CACHE` table contains information about rows that have recently been inserted into an InnoDB fulltext index. To avoid re-organizing the fulltext index each time a change is made, which would be very expensive, new changes are stored separately and only integrated when an OPTIMIZE TABLE is run.

The `SUPER` privilege is required to view the table, and it also requires the innodb_ft_aux_table system variable to be set.

It has the following columns:

Column	Description
WORD	Word from the text of a newly added row. Words can appear multiple times in the table, once per DOC_ID and POSITION combination.
FIRST_DOC_ID	First document ID where this word appears in the index.
LAST_DOC_ID	Last document ID where this word appears in the index.
DOC_COUNT	Number of rows containing this word in the index.
DOC_ID	Document ID of the newly added row, either an appropriate ID column or an internal InnoDB value.
POSITION	Position of this word instance within the DOC_ID, as an offset added to the previous POSITION instance.

Note that for `OPTIMIZE TABLE` to process InnoDB fulltext index data, the innodb_optimize_fulltext_only system variable needs to be set to `1`. When this is done, and an `OPTIMIZE TABLE` statement run, the `INNODB_FT_INDEX_CACHE` table will be emptied, and the INNODB_FT_INDEX_TABLE table will be updated.

Examples

```
SELECT * FROM INNODB_FT_INDEX_CACHE;
+------------+--------------+-------------+-----------+--------+----------+
| WORD       | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+------------+--------------+-------------+-----------+--------+----------+
| and        |            4 |           4 |         1 |      4 |        0 |
| arrived    |            4 |           4 |         1 |      4 |       20 |
| ate        |            1 |           1 |         1 |      1 |        4 |
| everybody  |            1 |           1 |         1 |      1 |        8 |
| goldilocks |            4 |           4 |         1 |      4 |        9 |
| hungry     |            3 |           3 |         1 |      3 |        8 |
| then       |            4 |           4 |         1 |      4 |        4 |
| wicked     |            2 |           2 |         1 |      2 |        4 |
| witch      |            2 |           2 |         1 |      2 |       11 |
+------------+--------------+-------------+-----------+--------+----------+
9 rows in set (0.00 sec)

INSERT INTO test.ft_innodb VALUES(3,'And she ate a pear');

SELECT * FROM INNODB_FT_INDEX_CACHE;
+------------+--------------+-------------+-----------+--------+----------+
| WORD       | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+------------+--------------+-------------+-----------+--------+----------+
| and        |            4 |           5 |         2 |      4 |        0 |
| and        |            4 |           5 |         2 |      5 |        0 |
| arrived    |            4 |           4 |         1 |      4 |       20 |
| ate        |            1 |           5 |         2 |      1 |        4 |
| ate        |            1 |           5 |         2 |      5 |        8 |
| everybody  |            1 |           1 |         1 |      1 |        8 |
| goldilocks |            4 |           4 |         1 |      4 |        9 |
| hungry     |            3 |           3 |         1 |      3 |        8 |
| pear       |            5 |           5 |         1 |      5 |       14 |
| she        |            5 |           5 |         1 |      5 |        4 |
| then       |            4 |           4 |         1 |      4 |        4 |
| wicked     |            2 |           2 |         1 |      2 |        4 |
| witch      |            2 |           2 |         1 |      2 |       11 |
+------------+--------------+-------------+-----------+--------+----------+
```

```
OPTIMIZE TABLE test.ft_innodb\G
*************************** 1. row ***************************
   Table: test.ft_innodb
      Op: optimize
Msg_type: note
Msg_text: Table does not support optimize, doing recreate + analyze instead
*************************** 2. row ***************************
   Table: test.ft_innodb
      Op: optimize
Msg_type: status
Msg_text: OK
2 rows in set (2.24 sec)

SELECT * FROM INNODB_FT_INDEX_CACHE;
+------------+--------------+-------------+-----------+--------+----------+
| WORD       | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+------------+--------------+-------------+-----------+--------+----------+
| and        |            4 |           5 |         2 |      4 |        0 |
| and        |            4 |           5 |         2 |      5 |        0 |
| arrived    |            4 |           4 |         1 |      4 |       20 |
| ate        |            1 |           5 |         2 |      1 |        4 |
| ate        |            1 |           5 |         2 |      5 |        8 |
| everybody  |            1 |           1 |         1 |      1 |        8 |
| goldilocks |            4 |           4 |         1 |      4 |        9 |
| hungry     |            3 |           3 |         1 |      3 |        8 |
| pear       |            5 |           5 |         1 |      5 |       14 |
| she        |            5 |           5 |         1 |      5 |        4 |
| then       |            4 |           4 |         1 |      4 |        4 |
| wicked     |            2 |           2 |         1 |      2 |        4 |
| witch      |            2 |           2 |         1 |      2 |       11 |
+------------+--------------+-------------+-----------+--------+----------+
13 rows in set (0.00 sec)
```

The `OPTIMIZE TABLE` statement has no effect, because the innodb_optimize_fulltext_only variable wasn't set:

```
SHOW VARIABLES LIKE 'innodb_optimize_fulltext_only';
+---------------------------------+-------+
| Variable_name                   | Value |
+---------------------------------+-------+
| innodb_optimize_fulltext_only | OFF   |
+---------------------------------+-------+

SET GLOBAL innodb_optimize_fulltext_only =1;

OPTIMIZE TABLE test.ft_innodb;
+----------------+----------+----------+----------+
| Table          | Op       | Msg_type | Msg_text |
+----------------+----------+----------+----------+
| test.ft_innodb | optimize | status   | OK       |
+----------------+----------+----------+----------+

SELECT * FROM INNODB_FT_INDEX_CACHE;
Empty set (0.00 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_FT_INDEX_TABLE Table

The Information Schema `INNODB_FT_INDEX_TABLE` table contains information about InnoDB fulltext indexes. To avoid re-organizing the fulltext index each time a change is made, which would be very expensive, new changes are stored separately and only integrated when an OPTIMIZE TABLE is run. See the INNODB_FT_INDEX_CACHE table.

The `SUPER` privilege is required to view the table, and it also requires the innodb_ft_aux_table system variable to be set.

It has the following columns:

Column	Description
WORD	Word from the text of a column with a fulltext index. Words can appear multiple times in the table, once per DOC_ID and POSITION combination.
FIRST_DOC_ID	First document ID where this word appears in the index.
LAST_DOC_ID	Last document ID where this word appears in the index.
DOC_COUNT	Number of rows containing this word in the index.
DOC_ID	Document ID of the newly added row, either an appropriate ID column or an internal InnoDB value.
POSITION	Position of this word instance within the DOC_ID, as an offset added to the previous POSITION instance.

Note that for `OPTIMIZE TABLE` to process InnoDB fulltext index data, the innodb_optimize_fulltext_only system variable needs to be set to `1`. When this is done, and an `OPTIMIZE TABLE` statement run, the INNODB_FT_INDEX_CACHE table will be emptied, and the `INNODB_FT_INDEX_TABLE` table will be updated.

Examples

```
SELECT * FROM INNODB_FT_INDEX_TABLE;
Empty set (0.00 sec)

SET GLOBAL innodb_optimize_fulltext_only =1;

OPTIMIZE TABLE test.ft_innodb;
+---------------+----------+----------+----------+
| Table         | Op       | Msg_type | Msg_text |
+---------------+----------+----------+----------+
| test.ft_innodb | optimize | status   | OK       |
+---------------+----------+----------+----------+

SELECT * FROM INNODB_FT_INDEX_TABLE;
+------------+--------------+-------------+-----------+--------+----------+
| WORD       | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+------------+--------------+-------------+-----------+--------+----------+
| and        |            4 |           5 |         2 |      4 |        0 |
| and        |            4 |           5 |         2 |      5 |        0 |
| arrived    |            4 |           4 |         1 |      4 |       20 |
| ate        |            1 |           5 |         2 |      1 |        4 |
| ate        |            1 |           5 |         2 |      5 |        8 |
| everybody  |            1 |           1 |         1 |      1 |        8 |
| goldilocks |            4 |           4 |         1 |      4 |        9 |
| hungry     |            3 |           3 |         1 |      3 |        8 |
| pear       |            5 |           5 |         1 |      5 |       14 |
| she        |            5 |           5 |         1 |      5 |        4 |
| then       |            4 |           4 |         1 |      4 |        4 |
| wicked     |            2 |           2 |         1 |      2 |        4 |
| witch      |            2 |           2 |         1 |      2 |       11 |
+------------+--------------+-------------+-----------+--------+----------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_LOCK_WAITS Table

The Information Schema `INNODB_LOCK_WAITS` table contains information about blocked InnoDB transactions. The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
REQUESTING_TRX_ID	Requesting transaction ID from the INNODB_TRX table.
REQUESTED_LOCK_ID	Lock ID from the INNODB.LOCKS table for the waiting transaction.
BLOCKING_TRX_ID	Blocking transaction ID from the INNODB_TRX table.
BLOCKING_LOCK_ID	Lock ID from the INNODB.LOCKS table of a lock held by a transaction that is blocking another transaction.

The table is often used in conjunction with the INNODB_LOCKS and INNODB_TRX tables to diagnose problematic locks and transactions.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_LOCKS Table

The Information Schema `INNODB_LOCKS` table stores information about locks that InnoDB transactions have requested but not yet acquired, or that are blocking another transaction.

It has the following columns:

Column	Description
LOCK_ID	Lock ID number - the format is not fixed, so do not rely upon the number for information.
LOCK_TRX_ID	Lock's transaction ID. Matches the INNODB_TRX.TRX_ID column.
LOCK_MODE	Lock mode. One of S (shared), X (exclusive), IS (intention shared), IX (intention exclusive row lock), S_GAP (shared gap lock), X_GAP (exclusive gap lock), IS_GAP (intention shared gap lock), IX_GAP (intention exclusive gap lock) or AUTO_INC (auto-increment table level lock).
LOCK_TYPE	Whether the lock is RECORD (row level) or TABLE level.
LOCK_TABLE	Name of the locked table,or table containing locked rows.
LOCK_INDEX	Index name if a RECORD LOCK_TYPE, or NULL if not.
LOCK_SPACE	Tablespace ID if a RECORD LOCK_TYPE, or NULL if not.
LOCK_PAGE	Locked record page number if a RECORD LOCK_TYPE, or NULL if not.
LOCK_REC	Locked record heap number if a RECORD LOCK_TYPE, or NULL if not.
LOCK_DATA	Locked record primary key as an SQL string if a RECORD LOCK_TYPE, or NULL if not. If no primary key exists, the internal InnoDB row_id number is instead used. To avoid unnecessary IO, also NULL if the locked record page is not in the buffer pool

The table is often used in conjunction with the INNODB_LOCK_WAITS and INNODB_TRX tables to diagnose problematic locks and transactions

Example

```
-- session 1
START TRANSACTION;
UPDATE t SET id = 15 WHERE id = 10;

-- session 2
DELETE FROM t WHERE id = 10;

-- session 1
USE information_schema;
SELECT l.*, t.*
    FROM information_schema.INNODB_LOCKS l
    JOIN information_schema.INNODB_TRX t
        ON l.lock_trx_id = t.trx_id
    WHERE trx_state = 'LOCK WAIT' \G
*************************** 1. row ***************************
                    lock_id: 840:40:3:2
                lock_trx_id: 840
                  lock_mode: X
                  lock_type: RECORD
                 lock_table: `test`.`t`
                 lock_index: PRIMARY
                 lock_space: 40
                  lock_page: 3
                   lock_rec: 2
                  lock_data: 10
                     trx_id: 840
                  trx_state: LOCK WAIT
                trx_started: 2019-12-23 18:43:46
     trx_requested_lock_id: 840:40:3:2
          trx_wait_started: 2019-12-23 18:43:46
                trx_weight: 2
       trx_mysql_thread_id: 46
                 trx_query: DELETE FROM t WHERE id = 10
       trx_operation_state: starting index read
         trx_tables_in_use: 1
         trx_tables_locked: 1
          trx_lock_structs: 2
     trx_lock_memory_bytes: 1136
           trx_rows_locked: 1
         trx_rows_modified: 0
   trx_concurrency_tickets: 0
       trx_isolation_level: REPEATABLE READ
         trx_unique_checks: 1
    trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
          trx_is_read_only: 0
trx_autocommit_non_locking: 0
```

.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_METRICS Table

The Information Schema `INNODB_METRICS` table contains a list of useful InnoDB performance metrics. Each row in the table represents an instrumented counter that can be stopped, started and reset, and which can be grouped together by module.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
NAME	Unique counter name.

Column	Description
SUBSYSTEM	InnoDB subsystem. See below for the matching module to use to enable/disable monitoring this subsystem with the innodb_monitor_enable and innodb_monitor_disable system variables.
COUNT	Count since being enabled.
MAX_COUNT	Maximum value since being enabled.
MIN_COUNT	Minimum value since being enabled.
AVG_COUNT	Average value since being enabled.
COUNT_RESET	Count since last being reset.
MAX_COUNT_RESET	Maximum value since last being reset.
MIN_COUNT_RESET	Minimum value since last being reset.
AVG_COUNT_RESET	Average value since last being reset.
TIME_ENABLED	Time last enabled.
TIME_DISABLED	Time last disabled
TIME_ELAPSED	Time since enabled
TIME_RESET	Time last reset.
ENABLED	1 if enabled, 0 otherwise
TYPE	Item type; one of counter, value, status_counter, set_owner, set_member.
COMMENT	Counter description.

Note: In MariaDB 10.4 and earlier the `ENABLED` column was called `STATUS`.

Enabling and Disabling Counters

Most of the counters are disabled by default. To enable them, use the innodb_monitor_enable system variable. You can either enable a variable by its name, for example:

```
SET GLOBAL innodb_monitor_enable = icp_match;
```

or enable a number of counters grouped by module. The `SUBSYSTEM` field indicates which counters are grouped together, but the following module names need to be used:

Module Name	Subsystem Field
module_metadata	metadata
module_lock	lock
module_buffer	buffer
module_buf_page	buffer_page_io
module_os	os
module_trx	transaction
module_purge	purge
module_compress	compression

Module Name	Subsystem Field
module_file	file_system
module_index	index
module_adaptive_hash	adaptive_hash_index From MariaDB 10.6.2, if innodb_adaptive_hash_index is disabled (the default), adaptive_hash_index will not be updated.
module_ibuf_system	change_buffer
module_srv	server
module_ddl	ddl
module_dml	dml
module_log	recovery
module_icp	icp

There are four counters in the `icp` subsystem:

```
SELECT NAME, SUBSYSTEM FROM INNODB_METRICS WHERE SUBSYSTEM='icp';
+------------------+-----------+
| NAME             | SUBSYSTEM |
+------------------+-----------+
| icp_attempts     | icp       |
| icp_no_match     | icp       |
| icp_out_of_range | icp       |
| icp_match        | icp       |
+------------------+-----------+
```

To enable them all, use the associated module name from the table above, `module_icp`.

```
SET GLOBAL innodb_monitor_enable = module_icp;
```

The `%` wildcard, used to represent any number of characters, can also be used when naming counters, for example:

```
SET GLOBAL innodb_monitor_enable = 'buffer%'
```

To disable counters, use the innodb_monitor_disable system variable, using the same naming rules as described above for enabling.

Counter status is not persistent, and will be reset when the server restarts. It is possible to use the options on the command line, or the `innodb_monitor_enable` option only in a configuration file.

Resetting Counters

Counters can also be reset. Resetting sets all the `*_COUNT_RESET` values to zero, while leaving the `*_COUNT` values, which perform counts since the counter was enabled, untouched. Resetting is performed with the innodb_monitor_reset (for individual counters) and innodb_monitor_reset_all (for all counters) system variables.

Simplification from MariaDB 10.6

From MariaDB 10.6, the interface was simplified by removing the following variables:

- buffer_LRU_batches_flush
- buffer_LRU_batch_flush_pages
- buffer_LRU_batches_evict
- buffer_LRU_batch_evict_pages

And by making the following reflect the status variables:

- buffer_LRU_batch_flush_total_pages: innodb_buffer_pool_pages_LRU_flushed
- buffer_LRU_batch_evict_total_pages: innodb_buffer_pool_pages_LRU_freed

The intention is to eventually remove the interface entirely (see MDEV-15706).

Examples

```
SELECT name,subsystem,type,comment FROM INFORMATION_SCHEMA.INNODB_METRICS\G
*************************** 1. row ***************************
     name: metadata_table_handles_opened
subsystem: metadata
     type: counter
  comment: Number of table handles opened
*************************** 2. row ***************************
     name: lock_deadlocks
subsystem: lock
     type: value
  comment: Number of deadlocks
*************************** 3. row ***************************
     name: lock_timeouts
subsystem: lock
     type: value
  comment: Number of lock timeouts
*************************** 4. row ***************************
     name: lock_rec_lock_waits
subsystem: lock
     type: counter
  comment: Number of times enqueued into record lock wait queue
*************************** 5. row ***************************
     name: lock_table_lock_waits
subsystem: lock
     type: counter
  comment: Number of times enqueued into table lock wait queue
*************************** 6. row ***************************
     name: lock_rec_lock_requests
subsystem: lock
     type: counter
  comment: Number of record locks requested
*************************** 7. row ***************************
     name: lock_rec_lock_created
subsystem: lock
     type: counter
  comment: Number of record locks created
*************************** 8. row ***************************
     name: lock_rec_lock_removed
subsystem: lock
     type: counter
  comment: Number of record locks removed from the lock queue
*************************** 9. row ***************************
     name: lock_rec_locks
subsystem: lock
     type: counter
  comment: Current number of record locks on tables
*************************** 10. row ***************************
     name: lock_table_lock_created
subsystem: lock
     type: counter
  comment: Number of table locks created

...

*************************** 207. row ***************************
     name: icp_attempts
subsystem: icp
     type: counter
  comment: Number of attempts for index push-down condition checks
*************************** 208. row ***************************
     name: icp_no_match
subsystem: icp
     type: counter
  comment: Index push-down condition does not match
*************************** 209. row ***************************
     name: icp_out_of_range
subsystem: icp
```

```
     type: counter
  comment: Index push-down condition out of range
*************************** 210. row ***************************
     name: icp_match
subsystem: icp
     type: counter
  comment: Index push-down condition matches
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_MUTEXES Table

The `INNODB_MUTEXES` table monitors mutex and rw locks waits. It has the following columns:

Column	Description
NAME	Name of the lock, as it appears in the source code.
CREATE_FILE	File name of the mutex implementation.
CREATE_LINE	Line number of the mutex implementation.
OS_WAITS	How many times the mutex occurred.

The `CREATE_FILE` and `CREATE_LINE` columns depend on the InnoDB/XtraDB version.

CURRENT

The table provides information about all columns listed in the previous table.

< 10.2.2

The table provides information about `rw_lock_t`, not about any mutexes.

The SHOW ENGINE INNODB STATUS statement provides similar information.

Examples

```
SELECT * FROM INNODB_MUTEXES;
+-----------------------------+--------------------+-------------+----------+
| NAME                        | CREATE_FILE        | CREATE_LINE | OS_WAITS |
+-----------------------------+--------------------+-------------+----------+
| &dict_sys->mutex            | dict0dict.cc       |         989 |        2 |
| &buf_pool->flush_state_mutex | buf0buf.cc         |        1388 |        1 |
| &log_sys->checkpoint_lock   | log0log.cc         |        1014 |        2 |
| &block->lock                | combined buf0buf.cc |        1120 |        1 |
+-----------------------------+--------------------+-------------+----------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_COLUMNS Table

The Information Schema `INNODB_SYS_COLUMNS` table contains information about InnoDB fields.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
TABLE_ID	Table identifier, matching the value from INNODB_SYS_TABLES.TABLE_ID.
NAME	Column name.
POS	Ordinal position of the column in the table, starting from 0. This value is adjusted when columns are added or removed.

Column	Description
MTYPE	Numeric column type identifier, (see the table below for an explanation of its values).
PRTYPE	Binary value of the InnoDB precise type, representing the data type, character set code and nullability.
LEN	Column length. For multi-byte character sets, represents the length in bytes.

The column `MTYPE` uses a numeric column type identifier, which has the following values:

Column Type Identifier	Description
1	VARCHAR
2	CHAR
3	FIXBINARY
4	BINARY
5	BLOB
6	INT
7	SYS_CHILD
8	SYS
9	FLOAT
10	DOUBLE
11	DECIMAL
12	VARMYSQL
13	MYSQL

Example

```
SELECT * FROM information_schema.INNODB_SYS_COLUMNS LIMIT 3\G
*************************** 1. row ***************************
TABLE_ID: 11
    NAME: ID
     POS: 0
   MTYPE: 1
  PRTYPE: 524292
     LEN: 0
*************************** 2. row ***************************
TABLE_ID: 11
    NAME: FOR_NAME
     POS: 0
   MTYPE: 1
  PRTYPE: 524292
     LEN: 0
*************************** 3. row ***************************
TABLE_ID: 11
    NAME: REF_NAME
     POS: 0
   MTYPE: 1
  PRTYPE: 524292
     LEN: 0
3 rows in set (0.00 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_DATAFILES Table

> **Note:**
>
> This table is deprecated and was removed in MariaDB 10.6.0.

The Information Schema `INNODB_SYS_DATAFILES` table contains information about InnoDB datafile paths. It was intended to provide metadata for tablespaces inside InnoDB tables, which was never implemented in MariaDB and was removed in MariaDB 10.6. The `PROCESS` privilege was required to view the table.

It contains the following columns:

Column	Description
SPACE	Numeric tablespace. Matches the INNODB_SYS_TABLES.SPACE value.
PATH	Tablespace datafile path.

Example

```
SELECT * FROM INNODB_SYS_DATAFILES;
+-------+--------------------------------+
| SPACE | PATH                           |
+-------+--------------------------------+
|    19 | ./test/t2.ibd                  |
|    20 | ./test/t3.ibd                  |
...
|    68 | ./test/animals.ibd             |
|    69 | ./test/animal_count.ibd        |
|    70 | ./test/t.ibd                   |
+-------+--------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_FIELDS Table

The Information Schema `INNODB_SYS_FIELDS` table contains information about fields that are part of an InnoDB index.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
INDEX_ID	Index identifier, matching the value from INNODB_SYS_INDEXES.INDEX_ID.
NAME	Field name, matching the value from INNODB_SYS_COLUMNS.NAME.
POS	Ordinal position of the field within the index, starting from 0. This is adjusted as columns are removed.

Example

```
SELECT * FROM information_schema.INNODB_SYS_FIELDS LIMIT 3\G
*************************** 1. row ***************************
INDEX_ID: 11
    NAME: ID
     POS: 0
*************************** 2. row ***************************
INDEX_ID: 12
    NAME: FOR_NAME
     POS: 0
*************************** 3. row ***************************
INDEX_ID: 13
    NAME: REF_NAME
     POS: 0
3 rows in set (0.00 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_FOREIGN Table

The Information Schema `INNODB_SYS_FOREIGN` table contains information about InnoDB foreign keys.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
ID	Foreign key name. Prior to MariaDB 12.1, this is preceded by the database name. From MariaDB 12.1, foreign key names are only required to be unique per table, not per database, so the redundant database name is not shown.
FOR_NAME	Database and table name of the foreign key child.
REF_NAME	Database and table name of the foreign key parent.
N_COLS	Number of foreign key index columns.
TYPE	Bit flag providing information about the foreign key.

The `TYPE` column provides a bit flag with information about the foreign key. This information is `OR`'ed together to read:

Bit Flag	Description
1	ON DELETE CASCADE
2	ON UPDATE SET NULL
4	ON UPDATE CASCADE
8	ON UPDATE SET NULL
16	ON DELETE NO ACTION
32	ON UPDATE NO ACTION

Example

Prior to MariaDB 12.1:

```
ELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN\G
*************************** 1. row ***************************
      ID: test/fk_book_author
FOR_NAME: test/book
REF_NAME: test/author
  N_COLS: 1
    TYPE: 1
...
```

From MariaDB 12.1:

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN\G
*************************** 1. row ***************************
      ID: fk_book_author
FOR_NAME: test/book
REF_NAME: test/author
  N_COLS: 1
    TYPE: 1
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_FOREIGN_COLS Table

The Information Schema `INNODB_SYS_FOREIGN_COLS` table contains information about InnoDB foreign key columns.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
ID	
FOR_COL_NAME	Child column name.
REF_COL_NAME	Parent column name.
POS	Ordinal position of the column in the table, starting from 0.

Examples

Prior to MariaDB 12.1:

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN_COLS\G
*************************** 1. row ***************************
          ID: test/fk_book_author
FOR_COL_NAME: author_id
REF_COL_NAME: id
         POS: 0
```

From MariaDB 12.1:

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN_COLS\G
*************************** 1. row ***************************
          ID: fk_book_author
FOR_COL_NAME: author_id
REF_COL_NAME: id
         POS: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_INDEXES Table

The Information Schema `INNODB_SYS_INDEXES` table contains information about InnoDB indexes.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Field	Type	Null	Default	Description
INDEX_ID	bigint(21) unsigned	NO	0	A unique index identifier.
NAME	varchar(64)	NO		Index name, lowercase for all user-created indexes, or uppercase for implicitly-created indexes; PRIMARY (primary key), GEN_CLUST_INDEX (index representing primary key where there isn't one), ID_IND, FOR_IND (validating foreign key constraint) , REF_IND.
TABLE_ID	bigint(21) unsigned	NO	0	Table identifier, matching the value from INNODB_SYS_TABLES.TABLE_ID.
TYPE	int(11)	NO	0	Numeric type identifier; one of 0 (secondary index), 1 (clustered index), 2 (unique index), 3 (primary index), 32 (full-text index).
N_FIELDS	int(11)	NO	0	Number of columns in the index. GEN_CLUST_INDEX's have a value of 0 as the index is not based on an actual column in the table.
PAGE_NO	int(11)	NO	0	Index B-tree's root page number. -1 (unused) for full-text indexes, as they are laid out over several auxiliary tables.
SPACE	int(11)	NO	0	Tablespace identifier where the index resides. 0 represents the InnoDB system tablespace, while any other value represents a table created in file-per-table mode (see the innodb_file_per_table system variable). Remains unchanged after a TRUNCATE TABLE statement, and not necessarily unique.
MERGE_THRESHOLD	int(11)	NO	0	

Example

```
SELECT * FROM information_schema.INNODB_SYS_INDEXES LIMIT 3\G
*************************** 1. row ***************************
       INDEX_ID: 11
           NAME: ID_IND
       TABLE_ID: 11
           TYPE: 3
       N_FIELDS: 1
        PAGE_NO: 302
          SPACE: 0
MERGE_THRESHOLD: 50
*************************** 2. row ***************************
       INDEX_ID: 12
           NAME: FOR_IND
       TABLE_ID: 11
           TYPE: 0
       N_FIELDS: 1
        PAGE_NO: 303
          SPACE: 0
MERGE_THRESHOLD: 50
*************************** 3. row ***************************
       INDEX_ID: 13
           NAME: REF_IND
       TABLE_ID: 11
           TYPE: 3
       N_FIELDS: 1
        PAGE_NO: 304
          SPACE: 0
MERGE_THRESHOLD: 50
3 rows in set (0.00 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_SEMAPHORE_WAITS Table

The Information Schema INNODB_SYS_SEMAPHORE_WAITS table is meant to contain information about current semaphore waits. At present it is not correctly populated. See MDEV-21330.

The PROCESS privilege is required to view the table.

It contains the following columns:

Column	Description
THREAD_ID	Thread id waiting for semaphore
OBJECT_NAME	Semaphore name
FILE	File name where semaphore was requested
LINE	Line number on above file
WAIT_TIME	Wait time
WAIT_OBJECT	
WAIT_TYPE	Object type (mutex, rw-lock)
HOLDER_THREAD_ID	Holder thread id
HOLDER_FILE	File name where semaphore was acquired
HOLDER_LINE	Line number for above
CREATED_FILE	Creation file name
CREATED_LINE	Line number for above
WRITER_THREAD	Last write request thread id

Column	Description
RESERVATION_MODE	Reservation mode (shared, exclusive)
READERS	Number of readers if only shared mode
WAITERS_FLAG	Flags
LOCK_WORD	Lock word (for developers)
LAST_READER_FILE	Removed
LAST_READER_LINE	Removed
LAST_WRITER_FILE	Last writer file name
LAST_WRITER_LINE	Above line number
OS_WAIT_COUNT	Wait count

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_TABLES Table

The Information Schema `INNODB_SYS_TABLES` table contains information about InnoDB tables.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Field	Type	Null	Default	Description
TABLE_ID	bigint(21) unsigned	NO	0	Unique InnoDB table identifier.
NAME	varchar(655)	NO		Database and table name, or the uppercase InnoDB system table name.
FLAG	int(11)	NO	0	See Flag below
N_COLS	int(11) unsigned (>= MariaDB 10.5) int(11) (<= MariaDB 10.4)	NO	0	Number of columns in the table. The count includes two or three hidden InnoDB system columns, appended to the end of the column list: DB_ROW_ID (if there is no primary key or unique index on NOT NULL columns), DB_TRX_ID, DB_ROLL_PTR.
SPACE	int(11) unsigned (>= MariaDB 10.5) int(11) (<= MariaDB 10.4)	NO	0	Tablespace identifier where the index resides. 0 represents the InnoDB system tablespace, while any other value represents a table created in file-per-table mode (see the innodb_file_per_table system variable). Remains unchanged after a TRUNCATE TABLE statement.
FILE_FORMAT	varchar(10)	YES	NULL	InnoDB file format (Antelope or Barracuda).
ROW_FORMAT	enum('Redundant', 'Compact', 'Compressed', 'Dynamic') (>= MariaDB 10.5)varchar(12) (<= MariaDB 10.4)	YES	NULL	InnoDB storage format (Compact, Redundant, Dynamic, or Compressed).
ZIP_PAGE_SIZE	int(11) unsigned	NO	0	For Compressed tables, the zipped page size.
SPACE_TYPE	enum('Single','System') (>= MariaDB 10.5)varchar(10) (<= MariaDB 10.4)	YES	NULL	

Flag

The flag field returns the dict_table_t::flags that correspond to the data dictionary record.

Bit	Description
0	Set if ROW_FORMAT is not REDUNDANT.
1 to 4	0, except for ROW_FORMAT=COMPRESSED, where they will determine the KEY_BLOCK_SIZE (the compressed page size).
5	Set for ROW_FORMAT=DYNAMIC or ROW_FORMAT=COMPRESSED.
6	Set if the DATA DIRECTORY attribute was present when the table was originally created.
7	Set if the page_compressed attribute is present.
8 to 11	Determine the page_compression_level.
12 13	Normally `00`, but `11` for “no-rollback tables”.

Note that the table flags returned here are not the same as tablespace flags (FSP_SPACE_FLAGS).

Example

```
SELECT * FROM information_schema.INNODB_SYS_TABLES LIMIT 2\G
*************************** 1. row ***************************
     TABLE_ID: 14
         NAME: SYS_DATAFILES
         FLAG: 0
       N_COLS: 5
        SPACE: 0
   ROW_FORMAT: Redundant
ZIP_PAGE_SIZE: 0
   SPACE_TYPE: System
*************************** 2. row ***************************
     TABLE_ID: 11
         NAME: SYS_FOREIGN
         FLAG: 0
       N_COLS: 7
        SPACE: 0
   ROW_FORMAT: Redundant
ZIP_PAGE_SIZE: 0
   SPACE_TYPE: System
2 rows in set (0.00 sec)
```

See Also

- InnoDB Data Dictionary Troubleshooting

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_TABLESPACES Table

The Information Schema `INNODB_SYS_TABLESPACES` table contains information about InnoDB tablespaces. Until MariaDB 10.5 it was based on the internal `SYS_TABLESPACES` table. This internal table was removed in MariaDB 10.6.0, so this Information Schema table has been repurposed to directly reflect the filesystem (fil_system.space_list).

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
SPACE	Unique InnoDB tablespace identifier.
NAME	Database and table name separated by a backslash, or the uppercase InnoDB system table name.

Column	Description
FLAG	1 if a DATA DIRECTORY option has been specified in CREATE TABLE, otherwise 0.
FILE_FORMAT	InnoDB file format.
ROW_FORMAT	InnoDB storage format used for this tablespace. If the Antelope file format is used, this value is always Compact or Redundant. When a table's checksum algorithm is full_crc32 (the default from MariaDB 10.5), the value can only be Compressed or NULL.
PAGE_SIZE	Page size in bytes for this tablespace. Until MariaDB 10.5.0, this was the value of the innodb_page_size variable. From MariaDB 10.6.0, contains the physical page size of a page (previously ZIP_PAGE_SIZE).
ZIP_PAGE_SIZE	Zip page size for this tablespace. Removed in MariaDB 10.6.0.
SPACE_TYPE	Tablespace type. Can be General for general tablespaces or Single for file-per-table tablespaces. Removed MariaDB 10.5.0.
FS_BLOCK_SIZE	File system block size.
FILE_SIZE	Maximum size of the file, uncompressed.
ALLOCATED_SIZE	Actual size of the file as per space allocated on disk.
FILENAME	Tablespace datafile path, previously part of the INNODB_SYS_DATAFILES table. Added in MariaDB 10.6.0.

Examples

```
DESC information_schema.innodb_sys_tablespaces;
+---------------+---------------------+------+-----+---------+-------+
| Field         | Type                | Null | Key | Default | Extra |
+---------------+---------------------+------+-----+---------+-------+
| SPACE         | int(11) unsigned    | NO   |     | 0       |       |
| NAME          | varchar(655)        | NO   |     |         |       |
| FLAG          | int(11) unsigned    | NO   |     | 0       |       |
| ROW_FORMAT    | varchar(22)         | YES  |     | NULL    |       |
| PAGE_SIZE     | int(11) unsigned    | NO   |     | 0       |       |
| ZIP_PAGE_SIZE | int(11) unsigned    | NO   |     | 0       |       |
| SPACE_TYPE    | varchar(10)         | YES  |     | NULL    |       |
| FS_BLOCK_SIZE | int(11) unsigned    | NO   |     | 0       |       |
| FILE_SIZE     | bigint(21) unsigned | NO   |     | 0       |       |
| ALLOCATED_SIZE | bigint(21) unsigned | NO   |     | 0       |       |
+---------------+---------------------+------+-----+---------+-------+
```

```
SELECT * FROM information_schema.INNODB_SYS_TABLESPACES LIMIT 2\G
*************************** 1. row ***************************
         SPACE: 2
          NAME: mysql/innodb_table_stats
          FLAG: 33
    ROW_FORMAT: Dynamic
     PAGE_SIZE: 16384
 ZIP_PAGE_SIZE: 0
    SPACE_TYPE: Single
 FS_BLOCK_SIZE: 4096
     FILE_SIZE: 98304
ALLOCATED_SIZE: 98304
*************************** 2. row ***************************
         SPACE: 3
          NAME: mysql/innodb_index_stats
          FLAG: 33
    ROW_FORMAT: Dynamic
     PAGE_SIZE: 16384
 ZIP_PAGE_SIZE: 0
    SPACE_TYPE: Single
 FS_BLOCK_SIZE: 4096
     FILE_SIZE: 98304
ALLOCATED_SIZE: 98304
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_TABLESTATS Table

The Information Schema `INNODB_SYS_TABLESTATS` table contains InnoDB status information. It can be used for developing new performance-related extensions, or high-level performance monitoring.

The `PROCESS` privilege is required to view the table.

Note that the MySQL InnoDB and Percona XtraDB versions of the tables differ (see XtraDB and InnoDB).

It contains the following columns:

Column	Description
TABLE_ID	Table ID, matching the INNODB_SYS_TABLES.TABLE_ID value.
SCHEMA	Database name (XtraDB only).
NAME	Table name, matching the INNODB_SYS_TABLES.NAME value.
STATS_INITIALIZED	Initialized if statistics have already been collected, otherwise Uninitialized.
NUM_ROWS	Estimated number of rows currently in the table. Updated after each statement modifying the data, but uncommitted transactions mean it may not be accurate.
CLUST_INDEX_SIZE	Number of pages on disk storing the clustered index, holding InnoDB table data in primary key order, or NULL if not statistics yet collected.
OTHER_INDEX_SIZE	Number of pages on disk storing secondary indexes for the table, or NULL if not statistics yet collected.
MODIFIED_COUNTER	Number of rows modified by statements modifying data.
AUTOINC	Auto_increment value.
REF_COUNT	Countdown to zero, when table metadata can be removed from the table cache. (InnoDB only)
MYSQL_HANDLES_OPENED	(XtraDB only).

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_SYS_VIRTUAL Table

The Information Schema `INNODB_SYS_VIRTUAL` table contains information about base columns of virtual columns. The `PROCESS` privilege is required to view the table.

It contains the following columns:

Field	Type	Null	Default
TABLE_ID	bigint(21) unsigned	NO	0
POS	int(11) unsigned	NO	0
BASE_POS	int(11) unsigned	NO	0

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_TABLESPACES_ENCRYPTION Table

The Information Schema `INNODB_TABLESPACES_ENCRYPTION` table contains metadata about encrypted InnoDB tablespaces. When you enable encryption for an InnoDB tablespace, an entry for the tablespace is added to this table. If you later disable encryption for the InnoDB tablespace, then the row still remains in this table, but the `ENCRYPTION_SCHEME` and `CURRENT_KEY_VERSION` columns will be set to `0`.

Viewing this table requires the PROCESS privilege. It contains the following columns:

Column	Description
SPACE	InnoDB tablespace ID.
NAME	Path to the InnoDB tablespace file, without the extension.
ENCRYPTION_SCHEME	Key derivation algorithm. Only 1 is currently used to represent an algorithm. If this value is 0, then the tablespace is unencrypted.
KEYSERVER_REQUESTS	Number of times InnoDB has had to request a key from the encryption key management plugin. The three most recent keys are cached internally.
MIN_KEY_VERSION	Minimum key version used to encrypt a page in the tablespace. Different pages may be encrypted with different key versions.
CURRENT_KEY_VERSION	Key version that will be used to encrypt pages. If this value is 0, then the tablespace is unencrypted.
KEY_ROTATION_PAGE_NUMBER	Page that a background encryption thread is currently rotating. If key rotation is not enabled, then the value will be NULL.
KEY_ROTATION_MAX_PAGE_NUMBER	When a background encryption thread starts rotating a tablespace, the field contains its current size. If key rotation is not enabled, then the value will be NULL.
CURRENT_KEY_ID	Key ID for the encryption key currently in use.
ROTATING_OR_FLUSHING	Current key rotation status. If this value is 1, then the background encryption threads are working on the tablespace. See MDEV-11738.

When the InnoDB system tablespace is encrypted, it is represented in this table with the special name: `innodb_system`.

Example

```
SELECT * FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME LIKE 'db_encrypt%';
+-------+-----------------------------------------------+------------------+--------------------+--------
---------+--------------------+-------------------------+-----------------------------+
| SPACE | NAME                                          | ENCRYPTION_SCHEME | KEYSERVER_REQUESTS |
MIN_KEY_VERSION | CURRENT_KEY_VERSION | KEY_ROTATION_PAGE_NUMBER | KEY_ROTATION_MAX_PAGE_NUMBER |
+-------+-----------------------------------------------+------------------+--------------------+--------
---------+--------------------+-------------------------+-----------------------------+
|    18 | db_encrypt/t_encrypted_existing_key           |                1 |                  1 |
1 |                  1 |                    NULL |                        NULL |
|    19 | db_encrypt/t_not_encrypted_existing_key       |                1 |                  0 |
1 |                  1 |                    NULL |                        NULL |
|    20 | db_encrypt/t_not_encrypted_non_existing_key   |                1 |                  0 |
4294967295 |          4294967295 |                    NULL |                        NULL |
|    21 | db_encrypt/t_default_encryption_existing_key |                1 |                  1 |
1 |                  1 |                    NULL |                        NULL |
|    22 | db_encrypt/t_encrypted_default_key            |                1 |                  1 |
1 |                  1 |                    NULL |                        NULL |
|    23 | db_encrypt/t_not_encrypted_default_key        |                1 |                  0 |
1 |                  1 |                    NULL |                        NULL |
|    24 | db_encrypt/t_defaults                         |                1 |                  1 |
1 |                  1 |                    NULL |                        NULL |
+-------+-----------------------------------------------+------------------+--------------------+--------
---------+--------------------+-------------------------+-----------------------------+
7 rows in set (0.00 sec)
```

See Also

- Encrypting Data for InnoDB / XtraDB
- Data at Rest Encryption
- Why Encrypt MariaDB Data?
- Encryption Key Management

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_TABLESPACES_SCRUBBING Table

> **Note:**
>
> This table was removed in MariaDB 10.5.2 - see MDEV-15528.

The Information Schema `INNODB_TABLESPACES_SCRUBBING` table contains data scrubbing information.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
SPACE	InnoDB table space id number.
NAME	Path to the table space file, without the extension.
COMPRESSED	The compressed page size, or zero if uncompressed.
LAST_SCRUB_COMPLETED	Date and time when the last scrub was completed, or NULL if never been performed.
CURRENT_SCRUB_STARTED	Date and time when the current scrub started, or NULL if never been performed.
CURRENT_SCRUB_ACTIVE_THREADS	Number of threads currently scrubbing the tablespace.
CURRENT_SCRUB_PAGE_NUMBER	Page that the scrubbing thread is currently scrubbing, or NULL if not enabled.

Column	Description
CURRENT_SCRUB_MAX_PAGE_NUMBER	When a scrubbing starts rotating a table space, the field contains its current size. NULL if not enabled.
ON_SSD	The field contains 1 when MariaDB detects that the table space is on a SSD based storage. 0 if not SSD or it could not be determined

Example

```
SELECT * FROM information_schema.INNODB_TABLESPACES_SCRUBBING LIMIT 1\G
*************************** 1. row ***************************
                       SPACE: 1
                        NAME: mysql/innodb_table_stats
                  COMPRESSED: 0
         LAST_SCRUB_COMPLETED: NULL
        CURRENT_SCRUB_STARTED: NULL
    CURRENT_SCRUB_PAGE_NUMBER: NULL
CURRENT_SCRUB_MAX_PAGE_NUMBER: 0
         ROTATING_OR_FLUSHING: 0
1 rows in set (0.00 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_TRX Table

The Information Schema `INNODB_TRX` table stores information about all currently executing InnoDB transactions.

It has the following columns:

Column	Description
TRX_ID	Unique transaction ID number.
TRX_STATE	Transaction execution state; one of RUNNING, LOCK WAIT, ROLLING BACK or COMMITTING.
TRX_STARTED	Time that the transaction started.
TRX_REQUESTED_LOCK_ID	If TRX_STATE is LOCK_WAIT, the INNODB_LOCKS.LOCK_ID value of the lock being waited on. NULL if any other state.
TRX_WAIT_STARTED	If TRX_STATE is LOCK_WAIT, the time the transaction started waiting for the lock, otherwise NULL.
TRX_WEIGHT	Transaction weight, based on the number of locked rows and the number of altered rows. To resolve deadlocks, lower weighted transactions are rolled back first. Transactions that have affected non-transactional tables are always treated as having a heavier weight.
TRX_MYSQL_THREAD_ID	Thread ID from the PROCESSLIST table (note that the locking and transaction information schema tables use a different snapshot from the processlist, so records may appear in one but not the other).
TRX_QUERY	SQL that the transaction is currently running.
TRX_OPERATION_STATE	Transaction's current state, or NULL.
TRX_TABLES_IN_USE	Number of InnoDB tables currently being used for processing the current SQL statement.
TRX_TABLES_LOCKED	Number of InnoDB tables that have row locks held by the current SQL statement.
TRX_LOCK_STRUCTS	Number of locks reserved by the transaction.
TRX_LOCK_MEMORY_BYTES	Total size in bytes of the memory used to hold the lock structures for the current transaction in memory.

Column	Description
TRX_ROWS_LOCKED	Number of rows the current transaction has locked. locked by this transaction. An approximation, and may include rows not visible to the current transaction that are delete-marked but physically present.
TRX_ROWS_MODIFIED	Number of rows added or changed in the current transaction.
TRX_CONCURRENCY_TICKETS	Indicates how much work the current transaction can do before being swapped out, see the innodb_concurrency_tickets system variable.
TRX_ISOLATION_LEVEL	Isolation level of the current transaction.
TRX_UNIQUE_CHECKS	Whether unique checks are on or off for the current transaction. Bulk data are a case where unique checks would be off.
TRX_FOREIGN_KEY_CHECKS	Whether foreign key checks are on or off for the current transaction. Bulk data are a case where foreign keys checks would be off.
TRX_LAST_FOREIGN_KEY_ERROR	Error message for the most recent foreign key error, or NULL if none.
TRX_ADAPTIVE_HASH_LATCHED	Whether the adaptive hash index is locked by the current transaction or not. One transaction at a time can change the adaptive hash index.
TRX_ADAPTIVE_HASH_TIMEOUT	Whether the adaptive hash index search latch should be relinquished immediately or reserved across all MariaDB calls. 0 if there is no contention on the adaptive hash index, in which case the latch is reserved until completion, otherwise counts down to zero and the latch is released after each row lookup.
TRX_IS_READ_ONLY	1 if a read-only transaction, otherwise 0.
TRX_AUTOCOMMIT_NON_LOCKING	1 if the transaction only contains this one statement, that is, a SELECT statement not using FOR UPDATE or LOCK IN SHARED MODE, and with autocommit on. If this and TRX_IS_READ_ONLY are both 1, the transaction can be optimized by the storrage engine to reduce some overheads

The table is often used in conjunction with the INNODB_LOCKS and INNODB_LOCK_WAITS tables to diagnose problematic locks and transactions.

XA transactions are not stored in this table. To see them, `XA RECOVER` can be used.

Example

```
-- session 1
START TRANSACTION;
UPDATE t SET id = 15 WHERE id = 10;

-- session 2
DELETE FROM t WHERE id = 10;

-- session 1
USE information_schema;
SELECT l.*, t.*
    FROM information_schema.INNODB_LOCKS l
    JOIN information_schema.INNODB_TRX t
        ON l.lock_trx_id = t.trx_id
    WHERE trx_state = 'LOCK WAIT' \G
*************************** 1. row ***************************
                   lock_id: 840:40:3:2
               lock_trx_id: 840
                 lock_mode: X
                 lock_type: RECORD
                lock_table: `test`.`t`
                lock_index: PRIMARY
                lock_space: 40
                 lock_page: 3
                  lock_rec: 2
                 lock_data: 10
                    trx_id: 840
                 trx_state: LOCK WAIT
               trx_started: 2019-12-23 18:43:46
     trx_requested_lock_id: 840:40:3:2
          trx_wait_started: 2019-12-23 18:43:46
                trx_weight: 2
       trx_mysql_thread_id: 46
                 trx_query: DELETE FROM t WHERE id = 10
       trx_operation_state: starting index read
         trx_tables_in_use: 1
         trx_tables_locked: 1
          trx_lock_structs: 2
     trx_lock_memory_bytes: 1136
           trx_rows_locked: 1
         trx_rows_modified: 0
   trx_concurrency_tickets: 0
       trx_isolation_level: REPEATABLE READ
         trx_unique_checks: 1
    trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
          trx_is_read_only: 0
trx_autocommit_non_locking: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema TEMP_TABLES_INFO Table

> **Note:**
>
> This table was removed and is no longer a part of MariaDB. See MDEV-12459 for an alternative.

The Information Schema `TEMP_TABLES_INFO` table contains information about active InnoDB temporary tables. All user and system-created temporary tables are reported when querying this table, with the exception of optimized internal temporary tables. The data is stored in memory.

Previously, InnoDB temp table metadata was rather stored in InnoDB system tables.

It has the following columns:

Column	Description
TABLE_ID	Table ID.
NAME	Table name.
N_COLS	Number of columns in the temporary table, including three hidden columns that InnoDB creates (DB_ROW_ID, DB_TRX_ID, and DB_ROLL_PTR).
SPACE	Numerical identifier for the tablespace identifier holding the temporary table. Compressed temporary tables are stored by default in separate per-table tablespaces in the temporary file directory. For non-compressed tables, the shared temporary table is named ibtmp1, found in the data directory. Always a non-zero value, and regenerated on server restart.
PER_TABLE_TABLESPACE	If TRUE, the temporary table resides in a separate per-table tablespace. If FALSE, it resides in the shared temporary tablespace.
IS_COMPRESSED	TRUE if the table is compressed.

The `PROCESS` privilege is required to view the table.

Examples

```
CREATE TEMPORARY TABLE t (i INT) ENGINE=INNODB;

SELECT * FROM INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO;
+----------+--------------+--------+-------+----------------------+---------------+
| TABLE_ID | NAME         | N_COLS | SPACE | PER_TABLE_TABLESPACE | IS_COMPRESSED |
+----------+--------------+--------+-------+----------------------+---------------+
|       39 | #sql1c93_3_1 |      4 |    64 | FALSE                | FALSE         |
+----------+--------------+--------+-------+----------------------+---------------+
```

Adding a compressed table:

```
SET GLOBAL innodb_file_format="Barracuda";

CREATE TEMPORARY TABLE t2 (i INT) ROW_FORMAT=COMPRESSED ENGINE=INNODB;

SELECT * FROM INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO;
+----------+--------------+--------+-------+----------------------+---------------+
| TABLE_ID | NAME         | N_COLS | SPACE | PER_TABLE_TABLESPACE | IS_COMPRESSED |
+----------+--------------+--------+-------+----------------------+---------------+
|       40 | #sql1c93_3_3 |      4 |    65 | TRUE                 | TRUE          |
|       39 | #sql1c93_3_1 |      4 |    64 | FALSE                | FALSE         |
+----------+--------------+--------+-------+----------------------+---------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema INNODB_UNDO_LOGS Table

> **Note:**
>
> The `INNODB_UNDO_LOGS` are a Percona enhancement, introduced in version MariaDB 5.5.27 and removed in 10.0.

The Information Schema `INNODB_UNDO_LOGS` table is a Percona enhancement, and is only available for XtraDB, not InnoDB (see XtraDB and InnoDB). It contains information about the InnoDB undo log, with each record being an undo log segment. It was removed in MariaDB 10.0.

It has the following columns:

Column	Description
TRX_ID	Unique transaction ID number, matching the value from the information_schema.INNODB_TRX table.

Column	Description
RSEG_ID	Rollback segment ID, matching the value from the information_schema.INNODB_RSEG table.
USEG_ID	Undo segment ID.
SEGMENT_TYPE	Indicates the operation type, for example INSERT or UPDATE.
STATE	Segment state; one of ACTIVE (contains active transaction undo log), CACHED, TO_FREE (insert undo segment can be freed), TO_PURGE (update undo segment won't be reused and can be purged when all undo data is removed) or PREPARED (segment of a prepared transaction).
SIZE	Size in pages of the segment.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema MyRocks Tables

- Information Schema ROCKSDB_CF_OPTIONS Table
- Information Schema ROCKSDB_CFSTATS Table
- Information Schema ROCKSDB_COMPACTION_STATS Table
- Information Schema ROCKSDB_DBSTATS Table
- Information Schema ROCKSDB_DDL Table
- Information Schema ROCKSDB_DEADLOCK Table
- Information Schema ROCKSDB_GLOBAL_INFO Table
- Information Schema ROCKSDB_INDEX_FILE_MAP Table
- Information Schema ROCKSDB_LOCKS Table
- Information Schema ROCKSDB_PERF_CONTEXT Table
- Information Schema ROCKSDB_PERF_CONTEXT_GLOBAL Table
- Information Schema ROCKSDB_SST_PROPS Table
- Information Schema ROCKSDB_TRX Table

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_CF_OPTIONS Table

The Information Schema `ROCKSDB_CF_OPTIONS` table is included as part of the MyRocks storage engine, and contains information about MyRocks column families.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
CF_NAME	Column family name.
OPTION_TYPE	
VALUE	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_CFSTATS Table

The Information Schema `ROCKSDB_CFSTATS` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
CF_NAME	
STAT_TYPE	
VALUE	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_COMPACTION_STATS Table

The Information Schema `ROCKSDB_COMPACTION_STATS` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
CF_NAME	
LEVEL	
TYPE	
VALUE	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_DBSTATS Table

The Information Schema `ROCKSDB_DBSTATS` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
STAT_TYPE	
VALUE	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_DDL Table

The Information Schema `ROCKSDB_DDL` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
TABLE_SCHEMA	
TABLE_NAME	
PARTITION_NAME	
INDEX_NAME	
COLUMN_FAMILY	

Column	Description
INDEX_NUMBER	
INDEX_TYPE	
KV_FORMAT_VERSION	
TTL_DURATION	
INDEX_FLAGS	
CF	
AUTO_INCREMENT	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_DEADLOCK Table

The Information Schema `ROCKSDB_DEADLOCK` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
DEADLOCK_ID	
TIMESTAMP	
TRANSACTION_ID	
CF_NAME	
WAITING_KEY	
LOCK_TYPE	
INDEX_NAME	
TABLE_NAME	
ROLLED_BACK	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_GLOBAL_INFO Table

The Information Schema `ROCKSDB_GLOBAL_INFO` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
TYPE	
NAME	
VALUE	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_INDEX_FILE_MAP Table

The Information Schema `ROCKSDB_INDEX_FILE_MAP` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
COLUMN_FAMILY	
INDEX_NUMBER	
SST_NAME	
NUM_ROWS	
DATA_SIZE	
ENTRY_DELETES	
ENTRY_SINGLEDELETES	
ENTRY_MERGES	
ENTRY_OTHERS	
DISTINCT_KEYS_PREFIX	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_LOCKS Table

The Information Schema `ROCKSDB_LOCKS` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
COLUMN_FAMILY_ID	
TRANSACTION_ID	
KEY	
MODE	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_PERF_CONTEXT Table

The Information Schema `ROCKSDB_PERF_CONTEXT` table is included as part of the MyRocks storage engine and includes per-table/partition counters .

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
TABLE_SCHEMA	
TABLE_NAME	
PARTITION_NAME	
STAT_TYPE	
VALUE	

Note: for multi-table queries, all counter increments are “billed” to the first table in the query:[1018] (https://github.com/facebook/mysql-5.6/issues/1018)

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_PERF_CONTEXT_GLOBAL Table

The Information Schema `ROCKSDB_PERF_CONTEXT_GLOBAL` table is included as part of the MyRocks storage engine and includes global counter information.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
STAT_TYPE	
VALUE	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_SST_PROPS Table

The Information Schema `ROCKSDB_SST_PROPS` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
SST_NAME	
COLUMN_FAMILY	
DATA_BLOCKS	
ENTRIES	
RAW_KEY_SIZE	
RAW_VALUE_SIZE	
DATA_BLOCK_SIZE	
INDEX_BLOCK_SIZE	
INDEX_PARTITIONS	
TOP_LEVEL_INDEX_SIZE	
FILTER_BLOCK_SIZE	
COMPRESSION_ALGO	
CREATION_TIME	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema ROCKSDB_TRX Table

The Information Schema `ROCKSDB_TRX` table is included as part of the MyRocks storage engine.

The `PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
TRANSACTION_ID	
STATE	
NAME	
WRITE_COUNT	

Column	Description
LOCK_COUNT	
TIMEOUT_SEC	
WAITING_KEY	
WAITING_COLUMN_FAMILY_ID	
IS_REPLICATION	
SKIP_TRX_API	
READ_ONLY	
HAS_DEADLOCK_DETECTION	
NUM_ONGOING_BULKLOAD	
THREAD_ID	
QUERY	

This page is licensed: CC BY-SA / Gnu FDL

Information Schema XtraDB Tables

- Information Schema CHANGED_PAGE_BITMAPS Table
- Information Schema XTRADB_INTERNAL_HASH_TABLES Table
- Information Schema XTRADB_READ_VIEW Table
- Information Schema XTRADB_RSEG Table

This page is licensed: CC BY-SA / Gnu FDL

Information Schema CHANGED_PAGE_BITMAPS Table

The Information Schema `CHANGED_PAGE_BITMAPS` table is a dummy table added to XtraDB with reset_table callback to allow `FLUSH NO_WRITE_TO_BINLOG CHANGED_PAGE_BITMAPS` to be called from `innobackupex`. It contains only one column, *dummy*.

For more information, see MDEV-7472.

The `PROCESS` privilege is required to view the table.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema XTRADB_INTERNAL_HASH_TABLES Table

The Information Schema `XTRADB_INTERNAL_HASH_TABLES` table contains InnoDB/XtraDB hash table memory usage information.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
INTERNAL_HASH_TABLE_NAME	Hash table name
TOTAL_MEMORY	Total memory
CONSTANT_MEMORY	Constant memory
VARIABLE_MEMORY	Variable memory

Example

```
SELECT * FROM information_schema.XTRADB_INTERNAL_HASH_TABLES;
+--------------------------------+-------------+-----------------+-----------------+
| INTERNAL_HASH_TABLE_NAME       | TOTAL_MEMORY | CONSTANT_MEMORY | VARIABLE_MEMORY |
+--------------------------------+-------------+-----------------+-----------------+
| Adaptive hash index            |      2217568 |         2213368 |            4200 |
| Page hash (buffer pool 0 only) |       139112 |          139112 |               0 |
| Dictionary Cache               |       613423 |          554768 |           58655 |
| File system                    |       816368 |          812272 |            4096 |
| Lock System                    |       332872 |          332872 |               0 |
| Recovery System                |            0 |               0 |               0 |
+--------------------------------+-------------+-----------------+-----------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Information Schema XTRADB_READ_VIEW Table

The Information Schema `XTRADB_READ_VIEW` table contains information about the oldest active transaction in the system.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
READ_VIEW_UNDO_NUMBER	
READ_VIEW_LOW_LIMIT_TRX_NUMBER	Highest transaction number at the time the view was created.
READ_VIEW_UPPER_LIMIT_TRX_ID	Highest transaction ID at the time the view was created. Should not see newer transactions with IDs equal to or greater than the value.
READ_VIEW_LOW_LIMIT_TRX_ID	Latest committed transaction ID at the time the oldest view was created. Should see all transactions with IDs equal to or less than the value.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema XTRADB_RSEG Table

The Information Schema `XTRADB_RSEG` table contains information about the XtraDB rollback segments.

The `PROCESS` privilege is required to view the table.

It has the following columns:

Column	Description
rseg_id	Rollback segment id.
space_id	Space where the segment is placed.
zip_size	Size in bytes of the compressed page size, or zero if uncompressed.
page_no	Page number of the segment header.
max_size	Maximum size in pages.
curr_size	Current size in pages.

The number of records will match the value set in the innodb_undo_logs variable (by default 128).

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema

See Performance Schema Digests.

Statement digests aggregate statistics for similar queries by removing specific data values, allowing you to identify performance patterns across statement types.

See Performance Schema Overview.

The Performance Schema is a feature for monitoring server performance that inspects internal execution details at a low level with minimal overhead.

See Performance Schema Status Variables.

This page lists status variables that track the internal health of the Performance Schema, such as counters for lost events due to memory constraints.

See Performance Schema System Variables.

Configure the Performance Schema using these system variables to control buffer sizes, set instrumentation limits, and enable specific consumers at startup.

See Performance Schema Tables.

Explore the tables in the performance_schema database, which expose granular metrics on server events, locks, threads, and I/O for detailed performance analysis.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema Digests

The Performance Schema digest is a normalized form of a statement, with the specific data values removed. It allows statistics to be gathered for similar kinds of statements.

For example:

```
SELECT * FROM customer WHERE age < 20
SELECT * FROM customer WHERE age < 30
```

With the data values removed, both of these statements normalize to:

```
SELECT * FROM customer WHERE age < ?
```

which is the digest text. The digest text is used to compute a digest. For example:

```
DIGEST_TEXT: SELECT * FROM `performance_schema` . `users`
DIGEST: 0f70cec4015f2a346df4ac0e9475d9f1
```

By contrast, the following two statements would not have the same digest as, while the data values are the same, they call upon different tables.

```
SELECT * FROM customer1 WHERE age < 20
SELECT * FROM customer2 WHERE age < 20
```

The digest text is limited to 1024 bytes. Queries exceeding this limit are truncated with '…', meaning that long queries that would otherwise have different digests may share the same digest.

Digest information is used in a number of performance scheme tables. These include

- events_statements_current
- events_statements_history

- events_statements_history_long
- events_statements_summary_by_digest (a summary table by schema and digest)

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema Overview

The Performance Schema is a feature for monitoring server performance.

Introduction

It is implemented as a storage engine, and so will appear in the list of storage engines available.

```
SHOW ENGINES;
+--------------------+---------+----------------------------------+--------------+------+------------+
| Engine             | Support | Comment                          | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------+--------------+------+------------+
| ...                |         |                                  |              |      |            |
| PERFORMANCE_SCHEMA | YES     | Performance Schema               | NO           | NO   | NO         |
| ...                |         |                                  |              |      |            |
+--------------------+---------+----------------------------------+--------------+------+------------+
```

However, `performance_schema` is not a regular storage engine for storing data, it's a mechanism for implementing the Performance Schema feature.

The storage engine contains a database called `performance_schema`, which in turn consists of a number of tables that can be queried with regular SQL statements, returning specific performance information.

```
USE performance_schema
```

```
SHOW TABLES;
+----------------------------------------------------+
| Tables_in_performance_schema                       |
+----------------------------------------------------+
| accounts                                           |
...
| users                                              |
+----------------------------------------------------+
80 rows in set (0.00 sec)
```

See List of Performance Schema Tables for a full list and links to detailed descriptions of each table. From MariaDB 10.5, there are 80 Performance Schema tables.

Activating the Performance Schema

The performance schema is disabled by default for performance reasons. You can check its current status by looking at the value of the performance_schema system variable.

```
SHOW VARIABLES LIKE 'performance_schema';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| performance_schema | ON    |
+--------------------+-------+
```

The performance schema cannot be activated at runtime - it must be set when the server starts by adding the following line in your `my.cnf` configuration file.

```
performance_schema=ON
```

From MariaDB 10.5, some memory is allocated dynamically, depending on load, number of connections, number of tables open etc.

Enabling the Performance Schema

You need to set up all consumers (starting collection of data) and instrumentations (what to collect):

```
UPDATE performance_schema.setup_consumers SET ENABLED = 'YES';
UPDATE performance_schema.setup_instruments SET ENABLED = 'YES', TIMED = 'YES';
```

You can decide what to enable/disable with `WHERE NAME like "%what_to_enable"`;
You can disable instrumentations by setting `ENABLED` to `"NO"`.

You can also do this in your my.cnf file.
The following enables all instrumentation of all stages (computation units) in MariaDB:

```
[mysqld]
performance_schema=ON
performance-schema-instrument='stage/%=ON'
performance-schema-consumer-events-stages-current=ON
performance-schema-consumer-events-stages-history=ON
performance-schema-consumer-events-stages-history-long=ON
```

Listing Performance Schema Variables

```
SHOW VARIABLES LIKE "perf%";
+----------------------------------------------------+-------+
| Variable_name                                      | Value |
+----------------------------------------------------+-------+
| performance_schema                                 | ON    |
...
| performance_schema_users_size                      | 100   |
+----------------------------------------------------+-------+
```

See Performance Schema System Variables for a full list of available system variables.

Note that the “consumer” events are not shown on this list, as they are only available as options, not as system variables, and they can only be enabled at startup.

Column Comments

From MariaDB 10.7.1, comments have been added to table columns in the Performance Schema. These can be viewed with, for example:

```
SELECT column_name, column_comment FROM information_schema.columns
  WHERE table_schema='performance_schema' AND table_name='file_instances';
...
*************************** 2. row ***************************
   column_name: EVENT_NAME
column_comment: Instrument name associated with the file.
*************************** 3. row ***************************
   column_name: OPEN_COUNT
column_comment: Open handles on the file. A value of greater than zero means
                that the file is currently open.
...
```

See Also

- Performance schema options
- SHOW ENGINE STATUS
- SHOW PROFILE``

- ANALYZE STATEMENT
- Performance schema in MySQL 5.6. All things here should also work for MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema Status Variables

This page documents status variables related to the Performance Schema. See Server Status Variables for a complete list of status variables that can be viewed with SHOW STATUS.

See also the Full list of MariaDB options, system and status variables.

`Performance_schema_accounts_lost`

- Description: Number of times a row could not be added to the performance schema accounts table due to it being full. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_cond_classes_lost`

- Description: Number of condition instruments that could not be loaded.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_cond_instances_lost`

- Description: Number of instances a condition object could not be created. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_digest_lost`

- Description: The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_file_classes_lost`

- Description: Number of file instruments that could not be loaded.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_file_handles_lost`

- Description: Number of instances a file object could not be opened. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_file_instances_lost`

- Description: Number of instances a file object could not be created. The global value can be flushed by FLUSH STATUS.

- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_hosts_lost`

- Description: Number of times a row could not be added to the performance schema hosts table due to it being full. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_index_stat_lost`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.5.2

`Performance_schema_locker_lost`

- Description: Number of events not recorded, due to either being recursive, or having a deeper nested events stack than the implementation limit. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_memory_classes_lost`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.5.2

`Performance_schema_metadata_lock_lost`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.5.2

`Performance_schema_mutex_classes_lost`

- Description: Number of mutual exclusion instruments that could not be loaded.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_mutex_instances_lost`

- Description: Number of instances a mutual exclusion object could not be created. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_nested_statement_lost`

- Description:

- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.5.2

`Performance_schema_prepared_statements_lost`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.5.2

`Performance_schema_program_lost`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.5.2

`Performance_schema_rwlock_classes_lost`

- Description: Number of read/write lock instruments that could not be loaded.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_rwlock_instances_lost`

- Description: Number of instances a read/write lock object could not be created. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_session_connect_attrs_lost`

- Description: Number of connections for which connection attribute truncation has occurred. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_socket_classes_lost`

- Description:
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_socket_instances_lost`

- Description: Number of instances a socket object could not be created. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_stage_classes_lost`

- Description: Number of stage event instruments that could not be loaded. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_statement_classes_lost`

- Description: Number of statement instruments that could not be loaded. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_table_handles_lost`

- Description: Number of instances a table object could not be opened. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_table_instances_lost`

- Description: Number of instances a table object could not be created. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_table_lock_stat_lost`

- Description:
- Scope: Global, Session
- Data Type: `numeric`
- Introduced: MariaDB 10.5.2

`Performance_schema_thread_classes_lost`

- Description: Number of thread instruments that could not be loaded.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_thread_instances_lost`

- Description: Number of instances thread object could not be created. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

`Performance_schema_users_lost`

- Description: Number of times a row could not be added to the performance schema users table due to it being full. The global value can be flushed by FLUSH STATUS.
- Scope: Global, Session
- Data Type: `numeric`

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema System Variables

The following variables are used with MariaDB's Performance Schema. See Performance Schema Options for Performance Schema options that are not system variables. See Server System Variables for a complete list of system variables and instructions on setting them.

See also the Full list of MariaDB options, system and status variables.

`performance_schema`

- Description: If set to `1` (`0` is default), enables the Performance Schema
- Command line: `--performance-schema=#`
- Scope: Global
- Dynamic: No
- Data Type: `boolean`
- Default Value: `OFF`

`performance_schema_accounts_size`

- Description: Maximum number of rows in the performance_schema.accounts table. If set to 0, the Performance Schema will not store statistics in the accounts table. Use `-1` (the default) for automated sizing.
- Command line: `--performance-schema-accounts-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_digests_size`

- Description: Maximum number of rows that can be stored in the events_statements_summary_by_digest table. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-digests-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_events_stages_history_long_size`

- Description: Number of rows in the events_stages_history_long table. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-events-stages-history-long-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_events_stages_history_size`

- Description: Number of rows per thread in the events_stages_history table. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-events-stages-history-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1024`

`performance_schema_events_statements_history_long_size`

- Description: Number of rows in the events_statements_history_long table. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-events-statements-history-long-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_events_statements_history_size`

- Description: Number of rows per thread in the events_statements_history table. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-events-statements-history-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1024`

`performance_schema_events_transactions_history_long_size`

- Description: Number of rows in events_transactions_history_long table. Use `0` to disable, `-1` for automated sizing.
- Command line: `--performance-schema-events-transactions-history-long-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`
- Introduced: MariaDB 10.5.2

`performance_schema_events_transactions_history_size`

- Description:Number of rows per thread in events_transactions_history. Use 0 to disable, -1 for automated sizing.
- Command line: `--performance-schema-events-transactions-history-size=#`
- Scope: Global

- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1024`
- Introduced: MariaDB 10.5.2

`performance_schema_events_waits_history_long_size`

- Description: Number of rows contained in the events_waits_history_long table. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-events-waits-history-long-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_events_waits_history_size`

- Description: Number of rows per thread contained in the events_waits_history table. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-events-waits-history-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1024`

`performance_schema_hosts_size`

- Description: Number of rows stored in the hosts table. If set to zero, no connection statistics are kept for the hosts table. `-1` (the default) for automated sizing.
- Command line: `--performance-schema-hosts-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_max_cond_classes`

- Description: Specifies the maximum number of condition instruments.
- Command line: `--performance-schema-max-cond-classes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `90` (>= MariaDB 10.5.1), `80` (<= MariaDB 10.5.0)
- Range: `0` to `256`

`performance_schema_max_cond_instances`

- Description: Specifies the maximum number of instrumented condition objects. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-max-cond-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_max_digest_length`

- Description: Maximum length considered for digest text, when stored in performance_schema tables.
- Command line: `--performance-schema-max-digest-length=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1024`
- Range: `0` to `1048576`

`performance_schema_max_file_classes`

- Description: Specifies the maximum number of file instruments.
- Command line: `--performance-schema-max-file-classes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `80` (>= MariaDB 10.5.2), `50` (<= MariaDB 10.5.1)
- Range: `0` to `256`

`performance_schema_max_file_handles`

- Description: Specifies the maximum number of opened file objects. Should always be higher than open_files_limit.
- Command line: `--performance-schema-max-file-handles=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `32768`
- Range: `-1` to `32768`

`performance_schema_max_file_instances`

- Description: Specifies the maximum number of instrumented file objects. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-max-file-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`

- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_max_index_stat`

- Description: Maximum number of index statistics for instrumented tables. Use 0 to disable, -1 for automated scaling.
- Command line: `--performance-schema-max-index-stat=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`
- Introduced: MariaDB 10.5.2

`performance_schema_max_memory_classes`

- Description: Maximum number of memory pool instruments.
- Command line: `--performance-schema-max-memory-classes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `320`
- Range: `0` to `1024`
- Introduced: MariaDB 10.5.2

`performance_schema_max_metadata_locks`

- Description: Maximum number of Performance Schema metadata locks. Use 0 to disable, -1 for automated scaling.
- Command line: `--performance-schema-max-metadata-locks=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `104857600`
- Introduced: MariaDB 10.5.2

`performance_schema_max_mutex_classes`

- Description: Specifies the maximum number of mutex instruments.
- Command line: `--performance-schema-max-mutex-classes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `210` (>= MariaDB 10.5.2), `200` (<= MariaDB 10.5.1)
- Range: `0` to `256`

`performance_schema_max_mutex_instances`

- Description: Specifies the maximum number of instrumented mutex instances. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-max-mutex-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `104857600`

`performance_schema_max_prepared_statement_instances`

- Description: Maximum number of instrumented prepared statements. Use 0 to disable, -1 for automated scaling.
- Command line: `--performance-schema-max-prepared-statement-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`
- Introduced: MariaDB 10.5.2

`performance_schema_max_program_instances`

- Description: Maximum number of instrumented programs. Use 0 to disable, -1 for automated scaling.
- Command line: `--performance-schema-max-program-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`
- Introduced: MariaDB 10.5.2

`performance_schema_max_rwlock_classes`

- Description: Specifies the maximum number of rwlock instruments.
- Command line: `--performance-schema-max-rwlock-classes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `50` (>= MariaDB 10.5.2), `40` (<= MariaDB 10.5.1)
- Range: `0` to `256`

`performance_schema_max_rwlock_instances`

- Description: Specifies the maximum number of instrumented rwlock objects. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-max-rwlock-instances=#`
- Scope: Global
- Dynamic: No

- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `104857600`

`performance_schema_max_socket_classes`

- Description: Specifies the maximum number of socket instruments.
- Command line: `--performance-schema-max-socket-classes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `10`
- Range: `0` to `256`

`performance_schema_max_socket_instances`

- Description: Specifies the maximum number of instrumented socket objects. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-max-socket-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_max_sql_text_length`

- Description: Maximum length of displayed sql text.
- Command line: `--performance-schema-max-sql-text-length=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `1024`
- Range: `0` to `1048576`
- Introduced: MariaDB 10.5.2

`performance_schema_max_stage_classes`

- Description: Specifies the maximum number of stage instruments.
- Command line: `--performance-schema-max-stage-classes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `160`
- Range: `0` to `256`

`performance_schema_max_statement_classes`

- Description: Specifies the maximum number of statement instruments. Automatically calculated at server build based on the number of available statements. Should be left as either autosized or disabled, as changing to any

positive value has no benefit and will most likely allocate unnecessary memory. Setting to zero disables all statement instrumentation, and no memory will be allocated for this purpose.

- Command line: `--performance-schema-max-statement-classes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: Autosized (see description)
- Range: `0` to `256`

`performance_schema_max_statement_stack`

- Description: Number of rows per thread in EVENTS_STATEMENTS_CURRENT.
- Command line: `--performance-schema-max-statement-stack=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `10`
- Range: `1` to `256`
- Introduced: MariaDB 10.5.2

`performance_schema_max_table_handles`

- Description: Specifies the maximum number of opened table objects. `0` for disabling, `-1` (the default) for automated sizing. See also the Performance Schema table_handles table.
- Command line: `--performance-schema-max-table-handles=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_max_table_instances`

- Description: Specifies the maximum number of instrumented table objects. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-max-table-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_max_table_lock_stat`

- Description: Maximum number of lock statistics for instrumented tables. Use 0 to disable, -1 for automated scaling.
- Command line: `--performance-schema-max-table-lock-stat=#`
- Scope: Global
- Dynamic: No

- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`
- Introduced: MariaDB 10.5.2

`performance_schema_max_thread_classes`

- Description: Specifies the maximum number of thread instruments.
- Command line: `--performance-schema-max-thread-classes=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `50`
- Range: `0` to `256`

`performance_schema_max_thread_instances`

- Description: Specifies how many of the running server threads (see max_connections and max_delayed_threads) can be instrumented. Should be greater than the sum of max_connections and max_delayed_threads. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-max-thread-instances=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_session_connect_attrs_size`

- Description: Per thread preallocated memory for holding connection attribute strings. Incremented if the strings are larger than the reserved space. `0` for disabling, `-1` (the default) for automated sizing.
- Command line: `--performance-schema-session-connect-attrs-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

`performance_schema_setup_actors_size`

- Description: The maximum number of rows to store in the performance schema setup_actors table. `-1` (from MariaDB 10.5.2) denotes automated sizing.
- Command line: `--performance-schema-setup-actors-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1` (>= MariaDB 10.5.2), `100` (<= MariaDB 10.5.1)
- Range: `-1` to `1024` (>= MariaDB 10.5.2), `0` to `1024` (<= MariaDB 10.5.1)

`performance_schema_setup_objects_size`

- Description: The maximum number of rows that can be stored in the performance schema setup_objects table. `-1` (from MariaDB 10.5.2) denotes automated sizing.
- Command line: `--performance-schema-setup-objects-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1` (>= MariaDB 10.5.2), `100` (<= MariaDB 10.5.1)
- Range: `-1` to `1048576` (>= MariaDB 10.5.2), `0` to `1048576` (<= MariaDB 10.5.1)

`performance_schema_users_size`

- Description: Number of rows in the performance_schema.users table. If set to 0, the Performance Schema will not store connection statistics in the users table. `-1` (the default) for automated sizing.
- Command line: `--performance-schema-users-size=#`
- Scope: Global
- Dynamic: No
- Data Type: `numeric`
- Default Value: `-1`
- Range: `-1` to `1048576`

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema Tables

- List of Performance Schema Tables
- Performance Schema accounts Table
- Performance Schema cond_instances Table
- Performance Schema events_stages_current Table
- Performance Schema events_stages_history Table
- Performance Schema events_stages_history_long Table
- Performance Schema events_stages_summary_by_account_by_event_name Table
- Performance Schema events_stages_summary_by_host_by_event_name Table
- Performance Schema events_stages_summary_by_thread_by_event_name Table
- Performance Schema events_stages_summary_by_user_by_event_name Table
- Performance Schema events_stages_summary_global_by_event_name Table
- Performance Schema events_statements_current Table
- Performance Schema events_statements_history Table
- Performance Schema events_statements_history_long Table
- Performance Schema events_statements_summary_by_account_by_event_name Table
- Performance Schema events_statements_summary_by_digest Table
- Performance Schema events_statements_summary_by_host_by_event_name Table
- Performance Schema events_statements_summary_by_program Table
- Performance Schema events_statements_summary_by_thread_by_event_name Table
- Performance Schema events_statements_summary_by_user_by_event_name Table
- Performance Schema events_statements_summary_global_by_event_name Table
- Performance Schema events_transactions_current Table

- Performance Schema events_transactions_history Table
- Performance Schema events_transactions_history_long Table
- Performance Schema events_transactions_summary_by_account_by_event_name Table
- Performance Schema events_transactions_summary_by_host_by_event_name Table
- Performance Schema events_transactions_summary_by_thread_by_event_name Table
- Performance Schema events_transactions_summary_by_user_by_event_name Table
- Performance Schema events_transactions_summary_global_by_event_name Table
- Performance Schema events_waits_current Table
- Performance Schema events_waits_history Table
- Performance Schema events_waits_history_long Table
- Performance Schema events_waits_summary_by_account_by_event_name Table
- Performance Schema events_waits_summary_by_host_by_event_name Table
- Performance Schema events_waits_summary_by_instance Table
- Performance Schema events_waits_summary_by_thread_by_event_name Table
- Performance Schema events_waits_summary_by_user_by_event_name Table
- Performance Schema events_waits_summary_global_by_event_name Table
- Performance Schema file_instances Table
- Performance Schema file_summary_by_event_name Table
- Performance Schema file_summary_by_instance Table
- Performance Schema global_status Table
- Performance Schema host_cache Table
- Performance Schema hosts Table
- Performance Schema memory_summary_by_account_by_event_name Table
- Performance Schema memory_summary_by_host_by_event_name Table
- Performance Schema memory_summary_by_thread_by_event_name Table
- Performance Schema memory_summary_by_user_by_event_name Table
- Performance Schema memory_summary_global_by_event_name Table
- Performance Schema metadata_locks Table
- Performance Schema mutex_instances Table
- Performance Schema objects_summary_global_by_type Table
- Performance Schema performance_timers Table
- Performance Schema prepared_statements_instances Table
- Performance Schema replication_applier_configuration Table
- Performance Schema replication_applier_status Table
- Performance Schema replication_applier_status_by_coordinator Table
- Performance Schema replication_applier_status_by_worker Table
- Performance Schema replication_connection_configuration Table
- Performance Schema rwlock_instances Table
- Performance Schema session_account_connect_attrs Table
- Performance Schema session_connect_attrs Table
- Performance Schema session_status Table
- Performance Schema setup_actors Table
- Performance Schema setup_consumers Table
- Performance Schema setup_instruments Table
- Performance Schema setup_objects Table

- Performance Schema setup_timers Table
- Performance Schema socket_instances Table
- Performance Schema socket_summary_by_event_name Table
- Performance Schema socket_summary_by_instance Table
- Performance Schema status_by_account Table
- Performance Schema status_by_host Table
- Performance Schema status_by_thread Table
- Performance Schema status_by_user Table
- Performance Schema table_handles Table
- Performance Schema table_io_waits_summary_by_index_usage Table
- Performance Schema table_io_waits_summary_by_table Table
- Performance Schema table_lock_waits_summary_by_table Table
- Performance Schema threads Table
- Performance Schema users Table
- Performance Schema user_variables_by_thread Table

This page is licensed: CC BY-SA / Gnu FDL

List of Performance Schema Tables

Below is a list of all Performance Schema tables as well as a brief description of each of them.

Table	Description
accounts	Client account connection statistics.
cond_instances	Synchronization object instances.
events_stages_current	Current stage events.
events_stages_history	Ten most recent stage events per thread.
events_stages_history_long	Ten thousand most recent stage events.
events_stages_summary_by_account_by_event_name	Summarized stage events per account and event name.
events_stages_summary_by_host_by_event_name	Summarized stage events per host and event name.
events_stages_summary_by_thread_by_event_name	Summarized stage events per thread and event name.
events_stages_summary_by_user_by_event_name	Summarized stage events per user name and event name.
events_stages_summary_global_by_event_name	Summarized stage events per event name.
events_statements_current	Current statement events.
events_statements_history	Ten most recent events per thread.
events_statements_history_long	Ten thousand most recent stage events.
events_statements_summary_by_account_by_event_name	Summarized statement events per account and event name.
events_statements_summary_by_digest	Summarized statement events by scheme and digest.
events_statements_summary_by_host_by_event_name	Summarized statement events by host and event name.
events_statements_summary_by_program	Events for a particular stored program.
events_statements_summary_by_thread_by_event_name	Summarized statement events by thread and event name.
events_statements_summary_by_user_by_event_name	Summarized statement events by user and event name.
events_statements_summary_global_by_event_name	Summarized statement events by event name.
events_transactions_current	Current transaction events for each thread.
events_transactions_history	Most recent completed transaction events for each thread.

Table	Description
events_transactions_history_long	Most recent completed transaction events that have ended globally.
events_transactions_summary_by_account_by_event_name	Transaction events aggregated by account and event.
events_transactions_summary_by_host_by_event_name	Transaction events aggregated by host and event..
events_transactions_summary_by_thread_by_event_name	Transaction events aggregated by thread and event..
events_transactions_summary_by_user_by_event_name	Transaction events aggregated by user and event..
events_transactions_summary_global_by_event_name	Transaction events aggregated by event name.
events_waits_current	Current wait events.
events_waits_history	Ten most recent wait events per thread.
events_waits_history_long	Ten thousand most recent wait events per thread.
events_waits_summary_by_account_by_event_name	Summarized wait events by account and event name.
events_waits_summary_by_host_by_event_name	Summarized wait events by host and event name.
events_waits_summary_by_instance	Summarized wait events by instance.
events_waits_summary_by_thread_by_event_name	Summarized wait events by thread and event name.
events_waits_summary_by_user_by_event_name	Summarized wait events by user and event name.
events_waits_summary_global_by_event_name	Summarized wait events by event name.
file_instances	Seen files.
file_summary_by_event_name	File events summarized by event name.
file_summary_by_instance	File events summarized by instance.
global_status	Global status variables and values.
host_cache	Host and IP information.
hosts	Connections by host.
memory_summary_by_account_by_event_name	Memory usage statistics aggregated by account and event.
memory_summary_by_host_by_event_name	Memory usage statistics aggregated by host. and event.
memory_summary_by_thread_by_event_name	Memory usage statistics aggregated by thread and event..
memory_summary_by_user_by_event_name	Memory usage statistics aggregated by user and event..
memory_summary_global_by_event_name	Memory usage statistics aggregated by event.
metadata_locks	Metadata locks.
mutex_instances	Seen mutexes.
objects_summary_global_by_type	Object wait events.
performance_timers	Available event timers.
prepared_statements_instances	Aggregate statistics of prepared statements.
replication_applier_configuration	Configuration settings affecting replica transactions.
replication_applier_status	General transaction execution status on the replica.
replication_applier_status_by_coordinator	Coordinator thread specific information.
replication_applier_status_by_worker	Replica worker thread specific information.
replication_connection_configuration	Rreplica's configuration settings used for connecting to the primary.
rwlock_instances	Seen read-write locks.
session_account_connect_attrs	Current session connection attributes.
session_connect_attrs	All session connection attributes.

Table	Description
session_status	Session status variables and values.
setup_actors	Details on foreground thread monitoring.
setup_consumers	Consumers for which event information is stored.
setup_instruments	Instrumented objects for which events are collected.
setup_objects	Objects to be monitored.
setup_timers	Currently selected event timers.
socket_instances	Active connections.
socket_summary_by_event_name	Timer and byte count statistics by socket instrument.
socket_summary_by_instance	Timer and byte count statistics by socket instance.
status_by_account	Status variable info by host/user account.
status_by_host	Status variable info by host.
status_by_thread	Status variable info about active foreground threads.
status_by_user	Status variable info by user.
table_handles	Table lock information.
table_io_waits_summary_by_index_usage	Aggregate table I/O wait events by index.
table_io_waits_summary_by_table	Aggregate table I/O wait events by table.
table_lock_waits_summary_by_table	Aggregate table lock wait events by table.
threads	Server thread information.
user_variables_by_thread	User-defined variables by thread.
users	Connection statistics by user.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema accounts Table

Description

Each account that connects to the server is stored as a row in the accounts table, along with current and total connections.

The table size is determined at startup by the value of the performance_schema_accounts_size system variable. If this is set to 0, account statistics will be disabled.

Column	Description
USER	The connection's client user name for the connection, or NULL if an internal thread.
HOST	The connection client's host name, or NULL if an internal thread.
CURRENT_CONNECTIONS	Current connections for the account.
TOTAL_CONNECTIONS	Total connections for the account.

The `USER` and `HOST` values shown here are the username and host used for user connections, not the patterns used to check permissions.

Example

```
SELECT * FROM performance_schema.accounts;
+------------------+-----------+---------------------+-------------------+
| USER             | HOST      | CURRENT_CONNECTIONS | TOTAL_CONNECTIONS |
+------------------+-----------+---------------------+-------------------+
| root             | localhost |                   1 |                 2 |
| NULL             | NULL      |                  20 |                23 |
| debian-sys-maint | localhost |                   0 |                35 |
+------------------+-----------+---------------------+-------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema cond_instances Table

Description

The `cond_instances` table lists all conditions while the server is executing. A condition, or instrumented condition object, is an internal code mechanism used for signalling that a specific event has occurred so that any threads waiting for this condition can continue.

The maximum number of conditions stored in the performance schema is determined by the performance_schema_max_cond_instances system variable.

Column	Description
NAME	Client user name for the connection, or NULL if an internal thread.
OBJECT_INSTANCE_BEGIN	Address in memory of the instrumented condition.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_stages_current Table

The `events_stages_current` table contains current stage events, with each row being a record of a thread and its most recent stage event.

The table contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if the event has not ended or timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if the event has not ended or timing is not collected.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. One of transaction, statement, stage or wait.

It is possible to empty this table with a `TRUNCATE TABLE` statement.

The related tables, events_stages_history and events_stages_history_long derive their values from the current events.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_stages_history Table

The `events_stages_history` table by default contains the ten most recent completed stage events per thread. This number can be adjusted by setting the performance_schema_events_stages_history_size system variable when the server starts up.

The table structure is identical to the events_stage_current table structure, and contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. One of transaction, statement, stage or wait.

It is possible to empty this table with a `TRUNCATE TABLE` statement.

events_stages_current and events_stages_history_long are related tables.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_stages_history_long Table

The `events_stages_history_long` table by default contains the ten thousand most recent completed stage events. This number can be adjusted by setting the performance_schema_events_stages_history_long_size system variable when the server starts up.

The table structure is identical to the `events_stage_current` table structure, and contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.

Column	Description
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. One of transaction, statement, stage or wait.

It is possible to empty this table with a `TRUNCATE TABLE` statement.

events_stages_current and events_stages_history are related tables.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_stages_summary_by_account_by_event_name Table

The table lists stage events, summarized by account and event name.

It contains the following columns:

Column	Description
USER	User. Used together with HOST and EVENT_NAME for grouping events.
HOST	Host. Used together with USER and EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER and HOST for grouping events.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.
AVG_TIMER_WAIT	Average wait time of the timed summarized events.
MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.

Example

```
SELECT * FROM events_stages_summary_by_account_by_event_name\G
...
*************************** 325. row ***************************
          USER: NULL
          HOST: NULL
    EVENT_NAME: stage/sql/Waiting for event metadata lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 326. row ***************************
          USER: NULL
          HOST: NULL
    EVENT_NAME: stage/sql/Waiting for commit lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 327. row ***************************
          USER: NULL
          HOST: NULL
    EVENT_NAME: stage/aria/Waiting for a resource
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_stages_summary_by_host_by_event_name Table

The table lists stage events, summarized by host and event name.

It contains the following columns:

Column	Description
HOST	Host. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with HOST for grouping events.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.
AVG_TIMER_WAIT	Average wait time of the timed summarized events.
MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.

Example

```
SELECT * FROM events_stages_summary_by_host_by_event_name\G
...
*************************** 216. row ***************************
          HOST: NULL
    EVENT_NAME: stage/sql/Waiting for event metadata lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 217. row ***************************
          HOST: NULL
    EVENT_NAME: stage/sql/Waiting for commit lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 218. row ***************************
          HOST: NULL
    EVENT_NAME: stage/aria/Waiting for a resource
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_stages_summary_by_thread_by_event_name Table

The table lists stage events, summarized by thread and event name.

It contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_NAME uniquely identifies the row.
EVENT_NAME	Event name. Used together with THREAD_ID for grouping events.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.
AVG_TIMER_WAIT	Average wait time of the timed summarized events.
MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.

Example

```
SELECT * FROM events_stages_summary_by_thread_by_event_name\G
...
*************************** 2287. row ***************************
     THREAD_ID: 64
    EVENT_NAME: stage/sql/Waiting for event metadata lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 2288. row ***************************
     THREAD_ID: 64
    EVENT_NAME: stage/sql/Waiting for commit lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 2289. row ***************************
     THREAD_ID: 64
    EVENT_NAME: stage/aria/Waiting for a resource
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_stages_summary_by_user_by_event_name Table

The table lists stage events, summarized by user and event name.

It contains the following columns:

Column	Description
USER	User. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER for grouping events.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.
AVG_TIMER_WAIT	Average wait time of the timed summarized events.
MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.

Example

```
SELECT * FROM events_stages_summary_by_user_by_event_name\G
...
*************************** 325. row ***************************
          USER: NULL
    EVENT_NAME: stage/sql/Waiting for event metadata lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 326. row ***************************
          USER: NULL
    EVENT_NAME: stage/sql/Waiting for commit lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 327. row ***************************
          USER: NULL
    EVENT_NAME: stage/aria/Waiting for a resource
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_stages_summary_global_by_event_name Table

The table lists stage events, summarized by thread and event name.

It contains the following columns:

Column	Description
EVENT_NAME	Event name.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.
AVG_TIMER_WAIT	Average wait time of the timed summarized events.
MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.

Example

```
SELECT * FROM events_stages_summary_global_by_event_name\G
...
*************************** 106. row ***************************
    EVENT_NAME: stage/sql/Waiting for trigger metadata lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 107. row ***************************
    EVENT_NAME: stage/sql/Waiting for event metadata lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 108. row ***************************
    EVENT_NAME: stage/sql/Waiting for commit lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 109. row ***************************
    EVENT_NAME: stage/aria/Waiting for a resource
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_current Table

The `events_statements_current` table contains current statement events, with each row being a record of a thread and its most recent statement event.

The table contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if the event has not ended or timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if the event has not ended or timing is not collected.
LOCK_TIME	Time in picoseconds spent waiting for locks. The time is calculated in microseconds but stored in picoseconds for compatibility with other timings.
SQL_TEXT	The SQL statement, or NULL if the command is not associated with an SQL statement.
DIGEST	Statement digest.
DIGEST_TEXT	Statement digest text.

Column	Description
CURRENT_SCHEMA	Statement's default database for the statement, or NULL if there was none.
OBJECT_SCHEMA	Reserved, currently NULL
OBJECT_NAME	Reserved, currently NULL
OBJECT_TYPE	Reserved, currently NULL
OBJECT_INSTANCE_BEGIN	Address in memory of the statement object.
MYSQL_ERRNO	Error code. See MariaDB Error Codes for a full list.
RETURNED_SQLSTATE	The SQLSTATE value.
MESSAGE_TEXT	Statement error message. See MariaDB Error Codes.
ERRORS	0 if SQLSTATE signifies completion (starting with 00) or warning (01), otherwise 1.
WARNINGS	Number of warnings from the diagnostics area.
ROWS_AFFECTED	Number of rows affected the statement affected.
ROWS_SENT	Number of rows returned.
ROWS_EXAMINED	Number of rows read during the statement's execution.
CREATED_TMP_DISK_TABLES	Number of on-disk temp tables created by the statement.
CREATED_TMP_TABLES	Number of temp tables created by the statement.
SELECT_FULL_JOIN	Number of joins performed by the statement which did not use an index.
SELECT_FULL_RANGE_JOIN	Number of joins performed by the statement which used a range search of the first table.
SELECT_RANGE	Number of joins performed by the statement which used a range of the first table.
SELECT_RANGE_CHECK	Number of joins without keys performed by the statement that check for key usage after each row.
SELECT_SCAN	Number of joins performed by the statement which used a full scan of the first table.
SORT_MERGE_PASSES	Number of merges passes by the sort algorithm performed by the statement. If too high, you may need to increase the sort_buffer_size.
SORT_RANGE	Number of sorts performed by the statement which used a range.
SORT_ROWS	Number of rows sorted by the statement.
SORT_SCAN	Number of sorts performed by the statement which used a full table scan.
NO_INDEX_USED	0 if the statement performed a table scan with an index, 1 if without an index.
NO_GOOD_INDEX_USED	0 if a good index was found for the statement, 1 if no good index was found. See the Range checked for each record description in the EXPLAIN article.
NESTING_EVENT_ID	Reserved, currently NULL.
NESTING_EVENT_TYPE	Reserved, currently NULL.

It is possible to empty this table with a `TRUNCATE TABLE` statement.

The related tables, events_statements_history and events_statements_history_long derive their values from the current events table.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_history Table

The `events_statements_history` table by default contains the ten most recent completed statement events per thread. This number can be adjusted by setting the performance_schema_events_statements_history_size system variable when the server starts up.

The table structure is identical to the events_statements_current table structure, and contains the following columns:

The table contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
LOCK_TIME	Time in picoseconds spent waiting for locks. The time is calculated in microseconds but stored in picoseconds for compatibility with other timings.
SQL_TEXT	The SQL statement, or NULL if the command is not associated with an SQL statement.
DIGEST	Statement digest.
DIGEST_TEXT	Statement digest text.
CURRENT_SCHEMA	Statement's default database for the statement, or NULL if there was none.
OBJECT_SCHEMA	Reserved, currently NULL
OBJECT_NAME	Reserved, currently NULL
OBJECT_TYPE	Reserved, currently NULL
OBJECT_INSTANCE_BEGIN	Address in memory of the statement object.
MYSQL_ERRNO	Error code. See MariaDB Error Codes for a full list.
RETURNED_SQLSTATE	The SQLSTATE value.
MESSAGE_TEXT	Statement error message. See MariaDB Error Codes.
ERRORS	0 if SQLSTATE signifies completion (starting with 00) or warning (01), otherwise 1.
WARNINGS	Number of warnings from the diagnostics area.
ROWS_AFFECTED	Number of rows affected the statement affected.
ROWS_SENT	Number of rows returned.
ROWS_EXAMINED	Number of rows read during the statement's execution.
CREATED_TMP_DISK_TABLES	Number of on-disk temp tables created by the statement.

Column	Description
CREATED_TMP_TABLES	Number of temp tables created by the statement.
SELECT_FULL_JOIN	Number of joins performed by the statement which did not use an index.
SELECT_FULL_RANGE_JOIN	Number of joins performed by the statement which used a range search of the first table.
SELECT_RANGE	Number of joins performed by the statement which used a range of the first table.
SELECT_RANGE_CHECK	Number of joins without keys performed by the statement that check for key usage after each row.
SELECT_SCAN	Number of joins performed by the statement which used a full scan of the first table.
SORT_MERGE_PASSES	Number of merge passes by the sort algorithm performed by the statement. If too high, you may need to increase the sort_buffer_size.
SORT_RANGE	Number of sorts performed by the statement which used a range.
SORT_ROWS	Number of rows sorted by the statement.
SORT_SCAN	Number of sorts performed by the statement which used a full table scan.
NO_INDEX_USED	0 if the statement performed a table scan with an index, 1 if without an index.
NO_GOOD_INDEX_USED	0 if a good index was found for the statement, 1 if no good index was found. See the Range checked for each record description in the EXPLAIN article.
NESTING_EVENT_ID	Reserved, currently NULL.
NESTING_EVENT_TYPE	Reserved, currently NULL.

It is possible to empty this table with a `TRUNCATE TABLE` statement.

events_statements_current and events_statements_history_long are related tables.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_history_long Table

The `events_statements_history_long` table by default contains the ten thousand most recent completed statement events. This number can be adjusted by setting the performance_schema_events_statements_history_long_size system variable when the server starts up.

The table structure is identical to the events_statements_current table structure, and contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected. Relative to the server start, not to the epoch, so cannot directly be converted to a timestamp.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.

Column	Description
LOCK_TIME	Time in picoseconds spent waiting for locks. The time is calculated in microseconds but stored in picoseconds for compatibility with other timings.
SQL_TEXT	The SQL statement, or NULL if the command is not associated with an SQL statement.
DIGEST	Statement digest.
DIGEST_TEXT	Statement digest text.
CURRENT_SCHEMA	Statement's default database for the statement, or NULL if there was none.
OBJECT_SCHEMA	Reserved, currently NULL
OBJECT_NAME	Reserved, currently NULL
OBJECT_TYPE	Reserved, currently NULL
OBJECT_INSTANCE_BEGIN	Address in memory of the statement object.
MYSQL_ERRNO	Error code. See MariaDB Error Codes for a full list.
RETURNED_SQLSTATE	The SQLSTATE value.
MESSAGE_TEXT	Statement error message. See MariaDB Error Codes.
ERRORS	0 if SQLSTATE signifies completion (starting with 00) or warning (01), otherwise 1.
WARNINGS	Number of warnings from the diagnostics area.
ROWS_AFFECTED	Number of rows affected the statement affected.
ROWS_SENT	Number of rows returned.
ROWS_EXAMINED	Number of rows read during the statement's execution.
CREATED_TMP_DISK_TABLES	Number of on-disk temp tables created by the statement.
CREATED_TMP_TABLES	Number of temp tables created by the statement.
SELECT_FULL_JOIN	Number of joins performed by the statement which did not use an index.
SELECT_FULL_RANGE_JOIN	Number of joins performed by the statement which used a range search of the first table.
SELECT_RANGE	Number of joins performed by the statement which used a range of the first table.
SELECT_RANGE_CHECK	Number of joins without keys performed by the statement that check for key usage after each row.
SELECT_SCAN	Number of joins performed by the statement which used a full scan of the first table.
SORT_MERGE_PASSES	Number of merge passes by the sort algorithm performed by the statement. If too high, you may need to increase the sort_buffer_size.
SORT_RANGE	Number of sorts performed by the statement which used a range.
SORT_ROWS	Number of rows sorted by the statement.
SORT_SCAN	Number of sorts performed by the statement which used a full table scan.
NO_INDEX_USED	0 if the statement performed a table scan with an index, 1 if without an index.
NO_GOOD_INDEX_USED	0 if a good index was found for the statement, 1 if no good index was found. See the Range checked for each record description in the EXPLAIN article.
NESTING_EVENT_ID	Reserved, currently NULL.
NESTING_EVENT_TYPE	Reserved, currently NULL.

It is possible to empty this table with a `TRUNCATE TABLE` statement.

events_statements_current and events_statements_history are related tables.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_summary_by_account_by_event_name Table

The Performance Schema events_statements_summary_by_account_by_event_name table contains statement events summarized by account and event name. It contains the following columns:

Column	Description
USER	User. Used together with HOST and EVENT_NAME for grouping events.
HOST	Host. Used together with USER and EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER and HOST for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_statements_summary_by_account_by_event_name\G
...
*************************** 521. row ***************************
                    USER: NULL
                    HOST: NULL
              EVENT_NAME: statement/com/Error
              COUNT_STAR: 0
          SUM_TIMER_WAIT: 0
          MIN_TIMER_WAIT: 0
          AVG_TIMER_WAIT: 0
          MAX_TIMER_WAIT: 0
           SUM_LOCK_TIME: 0
              SUM_ERRORS: 0
            SUM_WARNINGS: 0
       SUM_ROWS_AFFECTED: 0
           SUM_ROWS_SENT: 0
       SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
*************************** 522. row ***************************
                    USER: NULL
                    HOST: NULL
              EVENT_NAME: statement/com/
              COUNT_STAR: 0
          SUM_TIMER_WAIT: 0
          MIN_TIMER_WAIT: 0
          AVG_TIMER_WAIT: 0
          MAX_TIMER_WAIT: 0
           SUM_LOCK_TIME: 0
              SUM_ERRORS: 0
            SUM_WARNINGS: 0
       SUM_ROWS_AFFECTED: 0
           SUM_ROWS_SENT: 0
       SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_summary_by_digest Table

The Performance Schema digest is a hashed, normalized form of a statement with the specific data values removed. It allows statistics to be gathered for similar kinds of statements.

The Performance Schema `events_statements_summary_by_digest` table records statement events summarized by schema and digest. It contains the following columns:

Column	Description
SCHEMA NAME	Database name. Records are summarised together with DIGEST.
DIGEST	Performance Schema digest. Records are summarised together with SCHEMA NAME.
DIGEST TEXT	The unhashed form of the digest.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.
FIRST_SEEN	Time at which the digest was first seen.
LAST_SEEN	Time at which the digest was most recently seen.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

The `events_statements_summary_by_digest` table is limited in size by the performance_schema_digests_size system variable. Once the limit has been reached and the table is full, all entries are aggregated in a row with a `NULL` digest. The `COUNT_STAR` value of this `NULL` row indicates how many digests are recorded in the row and therefore gives an indication of whether `performance_schema_digests_size` should be increased to provide more accurate statistics.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_summary_by_host_by_event_name Table

The Performance Schema events_statements_summary_by_host_by_event_name table contains statement events summarized by host and event name. It contains the following columns:

Column	Description
HOST	Host. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with HOST for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_currentd table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOINW column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_statements_summary_by_host_by_event_name\G
...
*************************** 347. row ***************************
                      HOST: NULL
                EVENT_NAME: statement/com/Error
                COUNT_STAR: 0
            SUM_TIMER_WAIT: 0
            MIN_TIMER_WAIT: 0
            AVG_TIMER_WAIT: 0
            MAX_TIMER_WAIT: 0
             SUM_LOCK_TIME: 0
                SUM_ERRORS: 0
              SUM_WARNINGS: 0
         SUM_ROWS_AFFECTED: 0
             SUM_ROWS_SENT: 0
         SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
*************************** 348. row ***************************
                      HOST: NULL
                EVENT_NAME: statement/com/
                COUNT_STAR: 0
            SUM_TIMER_WAIT: 0
            MIN_TIMER_WAIT: 0
            AVG_TIMER_WAIT: 0
            MAX_TIMER_WAIT: 0
             SUM_LOCK_TIME: 0
                SUM_ERRORS: 0
              SUM_WARNINGS: 0
         SUM_ROWS_AFFECTED: 0
             SUM_ROWS_SENT: 0
         SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_summary_by_program Table

> **Note:**
>
> The `events_statements_summary_by_program` table, along with many other new Performance Schema tables, was added in MariaDB 10.5.2.

Each row in the Performance Schema events_statements_summary_by_program table summarizes events for a particular stored program (stored procedure, stored function, trigger or event).

It contains the following fields.

Column	Type	Null	Description
OBJECT_TYPE	enum('EVENT', 'FUNCTION', 'PROCEDURE', 'TABLE', 'TRIGGER')	YES	Object type for which the summary is generated.
OBJECT_SCHEMA	varchar(64)	NO	The schema of the object for which the summary is generated.
OBJECT_NAME	varchar(64)	NO	The name of the object for which the summary is generated.
COUNT_STAR	bigint(20) unsigned	NO	The number of summarized events (from events_statements_current). This value includes all events, whether timed or nontimed.
SUM_TIMER_WAIT	bigint(20) unsigned	NO	The number of summarized events (from events_statements_current). This value includes all events, whether timed or nontimed.
MIN_TIMER_WAIT	bigint(20) unsigned	NO	The minimum wait time of the summarized timed events.
AVG_TIMER_WAIT	bigint(20) unsigned	NO	The average wait time of the summarized timed events.
MAX_TIMER_WAIT	bigint(20) unsigned	NO	The maximum wait time of the summarized timed events.
COUNT_STATEMENTS	bigint(20) unsigned	NO	Total number of nested statements invoked during stored program execution.
SUM_STATEMENTS_WAIT	bigint(20) unsigned	NO	The total wait time of the summarized timed statements. This value is calculated only for timed statements because nontimed statements have a wait time of NULL. The same is true for the other xxx_STATEMENT_WAIT values.
MIN_STATEMENTS_WAIT	bigint(20) unsigned	NO	The minimum wait time of the summarized timed statements.
AVG_STATEMENTS_WAIT	bigint(20) unsigned	NO	The average wait time of the summarized timed statements.
MAX_STATEMENTS_WAIT	bigint(20) unsigned	NO	The maximum wait time of the summarized timed statements.
SUM_LOCK_TIME	bigint(20) unsigned	NO	The total time spent (in picoseconds) waiting for table locks for the summarized statements.
SUM_ERRORS	bigint(20) unsigned	NO	The total number of errors that occurend for the summarized statements.
SUM_WARNINGS	bigint(20) unsigned	NO	The total number of warnings that occurend for the summarized statements.
SUM_ROWS_AFFECTED	bigint(20) unsigned	NO	The total number of affected rows by the summarized statements.

Column	Type	Null	Description
SUM_ROWS_SENT	bigint(20) unsigned	NO	The total number of rows returned by the summarized statements.
SUM_ROWS_EXAMINED	bigint(20) unsigned	NO	The total number of rows examined by the summarized statements.The total number of affected rows by the summarized statements.
SUM_CREATED_TMP_DISK_TABLES	bigint(20) unsigned	NO	The total number of on-disk temporary tables created by the summarized statements.
SUM_CREATED_TMP_TABLES	bigint(20) unsigned	NO	The total number of in-memory temporary tables created by the summarized statements.
SUM_SELECT_FULL_JOIN	bigint(20) unsigned	NO	The total number of full joins executed by the summarized statements.
SUM_SELECT_FULL_RANGE_JOIN	bigint(20) unsigned	NO	The total number of range search joins executed by the summarized statements.
SUM_SELECT_RANGE	bigint(20) unsigned	NO	The total number of joins that used ranges on the first table executed by the summarized statements.
SUM_SELECT_RANGE_CHECK	bigint(20) unsigned	NO	The total number of joins that check for key usage after each row executed by the summarized statements.
SUM_SELECT_SCAN	bigint(20) unsigned	NO	The total number of joins that did a full scan of the first table executed by the summarized statements.
SUM_SORT_MERGE_PASSES	bigint(20) unsigned	NO	The total number of merge passes that the sort algorithm has had to do for the summarized statements.
SUM_SORT_RANGE	bigint(20) unsigned	NO	The total number of sorts that were done using ranges for the summarized statements.
SUM_SORT_ROWS	bigint(20) unsigned	NO	The total number of sorted rows that were sorted by the summarized statements.
SUM_SORT_SCAN	bigint(20) unsigned	NO	The total number of sorts that were done by scanning the table by the summarized statements.
SUM_NO_INDEX_USED	bigint(20) unsigned	NO	The total number of statements that performed a table scan without using an index.
SUM_NO_GOOD_INDEX_USED	bigint(20) unsigned	NO	The total number of statements where no good index was found.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_summary_by_thread_by_event_name Table

The Performance Schema `events_statements_summary_by_thread_by_event_name` table contains statement events summarized by thread and event name. It contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_NAME uniquely identifies the row.
EVENT_NAME	Event name. Used together with THREAD_ID for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.

Column	Description
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_statements_summary_by_thread_by_event_name\G
...
*************************** 3653. row ***************************
                  THREAD_ID: 64
                 EVENT_NAME: statement/com/Error
                 COUNT_STAR: 0
             SUM_TIMER_WAIT: 0
             MIN_TIMER_WAIT: 0
             AVG_TIMER_WAIT: 0
             MAX_TIMER_WAIT: 0
              SUM_LOCK_TIME: 0
                 SUM_ERRORS: 0
               SUM_WARNINGS: 0
          SUM_ROWS_AFFECTED: 0
              SUM_ROWS_SENT: 0
          SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
*************************** 3654. row ***************************
                  THREAD_ID: 64
                 EVENT_NAME: statement/com/
                 COUNT_STAR: 0
             SUM_TIMER_WAIT: 0
             MIN_TIMER_WAIT: 0
             AVG_TIMER_WAIT: 0
             MAX_TIMER_WAIT: 0
              SUM_LOCK_TIME: 0
                 SUM_ERRORS: 0
               SUM_WARNINGS: 0
          SUM_ROWS_AFFECTED: 0
              SUM_ROWS_SENT: 0
          SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_summary_by_user_by_event_name Table

The Performance Schema `events_statements_summary_by_user_by_event_name` table contains statement events summarized by user and event name. It contains the following columns:

Column	Description
USER	User. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_statements_summary_by_user_by_event_name\G
...
*************************** 521. row ***************************
                    USER: NULL
              EVENT_NAME: statement/com/Error
              COUNT_STAR: 0
          SUM_TIMER_WAIT: 0
          MIN_TIMER_WAIT: 0
          AVG_TIMER_WAIT: 0
          MAX_TIMER_WAIT: 0
           SUM_LOCK_TIME: 0
              SUM_ERRORS: 0
            SUM_WARNINGS: 0
       SUM_ROWS_AFFECTED: 0
           SUM_ROWS_SENT: 0
       SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
*************************** 522. row ***************************
                    USER: NULL
              EVENT_NAME: statement/com/
              COUNT_STAR: 0
          SUM_TIMER_WAIT: 0
          MIN_TIMER_WAIT: 0
          AVG_TIMER_WAIT: 0
          MAX_TIMER_WAIT: 0
           SUM_LOCK_TIME: 0
              SUM_ERRORS: 0
            SUM_WARNINGS: 0
       SUM_ROWS_AFFECTED: 0
           SUM_ROWS_SENT: 0
       SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_statements_summary_global_by_event_name Table

The Performance Schema `events_statements_summary_global_by_event_name` table contains statement events summarized by event name. It contains the following columns:

Column	Description
EVENT_NAME	Event name.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_statements_summary_global_by_event_name\G
...
*************************** 173. row ***************************
                EVENT_NAME: statement/com/Error
                COUNT_STAR: 0
            SUM_TIMER_WAIT: 0
            MIN_TIMER_WAIT: 0
            AVG_TIMER_WAIT: 0
            MAX_TIMER_WAIT: 0
             SUM_LOCK_TIME: 0
                SUM_ERRORS: 0
              SUM_WARNINGS: 0
         SUM_ROWS_AFFECTED: 0
             SUM_ROWS_SENT: 0
         SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
*************************** 174. row ***************************
                EVENT_NAME: statement/com/
                COUNT_STAR: 0
            SUM_TIMER_WAIT: 0
            MIN_TIMER_WAIT: 0
            AVG_TIMER_WAIT: 0
            MAX_TIMER_WAIT: 0
             SUM_LOCK_TIME: 0
                SUM_ERRORS: 0
              SUM_WARNINGS: 0
         SUM_ROWS_AFFECTED: 0
             SUM_ROWS_SENT: 0
         SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
     SUM_CREATED_TMP_TABLES: 0
       SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 0
      SUM_SORT_MERGE_PASSES: 0
             SUM_SORT_RANGE: 0
              SUM_SORT_ROWS: 0
              SUM_SORT_SCAN: 0
          SUM_NO_INDEX_USED: 0
     SUM_NO_GOOD_INDEX_USED: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_transactions_current Table

> **Note:**
>
> The `events_transactions_current` table is available from MariaDB 10.5.2.

The `events_transactions_current` table contains current transaction events for each thread.

The table size cannot be figured, and always stores one row for each thread, showing the current status of the thread's most recent monitored transaction event.

The table contains the following columns:

Column	Type	Description
THREAD_ID	bigint(20) unsigned	The thread associated with the event.
EVENT_ID	bigint(20) unsigned	The event id associated with the event.
END_EVENT_ID	bigint(20) unsigned	This column is set to NULL when the event starts and updated to the thread current event number when the event ends.
EVENT_NAME	varchar(128)	The name of the instrument from which the event was collected. This is a NAME value from the setup_instruments table.
STATE	enum('ACTIVE', 'COMMITTED', 'ROLLED BACK')	The current transaction state. The value is ACTIVE (after START TRANSACTION or BEGIN), COMMITTED (after COMMIT), or ROLLED BACK (after ROLLBACK).
TRX_ID	bigint(20) unsigned	Unused.
GTID	varchar(64)	Transaction GTID, using the format DOMAIN-SERVER_ID-SEQUENCE_NO.
XID_FORMAT_ID	int(11)	XA transaction format ID for GTRID and BQUAL values.
XID_GTRID	varchar(130)	XA global transaction ID.
XID_BQUAL	varchar(130)	XA transaction branch qualifier.
XA_STATE	varchar(64)	The state of the XA transaction. The value is ACTIVE (after XA START), IDLE (after XA END), PREPARED (after XA PREPARE), ROLLED BACK (after XA ROLLBACK), or COMMITTED (after XA COMMIT).
SOURCE	varchar(64)	The name of the source file containing the instrumented code that produced the event and the line number in the file at which the instrumentation occurs.
TIMER_START	bigint(20) unsigned	The unit is picoseconds. When event timing started. NULL if event has no timing information.
TIMER_END	bigint(20) unsigned	The unit is picoseconds. When event timing ended. NULL if event has no timing information.
TIMER_WAIT	bigint(20) unsigned	The unit is picoseconds. Event duration. NULL if event has not timing information.
ACCESS_MODE	enum('READ ONLY', 'READ WRITE')	Transaction access mode.
ISOLATION_LEVEL	varchar(64)	Transaction isolation level. One of: REPEATABLE READ, READ COMMITTED, READ UNCOMMITTED, or SERIALIZABLE.
AUTOCOMMIT	enum('YES','NO')	Whether autcommit mode was enabled when the transaction started.
NUMBER_OF_SAVEPOINTS	bigint(20) unsigned	The number of SAVEPOINT statements issued during the transaction.
NUMBER_OF_ROLLBACK_TO_SAVEPOINT	bigint(20) unsigned	The number of ROLLBACK_TO_SAVEPOINT statements issued during the transaction.
NUMBER_OF_RELEASE_SAVEPOINT	bigint(20) unsigned	The number of RELEASE_SAVEPOINT statements issued during the transaction.
OBJECT_INSTANCE_BEGIN	bigint(20) unsigned	Unused.
NESTING_EVENT_ID	bigint(20) unsigned	The EVENT_ID value of the event within which this event is nested.
NESTING_EVENT_TYPE	enum('TRANSACTION', 'STATEMENT', 'STAGE', 'WAIT')	The nesting event type.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_transactions_history Table

> **Note:**
>
> The `events_transactions_history` table is available from MariaDB 10.5.2.

The `events_transactions_history` table contains the most recent completed transaction events for each thread.

The number of records stored per thread in the table is determined by the performance_schema_events_transactions_history_size system variable, which is autosized on startup.

If adding a completed transaction event would cause the table to exceed this limit, the oldest thread row is discarded.

All of a thread's rows are discarded when the thread ends.

The table contains the following columns:

Column	Type	Description
THREAD_ID	bigint(20) unsigned	The thread associated with the event.
EVENT_ID	bigint(20) unsigned	The event id associated with the event.
END_EVENT_ID	bigint(20) unsigned	This column is set to NULL when the event starts and updated to the thread current event number when the event ends.
EVENT_NAME	varchar(128)	The name of the instrument from which the event was collected. This is a NAME value from the setup_instruments table.
STATE	enum('ACTIVE', 'COMMITTED',' ROLLED BACK')	The current transaction state. The value is ACTIVE (after START TRANSACTION or BEGIN), COMMITTED (after COMMIT), or ROLLED BACK (after ROLLBACK).
TRX_ID	bigint(20) unsigned	Unused.
GTID	varchar(64)	Transaction GTID, using the format DOMAIN-SERVER_ID-SEQUENCE_NO.
XID_FORMAT_ID	int(11)	XA transaction format ID for GTRID and BQUAL values.
XID_GTRID	varchar(130)	XA global transaction ID.
XID_BQUAL	varchar(130)	XA transaction branch qualifier.
XA_STATE	varchar(64)	The state of the XA transaction. The value is ACTIVE (after XA START), IDLE (after XA END), PREPARED (after XA PREPARE), ROLLED BACK (after XA ROLLBACK), or COMMITTED (after XA COMMIT).
SOURCE	varchar(64)	The name of the source file containing the instrumented code that produced the event and the line number in the file at which the instrumentation occurs.
TIMER_START	bigint(20) unsigned	The unit is picoseconds. When event timing started. NULL if event has no timing information.
TIMER_END	bigint(20) unsigned	The unit is picoseconds. When event timing ended. NULL if event has no timing information.
TIMER_WAIT	bigint(20) unsigned	The unit is picoseconds. Event duration. NULL if event has not timing information.
ACCESS_MODE	enum('READ ONLY', 'READ WRITE')	Transaction access mode.
ISOLATION_LEVEL	varchar(64)	Transaction isolation level. One of: REPEATABLE READ, READ COMMITTED, READ UNCOMMITTED, or SERIALIZABLE.
AUTOCOMMIT	enum('YES', 'NO')	NO

Column	Type	Description
NUMBER_OF_SAVEPOINTS	bigint(20) unsigned	The number of SAVEPOINT statements issued during the transaction.
NUMBER_OF_ROLLBACK_TO_SAVEPOINT	bigint(20) unsigned	The number of ROLLBACK_TO_SAVEPOINT statements issued during the transaction.
NUMBER_OF_RELEASE_SAVEPOINT	bigint(20) unsigned	The number of RELEASE_SAVEPOINT statements issued during the transaction.
OBJECT_INSTANCE_BEGIN	bigint(20) unsigned	Unused.
NESTING_EVENT_ID	bigint(20) unsigned	The EVENT_ID value of the event within which this event is nested.
NESTING_EVENT_TYPE	enum('TRANSACTION','STATEMENT', 'STAGE', 'WAIT')	The nesting event type.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_transactions_history_long Table

> **Note:**
>
> The `events_transactions_history_long` table is available from MariaDB 10.5.2.

The `events_transactions_history_long` table contains the most recent completed transaction events that have ended globally, across all threads.

The number of records stored in the table is determined by the performance_schema_events_transactions_history_long_size system variable, which is autosized on startup.

If adding a completed transaction would cause the table to exceed this limit, the oldest row, regardless of thread, is discarded.

The table contains the following columns:

Column	Type	Description
THREAD_ID	bigint(20) unsigned	The thread associated with the event.
EVENT_ID	bigint(20) unsigned	The event id associated with the event.
END_EVENT_ID	bigint(20) unsigned	This column is set to NULL when the event starts and updated to the thread current event number when the event ends.
EVENT_NAME	varchar(128)	The name of the instrument from which the event was collected. This is a NAME value from the setup_instruments table.
STATE	enum('ACTIVE', 'COMMITTED',' ROLLED BACK')	The current transaction state. The value is ACTIVE (after START TRANSACTION or BEGIN), COMMITTED (after COMMIT), or ROLLED BACK (after ROLLBACK).
TRX_ID	bigint(20) unsigned	Unused.
GTID	varchar(64)	Transaction GTID, using the format DOMAIN-SERVER_ID-SEQUENCE_NO.
XID_FORMAT_ID	int(11)	XA transaction format ID for GTRID and BQUAL values.
XID_GTRID	varchar(130)	XA global transaction ID.
XID_BQUAL	varchar(130)	XA transaction branch qualifier.

Column	Type	Description
XA_STATE	varchar(64)	The state of the XA transaction. The value is ACTIVE (after XA START), IDLE (after XA END), PREPARED (after XA PREPARE), ROLLED BACK (after XA ROLLBACK), or COMMITTED (after XA COMMIT).
SOURCE	varchar(64)	The name of the source file containing the instrumented code that produced the event and the line number in the file at which the instrumentation occurs.
TIMER_START	bigint(20) unsigned	The unit is picoseconds. When event timing started. NULL if event has no timing information.
TIMER_END	bigint(20) unsigned	The unit is picoseconds. When event timing ended. NULL if event has no timing information.
TIMER_WAIT	bigint(20) unsigned	The unit is picoseconds. Event duration. NULL if event has not timing information.
ACCESS_MODE	enum('READ ONLY', 'READ WRITE')	Transaction access mode.
ISOLATION_LEVEL	varchar(64)	Transaction isolation level. One of: REPEATABLE READ, READ COMMITTED, READ UNCOMMITTED, or SERIALIZABLE.
AUTOCOMMIT	enum('YES', 'NO')	NO
NUMBER_OF_SAVEPOINTS	bigint(20) unsigned	The number of SAVEPOINT statements issued during the transaction.
NUMBER_OF_ROLLBACK_TO_SAVEPOINT	bigint(20) unsigned	The number of ROLLBACK_TO_SAVEPOINT statements issued during the transaction.
NUMBER_OF_RELEASE_SAVEPOINT	bigint(20) unsigned	The number of RELEASE_SAVEPOINT statements issued during the transaction.
OBJECT_INSTANCE_BEGIN	bigint(20) unsigned	Unused.
NESTING_EVENT_ID	bigint(20) unsigned	The EVENT_ID value of the event within which this event is nested.
NESTING_EVENT_TYPE	enum('TRANSACTION','STATEMENT', 'STAGE', 'WAIT')	The nesting event type.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_transactions_summary_by_account_by_event_name Table

> **Note:**
>
> The `events_transactions_summary_by_account_by_event_name` table is available from MariaDB 10.5.2.

The `events_transactions_summary_by_account_by_event_name` table contains information on transaction events aggregated by account and event name.

The table contains the following columns:

Column	Type	Description
USER	char(32)	User for which summary is generated.
HOST	char(60)	Host for which summary is generated.
EVENT_NAME	varchar(128)	Event name for which summary is generated.
COUNT_STAR	bigint(20) unsigned	The number of summarized events. This value includes all events, whether timed or nontimed.

Column	Type	Description
SUM_TIMER_WAIT	bigint(20) unsigned	The total wait time of the summarized timed events. This value is calculated only for timed events because nontimed events have a wait time of NULL. The same is true for the other xxx_TIMER_WAIT values.
MIN_TIMER_WAIT	bigint(20) unsigned	The minimum wait time of the summarized timed events.
AVG_TIMER_WAIT	bigint(20) unsigned	The average wait time of the summarized timed events.
MAX_TIMER_WAIT	bigint(20) unsigned	The maximum wait time of the summarized timed events.
COUNT_READ_WRITE	bigint(20) unsigned	The total number of only READ/WRITE transaction events.
SUM_TIMER_READ_WRITE	bigint(20) unsigned	The total wait time of only READ/WRITE transaction events.
MIN_TIMER_READ_WRITE	bigint(20) unsigned	The minimum wait time of only READ/WRITE transaction events.
AVG_TIMER_READ_WRITE	bigint(20) unsigned	The average wait time of only READ/WRITE transaction events.
MAX_TIMER_READ_WRITE	bigint(20) unsigned	The maximum wait time of only READ/WRITE transaction events.
COUNT_READ_ONLY	bigint(20) unsigned	The total number of only READ ONLY transaction events.
SUM_TIMER_READ_ONLY	bigint(20) unsigned	The total wait time of only READ ONLY transaction events.
MIN_TIMER_READ_ONLY	bigint(20) unsigned	The minimum wait time of only READ ONLY transaction events.
AVG_TIMER_READ_ONLY	bigint(20) unsigned	The average wait time of only READ ONLY transaction events.
MAX_TIMER_READ_ONLY	bigint(20) unsigned	The maximum wait time of only READ ONLY transaction events.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_transactions_summary_by_host_by_event_name Table

> **Note:**
>
> The `events_transactions_summary_by_host_by_event_name` table is available from MariaDB 10.5.2.

The `events_transactions_summary_by_host_by_event_name` table contains information on transaction events aggregated by host and event name.

The table contains the following columns:

Column	Type	Description
HOST	char(60)	Host for which summary is generated.
EVENT_NAME	varchar(128)	Event name for which summary is generated.
COUNT_STAR	bigint(20) unsigned	The number of summarized events. This value includes all events, whether timed or nontimed.
SUM_TIMER_WAIT	bigint(20) unsigned	The total wait time of the summarized timed events. This value is calculated only for timed events because nontimed events have a wait time of NULL. The same is true for the other xxx_TIMER_WAIT values.

Column	Type	Description
MIN_TIMER_WAIT	bigint(20) unsigned	The minimum wait time of the summarized timed events.
AVG_TIMER_WAIT	bigint(20) unsigned	The average wait time of the summarized timed events.
MAX_TIMER_WAIT	bigint(20) unsigned	The maximum wait time of the summarized timed events.
COUNT_READ_WRITE	bigint(20) unsigned	The total number of only READ/WRITE transaction events.
SUM_TIMER_READ_WRITE	bigint(20) unsigned	The total wait time of only READ/WRITE transaction events.
MIN_TIMER_READ_WRITE	bigint(20) unsigned	The minimum wait time of only READ/WRITE transaction events.
AVG_TIMER_READ_WRITE	bigint(20) unsigned	The average wait time of only READ/WRITE transaction events.
MAX_TIMER_READ_WRITE	bigint(20) unsigned	The maximum wait time of only READ/WRITE transaction events.
COUNT_READ_ONLY	bigint(20) unsigned	The total number of only READ ONLY transaction events.
SUM_TIMER_READ_ONLY	bigint(20) unsigned	The total wait time of only READ ONLY transaction events.
MIN_TIMER_READ_ONLY	bigint(20) unsigned	The minimum wait time of only READ ONLY transaction events.
AVG_TIMER_READ_ONLY	bigint(20) unsigned	The average wait time of only READ ONLY transaction events.
MAX_TIMER_READ_ONLY	bigint(20) unsigned	The maximum wait time of only READ ONLY transaction events.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_transactions_summary_by_thread_by_event_name Table

> **Note:**
>
> The events_transactions_summary_by_thread_by_event_name table is available from MariaDB 10.5.2.

The `events_transactions_summary_by_thread_by_event_name` table contains information on transaction events aggregated by thread and event name.

The table contains the following columns:

```
+----------------------+---------------------+------+-----+---------+-------+
| Field                | Type                | Null | Key | Default | Extra |
+----------------------+---------------------+------+-----+---------+-------+
| THREAD_ID            | bigint(20) unsigned | NO   |     | NULL    |       |
| EVENT_NAME           | varchar(128)        | NO   |     | NULL    |       |
| COUNT_STAR           | bigint(20) unsigned | NO   |     | NULL    |       |
| SUM_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| MIN_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| AVG_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| MAX_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| COUNT_READ_WRITE     | bigint(20) unsigned | NO   |     | NULL    |       |
| SUM_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| MIN_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| AVG_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| MAX_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| COUNT_READ_ONLY      | bigint(20) unsigned | NO   |     | NULL    |       |
| SUM_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
| MIN_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
| AVG_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
| MAX_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
+----------------------+---------------------+------+-----+---------+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_transactions_summary_by_user_by_event_name Table

Note:

The `events_transactions_summary_by_user_by_event_name` table is available from MariaDB 10.5.2.

The `events_transactions_summary_by_user_by_event_name` table contains information on transaction events aggregated by user and event name.

The table contains the following columns:

```
+----------------------+---------------------+------+-----+---------+-------+
| Field                | Type                | Null | Key | Default | Extra |
+----------------------+---------------------+------+-----+---------+-------+
| USER                 | char(32)            | YES  |     | NULL    |       |
| EVENT_NAME           | varchar(128)        | NO   |     | NULL    |       |
| COUNT_STAR           | bigint(20) unsigned | NO   |     | NULL    |       |
| SUM_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| MIN_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| AVG_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| MAX_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| COUNT_READ_WRITE     | bigint(20) unsigned | NO   |     | NULL    |       |
| SUM_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| MIN_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| AVG_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| MAX_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| COUNT_READ_ONLY      | bigint(20) unsigned | NO   |     | NULL    |       |
| SUM_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
| MIN_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
| AVG_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
| MAX_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
+----------------------+---------------------+------+-----+---------+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_transactions_summary_global_by_event_name Table

> **Note:**
>
> The `events_transactions_summary_global_by_event_name` table is available from MariaDB 10.5.2.

The `events_transactions_summary_global_by_event_name` table contains information on transaction events aggregated by event name.

The table contains the following columns:

```
+----------------------+---------------------+------+-----+---------+-------+
| Field                | Type                | Null | Key | Default | Extra |
+----------------------+---------------------+------+-----+---------+-------+
| EVENT_NAME           | varchar(128)        | NO   |     | NULL    |       |
| COUNT_STAR           | bigint(20) unsigned | NO   |     | NULL    |       |
| SUM_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| MIN_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| AVG_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| MAX_TIMER_WAIT       | bigint(20) unsigned | NO   |     | NULL    |       |
| COUNT_READ_WRITE     | bigint(20) unsigned | NO   |     | NULL    |       |
| SUM_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| MIN_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| AVG_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| MAX_TIMER_READ_WRITE | bigint(20) unsigned | NO   |     | NULL    |       |
| COUNT_READ_ONLY      | bigint(20) unsigned | NO   |     | NULL    |       |
| SUM_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
| MIN_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
| AVG_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
| MAX_TIMER_READ_ONLY  | bigint(20) unsigned | NO   |     | NULL    |       |
+----------------------+---------------------+------+-----+---------+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_waits_current Table

The `events_waits_current` table contains the status of a thread's most recently monitored wait event, listing one event per thread.

The table contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if the event has not ended or timing is not collected.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if the event has not ended or timing is not collected.
SPINS	Number of spin rounds for a mutex, or NULL if spin rounds are not used, or spinning is not instrumented.
OBJECT_SCHEMA	Name of the schema that contains the table for table I/O objects, otherwise NULL for file I/O and synchronization objects.
OBJECT_NAME	File name for file I/O objects, table name for table I/O objects, the socket's IP:PORT value for a socket object or NULL for a synchronization object.
INDEX NAME	Name of the index, PRIMARY for the primary key, or NULL for no index used.

Column	Description
OBJECT_TYPE	FILE for a file object, TABLE or TEMPORARY TABLE for a table object, or NULL for a synchronization object.
OBJECT_INSTANCE_BEGIN	Address in memory of the object.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. Either statement, stage or wait.
OPERATION	Operation type, for example read, write or lock
NUMBER_OF_BYTES	Number of bytes that the operation read or wrote, or NULL for table I/O waits.
FLAGS	Reserved for use in the future.

It is possible to empty this table with a `TRUNCATE TABLE` statement.

The related tables, events_waits_history and events_waits_history_long derive their values from the current events.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_waits_history Table

The `events_waits_history` table by default contains the ten most recent completed wait events per thread. This number can be adjusted by setting the performance_schema_events_waits_history_size system variable when the server starts up.

The table structure is identical to the events_waits_current table structure, and contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
SPINS	Number of spin rounds for a mutex, or NULL if spin rounds are not used, or spinning is not instrumented.
OBJECT_SCHEMA	Name of the schema that contains the table for table I/O objects, otherwise NULL for file I/O and synchronization objects.
OBJECT_NAME	File name for file I/O objects, table name for table I/O objects, the socket's IP:PORT value for a socket object or NULL for a synchronization object.
INDEX NAME	Name of the index, PRIMARY for the primary key, or NULL for no index used.
OBJECT_TYPE	FILE for a file object, TABLE or TEMPORARY TABLE for a table object, or NULL for a synchronization object.
OBJECT_INSTANCE_BEGIN	Address in memory of the object.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. Either statement, stage or wait.
OPERATION	Operation type, for example read, write or lock
NUMBER_OF_BYTES	Number of bytes that the operation read or wrote, or NULL for table I/O waits.

Column	Description
FLAGS	Reserved for use in the future.

It is possible to empty this table with a `TRUNCATE TABLE` statement.

events_waits_current and events_waits_history_long are related tables.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_waits_history_long Table

The `events_waits_history_long` table by default contains the ten thousand most recent completed wait events. This number can be adjusted by setting the performance_schema_events_waits_history_long_size system variable when the server starts up.

The table structure is identical to the events_waits_current table structure, and contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together withTHREAD_IDuniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
SPINS	Number of spin rounds for a mutex, or NULL if spin rounds are not used, or spinning is not instrumented.
OBJECT_SCHEMA	Name of the schema that contains the table for table I/O objects, otherwise NULL for file I/O and synchronization objects.
OBJECT_NAME	File name for file I/O objects, table name for table I/O objects, the socket's IP:PORT value for a socket object or NULL for a synchronization object.
INDEX_NAME	Name of the index, PRIMARY for the primary key, or NULL for no index used.
OBJECT_TYPE	FILE for a file object, TABLE or TEMPORARY TABLE for a table object, or NULL for a synchronization object.
OBJECT_INSTANCE_BEGIN	Address in memory of the object.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. Either statement, stage or wait.
OPERATION	Operation type, for example read, write or lock
NUMBER_OF_BYTES	Number of bytes that the operation read or wrote, or NULL for table I/O waits.
FLAGS	Reserved for use in the future.

It is possible to empty this table with a `TRUNCATE TABLE` statement.

events_waits_current and events_waits_history are related tables.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_waits_summary_by_account_by_event_name Table

The Performance Schema `events_waits_summary_by_account_by_event_name` table contains wait events summarized by account and event name. It contains the following columns:

Column	Description
USER	User. Used together with HOST and EVENT_NAME for grouping events.
HOST	Host. Used together with USER and EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER and HOST for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_waits_summary_by_account_by_event_name\G
...
*************************** 915. row ***************************
          USER: NULL
          HOST: NULL
    EVENT_NAME: wait/io/socket/sql/server_tcpip_socket
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 916. row ***************************
          USER: NULL
          HOST: NULL
    EVENT_NAME: wait/io/socket/sql/server_unix_socket
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 917. row ***************************
          USER: NULL
          HOST: NULL
    EVENT_NAME: wait/io/socket/sql/client_connection
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 918. row ***************************
          USER: NULL
          HOST: NULL
    EVENT_NAME: idle
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_waits_summary_by_host_by_event_name Table

The Performance Schema `events_waits_summary_by_host_by_event_name` table contains wait events summarized by host and event name. It contains the following columns:

Column	Description
HOST	Host. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER and HOST for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_waits_summary_by_host_by_event_name\G
...
*************************** 610. row ***************************
          HOST: NULL
    EVENT_NAME: wait/io/socket/sql/server_unix_socket
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 611. row ***************************
          HOST: NULL
    EVENT_NAME: wait/io/socket/sql/client_connection
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 612. row ***************************
          HOST: NULL
    EVENT_NAME: idle
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_waits_summary_by_instance Table

The Performance Schema `events_waits_summary_by_instance` table contains wait events summarized by instance. It contains the following columns:

Column	Description
EVENT_NAME	Event name. Used together with OBJECT_INSTANCE_BEGIN for grouping events.
OBJECT_INSTANCE_BEGIN	If an instrument creates multiple instances, each instance has a unique OBJECT_INSTANCE_BEGIN value to allow for grouping by instance.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.

Column	Description
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_waits_summary_by_instance\G
...
*************************** 202. row ***************************
           EVENT_NAME: wait/io/file/sql/binlog
OBJECT_INSTANCE_BEGIN: 140578961969856
           COUNT_STAR: 6
       SUM_TIMER_WAIT: 90478331960
       MIN_TIMER_WAIT: 263344
       AVG_TIMER_WAIT: 15079721848
       MAX_TIMER_WAIT: 67760576376
*************************** 203. row ***************************
           EVENT_NAME: wait/io/file/sql/dbopt
OBJECT_INSTANCE_BEGIN: 140578961970560
           COUNT_STAR: 6
       SUM_TIMER_WAIT: 39891428472
       MIN_TIMER_WAIT: 387168
       AVG_TIMER_WAIT: 6648571412
       MAX_TIMER_WAIT: 24503293304
*************************** 204. row ***************************
           EVENT_NAME: wait/io/file/sql/dbopt
OBJECT_INSTANCE_BEGIN: 140578961971264
           COUNT_STAR: 6
       SUM_TIMER_WAIT: 39902495024
       MIN_TIMER_WAIT: 177888
       AVG_TIMER_WAIT: 6650415692
       MAX_TIMER_WAIT: 21026400404
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_waits_summary_by_thread_by_event_name Table

The Performance Schema `events_waits_summary_by_thread_by_event_name` table contains wait events summarized by thread and event name. It contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_NAME uniquely identifies the row.
EVENT_NAME	Event name. Used together with THREAD_ID for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_waits_summary_by_thread_by_event_name\G
...
*************************** 6424. row ***************************
     THREAD_ID: 64
    EVENT_NAME: wait/io/socket/sql/server_unix_socket
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 6425. row ***************************
     THREAD_ID: 64
    EVENT_NAME: wait/io/socket/sql/client_connection
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 6426. row ***************************
     THREAD_ID: 64
    EVENT_NAME: idle
    COUNT_STAR: 73
SUM_TIMER_WAIT: 22005252162000000
MIN_TIMER_WAIT: 3000000
AVG_TIMER_WAIT: 301441810000000
MAX_TIMER_WAIT: 4912417573000000
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_waits_summary_by_user_by_event_name Table

The Performance Schema `events_waits_summary_by_user_by_event_name` table contains wait events summarized by user and event name. It contains the following columns:

Column	Description
USER	User. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_waits_summary_by_user_by_event_name\G
...
*************************** 916. row ***************************
          USER: NULL
    EVENT_NAME: wait/io/socket/sql/server_unix_socket
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 917. row ***************************
          USER: NULL
    EVENT_NAME: wait/io/socket/sql/client_connection
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 918. row ***************************
          USER: NULL
    EVENT_NAME: idle
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema events_waits_summary_global_by_event_name Table

The Performance Schema `events_waits_summary_global_by_event_name` table contains wait events summarized by event name. It contains the following columns:

Column	Description
EVENT_NAME	Event name.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

The `*_TIMER_WAIT` columns only calculate results for timed events, as non-timed events have a `NULL` wait time.

Example

```
SELECT * FROM events_waits_summary_global_by_event_name\G
...
*************************** 303. row ***************************
    EVENT_NAME: wait/io/socket/sql/server_tcpip_socket
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 304. row ***************************
    EVENT_NAME: wait/io/socket/sql/server_unix_socket
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 305. row ***************************
    EVENT_NAME: wait/io/socket/sql/client_connection
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 306. row ***************************
    EVENT_NAME: idle
    COUNT_STAR: 265
SUM_TIMER_WAIT: 46861125181000000
MIN_TIMER_WAIT: 1000000
AVG_TIMER_WAIT: 176834434000000
MAX_TIMER_WAIT: 4912417573000000
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema file_instances Table

Description

The `file_instances` table lists instances of instruments seen by the Performance Schema when executing file I/O instrumentation, and the associated files. Only files that have been opened, and that have not been deleted, will be listed in the table.

The performance_schema_max_file_instances system variable specifies the maximum number of instrumented file objects.

Column	Description
FILE_NAME	File name.
EVENT_NAME	Instrument name associated with the file.
OPEN_COUNT	Open handles on the file. A value of greater than zero means that the file is currently open.

Example

```
SELECT * FROM performance_schema.file_instances WHERE OPEN_COUNT>0;
+---------------------------------------------------+-------------------------------------+------------
+
| FILE_NAME                                         | EVENT_NAME                          | OPEN_COUNT
|
+---------------------------------------------------+-------------------------------------+------------
+
| /var/log/mysql/mariadb-bin.index                  | wait/io/file/sql/binlog_index       |          1
|
| /var/lib/mysql/ibdata1                            | wait/io/file/innodb/innodb_data_file |          2
|
| /var/lib/mysql/ib_logfile0                        | wait/io/file/innodb/innodb_log_file  |          2
|
| /var/lib/mysql/ib_logfile1                        | wait/io/file/innodb/innodb_log_file  |          2
|
| /var/lib/mysql/mysql/gtid_slave_pos.ibd           | wait/io/file/innodb/innodb_data_file |          3
|
| /var/lib/mysql/mysql/innodb_index_stats.ibd       | wait/io/file/innodb/innodb_data_file |          3
|
| /var/lib/mysql/mysql/innodb_table_stats.ibd       | wait/io/file/innodb/innodb_data_file |          3
|
...
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema file_summary_by_event_name Table

The Performance Schema `file_summary_by_event_name` table contains file events summarized by event name. It contains the following columns:

Column	Description
EVENT_NAME	Event name.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, including FGETS, FGETC, FREAD, and READ.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
SUM_NUMBER_OF_BYTES_READ	Bytes read by read operations.
COUNT_WRITE	Number of all write operations, including FPUTS, FPUTC, FPRINTF, VFPRINTF, FWRITE, and PWRITE.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.

Column	Description
SUM_NUMBER_OF_BYTES_WRITE	Bytes written by write operations.
COUNT_MISC	Number of all miscellaneous operations not counted above, including CREATE, DELETE, OPEN, CLOSE, STREAM_OPEN, STREAM_CLOSE, SEEK, TELL, FLUSH, STAT, FSTAT, CHSIZE, RENAME, and SYNC.
SUM_TIMER_MISC	Total wait time of all miscellaneous operations that are timed.
MIN_TIMER_MISC	Minimum wait time of all miscellaneous operations that are timed.
AVG_TIMER_MISC	Average wait time of all miscellaneous operations that are timed.
MAX_TIMER_MISC	Maximum wait time of all miscellaneous operations that are timed.

I/O operations can be avoided by caching, in which case they will not be recorded in this table.

You can TRUNCATE the table, which will reset all counters to zero.

Example

```
SELECT * FROM file_summary_by_event_name\G
...
*************************** 49. row ***************************
               EVENT_NAME: wait/io/file/aria/MAD
               COUNT_STAR: 60
           SUM_TIMER_WAIT: 397234368
           MIN_TIMER_WAIT: 0
           AVG_TIMER_WAIT: 6620224
           MAX_TIMER_WAIT: 16808672
               COUNT_READ: 0
           SUM_TIMER_READ: 0
           MIN_TIMER_READ: 0
           AVG_TIMER_READ: 0
           MAX_TIMER_READ: 0
 SUM_NUMBER_OF_BYTES_READ: 0
              COUNT_WRITE: 0
          SUM_TIMER_WRITE: 0
          MIN_TIMER_WRITE: 0
          AVG_TIMER_WRITE: 0
          MAX_TIMER_WRITE: 0
SUM_NUMBER_OF_BYTES_WRITE: 0
               COUNT_MISC: 60
           SUM_TIMER_MISC: 397234368
           MIN_TIMER_MISC: 0
           AVG_TIMER_MISC: 6620224
           MAX_TIMER_MISC: 16808672
*************************** 50. row ***************************
               EVENT_NAME: wait/io/file/aria/control
               COUNT_STAR: 3
           SUM_TIMER_WAIT: 24055778544
           MIN_TIMER_WAIT: 0
           AVG_TIMER_WAIT: 8018592848
           MAX_TIMER_WAIT: 24027262400
               COUNT_READ: 1
           SUM_TIMER_READ: 24027262400
           MIN_TIMER_READ: 0
           AVG_TIMER_READ: 24027262400
           MAX_TIMER_READ: 24027262400
 SUM_NUMBER_OF_BYTES_READ: 52
              COUNT_WRITE: 0
          SUM_TIMER_WRITE: 0
          MIN_TIMER_WRITE: 0
          AVG_TIMER_WRITE: 0
          MAX_TIMER_WRITE: 0
SUM_NUMBER_OF_BYTES_WRITE: 0
               COUNT_MISC: 2
           SUM_TIMER_MISC: 28516144
           MIN_TIMER_MISC: 0
           AVG_TIMER_MISC: 14258072
           MAX_TIMER_MISC: 27262208
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema file_summary_by_instance Table

The Performance Schema `file_summary_by_instance` table contains file events summarized by instance. It contains the following columns:

Column	Description
FILE_NAME	File name.
EVENT_NAME	Event name.
OBJECT_INSTANCE_BEGIN	Address in memory. Together with FILE_NAME and EVENT_NAME uniquely identifies a row.

Column	Description
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, including FGETS, FGETC, FREAD, and READ.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
SUM_NUMBER_OF_BYTES_READ	Bytes read by read operations.
COUNT_WRITE	Number of all write operations, including FPUTS, FPUTC, FPRINTF, VFPRINTF, FWRITE, and PWRITE.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
SUM_NUMBER_OF_BYTES_WRITE	Bytes written by write operations.
COUNT_MISC	Number of all miscellaneous operations not counted above, including CREATE, DELETE, OPEN, CLOSE, STREAM_OPEN, STREAM_CLOSE, SEEK, TELL, FLUSH, STAT, FSTAT, CHSIZE, RENAME, and SYNC.
SUM_TIMER_MISC	Total wait time of all miscellaneous operations that are timed.
MIN_TIMER_MISC	Minimum wait time of all miscellaneous operations that are timed.
AVG_TIMER_MISC	Average wait time of all miscellaneous operations that are timed.
MAX_TIMER_MISC	Maximum wait time of all miscellaneous operations that are timed.

I/O operations can be avoided by caching, in which case they will not be recorded in this table.

You can TRUNCATE the table, which will reset all counters to zero.

Example

```
SELECT * FROM file_summary_by_instance\G
...
*************************** 204. row ***************************
                FILE_NAME: /var/lib/mysql/test/db.opt
               EVENT_NAME: wait/io/file/sql/dbopt
    OBJECT_INSTANCE_BEGIN: 140578961971264
               COUNT_STAR: 6
           SUM_TIMER_WAIT: 39902495024
           MIN_TIMER_WAIT: 177888
           AVG_TIMER_WAIT: 6650415692
           MAX_TIMER_WAIT: 21026400404
               COUNT_READ: 1
           SUM_TIMER_READ: 21026400404
           MIN_TIMER_READ: 21026400404
           AVG_TIMER_READ: 21026400404
           MAX_TIMER_READ: 21026400404
 SUM_NUMBER_OF_BYTES_READ: 65
              COUNT_WRITE: 0
          SUM_TIMER_WRITE: 0
          MIN_TIMER_WRITE: 0
          AVG_TIMER_WRITE: 0
          MAX_TIMER_WRITE: 0
SUM_NUMBER_OF_BYTES_WRITE: 0
               COUNT_MISC: 5
           SUM_TIMER_MISC: 18876094620
           MIN_TIMER_MISC: 177888
           AVG_TIMER_MISC: 3775218924
           MAX_TIMER_MISC: 18864558060
*************************** 205. row ***************************
                FILE_NAME: /var/log/mysql/mariadb-bin.000157
               EVENT_NAME: wait/io/file/sql/binlog
    OBJECT_INSTANCE_BEGIN: 140578961971968
               COUNT_STAR: 6
           SUM_TIMER_WAIT: 73985877680
           MIN_TIMER_WAIT: 251136
           AVG_TIMER_WAIT: 12330979468
           MAX_TIMER_WAIT: 73846656340
               COUNT_READ: 0
           SUM_TIMER_READ: 0
           MIN_TIMER_READ: 0
           AVG_TIMER_READ: 0
           MAX_TIMER_READ: 0
 SUM_NUMBER_OF_BYTES_READ: 0
              COUNT_WRITE: 2
          SUM_TIMER_WRITE: 62583004
          MIN_TIMER_WRITE: 27630192
          AVG_TIMER_WRITE: 31291284
          MAX_TIMER_WRITE: 34952812
SUM_NUMBER_OF_BYTES_WRITE: 369
               COUNT_MISC: 4
           SUM_TIMER_MISC: 73923294676
           MIN_TIMER_MISC: 251136
           AVG_TIMER_MISC: 18480823560
           MAX_TIMER_MISC: 73846656340
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema global_status Table

Note:

The `global_status` table is available from MariaDB 10.5.2.

The `global_status` table contains a list of status variables and their global values. The table only stores status variable statistics for threads which are instrumented, and does not collect statistics for `Com_xxx` variables.

The table contains the following columns:

Column	Description
VARIABLE_NAME	The global status variable name.
VARIABLE_VALUE	The global status variable value.

TRUNCATE TABLE resets global status variables, including thread, account, host, and user status, but not those that are never reset by the server.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema host_cache Table

The `host_cache` table contains host and IP information from the host_cache, used for avoiding DNS lookups for new client connections.

The host_cache table contains the following columns:

Column	Description
IP	Client IP address.
HOST	IP's resolved DNS host name, or NULL if unknown.
HOST_VALIDATED	YES if the IP-to-host DNS lookup was successful, and the HOST column can be used to avoid DNS calls, or NO if unsuccessful, in which case DNS lookup is performed for each connect until either successful or a permanent error.
SUM_CONNECT_ERRORS	Number of connection errors. Counts only protocol handshake errors for hosts that passed validation. These errors count towards max_connect_errors.
COUNT_HOST_BLOCKED_ERRORS	Number of blocked connections because SUM_CONNECT_ERRORS exceeded the max_connect_errors system variable.
COUNT_NAMEINFO_TRANSIENT_ERRORS	Number of transient errors during IP-to-host DNS lookups.
COUNT_NAMEINFO_PERMANENT_ERRORS	Number of permanent errors during IP-to-host DNS lookups.
COUNT_FORMAT_ERRORS	Number of host name format errors, for example a numeric host column.
COUNT_ADDRINFO_TRANSIENT_ERRORS	Number of transient errors during host-to-IP reverse DNS lookups.
COUNT_ADDRINFO_PERMANENT_ERRORS	Number of permanent errors during host-to-IP reverse DNS lookups.
COUNT_FCRDNS_ERRORS	Number of forward-confirmed reverse DNS errors, which occur when IP-to-host DNS lookup does not match the originating IP address.
COUNT_HOST_ACL_ERRORS	Number of errors occurring because no user from the host is permitted to log in. These attempts return error code 1130 ER_HOST_NOT_PRIVILEGED and do not proceed to username and password authentication.
COUNT_NO_AUTH_PLUGIN_ERRORS	Number of errors due to requesting an authentication plugin that was not available. This can be due to the plugin never having been loaded, or the load attempt failing.
COUNT_AUTH_PLUGIN_ERRORS	Number of errors reported by an authentication plugin. Plugins can increment COUNT_AUTHENTICATION_ERRORS or COUNT_HANDSHAKE_ERRORS instead, but, if specified or the error is unknown, this column is incremented.
COUNT_HANDSHAKE_ERRORS	Number of errors detected at the wire protocol level.

Column	Description
COUNT_PROXY_USER_ERRORS	Number of errors detected when a proxy user is proxied to a user that does not exist.
COUNT_PROXY_USER_ACL_ERRORS	Number of errors detected when a proxy user is proxied to a user that exists, but the proxy user doesn't have the PROXY privilege.
COUNT_AUTHENTICATION_ERRORS	Number of errors where authentication failed.
COUNT_SSL_ERRORS	Number of errors due to TLS problems.
COUNT_MAX_USER_CONNECTIONS_ERRORS	Number of errors due to the per-user quota being exceeded.
COUNT_MAX_USER_CONNECTIONS_PER_HOUR_ERRORS	Number of errors due to the per-hour quota being exceeded.
COUNT_DEFAULT_DATABASE_ERRORS	Number of errors due to the user not having permission to access the specified default database, or it not existing.
COUNT_INIT_CONNECT_ERRORS	Number of errors due to statements in the init_connect system variable.
COUNT_LOCAL_ERRORS	Number of local server errors, such as out-of-memory errors, unrelated to network, authentication, or authorization.
COUNT_UNKNOWN_ERRORS	Number of unknown errors that cannot be allocated to another column.
FIRST_SEEN	Timestamp of the first connection attempt by the IP.
LAST_SEEN	Timestamp of the most recent connection attempt by the IP.
FIRST_ERROR_SEEN	Timestamp of the first error seen from the IP.
LAST_ERROR_SEEN	Timestamp of the most recent error seen from the IP.

The `host_cache` table, along with the `host_cache`, is cleared with FLUSH HOSTS, TRUNCATE TABLE `host_cache` or by setting the host_cache_size system variable at runtime.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema hosts Table

Description

The `hosts` table contains a row for each host used by clients to connect to the server, containing current and total connections.

The size is determined by the performance_schema_hosts_size system variable, which, if set to zero, will disable connection statistics in the hosts table.

It contains the following columns:

Column	Description
HOST	Host name used by the client to connect, NULL for internal threads or user sessions that failed to authenticate.
CURRENT_CONNECTIONS	Current number of the host's connections.
TOTAL_CONNECTIONS	Total number of the host's connections

Example

```
SELECT * FROM hosts;
+-----------+--------------------+------------------+
| HOST      | CURRENT_CONNECTIONS | TOTAL_CONNECTIONS |
+-----------+--------------------+------------------+
| localhost |                  1 |               45 |
| NULL      |                 20 |               23 |
+-----------+--------------------+------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema memory_summary_by_account_by_event_name Table

Note:

The `memory_summary_by_account_by_event_name` table is available from MariaDB 10.5.2.

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- memory_summary_by_account_by_event_name
- memory_summary_by_host_by_event_name
- memory_summary_by_thread_by_event_name
- memory_summary_by_user_by_event_name
- memory_global_by_event_name

The `memory_summary_by_account_by_event_name` table contains memory usage statistics aggregated by account and event.

The table contains the following columns:

Field	Type	Null	Default	Description
USER	char(32)	YES	NULL	User portion of the account.
HOST	char(60)	YES	NULL	Host portion of the account.
EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).
HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).

Field	Type	Null	Default	Description
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.
CURRENT_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema memory_summary_by_host_by_event_name Table

> **Note:**
>
> The `memory_summary_by_host_by_event_name` table is available from MariaDB 10.5.2.

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- memory_summary_by_account_by_event_name
- memory_summary_by_host_by_event_name
- memory_summary_by_thread_by_event_name
- memory_summary_by_user_by_event_name
- memory_global_by_event_name

The `memory_summary_by_host_by_event_name` table contains memory usage statistics aggregated by host and event.

The table contains the following columns:

Field	Type	Null	Default	Description
HOST	char(60)	YES	NULL	Host portion of the account.
EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).
HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.

Field	Type	Null	Default	Description
CURRENT_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema memory_summary_by_thread_by_event_name Table

> **Note:**
>
> The `memory_summary_by_thread_by_event_name` table is available from MariaDB 10.5.2.

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- memory_summary_by_account_by_event_name
- memory_summary_by_host_by_event_name
- memory_summary_by_thread_by_event_name
- memory_summary_by_user_by_event_name
- memory_global_by_event_name

The `memory_summary_by_thread_by_event_name` table contains memory usage statistics aggregated by thread and event.

The table contains the following columns:

Field	Type	Null	Default	Description
THREAD_ID	bigint(20) unsigned	NO	NULL	Thread id.
EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).
HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.
CURRENT_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).

Field	Type	Null	Default	Description
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema memory_summary_by_user_by_event_name Table

> **Note:**
>
> The `memory_summary_by_user_by_event_name` table is available from MariaDB 10.5.2.

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- memory_summary_by_account_by_event_name
- memory_summary_by_host_by_event_name
- memory_summary_by_thread_by_event_name
- memory_summary_by_user_by_event_name
- memory_global_by_event_name

The `memory_summary_by_user_by_event_name` table contains memory usage statistics aggregated by user and event.

The table contains the following columns:

Field	Type	Null	Default	Description
USER	char(32)	YES	NULL	User portion of the account.
EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).
HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.
CURRENT_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema memory_summary_global_by_event_name Table

> **Note:**
>
> The `memory_summary_global_by_event_name` table is available from MariaDB 10.5.2.

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- memory_summary_by_account_by_event_name
- memory_summary_by_host_by_event_name
- memory_summary_by_thread_by_event_name
- memory_summary_by_user_by_event_name
- memory_summary_global_by_event_name

The `memory_summary_global_by_event_name` table contains memory usage statistics aggregated by event and event.

The table contains the following columns:

Field	Type	Null	Default	Description
EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).
HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.
CURRENT_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

Example

Seeing what memory was most often allocated for:

```
SELECT * FROM memory_summary_global_by_event_name
  ORDER BY count_alloc DESC LIMIT 1\G
*************************** 1. row ***************************
                 EVENT_NAME: memory/sql/QUICK_RANGE_SELECT::alloc
                COUNT_ALLOC: 147976
                 COUNT_FREE: 147976
   SUM_NUMBER_OF_BYTES_ALLOC: 600190656
    SUM_NUMBER_OF_BYTES_FREE: 600190656
              LOW_COUNT_USED: 0
          CURRENT_COUNT_USED: 0
             HIGH_COUNT_USED: 68
    LOW_NUMBER_OF_BYTES_USED: 0
CURRENT_NUMBER_OF_BYTES_USED: 0
   HIGH_NUMBER_OF_BYTES_USED: 275808
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema metadata_locks Table

> **Note:**
>
> The `metadata_locks` table is available from MariaDB 10.5.2.

Description

The `metadata_locks` table contains metadata lock information.

To enable metadata lock instrumentation at runtime, issue this statement:

```
UPDATE performance_schema.setup_instruments SET enabled='YES', timed='YES'
  WHERE name LIKE 'wait/lock/metadata%';
```

To enable it permanently, add this to the `[mariadb]` section of the configuration file (for instance, `my.cnf`):

```
performance-schema-instrument='wait/lock/metadata/sql/mdl=ON'
```

> **Note:**
>
> Performance Schema must be enabled in the configuration file for this to work (add `performance_schema=ON`).

The table is by default autosized, but the size can be configured with the performance_schema_max_metadata_locks system variable.

The table is read-only, and TRUNCATE TABLE cannot be used to empty the table.

The table contains the following columns:

- `OBJECT_TYPE`
  - `VARCHAR(64)`
  - Null: No
  - Default: `NULL`
  - Description: Object type. One of `BACKUP`, `COMMIT`, `EVENT`, `FUNCTION`, `GLOBAL`, `LOCKING SERVICE`, `PROCEDURE`, `SCHEMA`, `TABLE`, `TABLESPACE`, `TRIGGER` (unused) or `USER LEVEL LOCK`.
- `OBJECT_SCHEMA`
  - `VARCHAR(64)`
  - Null: Yes
  - Default: `NULL`

    - Description: Object schema.
- `OBJECT_NAME`
    - `VARCHAR(64)`
    - Null: Yes
    - Default: `NULL`
    - Description: Object name.
- `OBJECT_INSTANCE_BEGIN`
    - `BIGINT(20) UNSIGNED`
    - Null: No
    - Default: `NULL`
    - Description: Address in memory of the instrumented object.
- `LOCK_TYPE`
    - `VARCHAR(32)`
    - Null: No
    - Default: `NULL`
    - Description: Lock type. One of `BACKUP_ALTER_COPY`, `BACKUP_BLOCK_DDL`, `BACKUP_COMMIT`, `BACKUP_DDL`, `BACKUP_DML`, `BACKUP_FLUSH`, `BACKUP_FTWRL1`, `BACKUP_START`, `BACKUP_TRANS_DML`, `BACKUP_WAIT_COMMIT`, `BACKUP_WAIT_DDL`, `BACKUP_WAIT_FLUSH`. See the Backup Lock Types section for detailed descriptions.
- `LOCK_DURATION`
    - `VARCHAR(32)`
    - Null: No
    - Default: `NULL`
    - Description: Lock duration. One of `EXPLICIT` (locks released by explicit action, for example a global lock acquired with `FLUSH TABLES WITH READ LOCK`), `STATEMENT` (locks implicitly released at statement end), or `TRANSACTION` (locks implicitly released at transaction end).
- `LOCK_STATUS`
    - `VARCHAR(32)`
    - Null: No
    - Default: `NULL`
    - Description: Lock status. One of `GRANTED`, `KILLED`, `PENDING`, `POST_RELEASE_NOTIFY`, `PRE_ACQUIRE_NOTIFY`, `TIMEOUT`, or `VICTIM`.
- `SOURCE`
    - `VARCHAR(64)`
    - Null: Yes
    - Default: `NULL`
    - Description: Source file containing the instrumented code that produced the event, as well as the line number where the instrumentation occurred. This allows one to examine the source code involved.
- `OWNER_THREAD_ID`
    - `BIGINT(20) UNSIGNED`
    - Null: Yes
    - Default: `NULL`
    - Description: Thread that requested the lock.
- `OWNER_EVENT_ID`
    - `BIGINT(20) UNSIGNED`

- Null: Yes
- Default: `NULL`
- Description: Event that requested the lock.

Example

```
MariaDB [performance_schema]> SELECT * FROM metadata_locks\G
*************************** 1. row ***************************
          OBJECT_TYPE: TABLE
        OBJECT_SCHEMA: performance_schema
          OBJECT_NAME: metadata_locks
OBJECT_INSTANCE_BEGIN: 105553150198240
            LOCK_TYPE: SHARED_READ
        LOCK_DURATION: TRANSACTION
          LOCK_STATUS: GRANTED
               SOURCE:
      OWNER_THREAD_ID: 13
       OWNER_EVENT_ID: 1
1 row in set (0.001 sec)
```

Backup Lock Types

`BACKUP_*` lock types exposed through this table allow for granular locking, as opposed to the `FLUSH TABLES WITH READ LOCK` (FTWRL) statement which essentially freezes the entire database. With granular locking, backup tools like mariadb-backup can take consistent backups while keeping the database functioning.

BACKUP_ALTER_COPY

This lock is acquired by the backup engine when it needs to perform an `ALTER TABLE ... COPY` operation.

BACKUP_BLOCK_DDL

This lock is acquired by other operations that need to block DDL statements from being executed during a backup.

BACKUP_COMMIT

This lock is acquired by the backup engine when it needs to commit a transaction.

BACKUP_DDL

This lock is acquired by the backup engine when it needs to process DDL statements.

BACKUP_DML

This lock is acquired by the backup engine when it needs to read data from tables.

BACKUP_FLUSH

This lock is acquired when the backup engine flushes data to disk.

BACKUP_FTWRL1

This is a more refined, less aggressive version of the `FLUSH TABLES WITH READ LOCK` statement. The “1” in the name signifies it is a specific stage of the backup lock hierarchy.

- This lock acts as a short-lived synchronization point. It ensures all non-transactional tables (like MyISAM) are flushed to disk and that the binary log position is captured accurately.
- Since InnoDB handles its own consistency, this lock is primarily for “everything else” in your database – ensuring that the metadata and non-InnoDB tables are in a fixed state for a split second.

BACKUP_START

As the name suggests, this is the initialization lock. When a backup starts, the system needs to set a baseline.

- This lock **prevents DDL[45] operations**. This means you cannot `CREATE`, `ALTER`, `RENAME`, or `DROP` tables while this lock is held.
- This lock protects the table structure during file-copy operations.

BACKUP_TRANS_DML

This lock is the lightest of the three (`BACKUP_START`, `BACKUP_FTWRL1`). It is designed to ensure that DML[46] operations – like `INSERT`, `UPDATE`, and `DELETE` – don't interfere with the backup of transactional tables (for instance, InnoDB).

- This lock **prevents DML changes** to transactional tables during a specific phase of the backup.
- It ensures that the backup tool can reach a consistent state without the data constantly shifting under its feet, but it doesn't block simple `SELECT` queries.

BACKUP_WAIT_COMMIT

This lock is acquired by other operations that need to wait for the backup engine to finish processing commit statements.

BACKUP_WAIT_DDL

This lock is acquired by other operations that need to wait for the backup engine to finish processing DDL statements.

BACKUP_WAIT_FLUSH

This lock is acquired by other operations that need to wait for the backup engine to finish flushing data.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema mutex_instances Table

Description

The `mutex_instances` table lists all mutexes that the Performance Schema seeing while the server is executing.

A mutex is a code mechanism for ensuring that threads can only access resources one at a time. A second thread attempting to access a resource will find it protected by a mutex, and will wait for it to be unlocked.

The performance_schema_max_mutex_instances system variable specifies the maximum number of instrumented mutex instances.

Column	Description
NAME	Instrument name associated with the mutex.
OBJECT_INSTANCE_BEGIN	Memory address of the instrumented mutex.
LOCKED_BY_THREAD_ID	The THREAD_ID of the locking thread if a thread has a mutex locked, otherwise NULL.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema objects_summary_global_by_type Table

It aggregates object wait events, and contains the following columns:

Column	Description
OBJECT_TYPE	Groups records together with OBJECT_SCHEMA and OBJECT_NAME.
OBJECT_SCHEMA	Groups records together with OBJECT_TYPE and OBJECT_NAME.
OBJECT_NAME	Groups records together with OBJECT_SCHEMA and OBJECT_TYPE.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

You can TRUNCATE the table, which will reset all counters to zero.

Example

```
SELECT * FROM objects_summary_global_by_type\G
...
*************************** 101. row ***************************
   OBJECT_TYPE: TABLE
 OBJECT_SCHEMA: test
   OBJECT_NAME: v
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*************************** 102. row ***************************
   OBJECT_TYPE: TABLE
 OBJECT_SCHEMA: test
   OBJECT_NAME: xx2
    COUNT_STAR: 2
SUM_TIMER_WAIT: 1621920
MIN_TIMER_WAIT: 481344
AVG_TIMER_WAIT: 810960
MAX_TIMER_WAIT: 1140576
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema performance_timers Table

Description

The `performance_timers` table lists available event timers.

It contains the following columns:

Column	Description
TIMER_NAME	Time name, used in the setup_timers table.
TIMER_FREQUENCY	Number of timer units per second. Dependent on the processor speed.
TIMER_RESOLUTION	Number of timer units by which timed values increase each time.
TIMER_OVERHEAD	Minimum timer overhead, determined during initialization by calling the timer 20 times and selecting the smallest value. Total overhead will be at least double this, as the timer is called at the beginning and end of each timed event.

Any `NULL` values indicate that particular timer is not available on your platform, Any timer names with a non-NULL value can be used in the setup_timers table.

Example

```
SELECT * FROM performance_timers;
+-------------+-----------------+------------------+---------------------+
| TIMER_NAME  | TIMER_FREQUENCY | TIMER_RESOLUTION | TIMER_OVERHEAD      |
+-------------+-----------------+------------------+---------------------+
| CYCLE       |      2293651741 |                1 |                  28 |
| NANOSECOND  |      1000000000 |                1 |                  48 |
| MICROSECOND |         1000000 |                1 |                  52 |
| MILLISECOND |            1000 |             1000 | 9223372036854775807 |
| TICK        |             106 |                1 |                 496 |
+-------------+-----------------+------------------+---------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema prepared_statements_instances Table

> **Note:**
>
> The `prepared_statements_instances` table is available from MariaDB 10.5.2.

The `prepared_statements_instances` table contains aggregated statistics of prepared statements.

The maximum number of rows in the table is determined by the performance_schema_max_prepared_statement_instances system variable, which is by default autosized on startup.

The table contains the following columns:

```
+-----------------------------+-------------------------------------------------------------+------+-----+----
-----+-------+
| Field                       | Type                                                        | Null | Key |
Default | Extra |
+-----------------------------+-------------------------------------------------------------+------+-----+----
-----+-------+
| OBJECT_INSTANCE_BEGIN       | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| STATEMENT_ID                | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| STATEMENT_NAME              | varchar(64)                                                 | YES  |     |
NULL    |       |
| SQL_TEXT                    | longtext                                                    | NO   |     |
NULL    |       |
| OWNER_THREAD_ID             | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| OWNER_EVENT_ID              | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| OWNER_OBJECT_TYPE           | enum('EVENT','FUNCTION','PROCEDURE','TABLE','TRIGGER') | YES  |     |
NULL    |       |
| OWNER_OBJECT_SCHEMA         | varchar(64)                                                 | YES  |     |
NULL    |       |
| OWNER_OBJECT_NAME           | varchar(64)                                                 | YES  |     |
NULL    |       |
| TIMER_PREPARE               | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| COUNT_REPREPARE             | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| COUNT_EXECUTE               | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_TIMER_EXECUTE           | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| MIN_TIMER_EXECUTE           | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| AVG_TIMER_EXECUTE           | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| MAX_TIMER_EXECUTE           | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_LOCK_TIME               | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_ERRORS                  | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_WARNINGS                | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_ROWS_AFFECTED           | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_ROWS_SENT               | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_ROWS_EXAMINED           | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_CREATED_TMP_DISK_TABLES | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_CREATED_TMP_TABLES      | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_SELECT_FULL_JOIN        | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_SELECT_FULL_RANGE_JOIN  | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_SELECT_RANGE            | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_SELECT_RANGE_CHECK      | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_SELECT_SCAN             | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_SORT_MERGE_PASSES       | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
| SUM_SORT_RANGE              | bigint(20) unsigned                                         | NO   |     |
NULL    |       |
```

```
| SUM_SORT_ROWS               | bigint(20) unsigned                                                | NO   |     |
NULL    |       |
| SUM_SORT_SCAN               | bigint(20) unsigned                                                | NO   |     |
NULL    |       |
| SUM_NO_INDEX_USED           | bigint(20) unsigned                                                | NO   |     |
NULL    |       |
| SUM_NO_GOOD_INDEX_USED      | bigint(20) unsigned                                                | NO   |     |
NULL    |       |
+-----------------------------+--------------------------------------------------------------------+------+-----+----
-----+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema replication_applier_configuration Table

> **Note:**
>
> The `replication_applier_configuration` table is available from MariaDB 10.5.2.

The Performance Schema replication_applier_configuration table contains configuration settings affecting replica transactions.

It contains the following fields.

Field	Type	Null	Description
CHANNEL_NAME	char(64)	NO	Replication channel name.
DESIRED_DELAY	int(11)	NO	Target number of seconds the replica should be delayed to the master.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema replication_applier_status Table

> **Note:**
>
> The `replication_applier_status` table is available from MariaDB 10.5.2.

The Performance Schema replication_applier_status table contains information about the general transaction execution status on the replica.

It contains the following fields.

Field	Type	Null	Description
CHANNEL_NAME	char(64)	NO	The replication channel name.
SERVICE_STATE	enum('ON','OFF')	NO	Shows ON when the replication channel's applier threads are active or idle, OFF means that the applier threads are not active.
REMAINING_DELAY	int(10) unsigned	YES	Seconds the replica needs to wait to reach the desired delay from master.
COUNT_TRANSACTIONS_RETRIES	bigint(20) unsigned	NO	The number of retries that were made because the replication SQL thread failed to apply a transaction.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema replication_applier_status_by_coordinator Table

> **Note:**
>
> The `replication_applier_status_by_coordinator` table is available from MariaDB 10.5.2.

The Performance Schema replication_applier_status_by_coordinator table displays the status of the coordinator thread used in multi-threaded replicas to manage multiple worker threads.

It contains the following fields.

Column	Type	Null	Description
CHANNEL_NAME	varchar(256)	NO	Replication channel name.
THREAD_ID	bigint(20) unsigned	YES	The SQL/coordinator thread ID.
SERVICE_STATE	enum('ON','OFF')	NO	ON (thread exists and is active or idle) or OFF (thread no longer exists).
LAST_ERROR_NUMBER	int(11)	NO	Last error number that caused the SQL/coordinator thread to stop.
LAST_ERROR_MESSAGE	varchar(1024)	NO	Last error message that caused the SQL/coordinator thread to stop.
LAST_ERROR_TIMESTAMP	timestamp	NO	Timestamp that shows when the most recent SQL/coordinator error occurred.
LAST_SEEN_TRANSACTION	char(57)	NO	The transaction the worker has last seen.
LAST_TRANS_RETRY_COUNT	int(11)	NO	Total number of retries attempted by last transaction.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema replication_applier_status_by_worker Table

> **Note:**
>
> The `replication_applier_status_by_worker` table is available from MariaDB 10.6.0.

The Performance Schema replication_applier_status_by_worker table displays replica worker thread specific information.

It contains the following fields.

Column	Description
CHANNEL_NAME	Name of replication channel through which the transaction is received.
THREAD_ID	Thread_Id as displayed in the performance_schema.threads table for thread with name 'thread/sql/rpl_parallel_thread'. THREAD_ID will be NULL when worker threads are stopped due to error/force stop.
SERVICE_STATE	Whether or not the thread is running.
LAST_SEEN_TRANSACTION	Last GTID executed by worker
LAST_ERROR_NUMBER	Last Error that occurred on a particular worker.
LAST_ERROR_MESSAGE	Last error specific message.
LAST_ERROR_TIMESTAMP	Time stamp of last error.
WORKER_IDLE_TIME	Total idle time in seconds that the worker thread has spent waiting for work from SQL thread.

Column	Description
LAST_TRANS_RETRY_COUNT	Total number of retries attempted by last transaction.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema replication_connection_configuration Table

> **Note:**
>
> The `replication_connection_configuration` table is available from MariaDB 10.6.0.

The Performance Schema replication_connection_configuration table displays replica's configuration settings used for connecting to the primary.

It contains the following fields.

Column	Type	Null	Description
CHANNEL_NAME	varchar(256)	NO	The replication channel used.
HOST	char(60)	NO	The host name of the source that the replica is connected to.
PORT	int(11)	NO	The port used to connect to the source.
USER	char(32)	NO	The user name of the replication user account used to connect to the source.
USING_GTID	enum('NO', 'CURRENT_POS', 'SLAVE_POS')	NO	Whether replication is using GTIDs or not.
SSL_ALLOWED	enum('YES', 'NO', 'IGNORED')	NO	Whether SSL is allowed for the replica connection.
SSL_CA_FILE	varchar(512)	NO	Path to the file that contains one or more certificates for trusted Certificate Authorities (CA) to use for TLS.
SSL_CA_PATH	varchar(512)	NO	Path to a directory that contains one or more PEM files that contain X509 certificates for a trusted Certificate Authority (CA) to use for TLS.
SSL_CERTIFICATE	varchar(512)	NO	Path to the certificate used to authenticate the master.
SSL_CIPHER	varchar(512)	NO	Which cipher is used for encryption.
SSL_KEY	varchar(512)	NO	Path to the private key used for TLS.
SSL_VERIFY_SERVER_CERTIFICATE	enum('YES','NO')	NO	Whether the server certificate is verified as part of the SSL connection.
SSL_CRL_FILE	varchar(255)	NO	Path to the PEM file containing one or more revoked X.509 certificates.
SSL_CRL_PATH	varchar(255)	NO	PATH to a folder containing PEM files containing one or more revoked X.509 certificates.
CONNECTION_RETRY_INTERVAL	int(11)	NO	The number of seconds between connect retries.
CONNECTION_RETRY_COUNT	bigint(20) unsigned	NO	The number of times the replica can attempt to reconnect to the source in the event of a lost connection.

Column	Type	Null	Description
HEARTBEAT_INTERVAL	double(10,3) unsigned	NO	Number of seconds after which a heartbeat will be sent.
IGNORE_SERVER_IDS	longtext	NO	Binary log events from servers (ids) to ignore.
REPL_DO_DOMAIN_IDS	longtext	NO	Only apply binary logs from these domain ids.
REPL_IGNORE_DOMAIN_IDS	longtext	NO	Binary log events from domains to ignore.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema rwlock_instances Table

The `rwlock_instances` table lists all read write lock (rwlock) instances that the Performance Schema sees while the server is executing. A read write is a mechanism for ensuring threads can either share access to common resources, or have exclusive access.

The performance_schema_max_rwlock_instances system variable specifies the maximum number of instrumented rwlock objects.

The `rwlock_instances` table contains the following columns:

Column	Description
NAME	Instrument name associated with the read write lock
OBJECT_INSTANCE_BEGIN	Address in memory of the instrumented lock
WRITE_LOCKED_BY_THREAD_ID	THREAD_ID of the locking thread if locked in write (exclusive) mode, otherwise NULL.
READ_LOCKED_BY_COUNT	Count of current read locks held

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema session_account_connect_attrs Table

Description

The `session_account_connect_attrs` table shows connection attributes for the current session.

Applications can pass key/value connection attributes to the server when a connection is made. The session_connect_attrs and `session_account_connect_attrs` tables provide access to this information, for all sessions and the current session respectively.

The C API functions mysql_options() and mysql_optionsv() are used for passing connection attributes to the server.

`session_account_connect_attrs` contains the following columns:

Column	Description
PROCESSLIST_ID	Session connection identifier.
ATTR_NAME	Attribute name.
ATTR_VALUE	Attribute value.
ORDINAL_POSITION	Order in which attribute was added to the connection attributes.

Example

```
SELECT * FROM performance_schema.session_account_connect_attrs;
+----------------+-----------------+------------------+------------------+
| PROCESSLIST_ID | ATTR_NAME       | ATTR_VALUE       | ORDINAL_POSITION |
+----------------+-----------------+------------------+------------------+
|             45 | _os             | debian-linux-gnu |                0 |
|             45 | _client_name    | libmysql         |                1 |
|             45 | _pid            | 7711             |                2 |
|             45 | _client_version | 10.0.5           |                3 |
|             45 | _platform       | x86_64           |                4 |
|             45 | program_name    | mysql            |                5 |
+----------------+-----------------+------------------+------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema session_connect_attrs Table

Description

`session_connect_attrs` is a Performance Schema table that shows connection attributes for all sessions. The Performance Schema needs to be enabled for the table to be populated.

Applications can pass key/value connection attributes to the server when a connection is made. The `session_connect_attrs` and session_account_connect_attrs tables provide access to this information, for all sessions and the current session respectively.

The C API functions mysql_options() and mysql_optionsv() are used for passing connection attributes to the server.

`session_connect_attrs` contains the following columns:

Column	Description
PROCESSLIST_ID	Session connection identifier.
ATTR_NAME	Attribute name.
ATTR_VALUE	Attribute value.
ORDINAL_POSITION	Order in which attribute was added to the connection attributes.

Example

Returning the current connection's attributes:

```
SELECT * FROM performance_schema.session_connect_attrs WHERE processlist_id=CONNECTION_ID();
+----------------+-----------------+------------------+------------------+
| PROCESSLIST_ID | ATTR_NAME       | ATTR_VALUE       | ORDINAL_POSITION |
+----------------+-----------------+------------------+------------------+
|             45 | _os             | debian-linux-gnu |                0 |
|             45 | _client_name    | libmysql         |                1 |
|             45 | _pid            | 7711             |                2 |
|             45 | _client_version | 10.0.5           |                3 |
|             45 | _platform       | x86_64           |                4 |
|             45 | program_name    | mysql            |                5 |
+----------------+-----------------+------------------+------------------+
```

Using Other Connectors

JDBC

Connection attributes values are set using the option connectionAttributes.

Example using connection string `jdbc:mariadb://localhost/?connectionAttributes=test:test1,test2:test2Val,test3`:

```
SELECT * FROM performance_schema.session_connect_attrs WHERE processlist_id=17;
+---------------+-----------------+---------------------+------------------+
| PROCESSLIST_ID | ATTR_NAME       | ATTR_VALUE          | ORDINAL_POSITION |
+---------------+-----------------+---------------------+------------------+
|             17 | _client_name    | MariaDB Connector/J |                0 |
|             17 | _client_version | 3.1.3               |                1 |
|             17 | _server_host    | localhost           |                2 |
|             17 | _os             | Windows 11          |                3 |
|             17 | _thread         | 1                   |                4 |
|             17 | _java_vendor    | Oracle Corporation  |                5 |
|             17 | _java_version   | 19.0.2              |                6 |
|             17 | test            | test1               |                7 |
|             17 | test2           | test2Val            |                8 |
|             17 | test3           | NULL                |                9 |
+---------------+-----------------+---------------------+------------------+
```

Node.js

Connection attributes values are set using the option connectAttributes.

Example using connection

```
const conn = await mariadb.createConnection({
  host: 'localhost',
  user: 'root',
  connectAttributes: { test: 'test1', test2: 'test2Val', test3: 'f' }
 });
```

```
SELECT * FROM performance_schema.session_connect_attrs WHERE processlist_id=30;
+---------------+-----------------+-----------------------+------------------+
| PROCESSLIST_ID | ATTR_NAME       | ATTR_VALUE            | ORDINAL_POSITION |
+---------------+-----------------+-----------------------+------------------+
|             30 | _client_name    | MariaDB connector/Node |                0 |
|             30 | _client_version | 3.1.1                 |                1 |
|             30 | _server_host    | ::1                   |                2 |
|             30 | _os             | win32                 |                3 |
|             30 | _client_host    | NOSTROMO              |                4 |
|             30 | _node_version   | 18.15.0               |                5 |
|             30 | test            | test1                 |                6 |
|             30 | test2           | test2Val              |                7 |
|             30 | test3           | f                     |                8 |
+---------------+-----------------+-----------------------+------------------+
```

R2DBC

Connection attributes values are set using the option connectionAttributes.

Example using connection string `jdbc:mariadb://localhost/?connectionAttributes=test:test1,test2:test2Val,test3`

```
TreeMap<String, String> connectionAttributes = new TreeMap<>();
    connectionAttributes.put("entry1", "val1");
    connectionAttributes.put("entry2", "val2");

     MariadbConnectionConfiguration conf = MariadbConnectionConfiguration.builder()
             .host("localhost")
             .port(3306)
             .username("root")
             .connectionAttributes(connectionAttributes)
             .database("test")
             .build();
     MariadbConnectionFactory connFactory = new MariadbConnectionFactory(conf);
     MariadbConnection connection = connFactory.create().block();
```

```
SELECT * FROM performance_schema.session_connect_attrs WHERE processlist_id=34;
+----------------+-----------------+--------------------+------------------+
| PROCESSLIST_ID | ATTR_NAME       | ATTR_VALUE         | ORDINAL_POSITION |
+----------------+-----------------+--------------------+------------------+
|             34 | _client_name    | mariadb            |                0 |
|             34 | _client_version | 1.1.4              |                1 |
|             34 | _server_host    | localhost          |                2 |
|             34 | _os             | Windows 11         |                3 |
|             34 | _thread         | 49                 |                4 |
|             34 | _java_vendor    | Oracle Corporation |                5 |
|             34 | _java_version   | 19.0.2             |                6 |
|             34 | entry1          | val1               |                7 |
|             34 | entry2          | val2               |                8 |
+----------------+-----------------+--------------------+------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema session_status Table

> **Note:**
>
> The `session_status` table is available from MariaDB 10.5.2.

The `session_status` table contains a list of status variables for the current session. The table only stores status variable statistics for threads which are instrumented, and does not collect statistics for `Com_xxx` variables.

The table contains the following columns:

Column	Description
VARIABLE_NAME	The session status variable name.
VARIABLE_VALUE	The session status variable value.

It is not possible to empty this table with a `TRUNCATE TABLE` statement.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema setup_actors Table

The `setup_actors` table contains information for determining whether monitoring should be enabled for new client connection threads.

The default size is 100 rows, which can be changed by modifying the performance_schema_setup_actors_size system variable at server startup.

If a row in the table matches a new foreground thread's client and host, the matching `INSTRUMENTED` column in the threads table is set to either `YES` or `NO`, which allows selective application of instrumenting by host, by user, or combination thereof.

Column	Description
HOST	Host name, either a literal, or the % wildcard representing any host.
USER	User name, either a literal or the % wildcard representing any name.
ROLE	Unused

Initially, any user and host is matched:

```
SELECT * FROM performance_schema.setup_actors;
+------+------+------+
| HOST | USER | ROLE |
+------+------+------+
| %    | %    | %    |
+------+------+------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema setup_consumers Table

Lists the types of consumers for which event information is available.

The `setup_consumers` table contains the following columns:

Column	Description
NAME	Consumer name
ENABLED	YES or NO for whether or not the consumer is enabled. You can modify this column to ensure that event information is added, or is not added.

The table can be modified directly, or the server started with the option enabled, for example:

```
performance-schema-consumer-events-waits-history=ON
```

Example

```
SELECT * FROM performance_schema.setup_consumers;

+--------------------------------+---------+
| NAME                           | ENABLED |
+--------------------------------+---------+
| events_stages_current          | NO      |
| events_stages_history          | NO      |
| events_stages_history_long     | NO      |
| events_statements_current      | YES     |
| events_statements_history      | NO      |
| events_statements_history_long | NO      |
| events_waits_current           | NO      |
| events_waits_history           | NO      |
| events_waits_history_long      | NO      |
| global_instrumentation         | YES     |
| thread_instrumentation         | YES     |
| statements_digest              | YES     |
+--------------------------------+---------+
```

See Also

- Sys Schema ps_is_consumer_enabled function

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema setup_instruments Table

The `setup_instruments` table contains a list of instrumented object classes for which it is possible to collect events. There is one row for each instrument in the source code. When an instrument is enabled and executed, instances are created which are then stored in the cond_instances, file_instances, mutex_instances, rwlock_instances or socket_instance tables.

It contains the following columns:

Column	Description
NAME	Instrument name
ENABLED	Whether or not the instrument is enabled. It can be disabled, and the instrument will produce no events.
TIMED	Whether or not the instrument is timed. It can be set, but if disabled, events produced by the instrument will have NULL values for the corresponding TIMER_START, TIMER_END, and TIMER_WAIT values.

Example

Default settings with the Performance Schema enabled:

```
SELECT * FROM setup_instruments ORDER BY name;
+-------------------------------------------------------------------------------------+---------+-------+
| NAME                                                                                | ENABLED | TIMED |
+-------------------------------------------------------------------------------------+---------+-------+
| idle                                                                                | YES     | YES   |
| memory/csv/blobroot                                                                 | NO      | NO    |
| memory/csv/row                                                                      | NO      | NO    |
| memory/csv/tina_set                                                                 | NO      | NO    |
| memory/csv/TINA_SHARE                                                               | NO      | NO    |
| memory/csv/Transparent_file                                                         | NO      | NO    |
| memory/innodb/adaptive hash index                                                   | NO      | NO    |
| memory/innodb/btr0btr                                                               | NO      | NO    |
| memory/innodb/btr0buf                                                               | NO      | NO    |
| memory/innodb/btr0bulk                                                              | NO      | NO    |
| memory/innodb/btr0cur                                                               | NO      | NO    |
| memory/innodb/btr0pcur                                                              | NO      | NO    |
| memory/innodb/btr0sea                                                               | NO      | NO    |
| memory/innodb/buf0buf                                                               | NO      | NO    |
| memory/innodb/buf0dblwr                                                             | NO      | NO    |
| memory/innodb/buf0dump                                                              | NO      | NO    |
| memory/innodb/buf_buf_pool                                                          | NO      | NO    |
| memory/innodb/dict0dict                                                             | NO      | NO    |
| memory/innodb/dict0mem                                                              | NO      | NO    |
| memory/innodb/dict0stats                                                            | NO      | NO    |
| memory/innodb/dict_stats_bg_recalc_pool_t                                           | NO      | NO    |
| memory/innodb/dict_stats_index_map_t                                                | NO      | NO    |
| memory/innodb/dict_stats_n_diff_on_level                                            | NO      | NO    |
| memory/innodb/eval0eval                                                             | NO      | NO    |
| memory/innodb/fil0crypt                                                             | NO      | NO    |
| memory/innodb/fil0fil                                                               | NO      | NO    |
| memory/innodb/fsp0file                                                              | NO      | NO    |
| memory/innodb/fts0ast                                                               | NO      | NO    |
| memory/innodb/fts0blex                                                              | NO      | NO    |
| memory/innodb/fts0config                                                            | NO      | NO    |
| memory/innodb/fts0file                                                              | NO      | NO    |
| memory/innodb/fts0fts                                                               | NO      | NO    |
| memory/innodb/fts0opt                                                               | NO      | NO    |
| memory/innodb/fts0pars                                                              | NO      | NO    |
| memory/innodb/fts0que                                                               | NO      | NO    |
| memory/innodb/fts0sql                                                               | NO      | NO    |
| memory/innodb/fts0tlex                                                              | NO      | NO    |
| memory/innodb/gis0sea                                                               | NO      | NO    |
| memory/innodb/handler0alter                                                         | NO      | NO    |
| memory/innodb/hash0hash                                                             | NO      | NO    |
| memory/innodb/ha_innodb                                                             | NO      | NO    |
| memory/innodb/i_s                                                                   | NO      | NO    |
| memory/innodb/lexyy                                                                 | NO      | NO    |
| memory/innodb/lock0lock                                                             | NO      | NO    |
| memory/innodb/mem0mem                                                               | NO      | NO    |
| memory/innodb/os0event                                                              | NO      | NO    |
| memory/innodb/os0file                                                               | NO      | NO    |
| memory/innodb/other                                                                 | NO      | NO    |
| memory/innodb/pars0lex                                                              | NO      | NO    |
| memory/innodb/rem0rec                                                               | NO      | NO    |
| memory/innodb/row0ftsort                                                            | NO      | NO    |
| memory/innodb/row0import                                                            | NO      | NO    |
| memory/innodb/row0log                                                               | NO      | NO    |
| memory/innodb/row0merge                                                             | NO      | NO    |
| memory/innodb/row0mysql                                                             | NO      | NO    |
| memory/innodb/row0sel                                                               | NO      | NO    |
| memory/innodb/row_log_buf                                                           | NO      | NO    |
| memory/innodb/row_merge_sort                                                        | NO      | NO    |
| memory/innodb/srv0start                                                             | NO      | NO    |
| memory/innodb/std                                                                   | NO      | NO    |
| memory/innodb/sync0arr                                                              | NO      | NO    |
| memory/innodb/sync0debug                                                            | NO      | NO    |
| memory/innodb/sync0rw                                                               | NO      | NO    |
| memory/innodb/sync0start                                                            | NO      | NO    |
```

```
| memory/innodb/sync0types                                                | NO      | NO    |
| memory/innodb/trx0i_s                                                   | NO      | NO    |
| memory/innodb/trx0roll                                                  | NO      | NO    |
| memory/innodb/trx0rseg                                                  | NO      | NO    |
| memory/innodb/trx0seg                                                   | NO      | NO    |
| memory/innodb/trx0trx                                                   | NO      | NO    |
| memory/innodb/trx0undo                                                  | NO      | NO    |
| memory/innodb/ut0list                                                   | NO      | NO    |
| memory/innodb/ut0mem                                                    | NO      | NO    |
| memory/innodb/ut0new                                                    | NO      | NO    |
| memory/innodb/ut0pool                                                   | NO      | NO    |
| memory/innodb/ut0rbt                                                    | NO      | NO    |
| memory/innodb/ut0wqueue                                                 | NO      | NO    |
| memory/innodb/xtrabackup                                                | NO      | NO    |
| memory/memory/HP_INFO                                                   | NO      | NO    |
| memory/memory/HP_KEYDEF                                                 | NO      | NO    |
| memory/memory/HP_PTRS                                                   | NO      | NO    |
| memory/memory/HP_SHARE                                                  | NO      | NO    |
| memory/myisam/filecopy                                                  | NO      | NO    |
| memory/myisam/FTB                                                       | NO      | NO    |
| memory/myisam/FTPARSER_PARAM                                            | NO      | NO    |
| memory/myisam/FT_INFO                                                   | NO      | NO    |
| memory/myisam/ft_memroot                                                | NO      | NO    |
| memory/myisam/ft_stopwords                                              | NO      | NO    |
| memory/myisam/keycache_thread_var                                       | NO      | NO    |
| memory/myisam/MI_DECODE_TREE                                            | NO      | NO    |
| memory/myisam/MI_INFO                                                   | NO      | NO    |
| memory/myisam/MI_INFO::bulk_insert                                      | NO      | NO    |
| memory/myisam/MI_INFO::ft1_to_ft2                                       | NO      | NO    |
| memory/myisam/MI_SORT_PARAM                                             | NO      | NO    |
| memory/myisam/MI_SORT_PARAM::wordroot                                   | NO      | NO    |
| memory/myisam/MYISAM_SHARE                                              | NO      | NO    |
| memory/myisam/MYISAM_SHARE::decode_tables                               | NO      | NO    |
| memory/myisam/preload_buffer                                            | NO      | NO    |
| memory/myisam/record_buffer                                             | NO      | NO    |
| memory/myisam/SORT_FT_BUF                                               | NO      | NO    |
| memory/myisam/SORT_INFO::buffer                                         | NO      | NO    |
| memory/myisam/SORT_KEY_BLOCKS                                           | NO      | NO    |
| memory/myisam/stPageList::pages                                         | NO      | NO    |
| memory/myisammrg/children                                               | NO      | NO    |
| memory/myisammrg/MYRG_INFO                                              | NO      | NO    |
| memory/partition/ha_partition::file                                     | NO      | NO    |
| memory/partition/ha_partition::part_ids                                 | NO      | NO    |
| memory/partition/Partition_admin                                        | NO      | NO    |
| memory/partition/Partition_share                                        | NO      | NO    |
| memory/partition/partition_sort_buffer                                  | NO      | NO    |
| memory/performance_schema/accounts                                      | YES     | NO    |
| memory/performance_schema/cond_class                                    | YES     | NO    |
| memory/performance_schema/cond_instances                                | YES     | NO    |
| memory/performance_schema/events_stages_history                         | YES     | NO    |
| memory/performance_schema/events_stages_history_long                    | YES     | NO    |
| memory/performance_schema/events_stages_summary_by_account_by_event_name | YES   | NO    |
| memory/performance_schema/events_stages_summary_by_host_by_event_name   | YES     | NO    |
| memory/performance_schema/events_stages_summary_by_thread_by_event_name | YES     | NO    |
| memory/performance_schema/events_stages_summary_by_user_by_event_name   | YES     | NO    |
| memory/performance_schema/events_stages_summary_global_by_event_name    | YES     | NO    |
| memory/performance_schema/events_statements_current                     | YES     | NO    |
| memory/performance_schema/events_statements_current.sqltext             | YES     | NO    |
| memory/performance_schema/events_statements_current.tokens              | YES     | NO    |
| memory/performance_schema/events_statements_history                     | YES     | NO    |
| memory/performance_schema/events_statements_history.sqltext             | YES     | NO    |
| memory/performance_schema/events_statements_history.tokens              | YES     | NO    |
| memory/performance_schema/events_statements_history_long                | YES     | NO    |
| memory/performance_schema/events_statements_history_long.sqltext        | YES     | NO    |
| memory/performance_schema/events_statements_history_long.tokens         | YES     | NO    |
| memory/performance_schema/events_statements_summary_by_account_by_event_name | YES | NO    |
| memory/performance_schema/events_statements_summary_by_digest           | YES     | NO    |
| memory/performance_schema/events_statements_summary_by_digest.tokens    | YES     | NO    |
| memory/performance_schema/events_statements_summary_by_host_by_event_name | YES   | NO    |
```

```
| memory/performance_schema/events_statements_summary_by_program                 | YES     | NO    |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name    | YES     | NO    |
| memory/performance_schema/events_statements_summary_by_user_by_event_name      | YES     | NO    |
| memory/performance_schema/events_statements_summary_global_by_event_name       | YES     | NO    |
| memory/performance_schema/events_transactions_history                          | YES     | NO    |
| memory/performance_schema/events_transactions_history_long                     | YES     | NO    |
| memory/performance_schema/events_transactions_summary_by_account_by_event_name | YES     | NO    |
| memory/performance_schema/events_transactions_summary_by_host_by_event_name    | YES     | NO    |
| memory/performance_schema/events_transactions_summary_by_thread_by_event_name  | YES     | NO    |
| memory/performance_schema/events_transactions_summary_by_user_by_event_name    | YES     | NO    |
| memory/performance_schema/events_waits_history                                 | YES     | NO    |
| memory/performance_schema/events_waits_history_long                            | YES     | NO    |
| memory/performance_schema/events_waits_summary_by_account_by_event_name        | YES     | NO    |
| memory/performance_schema/events_waits_summary_by_host_by_event_name           | YES     | NO    |
| memory/performance_schema/events_waits_summary_by_thread_by_event_name         | YES     | NO    |
| memory/performance_schema/events_waits_summary_by_user_by_event_name           | YES     | NO    |
| memory/performance_schema/file_class                                           | YES     | NO    |
| memory/performance_schema/file_handle                                          | YES     | NO    |
| memory/performance_schema/file_instances                                       | YES     | NO    |
| memory/performance_schema/hosts                                                | YES     | NO    |
| memory/performance_schema/memory_class                                         | YES     | NO    |
| memory/performance_schema/memory_summary_by_account_by_event_name              | YES     | NO    |
| memory/performance_schema/memory_summary_by_host_by_event_name                 | YES     | NO    |
| memory/performance_schema/memory_summary_by_thread_by_event_name               | YES     | NO    |
| memory/performance_schema/memory_summary_by_user_by_event_name                 | YES     | NO    |
| memory/performance_schema/memory_summary_global_by_event_name                  | YES     | NO    |
| memory/performance_schema/metadata_locks                                       | YES     | NO    |
| memory/performance_schema/mutex_class                                          | YES     | NO    |
| memory/performance_schema/mutex_instances                                      | YES     | NO    |
| memory/performance_schema/prepared_statements_instances                        | YES     | NO    |
| memory/performance_schema/rwlock_class                                         | YES     | NO    |
| memory/performance_schema/rwlock_instances                                     | YES     | NO    |
| memory/performance_schema/scalable_buffer                                      | YES     | NO    |
| memory/performance_schema/session_connect_attrs                                | YES     | NO    |
| memory/performance_schema/setup_actors                                         | YES     | NO    |
| memory/performance_schema/setup_objects                                        | YES     | NO    |
| memory/performance_schema/socket_class                                         | YES     | NO    |
| memory/performance_schema/socket_instances                                     | YES     | NO    |
| memory/performance_schema/stage_class                                          | YES     | NO    |
| memory/performance_schema/statement_class                                      | YES     | NO    |
| memory/performance_schema/table_handles                                        | YES     | NO    |
| memory/performance_schema/table_io_waits_summary_by_index_usage                | YES     | NO    |
| memory/performance_schema/table_lock_waits_summary_by_table                    | YES     | NO    |
| memory/performance_schema/table_shares                                         | YES     | NO    |
| memory/performance_schema/threads                                              | YES     | NO    |
| memory/performance_schema/thread_class                                         | YES     | NO    |
| memory/performance_schema/users                                                | YES     | NO    |
| memory/sql/acl_cache                                                           | NO      | NO    |
| memory/sql/binlog_cache_mngr                                                   | NO      | NO    |
| memory/sql/binlog_pos                                                          | NO      | NO    |
| memory/sql/binlog_statement_buffer                                             | NO      | NO    |
| memory/sql/binlog_ver_1_event                                                  | NO      | NO    |
| memory/sql/bison_stack                                                         | NO      | NO    |
| memory/sql/Blob_mem_storage::storage                                           | NO      | NO    |
| memory/sql/DATE_TIME_FORMAT                                                    | NO      | NO    |
| memory/sql/dboptions_hash                                                      | NO      | NO    |
| memory/sql/DDL_LOG_MEMORY_ENTRY                                                | NO      | NO    |
| memory/sql/display_table_locks                                                 | NO      | NO    |
| memory/sql/errmsgs                                                             | NO      | NO    |
| memory/sql/Event_basic::mem_root                                               | NO      | NO    |
| memory/sql/Event_queue_element_for_exec::names                                 | NO      | NO    |
| memory/sql/Event_scheduler::scheduler_param                                    | NO      | NO    |
| memory/sql/Filesort_info::merge                                                | NO      | NO    |
| memory/sql/Filesort_info::record_pointers                                      | NO      | NO    |
| memory/sql/frm::string                                                         | NO      | NO    |
| memory/sql/gdl                                                                 | NO      | NO    |
| memory/sql/Gis_read_stream::err_msg                                            | NO      | NO    |
| memory/sql/global_system_variables                                             | NO      | NO    |
| memory/sql/handler::errmsgs                                                    | NO      | NO    |
```

```
| memory/sql/handlerton                                                | NO      | NO    |
| memory/sql/hash_index_key_buffer                                     | NO      | NO    |
| memory/sql/host_cache::hostname                                      | NO      | NO    |
| memory/sql/ignored_db                                                | NO      | NO    |
| memory/sql/JOIN_CACHE                                                | NO      | NO    |
| memory/sql/load_env_plugins                                          | NO      | NO    |
| memory/sql/Locked_tables_list::m_locked_tables_root                  | NO      | NO    |
| memory/sql/MDL_context::acquire_locks                                | NO      | NO    |
| memory/sql/MPVIO_EXT::auth_info                                      | NO      | NO    |
| memory/sql/MYSQL_BIN_LOG::basename                                   | NO      | NO    |
| memory/sql/MYSQL_BIN_LOG::index                                      | NO      | NO    |
| memory/sql/MYSQL_BIN_LOG::recover                                    | NO      | NO    |
| memory/sql/MYSQL_LOCK                                                | NO      | NO    |
| memory/sql/MYSQL_LOG::name                                           | NO      | NO    |
| memory/sql/mysql_plugin                                              | NO      | NO    |
| memory/sql/mysql_plugin_dl                                           | NO      | NO    |
| memory/sql/MYSQL_RELAY_LOG::basename                                 | NO      | NO    |
| memory/sql/MYSQL_RELAY_LOG::index                                    | NO      | NO    |
| memory/sql/my_str_malloc                                             | NO      | NO    |
| memory/sql/NAMED_ILINK::name                                         | NO      | NO    |
| memory/sql/native_functions                                          | NO      | NO    |
| memory/sql/plugin_bookmark                                           | NO      | NO    |
| memory/sql/plugin_int_mem_root                                       | NO      | NO    |
| memory/sql/plugin_mem_root                                           | NO      | NO    |
| memory/sql/Prepared_statement::main_mem_root                         | NO      | NO    |
| memory/sql/Prepared_statement_map                                    | NO      | NO    |
| memory/sql/PROFILE                                                   | NO      | NO    |
| memory/sql/Query_cache                                               | NO      | NO    |
| memory/sql/Queue::queue_item                                         | NO      | NO    |
| memory/sql/QUICK_RANGE_SELECT::alloc                                 | NO      | NO    |
| memory/sql/QUICK_RANGE_SELECT::mrr_buf_desc                          | NO      | NO    |
| memory/sql/Relay_log_info::group_relay_log_name                      | NO      | NO    |
| memory/sql/root                                                      | NO      | NO    |
| memory/sql/Row_data_memory::memory                                   | NO      | NO    |
| memory/sql/rpl_filter memory                                         | NO      | NO    |
| memory/sql/Rpl_info_file::buffer                                     | NO      | NO    |
| memory/sql/servers_cache                                             | NO      | NO    |
| memory/sql/SLAVE_INFO                                                | NO      | NO    |
| memory/sql/Sort_param::tmp_buffer                                    | NO      | NO    |
| memory/sql/sp_head::call_mem_root                                    | NO      | NO    |
| memory/sql/sp_head::execute_mem_root                                 | NO      | NO    |
| memory/sql/sp_head::main_mem_root                                    | NO      | NO    |
| memory/sql/sql_acl_mem                                               | NO      | NO    |
| memory/sql/sql_acl_memex                                             | NO      | NO    |
| memory/sql/String::value                                             | NO      | NO    |
| memory/sql/ST_SCHEMA_TABLE                                           | NO      | NO    |
| memory/sql/Sys_var_charptr::value                                    | NO      | NO    |
| memory/sql/TABLE                                                     | NO      | NO    |
| memory/sql/table_mapping::m_mem_root                                 | NO      | NO    |
| memory/sql/TABLE_RULE_ENT                                            | NO      | NO    |
| memory/sql/TABLE_SHARE::mem_root                                     | NO      | NO    |
| memory/sql/Table_triggers_list                                       | NO      | NO    |
| memory/sql/Table_trigger_dispatcher::m_mem_root                      | NO      | NO    |
| memory/sql/TC_LOG_MMAP::pages                                        | NO      | NO    |
| memory/sql/THD::db                                                   | NO      | NO    |
| memory/sql/THD::handler_tables_hash                                  | NO      | NO    |
| memory/sql/thd::main_mem_root                                        | NO      | NO    |
| memory/sql/THD::sp_cache                                             | NO      | NO    |
| memory/sql/THD::transactions::mem_root                               | NO      | NO    |
| memory/sql/THD::variables                                            | NO      | NO    |
| memory/sql/tz_storage                                                | NO      | NO    |
| memory/sql/udf_mem                                                   | NO      | NO    |
| memory/sql/Unique::merge_buffer                                      | NO      | NO    |
| memory/sql/Unique::sort_buffer                                       | NO      | NO    |
| memory/sql/user_conn                                                 | NO      | NO    |
| memory/sql/User_level_lock                                           | NO      | NO    |
| memory/sql/user_var_entry                                            | NO      | NO    |
| memory/sql/user_var_entry::value                                     | NO      | NO    |
| memory/sql/XID                                                       | NO      | NO    |
```

```
| stage/aria/Waiting for a resource                                    | NO      | NO    |
| stage/innodb/alter table (end)                                       | YES     | YES   |
| stage/innodb/alter table (insert)                                    | YES     | YES   |
| stage/innodb/alter table (log apply index)                           | YES     | YES   |
| stage/innodb/alter table (log apply table)                           | YES     | YES   |
| stage/innodb/alter table (merge sort)                                | YES     | YES   |
| stage/innodb/alter table (read PK and internal sort)                 | YES     | YES   |
| stage/innodb/buffer pool load                                        | YES     | YES   |
| stage/mysys/Waiting for table level lock                             | NO      | NO    |
| stage/sql/After apply log event                                      | NO      | NO    |
| stage/sql/After create                                               | NO      | NO    |
| stage/sql/After opening tables                                       | NO      | NO    |
| stage/sql/After table lock                                           | NO      | NO    |
| stage/sql/Allocating local table                                     | NO      | NO    |
| stage/sql/altering table                                             | NO      | NO    |
| stage/sql/Apply log event                                            | NO      | NO    |
| stage/sql/Changing master                                            | NO      | NO    |
| stage/sql/Checking master version                                    | NO      | NO    |
| stage/sql/checking permissions                                       | NO      | NO    |
| stage/sql/checking privileges on cached query                        | NO      | NO    |
| stage/sql/Checking query cache for query                             | NO      | NO    |
| stage/sql/closing tables                                             | NO      | NO    |
| stage/sql/Commit                                                     | NO      | NO    |
| stage/sql/Commit implicit                                            | NO      | NO    |
| stage/sql/Committing alter table to storage engine                   | NO      | NO    |
| stage/sql/Connecting to master                                       | NO      | NO    |
| stage/sql/Converting HEAP to Aria                                    | NO      | NO    |
| stage/sql/copy to tmp table                                          | YES     | YES   |
| stage/sql/Copying to group table                                     | NO      | NO    |
| stage/sql/Copying to tmp table                                       | NO      | NO    |
| stage/sql/Creating delayed handler                                   | NO      | NO    |
| stage/sql/Creating sort index                                        | NO      | NO    |
| stage/sql/creating table                                             | NO      | NO    |
| stage/sql/Creating tmp table                                         | NO      | NO    |
| stage/sql/Deleting from main table                                   | NO      | NO    |
| stage/sql/Deleting from reference tables                             | NO      | NO    |
| stage/sql/Discard_or_import_tablespace                               | NO      | NO    |
| stage/sql/Enabling keys                                              | NO      | NO    |
| stage/sql/End of update loop                                         | NO      | NO    |
| stage/sql/Executing                                                  | NO      | NO    |
| stage/sql/Execution of init_command                                  | NO      | NO    |
| stage/sql/Explaining                                                 | NO      | NO    |
| stage/sql/Filling schema table                                       | NO      | NO    |
| stage/sql/Finding key cache                                          | NO      | NO    |
| stage/sql/Finished reading one binlog; switching to next binlog      | NO      | NO    |
| stage/sql/Flushing relay log and master info repository.             | NO      | NO    |
| stage/sql/Flushing relay-log info file.                              | NO      | NO    |
| stage/sql/Freeing items                                              | NO      | NO    |
| stage/sql/Fulltext initialization                                    | NO      | NO    |
| stage/sql/Got handler lock                                           | NO      | NO    |
| stage/sql/Got old table                                              | NO      | NO    |
| stage/sql/init                                                       | NO      | NO    |
| stage/sql/init for update                                            | NO      | NO    |
| stage/sql/Insert                                                     | NO      | NO    |
| stage/sql/Invalidating query cache entries (table list)              | NO      | NO    |
| stage/sql/Invalidating query cache entries (table)                   | NO      | NO    |
| stage/sql/Killing slave                                              | NO      | NO    |
| stage/sql/Logging slow query                                         | NO      | NO    |
| stage/sql/Making temporary file (append) before replaying LOAD DATA INFILE | NO | NO |
| stage/sql/Making temporary file (create) before replaying LOAD DATA INFILE | NO | NO |
| stage/sql/Manage keys                                                | NO      | NO    |
| stage/sql/Master has sent all binlog to slave; waiting for more updates | NO   | NO    |
| stage/sql/Opening tables                                             | NO      | NO    |
| stage/sql/Optimizing                                                 | NO      | NO    |
| stage/sql/Preparing                                                  | NO      | NO    |
| stage/sql/preparing for alter table                                  | NO      | NO    |
| stage/sql/Processing binlog checkpoint notification                  | NO      | NO    |
| stage/sql/Processing requests                                        | NO      | NO    |
| stage/sql/Purging old relay logs                                     | NO      | NO    |
```

```
| stage/sql/Query end                                                          | NO    | NO    |
| stage/sql/Queueing master event to the relay log                             | NO    | NO    |
| stage/sql/Reading event from the relay log                                   | NO    | NO    |
| stage/sql/Reading semi-sync ACK from slave                                   | NO    | NO    |
| stage/sql/Recreating table                                                   | NO    | NO    |
| stage/sql/Registering slave on master                                        | NO    | NO    |
| stage/sql/Removing duplicates                                                | NO    | NO    |
| stage/sql/Removing tmp table                                                 | NO    | NO    |
| stage/sql/Rename                                                             | NO    | NO    |
| stage/sql/Rename result table                                                | NO    | NO    |
| stage/sql/Requesting binlog dump                                             | NO    | NO    |
| stage/sql/Reschedule                                                         | NO    | NO    |
| stage/sql/Reset for next command                                             | NO    | NO    |
| stage/sql/Rollback                                                           | NO    | NO    |
| stage/sql/Rollback_implicit                                                  | NO    | NO    |
| stage/sql/Searching rows for update                                          | NO    | NO    |
| stage/sql/Sending binlog event to slave                                      | NO    | NO    |
| stage/sql/Sending cached result to client                                    | NO    | NO    |
| stage/sql/Sending data                                                       | NO    | NO    |
| stage/sql/setup                                                              | NO    | NO    |
| stage/sql/Show explain                                                       | NO    | NO    |
| stage/sql/Slave has read all relay log; waiting for more updates             | NO    | NO    |
| stage/sql/Sorting                                                            | NO    | NO    |
| stage/sql/Sorting for group                                                  | NO    | NO    |
| stage/sql/Sorting for order                                                  | NO    | NO    |
| stage/sql/Sorting result                                                     | NO    | NO    |
| stage/sql/starting                                                           | NO    | NO    |
| stage/sql/Starting cleanup                                                   | NO    | NO    |
| stage/sql/Statistics                                                         | NO    | NO    |
| stage/sql/Stopping binlog background thread                                  | NO    | NO    |
| stage/sql/Storing result in query cache                                      | NO    | NO    |
| stage/sql/Storing row into queue                                             | NO    | NO    |
| stage/sql/System lock                                                        | NO    | NO    |
| stage/sql/table lock                                                         | NO    | NO    |
| stage/sql/Unlocking tables                                                   | NO    | NO    |
| stage/sql/Update                                                             | NO    | NO    |
| stage/sql/Updating                                                           | NO    | NO    |
| stage/sql/Updating main table                                                | NO    | NO    |
| stage/sql/Updating reference tables                                          | NO    | NO    |
| stage/sql/Upgrading lock                                                     | NO    | NO    |
| stage/sql/User lock                                                          | NO    | NO    |
| stage/sql/User sleep                                                         | NO    | NO    |
| stage/sql/Verifying table                                                    | NO    | NO    |
| stage/sql/Waiting for background binlog tasks                                | NO    | NO    |
| stage/sql/Waiting for backup lock                                            | NO    | NO    |
| stage/sql/Waiting for delay_list                                             | NO    | NO    |
| stage/sql/Waiting for event metadata lock                                    | NO    | NO    |
| stage/sql/Waiting for GTID to be written to binary log                       | NO    | NO    |
| stage/sql/Waiting for handler insert                                         | NO    | NO    |
| stage/sql/Waiting for handler lock                                           | NO    | NO    |
| stage/sql/Waiting for handler open                                           | NO    | NO    |
| stage/sql/Waiting for INSERT                                                 | NO    | NO    |
| stage/sql/Waiting for master to send event                                   | NO    | NO    |
| stage/sql/Waiting for master update                                          | NO    | NO    |
| stage/sql/Waiting for next activation                                        | NO    | NO    |
| stage/sql/Waiting for other master connection to process the same GTID       | NO    | NO    |
| stage/sql/Waiting for parallel replication deadlock handling to complete     | NO    | NO    |
| stage/sql/Waiting for prior transaction to commit                            | NO    | NO    |
| stage/sql/Waiting for prior transaction to start commit                      | NO    | NO    |
| stage/sql/Waiting for query cache lock                                       | NO    | NO    |
| stage/sql/Waiting for requests                                               | NO    | NO    |
| stage/sql/Waiting for room in worker thread event queue                      | NO    | NO    |
| stage/sql/Waiting for schema metadata lock                                   | NO    | NO    |
| stage/sql/Waiting for semi-sync ACK from slave                               | NO    | NO    |
| stage/sql/Waiting for semi-sync slave connection                             | NO    | NO    |
| stage/sql/Waiting for slave mutex on exit                                    | NO    | NO    |
| stage/sql/Waiting for slave thread to start                                  | NO    | NO    |
| stage/sql/Waiting for stored function metadata lock                          | NO    | NO    |
| stage/sql/Waiting for stored package body metadata lock                      | NO    | NO    |
```

stage/sql/Waiting for stored procedure metadata lock	NO	NO
stage/sql/Waiting for table flush	NO	NO
stage/sql/Waiting for table metadata lock	NO	NO
stage/sql/Waiting for the next event in relay log	NO	NO
stage/sql/Waiting for the scheduler to stop	NO	NO
stage/sql/Waiting for the slave SQL thread to advance position	NO	NO
stage/sql/Waiting for the slave SQL thread to free enough relay log space	NO	NO
stage/sql/Waiting for trigger metadata lock	NO	NO
stage/sql/Waiting for work from SQL thread	NO	NO
stage/sql/Waiting in MASTER_GTID_WAIT()	NO	NO
stage/sql/Waiting in MASTER_GTID_WAIT() (primary waiter)	NO	NO
stage/sql/Waiting on empty queue	NO	NO
stage/sql/Waiting to finalize termination	NO	NO
stage/sql/Waiting until MASTER_DELAY seconds after master executed event	NO	NO
stage/sql/Writing to binlog	NO	NO
statement/abstract/new_packet	YES	YES
statement/abstract/Query	YES	YES
statement/abstract/relay_log	YES	YES
statement/com/Binlog Dump	YES	YES
statement/com/Bulk_execute	YES	YES
statement/com/Change user	YES	YES
statement/com/Close stmt	YES	YES
statement/com/Com_multi	YES	YES
statement/com/Connect	YES	YES
statement/com/Connect Out	YES	YES
statement/com/Create DB	YES	YES
statement/com/Daemon	YES	YES
statement/com/Debug	YES	YES
statement/com/Delayed insert	YES	YES
statement/com/Drop DB	YES	YES
statement/com/Error	YES	YES
statement/com/Execute	YES	YES
statement/com/Fetch	YES	YES
statement/com/Field List	YES	YES
statement/com/Init DB	YES	YES
statement/com/Kill	YES	YES
statement/com/Long Data	YES	YES
statement/com/Ping	YES	YES
statement/com/Prepare	YES	YES
statement/com/Processlist	YES	YES
statement/com/Quit	YES	YES
statement/com/Refresh	YES	YES
statement/com/Register Slave	YES	YES
statement/com/Reset connection	YES	YES
statement/com/Reset stmt	YES	YES
statement/com/Set option	YES	YES
statement/com/Shutdown	YES	YES
statement/com/Slave_IO	YES	YES
statement/com/Slave_SQL	YES	YES
statement/com/Slave_worker	YES	YES
statement/com/Sleep	YES	YES
statement/com/Statistics	YES	YES
statement/com/Table Dump	YES	YES
statement/com/Time	YES	YES
statement/com/Unimpl get tid	YES	YES
statement/scheduler/event	YES	YES
statement/sp/agg_cfetch	YES	YES
statement/sp/cclose	YES	YES
statement/sp/cfetch	YES	YES
statement/sp/copen	YES	YES
statement/sp/cpop	YES	YES
statement/sp/cpush	YES	YES
statement/sp/cursor_copy_struct	YES	YES
statement/sp/error	YES	YES
statement/sp/freturn	YES	YES
statement/sp/hpop	YES	YES
statement/sp/hpush_jump	YES	YES
statement/sp/hreturn	YES	YES
statement/sp/jump	YES	YES

```
| statement/sp/jump_if_not                                                | YES     | YES   |
| statement/sp/preturn                                                    | YES     | YES   |
| statement/sp/set                                                        | YES     | YES   |
| statement/sp/set_case_expr                                              | YES     | YES   |
| statement/sp/set_trigger_field                                          | YES     | YES   |
| statement/sp/stmt                                                       | YES     | YES   |
| statement/sql/                                                          | YES     | YES   |
| statement/sql/alter_db                                                  | YES     | YES   |
| statement/sql/alter_db_upgrade                                          | YES     | YES   |
| statement/sql/alter_event                                               | YES     | YES   |
| statement/sql/alter_function                                            | YES     | YES   |
| statement/sql/alter_procedure                                           | YES     | YES   |
| statement/sql/alter_sequence                                            | YES     | YES   |
| statement/sql/alter_server                                              | YES     | YES   |
| statement/sql/alter_table                                               | YES     | YES   |
| statement/sql/alter_tablespace                                          | YES     | YES   |
| statement/sql/alter_user                                                | YES     | YES   |
| statement/sql/analyze                                                   | YES     | YES   |
| statement/sql/assign_to_keycache                                        | YES     | YES   |
| statement/sql/backup                                                    | YES     | YES   |
| statement/sql/backup_lock                                               | YES     | YES   |
| statement/sql/begin                                                     | YES     | YES   |
| statement/sql/binlog                                                    | YES     | YES   |
| statement/sql/call_procedure                                            | YES     | YES   |
| statement/sql/change_db                                                 | YES     | YES   |
| statement/sql/change_master                                             | YES     | YES   |
| statement/sql/check                                                     | YES     | YES   |
| statement/sql/checksum                                                  | YES     | YES   |
| statement/sql/commit                                                    | YES     | YES   |
| statement/sql/compound_sql                                              | YES     | YES   |
| statement/sql/create_db                                                 | YES     | YES   |
| statement/sql/create_event                                              | YES     | YES   |
| statement/sql/create_function                                           | YES     | YES   |
| statement/sql/create_index                                              | YES     | YES   |
| statement/sql/create_package                                            | YES     | YES   |
| statement/sql/create_package_body                                       | YES     | YES   |
| statement/sql/create_procedure                                          | YES     | YES   |
| statement/sql/create_role                                               | YES     | YES   |
| statement/sql/create_sequence                                           | YES     | YES   |
| statement/sql/create_server                                             | YES     | YES   |
| statement/sql/create_table                                              | YES     | YES   |
| statement/sql/create_trigger                                            | YES     | YES   |
| statement/sql/create_udf                                                | YES     | YES   |
| statement/sql/create_user                                               | YES     | YES   |
| statement/sql/create_view                                               | YES     | YES   |
| statement/sql/dealloc_sql                                               | YES     | YES   |
| statement/sql/delete                                                    | YES     | YES   |
| statement/sql/delete_multi                                              | YES     | YES   |
| statement/sql/do                                                        | YES     | YES   |
| statement/sql/drop_db                                                   | YES     | YES   |
| statement/sql/drop_event                                                | YES     | YES   |
| statement/sql/drop_function                                             | YES     | YES   |
| statement/sql/drop_index                                                | YES     | YES   |
| statement/sql/drop_package                                              | YES     | YES   |
| statement/sql/drop_package_body                                         | YES     | YES   |
| statement/sql/drop_procedure                                            | YES     | YES   |
| statement/sql/drop_role                                                 | YES     | YES   |
| statement/sql/drop_sequence                                             | YES     | YES   |
| statement/sql/drop_server                                               | YES     | YES   |
| statement/sql/drop_table                                                | YES     | YES   |
| statement/sql/drop_trigger                                              | YES     | YES   |
| statement/sql/drop_user                                                 | YES     | YES   |
| statement/sql/drop_view                                                 | YES     | YES   |
| statement/sql/empty_query                                               | YES     | YES   |
| statement/sql/error                                                     | YES     | YES   |
| statement/sql/execute_immediate                                         | YES     | YES   |
| statement/sql/execute_sql                                               | YES     | YES   |
| statement/sql/flush                                                     | YES     | YES   |
| statement/sql/get_diagnostics                                           | YES     | YES   |
```

```
| statement/sql/grant                                                | YES     | YES   |
| statement/sql/grant_role                                           | YES     | YES   |
| statement/sql/ha_close                                             | YES     | YES   |
| statement/sql/ha_open                                              | YES     | YES   |
| statement/sql/ha_read                                              | YES     | YES   |
| statement/sql/help                                                 | YES     | YES   |
| statement/sql/insert                                               | YES     | YES   |
| statement/sql/insert_select                                        | YES     | YES   |
| statement/sql/install_plugin                                       | YES     | YES   |
| statement/sql/kill                                                 | YES     | YES   |
| statement/sql/load                                                 | YES     | YES   |
| statement/sql/lock_tables                                          | YES     | YES   |
| statement/sql/optimize                                             | YES     | YES   |
| statement/sql/preload_keys                                         | YES     | YES   |
| statement/sql/prepare_sql                                          | YES     | YES   |
| statement/sql/purge                                                | YES     | YES   |
| statement/sql/purge_before_date                                    | YES     | YES   |
| statement/sql/release_savepoint                                    | YES     | YES   |
| statement/sql/rename_table                                         | YES     | YES   |
| statement/sql/rename_user                                          | YES     | YES   |
| statement/sql/repair                                               | YES     | YES   |
| statement/sql/replace                                              | YES     | YES   |
| statement/sql/replace_select                                       | YES     | YES   |
| statement/sql/reset                                                | YES     | YES   |
| statement/sql/resignal                                             | YES     | YES   |
| statement/sql/revoke                                               | YES     | YES   |
| statement/sql/revoke_all                                           | YES     | YES   |
| statement/sql/revoke_role                                          | YES     | YES   |
| statement/sql/rollback                                             | YES     | YES   |
| statement/sql/rollback_to_savepoint                                | YES     | YES   |
| statement/sql/savepoint                                            | YES     | YES   |
| statement/sql/select                                               | YES     | YES   |
| statement/sql/set_option                                           | YES     | YES   |
| statement/sql/show_authors                                         | YES     | YES   |
| statement/sql/show_binlogs                                         | YES     | YES   |
| statement/sql/show_binlog_events                                   | YES     | YES   |
| statement/sql/show_binlog_status                                   | YES     | YES   |
| statement/sql/show_charsets                                        | YES     | YES   |
| statement/sql/show_collations                                      | YES     | YES   |
| statement/sql/show_contributors                                    | YES     | YES   |
| statement/sql/show_create_db                                       | YES     | YES   |
| statement/sql/show_create_event                                    | YES     | YES   |
| statement/sql/show_create_func                                     | YES     | YES   |
| statement/sql/show_create_package                                  | YES     | YES   |
| statement/sql/show_create_package_body                             | YES     | YES   |
| statement/sql/show_create_proc                                     | YES     | YES   |
| statement/sql/show_create_table                                    | YES     | YES   |
| statement/sql/show_create_trigger                                  | YES     | YES   |
| statement/sql/show_create_user                                     | YES     | YES   |
| statement/sql/show_databases                                       | YES     | YES   |
| statement/sql/show_engine_logs                                     | YES     | YES   |
| statement/sql/show_engine_mutex                                    | YES     | YES   |
| statement/sql/show_engine_status                                   | YES     | YES   |
| statement/sql/show_errors                                          | YES     | YES   |
| statement/sql/show_events                                          | YES     | YES   |
| statement/sql/show_explain                                         | YES     | YES   |
| statement/sql/show_fields                                          | YES     | YES   |
| statement/sql/show_function_status                                 | YES     | YES   |
| statement/sql/show_generic                                         | YES     | YES   |
| statement/sql/show_grants                                          | YES     | YES   |
| statement/sql/show_keys                                            | YES     | YES   |
| statement/sql/show_open_tables                                     | YES     | YES   |
| statement/sql/show_package_body_status                             | YES     | YES   |
| statement/sql/show_package_status                                  | YES     | YES   |
| statement/sql/show_plugins                                         | YES     | YES   |
| statement/sql/show_privileges                                      | YES     | YES   |
| statement/sql/show_procedure_status                                | YES     | YES   |
| statement/sql/show_processlist                                     | YES     | YES   |
| statement/sql/show_profile                                         | YES     | YES   |
```

```
| statement/sql/show_profiles                                    | YES     | YES   |
| statement/sql/show_relaylog_events                             | YES     | YES   |
| statement/sql/show_slave_hosts                                 | YES     | YES   |
| statement/sql/show_slave_status                                | YES     | YES   |
| statement/sql/show_status                                      | YES     | YES   |
| statement/sql/show_storage_engines                             | YES     | YES   |
| statement/sql/show_tables                                      | YES     | YES   |
| statement/sql/show_table_status                                | YES     | YES   |
| statement/sql/show_triggers                                    | YES     | YES   |
| statement/sql/show_variables                                   | YES     | YES   |
| statement/sql/show_warnings                                    | YES     | YES   |
| statement/sql/shutdown                                         | YES     | YES   |
| statement/sql/signal                                           | YES     | YES   |
| statement/sql/start_all_slaves                                 | YES     | YES   |
| statement/sql/start_slave                                      | YES     | YES   |
| statement/sql/stop_all_slaves                                  | YES     | YES   |
| statement/sql/stop_slave                                       | YES     | YES   |
| statement/sql/truncate                                         | YES     | YES   |
| statement/sql/uninstall_plugin                                 | YES     | YES   |
| statement/sql/unlock_tables                                    | YES     | YES   |
| statement/sql/update                                           | YES     | YES   |
| statement/sql/update_multi                                     | YES     | YES   |
| statement/sql/xa_commit                                        | YES     | YES   |
| statement/sql/xa_end                                           | YES     | YES   |
| statement/sql/xa_prepare                                       | YES     | YES   |
| statement/sql/xa_recover                                       | YES     | YES   |
| statement/sql/xa_rollback                                      | YES     | YES   |
| statement/sql/xa_start                                         | YES     | YES   |
| transaction                                                    | NO      | NO    |
| wait/io/file/aria/control                                      | YES     | YES   |
| wait/io/file/aria/MAD                                          | YES     | YES   |
| wait/io/file/aria/MAI                                          | YES     | YES   |
| wait/io/file/aria/translog                                     | YES     | YES   |
| wait/io/file/csv/data                                          | YES     | YES   |
| wait/io/file/csv/metadata                                      | YES     | YES   |
| wait/io/file/csv/update                                        | YES     | YES   |
| wait/io/file/innodb/innodb_data_file                           | YES     | YES   |
| wait/io/file/innodb/innodb_log_file                            | YES     | YES   |
| wait/io/file/innodb/innodb_temp_file                           | YES     | YES   |
| wait/io/file/myisam/data_tmp                                   | YES     | YES   |
| wait/io/file/myisam/dfile                                      | YES     | YES   |
| wait/io/file/myisam/kfile                                      | YES     | YES   |
| wait/io/file/myisam/log                                        | YES     | YES   |
| wait/io/file/myisammrg/MRG                                     | YES     | YES   |
| wait/io/file/mysys/charset                                     | YES     | YES   |
| wait/io/file/mysys/cnf                                         | YES     | YES   |
| wait/io/file/partition/ha_partition::parfile                   | YES     | YES   |
| wait/io/file/sql/binlog                                        | YES     | YES   |
| wait/io/file/sql/binlog_cache                                  | YES     | YES   |
| wait/io/file/sql/binlog_index                                  | YES     | YES   |
| wait/io/file/sql/binlog_index_cache                            | YES     | YES   |
| wait/io/file/sql/binlog_state                                  | YES     | YES   |
| wait/io/file/sql/casetest                                      | YES     | YES   |
| wait/io/file/sql/dbopt                                         | YES     | YES   |
| wait/io/file/sql/des_key_file                                  | YES     | YES   |
| wait/io/file/sql/ERRMSG                                        | YES     | YES   |
| wait/io/file/sql/file_parser                                   | YES     | YES   |
| wait/io/file/sql/FRM                                           | YES     | YES   |
| wait/io/file/sql/global_ddl_log                                | YES     | YES   |
| wait/io/file/sql/init                                          | YES     | YES   |
| wait/io/file/sql/io_cache                                      | YES     | YES   |
| wait/io/file/sql/load                                          | YES     | YES   |
| wait/io/file/sql/LOAD_FILE                                     | YES     | YES   |
| wait/io/file/sql/log_event_data                                | YES     | YES   |
| wait/io/file/sql/log_event_info                                | YES     | YES   |
| wait/io/file/sql/map                                           | YES     | YES   |
| wait/io/file/sql/master_info                                   | YES     | YES   |
| wait/io/file/sql/misc                                          | YES     | YES   |
| wait/io/file/sql/partition_ddl_log                             | YES     | YES   |
```

```
| wait/io/file/sql/pid                                                | YES     | YES   |
| wait/io/file/sql/query_log                                          | YES     | YES   |
| wait/io/file/sql/relaylog                                           | YES     | YES   |
| wait/io/file/sql/relaylog_cache                                     | YES     | YES   |
| wait/io/file/sql/relaylog_index                                     | YES     | YES   |
| wait/io/file/sql/relaylog_index_cache                               | YES     | YES   |
| wait/io/file/sql/relay_log_info                                     | YES     | YES   |
| wait/io/file/sql/select_to_file                                     | YES     | YES   |
| wait/io/file/sql/send_file                                          | YES     | YES   |
| wait/io/file/sql/slow_log                                           | YES     | YES   |
| wait/io/file/sql/tclog                                              | YES     | YES   |
| wait/io/file/sql/trigger                                            | YES     | YES   |
| wait/io/file/sql/trigger_name                                       | YES     | YES   |
| wait/io/file/sql/wsrep_gra_log                                      | YES     | YES   |
| wait/io/socket/sql/client_connection                                | NO      | NO    |
| wait/io/socket/sql/server_tcpip_socket                              | NO      | NO    |
| wait/io/socket/sql/server_unix_socket                               | NO      | NO    |
| wait/io/table/sql/handler                                           | YES     | YES   |
| wait/lock/metadata/sql/mdl                                          | NO      | NO    |
| wait/lock/table/sql/handler                                         | YES     | YES   |
| wait/synch/cond/aria/BITMAP::bitmap_cond                            | NO      | NO    |
| wait/synch/cond/aria/COND_soft_sync                                 | NO      | NO    |
| wait/synch/cond/aria/SERVICE_THREAD_CONTROL::COND_control           | NO      | NO    |
| wait/synch/cond/aria/SHARE::key_del_cond                            | NO      | NO    |
| wait/synch/cond/aria/SORT_INFO::cond                                | NO      | NO    |
| wait/synch/cond/aria/TRANSLOG_BUFFER::prev_sent_to_disk_cond        | NO      | NO    |
| wait/synch/cond/aria/TRANSLOG_BUFFER::waiting_filling_buffer        | NO      | NO    |
| wait/synch/cond/aria/TRANSLOG_DESCRIPTOR::log_flush_cond            | NO      | NO    |
| wait/synch/cond/aria/TRANSLOG_DESCRIPTOR::new_goal_cond             | NO      | NO    |
| wait/synch/cond/innodb/commit_cond                                  | NO      | NO    |
| wait/synch/cond/myisam/MI_SORT_INFO::cond                           | NO      | NO    |
| wait/synch/cond/mysys/COND_alarm                                    | NO      | NO    |
| wait/synch/cond/mysys/COND_timer                                    | NO      | NO    |
| wait/synch/cond/mysys/IO_CACHE_SHARE::cond                          | NO      | NO    |
| wait/synch/cond/mysys/IO_CACHE_SHARE::cond_writer                   | NO      | NO    |
| wait/synch/cond/mysys/my_thread_var::suspend                        | NO      | NO    |
| wait/synch/cond/mysys/THR_COND_threads                              | NO      | NO    |
| wait/synch/cond/mysys/WT_RESOURCE::cond                             | NO      | NO    |
| wait/synch/cond/sql/Ack_receiver::cond                              | NO      | NO    |
| wait/synch/cond/sql/COND_binlog_send                                | NO      | NO    |
| wait/synch/cond/sql/COND_flush_thread_cache                         | NO      | NO    |
| wait/synch/cond/sql/COND_group_commit_orderer                       | NO      | NO    |
| wait/synch/cond/sql/COND_gtid_ignore_duplicates                     | NO      | NO    |
| wait/synch/cond/sql/COND_manager                                    | NO      | NO    |
| wait/synch/cond/sql/COND_parallel_entry                             | NO      | NO    |
| wait/synch/cond/sql/COND_prepare_ordered                            | NO      | NO    |
| wait/synch/cond/sql/COND_queue_state                                | NO      | NO    |
| wait/synch/cond/sql/COND_rpl_thread                                 | NO      | NO    |
| wait/synch/cond/sql/COND_rpl_thread_pool                            | NO      | NO    |
| wait/synch/cond/sql/COND_rpl_thread_queue                           | NO      | NO    |
| wait/synch/cond/sql/COND_rpl_thread_stop                            | NO      | NO    |
| wait/synch/cond/sql/COND_server_started                             | NO      | NO    |
| wait/synch/cond/sql/COND_slave_background                           | NO      | NO    |
| wait/synch/cond/sql/COND_start_thread                               | NO      | NO    |
| wait/synch/cond/sql/COND_thread_cache                               | NO      | NO    |
| wait/synch/cond/sql/COND_wait_gtid                                  | NO      | NO    |
| wait/synch/cond/sql/COND_wsrep_donor_monitor                        | NO      | NO    |
| wait/synch/cond/sql/COND_wsrep_gtid_wait_upto                       | NO      | NO    |
| wait/synch/cond/sql/COND_wsrep_joiner_monitor                       | NO      | NO    |
| wait/synch/cond/sql/COND_wsrep_ready                                | NO      | NO    |
| wait/synch/cond/sql/COND_wsrep_replaying                            | NO      | NO    |
| wait/synch/cond/sql/COND_wsrep_sst                                  | NO      | NO    |
| wait/synch/cond/sql/COND_wsrep_sst_init                             | NO      | NO    |
| wait/synch/cond/sql/COND_wsrep_wsrep_slave_threads                  | NO      | NO    |
| wait/synch/cond/sql/Delayed_insert::cond                            | NO      | NO    |
| wait/synch/cond/sql/Delayed_insert::cond_client                     | NO      | NO    |
| wait/synch/cond/sql/Event_scheduler::COND_state                     | NO      | NO    |
| wait/synch/cond/sql/Item_func_sleep::cond                           | NO      | NO    |
| wait/synch/cond/sql/Master_info::data_cond                          | NO      | NO    |
```

```
| wait/synch/cond/sql/Master_info::sleep_cond                                | NO      | NO    |
| wait/synch/cond/sql/Master_info::start_cond                                | NO      | NO    |
| wait/synch/cond/sql/Master_info::stop_cond                                 | NO      | NO    |
| wait/synch/cond/sql/MDL_context::COND_wait_status                          | NO      | NO    |
| wait/synch/cond/sql/MYSQL_BIN_LOG::COND_binlog_background_thread           | NO      | NO    |
| wait/synch/cond/sql/MYSQL_BIN_LOG::COND_binlog_background_thread_end       | NO      | NO    |
| wait/synch/cond/sql/MYSQL_BIN_LOG::COND_bin_log_updated                    | NO      | NO    |
| wait/synch/cond/sql/MYSQL_BIN_LOG::COND_queue_busy                         | NO      | NO    |
| wait/synch/cond/sql/MYSQL_BIN_LOG::COND_relay_log_updated                  | NO      | NO    |
| wait/synch/cond/sql/MYSQL_BIN_LOG::COND_xid_list                           | NO      | NO    |
| wait/synch/cond/sql/MYSQL_RELAY_LOG::COND_bin_log_updated                  | NO      | NO    |
| wait/synch/cond/sql/MYSQL_RELAY_LOG::COND_queue_busy                       | NO      | NO    |
| wait/synch/cond/sql/MYSQL_RELAY_LOG::COND_relay_log_updated                | NO      | NO    |
| wait/synch/cond/sql/PAGE::cond                                             | NO      | NO    |
| wait/synch/cond/sql/Query_cache::COND_cache_status_changed                 | NO      | NO    |
| wait/synch/cond/sql/Relay_log_info::data_cond                              | NO      | NO    |
| wait/synch/cond/sql/Relay_log_info::log_space_cond                         | NO      | NO    |
| wait/synch/cond/sql/Relay_log_info::start_cond                             | NO      | NO    |
| wait/synch/cond/sql/Relay_log_info::stop_cond                              | NO      | NO    |
| wait/synch/cond/sql/Rpl_group_info::sleep_cond                             | NO      | NO    |
| wait/synch/cond/sql/show_explain                                           | NO      | NO    |
| wait/synch/cond/sql/TABLE_SHARE::cond                                      | NO      | NO    |
| wait/synch/cond/sql/TABLE_SHARE::COND_rotation                             | NO      | NO    |
| wait/synch/cond/sql/TABLE_SHARE::tdc.COND_release                          | NO      | NO    |
| wait/synch/cond/sql/TC_LOG_MMAP::COND_active                               | NO      | NO    |
| wait/synch/cond/sql/TC_LOG_MMAP::COND_pool                                 | NO      | NO    |
| wait/synch/cond/sql/TC_LOG_MMAP::COND_queue_busy                           | NO      | NO    |
| wait/synch/cond/sql/THD::COND_wakeup_ready                                 | NO      | NO    |
| wait/synch/cond/sql/THD::COND_wsrep_thd                                    | NO      | NO    |
| wait/synch/cond/sql/User_level_lock::cond                                  | NO      | NO    |
| wait/synch/cond/sql/wait_for_commit::COND_wait_commit                      | NO      | NO    |
| wait/synch/cond/sql/wsrep_sst_thread                                       | NO      | NO    |
| wait/synch/mutex/aria/LOCK_soft_sync                                       | NO      | NO    |
| wait/synch/mutex/aria/LOCK_trn_list                                        | NO      | NO    |
| wait/synch/mutex/aria/PAGECACHE::cache_lock                                | NO      | NO    |
| wait/synch/mutex/aria/SERVICE_THREAD_CONTROL::LOCK_control                 | NO      | NO    |
| wait/synch/mutex/aria/SHARE::bitmap::bitmap_lock                           | NO      | NO    |
| wait/synch/mutex/aria/SHARE::close_lock                                    | NO      | NO    |
| wait/synch/mutex/aria/SHARE::intern_lock                                   | NO      | NO    |
| wait/synch/mutex/aria/SHARE::key_del_lock                                  | NO      | NO    |
| wait/synch/mutex/aria/SORT_INFO::mutex                                     | NO      | NO    |
| wait/synch/mutex/aria/THR_LOCK_maria                                       | NO      | NO    |
| wait/synch/mutex/aria/TRANSLOG_BUFFER::mutex                               | NO      | NO    |
| wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::dirty_buffer_mask_lock          | NO      | NO    |
| wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::file_header_lock                | NO      | NO    |
| wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::log_flush_lock                  | NO      | NO    |
| wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::purger_lock                     | NO      | NO    |
| wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::sent_to_disk_lock               | NO      | NO    |
| wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::unfinished_files_lock           | NO      | NO    |
| wait/synch/mutex/aria/TRN::state_lock                                      | NO      | NO    |
| wait/synch/mutex/csv/tina                                                  | NO      | NO    |
| wait/synch/mutex/csv/TINA_SHARE::mutex                                     | NO      | NO    |
| wait/synch/mutex/innodb/buf_dblwr_mutex                                    | NO      | NO    |
| wait/synch/mutex/innodb/buf_pool_mutex                                     | NO      | NO    |
| wait/synch/mutex/innodb/commit_cond_mutex                                  | NO      | NO    |
| wait/synch/mutex/innodb/dict_foreign_err_mutex                             | NO      | NO    |
| wait/synch/mutex/innodb/dict_sys_mutex                                     | NO      | NO    |
| wait/synch/mutex/innodb/fil_system_mutex                                   | NO      | NO    |
| wait/synch/mutex/innodb/flush_list_mutex                                   | NO      | NO    |
| wait/synch/mutex/innodb/fts_delete_mutex                                   | NO      | NO    |
| wait/synch/mutex/innodb/fts_doc_id_mutex                                   | NO      | NO    |
| wait/synch/mutex/innodb/ibuf_bitmap_mutex                                  | NO      | NO    |
| wait/synch/mutex/innodb/ibuf_mutex                                         | NO      | NO    |
| wait/synch/mutex/innodb/ibuf_pessimistic_insert_mutex                      | NO      | NO    |
| wait/synch/mutex/innodb/lock_mutex                                         | NO      | NO    |
| wait/synch/mutex/innodb/lock_wait_mutex                                    | NO      | NO    |
| wait/synch/mutex/innodb/log_flush_order_mutex                              | NO      | NO    |
| wait/synch/mutex/innodb/log_sys_mutex                                      | NO      | NO    |
| wait/synch/mutex/innodb/noredo_rseg_mutex                                  | NO      | NO    |
```

```
| wait/synch/mutex/innodb/page_zip_stat_per_index_mutex                | NO      | NO    |
| wait/synch/mutex/innodb/pending_checkpoint_mutex                     | NO      | NO    |
| wait/synch/mutex/innodb/purge_sys_pq_mutex                           | NO      | NO    |
| wait/synch/mutex/innodb/recalc_pool_mutex                            | NO      | NO    |
| wait/synch/mutex/innodb/recv_sys_mutex                               | NO      | NO    |
| wait/synch/mutex/innodb/redo_rseg_mutex                              | NO      | NO    |
| wait/synch/mutex/innodb/rtr_active_mutex                             | NO      | NO    |
| wait/synch/mutex/innodb/rtr_match_mutex                              | NO      | NO    |
| wait/synch/mutex/innodb/rtr_path_mutex                               | NO      | NO    |
| wait/synch/mutex/innodb/rw_lock_list_mutex                           | NO      | NO    |
| wait/synch/mutex/innodb/srv_innodb_monitor_mutex                     | NO      | NO    |
| wait/synch/mutex/innodb/srv_misc_tmpfile_mutex                       | NO      | NO    |
| wait/synch/mutex/innodb/srv_monitor_file_mutex                       | NO      | NO    |
| wait/synch/mutex/innodb/srv_threads_mutex                            | NO      | NO    |
| wait/synch/mutex/innodb/trx_mutex                                    | NO      | NO    |
| wait/synch/mutex/innodb/trx_pool_manager_mutex                       | NO      | NO    |
| wait/synch/mutex/innodb/trx_pool_mutex                               | NO      | NO    |
| wait/synch/mutex/innodb/trx_sys_mutex                                | NO      | NO    |
| wait/synch/mutex/myisam/MI_CHECK::print_msg                          | NO      | NO    |
| wait/synch/mutex/myisam/MI_SORT_INFO::mutex                          | NO      | NO    |
| wait/synch/mutex/myisam/MYISAM_SHARE::intern_lock                    | NO      | NO    |
| wait/synch/mutex/myisammrg/MYRG_INFO::mutex                          | NO      | NO    |
| wait/synch/mutex/mysys/BITMAP::mutex                                 | NO      | NO    |
| wait/synch/mutex/mysys/IO_CACHE::append_buffer_lock                  | NO      | NO    |
| wait/synch/mutex/mysys/IO_CACHE::SHARE_mutex                         | NO      | NO    |
| wait/synch/mutex/mysys/KEY_CACHE::cache_lock                         | NO      | NO    |
| wait/synch/mutex/mysys/LOCK_alarm                                    | NO      | NO    |
| wait/synch/mutex/mysys/LOCK_timer                                    | NO      | NO    |
| wait/synch/mutex/mysys/LOCK_uuid_generator                           | NO      | NO    |
| wait/synch/mutex/mysys/my_thread_var::mutex                          | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK::mutex                               | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK_charset                              | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK_heap                                 | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK_lock                                 | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK_malloc                               | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK_myisam                               | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK_myisam_mmap                          | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK_net                                  | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK_open                                 | NO      | NO    |
| wait/synch/mutex/mysys/THR_LOCK_threads                              | NO      | NO    |
| wait/synch/mutex/mysys/TMPDIR_mutex                                  | NO      | NO    |
| wait/synch/mutex/partition/Partition_share::auto_inc_mutex           | NO      | NO    |
| wait/synch/mutex/sql/Ack_receiver::mutex                             | NO      | NO    |
| wait/synch/mutex/sql/Cversion_lock                                   | NO      | NO    |
| wait/synch/mutex/sql/Delayed_insert::mutex                           | NO      | NO    |
| wait/synch/mutex/sql/Event_scheduler::LOCK_scheduler_state           | NO      | NO    |
| wait/synch/mutex/sql/gtid_waiting::LOCK_gtid_waiting                 | NO      | NO    |
| wait/synch/mutex/sql/hash_filo::lock                                 | NO      | NO    |
| wait/synch/mutex/sql/HA_DATA_PARTITION::LOCK_auto_inc                | NO      | NO    |
| wait/synch/mutex/sql/LOCK_active_mi                                  | NO      | NO    |
| wait/synch/mutex/sql/LOCK_after_binlog_sync                          | NO      | NO    |
| wait/synch/mutex/sql/LOCK_audit_mask                                 | NO      | NO    |
| wait/synch/mutex/sql/LOCK_binlog                                     | NO      | NO    |
| wait/synch/mutex/sql/LOCK_binlog_state                               | NO      | NO    |
| wait/synch/mutex/sql/LOCK_commit_ordered                             | NO      | NO    |
| wait/synch/mutex/sql/LOCK_crypt                                      | NO      | NO    |
| wait/synch/mutex/sql/LOCK_delayed_create                             | NO      | NO    |
| wait/synch/mutex/sql/LOCK_delayed_insert                             | NO      | NO    |
| wait/synch/mutex/sql/LOCK_delayed_status                             | NO      | NO    |
| wait/synch/mutex/sql/LOCK_des_key_file                               | NO      | NO    |
| wait/synch/mutex/sql/LOCK_error_log                                  | NO      | NO    |
| wait/synch/mutex/sql/LOCK_error_messages                             | NO      | NO    |
| wait/synch/mutex/sql/LOCK_event_queue                                | NO      | NO    |
| wait/synch/mutex/sql/LOCK_gdl                                        | NO      | NO    |
| wait/synch/mutex/sql/LOCK_global_index_stats                         | NO      | NO    |
| wait/synch/mutex/sql/LOCK_global_system_variables                    | NO      | NO    |
| wait/synch/mutex/sql/LOCK_global_table_stats                         | NO      | NO    |
| wait/synch/mutex/sql/LOCK_global_user_client_stats                   | NO      | NO    |
| wait/synch/mutex/sql/LOCK_item_func_sleep                            | NO      | NO    |
```

```
| wait/synch/mutex/sql/LOCK_load_client_plugin                                | NO      | NO    |
| wait/synch/mutex/sql/LOCK_manager                                           | NO      | NO    |
| wait/synch/mutex/sql/LOCK_parallel_entry                                    | NO      | NO    |
| wait/synch/mutex/sql/LOCK_plugin                                            | NO      | NO    |
| wait/synch/mutex/sql/LOCK_prepared_stmt_count                               | NO      | NO    |
| wait/synch/mutex/sql/LOCK_prepare_ordered                                   | NO      | NO    |
| wait/synch/mutex/sql/LOCK_rpl_semi_sync_master_enabled                      | NO      | NO    |
| wait/synch/mutex/sql/LOCK_rpl_status                                        | NO      | NO    |
| wait/synch/mutex/sql/LOCK_rpl_thread                                        | NO      | NO    |
| wait/synch/mutex/sql/LOCK_rpl_thread_pool                                   | NO      | NO    |
| wait/synch/mutex/sql/LOCK_server_started                                    | NO      | NO    |
| wait/synch/mutex/sql/LOCK_slave_background                                  | NO      | NO    |
| wait/synch/mutex/sql/LOCK_slave_state                                       | NO      | NO    |
| wait/synch/mutex/sql/LOCK_start_thread                                      | NO      | NO    |
| wait/synch/mutex/sql/LOCK_stats                                             | NO      | NO    |
| wait/synch/mutex/sql/LOCK_status                                            | NO      | NO    |
| wait/synch/mutex/sql/LOCK_system_variables_hash                             | NO      | NO    |
| wait/synch/mutex/sql/LOCK_table_cache                                       | NO      | NO    |
| wait/synch/mutex/sql/LOCK_thread_cache                                      | NO      | NO    |
| wait/synch/mutex/sql/LOCK_thread_id                                         | NO      | NO    |
| wait/synch/mutex/sql/LOCK_unused_shares                                     | NO      | NO    |
| wait/synch/mutex/sql/LOCK_user_conn                                         | NO      | NO    |
| wait/synch/mutex/sql/LOCK_uuid_short_generator                              | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_cluster_config                              | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_config_state                                | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_desync                                      | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_donor_monitor                               | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_group_commit                                | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_gtid_wait_upto                              | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_joiner_monitor                              | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_ready                                       | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_replaying                                   | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_slave_threads                               | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_SR_pool                                     | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_SR_store                                    | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_sst                                         | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_sst_init                                    | NO      | NO    |
| wait/synch/mutex/sql/LOG::LOCK_log                                          | NO      | NO    |
| wait/synch/mutex/sql/Master_info::data_lock                                 | NO      | NO    |
| wait/synch/mutex/sql/Master_info::run_lock                                  | NO      | NO    |
| wait/synch/mutex/sql/Master_info::sleep_lock                                | NO      | NO    |
| wait/synch/mutex/sql/Master_info::start_stop_lock                           | NO      | NO    |
| wait/synch/mutex/sql/MDL_wait::LOCK_wait_status                             | NO      | NO    |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_binlog_background_thread           | NO      | NO    |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_binlog_end_pos                     | NO      | NO    |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_index                              | NO      | NO    |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_xid_list                           | NO      | NO    |
| wait/synch/mutex/sql/MYSQL_RELAY_LOG::LOCK_binlog_end_pos                   | NO      | NO    |
| wait/synch/mutex/sql/MYSQL_RELAY_LOG::LOCK_index                            | NO      | NO    |
| wait/synch/mutex/sql/PAGE::lock                                             | NO      | NO    |
| wait/synch/mutex/sql/Query_cache::structure_guard_mutex                     | NO      | NO    |
| wait/synch/mutex/sql/Relay_log_info::data_lock                              | NO      | NO    |
| wait/synch/mutex/sql/Relay_log_info::log_space_lock                         | NO      | NO    |
| wait/synch/mutex/sql/Relay_log_info::run_lock                               | NO      | NO    |
| wait/synch/mutex/sql/Rpl_group_info::sleep_lock                             | NO      | NO    |
| wait/synch/mutex/sql/Slave_reporting_capability::err_lock                   | NO      | NO    |
| wait/synch/mutex/sql/TABLE_SHARE::LOCK_ha_data                              | NO      | NO    |
| wait/synch/mutex/sql/TABLE_SHARE::LOCK_rotation                             | NO      | NO    |
| wait/synch/mutex/sql/TABLE_SHARE::LOCK_share                                | NO      | NO    |
| wait/synch/mutex/sql/TABLE_SHARE::tdc.LOCK_table_share                      | NO      | NO    |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_active                               | NO      | NO    |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_pending_checkpoint                   | NO      | NO    |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_pool                                 | NO      | NO    |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_sync                                 | NO      | NO    |
| wait/synch/mutex/sql/THD::LOCK_thd_data                                     | NO      | NO    |
| wait/synch/mutex/sql/THD::LOCK_thd_kill                                     | NO      | NO    |
| wait/synch/mutex/sql/THD::LOCK_wakeup_ready                                 | NO      | NO    |
| wait/synch/mutex/sql/tz_LOCK                                                | NO      | NO    |
| wait/synch/mutex/sql/wait_for_commit::LOCK_wait_commit                      | NO      | NO    |
```

```
| wait/synch/mutex/sql/wsrep_sst_thread                                          | NO      | NO    |
| wait/synch/rwlock/aria/KEYINFO::root_lock                                      | NO      | NO    |
| wait/synch/rwlock/aria/SHARE::mmap_lock                                        | NO      | NO    |
| wait/synch/rwlock/aria/TRANSLOG_DESCRIPTOR::open_files_lock                    | NO      | NO    |
| wait/synch/rwlock/myisam/MYISAM_SHARE::key_root_lock                           | NO      | NO    |
| wait/synch/rwlock/myisam/MYISAM_SHARE::mmap_lock                               | NO      | NO    |
| wait/synch/rwlock/mysys/SAFE_HASH::mutex                                       | NO      | NO    |
| wait/synch/rwlock/proxy_proto/rwlock                                           | NO      | NO    |
| wait/synch/rwlock/sql/CRYPTO_dynlock_value::lock                               | NO      | NO    |
| wait/synch/rwlock/sql/LOCK_all_status_vars                                     | NO      | NO    |
| wait/synch/rwlock/sql/LOCK_dboptions                                           | NO      | NO    |
| wait/synch/rwlock/sql/LOCK_grant                                               | NO      | NO    |
| wait/synch/rwlock/sql/LOCK_SEQUENCE                                            | NO      | NO    |
| wait/synch/rwlock/sql/LOCK_ssl_refresh                                         | NO      | NO    |
| wait/synch/rwlock/sql/LOCK_system_variables_hash                               | NO      | NO    |
| wait/synch/rwlock/sql/LOCK_sys_init_connect                                    | NO      | NO    |
| wait/synch/rwlock/sql/LOCK_sys_init_slave                                      | NO      | NO    |
| wait/synch/rwlock/sql/LOGGER::LOCK_logger                                      | NO      | NO    |
| wait/synch/rwlock/sql/MDL_context::LOCK_waiting_for                            | NO      | NO    |
| wait/synch/rwlock/sql/MDL_lock::rwlock                                         | NO      | NO    |
| wait/synch/rwlock/sql/Query_cache_query::lock                                  | NO      | NO    |
| wait/synch/rwlock/sql/TABLE_SHARE::LOCK_stat_serial                            | NO      | NO    |
| wait/synch/rwlock/sql/THD_list::lock                                           | NO      | NO    |
| wait/synch/rwlock/sql/THR_LOCK_servers                                         | NO      | NO    |
| wait/synch/rwlock/sql/THR_LOCK_udf                                             | NO      | NO    |
| wait/synch/rwlock/sql/Vers_field_stats::lock                                   | NO      | NO    |
| wait/synch/sxlock/innodb/btr_search_latch                                      | NO      | NO    |
| wait/synch/sxlock/innodb/dict_operation_lock                                   | NO      | NO    |
| wait/synch/sxlock/innodb/fil_space_latch                                       | NO      | NO    |
| wait/synch/sxlock/innodb/fts_cache_init_rw_lock                                | NO      | NO    |
| wait/synch/sxlock/innodb/fts_cache_rw_lock                                     | NO      | NO    |
| wait/synch/sxlock/innodb/index_online_log                                      | NO      | NO    |
| wait/synch/sxlock/innodb/index_tree_rw_lock                                    | NO      | NO    |
| wait/synch/sxlock/innodb/trx_i_s_cache_lock                                    | NO      | NO    |
| wait/synch/sxlock/innodb/trx_purge_latch                                       | NO      | NO    |
+--------------------------------------------------------------------------------+---------+-------+
996 rows in set (0.005 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema setup_objects Table

Description

The `setup_objects` table determines whether objects are monitored by the performance schema or not. By default limited to 100 rows, this can be changed by setting the performance_schema_setup_objects_size system variable when the server starts.

It contains the following columns:

Column	Description
OBJECT_TYPE	Type of object to instrument, currently only . Currently, only TABLE', for base table.
OBJECT_SCHEMA	Schema containing the object, either the literal or % for any schema.
OBJECT_NAME	Name of the instrumented object, either the literal or % for any object.
ENABLED	Whether the object's events are instrumented or not. Can be disabled, in which case monitoring is not enabled for those objects.
TIMED	Whether the object's events are timed or not. Can be modified.

When the Performance Schema looks for matches in the `setup_objects`, there may be more than one row matching, with different `ENABLED` and `TIMED` values. It looks for the most specific matches first, that is, it will first

look for the specific database and table name combination, then the specific database, only then falling back to a wildcard for both.

Rows can be added or removed from the table, while for existing rows, only the `TIMED` and `ENABLED` columns can be updated. By default, all tables except those in the `performance_schema`, `information_schema` and `mysql` databases are instrumented.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema setup_timers Table

Description

The `setup_timers` table shows the currently selected event timers. Deprecated since 10.5, removed in 12.0.

It contains the following columns:

Column	Description
NAME	Type of instrument the timer is used for.
TIMER_NAME	Timer applying to the instrument type. Can be modified.

The `TIMER_NAME` value can be changed to choose a different timer, and can be any non-NULL value in the performance_timers.TIMER_NAME column.

If you modify the table, monitoring is immediately affected, and currently monitored events would use a combination of old and new timers, which is probably undesirable. It is best to reset the Performance Schema statistics if you make changes to this table.

Example

```
SELECT * FROM setup_timers;
+-----------+-------------+
| NAME      | TIMER_NAME  |
+-----------+-------------+
| idle      | MICROSECOND |
| wait      | CYCLE       |
| stage     | NANOSECOND  |
| statement | NANOSECOND  |
+-----------+-------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema socket_instances Table

The `socket_instances` table lists active server connections, with each record being a Unix socket file or TCP/IP connection.

The `socket_instances` table contains the following columns:

Column	Description
EVENT_NAME	NAME from the setup_instruments table, and the name of the wait/io/socket/* instrument that produced the event.
OBJECT_INSTANCE_BEGIN	Memory address of the object.
THREAD_ID	Thread identifier that the server assigns to each socket.
SOCKET_ID	The socket's internal file handle.

Column	Description
IP	Client IP address. Blank for Unix socket file, otherwise an IPv4 or IPv6 address. Together with the PORT identifies the connection.
PORT	TCP/IP port number, from 0 to 65535. Together with the IP identifies the connection.
STATE	Socket status, either IDLE if waiting to receive a request from a client, or ACTIVE

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema socket_summary_by_event_name Table

It aggregates timer and byte count statistics for all socket I/O operations by socket instrument.

Column	Description
EVENT_NAME	Socket instrument.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, including RECV, RECVFROM, and RECVMSG.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
SUM_NUMBER_OF_BYTES_READ	Bytes read by read operations.
COUNT_WRITE	Number of all write operations, including SEND, SENDTO, and SENDMSG.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
SUM_NUMBER_OF_BYTES_WRITE	Bytes written by write operations.
COUNT_MISC	Number of all miscellaneous operations not counted above, including CONNECT, LISTEN, ACCEPT, CLOSE, and SHUTDOWN.
SUM_TIMER_MISC	Total wait time of all miscellaneous operations that are timed.
MIN_TIMER_MISC	Minimum wait time of all miscellaneous operations that are timed.
AVG_TIMER_MISC	Average wait time of all miscellaneous operations that are timed.
MAX_TIMER_MISC	Maximum wait time of all miscellaneous operations that are timed.

You can TRUNCATE the table, which will reset all counters to zero.

Example

```
SELECT * FROM socket_summary_by_event_name\G
*************************** 1. row ***************************
               EVENT_NAME: wait/io/socket/sql/server_tcpip_socket
               COUNT_STAR: 0
           SUM_TIMER_WAIT: 0
           MIN_TIMER_WAIT: 0
           AVG_TIMER_WAIT: 0
           MAX_TIMER_WAIT: 0
               COUNT_READ: 0
           SUM_TIMER_READ: 0
           MIN_TIMER_READ: 0
           AVG_TIMER_READ: 0
           MAX_TIMER_READ: 0
 SUM_NUMBER_OF_BYTES_READ: 0
              COUNT_WRITE: 0
          SUM_TIMER_WRITE: 0
          MIN_TIMER_WRITE: 0
          AVG_TIMER_WRITE: 0
          MAX_TIMER_WRITE: 0
SUM_NUMBER_OF_BYTES_WRITE: 0
               COUNT_MISC: 0
           SUM_TIMER_MISC: 0
           MIN_TIMER_MISC: 0
           AVG_TIMER_MISC: 0
           MAX_TIMER_MISC: 0
*************************** 2. row ***************************
               EVENT_NAME: wait/io/socket/sql/server_unix_socket
               COUNT_STAR: 0
           SUM_TIMER_WAIT: 0
           MIN_TIMER_WAIT: 0
           AVG_TIMER_WAIT: 0
           MAX_TIMER_WAIT: 0
               COUNT_READ: 0
           SUM_TIMER_READ: 0
           MIN_TIMER_READ: 0
           AVG_TIMER_READ: 0
           MAX_TIMER_READ: 0
 SUM_NUMBER_OF_BYTES_READ: 0
              COUNT_WRITE: 0
          SUM_TIMER_WRITE: 0
          MIN_TIMER_WRITE: 0
          AVG_TIMER_WRITE: 0
          MAX_TIMER_WRITE: 0
SUM_NUMBER_OF_BYTES_WRITE: 0
               COUNT_MISC: 0
           SUM_TIMER_MISC: 0
           MIN_TIMER_MISC: 0
           AVG_TIMER_MISC: 0
           MAX_TIMER_MISC: 0
...
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema socket_summary_by_instance Table

It aggregates timer and byte count statistics for all socket I/O operations by socket instance.

Column	Description
EVENT_NAME	Socket instrument.
OBJECT_INSTANCE_BEGIN	Address in memory.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.

Column	Description
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, including RECV, RECVFROM, and RECVMSG.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
SUM_NUMBER_OF_BYTES_READ	Bytes read by read operations.
COUNT_WRITE	Number of all write operations, including SEND, SENDTO, and SENDMSG.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
SUM_NUMBER_OF_BYTES_WRITE	Bytes written by write operations.
COUNT_MISC	Number of all miscellaneous operations not counted above, including CONNECT, LISTEN, ACCEPT, CLOSE, and SHUTDOWN.
SUM_TIMER_MISC	Total wait time of all miscellaneous operations that are timed.
MIN_TIMER_MISC	Minimum wait time of all miscellaneous operations that are timed.
AVG_TIMER_MISC	Average wait time of all miscellaneous operations that are timed.
MAX_TIMER_MISC	Maximum wait time of all miscellaneous operations that are timed.

The corresponding row in the table is deleted when a connection terminates.

You can TRUNCATE the table, which will reset all counters to zero.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema status_by_account Table

> **Note:**
>
> The `status_by_account` table is available from MariaDB 10.5.2.

The `status_by_account` table contains status variable information by user/host account. The table does not collect statistics for `Com_xxx` variables.

The table contains the following columns:

Column	Description
USER	User for which the status variable is reported.
HOST	Host for which the status variable is reported.
VARIABLE_NAME	Status variable name.
VARIABLE_VALUE	Aggregated status variable value

If TRUNCATE TABLE is run, will aggregate the status from terminated sessions to user and host status, then reset the account status.

If FLUSH STATUS is run, session status from all active sessions are added to the global status variables, the status of all active sessions are reset, and values aggregated from disconnected sessions are reset.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema status_by_host Table

> **Note:**
>
> The `status_by_host` table is available from MariaDB 10.5.2.

The `status_by_host` table contains status variable information by host. The table does not collect statistics for `Com_xxx` variables.

The table contains the following columns:

Column	Description
HOST	Host for which the status variable is reported.
VARIABLE_NAME	Status variable name.
VARIABLE_VALUE	Aggregated status variable value

If TRUNCATE TABLE is run, will reset the aggregated host status from terminated sessions.

If FLUSH STATUS is run, session status from all active sessions are added to the global status variables, the status of all active sessions are reset, and values aggregated from disconnected sessions are reset.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema status_by_thread Table

> **Note:**
>
> The `session_status` table is available from MariaDB 10.5.2.

The `status_by_thread` table contains status variable information about active foreground threads. The table does not collect statistics for `Com_xxx` variables.

The table contains the following columns:

Column	Description
THREAD_ID	The thread identifier of the session in which the status variable is defined.
VARIABLE_NAME	Status variable name.
VARIABLE_VALUE	Aggregated status variable value.

If TRUNCATE TABLE is run, will aggregate the status for all threads to the global status and account status, then reset the thread status. If account statistics are not collected but host and user status are, the session status is added to host and user status.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema status_by_user Table

> **Note:**
>
> The `status_by_account` table is available from MariaDB 10.5.2.

The `status_by_account` table contains status variable information by user. The table does not collect statistics for `Com_xxx` variables.

The table contains the following columns:

Column	Description
USER	User for which the status variable is reported.
VARIABLE_NAME	Status variable name.
VARIABLE_VALUE	Aggregated status variable value

If TRUNCATE TABLE is run, will reset the aggregated user status from terminated sessions.

If FLUSH STATUS is run, session status from all active sessions are added to the global status variables, the status of all active sessions are reset, and values aggregated from disconnected sessions are reset.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema table_handles Table

> **Note:**
>
> The `table_handles` table is available from MariaDB 10.5.2.

The `table_handles` table contains table lock information. It uses the `wait/lock/table/sql/handler` instrument, which is enabled by default.

Information includes which table handles are open, which sessions are holding the locks, and how they are locked.

The table is read-only, and TRUNCATE TABLE cannot be performed on the table.

The maximum number of opened table objects is determined by the performance_schema_max_table_handles system variable.

The table contains the following columns:

Column	Description
OBJECT_TYPE	The table opened by a table handle.
OBJECT_SCHEMA	The schema that contains the object.
OBJECT_NAME	The name of the instrumented object.
OBJECT_INSTANCE_BEGIN	The table handle address in memory.
OWNER_THREAD_ID	The thread owning the table handle.
OWNER_EVENT_ID	The event which caused the table handle to be opened.
INTERNAL_LOCK	The table lock used at the SQL level.
EXTERNAL_LOCK	The table lock used at the storage engine level.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema table_io_waits_summary_by_index_usage Table

The `table_io_waits_summary_by_index_usage` table records table I/O waits by index.

Column	Description
OBJECT_TYPE	TABLE in the case of all indexes.
OBJECT_SCHEMA	Schema name.
OBJECT_NAME	Table name.
INDEX_NAME	Index name, or PRIMARY for the primary index, NULL for no index (inserts are counted in this case).
COUNT_STAR	Number of summarized events and the sum of the x_READ and x_WRITE columns.
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, and the sum of the equivalent x_FETCH columns.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
COUNT_WRITE	Number of all write operations, and the sum of the equivalent x_INSERT, x_UPDATE and x_DELETE columns.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
COUNT_FETCH	Number of all fetch operations.
SUM_TIMER_FETCH	Total wait time of all fetch operations that are timed.
MIN_TIMER_FETCH	Minimum wait time of all fetch operations that are timed.
AVG_TIMER_FETCH	Average wait time of all fetch operations that are timed.
MAX_TIMER_FETCH	Maximum wait time of all fetch operations that are timed.
COUNT_INSERT	Number of all insert operations.
SUM_TIMER_INSERT	Total wait time of all insert operations that are timed.
MIN_TIMER_INSERT	Minimum wait time of all insert operations that are timed.
AVG_TIMER_INSERT	Average wait time of all insert operations that are timed.
MAX_TIMER_INSERT	Maximum wait time of all insert operations that are timed.
COUNT_UPDATE	Number of all update operations.
SUM_TIMER_UPDATE	Total wait time of all update operations that are timed.
MIN_TIMER_UPDATE	Minimum wait time of all update operations that are timed.
AVG_TIMER_UPDATE	Average wait time of all update operations that are timed.
MAX_TIMER_UPDATE	Maximum wait time of all update operations that are timed.
COUNT_DELETE	Number of all delete operations.
SUM_TIMER_DELETE	Total wait time of all delete operations that are timed.
MIN_TIMER_DELETE	Minimum wait time of all delete operations that are timed.
AVG_TIMER_DELETE	Average wait time of all delete operations that are timed.
MAX_TIMER_DELETE	Maximum wait time of all delete operations that are timed.

You can TRUNCATE the table, which will reset all counters to zero. The table is also truncated if the table_io_waits_summary_by_table table is truncated.

If a table's index structure is changed, index statistics recorded in this table may also be reset.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema table_io_waits_summary_by_table Table

The `table_io_waits_summary_by_table` table records table I/O waits by table.

Column	Description
OBJECT_TYPE	Since this table records waits by table, always set to TABLE.
OBJECT_SCHEMA	Schema name.
OBJECT_NAME	Table name.
COUNT_STAR	Number of summarized events and the sum of the x_READ and x_WRITE columns.
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, and the sum of the equivalent x_FETCH columns.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
COUNT_WRITE	Number of all write operations, and the sum of the equivalent x_INSERT, x_UPDATE and x_DELETE columns.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
COUNT_FETCH	Number of all fetch operations.
SUM_TIMER_FETCH	Total wait time of all fetch operations that are timed.
MIN_TIMER_FETCH	Minimum wait time of all fetch operations that are timed.
AVG_TIMER_FETCH	Average wait time of all fetch operations that are timed.
MAX_TIMER_FETCH	Maximum wait time of all fetch operations that are timed.
COUNT_INSERT	Number of all insert operations.
SUM_TIMER_INSERT	Total wait time of all insert operations that are timed.
MIN_TIMER_INSERT	Minimum wait time of all insert operations that are timed.
AVG_TIMER_INSERT	Average wait time of all insert operations that are timed.
MAX_TIMER_INSERT	Maximum wait time of all insert operations that are timed.
COUNT_UPDATE	Number of all update operations.
SUM_TIMER_UPDATE	Total wait time of all update operations that are timed.
MIN_TIMER_UPDATE	Minimum wait time of all update operations that are timed.
AVG_TIMER_UPDATE	Average wait time of all update operations that are timed.

Column	Description
MAX_TIMER_UPDATE	Maximum wait time of all update operations that are timed.
COUNT_DELETE	Number of all delete operations.
SUM_TIMER_DELETE	Total wait time of all delete operations that are timed.
MIN_TIMER_DELETE	Minimum wait time of all delete operations that are timed.
AVG_TIMER_DELETE	Average wait time of all delete operations that are timed.
MAX_TIMER_DELETE	Maximum wait time of all delete operations that are timed.

You can TRUNCATE the table, which will reset all counters to zero. Truncating this table will also truncate the table_io_waits_summary_by_index_usage table.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema table_lock_waits_summary_by_table Table

The `table_lock_waits_summary_by_table` table records table lock waits by table.

Column	Description
OBJECT_TYPE	Since this table records waits by table, always set to TABLE.
OBJECT_SCHEMA	Schema name.
OBJECT_NAME	Table name.
COUNT_STAR	Number of summarized events and the sum of the x_READ and x_WRITE columns.
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, and the sum of the equivalent x_READ_NORMAL, x_READ_WITH_SHARED_LOCKS, x_READ_HIGH_PRIORITY and x_READ_NO_INSERT columns.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
COUNT_WRITE	Number of all write operations, and the sum of the equivalent x_WRITE_ALLOW_WRITE, x_WRITE_CONCURRENT_INSERT, x_WRITE_DELAYED, x_WRITE_LOW_PRIORITY and x_WRITE_NORMAL columns.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
COUNT_READ_NORMAL	Number of all internal read normal locks.
SUM_TIMER_READ_NORMAL	Total wait time of all internal read normal locks that are timed.
MIN_TIMER_READ_NORMAL	Minimum wait time of all internal read normal locks that are timed.
AVG_TIMER_READ_NORMAL	Average wait time of all internal read normal locks that are timed.

Column	Description
MAX_TIMER_READ_NORMAL	Maximum wait time of all internal read normal locks that are timed.
COUNT_READ_WITH_SHARED_LOCKS	Number of all internal read with shared locks.
SUM_TIMER_READ_WITH_SHARED_LOCKS	Total wait time of all internal read with shared locks that are timed.
MIN_TIMER_READ_WITH_SHARED_LOCKS	Minimum wait time of all internal read with shared locks that are timed.
AVG_TIMER_READ_WITH_SHARED_LOCKS	Average wait time of all internal read with shared locks that are timed.
MAX_TIMER_READ_WITH_SHARED_LOCKS	Maximum wait time of all internal read with shared locks that are timed.
COUNT_READ_HIGH_PRIORITY	Number of all internal read high priority locks.
SUM_TIMER_READ_HIGH_PRIORITY	Total wait time of all internal read high priority locks that are timed.
MIN_TIMER_READ_HIGH_PRIORITY	Minimum wait time of all internal read high priority locks that are timed.
AVG_TIMER_READ_HIGH_PRIORITY	Average wait time of all internal read high priority locks that are timed.
MAX_TIMER_READ_HIGH_PRIORITY	Maximum wait time of all internal read high priority locks that are timed.
COUNT_READ_NO_INSERT	Number of all internal read no insert locks.
SUM_TIMER_READ_NO_INSERT	Total wait time of all internal read no insert locks that are timed.
MIN_TIMER_READ_NO_INSERT	Minimum wait time of all internal read no insert locks that are timed.
AVG_TIMER_READ_NO_INSERT	Average wait time of all internal read no insert locks that are timed.
MAX_TIMER_READ_NO_INSERT	Maximum wait time of all internal read no insert locks that are timed.
COUNT_READ_EXTERNAL	Number of all external read locks.
SUM_TIMER_READ_EXTERNAL	Total wait time of all external read locks that are timed.
MIN_TIMER_READ_EXTERNAL	Minimum wait time of all external read locks that are timed.
AVG_TIMER_READ_EXTERNAL	Average wait time of all external read locks that are timed.
MAX_TIMER_READ_EXTERNAL	Maximum wait time of all external read locks that are timed.
COUNT_WRITE_ALLOW_WRITE	Number of all internal read normal locks.
SUM_TIMER_WRITE_ALLOW_WRITE	Total wait time of all internal write allow write locks that are timed.
MIN_TIMER_WRITE_ALLOW_WRITE	Minimum wait time of all internal write allow write locks that are timed.

Column	Description
AVG_TIMER_WRITE_ALLOW_WRITE	Average wait time of all internal write allow write locks that are timed.
MAX_TIMER_WRITE_ALLOW_WRITE	Maximum wait time of all internal write allow write locks that are timed.
COUNT_WRITE_CONCURRENT_INSERT	Number of all internal concurrent insert write locks.
SUM_TIMER_WRITE_CONCURRENT_INSERT	Total wait time of all internal concurrent insert write locks that are timed.
MIN_TIMER_WRITE_CONCURRENT_INSERT	Minimum wait time of all internal concurrent insert write locks that are timed.
AVG_TIMER_WRITE_CONCURRENT_INSERT	Average wait time of all internal concurrent insert write locks that are timed.
MAX_TIMER_WRITE_CONCURRENT_INSERT	Maximum wait time of all internal concurrent insert write locks that are timed.
COUNT_WRITE_DELAYED	Number of all internal write delayed locks.
SUM_TIMER_WRITE_DELAYED	Total wait time of all internal write delayed locks that are timed.
MIN_TIMER_WRITE_DELAYED	Minimum wait time of all internal write delayed locks that are timed.
AVG_TIMER_WRITE_DELAYED	Average wait time of all internal write delayed locks that are timed.
MAX_TIMER_WRITE_DELAYED	Maximum wait time of all internal write delayed locks that are timed.
COUNT_WRITE_LOW_PRIORITY	Number of all internal write low priority locks.
SUM_TIMER_WRITE_LOW_PRIORITY	Total wait time of all internal write low priority locks that are timed.
MIN_TIMER_WRITE_LOW_PRIORITY	Minimum wait time of all internal write low priority locks that are timed.
AVG_TIMER_WRITE_LOW_PRIORITY	Average wait time of all internal write low priority locks that are timed.
MAX_TIMER_WRITE_LOW_PRIORITY	Maximum wait time of all internal write low priority locks that are timed.
COUNT_WRITE_NORMAL	Number of all internal write normal locks.
SUM_TIMER_WRITE_NORMAL	Total wait time of all internal write normal locks that are timed.
MIN_TIMER_WRITE_NORMAL	Minimum wait time of all internal write normal locks that are timed.
AVG_TIMER_WRITE_NORMAL	Average wait time of all internal write normal locks that are timed.
MAX_TIMER_WRITE_NORMAL	Maximum wait time of all internal write normal locks that are timed.
COUNT_WRITE_EXTERNAL	Number of all external write locks.
SUM_TIMER_WRITE_EXTERNAL	Total wait time of all external write locks that are timed.

Column	Description
MIN_TIMER_WRITE_EXTERNAL	Minimum wait time of all external write locks that are timed.
AVG_TIMER_WRITE_EXTERNAL	Average wait time of all external write locks that are timed.
MAX_TIMER_WRITE_EXTERNAL	Maximum wait time of all external write locks that are timed.

You can TRUNCATE the table, which will reset all counters to zero.

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema threads Table

Each server thread is represented as a row in the `threads` table.

The `threads` table contains the following columns:

Column	Description
THREAD_ID	A unique thread identifier.
NAME	Name associated with the server's thread instrumentation code, for example thread/sql/main for the server's main() function, and thread/sql/one_connection for a user connection.
TYPE	FOREGROUND or BACKGROUND, depending on the thread type. User connection threads are FOREGROUND, internal server threads are BACKGROUND.
PROCESSLIST_ID	The PROCESSLIST.ID value for threads displayed in the INFORMATION_SCHEMA.PROCESSLIST table, or 0 for background threads. Also corresponds with the CONNECTION_ID() return value for the thread.
PROCESSLIST_USER	Foreground thread user, or NULL for a background thread.
PROCESSLIST_HOST	Foreground thread host, or NULL for a background thread.
PROCESSLIST_DB	Thread's default database, or NULL if none exists.
PROCESSLIST_COMMAND	Type of command executed by the thread. These correspond to the COM_xxx client/server protocol commands, and the Com_xxx status variables. See Thread Command Values.
PROCESSLIST_TIME	Time in seconds the thread has been in its current state.
PROCESSLIST_STATE	Action, event or state indicating what the thread is doing.
PROCESSLIST_INFO	Statement being executed by the thread, or NULL if a statement is not being executed. If a statement results in calling other statements, such as for a stored procedure, the innermost statement from the stored procedure is shown here.
PARENT_THREAD_ID	THREAD_ID of the parent thread, if any. Subthreads can for example be spawned as a result of INSERT DELAYED statements.

Column	Description
ROLE	Unused.
INSTRUMENTED	YES or NO for Whether the thread is instrumented or not. For foreground threads, the initial value is determined by whether there's a user/host match in the setup_actors table. Subthreads are again matched, while for background threads, this will be set to YES by default. To monitor events that the thread executes, INSTRUMENTED must be YES and the thread_instrumentation consumer in the setup_consumers table must also be YES.
HISTORY	YES or NO for Whether to log historical events for the thread. For foreground threads, the initial value is determined by whether there's a user/host match in the setup_actors table. Subthreads are again matched, while for background threads, this will be set to YES by default. To monitor events that the thread executes, INSTRUMENTED must be YES and the thread_instrumentation consumer in the setup_consumers table must also be YES. Added in MariaDB 10.5.
CONNECTION_TYPE	The protocol used to establish the connection. One of TCP/IP, SSL/TLS, Socket, Named Pipe, Shared Memory, or NULL for background threads. Added in MariaDB 10.5.
THREAD_OS_ID	The thread or task identifier as defined by the underlying operating system, if there is one. Added in MariaDB 10.5

Example

```
SELECT * FROM performance_schema.threads\G;
*************************** 1. row ***************************
          THREAD_ID: 1
               NAME: thread/sql/main
               TYPE: BACKGROUND
     PROCESSLIST_ID: NULL
   PROCESSLIST_USER: NULL
   PROCESSLIST_HOST: NULL
     PROCESSLIST_DB: NULL
PROCESSLIST_COMMAND: NULL
   PROCESSLIST_TIME: 215859
  PROCESSLIST_STATE: Table lock
   PROCESSLIST_INFO: INTERNAL DDL LOG RECOVER IN PROGRESS
   PARENT_THREAD_ID: NULL
               ROLE: NULL
       INSTRUMENTED: YES
...
*************************** 21. row ***************************
          THREAD_ID: 64
               NAME: thread/sql/one_connection
               TYPE: FOREGROUND
     PROCESSLIST_ID: 44
   PROCESSLIST_USER: root
   PROCESSLIST_HOST: localhost
     PROCESSLIST_DB: NULL
PROCESSLIST_COMMAND: Query
   PROCESSLIST_TIME: 0
  PROCESSLIST_STATE: Sending data
   PROCESSLIST_INFO: SELECT * FROM performance_schema.threads
   PARENT_THREAD_ID: NULL
               ROLE: NULL
       INSTRUMENTED: YES
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema users Table

Description

Each user that connects to the server is stored as a row in the `users` table, along with current and total connections.

The table size is determined at startup by the value of the performance_schema_users_size system variable. If this is set to `0`, user statistics will be disabled.

Column	Description
USER	The connection's client user name for the connection, or NULL if an internal thread.
CURRENT_CONNECTIONS	Current connections for the user.
TOTAL_CONNECTIONS	Total connections for the user.

Example

```
SELECT * FROM performance_schema.users;
+-----------------+---------------------+-------------------+
| USER            | CURRENT_CONNECTIONS | TOTAL_CONNECTIONS |
+-----------------+---------------------+-------------------+
| debian-sys-maint |                   0 |                35 |
| NULL            |                  20 |                23 |
| root            |                   1 |                 2 |
+-----------------+---------------------+-------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Performance Schema user_variables_by_thread Table

> **Note:**
>
> The `user_variables_by_thread` table is available from MariaDB 10.5.2.

The `user_variables_by_thread` table contains information about user-defined variables and the threads that defined them.

TRUNCATE TABLE cannot be performed on the table.

The table contains the following columns:

Column	Description
THREAD_ID	The thread identifier of the session in which the variable is defined.
VARIABLE_NAME	The variable name, without the leading @ character.
VARIABLE_VALUE	The variable value

Example

```
SET @var = 0;

SELECT * FROM user_variables_by_thread;
+-----------+---------------+----------------+
| THREAD_ID | VARIABLE_NAME | VARIABLE_VALUE |
+-----------+---------------+----------------+
|        11 | var           | 0              |
+-----------+---------------+----------------+
```

See Also

- User-defined variables
- Information Schema USER_VARIABLES Table

This page is licensed: CC BY-SA / Gnu FDL

Sys Schema

See Sys Schema sys_config Table.

The sys_config table holds persistent configuration options for the Sys Schema, stored using the Aria storage engine to maintain settings across restarts.

See Sys Schema Stored Functions.

Explore sys_schema stored functions in MariaDB Server. These functions simplify querying performance and configuration data, offering a user-friendly interface for database diagnostics.

See Sys Schema Stored Procedures.

Explore sys_schema stored procedures in MariaDB Server. These procedures simplify complex administrative and diagnostic tasks, offering streamlined access to performance and configuration insights.

See Sys Schema Views.

Explore sys_schema views in MariaDB Server. These views offer simplified, aggregated insights into server performance, I/O, and memory usage, making diagnostics and monitoring easier.

This page is licensed: CC BY-SA / Gnu FDL

Sys Schema sys_config Table

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

The *sys.sys_config* table holds configuration options for the Sys Schema.

This is a persistent table (using the Aria storage engine), with the configuration persisting across upgrades (new options are added with INSERT IGNORE.

The table also has two related triggers, which maintain the user that INSERTs or UPDATEs the configuration - sys_config_insert_set_user and sys_config_update_set_user respectively.

Its structure is as follows:

```
+----------+--------------+------+-----+-------------------+-----------------------------+
| Field    | Type         | Null | Key | Default           | Extra                       |
+----------+--------------+------+-----+-------------------+-----------------------------+
| variable | varchar(128) | NO   | PRI | NULL              |                             |
| value    | varchar(128) | YES  |     | NULL              |                             |
| set_time | timestamp    | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| set_by   | varchar(128) | YES  |     | NULL              |                             |
+----------+--------------+------+-----+-------------------+-----------------------------+
```

> **Warning:**
>
> Note, when functions check for configuration options, they first check whether a similar named user variable exists with a value, and if this is not set then pull the configuration option from this table in to that named user variable. This is done for performance reasons (to not continually `SELECT` from the table), however this comes with the side effect that once inited, the values last with the session, somewhat like how session variables are inited from global variables. If the values within this table are changed, they will not take effect until the user logs in again.

Options Included

Variable	Default Value	Description
statement_truncate_len	64	Sets the size to truncate statements to, for the format_statement function.
statement_performance_analyzer.limit	100	The maximum number of rows to include for the views that does not have a built-in limit (e.g. the 95th percentile view). If not set the limit is 100.
statement_performance_analyzer.view	NULL	Used together with the 'custom' view. If the value contains a space, it is considered a query, otherwise it must be an existing view querying the performance_schema.events_statements_summary_by_digest table.
diagnostics.allow_is_tables	OFF	Specifies whether it is allowed to do table scan queries on information_schema.TABLES for the diagnostics procedure.
diagnostics.include_raw	OFF	Set to 'ON' to include the raw data (e.g. the original output of "SELECT * FROM sys.metrics") for the diagnostics procedure.
ps_thread_trx_info.max_length	65535	Sets the maximum output length for JSON object output by the ps_thread_trx_info() function.

Notes

Some early versions of *sys_config* were stored in InnoDB format.

This page is licensed: CC BY-SA / Gnu FDL

Sys Schema Stored Functions

- extract_schema_from_file_name
- extract_table_from_file_name
- format_bytes
- format_path
- format_statement
- format_time
- list_add
- list_drop
- ps_is_account_enabled
- ps_is_consumer_enabled
- ps_is_instrument_default_enabled
- ps_is_instrument_default_timed
- ps_is_thread_instrumented
- ps_thread_account
- ps_thread_id
- ps_thread_stack
- ps_thread_trx_info
- quote_identifier
- sys_get_config
- version_major
- version_minor
- version_patch

This page is licensed: CC BY-SA / Gnu FDL

extract_schema_from_file_name

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.extract_schema_from_file_name(path)
```

Description

`extract_schema_from_file_name` is a stored function available with the Sys Schema.

Given a file path, it returns the schema (database) name. The file name is assumed to be within the schema directory, and therefore the function will not return the expected result with partitions, or when tables are defined using the DATA_DIRECTORY table option.

The function does not examine anything on disk. The return value, a VARCHAR(64), is determined solely from the provided path.

Examples

```
SELECT sys.extract_schema_from_file_name('/usr/local/mysql/data/db/t1.ibd');
+---------------------------------------------------------------------+
| sys.extract_schema_from_file_name('/usr/local/mysql/data/db/t1.ibd') |
+---------------------------------------------------------------------+
| db                                                                  |
+---------------------------------------------------------------------+
```

See also

- extract_table_from_file_name()

This page is licensed: CC BY-SA / Gnu FDL

extract_table_from_file_name

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.extract_table_from_file_name(path)
```

Description

`extract_table_from_file_name` is a stored function available with the Sys Schema.

Given a file path, it returns the table name.

The function does not examine anything on disk. The return value, a VARCHAR(64), is determined solely from the provided path.

Examples

```
SELECT sys.extract_table_from_file_name('/usr/local/mysql/data/db/t1.ibd');
+---------------------------------------------------------------------+
| sys.extract_table_from_file_name('/usr/local/mysql/data/db/t1.ibd') |
+---------------------------------------------------------------------+
| t1                                                                  |
+---------------------------------------------------------------------+
```

See also

- extract_schema_from_file_name()

This page is licensed: CC BY-SA / Gnu FDL

format_bytes

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.format_bytes(double)
```

Description

`format_bytes` is a stored function available with the Sys Schema.

Given a byte count, returns a string consisting of a value and the units in a human-readable format. The units will be in bytes, KiB (kibibytes), MiB (mebibytes), GiB (gibibytes), TiB (tebibytes), or PiB (pebibytes).

The binary prefixes (kibi, mebi, gibi, tebi and pebi) were created in December 1998 by the International Electrotechnical Commission to avoid possible ambiguity, as the widely-used prefixes kilo, mega, giga, tera and peta can be used to refer to both the power-of-10 decimal system multipliers and the power-of-two binary system multipliers.

From MariaDB 11.8, the built-in FORMAT_BYTES function can instead be used. The functions are similar, except that `FORMAT_BYTES` also displays exbibytes.

Examples

```
SELECT sys.format_bytes(1000),sys.format_bytes(1024);
+-----------------------+-----------------------+
| sys.format_bytes(1000) | sys.format_bytes(1024) |
+-----------------------+-----------------------+
| 1000 bytes             | 1.00 KiB              |
+-----------------------+-----------------------+

SELECT sys.format_bytes(1000000),sys.format_bytes(1048576);
+--------------------------+--------------------------+
| sys.format_bytes(1000000) | sys.format_bytes(1048576) |
+--------------------------+--------------------------+
| 976.56 KiB               | 1.00 MiB                 |
+--------------------------+--------------------------+

SELECT sys.format_bytes(1000000000),sys.format_bytes(1073741874);
+-----------------------------+-----------------------------+
| sys.format_bytes(1000000000) | sys.format_bytes(1073741874) |
+-----------------------------+-----------------------------+
| 953.67 MiB                  | 1.00 GiB                    |
+-----------------------------+-----------------------------+

SELECT sys.format_bytes(1000000000000),sys.format_bytes(1099511627776);
+--------------------------------+--------------------------------+
| sys.format_bytes(1000000000000) | sys.format_bytes(1099511627776) |
+--------------------------------+--------------------------------+
| 931.32 GiB                     | 1.00 TiB                       |
+--------------------------------+--------------------------------+

SELECT sys.format_bytes(1000000000000000),sys.format_bytes(1125899906842624);
+-----------------------------------+-----------------------------------+
| sys.format_bytes(1000000000000000) | sys.format_bytes(1125899906842624) |
+-----------------------------------+-----------------------------------+
| 909.49 TiB                        | 1.00 PiB                          |
+-----------------------------------+-----------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

format_path

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.format_path(path)
```

Description

`format_path` is a stored function available with the Sys Schema that, given a path, returns a modified path after replacing subpaths matching the values of various system variables with the variable name.

The system variables that are matched are, in order:

- datadir
- tmpdir
- slave_load_tmpdir
- innodb_data_home_dir
- innodb_log_group_home_dir
- innodb_undo_directory

- basedir

Examples

```
SELECT @@tmpdir;
+-----------------------------------+
| @@tmpdir                          |
+-----------------------------------+
| /home/ian/sandboxes/msb_10_8_2/tmp |
+-----------------------------------+

SELECT sys.format_path('/home/ian/sandboxes/msb_10_8_2/tmp/testdb.ibd');
+-------------------------------------------------------------------+
| sys.format_path('/home/ian/sandboxes/msb_10_8_2/tmp/testdb.ibd') |
+-------------------------------------------------------------------+
| @@tmpdir/testdb.ibd                                               |
+-------------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

format_statement

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.format_statement(statement)
```

Description

Returns a reduced length string. The length is specified by the statement_truncate_len configuration option (default 64), and the removed part of the string (if any) is replaced with an ellipsis (three dots).

The function is intended for use in formatting lengthy SQL statements to a fixed length.

Examples

Default truncation length 64:

```
SELECT sys.format_statement(
  'SELECT field1, field2, field3, field4, field5, field6 FROM table1'
  ) AS formatted_statement;
+-------------------------------------------------------------------+
| formatted_statement                                               |
+-------------------------------------------------------------------+
| SELECT field1, field2, field3, ... d4, field5, field6 FROM table1 |
+-------------------------------------------------------------------+
```

Reducing the truncation length to 48:

```
SET @sys.statement_truncate_len = 48;

SELECT sys.format_statement(
  'SELECT field1, field2, field3, field4, field5, field6 FROM table1'
  ) AS formatted_statement;
+-------------------------------------------------+
| formatted_statement                             |
+-------------------------------------------------+
| SELECT field1, field2, ... d5, field6 FROM table1 |
+-------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

format_time

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.format_time(picoseconds)
```

Description

`format_time` is a stored function available with the Sys Schema. Given a time in picoseconds, returns a human-readable time value and unit indicator. Unit can be:

- ps - picoseconds
- ns - nanoseconds
- us - microseconds
- ms - milliseconds
- s - seconds
- m - minutes
- h - hours
- d - days
- w - weeks

This function is very similar to the FORMAT_PICO_TIME function introduced in MariaDB 11.0.2, but with the following differences:

- Represents minutes as `m` rather than `min`.
- Represent weeks.

Examples

```
SELECT
    sys.format_time(43) AS ps,
    sys.format_time(4321) AS ns,
    sys.format_time(43211234) AS us,
    sys.format_time(432112344321) AS ms,
    sys.format_time(43211234432123) AS s,
    sys.format_time(432112344321234) AS m,
    sys.format_time(4321123443212345) AS h,
    sys.format_time(43211234432123545) AS d,
    sys.format_time(432112344321234543) AS w;
+-------+---------+----------+-----------+---------+--------+--------+--------+---------+
| ps    | ns      | us       | ms        | s       | m      | h      | d      | w       |
+-------+---------+----------+-----------+---------+--------+--------+--------+---------+
| 43 ps | 4.32 ns | 43.21 us | 432.11 ms | 43.21 s | 7.20 m | 1.20 h | 5.00 d | 71.45 w |
+-------+---------+----------+-----------+---------+--------+--------+--------+---------+
```

This page is licensed: CC BY-SA / Gnu FDL

list_add

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.list_add(list,value)
```

Description

`list_add` is a stored function available with the Sys Schema.

It takes a *list* to be modified and a *value* to be added to the list, returning the resulting value. This can be used, for example, to add a value to a system variable taking a comma-delimited list of options, such as sql_mode.

The related function list_drop can be used to drop a value from a list.

Examples

```
SELECT @@sql_mode;
+-----------------------------------------------------------------------+
| @@sql_mode                                                            |
+-----------------------------------------------------------------------+
| STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,
NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------+

SET @@sql_mode = sys.list_add(@@sql_mode, 'NO_ZERO_DATE');

SELECT @@sql_mode;
+-----------------------------------------------------------------------+
| @@sql_mode                                                            |
+-----------------------------------------------------------------------+
| STRICT_TRANS_TABLES,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,
NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------+
```

See Also

- list_drop

This page is licensed: CC BY-SA / Gnu FDL

list_drop

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.list_drop(list,value)
```

Description

`list_drop` is a stored function available with the Sys Schema.

It takes a *list* to be modified and a *value* to be dropped from the list, returning the resulting value. This can be used, for example, to remove a value from a system variable taking a comma-delimited list of options, such as sql_mode.

The related function list_add can be used to add a value to a list.

Examples

```
SELECT @@sql_mode;
+-------------------------------------------------------------------------+
| @@sql_mode                                                              |
+-------------------------------------------------------------------------+
| STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,
NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-------------------------------------------------------------------------+

SET @@sql_mode = sys.list_drop(@@sql_mode, 'NO_ENGINE_SUBSTITUTION');

SELECT @@sql_mode;
+-------------------------------------------------------------------------+
| @@sql_mode                                                              |
+-------------------------------------------------------------------------+
| STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,
NO_AUTO_CREATE_USER |
+-------------------------------------------------------------------------+
```

See Also

- list_add

This page is licensed: CC BY-SA / Gnu FDL

ps_is_account_enabled

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.ps_is_account_enabled(host,user)
```

Description

`ps_is_account_enabled` is a stored function available with the Sys Schema.

It takes *host* and *user* arguments, and returns an ENUM('YES','NO') depending on whether Performance Schema instrumentation for the given account is enabled.

Examples

```
SELECT sys.ps_is_account_enabled('localhost', 'root');
+-----------------------------------------------+
| sys.ps_is_account_enabled('localhost', 'root') |
+-----------------------------------------------+
| YES                                           |
+-----------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ps_is_consumer_enabled

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.ps_is_consumer_enabled(consumer)
```

Description

`ps_is_consumer_enabled` is a stored function available with the Sys Schema.

It returns an ENUM('YES','NO') depending on whether Performance Schema instrumentation for the given consumer is enabled, and NULL if not given a valid consumer name.

Examples

```
SELECT sys.ps_is_consumer_enabled('global_instrumentation');
+-----------------------------------------------------+
| sys.ps_is_consumer_enabled('global_instrumentation') |
+-----------------------------------------------------+
| YES                                                  |
+-----------------------------------------------------+

SELECT sys.ps_is_consumer_enabled('events_stages_current');
+-----------------------------------------------------+
| sys.ps_is_consumer_enabled('events_stages_current') |
+-----------------------------------------------------+
| NO                                                  |
+-----------------------------------------------------+

SELECT sys.ps_is_consumer_enabled('nonexistent_consumer');
+-----------------------------------------------------+
| sys.ps_is_consumer_enabled('nonexistent_consumer') |
+-----------------------------------------------------+
| NULL                                                |
+-----------------------------------------------------+
```

See Also

- Performance Schema setup_consumers Table

This page is licensed: CC BY-SA / Gnu FDL

ps_is_instrument_default_enabled

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.ps_is_instrument_default_enabled(instrument)
```

Description

`ps_is_instrument_default_enabled` is a stored function available with the Sys Schema.

It returns `YES` if the given Performance Schema *instrument* is enabled by default, and `NO` if it is not, does not exist, or is a NULL value.

Examples

```
SELECT sys.ps_is_instrument_default_enabled('statement/sql/select');
+-------------------------------------------------------------+
| sys.ps_is_instrument_default_enabled('statement/sql/select') |
+-------------------------------------------------------------+
| YES                                                          |
+-------------------------------------------------------------+

SELECT sys.ps_is_instrument_default_enabled('memory/sql/udf_mem');
+-----------------------------------------------------------+
| sys.ps_is_instrument_default_enabled('memory/sql/udf_mem') |
+-----------------------------------------------------------+
| NO                                                        |
+-----------------------------------------------------------+

SELECT sys.ps_is_instrument_default_enabled('memory/sql/nonexistent');
+--------------------------------------------------------------+
| sys.ps_is_instrument_default_enabled('memory/sql/nonexistent') |
+--------------------------------------------------------------+
| NO                                                           |
+--------------------------------------------------------------+

SELECT sys.ps_is_instrument_default_enabled(NULL);
+-----------------------------------------+
| sys.ps_is_instrument_default_enabled(NULL) |
+-----------------------------------------+
| NO                                      |
+-----------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ps_is_instrument_default_timed

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.ps_is_instrument_default_timed(instrument)
```

Description

`ps_is_instrument_default_timed` is a stored function available with the Sys Schema.

It returns `YES` if the given Performance Schema *instrument* is timed by default, and `NO` if it is not, does not exist, or is a `NULL` value.

Examples

```
SELECT sys.ps_is_instrument_default_timed('statement/sql/select');
+-----------------------------------------------------------+
| sys.ps_is_instrument_default_timed('statement/sql/select') |
+-----------------------------------------------------------+
| YES                                                       |
+-----------------------------------------------------------+

SELECT sys.ps_is_instrument_default_timed('memory/sql/udf_mem');
+---------------------------------------------------------+
| sys.ps_is_instrument_default_timed('memory/sql/udf_mem') |
+---------------------------------------------------------+
| NO                                                      |
+---------------------------------------------------------+

SELECT sys.ps_is_instrument_default_timed('memory/sql/nonexistent');
+-------------------------------------------------------------+
| sys.ps_is_instrument_default_timed('memory/sql/udf_memsds') |
+-------------------------------------------------------------+
| NO                                                          |
+-------------------------------------------------------------+

SELECT sys.ps_is_instrument_default_timed(NULL);
+-----------------------------------------+
| sys.ps_is_instrument_default_timed(NULL) |
+-----------------------------------------+
| NO                                      |
+-----------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ps_is_thread_instrumented

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.ps_is_thread_instrumented(connection_id)
```

Description

`ps_is_thread_instrumented` is a stored function available with the Sys Schema that returns whether or not Performance Schema instrumentation for the given *connection_id* is enabled.

- `YES` - instrumentation is enabled
- `NO` - instrumentation is not enabled
- `UNKNOWN` - the connection ID is unknown
- `NULL` - `NULL` value

Examples

```
SELECT sys.ps_is_thread_instrumented(CONNECTION_ID());
+-----------------------------------------------+
| sys.ps_is_thread_instrumented(CONNECTION_ID()) |
+-----------------------------------------------+
| YES                                           |
+-----------------------------------------------+

SELECT sys.ps_is_thread_instrumented(2042);
+-------------------------------------+
| sys.ps_is_thread_instrumented(2042) |
+-------------------------------------+
| UNKNOWN                             |
+-------------------------------------+

SELECT sys.ps_is_thread_instrumented(NULL);
+-------------------------------------+
| sys.ps_is_thread_instrumented(NULL) |
+-------------------------------------+
| NULL                                |
+-------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ps_thread_account

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.ps_thread_account(thread_id)
```

Description

`ps_thread_account` is a stored function available with the Sys Schema that returns the account (username@hostname) associated with the given *thread_id*.

Returns `NULL` if the thread_id is not found.

Examples

```
SELECT sys.ps_thread_account(sys.ps_thread_id(CONNECTION_ID()));
+-----------------------------------------------------------+
| sys.ps_thread_account(sys.ps_thread_id(CONNECTION_ID())) |
+-----------------------------------------------------------+
| msandbox@localhost                                        |
+-----------------------------------------------------------+

SELECT sys.ps_thread_account(sys.ps_thread_id(2042));
+-----------------------------------------------+
| sys.ps_thread_account(sys.ps_thread_id(2042)) |
+-----------------------------------------------+
| NULL                                          |
+-----------------------------------------------+

SELECT sys.ps_thread_account(sys.ps_thread_id(NULL));
+-----------------------------------------------+
| sys.ps_thread_account(sys.ps_thread_id(NULL)) |
+-----------------------------------------------+
| msandbox@localhost                            |
+-----------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ps_thread_id

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.ps_thread_id(connection_id)
```

Description

`ps_thread_id` is a stored function available with the Sys Schema that returns the thread_id associated with the given *connection_id*. If the *connection_id* is `NULL`, returns the thread_id for the current connection.

Examples

```
SELECT * FROM performance_schema.threads\G
*************************** 13. row ***************************
          THREAD_ID: 13
               NAME: thread/sql/one_connection
               TYPE: FOREGROUND
     PROCESSLIST_ID: 3
   PROCESSLIST_USER: msandbox
   PROCESSLIST_HOST: localhost
     PROCESSLIST_DB: test
PROCESSLIST_COMMAND: Query
   PROCESSLIST_TIME: 0
  PROCESSLIST_STATE: Sending data
   PROCESSLIST_INFO: SELECT * FROM performance_schema.threads
   PARENT_THREAD_ID: 1
               ROLE: NULL
       INSTRUMENTED: YES
            HISTORY: YES
    CONNECTION_TYPE: Socket
       THREAD_OS_ID: 24379

SELECT sys.ps_thread_id(3);
+---------------------+
| sys.ps_thread_id(3) |
+---------------------+
|                  13 |
+---------------------+

SELECT sys.ps_thread_id(NULL);
+------------------------+
| sys.ps_thread_id(NULL) |
+------------------------+
|                     13 |
+------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ps_thread_stack

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.ps_thread_stack(thread_id, verbose)
```

Description

`ps_thread_stack` is a stored function available with the Sys Schema that, for a given *thread_id*, returns all statements, stages, and events within the Performance Schema, as a JSON formatted stack.

The boolean *verbose* argument specifies whether or not to include `file:lineno` information in the events.

Examples

```
SELECT sys.ps_thread_stack(13, FALSE) AS thread_stack\G
*************************** 1. row ***************************
thread_stack: {"rankdir": "LR","nodesep": "0.10",
  "stack_created": "2022-03-28 16:01:06",
  "mysql_version": "10.8.2-MariaDB",
  "mysql_user": "msandbox@localhost",
  "events": []}
```

This page is licensed: CC BY-SA / Gnu FDL

ps_thread_trx_info

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.ps_thread_trx_info(thread_id)
```

Description

`ps_thread_trx_info` is a stored function available with the Sys Schema.

It returns a JSON object with information about the thread specified by the given *thread_id*. This information includes:

- the current transaction;
- executed statements (derived from the Performance Schema events_transactions_current Table and the Performance Schema events_statements_history Table (full data will only returned if the consumers for those tables are enabled).

The maximum length of the returned JSON object is determined by the value of the ps_thread_trx_info.max_length sys_config option (by default 65535). If the returned value exceeds this length, a JSON object error is returned.

See Also

- Sys Schema sys_config Table

This page is licensed: CC BY-SA / Gnu FDL

quote_identifier

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.quote_identifier(str)
```

Description

`quote_identifier` is a stored function available with the Sys Schema.

It quotes a string to produce a result that can be used as an identifier in an SQL statement. The string is returned enclosed by backticks (“`") and with each instance of backtick ("`”) doubled. If the argument is `NULL`, the

return value is the word "`NULL`" without enclosing backticks.

Examples

```
SELECT sys.quote_identifier("Identifier with spaces");
+-----------------------------------------------+
| sys.quote_identifier("Identifier with spaces") |
+-----------------------------------------------+
| `Identifier with spaces`                      |
+-----------------------------------------------+

SELECT sys.quote_identifier("Identifier` containing `backticks");
+-------------------------------------------------------------+
| sys.quote_identifier("Identifier` containing `backticks") |
+-------------------------------------------------------------+
| `Identifier`` containing ``backticks`                      |
+-------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

sys_get_config

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.sys_get_config(name,default)
```

Description

`sys_get_config` is a stored function available with the Sys Schema.

The function returns a configuration option value from the sys_config table. It takes two arguments; *name*, a configuration option name, and *default*, which is returned if the given option does not exist in the table.

Both arguments are `VARCHAR(128)` and can be `NULL`. Returns `NULL` if *name* is `NULL`, or if the given option is not found and *default* is `NULL`.

Examples

```
SELECT sys.sys_get_config('ps_thread_trx_info.max_length',NULL);
+-----------------------------------------------------------+
| sys.sys_get_config('ps_thread_trx_info.max_length',NULL) |
+-----------------------------------------------------------+
| 65535                                                     |
+-----------------------------------------------------------+
```

See Also

- Sys Schema sys_config Table

This page is licensed: CC BY-SA / Gnu FDL

version_major

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.version_major()
```

Description

`version_major` is a stored function available with the Sys Schema.

It returns the MariaDB Server major release version.

Examples

```
SELECT VERSION(),
 sys.version_major() AS major,
 sys.version_minor() AS minor,
 sys.version_patch() AS patch;
+----------------+-------+-------+-------+
| VERSION()      | major | minor | patch |
+----------------+-------+-------+-------+
| 10.8.2-MariaDB |    10 |     8 |     2 |
+----------------+-------+-------+-------+
```

See Also

- version_minor
- version_patch

This page is licensed: CC BY-SA / Gnu FDL

version_minor

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.version_minor()
```

Description

`version_minor` is a stored function available with the Sys Schema.

It returns the MariaDB Server minor release version.

Examples

```
SELECT VERSION(),
 sys.version_major() AS major,
 sys.version_minor() AS minor,
 sys.version_patch() AS patch;
+----------------+-------+-------+-------+
| VERSION()      | major | minor | patch |
+----------------+-------+-------+-------+
| 10.8.2-MariaDB |    10 |     8 |     2 |
+----------------+-------+-------+-------+
```

See Also

- version_major

- version_patch

This page is licensed: CC BY-SA / Gnu FDL

version_patch

> **Note:**
>
> Sys Schema is available from MariaDB 10.6.

Syntax

```
sys.version_patch()
```

Description

`version_patch` is a stored function available with the Sys Schema.

It returns the MariaDB Server patch release version.

Examples

```
SELECT VERSION(),
 sys.version_major() AS major,
 sys.version_minor() AS minor,
 sys.version_patch() AS patch;
+----------------+-------+-------+-------+
| VERSION()      | major | minor | patch |
+----------------+-------+-------+-------+
| 10.8.2-MariaDB |    10 |     8 |     2 |
+----------------+-------+-------+-------+
```

See Also

- version_major
- version_minor

This page is licensed: CC BY-SA / Gnu FDL

Sys Schema Stored Procedures

See create_synonym_db.

The create_synonym_db stored procedure creates a new database that contains views mirroring all tables from a source database, useful for creating aliases.

See optimizer_switch Helper Functions.

These helper functions allow you to easily enable or disable specific optimizer_switch flags for the current session.

See ps_trace_statement_digest.

This procedure traces a specific statement digest in the Performance Schema, capturing details about its execution for performance analysis.

See ps_trace_thread.

The ps_trace_thread procedure captures a trace of Performance Schema instrumentation for a specific thread and dumps it to a .dot formatted graph file.

See ps_truncate_all_tables.

This procedure truncates all Performance Schema summary tables, effectively resetting all aggregated performance statistics.

See statement_performance_analyzer.

This diagnostic procedure creates a report of the statements currently running or recently run on the server, aiding in performance troubleshooting.

See table_exists.

The table_exists procedure checks for the existence of a specific table, view, or temporary table within a given database.

This page is licensed: CC BY-SA / Gnu FDL

create_synonym_db

Syntax

```
create_synonym_db(db_name,synonym)

# db_name (VARCHAR(64))
# synonym (VARCHAR(64))
```

Description

`create_synonym_db` is a stored procedure available with the Sys Schema.

Takes a source database name *db_name* and *synonym* name and creates a synonym database with views that point to all of the tables within the source database. Useful for example for creating a synonym for the performance_schema or information_schema databases.

Returns an error if the source database doesn't exist, or the synonym already exists.

Example

```
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+

CALL sys.create_synonym_db('performance_schema', 'perf');
+----------------------------------------+
| summary                                |
+----------------------------------------+
| Created 81 views in the `perf` database |
+----------------------------------------+

SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| perf               |
| performance_schema |
| sys                |
| test               |
+--------------------+

SHOW FULL TABLES FROM perf;
+-----------------------------------------------------+------------+
| Tables_in_perf                                      | Table_type |
+-----------------------------------------------------+------------+
| accounts                                            | VIEW       |
| cond_instances                                      | VIEW       |
| events_stages_current                               | VIEW       |
| events_stages_history                               | VIEW       |
| events_stages_history_long                          | VIEW       |
...
```

This page is licensed: CC BY-SA / Gnu FDL

optimizer_switch Helper Functions

Syntax

```
optimizer_switch_on()
optimizer_switch_off()
optimizer_switch_choice("on" | "off")
```

Description

The above procedures can be used to check which optimizer_switch options are `on` or `off`. The result set is sorted according to optimizer_switch option name.

Example

```
SELECT @@optimizer_switch\G
*************************** 1. row ***************************
index_merge=on,index_merge_union=on,index_merge_sort_union=on,index_merge_intersection=on,
index_merge_sort_intersection=off,engine_condition_pushdown=off,index_condition_pushdown=on,
derived_merge=on,derived_with_keys=on,firstmatch=on,loosescan=on,materialization=on,
in_to_exists=on,semijoin=on,partial_match_rowid_merge=on,partial_match_table_scan=on,
subquery_cache=on,mrr=off,mrr_cost_based=off,mrr_sort_keys=off,outer_join_with_cache=on,
semijoin_with_cache=on,join_cache_incremental=on,join_cache_hashed=on,join_cache_bka=on,
optimize_join_buffer_size=on,table_elimination=on,extended_keys=on,exists_to_in=on,
orderby_uses_equalities=on,condition_pushdown_for_derived=on,split_materialized=on,
condition_pushdown_for_subquery=on,rowid_filter=on,condition_pushdown_from_having=on,
not_null_range_scan=off

call sys.optimizer_switch_on();
+---------------------------------+------+
| option                          | opt  |
+---------------------------------+------+
| condition_pushdown_for_derived  | on   |
| condition_pushdown_for_subquery | on   |
| condition_pushdown_from_having  | on   |
| derived_merge                   | on   |
| derived_with_keys               | on   |
| exists_to_in                    | on   |
| extended_keys                   | on   |
| firstmatch                      | on   |
| index_condition_pushdown        | on   |
| index_merge                     | on   |
| index_merge_intersection        | on   |
| index_merge_sort_union          | on   |
| index_merge_union               | on   |
| in_to_exists                    | on   |
| join_cache_bka                  | on   |
| join_cache_hashed               | on   |
| join_cache_incremental          | on   |
| loosescan                       | on   |
| materialization                 | on   |
| optimize_join_buffer_size       | on   |
| orderby_uses_equalities         | on   |
| outer_join_with_cache           | on   |
| partial_match_rowid_merge       | on   |
| partial_match_table_scan        | on   |
| rowid_filter                    | on   |
| semijoin                        | on   |
| semijoin_with_cache             | on   |
| split_materialized              | on   |
| subquery_cache                  | on   |
| table_elimination               | on   |
+---------------------------------+------+

call sys.optimizer_switch_off();
+-------------------------------+------+
| option                        | opt  |
+-------------------------------+------+
| engine_condition_pushdown     | off  |
| index_merge_sort_intersection | off  |
| mrr                           | off  |
| mrr_cost_based                | off  |
| mrr_sort_keys                 | off  |
| not_null_range_scan           | off  |
+-------------------------------+------+
```

Notes

`sys.optimizer_switch_on()` is a shortcut for `sys.optimizer_switch_choice("on")` .

`sys.optimizer_switch_off()` is a shortcut for `sys.optimizer_switch_choice("off")` .

See Also

- optimizer-switch Documentation for optimizer_switch

This page is licensed: CC BY-SA / Gnu FDL

ps_trace_statement_digest

Syntax

```
ps_trace_statement_digest(in_digest, in_runtime, in_interval, in_start_fresh, in_auto_enable)
```

Description

`ps_trace_statement_digest` is a stored procedure available with the Sys Schema.

Parameters:

- in_digest `VARCHAR(32)`: The statement digest identifier to analyze.
- in_runtime `INT`: Specifies the duration to run the analysis in seconds.
- in_interval `DECIMAL(2,2)`: The analysis interval measured in seconds (including fraction values), at which snapshots are taken.
- in_start_fresh `BOOLEAN`: Determines whether to truncate the Performance Schema events_statements_history_long and events_stages_history_long tables before starting.
- in_auto_enable `BOOLEAN`: Determines whether to automatically enable required consumers.

Examples

```
CALL sys.ps_trace_statement_digest('891ec6860f98ba46d89dd20b0c03652c', 5, 0.5, TRUE, TRUE);
```

This page is licensed: CC BY-SA / Gnu FDL

ps_trace_thread

Syntax

```
ps_trace_thread(thread_id, outfile, max_runtime, interval, start_fresh, auto_setup, debug)
```

Description

`ps_trace_thread` is a stored procedure available with the Sys Schema.

Parameters:

- thread_id `INT`: The thread to trace.
- outfile `VARCHAR(255)`: Name of the .dot file to be create.
- max_runtime `DECIMAL(20,2)`: Maximum time in seconds to collect data. Fractional seconds can be used, and `NULL` results in data being collected for the default sixty seconds.
- interval `DECIMAL(20,2)`: Time in seconds to sleep between data collection. Fractional seconds can be used, and `NULL` results in the sleep being the default one second.
- start_fresh `BOOLEAN`: Whether to reset all Performance Schema data before tracing.
- auto_setup `BOOLEAN`: Whether to disable all other threads, enable all instruments and consumers, and reset the settings at the end of the run.
- debug `BOOLEAN`: Whether to include `file:lineno` information in the graph.

Dumps all Performance Schema data for an instrumented thread to a .dot formatted graph file (for use with the DOT graph description language). All returned result sets should be used for a complete graph.

Session binary logging is disabled during execution, by adjusting the sql_log_bin session value (note the permissions required).

Examples

```
CALL sys.ps_trace_thread(25, CONCAT('/tmp/stack-', REPLACE(NOW(), ' ', '-'), '.dot'),
  NULL, NULL, TRUE, TRUE, TRUE);

--------------------+
| summary            |
+--------------------+
| Disabled 0 threads |
+--------------------+

+-------------------------------------------+
| Info                                      |
+-------------------------------------------+
| Data collection starting for THREAD_ID = 25 |
+-------------------------------------------+

+-----------------------------------------------------------+
| Info                                                      |
+-----------------------------------------------------------+
| Stack trace written to /tmp/stack-2023-04-05-19:06:29.dot |
+-----------------------------------------------------------+

+-------------------------------------------------------------------+
| Convert to PDF                                                    |
+-------------------------------------------------------------------+
| dot -Tpdf -o /tmp/stack_25.pdf /tmp/stack-2023-04-05-19:06:29.dot |
+-------------------------------------------------------------------+

+-------------------------------------------------------------------+
| Convert to PNG                                                    |
+-------------------------------------------------------------------+
| dot -Tpng -o /tmp/stack_25.png /tmp/stack-2023-04-05-19:06:29.dot |
+-------------------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

ps_truncate_all_tables

Syntax

```
ps_truncate_all_tables(bool display)
```

Description

`ps_truncate_all_tables` is a stored procedure available with the Sys Schema.

The procedure resets all aggregated instrumentation as a snapshot, producing a result set indicating the number of truncated tables. The boolean parameter *display* specifies whether to display each TRUNCATE TABLE statement before execution.

Examples

```
CALL sys.ps_truncate_all_tables(false);
+---------------------+
| summary             |
+---------------------+
| Truncated 44 tables |
+---------------------+
```

```
CALL sys.ps_truncate_all_tables(true);
+-----------------------------------------------------------------+
| status                                                          |
+-----------------------------------------------------------------+
| Running: TRUNCATE TABLE performance_schema.events_stages_history |
+-----------------------------------------------------------------+

...

+-------------------------------------------------------------------------------+
| status                                                                        |
+-------------------------------------------------------------------------------+
| Running: TRUNCATE TABLE performance_schema.table_lock_waits_summary_by_table |
+-------------------------------------------------------------------------------+

+---------------------+
| summary             |
+---------------------+
| Truncated 44 tables |
+---------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

statement_performance_analyzer

Syntax

```
statement_performance_analyzer(in_action,in_table, in_views)

# in_action ENUM('snapshot', 'overall', 'delta', 'create_tmp',
                 'create_table', 'save', 'cleanup')
# in_table VARCHAR(129)
# in_views SET ('with_runtimes_in_95th_percentile', 'analysis',
                'with_errors_or_warnings', 'with_full_table_scans',
                'with_sorting', 'with_temp_tables', 'custom')
```

Description

`statement_performance_analyzer` is a stored procedure available with the Sys Schema which returns a report on running statements.

The following options from the sys_config table impact the output:

- statement_performance_analyzer.limit - maximum number of rows (default `100`) returned for views that have no built-in limit.
- statement_performance_analyzer.view - custom query/view to be used (default `NULL`). If the statement_performance_analyzer.limit configuration option is greater than 0, there can't be a `LIMIT` clause in the query/view definition.

If the debug option is set (default `OFF`), the procedure will also produce debugging output.

This page is licensed: CC BY-SA / Gnu FDL

table_exists

Syntax

```
table_exists(in_db_name,in_table_name, out_table_type)

# in_db_name VARCHAR(64)
# in_table_name VARCHAR(64)
# out_table_type ENUM('', 'BASE TABLE', 'VIEW', 'TEMPORARY')
```

Description

`table_exists` is a stored procedure available with the Sys Schema.

Given a database *in_db_name* and table name *in_table_name*, returns the table type in the `OUT` parameter *out_table_type*. The return value is an `ENUM` field containing one of:

- '' - the table does not exist
- '`BASE TABLE`' - a regular table
- '`VIEW`' - a view
- '`TEMPORARY`' - a temporary table

Examples

```
CALL sys.table_exists('mysql', 'time_zone', @table_type); SELECT @table_type;
+-------------+
| @table_type |
+-------------+
| BASE TABLE  |
+-------------+

CALL sys.table_exists('mysql', 'user', @table_type); SELECT @table_type;
+-------------+
| @table_type |
+-------------+
| VIEW        |
+-------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Sys Schema Views

- host_summary and x$host_summary Sys Schema Views
- host_summary_by_file_io and x$host_summary_by_file_io Sys Schema Views
- host_summary_by_file_io_type and x$host_summary_by_file_io_type Sys Schema Views
- host_summary_by_stages and x$host_summary_by_stages Sys Schema Views
- host_summary_by_statement_type and x$host_summary_by_statement_type Sys Schema Views
- host_summary_by_statement_latency and x$host_summary_by_statement_latency Sys Schema Views
- innodb_buffer_stats_by_schema and x$innodb_buffer_stats_by_schema Sys Schema Views
- innodb_buffer_stats_by_table and x$innodb_buffer_stats_by_table Sys Schema Views
- innodb_lock_waits and x$innodb_lock_waits Sys Schema Views
- io_by_thread_by_latency and x$io_by_thread_by_latency Sys Schema Views
- io_global_by_file_by_bytes and x$io_global_by_file_by_bytes Sys Schema Views
- io_global_by_file_by_latency and x$io_global_by_file_by_latency Sys Schema Views
- io_global_by_wait_by_bytes and x$io_global_by_wait_by_bytes Sys Schema Views
- io_global_by_wait_by_latency and x$io_global_by_wait_by_latency Sys Schema Views

- latest_file_io and x$latest_file_io Sys Schema Views
- memory_by_host_by_current_bytes and x$memory_by_host_by_current_bytes Views
- metrics Sys Schema View
- privileges_by_table_by_level Sys Schema View
- schema_auto_increment_columns Sys Schema View
- schema_object_overview Sys Schema View

This page is licensed: CC BY-SA / Gnu FDL

host_summary and x$host_summary Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `host_summary` and `x$host_summary` views contain host activity information, grouped by host. The `host_summary` view is intended to be easier for human reading, while the `x$host_summary` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
host	Host that the client connected from, or background for background threads (where the HOST column in the underlying Performance Schema table is NULL).
statements	Total number of statements for the host.
statement_latency	Total wait time of timed statements for the host.
statement_avg_latency	Average wait time per timed statement for the host.
table_scans	Total table scans for the host.
file_ios	Total file I/O events for the host.
file_io_latency	Total wait time of timed file I/O events for the host.
current_connections	Current connections for the host.
total_connections	Total connections for the host.
unique_users	Number of distinct users for the host.
current_memory	Current allocated memory for the host.
total_memory_allocated	Total allocated memory for the host.

Example

```
SELECT * FROM sys.host_summary\G
*************************** 1. row ***************************
                    host: localhost
              statements: 59
     statement_latency: 148.11 ms
 statement_avg_latency: 2.51 ms
           table_scans: 11
              file_ios: 2065
       file_io_latency: 79.57 ms
   current_connections: 1
     total_connections: 3
          unique_users: 1
        current_memory: -2672 bytes
total_memory_allocated: 0 bytes

SELECT * FROM sys.x$host_summary\G
*************************** 1. row ***************************
                    host: localhost
              statements: 98
     statement_latency: 160926285000
 statement_avg_latency: 1642104948.9796
           table_scans: 12
              file_ios: 2071
       file_io_latency: 79742533755
   current_connections: 1
     total_connections: 3
          unique_users: 1
        current_memory: -2672
total_memory_allocated: 0
```

This page is licensed: CC BY-SA / Gnu FDL

host_summary_by_file_io and x$host_summary_by_file_io Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `host_summary_by_file_io` and `x$host_summary_by_file_io` views summarize file I/O, grouped by host. Rows are sorted by descending total file I/O latency by default.

The `host_summary_by_file_io` view is intended to be easier for human reading, while the `and` x$host_summary_by_file_io `view provides the data in raw form, intended for tools that process the data.`

They contain the following columns:

Column	Description
host	Host that the client connected from, or background for background threads (where the HOST column in the underlying Performance Schema table is NULL).
ios	Total file I/O events for the host.
ios_latency	Total wait time of timed file I/O events for the host.

Example

```
SELECT * FROM sys.host_summary_by_file_io\G
*************************** 1. row ***************************
      host: localhost
       ios: 6526
io_latency: 490.28 ms
*************************** 2. row ***************************
      host: background
       ios: 457
io_latency: 151.39 ms

SELECT * FROM sys.x$host_summary_by_file_io\G
*************************** 1. row ***************************
      host: localhost
       ios: 6532
io_latency: 490447878974
*************************** 2. row ***************************
      host: background
       ios: 457
io_latency: 151388125856
```

This page is licensed: CC BY-SA / Gnu FDL

host_summary_by_file_io_type and x$host_summary_by_file_io_type Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `host_summary_by_file_io_type` and `x$host_summary_by_file_io_type` views summarize file I/O, grouped by host and event type. Rows are sorted by host and descending total I/O latency by default. The `host_summary_by_file_io_type` view is intended to be easier for human reading, while the `x$host_summary_by_file_io_type` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
host	Host that the client connected from, or background for background threads (where the HOST column in the underlying Performance Schema table is NULL).
event_name	File I/O event name.
total	Total number of occurrences of the file I/O event for the host.
total_latency	Total wait time of timed occurrences of the file I/O event for the host.
max_latency	Maximum single wait time of timed occurrences of the file I/O event for the host.

Example

```
SELECT * FROM sys.host_summary_by_file_io_type;
+------------+-------------------------------------------+-------+---------------+-------------+
| host       | event_name                                | total | total_latency | max_latency |
+------------+-------------------------------------------+-------+---------------+-------------+
| background | wait/io/file/innodb/innodb_log_file       |    45 | 109.80 ms     | 26.48 ms    |
| background | wait/io/file/innodb/innodb_data_file      |   195 | 29.47 ms      | 1.23 ms     |
| background | wait/io/file/sql/global_ddl_log           |     4 | 4.45 ms       | 4.33 ms     |
...
| localhost  | wait/io/file/csv/data                     |     4 | 25.98 us      | 9.60 us     |
| localhost  | wait/io/file/partition/ha_partition::parfile |     1 | 14.19 us      | 14.19 us    |
| localhost  | wait/io/file/myisam/kfile                 |     1 | 11.95 us      | 11.95 us    |
+------------+-------------------------------------------+-------+---------------+-------------+

SELECT * FROM sys.x$host_summary_by_file_io_type;
+------------+-------------------------------------------+-------+---------------+-------------+
| host       | event_name                                | total | total_latency | max_latency |
+------------+-------------------------------------------+-------+---------------+-------------+
| background | wait/io/file/innodb/innodb_log_file       |    45 |  109804643160 | 26478157582 |
| background | wait/io/file/innodb/innodb_data_file      |   195 |   29469738630 |  1226986584 |
| background | wait/io/file/sql/global_ddl_log           |     4 |    4447263252 |  4327780456 |
| localhost  | wait/io/file/csv/data                     |     4 |      25978718 |     9603922 |
| localhost  | wait/io/file/partition/ha_partition::parfile |     1 |      14191190 |    14191190 |
| localhost  | wait/io/file/myisam/kfile                 |     1 |      11954300 |    11954300 |
+------------+-------------------------------------------+-------+---------------+-------------+
```

This page is licensed: CC BY-SA / Gnu FDL

host_summary_by_stages and x$host_summary_by_stages Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `host_summary_by_stages` and `x$host_summary_by_stages` views summarize statement stages, grouped by host. Rows are sorted by host and descending total latency by default. The `host_summary_by_stages` view is intended to be easier for human reading, while the `x$host_summary_by_stages` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
host	Host that the client connected from, or background for background threads (where the HOST column in the underlying Performance Schema table is NULL).
event_name	Stage event name.
total	Total number of occurrences of the file stage event for the host.
total_latency	Total wait time of timed occurrences of the stage event for the host.
avg_latency	Average wait time per timed occurrence of the stage event for the host.

Example

```
SELECT * FROM sys.host_summary_by_stages\G
*************************** 1. row ***************************
         host: background
   event_name: stage/innodb/buffer pool load
        total: 1
total_latency: 3.75 ms
  avg_latency: 3.75 ms

SELECT * FROM sys.x$host_summary_by_stages\G
*************************** 1. row ***************************
         host: background
   event_name: stage/innodb/buffer pool load
        total: 1
total_latency: 3747098000
  avg_latency: 3747098000
```

This page is licensed: CC BY-SA / Gnu FDL

host_summary_by_statement_type and x$host_summary_by_statement_type Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `host_summary_by_statement_type` and `x$host_summary_by_statement_type` views summarize information about executed statements, grouped by host and statement type. Rows are sorted by host and descending total latency by default.

The `host_summary_by_statement_type` view is intended to be easier for human reading, while the `x$host_summary_by_statement_type` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
host	Host that the client connected from, or background for background threads (where the HOST column in the underlying Performance Schema table is NULL).
statement	Final component of the statement event name, for example create_table or select.
total	Total number of statement occurrences for the host.
total_latency	Total wait time of timed statements of the statement event for the host.
max_latency	Maximum single wait time of timed occurrences of the statement event for the host.
lock_latency	Total time spent by timed occurrences of the statement event for the host waiting for locks.
rows_sent	Total number of rows returned by occurrences of the statement event for the host.
rows_examined	Total number of rows read from storage engines by occurrences of the statement event for the host.
rows_affected	Total number of rows affected by occurrences of the statement event for the host.
full_scans	Total number of full table scans by occurrences of the statement event for the host.

Example

```
SELECT * FROM sys.host_summary_by_statement_type\G
*************************** 1. row ***************************
         host: localhost
    statement: create_table
        total: 18
total_latency: 366.93 ms
  max_latency: 48.02 ms
 lock_latency: 3.16 ms
    rows_sent: 0
rows_examined: 0
rows_affected: 0
   full_scans: 0
*************************** 2. row ***************************
         host: localhost
    statement: select
        total: 27
total_latency: 339.16 ms
  max_latency: 64.51 ms
 lock_latency: 205.61 ms
    rows_sent: 750599937895926
rows_examined: 13925
rows_affected: 0
   full_scans: 21

...

SELECT * FROM sys.x$host_summary_by_statement_type\G
*************************** 1. row ***************************
         host: localhost
    statement: create_table
        total: 18
total_latency: 366927804000
  max_latency: 48023563000
 lock_latency: 3156000000
    rows_sent: 0
rows_examined: 0
rows_affected: 0
   full_scans: 0
*************************** 2. row ***************************
         host: localhost
    statement: select
        total: 28
total_latency: 343873182000
  max_latency: 64507216000
 lock_latency: 205984000000
    rows_sent: 750678474440767
rows_examined: 14370
rows_affected: 0
   full_scans: 22
```

This page is licensed: CC BY-SA / Gnu FDL

host_summary_by_statement_latency and x$host_summary_by_statement_latency Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `host_summary_by_statement_latency` and `x$host_summary_by_statement_latency` views summarize statement statistics, grouped by host. Rows are sorted by descending total latency by default. The `host_summary_by_statement_latency` view is intended to be easier for human reading, while the

`x$host_summary_by_statement_latency` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
host	Host that the client connected from, or background for background threads (where the HOST column in the underlying Performance Schema table is NULL).
total	Total number of statements for the host.
max_latency	Maximum single wait time of timed statements for the host.
lock_latency	Total time spent by timed statements for the host waiting for locks.
total_latency	Total wait time of timed statements for the host.
rows_sent	Total number of rows returned by statements for the host.
rows_examined	Total number of rows read from storage engines by statements for the host.
rows_affected	Total number of rows affected by statements for the host.
full_scans	Total number of full table scans by statements for the host.

Example

```
SELECT * FROM sys.host_summary_by_statement_latency\G
*************************** 1. row ***************************
         host: localhost
        total: 1042
total_latency: 816.89 ms
  max_latency: 64.51 ms
 lock_latency: 215.64 ms
    rows_sent: 750599937895985
rows_examined: 13548
rows_affected: 6
   full_scans: 33
*************************** 2. row ***************************
         host: background
        total: 0
total_latency: 0 ps
  max_latency: 0 ps
 lock_latency: 0 ps
    rows_sent: 0
rows_examined: 0
rows_affected: 0
   full_scans: 0

SELECT * FROM sys.x$host_summary_by_statement_latency\G
*************************** 1. row ***************************
         host: localhost
        total: 1041
total_latency: 812132706000
  max_latency: 64507216000
 lock_latency: 215301000000
    rows_sent: 750599937895983
rows_examined: 13110
rows_affected: 6
   full_scans: 32
*************************** 2. row ***************************
         host: background
        total: 0
total_latency: 0
  max_latency: 0
 lock_latency: 0
    rows_sent: 0
rows_examined: 0
rows_affected: 0
   full_scans: 0
```

This page is licensed: CC BY-SA / Gnu FDL

innodb_buffer_stats_by_schema and x$innodb_buffer_stats_by_schema Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Querying these views can have a noticeable performance impact on a production server.

Description

The `innodb_buffer_stats_by_schema` and `x$innodb_buffer_stats_by_schema` views summarize information from the Information Schema INNODB_BUFFER_PAGE table, grouped by schema. Rows are sorted by descending buffer size by default.

The `innodb_buffer_stats_by_schema` view is intended to be easier for human reading, while the `x$innodb_buffer_stats_by_schema` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
object_schema	Object schema name, or InnoDB System if the table belongs to the InnoDB storage engine.
allocated	Total number of bytes allocated for the schema.
data	Total number of data bytes allocated for the schema.
pages	Total number of pages allocated for the schema.
pages_hashed	Total number of hashed pages allocated for the schema.
pages_old	Total number of old pages allocated for the schema.
rows_cached	Total number of cached rows for the schema.

Example

```
SELECT * FROM sys.innodb_buffer_stats_by_schema\G
*************************** 1. row ***************************
object_schema: InnoDB System
    allocated: 160.00 KiB
         data: 6.21 KiB
        pages: 10
 pages_hashed: 10
    pages_old: 10
  rows_cached: 21
*************************** 2. row ***************************
object_schema: mysql
    allocated: 112.00 KiB
         data: 1.73 KiB
        pages: 7
 pages_hashed: 7
    pages_old: 7
  rows_cached: 5
*************************** 3. row ***************************
object_schema: test
    allocated: 64.00 KiB
         data: 0 bytes
        pages: 4
 pages_hashed: 4
    pages_old: 4
  rows_cached: 0

SELECT * FROM sys.x$innodb_buffer_stats_by_schema\G
*************************** 1. row ***************************
object_schema: InnoDB System
    allocated: 163840
         data: 6362
        pages: 10
 pages_hashed: 0
    pages_old: 0
  rows_cached: 21
*************************** 2. row ***************************
object_schema: mysql
    allocated: 114688
         data: 1775
        pages: 7
 pages_hashed: 0
    pages_old: 0
  rows_cached: 5
*************************** 3. row ***************************
object_schema: test
    allocated: 65536
         data: 0
        pages: 4
 pages_hashed: 0
    pages_old: 0
  rows_cached: 0
```

This page is licensed: CC BY-SA / Gnu FDL

innodb_buffer_stats_by_table and x$innodb_buffer_stats_by_table Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Querying these views can have a noticeable performance impact on a production server.

Description

The `innodb_buffer_stats_by_table` and `x$innodb_buffer_stats_by_table` views summarize information from the Information Schema INNODB_BUFFER_PAGE table, grouped by schema and table. Rows are sorted by descending buffer size by default.

The `innodb_buffer_stats_by_table` view is intended to be easier for human reading, while the `x$innodb_buffer_stats_by_table` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
object_schema	Object schema name, or InnoDB System if the table belongs to the InnoDB storage engine.
object_name	Table name.
allocated	Total number of bytes allocated for the table.
data	Number of data bytes allocated for the table.
pages	Total number of pages allocated for the table.
pages_hashed	Total number of hashed pages allocated for the table.
pages_old	Total number of old pages allocated for the table.
rows_cached	Total number of cached rows for the table.

Example

```
SELECT * FROM sys.innodb_buffer_stats_by_table\G
*************************** 1. row ***************************
object_schema: mysql
  object_name: transaction_registry
    allocated: 64.00 KiB
         data: 0 bytes
        pages: 4
 pages_hashed: 4
    pages_old: 4
  rows_cached: 0
*************************** 2. row ***************************
object_schema: InnoDB System
  object_name: SYS_FOREIGN
    allocated: 48.00 KiB
         data: 0 bytes
        pages: 3
 pages_hashed: 3
    pages_old: 3
  rows_cached: 0
*************************** 3. row ***************************
object_schema: InnoDB System
  object_name: SYS_TABLES
    allocated: 32.00 KiB
         data: 1.07 KiB
        pages: 2
 pages_hashed: 2
    pages_old: 2
  rows_cached: 10

...

 SELECT * FROM sys.x$innodb_buffer_stats_by_table\G
*************************** 1. row ***************************
object_schema: mysql
  object_name: transaction_registry
    allocated: 65536
         data: 0
        pages: 4
 pages_hashed: 0
    pages_old: 0
  rows_cached: 0
*************************** 2. row ***************************
object_schema: InnoDB System
  object_name: SYS_FOREIGN
    allocated: 49152
         data: 0
        pages: 3
 pages_hashed: 0
    pages_old: 0
  rows_cached: 0
*************************** 3. row ***************************
object_schema: InnoDB System
  object_name: SYS_TABLES
    allocated: 32768
         data: 1100
        pages: 2
 pages_hashed: 0
    pages_old: 0
  rows_cached: 10
....
```

This page is licensed: CC BY-SA / Gnu FDL

innodb_lock_waits and x$innodb_lock_waits Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `innodb_lock_waits` and `x$innodb_lock_waits` views summarize InnoDB locks that transactions are waiting upon, by default sorted in descending buffer size.

The `innodb_lock_waits` view is intended to be easier for human reading, while the `x$innodb_lock_waits` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
wait_started	Time that lock wait began.
wait_age	TIME value for the length of the lock wait.
wait_age_secs	Seconds value for the length of the lock wait.
locked_table_schema	Schema containing the locked table.
locked_table_name	Name of the locked table.
locked_table_partition	Name of the locked partition, or NULL if none.
locked_table_subpartition	Name of the locked subpartition, or NULL if none.
locked_index	Name of the locked index.
locked_type	Type of the waiting lock.
waiting_trx_id	ID of the waiting transaction.
waiting_trx_started	Time that the waiting transaction started.
waiting_trx_age	TIME value for the length of time that the transaction has been waiting.
waiting_trx_rows_locked	Number of rows locked by the waiting transaction.
waiting_trx_rows_modified	Number of rows modified by the waiting transaction.
waiting_pid	Processlist ID of the waiting transaction.
waiting_query	Statement waiting for the lock.
waiting_lock_id	ID of the waiting lock.
waiting_lock_mode	Mode of the waiting lock.
blocking_trx_id	ID of the transaction blocking the waiting lock.
blocking_pid	Processlist ID of the blocking transaction.
blocking_query	Statement the blocking transaction is executing, or NULL if the session that issued the blocking query has become idle.
blocking_lock_id	ID of the lock blocking the waiting lock.
blocking_lock_mode	Mode of the lock blocking the waiting lock.
blocking_trx_started	Time the blocking transaction started.
blocking_trx_age	TIME value for how long the blocking transaction has been executing.
blocking_trx_rows_locked	Number of rows locked by the blocking transaction.
blocking_trx_rows_modified	Number of rows modified by the blocking transaction.
sql_kill_blocking_query	KILL statement that could be used to kill the blocking statement.

Column	Description
sql_kill_blocking_connection	KILL statement that could be used to kill the blocking statement session.

This page is licensed: CC BY-SA / Gnu FDL

io_by_thread_by_latency and x$io_by_thread_by_latency Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `io_by_thread_by_latency` and `x$io_by_thread_by_latency` view summarize I/O consumers to display time waiting for I/O, grouped by thread. Rows are sorted by descending total I/O latency by default.

The `io_by_thread_by_latency` view is intended to be easier for human reading, while the `x$io_by_thread_by_latency` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
user	The account associated with a foreground thread, or the background thread name
total	Total number of I/O events allocated for the thread.
total_latency	Total wait time of timed I/O events for the thread.
min_latency	Minimum single wait time of timed I/O events for the thread.
avg_latency	Average wait time per timed I/O event for the thread.
min_latency	Maximum single wait time of timed I/O events for the thread.
thread_id	Thread id.
processlist_id	Processlist id for foreground threads, or NULL for background threads.

Example

```
SELECT * FROM sys.io_by_thread_by_latency\G
*************************** 1. row ***************************
          user: main
         total: 378
 total_latency: 40.11 ms
   min_latency: 570.21 ns
   avg_latency: 206.02 us
   max_latency: 4.33 ms
     thread_id: 1
processlist_id: NULL
*************************** 2. row ***************************
          user: msandbox@localhost
         total: 521
 total_latency: 10.28 ms
   min_latency: 775.04 ns
   avg_latency: 21.79 us
   max_latency: 977.79 us
     thread_id: 89
processlist_id: 7

...

SELECT * FROM sys.x$io_by_thread_by_latency\G
*************************** 1. row ***************************
          user: main
         total: 378
 total_latency: 40106340880
   min_latency: 570208
   avg_latency: 206016046.6000
   max_latency: 4327780456
     thread_id: 1
processlist_id: NULL
*************************** 2. row ***************************
          user: msandbox@localhost
         total: 498
 total_latency: 9637694714
   min_latency: 775040
   avg_latency: 21364289.0000
   max_latency: 977787350
     thread_id: 89
processlist_id: 7

...
```

This page is licensed: CC BY-SA / Gnu FDL

io_global_by_file_by_bytes and x$io_global_by_file_by_bytes Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `io_global_by_file_by_bytes` and `x$io_global_by_file_by_bytes` views summarize global I/O consumers showing I/O in bytes, grouped by file. Rows are sorted by descending total I/O (bytes read and written) by default.

The `io_global_by_file_by_bytes` view is intended to be easier for human reading, while the `x$io_global_by_file_by_bytes` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
file	File path name.
count_read	Total number of read events for the file.
total_read	Total number of bytes read from the file.
avg_read	Average number of bytes per read from the file.
count_write	Total number of write events for the file.
total_written	Total number of bytes written to the file.
avg_write	Average number of bytes per write to the file.
total	Total number of bytes read and written for the file.
write_pct	Percentage of total I/O bytes that were writes.

Example

```
SELECT * FROM sys.io_global_by_file_by_bytes\G
...
*************************** 3. row ***************************
         file: @@datadir/ddl_recovery.log
   count_read: 0
   total_read: 0 bytes
     avg_read: 0 bytes
  count_write: 114
total_written: 220.17 KiB
    avg_write: 1.93 KiB
        total: 220.17 KiB
    write_pct: 100.00
*************************** 4. row ***************************
         file: @@datadir/ib_logfile0
   count_read: 6
   total_read: 66.50 KiB
     avg_read: 11.08 KiB
  count_write: 43
total_written: 81.00 KiB
    avg_write: 1.88 KiB
        total: 147.50 KiB
    write_pct: 54.92
...

SELECT * FROM sys.x$io_global_by_file_by_bytes\G
...
*************************** 3. row ***************************
         file: /home/ian/sandboxes/msb_10_6_19/data/ddl_recovery.log
   count_read: 0
   total_read: 0
     avg_read: 0.0000
  count_write: 114
total_written: 225459
    avg_write: 1977.7105
        total: 225459
    write_pct: 100.00
*************************** 4. row ***************************
         file: /home/ian/sandboxes/msb_10_6_19/data/ib_logfile0
   count_read: 6
   total_read: 68096
     avg_read: 11349.3333
  count_write: 43
total_written: 82944
    avg_write: 1928.9302
        total: 151040
    write_pct: 54.92
...
```

This page is licensed: CC BY-SA / Gnu FDL

io_global_by_file_by_latency and x$io_global_by_file_by_latency Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `io_global_by_file_by_latency` and `x$io_global_by_file_by_latency` views summarize global I/O consumers to display time waiting for I/O, grouped by file. Rows are sorted by descending total latency by default.

The `io_global_by_file_by_latency` view is intended to be easier for human reading, while the `x$io_global_by_file_by_latency` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
file	File path name.
total	Total number of I/O events for the file.
total_latency	Total wait time of timed I/O events for the file.
count_read	Total number of read I/O events for the file.
read_latency	Total wait time of timed read I/O events for the file.
count_write	Total number of write I/O events for the file.
write_latency	Total wait time of timed write I/O events for the file.
count_misc	Total number of other I/O events for the file.
misc_latency	Total wait time of timed other I/O events for the file.

Example

```
SELECT * FROM sys.io_global_by_file_by_latency\G
*************************** 1. row ***************************
         file: @@datadir/ddl_recovery.log
        total: 222
total_latency: 288.64 ms
   count_read: 0
 read_latency: 0 ps
  count_write: 114
write_latency: 2.59 ms
   count_misc: 108
 misc_latency: 286.05 ms
*************************** 2. row ***************************
         file: @@datadir/ib_logfile0
        total: 95
total_latency: 165.29 ms
   count_read: 6
 read_latency: 61.04 us
  count_write: 43
write_latency: 1.31 ms
   count_misc: 46
 misc_latency: 163.92 ms
...

SELECT * FROM sys.x$io_global_by_file_by_latency\G
*************************** 1. row ***************************
         file: /home/ian/sandboxes/msb_10_6_19/data/ddl_recovery.log
        total: 222
total_latency: 288641408158
   count_read: 0
 read_latency: 0
  count_write: 114
write_latency: 2594925264
   count_misc: 108
 misc_latency: 286046482894
*************************** 2. row ***************************
         file: /home/ian/sandboxes/msb_10_6_19/data/ib_logfile0
        total: 95
total_latency: 165291020006
   count_read: 6
 read_latency: 61040974
  count_write: 43
write_latency: 1310187820
   count_misc: 46
 misc_latency: 163919791212
...
```

This page is licensed: CC BY-SA / Gnu FDL

io_global_by_wait_by_bytes and x$io_global_by_wait_by_bytes Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `io_global_by_wait_by_bytes` and `x$io_global_by_wait_by_bytes` views summarize global I/O consumers, displaying amount of I/O and time waiting for I/O, grouped by event. Rows are sorted by descending total I/O (bytes read and written) by default.

The `io_global_by_wait_by_bytes` view is intended to be easier for human reading, while the `x$io_global_by_wait_by_bytes` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
event_name	I/O event name. The wait/io/file prefix is stripped.
total	Total number of occurrences of the I/O event.
total_latency	Total wait time of timed occurrences of the I/O event
min_latency	Minimum single wait time of timed occurrences of the I/O event.
avg_latency	Average wait time per timed occurrence of the I/O event.
max_latency	Maximum single wait time of timed occurrences of the I/O event.
count_read	Total number of read I/O events for the file.
total_read	Total number of bytes read for the I/O event.
avg_read	Average number of bytes per read for the I/O event.
count_write	Total number of write requests for the I/O event.
total_written	Number of bytes written for the I/O event.
avg_written	Average number of bytes per write for the I/O event.
total_requested	Total number of bytes (read and write) for the I/O event.

Example

```
SELECT * FROM sys.io_global_by_wait_by_bytes\G
*************************** 1. row ***************************
     event_name: innodb/innodb_data_file
          total: 220
  total_latency: 38.96 ms
    min_latency: 0 ps
    avg_latency: 177.09 us
    max_latency: 4.07 ms
     count_read: 174
     total_read: 4.73 MiB
       avg_read: 27.86 KiB
    count_write: 0
  total_written: 0 bytes
    avg_written: 0 bytes
total_requested: 4.73 MiB
*************************** 2. row ***************************
     event_name: aria/MAD
          total: 1107
  total_latency: 18.27 ms
    min_latency: 0 ps
    avg_latency: 16.50 us
    max_latency: 204.97 us
     count_read: 105
     total_read: 840.00 KiB
       avg_read: 8.00 KiB
    count_write: 0
  total_written: 0 bytes
    avg_written: 0 bytes
total_requested: 840.00 KiB

...

SELECT * FROM sys.x$io_global_by_wait_by_bytes\G
*************************** 1. row ***************************
     event_name: innodb/innodb_data_file
          total: 220
  total_latency: 38959722138
    min_latency: 0
    avg_latency: 177089374
    max_latency: 4065566778
     count_read: 174
     total_read: 4964352
       avg_read: 28530.7586
    count_write: 0
  total_written: 0
    avg_written: 0.0000
total_requested: 4964352
*************************** 2. row ***************************
     event_name: aria/MAD
          total: 1107
  total_latency: 18270683624
    min_latency: 0
    avg_latency: 16504546
    max_latency: 204973168
     count_read: 105
     total_read: 860160
       avg_read: 8192.0000
    count_write: 0
  total_written: 0
    avg_written: 0.0000
total_requested: 860160

...
```

This page is licensed: CC BY-SA / Gnu FDL

io_global_by_wait_by_latency and x$io_global_by_wait_by_latency Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `io_global_by_wait_by_latency` and `x$io_global_by_wait_by_latency` views summarize global I/O consumers, displaying I/O and time waiting for I/O, grouped by event. Rows are sorted by descending total latency by default.

The `io_global_by_wait_by_latency` view is intended to be easier for human reading, while the `x$io_global_by_wait_by_latency` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
event_name	I/O event name. The wait/io/file prefix is stripped.
total	Total number of occurrences of the I/O event.
total_latency	Total wait time of timed occurrences of the I/O event.
min_latency	Minimum single wait time of timed occurrences of the I/O event.
avg_latency	Average wait time per timed occurrence of the I/O event.
max_latency	Maximum single wait time of timed occurrences of the I/O event.
count_read	Total number of read request for the I/O event.
total_read	Total number of bytes read for the I/O event.
avg_read	Average number of bytes per read for the I/O event.
count_write	Total number of write requests for the I/O event.
total_written	Number of bytes written for the I/O event.
avg_written	Average number of bytes per write for the I/O event.
total_requested	Total number of bytes (read and write) for the I/O event.

Example

```
SELECT * FROM sys.io_global_by_wait_by_latency\G
*************************** 1. row ***************************
   event_name: sql/global_ddl_log
        total: 223
total_latency: 288.66 ms
  avg_latency: 1.29 ms
  max_latency: 26.07 ms
 read_latency: 0 ps
write_latency: 2.59 ms
 misc_latency: 286.07 ms
   count_read: 0
   total_read: 0 bytes
     avg_read: 0 bytes
  count_write: 114
total_written: 220.17 KiB
  avg_written: 1.93 KiB
*************************** 2. row ***************************
   event_name: innodb/innodb_log_file
        total: 95
total_latency: 165.29 ms
  avg_latency: 1.74 ms
  max_latency: 26.48 ms
 read_latency: 61.04 us
write_latency: 1.31 ms
 misc_latency: 163.92 ms
   count_read: 6
   total_read: 66.50 KiB
     avg_read: 11.08 KiB
  count_write: 43
total_written: 81.00 KiB
  avg_written: 1.88 KiB
...

SELECT * FROM sys.x$io_global_by_wait_by_latency\G
*************************** 1. row ***************************
   event_name: sql/global_ddl_log
        total: 223
total_latency: 288663966666
  avg_latency: 1294456930
  max_latency: 26072142152
 read_latency: 0
write_latency: 2594925264
 misc_latency: 286069041402
   count_read: 0
   total_read: 0
     avg_read: 0.0000
  count_write: 114
total_written: 225459
  avg_written: 1977.7105
*************************** 2. row ***************************
   event_name: innodb/innodb_log_file
        total: 95
total_latency: 165291020006
  avg_latency: 1739905288
  max_latency: 26478157582
 read_latency: 61040974
write_latency: 1310187820
 misc_latency: 163919791212
   count_read: 6
   total_read: 68096
     avg_read: 11349.3333
  count_write: 43
total_written: 82944
  avg_written: 1928.9302
...
```

This page is licensed: CC BY-SA / Gnu FDL

latest_file_io and x$latest_file_io Sys Schema Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `latest_file_io` and `x$latest_file_io` views summarize file I/O activity, grouped by file and thread. Rows are sorted by most recent I/O by default.

The `latest_file_io` view is intended to be easier for human reading, while the `x$latest_file_io` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
thread	Account associated with the thread for foreground threads (port number for TCP/IP connections), or thread name and thread ID for background threads.
total	Total number of occurrences of the I/O event.
file	File path name.
latency	Wait time of the file I/O event.
operation	Type of operation
requested	Number of bytes requested for the file I/O event.

This page is licensed: CC BY-SA / Gnu FDL

memory_by_host_by_current_bytes and x$memory_by_host_by_current_bytes Views

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `memory_by_host_by_current_bytes` and `x$memory_by_host_by_current_bytes` summarize memory use grouped by host. Rows by default are sorted by descending amount of memory used.

The `memory_by_host_by_current_bytes` view is intended to be easier for human reading, while the `x$memory_by_host_by_current_bytes` view provides the data in raw form, intended for tools that process the data.

They contain the following columns:

Column	Description
host	Host from which the client connected. If the HOST column in the underlying Performance Schema table is NULL, rows are assumed to be for background threads, and the background host name is used.
current_count_used	Current number of allocated memory blocks that have not yet been freed for the host.
current_allocated	Current number of allocated bytes that have not yet been freed for the host.

Column	Description
current_avg_alloc	Current number of allocated bytes per memory block for the host.
current_max_alloc	Largest single current memory allocation in bytes for the host.
total_allocated	Total memory allocation in bytes for the host.

This page is licensed: CC BY-SA / Gnu FDL

metrics Sys Schema View

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

The `metrics` view contains global status variables (as found in the Performance Schema global_status Table), InnoDB metrics (as found in the Information Schema INNODB_METRICS Table, current and total memory allocation, based on the Performance Schema memory instrumentation, as well the current time in human readable and Unix timestamp formats.

It contains the following columns:

Column	Description
Variable_name	The name of the metric. One of the VARIABLE_NAME column if a global_status table, the NAME column if an InnoDB metric, or a description for other metrics.
Variable_value	The metric value. One of the VARIABLE_VALUE column if a global status variable, the COUNT column for InnoDB metrics, the related column from the Performance Schema memory_summary_global_by_event_name table, the value of NOW(3) or UNIX_TIMESTAMP(NOW(3)).
Type	Metric type. One of Global Status, InnoDB Metrics - % (with % being the value of the SUBSYSTEM column in the INNODB_METRICS table, Performance Schema or System Time
Enabled	Whether the metric is enabled. Always YES for global status variables and the current time. For InnoDB metrics, YES only if the STATUS column of the INNODB_METRICS table, otherwise NO. For memory metrics: YES, NO or PARTIAL (for metrics where not all memory/% instruments are enabled). Performance Schema memory instruments are always enabled)

Example

```
SELECT * FROM sys.metrics\G
*************************** 1. row ***************************
 Variable_name: aborted_clients
Variable_value: 0
          Type: Global Status
       Enabled: YES
*************************** 2. row ***************************
 Variable_name: aborted_connects
Variable_value: 0
          Type: Global Status
       Enabled: YES

...

*************************** 578. row ***************************
 Variable_name: trx_undo_slots_used
Variable_value: 0
          Type: InnoDB Metrics - transaction
       Enabled: YES
*************************** 579. row ***************************
 Variable_name: NOW()
Variable_value: 2024-09-09 16:16:08.745
          Type: System Time
       Enabled: YES
*************************** 580. row ***************************
 Variable_name: UNIX_TIMESTAMP()
Variable_value: 1725891368.745
          Type: System Time
       Enabled: YES
```

This page is licensed: CC BY-SA / Gnu FDL

privileges_by_table_by_level Sys Schema View

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

> **Note:**
>
> This view is available from MariaDB 11.4.

Description

Shows granted privileges broken down by table on which they allow access and level on which they were granted.

For example, if a user `x` has `SELECT` privilege granted `ON db.*`, this view will list all tables in the `db` schema with the user `x` having `SELECT` privilege on them. This is different from INFORMATION_SCHEMA.TABLE_PRIVILEGES, which only lists privileges granted on the table level.

Column	Description
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
GRANTEE	Account name that was granted the privilege.
PRIVILEGE	Privilege, such as SELECT or DROP.
LEVEL	Privilege level, such as GLOBAL or SCHEMA.

Example

```
SELECT * FROM sys.privileges_by_table_by_level;
+--------------+------------+--------------------------+----------------+--------+
| TABLE_SCHEMA | TABLE_NAME | GRANTEE                  | PRIVILEGE      | LEVEL  |
+--------------+------------+--------------------------+----------------+--------+
...
| test         | t2         | 'root'@'localhost'       | SELECT         | GLOBAL |
| test         | t1         | 'root'@'localhost'       | SELECT         | GLOBAL |
| test         | t3         | 'root'@'localhost'       | SELECT         | GLOBAL |
| test         | t2         | 'root'@'localhost'       | INSERT         | GLOBAL |
| test         | t1         | 'root'@'localhost'       | INSERT         | GLOBAL |
| test         | t3         | 'root'@'localhost'       | INSERT         | GLOBAL |
| test         | t2         | 'root'@'localhost'       | UPDATE         | GLOBAL |
| test         | t1         | 'root'@'localhost'       | UPDATE         | GLOBAL |
| test         | t3         | 'root'@'localhost'       | UPDATE         | GLOBAL |
| test         | t2         | 'root'@'localhost'       | DELETE         | GLOBAL |
| test         | t1         | 'root'@'localhost'       | DELETE         | GLOBAL |
| test         | t3         | 'root'@'localhost'       | DELETE         | GLOBAL |
| test         | t2         | 'root'@'localhost'       | CREATE         | GLOBAL |
| test         | t1         | 'root'@'localhost'       | CREATE         | GLOBAL |
| test         | t3         | 'root'@'localhost'       | CREATE         | GLOBAL |
| test         | t2         | 'root'@'localhost'       | DROP           | GLOBAL |
| test         | t1         | 'root'@'localhost'       | DROP           | GLOBAL |
| test         | t3         | 'root'@'localhost'       | DROP           | GLOBAL |
| test         | t2         | 'root'@'localhost'       | REFERENCES     | GLOBAL |
| test         | t1         | 'root'@'localhost'       | REFERENCES     | GLOBAL |
| test         | t3         | 'root'@'localhost'       | REFERENCES     | GLOBAL |
| test         | t2         | 'root'@'localhost'       | INDEX          | GLOBAL |
| test         | t1         | 'root'@'localhost'       | INDEX          | GLOBAL |
| test         | t3         | 'root'@'localhost'       | INDEX          | GLOBAL |
| test         | t2         | 'root'@'localhost'       | ALTER          | GLOBAL |
| test         | t1         | 'root'@'localhost'       | ALTER          | GLOBAL |
| test         | t3         | 'root'@'localhost'       | ALTER          | GLOBAL |
| test         | t2         | 'root'@'localhost'       | SHOW VIEW      | GLOBAL |
| test         | t1         | 'root'@'localhost'       | SHOW VIEW      | GLOBAL |
| test         | t3         | 'root'@'localhost'       | SHOW VIEW      | GLOBAL |
| test         | t2         | 'root'@'localhost'       | TRIGGER        | GLOBAL |
| test         | t1         | 'root'@'localhost'       | TRIGGER        | GLOBAL |
| test         | t3         | 'root'@'localhost'       | TRIGGER        | GLOBAL |
| test         | t2         | 'root'@'localhost'       | DELETE HISTORY | GLOBAL |
| test         | t1         | 'root'@'localhost'       | DELETE HISTORY | GLOBAL |
| test         | t3         | 'root'@'localhost'       | DELETE HISTORY | GLOBAL |
| test         | t2         | 'PUBLIC'@''              | SELECT         | SCHEMA |
| test         | t1         | 'PUBLIC'@''              | SELECT         | SCHEMA |
| test         | t3         | 'PUBLIC'@''              | SELECT         | SCHEMA |
| test         | t2         | 'PUBLIC'@''              | INSERT         | SCHEMA |
| test         | t1         | 'PUBLIC'@''              | INSERT         | SCHEMA |
| test         | t3         | 'PUBLIC'@''              | INSERT         | SCHEMA |
| test         | t2         | 'PUBLIC'@''              | UPDATE         | SCHEMA |
| test         | t1         | 'PUBLIC'@''              | UPDATE         | SCHEMA |
| test         | t3         | 'PUBLIC'@''              | UPDATE         | SCHEMA |
| test         | t2         | 'PUBLIC'@''              | DELETE         | SCHEMA |
| test         | t1         | 'PUBLIC'@''              | DELETE         | SCHEMA |
| test         | t3         | 'PUBLIC'@''              | DELETE         | SCHEMA |
| test         | t2         | 'PUBLIC'@''              | CREATE         | SCHEMA |
| test         | t1         | 'PUBLIC'@''              | CREATE         | SCHEMA |
| test         | t3         | 'PUBLIC'@''              | CREATE         | SCHEMA |
| test         | t2         | 'PUBLIC'@''              | DROP           | SCHEMA |
| test         | t1         | 'PUBLIC'@''              | DROP           | SCHEMA |
| test         | t3         | 'PUBLIC'@''              | DROP           | SCHEMA |
| test         | t2         | 'PUBLIC'@''              | REFERENCES     | SCHEMA |
| test         | t1         | 'PUBLIC'@''              | REFERENCES     | SCHEMA |
| test         | t3         | 'PUBLIC'@''              | REFERENCES     | SCHEMA |
| test         | t2         | 'PUBLIC'@''              | INDEX          | SCHEMA |
| test         | t1         | 'PUBLIC'@''              | INDEX          | SCHEMA |
| test         | t3         | 'PUBLIC'@''              | INDEX          | SCHEMA |
| test         | t2         | 'PUBLIC'@''              | ALTER          | SCHEMA |
| test         | t1         | 'PUBLIC'@''              | ALTER          | SCHEMA |
```

```
| test         | t3         | 'PUBLIC'@''               | ALTER          | SCHEMA |
| test         | t2         | 'PUBLIC'@''               | SHOW VIEW      | SCHEMA |
| test         | t1         | 'PUBLIC'@''               | SHOW VIEW      | SCHEMA |
| test         | t3         | 'PUBLIC'@''               | SHOW VIEW      | SCHEMA |
| test         | t2         | 'PUBLIC'@''               | TRIGGER        | SCHEMA |
| test         | t1         | 'PUBLIC'@''               | TRIGGER        | SCHEMA |
| test         | t3         | 'PUBLIC'@''               | TRIGGER        | SCHEMA |
| test         | t2         | 'PUBLIC'@''               | DELETE HISTORY | SCHEMA |
| test         | t1         | 'PUBLIC'@''               | DELETE HISTORY | SCHEMA |
| test         | t3         | 'PUBLIC'@''               | DELETE HISTORY | SCHEMA |
+--------------+------------+---------------------------+----------------+--------+
```

See Also

- GRANT (description of the privileges and how to grant them)
- INFORMATION_SCHEMA.TABLE_PRIVILEGES
- MDEV-24486

This page is licensed: CC BY-SA / Gnu FDL

schema_auto_increment_columns Sys Schema View

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

Information about AUTO_INCREMENT columns, sorted by descending usage ratio and maximum column value. Does not include columns in the `mysql`, `sys`, `information_schema` and `performance_schema` schemas.

Contains the following columns:

Column	Description
table_schema	Schema name containing the table with the auto_increment attribute.
table_name	Table containing the auto_increment attribute.
column_name	Name of the column containing the auto_increment attribute.
data_type	Data type of the auto_increment column, for example tinyint.
column_type	Data type of the auto_increment column, plus additional information for example tinyint(3) unsigned.
is_signed	1 if the column is signed,0 if not.
is_unsigned	1 if the column is unsigned,0 if it is.
max_value	Maximum possible value for the column, for example 255 for an unsigned tinyint.
auto_increment	Current auto_increment value for the column.
auto_increment_ratio	Ratio of used to maximum value for the auto_increment column.

Example

```
CREATE OR REPLACE TABLE animals (
    id TINYINT NOT NULL AUTO_INCREMENT,
    name CHAR(30) NOT NULL,
    PRIMARY KEY (id)
);

 INSERT INTO animals (name) VALUES
    ('dog'),('cat'),('penguin'),
    ('fox'),('whale'),('ostrich');

SELECT * FROM sys.schema_auto_increment_columns\G
*************************** 1. row ***************************
        table_schema: test
          table_name: animals
         column_name: id
           data_type: tinyint
         column_type: tinyint(4)
           is_signed: 1
         is_unsigned: 0
           max_value: 127
      auto_increment: 7
auto_increment_ratio: 0.0551
```

This page is licensed: CC BY-SA / Gnu FDL

schema_object_overview Sys Schema View

> **Note:**
>
> Sys Schema views are available from MariaDB 10.6.

Description

A count of the number of objects within each schema, sorted by schema and object.

Contains the following columns:

Column	Description
db	Schema name
object_type	Object name
count	Count of the number of objects

Example

```
SELECT * FROM sys.schema_object_overview;
+--------------------+---------------+-------+
| db                 | object_type   | count |
+--------------------+---------------+-------+
| information_schema | SYSTEM VIEW   |    79 |
| mysql              | BASE TABLE    |    30 |
| mysql              | INDEX (BTREE) |    76 |
| mysql              | PROCEDURE     |     2 |
| mysql              | VIEW          |     1 |
| performance_schema | BASE TABLE    |    81 |
| sys                | BASE TABLE    |     1 |
| sys                | FUNCTION      |    22 |
| sys                | INDEX (BTREE) |     1 |
| sys                | PROCEDURE     |    26 |
| sys                | VIEW          |   100 |
+--------------------+---------------+-------+
```

This page is licensed: CC BY-SA / Gnu FDL

The mysql Database Tables

See mysql.column_stats Table.

The mysql.column_stats table stores engine-independent column statistics, such as histograms, used by the optimizer to improve query execution plans.

See mysql.columns_priv Table.

The mysql.columns_priv table records column-level privileges granted to users, detailing specific access rights for individual columns.

See mysql.db Table.

The mysql.db table stores database-level privileges, determining which users have access to specific databases and what actions they can perform.

See mysql.event Table.

The mysql.event table contains the definitions and scheduling information for events created with the CREATE EVENT statement.

See mysql.func Table.

The mysql.func table registers user-defined functions (UDFs), storing their names and the shared library files containing their code.

See mysql.general_log Table.

The mysql.general_log table captures a record of all SQL statements received by the server when general query logging is enabled and written to tables.

See mysql.global_priv Table.

The mysql.global_priv table stores global privileges and account properties for all users, replacing the older mysql.user table structure.

See mysql.gtid_slave_pos Table.

The mysql.gtid_slave_pos table tracks the Global Transaction ID (GTID) of the last applied transaction on a replica to ensure replication consistency.

See mysql.help_category Table.

The mysql.help_category table stores category information for the server-side help system, organizing help topics into a hierarchy.

See mysql.help_keyword Table.

The mysql.help_keyword table maps keywords to help topics, facilitating keyword-based searches within the MariaDB help system.

See mysql.help_relation Table.

The mysql.help_relation table links help keywords to help topics, defining the structure of the server-side help system.

See mysql.help_topic Table.

The mysql.help_topic table stores the detailed content of help topics, including descriptions and examples displayed by the HELP command.

See mysql.index_stats Table.

The mysql.index_stats table stores engine-independent index statistics, such as cardinality, used by the optimizer to plan query execution.

See mysql.innodb_index_stats.

The mysql.innodb_index_stats table stores persistent index statistics for InnoDB, allowing optimizer plans to remain stable across restarts.

See mysql.innodb_table_stats.

The mysql.innodb_table_stats table holds persistent table-level statistics for InnoDB, such as row counts, used for query optimization.

See mysql.password_reuse_check_history Table.

This table stores a history of password hashes to enforce security policies regarding password reuse when the relevant plugin is enabled.

See mysql.plugin Table.

The mysql.plugin table records information about installed server plugins, ensuring they are reloaded automatically upon server startup.

See mysql.proc Table.

The mysql.proc table stores the definitions, body, and metadata for stored procedures and functions created on the server.

See mysql.procs_priv Table.

The mysql.procs_priv table records privileges granted to users specifically for executing or altering stored procedures and functions.

See mysql.proxies_priv Table.

The mysql.proxies_priv table manages proxy user privileges, defining which accounts are authorized to proxy as other users.

See mysql.roles_mapping Table.

The mysql.roles_mapping table manages role assignments, linking user accounts to the roles they have been granted.

See mysql.servers Table.

The mysql.servers table stores connection information for remote servers, used by the FEDERATED and Spider storage engines.

See mysql.slow_log Table.

The mysql.slow_log table records details of queries that exceed the long_query_time threshold when slow query logging to tables is enabled.

See mysql.table_stats Table.

The mysql.table_stats table stores engine-independent statistics about tables, such as row counts, to assist the optimizer.

See mysql.tables_priv Table.

The mysql.tables_priv table records table-level privileges granted to users, specifying which actions they can perform on specific tables.

See mysql.time_zone Table.

The mysql.time_zone table assigns a unique ID to each time zone supported by the server, linking to other time zone system tables.

See mysql.time_zone_leap_second Table.

The mysql.time_zone_leap_second table lists leap second corrections to be applied to specific time zones.

See mysql.time_zone_name Table.

The mysql.time_zone_name table maps human-readable time zone names (e.g., “Europe/Berlin”) to their internal time zone IDs.

See mysql.time_zone_transition Table.

The mysql.time_zone_transition table defines the exact times at which daylight saving time or other time zone transitions occur.

See mysql.time_zone_transition_type Table.

The mysql.time_zone_transition_type table describes the properties of time zone transitions, such as the offset and abbreviation.

See mysql.transaction_registry Table.

The mysql.transaction_registry table is used by system-versioned tables to track transaction IDs and their commit timestamps.

See mysql.user Table.

Complete mysql.user table reference: mysql.global_priv view, Host/User identifiers, plugin/authentication_string fields, and GRANT/CREATE USER integration.

See Spider mysql Database Tables.

Explore Spider-related tables within the mysql database. These system tables store crucial configuration and metadata for the Spider storage engine, essential for distributed deployments.

This page is licensed: CC BY-SA / Gnu FDL

mysql.column_stats Table

The `mysql.column_stats` table is one of three tables storing data used for Engine-independent table statistics. The others are mysql.table_stats and mysql.index_stats.

It is populated when the ANALYZE TABLE statement is run, although not by default. See Collecting Statistics with the ANALYZE TABLE Statement for details.

Note that statistics for blob and text columns are not collected. If explicitly specified, a warning is returned.

It is possible to manually update the table and, unlike most system tables, there are some scenarios where this could be useful. See Manual updates to statistics tables for details.

This table uses the Aria storage engine.

The `mysql.column_stats` table contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	varchar(64)	NO	PRI	NULL	Database the table is in.
table_name	varchar(64)	NO	PRI	NULL	Table name.
column_name	varchar(64)	NO	PRI	NULL	Name of the column.
min_value	varbinary(255)	YES		NULL	
max_value	varbinary(255)	YES		NULL	
nulls_ratio	decimal(12,4)	YES		NULL	Fraction of NULL values (0- no NULLs, 0.5 - half values are NULLs, 1 - all values are NULLs).
avg_length	decimal(12,4)	YES		NULL	Average length of column value, in bytes. Counted as if one ran SELECT AVG(LENGTH(col)). This doesn't count NULL bytes, assumes endspace removal for CHAR(n), etc.
avg_frequency	decimal(12,4)	YES		NULL	Average number of records with the same value
hist_size	tinyint(3) unsigned	YES		NULL	Histogram size in bytes, from 0-255, or, from MariaDB 10.8, number of buckets if the histogram type is JSON_HB.
hist_type	enum('SINGLE_PREC_HB', 'DOUBLE_PREC_HB') (from MariaDB 10.8) enum('SINGLE_PREC_HB', 'DOUBLE_PREC_HB','JSON_HB') (until MariaDB 10.7)	YES		NULL	Histogram type. See the histogram_type system variable.
histogram	blob (from MariaDB 10.7) varbinary(255) (until MariaDB 10.7)	YES		NULL	

This page is licensed: CC BY-SA / Gnu FDL

mysql.columns_priv Table

The `mysql.columns_priv` table contains information about column-level privileges. The table can be queried and although it is possible to directly update it, it is best to use GRANT for setting privileges.

Note that the MariaDB privileges occur at many levels. A user may be granted a privilege at the column level, but may still not have permission on a table level, for example. See privileges for a more complete view of the MariaDB privilege system.

The INFORMATION_SCHEMA.COLUMN_PRIVILEGES table derives its contents from `mysql.columns_priv`.

This table uses the Aria storage engine.

The `mysql.columns_priv` table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User, Db , Table_name andColumn_name makes up the unique identifier for this record.
Db	char(64)	NO	PRI		Database name (together with User, Host , Table_name andColumn_name makes up the unique identifier for this record.
User	char(80)	NO	PRI		User (together with Host, Db , Table_name andColumn_name makes up the unique identifier for this record.
Table_name	char(64)	NO	PRI		Table name (together with User, Db , Host andColumn_name makes up the unique identifier for this record.
Column_name	char(64)	NO	PRI		Column name (together with User, Db , Table_name andHost makes up the unique identifier for this record.
Timestamp	timestamp	NO		CURRENT_TIMESTAMP	
Column_priv	set('Select', 'Insert', 'Update', 'References')	NO			The privilege type. See Column Privileges for details.

The Acl_column_grants status variable indicates how many rows the `mysql.columns_priv` table contains.

This page is licensed: CC BY-SA / Gnu FDL

mysql.db Table

The `mysql.db` table contains information about database-level privileges. The table can be queried and although it is possible to directly update it, it is best to use GRANT for setting privileges.

Note that the MariaDB privileges occur at many levels. A user may not be granted a privilege at the database level, but may still have permission on a table level, for example. See privileges for a more complete view of the MariaDB privilege system.

This table uses the Aria storage engine.

The `mysql.db` table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User and Db makes up the unique identifier for this record. Until MariaDB 5.5, if the host field was blank, the corresponding record in the mysql.host table would be examined. From MariaDB 10.0, a blank host field is the same as the % wildcard.
Db	char(64)	NO	PRI		Database (together with User and Host makes up the unique identifier for this record.
User	char(80)	NO	PRI		User (together with Host and Db makes up the unique identifier for this record.
Select_priv	enum('N','Y')	NO		N	Can perform SELECT statements.
Insert_priv	enum('N','Y')	NO		N	Can perform INSERT statements.

Field	Type	Null	Key	Default	Description
Update_priv	enum('N','Y')	NO		N	Can perform UPDATE statements.
Delete_priv	enum('N','Y')	NO		N	Can perform DELETE statements.
Create_priv	enum('N','Y')	NO		N	Can CREATE TABLE's.
Drop_priv	enum('N','Y')	NO		N	Can DROP DATABASE's or DROP TABLE's.
Grant_priv	enum('N','Y')	NO		N	User can grant privileges they possess.
References_priv	enum('N','Y')	NO		N	Unused
Index_priv	enum('N','Y')	NO		N	Can create an index on a table using the CREATE INDEX statement. Without the INDEX privilege, user can still create indexes when creating a table using the CREATE TABLE statement if the user has have the CREATE privilege, and user can create indexes using the ALTER TABLE statement if they have the ALTER privilege.
Alter_priv	enum('N','Y')	NO		N	Can perform ALTER TABLE statements.
Create_tmp_table_priv	enum('N','Y')	NO		N	Can create temporary tables with the CREATE TEMPORARY TABLE statement.
Lock_tables_priv	enum('N','Y')	NO		N	Acquire explicit locks using the LOCK TABLES statement; user also needs to have the SELECT privilege on a table in order to lock it.
Create_view_priv	enum('N','Y')	NO		N	Can create a view using the CREATE_VIEW statement.
Show_view_priv	enum('N','Y')	NO		N	Can show the CREATE VIEW statement to create a view using the SHOW CREATE VIEW statement.
Create_routine_priv	enum('N','Y')	NO		N	Can create stored programs using the CREATE PROCEDURE and CREATE FUNCTION statements.
Alter_routine_priv	enum('N','Y')	NO		N	Can change the characteristics of a stored function using the ALTER FUNCTION statement.
Execute_priv	enum('N','Y')	NO		N	Can execute stored procedure or functions.
Event_priv	enum('N','Y')	NO		N	Create, drop and alter events.

Field	Type	Null	Key	Default	Description
Trigger_priv	enum('N','Y')	NO		N	Can execute triggers associated with tables the user updates, execute the CREATE TRIGGER and DROP TRIGGER statements.
Delete_history_priv	enum('N','Y')	NO		N	Can delete rows created through system versioning.

The Acl_database_grants status variable indicates how many rows the `mysql.db` table contains.

This page is licensed: CC BY-SA / Gnu FDL

mysql.event Table

The `mysql.event` table contains information about MariaDB events. Similar information can be obtained by viewing the INFORMATION_SCHEMA.EVENTS table, or with the SHOW EVENTS and SHOW CREATE EVENT statements.

The table is upgraded live, and there is no need to restart the server if the table has changed.

This table uses the Aria storage engine.

The `mysql.event` table contains the following fields:

Field	Type	Null	Key	Default	Description
db	char(64)	NO	PRI		
name	char(64)	NO	PRI		
body	longblob	NO		NULL	
definer	char(141)	NO			
execute_at	datetime	YES		NULL	
interval_value	int(11)	YES		NULL	
interval_field	enum('YEAR', 'QUARTER', 'MONTH', 'DAY', 'HOUR', 'MINUTE', 'WEEK', 'SECOND', 'MICROSECOND', 'YEAR_MONTH', 'DAY_HOUR', 'DAY_MINUTE', 'DAY_SECOND', 'HOUR_MINUTE', 'HOUR_SECOND', 'MINUTE_SECOND', 'DAY_MICROSECOND', 'HOUR_MICROSECOND', 'MINUTE_MICROSECOND', 'SECOND_MICROSECOND')	YES		NULL	

Field	Type	Null	Key	Default	Description
created	timestamp	NO		CURRENT_TIMESTAMP	
modified	timestamp	NO		0000-00-00 00:00:00	
last_executed	datetime	YES		NULL	
starts	datetime	YES		NULL	
ends	datetime	YES		NULL	
status	enum('ENABLED', 'DISABLED', 'SLAVESIDE_DISABLED')	NO		ENABLED	Current status of the event, one of enabled, disabled, or disabled on the slaveside.
on_completion	enum('DROP','PRESERVE')	NO		DROP	
sql_mode	set('REAL_AS_FLOAT', 'PIPES_AS_CONCAT', 'ANSI_QUOTES', 'IGNORE_SPACE', 'IGNORE_BAD_TABLE_OPTIONS', 'ONLY_FULL_GROUP_BY', 'NO_UNSIGNED_SUBTRACTION', 'NO_DIR_IN_CREATE', 'POSTGRESQL', 'ORACLE', 'MSSQL', 'DB2', 'MAXDB', 'NO_KEY_OPTIONS', 'NO_TABLE_OPTIONS', 'NO_FIELD_OPTIONS', 'MYSQL323', 'MYSQL40', 'ANSI', 'NO_AUTO_VALUE_ON_ZERO', 'NO_BACKSLASH_ESCAPES', 'STRICT_TRANS_TABLES', 'STRICT_ALL_TABLES', 'NO_ZERO_IN_DATE', 'NO_ZERO_DATE', 'INVALID_DATES', 'ERROR_FOR_DIVISION_BY_ZERO', 'TRADITIONAL', 'NO_AUTO_CREATE_USER', 'HIGH_NOT_PRECEDENCE', 'NO_ENGINE_SUBSTITUTION', 'PAD_CHAR_TO_FULL_LENGTH')	NO			The SQL_MODE at the time the event was created.
comment	char(64)	NO			
originator	int(10) unsigned	NO		NULL	

Field	Type	Null	Key	Default	Description
time_zone	char(64)	NO		SYSTEM	
character_set_client	char(32)	YES		NULL	
collation_connection	char(32)	YES		NULL	
db_collation	char(32)	YES		NULL	
body_utf8	longblob	YES		NULL	

This page is licensed: CC BY-SA / Gnu FDL

mysql.func Table

The `mysql.func` table stores information about user-defined functions (UDFs) created with the CREATE FUNCTION UDF statement.

This table uses the Aria storage engine.

The `mysql.func` table contains the following fields:

Field	Type	Null	Key	Default	Description
name	char(64)	NO	PRI		UDF name
ret	tinyint(1)	NO		0	
dl	char(128)	NO			Shared library name
type	enum('function','aggregate')	NO		NULL	Type, either function or aggregate. Aggregate functions are summary functions such as SUM() and AVG().

Example

```
SELECT * FROM mysql.func;
+------------------------------+-----+--------------+-----------+
| name                         | ret | dl           | type      |
+------------------------------+-----+--------------+-----------+
| spider_direct_sql            |   2 | ha_spider.so | function  |
| spider_bg_direct_sql         |   2 | ha_spider.so | aggregate |
| spider_ping_table            |   2 | ha_spider.so | function  |
| spider_copy_tables           |   2 | ha_spider.so | function  |
| spider_flush_table_mon_cache |   2 | ha_spider.so | function  |
+------------------------------+-----+--------------+-----------+
```

This page is licensed: CC BY-SA / Gnu FDL

mysql.general_log Table

The `mysql.general_log` table stores the contents of the General Query Log if general logging is active and the output is being written to table (see Writing logs into tables).

It contains the following fields:

Field	Type	Null	Key	Default	Description
event_time	timestamp(6)	NO		CURRENT_TIMESTAMP(6)	Time the query was executed.
user_host	mediumtext	NO		NULL	User and host combination.
thread_id	int(11)	NO		NULL	Thread id.
server_id	int(10) unsigned	NO		NULL	Server id.
command_type	varchar(64)	NO		NULL	Type of command.
argument	mediumtext	NO		NULL	Full query.

Example

```
SELECT * FROM mysql.general_log\G
*************************** 1. row ***************************
  event_time: 2014-11-11 08:40:04.117177
   user_host: root[root] @ localhost []
   thread_id: 74
   server_id: 1
command_type: Query
    argument: SELECT * FROM test.s
*************************** 2. row ***************************
  event_time: 2014-11-11 08:40:10.501131
   user_host: root[root] @ localhost []
   thread_id: 74
   server_id: 1
command_type: Query
    argument: SELECT * FROM mysql.general_log
...
```

This page is licensed: CC BY-SA / Gnu FDL

mysql.global_priv Table

The `mysql.global_priv` table contains information about users that have permission to access the MariaDB server, and their global privileges.

Note that the MariaDB privileges occur at many levels. A user may not be granted `CREATE` privilege at the user level, but may still have `CREATE` permission on certain tables or databases, for example. See privileges for a more complete view of the MariaDB privilege system.

The `mysql.global_priv` table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User makes up the unique identifier for this account).
User	char(80)	NO	PRI		User (together with Host makes up the unique identifier for this account).
Priv	longtext	NO			Global privileges, granted to the account and other account properties

From MariaDB 10.5.2, in order to help the server understand which version a privilege record was written by, the `priv` field contains a new JSON field, `version_id` (MDEV-21704).

From MariaDB 13.1, the `Priv` field also holds a `denies` array, recording the negative privileges created with DENY. Denies are stored here for every privilege level, not in `mysql.db`, `mysql.tables_priv`, or `mysql.columns_priv`. See The denies Field.

Examples

```
SELECT * FROM mysql.global_priv;
+-----------+-------------+-------------------------------------------------------------------------------
-------------------------------------------------------+
| Host      | User        | Priv
|
+-----------+-------------+-------------------------------------------------------------------------------
-------------------------------------------------------+
| localhost | root        | {"access":
1844674407370955161 5,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E74
754DA6682D04747"} |
| 127.%     | msandbox    |
{"access":1073740799,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E74
754DA6682D04747"}            |
| localhost | msandbox    |
{"access":1073740799,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E74
754DA6682D04747"}            |
| localhost | msandbox_rw |
{"access":487487,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E74754D
A6682D04747"}                |
| 127.%     | msandbox_rw |
{"access":487487,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E74754D
A6682D04747"}                |
| 127.%     | msandbox_ro |
{"access":262145,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E74754D
A6682D04747"}                |
| localhost | msandbox_ro |
{"access":262145,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E74754D
A6682D04747"}                |
| 127.%     | rsandbox    |
{"access":524288,"plugin":"mysql_native_password","authentication_string":"*B07EB15A2E7BD9620DAE47B194D5B
9DBA14377AD"}                |
+-----------+-------------+-------------------------------------------------------------------------------
-------------------------------------------------------+
```

Readable format:

```
SELECT CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)) FROM mysql.global_priv;

+-----------------------------------------------------------------------------------------+
| CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv))                                    |
+-----------------------------------------------------------------------------------------+
| root@localhost => {
    "access": 18446744073709551615,
    "plugin": "mysql_native_password",
    "authentication_string": "*6C387FC3893DBA1E3BA155E74754DA6682D04747"
} |
| msandbox@127.% => {
    "access": 1073740799,
    "plugin": "mysql_native_password",
    "authentication_string": "*6C387FC3893DBA1E3BA155E74754DA6682D04747"
}           |
+-----------------------------------------------------------------------------------------+
```

A particular user:

```
SELECT CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)) FROM mysql.global_priv
  WHERE user='marijn';
+-----------------------------------------------------------------------------------------+
| CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv))                                    |
+-----------------------------------------------------------------------------------------+
| marijn@localhost => {
    "access": 0,
    "plugin": "mysql_native_password",
    "authentication_string": "",
    "account_locked": true,
    "password_last_changed": 1558017158
} |
+-----------------------------------------------------------------------------------------+
```

From MariaDB 10.5.2:

```
GRANT FILE ON *.* TO user1@localhost;
SELECT Host, User, JSON_DETAILED(Priv) FROM mysql.global_priv WHERE user='user1'\G

*************************** 1. row ***************************
               Host: localhost
               User: user1
JSON_DETAILED(Priv): {
    "access": 512,
    "plugin": "mysql_native_password",
    "authentication_string": "",
    "password_last_changed": 1581070979,
    "version_id": 100502
}
```

Mapping the `access` Field Values to Grants

The `access` field contains the grants of the user which can be mapped to individual grants with the following table. The most up-to-date information can be found in the `sql/privilege.h` file in the source code.

Grant	Bit
SELECT	(1UL << 0)
INSERT	(1UL << 1)
UPDATE	(1UL << 2)
DELETE	(1UL << 3)
CREATE	(1UL << 4)
DROP	(1UL << 5)

Grant	Bit
RELOAD	(1UL << 6)
SHUTDOWN	(1UL << 7)
PROCESS	(1UL << 8)
FILE	(1UL << 9)
GRANT	(1UL << 10)
REFERENCES	(1UL << 11)
INDEX	(1UL << 12)
ALTER	(1UL << 13)
SHOW_DB	(1UL << 14)
SUPER	(1UL << 15)
CREATE_TMP	(1UL << 16)
LOCK_TABLES	(1UL << 17)
EXECUTE	(1UL << 18)
REPL_SLAVE	(1UL << 19)
BINLOG_MONITOR	(1UL << 20)
CREATE_VIEW	(1UL << 21)
SHOW_VIEW	(1UL << 22)
CREATE_PROC	(1UL << 23)
ALTER_PROC	(1UL << 24)
CREATE_USER	(1UL << 25)
EVENT	(1UL << 26)
TRIGGER	(1UL << 27)
CREATE_TABLESPACE	(1UL << 28)
DELETE_HISTORY	(1UL << 29)
SET_USER	(1UL << 30)
FEDERATED_ADMIN	(1UL << 31)
CONNECTION_ADMIN	(1ULL << 32)
READ_ONLY_ADMIN	(1ULL << 33)
REPL_SLAVE_ADMIN	(1ULL << 34)
REPL_MASTER_ADMIN	(1ULL << 35)
BINLOG_ADMIN	(1ULL << 36)
BINLOG_REPLAY	(1ULL << 37)
SLAVE_MONITOR	(1ULL << 38)
SHOW_CREATE_ROUTINE	(1ULL << 39)

The `denies` Field

From MariaDB 13.1, the `denies` array records the privileges that DENY blocks for an account or role. Each element describes one privilege level:

```
DENY SELECT ON hr.* TO user1@localhost;
DENY SELECT (salary) ON hr.staff TO user1@localhost;
SELECT JSON_PRETTY(JSON_EXTRACT(Priv, '$.denies')) FROM mysql.global_priv
  WHERE User='user1'\G

*************************** 1. row ***************************
JSON_PRETTY(JSON_EXTRACT(Priv, '$.denies')): [
    {
        "type": "db",
        "db": "hr",
        "bits": 1
    },
    {
        "type": "column",
        "db": "hr",
        "table": "staff",
        "column": "salary",
        "bits": 1
    }
]
```

The fields of an element are:

Field	Description
`type`	The privilege level: `global`, `db`, `table`, `column`, `function`, `procedure`, `package`, or `package body`.
`db`	Database name. Absent for `global`.
`table`	Table name, for `table` and `column` entries.
`column`	Column name, for `column` entries.
`function`, `procedure`, `package`, `package body`	Routine name, named after the entry's `type`.
`bits`	The denied privileges, using the same bit values as the `access` field.

An account with no denies has either no `denies` key or an empty array.

> **Warning:**
>
> Do not edit the `denies` array with `UPDATE`. A malformed entry is skipped when privileges are loaded, with a `Malformed DENY entry in mysql.global_priv` warning written to the error log, which leaves the privilege allowed. Use `DENY` and `REVOKE DENY` instead.

This page is licensed: CC BY-SA / Gnu FDL

mysql.gtid_slave_pos Table

The `mysql.gtid_slave_pos` table is used in replication by replica servers to keep track of their current position (the global transaction ID of the last transaction applied). Using the table allows the replica to maintain a consistent value for the gtid_slave_pos system variable across server restarts. See Global Transaction ID.

You should never attempt to modify the table directly. If you do need to change the global `gtid_slave_pos` value, use `SET GLOBAL gtid_slave_pos = ...` instead.

The table is updated with the new position as part of each transaction committed during replication. This makes it preferable that the table is using the same storage engine as the tables otherwise being modified in the transaction, since otherwise a multi-engine transaction is needed that can reduce performance.

Multiple versions of this table are supported, each using a different storage engine. This is selected with the gtid_pos_auto_engines option, by giving a comma-separated list of engine names. The server will then on-demand

create an extra version of the table using the appropriate storage engine, and select the table version using the same engine as the rest of the transaction, avoiding multi-engine transactions.

When `gtid_pos_auto_engines=innodb,rocksdb` is set, the tables `mysql.gtid_slave_pos_InnoDB` and `mysql.gtid_slave_pos_RocksDB` are created and used, if needed. If there is no match to the storage engine, the default `mysql.gtid_slave_pos` table is used; this also happens if non-transactional updates (like MyISAM) are replicated, since there is then no active transaction at the time of the `mysql.gtid_slave_pos` table update.

The default `mysql.gtid_slave_pos` table is initially created using the default storage engine set for the server (which itself defaults to InnoDB). If the application load is primarily non-transactional MyISAM or Aria tables, it can be beneficial to change the storage engine to avoid including an InnoDB update with every operation:

```
ALTER TABLE mysql.gtid_slave_pos ENGINE=MyISAM;
```

The `mysql.gtid_slave_pos` table should not be changed manually in any other way. It is preferable to use the `gtid_pos_auto_engines` server variable to get the GTID position updates to use the RocksDB storage engine.

Note that, for scalability reasons, the automatic creation of a new `mysql.gtid_slave_posXXX` table happens asynchronously when the first transaction with the new storage engine is committed. So the very first few transactions will update the old version of the table, until the new version is created and available.

The table `mysql.gtid_slave_pos` contains the following fields:

Field	Type	Null	Key	Default	Description
domain_id	int(10) unsigned	NO	PRI	NULL	Domain id (see Global Transaction ID domain ID.
sub_id	bigint(20) unsigned	NO	PRI	NULL	This field enables multiple parallel transactions within same domain_id to update this table without contention. At any instant, the replication state corresponds to records with largest sub_id for each domain_id.
server_id	int(10) unsigned	NO		NULL	Server id.
seq_no	bigint(20) unsigned	NO		NULL	Sequence number, an integer that is monotonically increasing for each new event group logged into the binlog.

Some status variables are available to monitor the use of the different `gtid_slave_pos` table versions:

Transactions_gtid_foreign_engine

Number of replicated transactions where the update of the `gtid_slave_pos` table had to choose a storage engine that did not otherwise participate in the transaction. This can indicate that setting `gtid_pos_auto_engines` might be useful.

Rpl_transactions_multi_engine

Number of replicated transactions that involved changes in multiple (transactional) storage engines, before considering the update of `gtid_slave_pos`. These are transactions that were already cross-engine, independent of the GTID position update introduced by replication

Transactions_multi_engine

Number of transactions that changed data in multiple (transactional) storage engines. If this is significantly larger than `Rpl_transactions_multi_engine`, it indicates that setting `gtid_pos_auto_engines` could reduce the need for cross-engine transactions.

This page is licensed: CC BY-SA / Gnu FDL

mysql.help_category Table

`mysql.help_category` is one of the four tables used by the HELP command. It is populated when the server is installed by the `fill_help_tables.sql` script. The other help tables are help_relation, help_topic and help_keyword.

This table uses the Aria storage engine.

The `mysql.help_category` table contains the following fields:

Field	Type	Null	Key	Default	Description
help_category_id	smallint(5) unsigned	NO	PRI	NULL	
name	char(64)	NO	UNI	NULL	
parent_category_id	smallint(5) unsigned	YES		NULL	
url	char(128)	NO		NULL	

Example

```
SELECT * FROM help_category;
+------------------+----------------------------------------------+--------------------+-----+
| help_category_id | name                                         | parent_category_id | url |
+------------------+----------------------------------------------+--------------------+-----+
|                1 | Geographic                                   |                  0 |     |
|                2 | Polygon properties                           |                 34 |     |
|                3 | WKT                                          |                 34 |     |
|                4 | Numeric Functions                            |                 38 |     |
|                5 | Plugins                                      |                 35 |     |
|                6 | MBR                                          |                 34 |     |
|                7 | Control flow functions                       |                 38 |     |
|                8 | Transactions                                 |                 35 |     |
|                9 | Help Metadata                                |                 35 |     |
|               10 | Account Management                           |                 35 |     |
|               11 | Point properties                             |                 34 |     |
|               12 | Encryption Functions                         |                 38 |     |
|               13 | LineString properties                        |                 34 |     |
|               14 | Miscellaneous Functions                      |                 38 |     |
|               15 | Logical operators                            |                 38 |     |
|               16 | Functions and Modifiers for Use with GROUP BY |                 35 |     |
|               17 | Information Functions                        |                 38 |     |
|               18 | Comparison operators                         |                 38 |     |
|               19 | Bit Functions                                |                 38 |     |
|               20 | Table Maintenance                            |                 35 |     |
|               21 | User-Defined Functions                       |                 35 |     |
|               22 | Data Types                                   |                 35 |     |
|               23 | Compound Statements                          |                 35 |     |
|               24 | Geometry constructors                        |                 34 |     |
|               25 | GeometryCollection properties                |                  1 |     |
|               26 | Administration                               |                 35 |     |
|               27 | Data Manipulation                            |                 35 |     |
|               28 | Utility                                      |                 35 |     |
|               29 | Language Structure                           |                 35 |     |
|               30 | Geometry relations                           |                 34 |     |
|               31 | Date and Time Functions                      |                 38 |     |
|               32 | WKB                                          |                 34 |     |
|               33 | Procedures                                   |                 35 |     |
|               34 | Geographic Features                          |                 35 |     |
|               35 | Contents                                     |                  0 |     |
|               36 | Geometry properties                          |                 34 |     |
|               37 | String Functions                             |                 38 |     |
|               38 | Functions                                    |                 35 |     |
|               39 | Data Definition                              |                 35 |     |
+------------------+----------------------------------------------+--------------------+-----+
```

This page is licensed: CC BY-SA / Gnu FDL

mysql.help_keyword Table

`mysql.help_keyword` is one of the four tables used by the HELP command. It is populated when the server is installed by the `fill_help_tables.sql` script. The other help tables are help_relation, help_category and help_topic.

This table uses the Aria storage engine.

The `mysql.help_keyword` table contains the following fields:

Field	Type	Null	Key	Default	Description
help_keyword_id	int(10) unsigned	NO	PRI	NULL	
name	char(64)	NO	UNI	NULL	

Example

```
SELECT * FROM help_keyword;
+-----------------+-------------------------------+
| help_keyword_id | name                          |
+-----------------+-------------------------------+
|               0 | JOIN                          |
|               1 | HOST                          |
|               2 | REPEAT                        |
|               3 | SERIALIZABLE                  |
|               4 | REPLACE                       |
|               5 | AT                            |
|               6 | SCHEDULE                      |
|               7 | RETURNS                       |
|               8 | STARTS                        |
|               9 | MASTER_SSL_CA                 |
|              10 | NCHAR                         |
|              11 | COLUMNS                       |
|              12 | COMPLETION                    |
...
```

This page is licensed: CC BY-SA / Gnu FDL

mysql.help_relation Table

`mysql.help_relation` is one of the four tables used by the HELP command. It is populated when the server is installed by the `fill_help_tables.sql` script. The other help tables are help_topic, help_category and help_keyword.

This table uses the Aria storage engine.

The `mysql.help_relation` table contains the following fields:

Field	Type	Null	Key	Default	Description
help_topic_id	int(10) unsigned	NO	PRI	NULL	
help_keyword_id	int(10) unsigned	NO	PRI	NULL	

Example

```
...
|           106 |             456 |
|           463 |             456 |
|           468 |             456 |
|           463 |             457 |
|           194 |             458 |
|           478 |             458 |
|           374 |             459 |
|           459 |             459 |
|            39 |             460 |
|            58 |             460 |
|           185 |             460 |
|           264 |             460 |
|           269 |             460 |
|           209 |             461 |
|           468 |             461 |
|           201 |             462 |
|           468 |             463 |
+---------------+-----------------+
```

This page is licensed: CC BY-SA / Gnu FDL

mysql.help_topic Table

`mysql.help_topic` is one of the four tables used by the HELP command. It is populated when the server is installed by the `fill_help_tables.sql` script. The other help tables are help_relation, help_category and help_keyword.

This table uses the Aria storage engine.

The `mysql.help_topic` table contains the following fields:

Field	Type	Null	Key	Default	Description
help_topic_id	int(10) unsigned	NO	PRI	NULL	
name	char(64)	NO	UNI	NULL	
help_category_id	smallint(5) unsigned	NO		NULL	
description	text	NO		NULL	
example	text	NO		NULL	
url	char(128)	NO		NULL	

Example

```
SELECT * FROM help_topic\G;
...
*************************** 704. row ***************************
   help_topic_id: 692
           name: JSON_DEPTH
help_category_id: 41
     description: JSON functions were added in MariaDB 10.2.3.

Syntax
------
JSON_DEPTH(json_doc)

Description
-----------
Returns the maximum depth of the given JSON document, or
NULL if the argument is null. An error will occur if the
argument is an invalid JSON document.
Scalar values or empty arrays or objects have a depth of 1.
Arrays or objects that are not empty but contain only
elements or member values of depth 1 will have a depth of 2.
In other cases, the depth will be greater than 2.

Examples
--------
SELECT JSON_DEPTH('[]'), JSON_DEPTH('true'),
JSON_DEPTH('{}');
+------------------+--------------------+------------------+
| JSON_DEPTH('[]') | JSON_DEPTH('true') |
JSON_DEPTH('{}') |
+------------------+--------------------+------------------+
| 1 | 1 | 1 |
+------------------+--------------------+------------------+

SELECT JSON_DEPTH('[1, 2, 3]'), JSON_DEPTH('[[], {},
[]]');
+-------------------------+----------------------------+
| JSON_DEPTH('[1, 2, 3]') | JSON_DEPTH('[[], {}, []]') |
+-------------------------+----------------------------+
| 2 | 2 |
+-------------------------+----------------------------+

SELECT JSON_DEPTH('[1, 2, [3, 4, 5, 6], 7]');
+---------------------------------------+
| JSON_DEPTH('[1, 2, [3, 4, 5, 6], 7]') |
+---------------------------------------+
| 3 |
+---------------------------------------+

URL: https://mariadb.com/kb/en/json_depth/
         example:
             url: https://mariadb.com/kb/en/json_depth/
```

This page is licensed: CC BY-SA / Gnu FDL

mysql.index_stats Table

The `mysql.index_stats` table is one of three tables storing data used for Engine-independent table statistics. The others are mysql.column_stats and mysql.table_stats.

It is populated when the ANALYZE TABLE statement is run, although not by default. See Collecting Statistics with the ANALYZE TABLE Statement for details.

It is possible to manually update the table and, unlike most system tables, there are some scenarios where this could be useful. See Manual updates to statistics tables for details.

This table uses the Aria storage engine.

The `mysql.index_stats` table contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	varchar(64)	NO	PRI	NULL	Database the table is in.
table_name	varchar(64)	NO	PRI	NULL	Table name
index_name	varchar(64)	NO	PRI	NULL	Name of the index
prefix_arity	int(11) unsigned	NO	PRI	NULL	Index prefix length. 1 for the first keypart, 2 for the first two, and so on. InnoDB's extended keys are supported.
avg_frequency	decimal(12,4)	YES		NULL	Average number of records one will find for given values of (keypart1, keypart2, ..), provided the values will be found in the table.

It is possible to manually update the table. See Manual updates to statistics tables for details.

This page is licensed: CC BY-SA / Gnu FDL

mysql.innodb_index_stats

The `mysql.innodb_index_stats` table stores data related to particular InnoDB Persistent Statistics, and contains multiple rows for each index.

This table, along with the related mysql.innodb_table_stats table, can be manually updated in order to force or test differing query optimization plans. After updating, FLUSH TABLE innodb_index_stats is required to load the changes.

`mysql.innodb_index_stats` is not replicated, although any ANALYZE TABLE statements on the table will be by default..

It contains the following fields:

Field	Type	Null	Key	Default	Description
database_name	varchar(64)	NO	PRI	NULL	Database name.
table_name	varchar(64)	NO	PRI	NULL	Table, partition or subpartition name.
index_name	varchar(64)	NO	PRI	NULL	Index name.
last_update	timestamp	NO		current_timestamp()	Time that this row was last updated.
stat_name	varchar(64)	NO	PRI	NULL	Statistic name.
stat_value	bigint(20) unsigned	NO		NULL	Estimated statistic value.
sample_size	bigint(20) unsigned	YES		NULL	Number of pages sampled for the estimated statistic value.
stat_description	varchar(1024)	NO		NULL	Statistic description.

Example

```
SELECT * FROM mysql.innodb_index_stats\G
*************************** 1. row ***************************
   database_name: mysql
      table_name: gtid_slave_pos
      index_name: PRIMARY
     last_update: 2017-08-19 20:38:34
       stat_name: n_diff_pfx01
      stat_value: 0
     sample_size: 1
stat_description: domain_id
*************************** 2. row ***************************
   database_name: mysql
      table_name: gtid_slave_pos
      index_name: PRIMARY
     last_update: 2017-08-19 20:38:34
       stat_name: n_diff_pfx02
      stat_value: 0
     sample_size: 1
stat_description: domain_id,sub_id
*************************** 3. row ***************************
   database_name: mysql
      table_name: gtid_slave_pos
      index_name: PRIMARY
     last_update: 2017-08-19 20:38:34
       stat_name: n_leaf_pages
      stat_value: 1
     sample_size: NULL
stat_description: Number of leaf pages in the index
*************************** 4. row ***************************
   database_name: mysql
      table_name: gtid_slave_pos
      index_name: PRIMARY
     last_update: 2017-08-19 20:38:34
       stat_name: size
      stat_value: 1
     sample_size: NULL
stat_description: Number of pages in the index
*************************** 5. row ***************************
   database_name: test
      table_name: ft
      index_name: FTS_DOC_ID_INDEX
     last_update: 2017-09-15 12:58:39
       stat_name: n_diff_pfx01
      stat_value: 0
     sample_size: 1
stat_description: FTS_DOC_ID
*************************** 6. row ***************************
   database_name: test
      table_name: ft
      index_name: FTS_DOC_ID_INDEX
     last_update: 2017-09-15 12:58:39
       stat_name: n_leaf_pages
      stat_value: 1
     sample_size: NULL
stat_description: Number of leaf pages in the index
...
```

See Also

- InnoDB Persistent Statistics
- mysql.innodb_table_stats
- ANALYZE TABLE

This page is licensed: CC BY-SA / Gnu FDL

mysql.innodb_table_stats

The `mysql.innodb_table_stats` table stores data related to InnoDB Persistent Statistics, and contains one row per table.

This table, along with the related mysql.innodb_index_stats table, can be manually updated in order to force or test differing query optimization plans. After updating, FLUSH TABLE innodb_table_stats is required to load the changes.

`mysql.innodb_table_stats` is not replicated, although any ANALYZE TABLE statements on the table will be by default.

It contains the following fields:

Field	Type	Null	Key	Default	Description
database_name	varchar(64)	NO	PRI	NULL	Database name.
table_name	varchar(64)	NO	PRI	NULL	Table, partition or subpartition name.
last_update	timestamp	NO		current_timestamp()	Time that this row was last updated.
n_rows	bigint(20) unsigned	NO		NULL	Number of rows in the table.
clustered_index_size	bigint(20) unsigned	NO		NULL	Size, in pages, of the primary index.
sum_of_other_index_sizes	bigint(20) unsigned	NO		NULL	Size, in pages, of non-primary indexes.

Example

```
SELECT * FROM mysql.innodb_table_stats\G
*************************** 1. row ***************************
           database_name: mysql
              table_name: gtid_slave_pos
             last_update: 2017-08-19 20:38:34
                 n_rows: 0
    clustered_index_size: 1
sum_of_other_index_sizes: 0
*************************** 2. row ***************************
           database_name: test
              table_name: ft
             last_update: 2017-09-15 12:58:39
                 n_rows: 0
    clustered_index_size: 1
sum_of_other_index_sizes: 2
...
```

See Also

- InnoDB Persistent Statistics
- mysql.innodb_index_stats
- ANALYZE TABLE

This page is licensed: CC BY-SA / Gnu FDL

mysql.password_reuse_check_history Table

> **Note:**
>
> This table is available from MariaDB 10.7.

The mysql.password_reuse_check_history Table is installed as part of the password_reuse_check plugin.

The `mysql.password_reuse_check_history` table stores old passwords, so that when a user sets a new password, it can be checked for purposes of preventing password reuse.

It contains the following fields:

Field	Type	Null	Key	Default	Description
hash	binary(64)	NO	PRI	NULL	
time	timestamp	NO	MUL	current_timestamp()	

This page is licensed: CC BY-SA / Gnu FDL

mysql.plugin Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.plugin` table can be queried to get information about installed plugins.

This table only contains information about plugins that have been installed via the following methods:

- The INSTALL SONAME statement.
- The INSTALL PLUGIN statement.
- The mariadb-plugin utility.

This table does not contain information about:

- Built-in plugins.
- Plugins loaded with the –plugin-load-add option.
- Plugins loaded with the –plugin-load option.

This table only contains enough information to reload the plugin when the server is restarted, which means it only contains the plugin name and the plugin library.

This table uses the Aria storage engine.

The `mysql.plugin` table contains the following fields:

Field	Type	Null	Key	Default	Description
name	varchar(64)	NO	PRI		Plugin name.
dl	varchar(128)	NO			Name of the plugin library.

Example

```
SELECT * FROM mysql.plugin;
+---------------------------+-----------------------+
| name                      | dl                    |
+---------------------------+-----------------------+
| spider                    | ha_spider.so          |
| spider_alloc_mem          | ha_spider.so          |
| METADATA_LOCK_INFO        | metadata_lock_info.so |
| OQGRAPH                   | ha_oqgraph.so         |
| QUERY_RESPONSE_TIME       | query_response_time.so |
| QUERY_RESPONSE_TIME_AUDIT | query_response_time.so |
| LOCALES                   | locales.so            |
| sequence                  | ha_sequence.so        |
+---------------------------+-----------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

mysql.proc Table

The `mysql.proc` table contains information about stored procedures and stored functions. It contains similar information to that stored in the INFORMATION SCHEMA.ROUTINES table.

This table uses the Aria storage engine.

The `mysql.proc` table contains the following fields:

Field	Type	Null	Key	Default	Description
db	char(64)	NO	PRI		Database name.
name	char(64)	NO	PRI		Routine name.
type	enum('FUNCTION','PROCEDURE','PACKAGE', 'PACKAGE BODY')	NO	PRI	NULL	Whether stored procedure, stored function or a package or package body.
specific_name	char(64)	NO			
language	enum('SQL')	NO		SQL	Always SQL.
sql_data_access	enum('CONTAINS_SQL', 'NO_SQL', 'READS_SQL_DATA', 'MODIFIES_SQL_DATA')	NO		CONTAINS_SQL	
is_deterministic	enum('YES','NO')	NO		NO	Whether the routine is deterministic (can produce only one result for a given list of parameters) or not.
security_type	enum('INVOKER','DEFINER')	NO		DEFINER	INVOKER or DEFINER. Indicates which user's privileges apply to this routine.
param_list	blob	NO		NULL	List of parameters.
returns	longblob	NO		NULL	What the routine returns.

Field	Type	Null	Key	Default	Description
body	longblob	NO		NULL	Definition of the routine.
definer	char(141)	NO			If the security_type is DEFINER, this value indicates which user defined this routine.
created	timestamp	NO		CURRENT_TIMESTAMP	Date and time the routine was created.
modified	timestamp	NO		0000-00-00 00:00:00	Date and time the routine was modified.
sql_mode	set('REAL_AS_FLOAT', 'PIPES_AS_CONCAT', 'ANSI_QUOTES', 'IGNORE_SPACE', 'IGNORE_BAD_TABLE_OPTIONS', 'ONLY_FULL_GROUP_BY', 'NO_UNSIGNED_SUBTRACTION', 'NO_DIR_IN_CREATE', 'POSTGRESQL', 'ORACLE', 'MSSQL', 'DB2', 'MAXDB', 'NO_KEY_OPTIONS', 'NO_TABLE_OPTIONS', 'NO_FIELD_OPTIONS', 'MYSQL323', 'MYSQL40', 'ANSI', 'NO_AUTO_VALUE_ON_ZERO', 'NO_BACKSLASH_ESCAPES', 'STRICT_TRANS_TABLES', 'STRICT_ALL_TABLES', 'NO_ZERO_IN_DATE', 'NO_ZERO_DATE', 'INVALID_DATES', 'ERROR_FOR_DIVISION_BY_ZERO', 'TRADITIONAL', 'NO_AUTO_CREATE_USER', 'HIGH_NOT_PRECEDENCE', 'NO_ENGINE_SUBSTITUTION', 'PAD_CHAR_TO_FULL_LENGTH', 'EMPTY_STRING_IS_NULL', 'SIMULTANEOUS_ASSIGNMENT')	NO			The SQL_MODE at the time the routine was created.
comment	text	NO		NULL	Comment associated with the routine.
character_set_client	char(32)	YES		NULL	The character set used by the client that created the routine.
collation_connection	char(32)	YES		NULL	The collation (and character set) used by the connection that created the routine.

Field	Type	Null	Key	Default	Description
db_collation	char(32)	YES		NULL	The default collation (and character set) for the database, at the time the routine was created.
body_utf8	longblob	YES		NULL	Definition of the routine in utf8.
aggregate	enum('NONE', 'GROUP')	NO		NONE	
Field	Type	Null	Key	Default	Description

See Also

- Stored Procedure Internals
- MySQL to MariaDB migration: handling privilege table differences when using mysqldump

This page is licensed: CC BY-SA / Gnu FDL

mysql.procs_priv Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.procs_priv` table contains information about stored procedure and stored function privileges. See CREATE PROCEDURE and CREATE FUNCTION on creating these.

The INFORMATION_SCHEMA.ROUTINES table derives its contents from `mysql.procs_priv`.

This table uses the Aria storage engine.

The `mysql.procs_priv` table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with Db, User, Routine_name and Routine_type makes up the unique identifier for this record).
Db	char(64)	NO	PRI		Database (together with Host, User, Routine_name and Routine_type makes up the unique identifier for this record).
User	char(80)	NO	PRI		User (together with Host, Db, Routine_name and Routine_type makes up the unique identifier for this record).
Routine_name	char(64)	NO	PRI		Routine_name (together with Host, Db User and Routine_type makes up the unique identifier for this record).
Routine_type	enum('FUNCTION','PROCEDURE', 'PACKAGE', 'PACKAGE BODY')	NO	PRI	NULL	Whether the routine is a stored procedure, stored function, package or package body.
Grantor	char(141)	NO	MUL		

Field	Type	Null	Key	Default	Description
Proc_priv	set('Execute','Alter Routine','Grant')	NO			The routine privilege. See Function Privileges and Procedure Privileges for details.
Timestamp	timestamp	NO		CURRENT_TIMESTAMP	

The Acl_function_grants status variable indicates how many rows the `mysql.columns_priv` table contains with the `FUNCTION` routine type.

The Acl_procedure_grants status variable indicates how many rows the `mysql.columns_priv` table contains with the `PROCEDURE` routine type.

This page is licensed: CC BY-SA / Gnu FDL

mysql.proxies_priv Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.proxies_priv` table contains information about proxy privileges. The table can be queried and although it is possible to directly update it, it is best to use GRANT for setting privileges.

This table uses the Aria storage engine.

The `mysql.proxies_priv` table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		
User	char(80)	NO	PRI		
Proxied_host	char(60)	NO	PRI		
Proxied_user	char(80)	NO	PRI		
With_grant	tinyint(1)	NO		0	
Grantor	char(141)	NO	MUL		
Timestamp	timestamp	NO		CURRENT_TIMESTAMP	

The Acl_proxy_users status variable indicates how many rows the `mysql.proxies_priv` table contains.

This page is licensed: CC BY-SA / Gnu FDL

mysql.roles_mapping Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.roles_mapping` table contains information about mariaDB roles.

This table uses the Aria storage engine.

The `mysql.roles_mapping` table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User and Role makes up the unique identifier for this record.
User	char(80)	NO	PRI		User (together with Host and Role makes up the unique identifier for this record.
Role	char(80)	NO	PRI		Role (together with Host and User makes up the unique identifier for this record.
Admin_option	enum('N','Y')	NO		N	Whether the role can be granted (see the CREATE ROLE WITH ADMIN clause).

The Acl_role_grants status variable indicates how many rows the `mysql.roles_mapping` table contains.

This page is licensed: CC BY-SA / Gnu FDL

mysql.servers Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.servers` table contains information about servers as used by the Spider, FEDERATED or FederatedX, Connect storage engines.

The contents are modified by the CREATE SERVER, ALTER SERVER and DROP SERVER statements.

This table uses the Aria storage engine.

The `mysql.servers` table contains the following fields:

Field	Type	Null	Key	Default	Description
Server_name	char(64)	NO	PRI		
Host	char(64)	NO			
Db	char(64)	NO			
Username	char(80)	NO			
Password	char(64)	NO			
Port	int(4)	NO		0	
Socket	char(64)	NO			
Wrapper	char(64)	NO			mysql or mariadb
Owner	char(64)	NO			

Example

```
SELECT * FROM mysql.servers\G
*************************** 1. row ***************************
Server_name: s
       Host: 192.168.1.106
         Db: test
   Username: Remote
   Password:
       Port: 0
     Socket:
    Wrapper: mariadb
      Owner:
```

This page is licensed: CC BY-SA / Gnu FDL

mysql.slow_log Table

System tables should not normally be edited directly. Use the related SQL statements instead.

The `mysql.slow_log` table stores the contents of the Slow Query Log if slow logging is active and the output is being written to table (see Writing logs into tables).

It contains the following fields:

Field	Type	Null	Key	Default	Description
start_time	timestamp(6)	NO		CURRENT_TIMESTAMP(6)	Time the query began.

Field	Type	Null	Key	Default	Description
user_host	mediumtext	NO		NULL	User and host combination.
query_time	time(6)	NO		NULL	Total time the query took to execute.
lock_time	time(6)	NO		NULL	Total time the query was locked.
rows_sent	int(11)	NO		NULL	Number of rows sent.
rows_examined	int(11)	NO		NULL	Number of rows examined.
db	varchar(512)	NO		NULL	Default database.
last_insert_id	int(11)	NO		NULL	last_insert_id.
insert_id	int(11)	NO		NULL	Insert id.
server_id	int(10) unsigned	NO		NULL	The server's id.
sql_text	mediumtext	NO		NULL	Full query.
thread_id	bigint(21) unsigned	NO		NULL	Thread id.
rows_affected	int(11)	NO		NULL	Number of rows affected by an UPDATE or DELETE (from MariaDB 10.1.2)

Example

```
SELECT * FROM mysql.slow_log\G
...
*************************** 2. row ***************************
    start_time: 2014-11-11 07:56:28.721519
     user_host: root[root] @ localhost []
    query_time: 00:00:12.000215
     lock_time: 00:00:00.000000
     rows_sent: 1
 rows_examined: 0
            db: test
last_insert_id: 0
     insert_id: 0
     server_id: 1
      sql_text: SELECT SLEEP(12)
     thread_id: 74
...
```

This page is licensed: CC BY-SA / Gnu FDL

mysql.table_stats Table

The `mysql.table_stats` table is one of three tables storing data used for Engine-independent table statistics. The others are mysql.column_stats and mysql.index_stats.

It is populated when the ANALYZE TABLE statement is run, although not by default. See Collecting Statistics with the ANALYZE TABLE Statement for details.

It is possible to manually update the table and, unlike most system tables, there are some scenarios where this could be useful. See Manual updates to statistics tables for details.

This table uses the Aria storage engine.

The `mysql.table_stats` table contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	varchar(64)	NO	PRI	NULL	Database the table is in .
table_name	varchar(64)	NO	PRI	NULL	Table name.
cardinality	bigint(21) unsigned	YES		NULL	Number of records in the table.

This page is licensed: CC BY-SA / Gnu FDL

mysql.tables_priv Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.tables_priv` table contains information about table-level privileges. The table can be queried and although it is possible to directly update it, it is best to use GRANT for setting privileges.

Note that the MariaDB privileges occur at many levels. A user may be granted a privilege at the table level, but may still not have permission on a database level, for example. See privileges for a more complete view of the MariaDB privilege system.

The INFORMATION_SCHEMA.TABLE_PRIVILEGES table derives its contents from `mysql.tables_priv`.

This table uses the Aria storage engine.

The `mysql.tables_priv` table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User, Db and Table_namemakes up the unique identifier for this record.
Db	char(64)	NO	PRI		Database (together with User, Host and Table_namemakes up the unique identifier for this record.
User	char(80)	NO	PRI		User (together with Host, Db and Table_namemakes up the unique identifier for this record.
Table_name	char(64)	NO	PRI		Table name (together with User, Db and Tablemakes up the unique identifier for this record.
Grantor	char(141)	NO	MUL		
Timestamp	timestamp	NO		CURRENT_TIMESTAMP	
Table_priv	set('Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter', 'Create View', 'Show view', 'Trigger', 'Delete versioning rows')	NO			The table privilege type. See Table Privileges for details.
Column_priv	set('Select', 'Insert', 'Update', 'References')	NO			The column privilege type. See Column Privileges for details.

The Acl_table_grants status variable indicates how many rows the `mysql.tables_priv` table contains.

This page is licensed: CC BY-SA / Gnu FDL

mysql.time_zone Table

The `mysql.time_zone` table is one of the mysql system tables that can contain time zone information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the mysql time zone tables using the mariadb-tzinfo-to-sql utility. See Time Zones for details.

This table uses the Aria storage engine.

The `mysql.time_zone` table contains the following fields:

Field	Type	Null	Key	Default	Description
Time_zone_id	int(10) unsigned	NO	PRI	NULL	ID field, auto_increments.
Use_leap_seconds	enum('Y','N')	NO		N	Whether or not leap seconds are used.

Example

```
SELECT * FROM mysql.time_zone;
+--------------+------------------+
| Time_zone_id | Use_leap_seconds |
+--------------+------------------+
|            1 | N                |
|            2 | N                |
|            3 | N                |
|            4 | N                |
|            5 | N                |
|            6 | N                |
|            7 | N                |
|            8 | N                |
|            9 | N                |
|           10 | N                |
...
+--------------+------------------+
```

See Also

- mysql.time_zone_leap_second table
- mysql.time_zone_name table
- mysql.time_zone_transition table
- mysql.time_zone_transition_type table

This page is licensed: CC BY-SA / Gnu FDL

mysql.time_zone_leap_second Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.time_zone_leap_second` table is one of the mysql system tables that can contain time zone information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the mysql time zone tables using the mariadb-tzinfo-to-sql utility. See Time Zones for details.

This table uses the Aria storage engine.

The `mysql.time_zone_leap_second` table contains the following fields:

Field	Type	Null	Key	Default	Description
Transition_time	bigint(20)	NO	PRI	NULL	
Correction	int(11)	NO		NULL	

See Also

- mysql.time_zone table
- mysql.time_zone_name table
- mysql.time_zone_transition table
- mysql.time_zone_transition_type table

This page is licensed: CC BY-SA / Gnu FDL

mysql.time_zone_name Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.time_zone_name` table is one of the mysql system tables that can contain time zone information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the mysql time zone tables using the mariadb-tzinfo-to-sql utility. See Time Zones for details.

This table uses the Aria storage engine.

The `mysql.time_zone_name` table contains the following fields:

Field	Type	Null	Key	Default	Description
Name	char(64)	NO	PRI	NULL	Name of the time zone.
Time_zone_id	int(10) unsigned	NO	PRI	NULL	ID field, auto_increments.

Example

```
SELECT * FROM mysql.time_zone_name;
+--------------------+--------------+
| Name               | Time_zone_id |
+--------------------+--------------+
| Africa/Abidjan     |            1 |
| Africa/Accra       |            2 |
| Africa/Addis_Ababa |            3 |
| Africa/Algiers     |            4 |
| Africa/Asmara      |            5 |
| Africa/Asmera      |            6 |
| Africa/Bamako      |            7 |
| Africa/Bangui      |            8 |
| Africa/Banjul      |            9 |
| Africa/Bissau      |           10 |
...
+--------------------+--------------+
```

See Also

- mysql.time_zone table
- mysql.time_zone_leap_second table
- mysql.time_zone_transition table
- mysql.time_zone_transition_type table

This page is licensed: CC BY-SA / Gnu FDL

mysql.time_zone_transition Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.time_zone_transition` table is one of the `mysql` system tables that can contain time zone information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the `mysql` time zone tables using the mariadb-tzinfo-to-sql utility. See Time Zones for details.

This table uses the Aria storage engine.

The `mysql.time_zone_transition` table contains the following fields:

Field	Type	Null	Key	Default	Description
Time_zone_id	int(10) unsigned	NO	PRI	NULL	
Transition_time	bigint(20)	NO	PRI	NULL	
Transition_type_id	int(10) unsigned	NO		NULL	

Example

```
SELECT * FROM mysql.time_zone_transition;
+--------------+-----------------+--------------------+
| Time_zone_id | Transition_time | Transition_type_id |
+--------------+-----------------+--------------------+
|            1 |     -1830383032 |                  1 |
|            2 |     -1640995148 |                  2 |
|            2 |     -1556841600 |                  1 |
|            2 |     -1546388400 |                  2 |
|            2 |     -1525305600 |                  1 |
|            2 |     -1514852400 |                  2 |
|            2 |     -1493769600 |                  1 |
|            2 |     -1483316400 |                  2 |
|            2 |     -1462233600 |                  1 |
|            2 |     -1451780400 |                  2 |
...
+--------------+-----------------+--------------------+
```

See Also

- mysql.time_zone table
- mysql.time_zone_leap_second table
- mysql.time_zone_name table
- mysql.time_zone_transition_type table

This page is licensed: CC BY-SA / Gnu FDL

mysql.time_zone_transition_type Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.time_zone_transition_type` table is one of the `mysql` system tables that can contain time zone information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the `mysql` time zone tables using the mariadb-tzinfo-to-sql utility. See Time Zones for details.

This table uses the Aria storage engine.

The `mysql.time_zone_transition_type` table contains the following fields:

Field	Type	Null	Key	Default	Description
Time_zone_id	int(10) unsigned	NO	PRI	NULL	
Transition_type_id	int(10) unsigned	NO	PRI	NULL	
Offset	int(11)	NO		0	
Is_DST	tinyint(3) unsigned	NO		0	
Abbreviation	char(8)	NO			

Example

```
SELECT * FROM mysql.time_zone_transition_type;
+--------------+--------------------+--------+--------+--------------+
| Time_zone_id | Transition_type_id | Offset | Is_DST | Abbreviation |
+--------------+--------------------+--------+--------+--------------+
|            1 |                  0 |   -968 |      0 | LMT          |
|            1 |                  1 |      0 |      0 | GMT          |
|            2 |                  0 |    -52 |      0 | LMT          |
|            2 |                  1 |   1200 |      1 | GHST         |
|            2 |                  2 |      0 |      0 | GMT          |
|            3 |                  0 |   8836 |      0 | LMT          |
|            3 |                  1 |  10800 |      0 | EAT          |
|            3 |                  2 |   9000 |      0 | BEAT         |
|            3 |                  3 |   9900 |      0 | BEAUT        |
|            3 |                  4 |  10800 |      0 | EAT          |
...
+--------------+--------------------+--------+--------+--------------+
```

See Also

- mysql.time_zone table
- mysql.time_zone_leap_second table
- mysql.time_zone_name table
- mysql.time_zone_transition table

This page is licensed: CC BY-SA / Gnu FDL

mysql.transaction_registry Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The `mysql.transaction_registry` table is part of system-versioned tables. It is used for transaction-precise versioning, and contains the following fields:

Field	Type	Null	Key	Default	Description
transaction_id	bigint(20) unsigned	NO	Primary	NULL	
commit_id	bigint(20) unsigned	NO	Unique	NULL	
begin_timestamp	timestamp(6)	NO	Multiple	0000-00-00 00:00:00.000000	Timestamp when the transaction began (BEGIN statement), however see MDEV-16024.
commit	timestamp(6)	NO	Multiple	0000-00-00 00:00:00.000000	Timestamp when the transaction was committed.
isolation_level	enum('READ-UNCOMMITTED','READ-COMMITTED','REPEATABLE-READ','SERIALIZABLE')	NO		NULL	Transaction isolation level.

This page is licensed: CC BY-SA / Gnu FDL

mysql.user Table

1. PARTITION BY SYSTEM_TIME is not supported for transaction-precise system-versioned tables. Attempting

The mysql.global_priv table has replaced the `mysql.user` table, and `mysql.user` should be considered obsolete. It is now a view into `mysql.global_priv` created for compatibility with older applications and monitoring scripts. New

tools are supposed to use `INFORMATION_SCHEMA` tables. The dedicated `mariadb.sys` user is created as the definer of the view. Previously, `root` was the definer, which resulted in privilege problems when this username was changed (MDEV-19650).

The `mysql.user` table contains information about users that have permission to access the MariaDB server, and their global privileges. The table can be queried and although it is possible to directly update it, it is best to use GRANT and CREATE USER for adding users and privileges.

Note that the MariaDB privileges occur at many levels. A user may not be granted `create` privilege at the user level, but may still have `create` permission on certain tables or databases, for example. See privileges for a more complete view of the MariaDB privilege system.

The `mysql.user` table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User makes up the unique identifier for this account.
User	char(80)	NO	PRI		User (together with Host makes up the unique identifier for this account.
Password	longtext	NO			Hashed password, generated by the PASSWORD() function.
Select_priv	enum('N','Y')	NO		N	Can perform SELECT statements.
Insert_priv	enum('N','Y')	NO		N	Can perform INSERT statements.
Update_priv	enum('N','Y')	NO		N	Can perform UPDATE statements.
Delete_priv	enum('N','Y')	NO		N	Can perform DELETE statements.
Create_priv	enum('N','Y')	NO		N	Can CREATE DATABASE's or CREATE TABLE's.
Drop_priv	enum('N','Y')	NO		N	Can DROP DATABASE's or DROP TABLE's.
Reload_priv	enum('N','Y')	NO		N	Can execute FLUSH statements or equivalent mariadb-admin commands.
Shutdown_priv	enum('N','Y')	NO		N	Can shut down the server with SHUTDOWN or mariadb-admin shutdown.
Process_priv	enum('N','Y')	NO		N	Can show information about active processes, via SHOW PROCESSLIST or mariadb-admin processlist.

Field	Type	Null	Key	Default	Description
File_priv	enum('N','Y')	NO		N	Read and write files on the server, using statements like LOAD DATA INFILE or functions like LOAD_FILE(). Also needed to create CONNECT outward tables. MariaDB server must have permission to access those files.
Grant_priv	enum('N','Y')	NO		N	User can grant privileges they possess.
References_priv	enum('N','Y')	NO		N	Unused
Index_priv	enum('N','Y')	NO		N	Can create an index on a table using the CREATE INDEX statement. Without the INDEX privilege, user can still create indexes when creating a table using the CREATE TABLE statement if the user has have the CREATE privilege, and user can create indexes using the ALTER TABLE statement if they have the ALTER privilege.
Alter_priv	enum('N','Y')	NO		N	Can perform ALTER TABLE statements.
Show_db_priv	enum('N','Y')	NO		N	Can list all databases using the SHOW DATABASES statement. Without the SHOW DATABASES privilege, user can still issue the SHOW DATABASES statement, but it will only list databases containing tables on which they have privileges.
Super_priv	enum('N','Y')	NO		N	Can execute superuser statements: CHANGE MASTER TO, KILL (users who do not have this privilege can only KILL their own threads), PURGE LOGS, SET global system variables, or the mariadb-admin debug command. Also, this permission allows the user to write data even if the read_only startup option is set, enable or disable logging, enable or disable replication on slaves, specify a DEFINER for statements that support that clause, connect once after reaching the MAX_CONNECTIONS. If a statement has been specified for the init-connect mysqld option, that command will not be executed when a user with SUPER privileges connects to the server.
Create_tmp_table_priv	enum('N','Y')	NO		N	Can create temporary tables with the CREATE TEMPORARY TABLE statement.
Lock_tables_priv	enum('N','Y')	NO		N	Acquire explicit locks using the LOCK TABLES statement; user also needs to have the SELECT privilege on a table in order to lock it.
Execute_priv	enum('N','Y')	NO		N	Can execute stored procedure or functions.
Repl_slave_priv	enum('N','Y')	NO		N	Accounts used by slave servers on the master need this privilege. This is needed to get the updates made on the master.
Repl_client_priv	enum('N','Y')	NO		N	Can execute SHOW MASTER STATUS and SHOW SLAVE STATUS statements.
Create_view_priv	enum('N','Y')	NO		N	Can create a view using the CREATE_VIEW statement.

Field	Type	Null	Key	Default	Description
Show_view_priv	enum('N','Y')	NO		N	Can show the CREATE VIEW statement to create a view using the SHOW CREATE VIEW statement.
Create_routine_priv	enum('N','Y')	NO		N	Can create stored programs using the CREATE PROCEDURE and CREATE FUNCTION statements.
Alter_routine_priv	enum('N','Y')	NO		N	Can change the characteristics of a stored function using the ALTER FUNCTION statement.
Create_user_priv	enum('N','Y')	NO		N	Can create a user using the CREATE USER statement, or implicitly create a user with the GRANT statement.
Event_priv	enum('N','Y')	NO		N	Create, drop and alter events.
Trigger_priv	enum('N','Y')	NO		N	Can execute triggers associated with tables the user updates, execute the CREATE TRIGGER and DROP TRIGGER statements.
Create_tablespace_priv	enum('N','Y')	NO		N	
Delete_history_priv	enum('N','Y')	NO		N	Can delete rows created through system versioning.
ssl_type	enum('', 'ANY', 'X509', 'SPECIFIED')	NO			TLS type - see TLS options.
ssl_cipher	blob	NO		NULL	TLS cipher - see TLS options.
x509_issuer	blob	NO		NULL	X509 cipher - see TLS options.
x509_subject	blob	NO		NULL	SSL subject - see TLS options.
max_questions	int(11) unsigned	NO		0	Number of queries the user can perform per hour. Zero is unlimited. See per-account resource limits.

Field	Type	Null	Key	Default	Description
max_updates	int(11) unsigned	NO		0	Number of updates the user can perform per hour. Zero is unlimited. See per-account resource limits.
max_connections	int(11) unsigned	NO		0	Number of connections the account can start per hour. Zero is unlimited. See per-account resource limits.
max_user_connections	int(11)	NO		0	Number of simultaneous connections the account can have. Zero is unlimited. See per-account resource limits.
plugin	char(64)	NO			Authentication plugin used on connection. If empty, uses the default.
authentication_string	text	NO		NULL	Authentication string for the authentication plugin.
password_expired	enum('N','Y')	NO		N	MySQL-compatibility option, not implemented in MariaDB.
is_role	enum('N','Y')	NO		N	Whether the user is a role.
default_role	char(80)	NO		N	Role which will be enabled on user login automatically.
max_statement_time	decimal(12,6)	NO		0.000000	If non-zero, how long queries can run before being killed automatically.
Field	Type	Null	Key	Default	Description

The Acl_roles status variable indicates how many rows the `mysql.user` table contains where `is_role='Y'`.

The Acl_users status variable, indicates how many rows the `mysql.user` table contains where `is_role='N'`.

Authentication Plugin

When the `plugin` column is empty, MariaDB defaults to authenticating accounts with either the mysql_native_password or the mysql_old_password plugins. It decides which based on the hash used in the value for the `Password` column. When there's no password set or when the 4.1 password hash is used, (which is 41 characters long), MariaDB uses the mysql_native_password plugin. The mysql_old_password plugin is used with pre-4.1 password hashes, (which are 16 characters long).

MariaDB also supports the use of alternative authentication plugins. When the `plugin` column is not empty for the given account, MariaDB uses it to authenticate connection attempts. The specific plugin then uses the value of either the `Password` column or the `authentication_string` column to authenticate the user.

A specific authentication plugin can be used for an account by providing the `IDENTIFIED VIA authentication_plugin` clause with the CREATE USER, ALTER USER, or GRANT statements.

For example, the following statement would create an account that authenticates with the PAM authentication plugin:

```
CREATE USER foo2@test IDENTIFIED VIA pam;
```

If the specific authentication plugin uses the `authentication_string` column, then this value for the account can be specified after a `USING` or `AS` keyword. For example, the PAM authentication plugin accepts a service name that would go into the `authentication_string` column for the account:

```
CREATE USER foo2@test IDENTIFIED VIA pam USING 'mariadb';
```

This page is licensed: CC BY-SA / Gnu FDL

Spider mysql Database Tables

See mysql.spider_link_failed_log Table.

This table logs failures when attempting to establish connections to remote Spider links, helping to diagnose network or configuration issues.

See mysql.spider_link_mon_servers Table.

This table defines the remote servers used for link monitoring in the Spider storage engine, ensuring high availability and failover handling.

See mysql.spider_table_crd Table.

The mysql.spider_table_crd table stores cardinality statistics for Spider tables, which the optimizer uses to create efficient query plans.

See mysql.spider_table_position_for_recovery Table.

This table records XA transaction positions for Spider tables, essential for recovering distributed transactions after a crash.

See mysql.spider_table_sts Table.

The mysql.spider_table_sts table holds statistics such as row counts and data length for Spider tables, supporting the optimizer.

See mysql.spider_tables Table.

This table stores specific parameters and metadata for Spider tables, defining how they map to remote backend tables.

See mysql.spider_xa Table.

The mysql.spider_xa table tracks the status of XA transactions involving Spider tables, ensuring atomicity across distributed nodes.

See mysql.spider_xa_failed_log Table.

This table logs details of failed XA transactions in the Spider storage engine, providing a record for troubleshooting distributed transaction errors.

See mysql.spider_xa_member Table.

The mysql.spider_xa_member table lists the member nodes participating in a distributed XA transaction managed by the Spider engine.

This page is licensed: CC BY-SA / Gnu FDL

mysql.spider_link_failed_log Table

The `mysql.spider_link_failed_log` table is installed by the Spider storage engine.

This table uses the Aria storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO			
table_name	char(199)	NO			
link_id	char(64)	NO			
failed_time	timestamp	NO		current_timestamp()	

This page is licensed: CC BY-SA / Gnu FDL

mysql.spider_link_mon_servers Table

The `mysql.spider_link_mon_servers` table is installed by the Spider storage engine.

This table uses the Aria storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		
table_name	char(199)	NO	PRI		
link_id	char(64)	NO	PRI		
sid	int(10) unsigned	NO	PRI	0	
server	char(64)	YES		NULL	
scheme	char(64)	YES		NULL	
host	char(64)	YES		NULL	
port	char(5)	YES		NULL	
socket	text	YES		NULL	
username	char(64)	YES		NULL	
password	char(64)	YES		NULL	
ssl_ca	text	YES		NULL	
ssl_capath	text	YES		NULL	
ssl_cert	text	YES		NULL	
ssl_cipher	char(64)	YES		NULL	
ssl_key	text	YES		NULL	
ssl_verify_server_cert	tinyint(4)	NO		0	
default_file	text	YES		NULL	

Field	Type	Null	Key	Default	Description
default_group	char(64)	YES		NULL	
dsn	char(64)	YES		NULL	
filedsn	text	YES		NULL	
driver	char(64)	YES		NULL	

This page is licensed: CC BY-SA / Gnu FDL

mysql.spider_table_crd Table

The `mysql.spider_table_crd` table is installed by the Spider storage engine.

This table uses the Aria storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		
table_name	char(199)	NO	PRI		
key_seq	int(10) unsigned	NO	PRI	0	
cardinality	bigint(20)	NO		0	

This page is licensed: CC BY-SA / Gnu FDL

mysql.spider_table_position_for_recovery Table

The `mysql.spider_table_position_for_recovery` table is installed by the Spider storage engine.

This table uses the Aria storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		
table_name	char(199)	NO	PRI		
failed_link_id	int(11)	NO	PRI	0	
source_link_id	int(11)	NO	PRI	0	
file	text	YES		NULL	
position	text	YES		NULL	
gtid	text	YES		NULL	

This page is licensed: CC BY-SA / Gnu FDL

mysql.spider_table_sts Table

The `mysql.spider_table_sts` table is installed by the Spider storage engine.

This table uses the Aria storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		

Field	Type	Null	Key	Default	Description
table_name	char(199)	NO	PRI		
data_file_length	bigint(20) unsigned	NO		0	
max_data_file_length	bigint(20) unsigned	NO		0	
index_file_length	bigint(20) unsigned	NO		0	
records	bigint(20) unsigned	NO		0	
mean_rec_length	bigint(20) unsigned	NO		0	
check_time	datetime	NO		0000-00-00 00:00:00	
create_time	datetime	NO		0000-00-00 00:00:00	
update_time	datetime	NO		0000-00-00 00:00:00	
checksum	bigint(20) unsigned	YES		NULL	

This page is licensed: CC BY-SA / Gnu FDL

mysql.spider_tables Table

The `mysql.spider_tables` table is installed by the Spider storage engine.

This table uses the Aria storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		
table_name	char(199)	NO	PRI		
link_id	int(11)	NO	PRI	0	
priority	bigint(20)	NO	MUL	0	
server	char(64)	YES		NULL	
scheme	char(64)	YES		NULL	
host	char(64)	YES		NULL	
port	char(5)	YES		NULL	
socket	text	YES		NULL	
username	char(64)	YES		NULL	
password	char(64)	YES		NULL	
ssl_ca	text	YES		NULL	
ssl_capath	text	YES		NULL	
ssl_cert	text	YES		NULL	
ssl_cipher	char(64)	YES		NULL	
ssl_key	text	YES		NULL	
ssl_verify_server_cert	tinyint(4)	NO		0	
monitoring_binlog_pos_at_failing	tinyint(4)	NO		0	
default_file	text	YES		NULL	
default_group	char(64)	YES		NULL	
dsn	char(64)	YES		NULL	
filedsn	text	YES		NULL	

Field	Type	Null	Key	Default	Description
driver	char(64)	YES		NULL	
tgt_db_name	char(64)	YES		NULL	
tgt_table_name	char(64)	YES		NULL	
link_status	tinyint(4)	NO		1	
block_status	tinyint(4)	NO		0	
static_link_id	char(64)	YES		NULL	

This page is licensed: CC BY-SA / Gnu FDL

mysql.spider_xa Table

The `mysql.spider_xa` table is installed by the Spider storage engine.

This table uses the Aria storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
format_id	int(11)	NO	PRI	0	
gtrid_length	int(11)	NO	PRI	0	
bqual_length	int(11)	NO		0	
data	binary(128)	NO	PRI		
status	char(8)	NO	MUL		

This page is licensed: CC BY-SA / Gnu FDL

mysql.spider_xa_failed_log Table

The `mysql.spider_xa_failed_log` table is installed by the Spider storage engine.

This table uses the Aria storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
format_id	int(11)	NO		0	
gtrid_length	int(11)	NO		0	
bqual_length	int(11)	NO		0	
data	binary(128)	NO	MUL		
scheme	char(64)	NO			
host	char(64)	NO			
port	char(5)	NO			
socket	text	NO		NULL	
username	char(64)	NO			
password	char(64)	NO			
ssl_ca	text	YES		NULL	
ssl_capath	text	YES		NULL	
ssl_cert	text	YES		NULL	

Field	Type	Null	Key	Default	Description
ssl_cipher	char(64)	YES		NULL	
ssl_key	text	YES		NULL	
ssl_verify_server_cert	tinyint(4)	NO		0	
default_file	text	YES		NULL	
default_group	char(64)	YES		NULL	
dsn	char(64)	YES		NULL	
filedsn	text	YES		NULL	
driver	char(64)	YES		NULL	
thread_id	int(11)	YES		NULL	
status	char(8)	NO			
failed_time	timestamp	NO		current_timestamp()	

This page is licensed: CC BY-SA / Gnu FDL

mysql.spider_xa_member Table

The `mysql.spider_xa_member` table is installed by the Spider storage engine.

This table uses the Aria storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
format_id	int(11)	NO		0	
gtrid_length	int(11)	NO		0	
bqual_length	int(11)	NO		0	
data	binary(128)	NO	MUL		
scheme	char(64)	NO			
host	char(64)	NO			
port	char(5)	NO			
socket	text	NO		NULL	
username	char(64)	NO			
password	char(64)	NO			
ssl_ca	text	YES		NULL	
ssl_capath	text	YES		NULL	
ssl_cert	text	YES		NULL	
ssl_cipher	char(64)	YES		NULL	
ssl_key	text	YES		NULL	
ssl_verify_server_cert	tinyint(4)	NO		0	
default_file	text	YES		NULL	
default_group	char(64)	YES		NULL	
dsn	char(64)	YES		NULL	
filedsn	text	YES		NULL	
driver	char(64)	YES		NULL	

This page is licensed: CC BY-SA / Gnu FDL

Error Codes

See MariaDB Error Code Reference.

An overview of MariaDB error codes, including those shared with MySQL and those specific to MariaDB, with their symbolic names and messages.

See Operating System Error Codes.

Common Linux and Windows operating-system error codes MariaDB may report, and how to look them up with the perror tool.

See MariaDB Error Codes 1000 to 1099.

MariaDB server error codes 1000 to 1099, each with its symbolic name and message.

See MariaDB Error Codes 1100 to 1199.

MariaDB server error codes 1100 to 1199, each with its symbolic name and message.

See MariaDB Error Codes 1200 to 1299.

MariaDB server error codes 1200 to 1299, each with its symbolic name and message.

See MariaDB Error Codes 1300 to 1399.

MariaDB server error codes 1300 to 1399, each with its symbolic name and message.

See MariaDB Error Codes 1400 to 1499.

MariaDB server error codes 1400 to 1499, each with its symbolic name and message.

See MariaDB Error Codes 1500 to 1599.

MariaDB server error codes 1500 to 1599, each with its symbolic name and message.

See MariaDB Error Codes 1600 to 1699.

MariaDB server error codes 1600 to 1699, each with its symbolic name and message.

See MariaDB Error Codes 1700 to 1799.

MariaDB server error codes 1700 to 1799, each with its symbolic name and message.

See MariaDB Error Codes 1800 to 1899.

MariaDB server error codes 1800 to 1899, each with its symbolic name and message.

See MariaDB Error Codes 1900 to 1999.

MariaDB server error codes 1900 to 1999, each with its symbolic name and message.

See MariaDB Error Codes 3000 to 3099.

MariaDB server error codes 3000 to 3099, each with its symbolic name and message.

See MariaDB Error Codes 4000 to 4099.

MariaDB server error codes 4000 to 4099, each with its symbolic name and message.

See MariaDB Error Codes 4100 to 4199.

MariaDB server error codes 4100 to 4199, each with its symbolic name and message.

See MariaDB Error Codes 4200 to 4299.

MariaDB server error codes 4200 to 4299, each with its symbolic name and message.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Code Reference

> **Warning:**
>
> Error codes from 4195 onward have been shifted!

MariaDB shares error codes with MySQL. There are also a number of error codes specific to MariaDB.

An example of an error code:

```
SELECT * FROM x;
ERROR 1046 (3D000): No database selected
```

There are three pieces of information returned in an error:

- A numeric error code, in this case `1046`. Error codes from `1900` and up are specific to MariaDB, while error codes from `1000` to 1800 are shared by MySQL and MariaDB.
- An SQLSTATE value, consisting of five characters, in this case `3D000`. These codes are standard to ODBC and ANSI SQL. When MariaDB cannot allocate a standard `SQLSTATE` code, a generic `HY000`, or general error, is used.
- A string describing the error, in this case `No database selected`.

New error codes are being continually being added as new features are added. For a definitive list, see the file `sql/share/errmsg-utf8.txt`, as well as `include/mysqld_error.h` in the build directory, generated by the `comp_err` tool. The perror tool can be used to get the error message associated with a given error code.

Shared MariaDB/MySQL Error Codes

Error Code	SQLSTATE	Error	Description
1000	HY000	ER_HASHCHK	hashchk
1001	HY000	ER_NISAMCHK	isamchk
1002	HY000	ER_NO	NO
1003	HY000	ER_YES	YES
1004	HY000	ER_CANT_CREATE_FILE	Can’t create file ‘%s’ (errno: %d)
1005	HY000	ER_CANT_CREATE_TABLE	Can’t create table ‘%s’ (errno: %d)
1006	HY000	ER_CANT_CREATE_DB	Can’t create database ‘%s’ (errno: %d)
1007	HY000	ER_DB_CREATE_EXISTS	Can’t create database ‘%s’; database exists
1008	HY000	ER_DB_DROP_EXISTS	Can’t drop database ‘%s’; database doesn’t exist
1009	HY000	ER_DB_DROP_DELETE	Error dropping database (can’t delete ‘%s’, errno: %d)

Error Code	SQLSTATE	Error	Description
1010	HY000	ER_DB_DROP_RMDIR	Error dropping database (can't rmdir '%s', errno: %d)
1011	HY000	ER_CANT_DELETE_FILE	Error on delete of '%s' (errno: %d)
1012	HY000	ER_CANT_FIND_SYSTEM_REC	Can't read record in system table
1013	HY000	ER_CANT_GET_STAT	Can't get status of '%s' (errno: %d)
1014	HY000	ER_CANT_GET_WD	Can't get working directory (errno: %d)
1015	HY000	ER_CANT_LOCK	Can't lock file (errno: %d)
1016	HY000	ER_CANT_OPEN_FILE	Can't open file: '%s' (errno: %d)
1017	HY000	ER_FILE_NOT_FOUND	Can't find file: '%s' (errno: %d)
1018	HY000	ER_CANT_READ_DIR	Can't read dir of '%s' (errno: %d)
1019	HY000	ER_CANT_SET_WD	Can't change dir to '%s' (errno: %d)
1020	HY000	ER_CHECKREAD	Record has changed since last read in table '%s'
1021	HY000	ER_DISK_FULL	Disk full (%s); waiting for someone to free some space…
1022	23000	ER_DUP_KEY	Can't write; duplicate key in table '%s'
1023	HY000	ER_ERROR_ON_CLOSE	Error on close of '%s' (errno: %d)
1024	HY000	ER_ERROR_ON_READ	Error reading file '%s' (errno: %d)
1025	HY000	ER_ERROR_ON_RENAME	Error on rename of '%s' to '%s' (errno: %d)
1026	HY000	ER_ERROR_ON_WRITE	Error writing file '%s' (errno: %d)
1027	HY000	ER_FILE_USED	'%s' is locked against change
1028	HY000	ER_FILSORT_ABORT	Sort aborted
1029	HY000	ER_FORM_NOT_FOUND	View '%s' doesn't exist for '%s'
1030	HY000	ER_GET_ERRN	Got error %d from storage engine
1031	HY000	ER_ILLEGAL_HA	Table storage engine for '%s' doesn't have this option
1032	HY000	ER_KEY_NOT_FOUND	Can't find record in '%s'
1033	HY000	ER_NOT_FORM_FILE	Incorrect information in file: '%s'

Error Code	SQLSTATE	Error	Description
1034	HY000	ER_NOT_KEYFILE	Incorrect key file for table ‘%s’; try to repair it
1035	HY000	ER_OLD_KEYFILE	Old key file for table ‘%s’; repair it!
1036	HY000	ER_OPEN_AS_READONLY	Table ‘%s’ is read only
1037	HY001	ER_OUTOFMEMORY	Out of memory; restart server and try again (needed %d bytes)
1038	HY001	ER_OUT_OF_SORTMEMORY	Out of sort memory, consider increasing server sort buffer size
1039	HY000	ER_UNEXPECTED_EOF	Unexpected EOF found when reading file ‘%s’ (Errno: %d)
1040	08004	ER_CON_COUNT_ERROR	Too many connections
1041	HY000	ER_OUT_OF_RESOURCES	Out of memory; check if mysqld or some other process uses all available memory; if not, you may have to use ‘ulimit’ to allow mysqld to use more memory or you can add more swap space
1042	08S01	ER_BAD_HOST_ERROR	Can’t get hostname for your address
1043	08S01	ER_HANDSHAKE_ERROR	Bad handshake
1044	42000	ER_DBACCESS_DENIED_ERROR	Access denied for user ‘%s’@‘%s’ to database ‘%s’
1045	28000	ER_ACCESS_DENIED_ERROR	Access denied for user ‘%s’@‘%s’ (using password: %s)
1046	3D000	ER_NO_DB_ERROR	No database selected
1047	08S01	ER_UNKNOWN_COM_ERROR	Unknown command
1048	23000	ER_BAD_NULL_ERROR	Column ‘%s’ cannot be null
1049	42000	ER_BAD_DB_ERROR	Unknown database ‘%s’
1050	42S01	ER_TABLE_EXISTS_ERROR	Table ‘%s’ already exists
1051	42S02	ER_BAD_TABLE_ERROR	Unknown table ‘%s’
1052	23000	ER_NON_UNIQ_ERROR	Column ‘%s’ in %s is ambiguous
1053	08S01	ER_SERVER_SHUTDOWN	Server shutdown in progress
1054	42S22	ER_BAD_FIELD_ERROR	Unknown column ‘%s’ in ‘%s’
1055	42000	ER_WRONG_FIELD_WITH_GROUP	‘%s’ isn’t in GROUP BY
1056	42000	ER_WRONG_GROUP_FIELD	Can’t group on ‘%s’
1057	42000	ER_WRONG_SUM_SELECT	Statement has sum functions and columns in same statement

Error Code	SQLSTATE	Error	Description
1058	21S01	ER_WRONG_VALUE_COUNT	Column count doesn't match value count
1059	42000	ER_TOO_LONG_IDENT	Identifier name '%s' is too long
1060	42S21	ER_DUP_FIELDNAME	Duplicate column name '%s'
1061	42000	ER_DUP_KEYNAME	Duplicate key name '%s'
1062	23000	ER_DUP_ENTRY	Duplicate entry '%s' for key %d
1063	42000	ER_WRONG_FIELD_SPEC	Incorrect column specifier for column '%s'
1064	42000	ER_PARSE_ERROR	%s near '%s' at line %d
1065	42000	ER_EMPTY_QUERY	Query was empty
1066	42000	ER_NONUNIQ_TABLE	Not unique table/alias: '%s'
1067	42000	ER_INVALID_DEFAULT	Invalid default value for '%s'
1068	42000	ER_MULTIPLE_PRI_KEY	Multiple primary key defined
1069	42000	ER_TOO_MANY_KEYS	Too many keys specified; max %d keys allowed
1070	42000	ER_TOO_MANY_KEY_PARTS	Too many key parts specified; max %d parts allowed
1071	42000	ER_TOO_LONG_KEY	Specified key was too long; max key length is %d bytes
1072	42000	ER_KEY_COLUMN_DOES_NOT_EXITS	Key column '%s' doesn't exist in table
1073	42000	ER_BLOB_USED_AS_KEY	BLOB column '%s' can't be used in key specification with the used table type
1074	42000	ER_TOO_BIG_FIELDLENGTH	Column length too big for column '%s' (max = %lu); use BLOB or TEXT instead
1075	42000	ER_WRONG_AUTO_KEY	Incorrect table definition; there can be only one auto column and it must be defined as a key
1076	HY000	ER_READY	%s: ready for connections. Version: '%s' socket: '%s' port: %d
1077	HY000	ER_NORMAL_SHUTDOWN	%s: Normal shutdown
1078	HY000	ER_GOT_SIGNAL	%s: Got signal %d. Aborting!
1079	HY000	ER_SHUTDOWN_COMPLETE	%s: Shutdown complete
1080	08S01	ER_FORCING_CLOSE	%s: Forcing close of thread %ld user: '%s'
1081	08S01	ER_IPSOCK_ERROR	Can't create IP socket
1082	42S12	ER_NO_SUCH_INDEX	Table '%s' has no index like the one used in CREATE INDEX; recreate the table

Error Code	SQLSTATE	Error	Description
1083	42000	ER_WRONG_FIELD_TERMINATORS	Field separator argument is not what is expected; check the manual
1084	42000	ER_BLOBS_AND_NO_TERMINATED	You can't use fixed rowlength with BLOBs; please use 'fields terminated by'
1085	HY000	ER_TEXTFILE_NOT_READABLE	The file '%s' must be in the database directory or be readable by all
1086	HY000	ER_FILE_EXISTS_ERROR	File '%s' already exists
1087	HY000	ER_LOAD_INF	Records: %ld Deleted: %ld Skipped: %ld Warnings: %ld
1088	HY000	ER_ALTER_INF	Records: %ld Duplicates: %ld
1089	HY000	ER_WRONG_SUB_KEY	Incorrect prefix key; the used key part isn't a string, the used length is longer than the key part, or the storage engine doesn't support unique prefix keys
1090	42000	ER_CANT_REMOVE_ALL_FIELDS	You can't delete all columns with ALTER TABLE; use DROP TABLE instead
1091	42000	ER_CANT_DROP_FIELD_OR_KEY	Can't DROP '%s'; check that column/key exists
1092	HY000	ER_INSERT_INF	Records: %ld Duplicates: %ld Warnings: %ld
1093	HY000	ER_UPDATE_TABLE_USED	You can't specify target table '%s' for update in FROM clause
1094	HY000	ER_NO_SUCH_THREAD	Unknown thread id: %lu
1095	HY000	ER_KILL_DENIED_ERROR	You are not owner of thread %lu
1096	HY000	ER_NO_TABLES_USED	No tables used
1097	HY000	ER_TOO_BIG_SET	Too many strings for column %s and SET
1098	HY000	ER_NO_UNIQUE_LOGFILE	Can't generate a unique log-filename %s.(1-999)
1099	HY000	ER_TABLE_NOT_LOCKED_FOR_WRITE	Table '%s' was locked with a READ lock and can't be updated

Error Code	SQLSTATE	Error	Description
1100	HY000	ER_TABLE_NOT_LOCKED	Table '%s' was not locked with LOCK TABLES
1101		ER_UNUSED_17	You should never see it
1102	42000	ER_WRONG_DB_NAME	Incorrect database name '%s'
1103	42000	ER_WRONG_TABLE_NAME	Incorrect table name '%s'
1104	42000	ER_TOO_BIG_SELECT	The SELECT would examine more than MAX_JOIN_SIZE rows; check your WHERE and use SET SQL_BIG_SELECTS=1 or SET MAX_JOIN_SIZE= ## if the SELECT is okay
1105	HY000	ER_UNKNOWN_ERROR	Unknown error

Error Code	SQLSTATE	Error	Description
1106	42000	ER_UNKNOWN_PROCEDURE	Unknown procedure '%s'
1107	42000	ER_WRONG_PARAMCOUNT_TO_PROCEDURE	Incorrect parameter count to procedure '%s'
1108	HY000	ER_WRONG_PARAMETERS_TO_PROCEDURE	Incorrect parameters to procedure '%s'
1109	42S02	ER_UNKNOWN_TABLE	Unknown table '%s' in %s
1110	42000	ER_FIELD_SPECIFIED_TWICE	Column '%s' specified twice
1111	HY000	ER_INVALID_GROUP_FUNC_USE	Invalid use of group function
1112	42000	ER_UNSUPPORTED_EXTENSION	Table '%s' uses an extension that doesn't exist in this MariaDB version
1113	42000	ER_TABLE_MUST_HAVE_COLUMNS	A table must have at least 1 column
1114	HY000	ER_RECORD_FILE_FULL	The table '%s' is full
1115	42000	ER_UNKNOWN_CHARACTER_SET	Unknown character set: '%s'
1116	HY000	ER_TOO_MANY_TABLES	Too many tables; MariaDB can only use %d tables in a join
1117	HY000	ER_TOO_MANY_FIELDS	Too many columns
1118	42000	ER_TOO_BIG_ROWSIZE	Row size too large. The maximum row size for the used table type, not counting BLOBs, is %ld. You have to change some columns to TEXT or BLOBs
1119	HY000	ER_STACK_OVERRUN	Thread stack overrun: Used: %ld of a %ld stack. Use 'mysqld –thread_stack=#' to specify a bigger stack if needed
1120	42000	ER_WRONG_OUTER_JOIN	Cross dependency found in OUTER JOIN; examine your ON conditions
1121	42000	ER_NULL_COLUMN_IN_INDEX	Table handler doesn't support NULL in given index. Please change column '%s' to be NOT NULL or use another handler
1122	HY000	ER_CANT_FIND_UDF	Can't load function '%s'
1123	HY000	ER_CANT_INITIALIZE_UDF	Can't initialize function '%s'; %s
1124	HY000	ER_UDF_NO_PATHS	No paths allowed for shared library
1125	HY000	ER_UDF_EXISTS	Function '%s' already exists
1126	HY000	ER_CANT_OPEN_LIBRARY	Can't open shared library '%s' (Errno: %d %s)
1127	HY000	ER_CANT_FIND_DL_ENTRY	Can't find symbol '%s' in library
1128	HY000	ER_FUNCTION_NOT_DEFINED	Function '%s' is not defined
1129	HY000	ER_HOST_IS_BLOCKED	Host '%s' is blocked because of many connection errors; unblock with 'mysqladmin flush-hosts'

Error Code	SQLSTATE	Error	Description
1130	HY000	ER_HOST_NOT_PRIVILEGED	Host ‘%s’ is not allowed to connect to this MariaDB server
1131	42000	ER_PASSWORD_ANONYMOUS_USER	You are using MariaDB as an anonymous user and anonymous users are not allowed to change passwords
1132	42000	ER_PASSWORD_NOT_ALLOWED	You must have privileges to update tables in the mysql database to be able to change passwords for others
1133	42000	ER_PASSWORD_NO_MATCH	Can’t find any matching row in the user table
1134	HY000	ER_UPDATE_INF	Rows matched: %ld Changed: %ld Warnings: %ld
1135	HY000	ER_CANT_CREATE_THREAD	Can’t create a new thread (Errno %d); if you are not out of available memory, you can consult the manual for a possible OS-dependent bug
1136	21S01	ER_WRONG_VALUE_COUNT_ON_ROW	Column count doesn’t match value count at row %ld
1137	HY000	ER_CANT_REOPEN_TABLE	Can’t reopen table: ‘%s’
1138	22004	ER_INVALID_USE_OF_NULL	Invalid use of NULL value
1139	42000	ER_REGEXP_ERROR	Got error ‘%s’ from regexp
1140	42000	ER_MIX_OF_GROUP_FUNC_AND_FIELDS	Mixing of GROUP columns (MIN(),MAX(),COUNT(),…) with no GROUP columns is illegal if there is no GROUP BY clause
1141	42000	ER_NONEXISTING_GRANT	There is no such grant defined for user ‘%s’ on host ‘%s’
1142	42000	ER_TABLEACCESS_DENIED_ERROR	%s command denied to user ‘%s’@‘%s’ for table ‘%s’
1143	42000	ER_COLUMNACCESS_DENIED_ERROR	%s command denied to user ‘%s’@‘%s’ for column ‘%s’ in table ‘%s’
1144	42000	ER_ILLEGAL_GRANT_FOR_TABLE	Illegal GRANT/REVOKE command; please consult the manual to see which privileges can be used
1145	42000	ER_GRANT_WRONG_HOST_OR_USER	The host or user argument to GRANT is too long
1146	42S02	ER_NO_SUCH_TABLE	Table ‘%s.%s’ doesn’t exist
1147	42000	ER_NONEXISTING_TABLE_GRANT	There is no such grant defined for user ‘%s’ on host ‘%s’ on table ‘%s’
1148	42000	ER_NOT_ALLOWED_COMMAND	The used command is not allowed with this MariaDB version
1149	42000	ER_SYNTAX_ERROR	You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use
1150	HY000	ER_DELAYED_CANT_CHANGE_LOCK	Delayed insert thread couldn’t get requested lock for table %s
1151	HY000	ER_TOO_MANY_DELAYED_THREADS	Too many delayed threads in use
1152	08S01	ER_ABORTING_CONNECTION	Aborted connection %ld to db: ‘%s’ user: ‘%s’ (%s)
1153	08S01	ER_NET_PACKET_TOO_LARGE	Got a packet bigger than ‘max_allowed_packet’ bytes

Error Code	SQLSTATE	Error	Description
1154	08S01	ER_NET_READ_ERROR_FROM_PIPE	Got a read error from the connection pipe
1155	08S01	ER_NET_FCNTL_ERROR	Got an error from fcntl()
1156	08S01	ER_NET_PACKETS_OUT_OF_ORDER	Got packets out of order
1157	08S01	ER_NET_UNCOMPRESS_ERROR	Couldn't uncompress communication packet
1158	08S01	ER_NET_READ_ERROR	Got an error reading communication packets
1159	08S01	ER_NET_READ_INTERRUPTED	Got timeout reading communication packets
1160	08S01	ER_NET_ERROR_ON_WRITE	Got an error writing communication packets
1161	08S01	ER_NET_WRITE_INTERRUPTED	Got timeout writing communication packets
1162	42000	ER_TOO_LONG_STRING	Result string is longer than 'max_allowed_packet' bytes
1163	42000	ER_TABLE_CANT_HANDLE_BLOB	The used table type doesn't support BLOB/TEXT columns
1164	42000	ER_TABLE_CANT_HANDLE_AUTO_INCREMENT	The used table type doesn't support AUTO_INCREMENT columns
1165	HY000	ER_DELAYED_INSERT_TABLE_LOCKED	INSERT DELAYED can't be used with table '%s' because it is locked with LOCK TABLES
1166	42000	ER_WRONG_COLUMN_NAME	Incorrect column name '%s'
1167	42000	ER_WRONG_KEY_COLUMN	The used storage engine can't index column '%s'
1168	HY000	ER_WRONG_MRG_TABLE	Unable to open underlying table which is differently defined or of non-MyISAM type or doesn't exist
1169	23000	ER_DUP_UNIQUE	Can't write, because of unique constraint, to table '%s'
1170	42000	ER_BLOB_KEY_WITHOUT_LENGTH	BLOB/TEXT column '%s' used in key specification without a key length
1171	42000	ER_PRIMARY_CANT_HAVE_NULL	All parts of a PRIMARY KEY must be NOT NULL; if you need NULL in a key, use UNIQUE instead
1172	42000	ER_TOO_MANY_ROWS	Result consisted of more than one row
1173	42000	ER_REQUIRES_PRIMARY_KEY	This table type requires a primary key
1174	HY000	ER_NO_RAID_COMPILED	This version of MariaDB is not compiled with RAID support
1175	HY000	ER_UPDATE_WITHOUT_KEY_IN_SAFE_MODE	You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column
1176	42000	ER_KEY_DOES_NOT_EXITS	Key '%s' doesn't exist in table '%s'
1177	42000	ER_CHECK_NO_SUCH_TABLE	Can't open table
1178	42000	ER_CHECK_NOT_IMPLEMENTED	The storage engine for the table doesn't support %s

Error Code	SQLSTATE	Error	Description
1179	25000	ER_CANT_DO_THIS_DURING_AN_TRANSACTION	You are not allowed to execute this command in a transaction
1180	HY000	ER_ERROR_DURING_COMMIT	Got error %d during COMMIT
1181	HY000	ER_ERROR_DURING_ROLLBACK	Got error %d during ROLLBACK
1182	HY000	ER_ERROR_DURING_FLUSH_LOGS	Got error %d during FLUSH_LOGS
1183	HY000	ER_ERROR_DURING_CHECKPOINT	Got error %d during CHECKPOINT
1184	08S01	ER_NEW_ABORTING_CONNECTION	Aborted connection %ld to db: '%s' user: '%s' host: '%s' (%s)
1185		ER_UNUSED_10	You should never see it
1186	HY000	ER_FLUSH_MASTER_BINLOG_CLOSED	Binlog closed, cannot RESET MASTER
1187	HY000	ER_INDEX_REBUILD	Failed rebuilding the index of dumped table '%s'
1188	HY000	ER_MASTER	Error from master: '%s'
1189	08S01	ER_MASTER_NET_READ	Net error reading from master
1190	08S01	ER_MASTER_NET_WRITE	Net error writing to master
1191	HY000	ER_FT_MATCHING_KEY_NOT_FOUND	Can't find FULLTEXT index matching the column list
1192	HY000	ER_LOCK_OR_ACTIVE_TRANSACTION	Can't execute the given command because you have active locked tables or an active transaction
1193	HY000	ER_UNKNOWN_SYSTEM_VARIABLE	Unknown system variable '%s'
1194	HY000	ER_CRASHED_ON_USAGE	Table '%s' is marked as crashed and should be repaired
1195	HY000	ER_CRASHED_ON_REPAIR	Table '%s' is marked as crashed and last (automatic?) repair failed
1196	HY000	ER_WARNING_NOT_COMPLETE_ROLLBACK	Some non-transactional changed tables couldn't be rolled back
1197	HY000	ER_TRANS_CACHE_FULL	Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage; increase this mysqld variable and try again
1198	HY000	ER_SLAVE_MUST_STOP	This operation cannot be performed with a running slave; run STOP SLAVE first
1199	HY000	ER_SLAVE_NOT_RUNNING	This operation requires a running slave; configure slave and do START SLAVE

Error Code	SQLSTATE	Error	Description
1200	HY000	ER_BAD_SLAVE	The server is not configured as slave; fix in config file or with CHANGE MASTER TO
1201	HY000	ER_MASTER_INFO	Could not initialize master info structure; more error messages can be found in the MariaDB error log

Error Code	SQLSTATE	Error	Description
1202	HY000	ER_SLAVE_THREAD	Could not create slave thread; check system resources
1203	42000	ER_TOO_MANY_USER_CONNECTIONS	User %s already has more than 'max_user_connections' active connections
1204	HY000	ER_SET_CONSTANTS_ONLY	You may only use constant expressions with SET
1205	HY000	ER_LOCK_WAIT_TIMEOUT	Lock wait timeout exceeded; try restarting transaction
1206	HY000	ER_LOCK_TABLE_FULL	The total number of locks exceeds the lock table size
1207	25000	ER_READ_ONLY_TRANSACTION	Update locks cannot be acquired during a READ UNCOMMITTED transaction
1208	HY000	ER_DROP_DB_WITH_READ_LOCK	DROP DATABASE not allowed while thread is holding global read lock
1209	HY000	ER_CREATE_DB_WITH_READ_LOCK	CREATE DATABASE not allowed while thread is holding global read lock
1210	HY000	ER_WRONG_ARGUMENTS	Incorrect arguments to %s
1211	42000	ER_NO_PERMISSION_TO_CREATE_USER	'%s'@'%s' is not allowed to create new users
1212	HY000	ER_UNION_TABLES_IN_DIFFERENT_DIR	Incorrect table definition; all MERGE tables must be in the same database
1213	40001	ER_LOCK_DEADLOCK	Deadlock found when trying to get lock; try restarting transaction
1214	HY000	ER_TABLE_CANT_HANDLE_FT	The used table type doesn't support FULLTEXT indexes
1215	HY000	ER_CANNOT_ADD_FOREIGN	Cannot add foreign key constraint
1216	23000	ER_NO_REFERENCED_ROW	Cannot add or update a child row: a foreign key constraint fails
1217	23000	ER_ROW_IS_REFERENCED	Cannot delete or update a parent row: a foreign key constraint fails
1218	08S01	ER_CONNECT_TO_MASTER	Error connecting to master: %s
1219	HY000	ER_QUERY_ON_MASTER	Error running query on master: %s
1220	HY000	ER_ERROR_WHEN_EXECUTING_COMMAND	Error when executing command %s: %s
1221	HY000	ER_WRONG_USAGE	Incorrect usage of %s and %s
1222	21000	ER_WRONG_NUMBER_OF_COLUMNS_IN_SELECT	The used SELECT statements have a different number of columns
1223	HY000	ER_CANT_UPDATE_WITH_READLOCK	Can't execute the query because you have a conflicting read lock
1224	HY000	ER_MIXING_NOT_ALLOWED	Mixing of transactional and non-transactional tables is disabled
1225	HY000	ER_DUP_ARGUMENT	Option '%s' used twice in statement

Error Code	SQLSTATE	Error	Description
1226	42000	ER_USER_LIMIT_REACHED	User '%s' has exceeded the '%s' resource (current value: %ld)
1227	42000	ER_SPECIFIC_ACCESS_DENIED_ERROR	Access denied; you need (at least one of) the %s privilege(s) for this operation
1228	HY000	ER_LOCAL_VARIABLE	Variable '%s' is a SESSION variable and can't be used with SET GLOBAL
1229	HY000	ER_GLOBAL_VARIABLE	Variable '%s' is a GLOBAL variable and should be set with SET GLOBAL
1230	42000	ER_NO_DEFAULT	Variable '%s' doesn't have a default value
1231	42000	ER_WRONG_VALUE_FOR_VAR	Variable '%s' can't be set to the value of '%s'
1232	42000	ER_WRONG_TYPE_FOR_VAR	Incorrect argument type to variable '%s'
1233	HY000	ER_VAR_CANT_BE_READ	Variable '%s' can only be set, not read
1234	42000	ER_CANT_USE_OPTION_HERE	Incorrect usage/placement of '%s'
1235	42000	ER_NOT_SUPPORTED_YET	This version of MariaDB doesn't yet support '%s'
1236	HY000	ER_MASTER_FATAL_ERROR_READING_BINLOG	Got fatal error %d from master when reading data from binary log: '%s'
1237	HY000	ER_SLAVE_IGNORED_TABLE	Slave SQL thread ignored the query because of replicate-*-table rules
1238	HY000	ER_INCORRECT_GLOBAL_LOCAL_VAR	Variable '%s' is a %s variable
1239	42000	ER_WRONG_FK_DEF	Incorrect foreign key definition for '%s': %s
1240	HY000	ER_KEY_REF_DO_NOT_MATCH_TABLE_REF	Key reference and table reference don't match
1241	21000	ER_OPERAND_COLUMNS	Operand should contain %d column(s)
1242	21000	ER_SUBQUERY_NO_1_ROW	Subquery returns more than 1 row
1243	HY000	ER_UNKNOWN_STMT_HANDLER	Unknown prepared statement handler (%.*s) given to %s
1244	HY000	ER_CORRUPT_HELP_DB	Help database is corrupt or does not exist
1245	HY000	ER_CYCLIC_REFERENCE	Cyclic reference on subqueries
1246	HY000	ER_AUTO_CONVERT	Converting column '%s' from %s to %s
1247	42S22	ER_ILLEGAL_REFERENCE	Reference '%s' not supported (%s)
1248	42000	ER_DERIVED_MUST_HAVE_ALIAS	Every derived table must have its own alias
1249	01000	ER_SELECT_REDUCED	Select %u was reduced during optimization

Error Code	SQLSTATE	Error	Description
1250	42000	ER_TABLENAME_NOT_ALLOWED_HERE	Table '%s' from one of the SELECTs cannot be used in %s
1251	08004	ER_NOT_SUPPORTED_AUTH_MODE	Client does not support authentication protocol requested by server; consider upgrading MariaDB client
1252	42000	ER_SPATIAL_CANT_HAVE_NULL	All parts of a SPATIAL index must be NOT NULL
1253	42000	ER_COLLATION_CHARSET_MISMATCH	COLLATION '%s' is not valid for CHARACTER SET '%s'
1254	HY000	ER_SLAVE_WAS_RUNNING	Slave is already running
1255	HY000	ER_SLAVE_WAS_NOT_RUNNING	Slave already has been stopped
1256	HY000	ER_TOO_BIG_FOR_UNCOMPRESS	Uncompressed data size too large; the maximum size is %d (probably, length of uncompressed data was corrupted)
1257	HY000	ER_ZLIB_Z_MEM_ERROR	ZLIB: Not enough memory
1258	HY000	ER_ZLIB_Z_BUF_ERROR	ZLIB: Not enough room in the output buffer (probably, length of uncompressed data was corrupted)
1259	HY000	ER_ZLIB_Z_DATA_ERROR	ZLIB: Input data corrupted
1260	HY000	ER_CUT_VALUE_GROUP_CONCAT	Row %u was cut by GROUP_CONCAT()
1261	01000	ER_WARN_TOO_FEW_RECORDS	Row %ld doesn't contain data for all columns
1262	01000	ER_WARN_TOO_MANY_RECORDS	Row %ld was truncated; it contained more data than there were input columns
1263	22004	ER_WARN_NULL_TO_NOTNULL	Column set to default value; NULL supplied to NOT NULL column '%s' at row %ld
1264	22003	ER_WARN_DATA_OUT_OF_RANGE	Out of range value for column '%s' at row %ld
1265	01000	WARN_DATA_TRUNCATED	Data truncated for column '%s' at row %ld
1266	HY000	ER_WARN_USING_OTHER_HANDLER	Using storage engine %s for table '%s'
1267	HY000	ER_CANT_AGGREGATE_2COLLATIONS	Illegal mix of collations (%s,%s) and (%s,%s) for operation '%s'
1268	HY000	ER_DROP_USER	Cannot drop one or more of the requested users
1269	HY000	ER_REVOKE_GRANTS	Can't revoke all privileges for one or more of the requested users
1270	HY000	ER_CANT_AGGREGATE_3COLLATIONS	Illegal mix of collations (%s,%s), (%s,%s), (%s,%s) for operation '%s'
1271	HY000	ER_CANT_AGGREGATE_NCOLLATIONS	Illegal mix of collations for operation '%s'
1272	HY000	ER_VARIABLE_IS_NOT_STRUCT	Variable '%s' is not a variable component (can't be used as XXXX.variable_name)

Error Code	SQLSTATE	Error	Description
1273	HY000	ER_UNKNOWN_COLLATION	Unknown collation: '%s'
1274	HY000	ER_SLAVE_IGNORED_SSL_PARAMS	SSL parameters in CHANGE MASTER are ignored because this MariaDB slave was compiled without SSL support; they can be used later if MariaDB slave with SSL is started
1275	HY000	ER_SERVER_IS_IN_SECURE_AUTH_MODE	Server is running in –secure-auth mode, but '%s'@'%s' has a password in the old format; please change the password to the new format
1276	HY000	ER_WARN_FIELD_RESOLVED	Field or reference '%s%s%s%s%s' of SELECT #%d was resolved in SELECT #%d
1277	HY000	ER_BAD_SLAVE_UNTIL_COND	Incorrect parameter or combination of parameters for START SLAVE UNTIL
1278	HY000	ER_MISSING_SKIP_SLAVE	It is recommended to use –skip-slave-start when doing step-by-step replication with START SLAVE UNTIL; otherwise, you will get problems if you get an unexpected slave's mysqld restart
1279	HY000	ER_UNTIL_COND_IGNORED	SQL thread is not to be started so UNTIL options are ignored
1280	42000	ER_WRONG_NAME_FOR_INDEX	Incorrect index name '%s'
1281	42000	ER_WRONG_NAME_FOR_CATALOG	Incorrect catalog name '%s'
1282	HY000	ER_WARN_QC_RESIZE	Query cache failed to set size %lu; new query cache size is %lu
1283	HY000	ER_BAD_FT_COLUMN	Column '%s' cannot be part of FULLTEXT index
1284	HY000	ER_UNKNOWN_KEY_CACHE	Unknown key cache '%s'
1285	HY000	ER_WARN_HOSTNAME_WONT_WORK	MariaDB is started in –skip-name-resolve mode; you must restart it without this switch for this grant to work
1286	42000	ER_UNKNOWN_STORAGE_ENGINE	Unknown storage engine '%s'
1287	HY000	ER_WARN_DEPRECATED_SYNTAX	'%s' is deprecated and will be removed in a future release. Please use %s instead
1288	HY000	ER_NON_UPDATABLE_TABLE	The target table %s of the %s is not updatable
1289	HY000	ER_FEATURE_DISABLED	The '%s' feature is disabled; you need MariaDB built with '%s' to have it working
1290	HY000	ER_OPTION_PREVENTS_STATEMENT	The MariaDB server is running with the %s option so it cannot execute this statement
1291	HY000	ER_DUPLICATED_VALUE_IN_TYPE	Column '%s' has duplicated value '%s' in %s
1292	22007	ER_TRUNCATED_WRONG_VALUE	Truncated incorrect %s value: '%s'
1293	HY000	ER_TOO_MUCH_AUTO_TIMESTAMP_COLS	Incorrect table definition; there can be only one TIMESTAMP column with CURRENT_TIMESTAMP in DEFAULT or ON UPDATE clause
1294	HY000	ER_INVALID_ON_UPDATE	Invalid ON UPDATE clause for '%s' column
1295	HY000	ER_UNSUPPORTED_PS	This command is not supported in the prepared statement protocol yet

Error Code	SQLSTATE	Error	Description
1296	HY000	ER_GET_ERRMSG	Got error %d '%s' from %s
1297	HY000	ER_GET_TEMPORARY_ERRMSG	Got temporary error %d '%s' from %s
1298	HY000	ER_UNKNOWN_TIME_ZONE	Unknown or incorrect time zone: '%s'
1299	HY000	ER_WARN_INVALID_TIMESTAMP	Invalid TIMESTAMP value in column '%s' at row %ld

Error Code	SQLSTATE	Error	Description
1300	HY000	ER_INVALID_CHARACTER_STRING	Invalid %s character string: '%s'
1301	HY000	ER_WARN_ALLOWED_PACKET_OVERFLOWED	Result of %s() was larger than max_allowed_packet (%ld) - truncated
1302	HY000	ER_CONFLICTING_DECLARATIONS	Conflicting declarations: '%s%s' and '%s%s'
1303	2F003	ER_SP_NO_RECURSIVE_CREATE	Can't create a %s from within another stored routine
1304	42000	ER_SP_ALREADY_EXISTS	%s %s already exists
1305	42000	ER_SP_DOES_NOT_EXIST	%s %s does not exist
1306	HY000	ER_SP_DROP_FAILED	Failed to DROP %s %s
1307	HY000	ER_SP_STORE_FAILED	Failed to CREATE %s %s
1308	42000	ER_SP_LILABEL_MISMATCH	%s with no matching label: %s
1309	42000	ER_SP_LABEL_REDEFINE	Redefining label %s
1310	42000	ER_SP_LABEL_MISMATCH	End-label %s without match
1311	01000	ER_SP_UNINIT_VAR	Referring to uninitialized variable %s
1312	0A000	ER_SP_BADSELECT	PROCEDURE %s can't return a result set in the given context
1313	42000	ER_SP_BADRETURN	RETURN is only allowed in a FUNCTION
1314	0A000	ER_SP_BADSTATEMENT	%s is not allowed in stored procedures
1315	42000	ER_UPDATE_LOG_DEPRECATED_IGNORED	The update log is deprecated and replaced by the binary log; SET SQL_LOG_UPDATE has been ignored. This option will be removed in MariaDB 5.6.
1316	42000	ER_UPDATE_LOG_DEPRECATED_TRANSLATED	The update log is deprecated and replaced by the binary log; SET SQL_LOG_UPDATE has been translated to SET SQL_LOG_BIN. This option will be removed in MariaDB 5.6.
1317	70100	ER_QUERY_INTERRUPTED	Query execution was interrupted
1318	42000	ER_SP_WRONG_NO_OF_ARGS	Incorrect number of arguments for %s %s; expected %u, got %u
1319	42000	ER_SP_COND_MISMATCH	Undefined CONDITION: %s

Error Code	SQLSTATE	Error	Description
1320	42000	ER_SP_NORETURN	No RETURN found in FUNCTION %s
1321	2F005	ER_SP_NORETURNEND	FUNCTION %s ended without RETURN
1322	42000	ER_SP_BAD_CURSOR_QUERY	Cursor statement must be a SELECT
1323	42000	ER_SP_BAD_CURSOR_SELECT	Cursor SELECT must not have INTO
1324	42000	ER_SP_CURSOR_MISMATCH	Undefined CURSOR: %s
1325	24000	ER_SP_CURSOR_ALREADY_OPEN	Cursor is already open
1326	24000	ER_SP_CURSOR_NOT_OPEN	Cursor is not open
1327	42000	ER_SP_UNDECLARED_VAR	Undeclared variable: %s
1328	HY000	ER_SP_WRONG_NO_OF_FETCH_ARGS	Incorrect number of FETCH variables
1329	02000	ER_SP_FETCH_NO_DATA	No data - zero rows fetched, selected, or processed
1330	42000	ER_SP_DUP_PARAM	Duplicate parameter: %s
1331	42000	ER_SP_DUP_VAR	Duplicate variable: %s
1332	42000	ER_SP_DUP_COND	Duplicate condition: %s
1333	42000	ER_SP_DUP_CURS	Duplicate cursor: %s
1334	HY000	ER_SP_CANT_ALTER	Failed to ALTER %s %s
1335	0A000	ER_SP_SUBSELECT_NYI	Subquery value not supported
1336	0A000	ER_STMT_NOT_ALLOWED_IN_SF_OR_TRG	%s is not allowed in stored function or trigger
1337	42000	ER_SP_VARCOND_AFTER_CURSHNDLR	Variable or condition declaration after cursor or handler declaration
1338	42000	ER_SP_CURSOR_AFTER_HANDLER	Cursor declaration after handler declaration
1339	20000	ER_SP_CASE_NOT_FOUND	Case not found for CASE statement
1340	HY000	ER_FPARSER_TOO_BIG_FILE	Configuration file ‘%s’ is too big
1341	HY000	ER_FPARSER_BAD_HEADER	Malformed file type header in file ‘%s’
1342	HY000	ER_FPARSER_EOF_IN_COMMENT	Unexpected end of file while parsing comment ‘%s’
1343	HY000	ER_FPARSER_ERROR_IN_PARAMETER	Error while parsing parameter ‘%s’ (line: ‘%s’)
1344	HY000	ER_FPARSER_EOF_IN_UNKNOWN_PARAMETER	Unexpected end of file while skipping unknown parameter ‘%s’
1345	HY000	ER_VIEW_NO_EXPLAIN	EXPLAIN/SHOW can not be issued; lacking privileges for underlying table

Error Code	SQLSTATE	Error	Description
1346	HY000	ER_FRM_UNKNOWN_TYPE	File ‘%s’ has unknown type ‘%s’ in its header
1347	HY000	ER_WRONG_OBJECT	‘%s.%s’ is not %s
1348	HY000	ER_NONUPDATEABLE_COLUMN	Column ‘%s’ is not updatable
1349	HY000	ER_VIEW_SELECT_DERIVED	View’s SELECT contains a subquery in the FROM clause
1350	HY000	ER_VIEW_SELECT_CLAUSE	View’s SELECT contains a ‘%s’ clause
1351	HY000	ER_VIEW_SELECT_VARIABLE	View’s SELECT contains a variable or parameter
1352	HY000	ER_VIEW_SELECT_TMPTABLE	View’s SELECT refers to a temporary table ‘%s’
1353	HY000	ER_VIEW_WRONG_LIST	View’s SELECT and view’s field list have different column counts
1354	HY000	ER_WARN_VIEW_MERGE	View merge algorithm can’t be used here for now (assumed undefined algorithm)
1355	HY000	ER_WARN_VIEW_WITHOUT_KEY	View being updated does not have complete key of underlying table in it
1356	HY000	ER_VIEW_INVALID	View ‘%s.%s’ references invalid table(s) or column(s) or function(s) or definer/invoker of view lack rights to use them
1357	HY000	ER_SP_NO_DROP_SP	Can’t drop or alter a %s from within another stored routine
1358	HY000	ER_SP_GOTO_IN_HNDLR	GOTO is not allowed in a stored procedure handler
1359	HY000	ER_TRG_ALREADY_EXISTS	Trigger already exists
1360	HY000	ER_TRG_DOES_NOT_EXIST	Trigger does not exist
1361	HY000	ER_TRG_ON_VIEW_OR_TEMP_TABLE	Trigger’s ‘%s’ is view or temporary table
1362	HY000	ER_TRG_CANT_CHANGE_ROW	Updating of %s row is not allowed in %strigger
1363	HY000	ER_TRG_NO_SUCH_ROW_IN_TRG	There is no %s row in %s trigger
1364	HY000	ER_NO_DEFAULT_FOR_FIELD	Field ‘%s’ doesn’t have a default value
1365	22012	ER_DIVISION_BY_ZER	Division by 0
1366	HY000	ER_TRUNCATED_WRONG_VALUE_FOR_FIELD	Incorrect %s value: ‘%s’ for column ‘%s’ at row %ld
1367	22007	ER_ILLEGAL_VALUE_FOR_TYPE	Illegal %s ‘%s’ value found during parsing
1368	HY000	ER_VIEW_NONUPD_CHECK	CHECK OPTION on non-updatable view ‘%s.%s’
1369	HY000	ER_VIEW_CHECK_FAILED	CHECK OPTION failed ‘%s.%s’

Error Code	SQLSTATE	Error	Description
1370	42000	ER_PROCACCESS_DENIED_ERROR	%s command denied to user '%s'@'%s' for routine '%s'
1371	HY000	ER_RELAY_LOG_FAIL	Failed purging old relay logs: %s
1372	HY000	ER_PASSWD_LENGTH	Password hash should be a %d-digit hexadecimal number
1373	HY000	ER_UNKNOWN_TARGET_BINLOG	Target log not found in binlog index
1374	HY000	ER_IO_ERR_LOG_INDEX_READ	I/O error reading log index file
1375	HY000	ER_BINLOG_PURGE_PROHIBITED	Server configuration does not permit binlog purge
1376	HY000	ER_FSEEK_FAIL	Failed on fseek()
1377	HY000	ER_BINLOG_PURGE_FATAL_ERR	Fatal error during log purge
1378	HY000	ER_LOG_IN_USE	A purgeable log is in use, will not purge
1379	HY000	ER_LOG_PURGE_UNKNOWN_ERR	Unknown error during log purge
1380	HY000	ER_RELAY_LOG_INIT	Failed initializing relay log position: %s
1381	HY000	ER_NO_BINARY_LOGGING	You are not using binary logging
1382	HY000	ER_RESERVED_SYNTAX	The '%s' syntax is reserved for purposes internal to the MariaDB server
1383	HY000	ER_WSAS_FAILED	WSAStartup Failed
1384	HY000	ER_DIFF_GROUPS_PROC	Can't handle procedures with different groups yet
1385	HY000	ER_NO_GROUP_FOR_PROC	Select must have a group with this procedure
1386	HY000	ER_ORDER_WITH_PROC	Can't use ORDER clause with this procedure
1387	HY000	ER_LOGGING_PROHIBIT_CHANGING_OF	Binary logging and replication forbid changing the global server %s
1388	HY000	ER_NO_FILE_MAPPING	Can't map file: %s, errno: %d
1389	HY000	ER_WRONG_MAGIC	Wrong magic in %s
1390	HY000	ER_PS_MANY_PARAM	Prepared statement contains too many placeholders
1391	HY000	ER_KEY_PART_0	Key part '%s' length cannot be 0
1392	HY000	ER_VIEW_CHECKSUM	View text checksum failed

Error Code	SQLSTATE	Error	Description
1393	HY000	ER_VIEW_MULTIUPDATE	Can not modify more than one base table through a join view ‘%s.%s’
1394	HY000	ER_VIEW_NO_INSERT_FIELD_LIST	Can not insert into join view ‘%s.%s’ without fields list
1395	HY000	ER_VIEW_DELETE_MERGE_VIEW	Can not delete from join view ‘%s.%s’
1396	HY000	ER_CANNOT_USER	Operation %s failed for %s
1397	XAE04	ER_XAER_NOTA	XAER_NOTA: Unknown XID
1398	XAE05	ER_XAER_INVAL	XAER_INVAL: Invalid arguments (or unsupported command)
1399	XAE07	ER_XAER_RMFAIL	XAER_RMFAIL: The command cannot be executed when global transaction is in the %s state

Error Code	SQLSTATE	Error	Description
1400	XAE09	ER_XAER_OUTSIDE	XAER_OUTSIDE: Some work is done outside global transaction
1401	XAE03	ER_XAER_RMERR	XAER_RMERR: Fatal error occurred in the transaction branch - check your data for consistency
1402	XA100	ER_XA_RBROLLBACK	XA_RBROLLBACK: Transaction branch was rolled back
1403	42000	ER_NONEXISTING_PROC_GRANT	There is no such grant defined for user ‘%s’ on host ‘%s’ on routine ‘%s’
1404	HY000	ER_PROC_AUTO_GRANT_FAIL	Failed to grant EXECUTE and ALTER ROUTINE privileges
1405	HY000	ER_PROC_AUTO_REVOKE_FAIL	Failed to revoke all privileges to dropped routine
1406	22001	ER_DATA_TOO_LONG	Data too long for column ‘%s’ at row %ld
1407	42000	ER_SP_BAD_SQLSTATE	Bad SQLSTATE: ‘%s’
1408	HY000	ER_STARTUP	%s: ready for connections. Version: ‘%s’ socket: ‘%s’ port: %d %s
1409	HY000	ER_LOAD_FROM_FIXED_SIZE_ROWS_TO_VAR	Can’t load value from file with fixed size rows to variable
1410	42000	ER_CANT_CREATE_USER_WITH_GRANT	You are not allowed to create a user with GRANT
1411	HY000	ER_WRONG_VALUE_FOR_TYPE	Incorrect %s value: ‘%s’ for function %s
1412	HY000	ER_TABLE_DEF_CHANGED	Table definition has changed, please retry transaction
1413	42000	ER_SP_DUP_HANDLER	Duplicate handler declared in the same block
1414	42000	ER_SP_NOT_VAR_ARG	OUT or INOUT argument %d for routine %s is not a variable or NEW pseudo-variable in BEFORE trigger

Error Code	SQLSTATE	Error	Description
1415	0A000	ER_SP_NO_RETSET	Not allowed to return a result set from a %s
1416	22003	ER_CANT_CREATE_GEOMETRY_OBJECT	Cannot get geometry object from data you send to the GEOMETRY field
1417	HY000	ER_FAILED_ROUTINE_BREAK_BINLOG	A routine failed and has neither NO SQL nor READS SQL DATA in its declaration and binary logging is enabled; if non-transactional tables were updated, the binary log will miss their changes
1418	HY000	ER_BINLOG_UNSAFE_ROUTINE	This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled (you *might* want to use the less safe log_bin_trust_function_creators variable)
1419	HY000	ER_BINLOG_CREATE_ROUTINE_NEED_SUPER	You do not have the SUPER privilege and binary logging is enabled (you *might* want to use the less safe log_bin_trust_function_creators variable)
1420	HY000	ER_EXEC_STMT_WITH_OPEN_CURSOR	You can't execute a prepared statement which has an open cursor associated with it. Reset the statement to re-execute it.
1421	HY000	ER_STMT_HAS_NO_OPEN_CURSOR	The statement (%lu) has no open cursor.
1422	HY000	ER_COMMIT_NOT_ALLOWED_IN_SF_OR_TRG	Explicit or implicit commit is not allowed in stored function or trigger.
1423	HY000	ER_NO_DEFAULT_FOR_VIEW_FIELD	Field of view '%s.%s' underlying table doesn't have a default value
1424	HY000	ER_SP_NO_RECURSION	Recursive stored functions and triggers are not allowed.
1425	42000	ER_TOO_BIG_SCALE	Too big scale %d specified for column '%s'. Maximum is %lu.
1426	42000	ER_TOO_BIG_PRECISION	Too big precision %d specified for column '%s'. Maximum is %lu.
1427	42000	ER_M_BIGGER_THAN_D	For float(M,D, double(M,D or decimal(M,D, M must be >= D (column '%s').
1428	HY000	ER_WRONG_LOCK_OF_SYSTEM_TABLE	You can't combine write-locking of system tables with other tables or lock types
1429	HY000	ER_CONNECT_TO_FOREIGN_DATA_SOURCE	Unable to connect to foreign data source: %s
1430	HY000	ER_QUERY_ON_FOREIGN_DATA_SOURCE	There was a problem processing the query on the foreign data source. Data source error: %s
1431	HY000	ER_FOREIGN_DATA_SOURCE_DOESNT_EXIST	The foreign data source you are trying to reference does not exist. Data source error: %s
1432	HY000	ER_FOREIGN_DATA_STRING_INVALID_CANT_CREATE	Can't create federated table. The data source connection string '%s' is not in the correct format
1433	HY000	ER_FOREIGN_DATA_STRING_INVALID	The data source connection string '%s' is not in the correct format
1434	HY000	ER_CANT_CREATE_FEDERATED_TABLE	Can't create federated table. Foreign data src error: %s
1435	HY000	ER_TRG_IN_WRONG_SCHEMA	Trigger in wrong schema
1436	HY000	ER_STACK_OVERRUN_NEED_MORE	Thread stack overrun: %ld bytes used of a %ld byte stack, and %ld bytes needed. Use 'mysqld –thread_stack=#' to specify a bigger stack.

Error Code	SQLSTATE	Error	Description
1437	42000	ER_TOO_LONG_BODY	Routine body for ‘%s’ is too long
1438	HY000	ER_WARN_CANT_DROP_DEFAULT_KEYCACHE	Cannot drop default keycache
1439	42000	ER_TOO_BIG_DISPLAYWIDTH	Display width out of range for column ‘%s’ (max = %lu)
1440	XAE08	ER_XAER_DUPID	XAER_DUPID: The XID already exists
1441	22008	ER_DATETIME_FUNCTION_OVERFLOW	Datetime function: %s field overflow
1442	HY000	ER_CANT_UPDATE_USED_TABLE_IN_SF_OR_TRG	Can’t update table ‘%s’ in stored function/trigger because it is already used by statement which invoked this stored function/trigger.
1443	HY000	ER_VIEW_PREVENT_UPDATE	The definition of table ‘%s’ prevents operation %s on table ‘%s’.
1444	HY000	ER_PS_NO_RECURSION	The prepared statement contains a stored routine call that refers to that same statement. It’s not allowed to execute a prepared statement in such a recursive manner
1445	HY000	ER_SP_CANT_SET_AUTOCOMMIT	Not allowed to set autocommit from a stored function or trigger
1446	HY000	ER_MALFORMED_DEFINER	Definer is not fully qualified
1447	HY000	ER_VIEW_FRM_NO_USER	View ‘%s’.’%s’ has no definer information (old table format). Current user is used as definer. Please recreate the view!
1448	HY000	ER_VIEW_OTHER_USER	You need the SUPER privilege for creation view with ‘%s’@‘%s’ definer
1449	HY000	ER_NO_SUCH_USER	The user specified as a definer (‘%s’@‘%s’) does not exist
1450	HY000	ER_FORBID_SCHEMA_CHANGE	Changing schema from ‘%s’ to ‘%s’ is not allowed.
1451	23000	ER_ROW_IS_REFERENCED_2	Cannot delete or update a parent row: a foreign key constraint fails (%s)
1452	23000	ER_NO_REFERENCED_ROW_2	Cannot add or update a child row: a foreign key constraint fails (%s)
1453	42000	ER_SP_BAD_VAR_SHADOW	Variable ‘%s’ must be quoted with `...`, or renamed
1454	HY000	ER_TRG_NO_DEFINER	No definer attribute for trigger ‘%s’.’%s’. The trigger will be activated under the authorization of the invoker, which may have insufficient privileges. Please recreate the trigger.
1455	HY000	ER_OLD_FILE_FORMAT	‘%s’ has an old format, you should re-create the ‘%s’ object(s)
1456	HY000	ER_SP_RECURSION_LIMIT	Recursive limit %d (as set by the max_sp_recursion_depth variable) was exceeded for routine %s
1457	HY000	ER_SP_PROC_TABLE_CORRUPT	Failed to load routine %s. The table mysql.proc is missing, corrupt, or contains bad data (internal code %d)
1458	42000	ER_SP_WRONG_NAME	Incorrect routine name ‘%s’
1459	HY000	ER_TABLE_NEEDS_UPGRADE	Table upgrade required. Please do “REPAIR TABLE `%s`” or dump/reload to fix it!

Error Code	SQLSTATE	Error	Description
1460	42000	ER_SP_NO_AGGREGATE	AGGREGATE is not supported for stored functions
1461	42000	ER_MAX_PREPARED_STMT_COUNT_REACHED	Can't create more than max_prepared_stmt_count statements (current value: %lu)
1462	HY000	ER_VIEW_RECURSIVE	`%s`.`%s` contains view recursion
1463	42000	ER_NON_GROUPING_FIELD_USED	Non-grouping field '%s' is used in %s clause
1464	HY000	ER_TABLE_CANT_HANDLE_SPKEYS	The used table type doesn't support SPATIAL indexes
1465	HY000	ER_NO_TRIGGERS_ON_SYSTEM_SCHEMA	Triggers can not be created on system tables
1466	HY000	ER_REMOVED_SPACES	Leading spaces are removed from name '%s'
1467	HY000	ER_AUTOINC_READ_FAILED	Failed to read auto-increment value from storage engine
1468	HY000	ER_USERNAME	user name
1469	HY000	ER_HOSTNAME	host name
1470	HY000	ER_WRONG_STRING_LENGTH	String '%s' is too long for %s (should be no longer than %d)
1471	HY000	ER_NON_INSERTABLE_TABLE	The target table %s of the %s is not insertable-into
1472	HY000	ER_ADMIN_WRONG_MRG_TABLE	Table '%s' is differently defined or of non-MyISAM type or doesn't exist
1473	HY000	ER_TOO_HIGH_LEVEL_OF_NESTING_FOR_SELECT	Too high level of nesting for select
1474	HY000	ER_NAME_BECOMES_EMPTY	Name '%s' has become ''
1475	HY000	ER_AMBIGUOUS_FIELD_TERM	First character of the FIELDS TERMINATED string is ambiguous; please use non-optional and non-empty FIELDS ENCLOSED BY
1476	HY000	ER_FOREIGN_SERVER_EXISTS	The foreign server, %s, you are trying to create already exists.
1477	HY000	ER_FOREIGN_SERVER_DOESNT_EXIST	The foreign server name you are trying to reference does not exist. Data source error: %s
1478	HY000	ER_ILLEGAL_HA_CREATE_OPTION	Table storage engine '%s' does not support the create option '%s'
1479	HY000	ER_PARTITION_REQUIRES_VALUES_ERROR	Syntax error: %s PARTITIONING requires definition of VALUES %s for each partition
1480	HY000	ER_PARTITION_WRONG_VALUES_ERROR	Only %s PARTITIONING can use VALUES %s in partition definition
1481	HY000	ER_PARTITION_MAXVALUE_ERROR	MAXVALUE can only be used in last partition definition
1482	HY000	ER_PARTITION_SUBPARTITION_ERROR	Subpartitions can only be hash partitions and by key

Error Code	SQLSTATE	Error	Description
1483	HY000	ER_PARTITION_SUBPART_MIX_ERROR	Must define subpartitions on all partitions if on one partition
1484	HY000	ER_PARTITION_WRONG_NO_PART_ERROR	Wrong number of partitions defined, mismatch with previous setting
1485	HY000	ER_PARTITION_WRONG_NO_SUBPART_ERROR	Wrong number of subpartitions defined, mismatch with previous setting
1486	HY000	ER_CONST_EXPR_IN_PARTITION_FUNC_ERROR	Constant/Random expression in (sub) partitioning function is not allowed
1486	HY000	ER_WRONG_EXPR_IN_PARTITION_FUNC_ERROR	Constant, random or timezone-dependent expressions in (sub) partitioning function are not allowed
1487	HY000	ER_NO_CONST_EXPR_IN_RANGE_OR_LIST_ERROR	Expression in RANGE/LIST VALUES must be constant
1488	HY000	ER_FIELD_NOT_FOUND_PART_ERROR	Field in list of fields for partition function not found in table
1489	HY000	ER_LIST_OF_FIELDS_ONLY_IN_HASH_ERROR	List of fields is only allowed in KEY partitions
1490	HY000	ER_INCONSISTENT_PARTITION_INFO_ERROR	The partition info in the frm file is not consistent with what can be written into the frm file
1491	HY000	ER_PARTITION_FUNC_NOT_ALLOWED_ERROR	The %s function returns the wrong type
1492	HY000	ER_PARTITIONS_MUST_BE_DEFINED_ERROR	For %s partitions each partition must be defined
1493	HY000	ER_RANGE_NOT_INCREASING_ERROR	VALUES LESS THAN value must be strictly increasing for each partition
1494	HY000	ER_INCONSISTENT_TYPE_OF_FUNCTIONS_ERROR	VALUES value must be of same type as partition function
1495	HY000	ER_MULTIPLE_DEF_CONST_IN_LIST_PART_ERROR	Multiple definition of same constant in list partitioning
1496	HY000	ER_PARTITION_ENTRY_ERROR	Partitioning can not be used stand-alone in query
1497	HY000	ER_MIX_HANDLER_ERROR	The mix of handlers in the partitions is not allowed in this version of MariaDB
1498	HY000	ER_PARTITION_NOT_DEFINED_ERROR	For the partitioned engine it is necessary to define all %s
1499	HY000	ER_TOO_MANY_PARTITIONS_ERROR	Too many partitions (including subpartitions) were defined

Error Code	SQLSTATE	Error	Description
1500	HY000	ER_SUBPARTITION_ERROR	It is only possible to mix RANGE/LIST partitioning with HASH/KEY partitioning for subpartitioning
1501	HY000	ER_CANT_CREATE_HANDLER_FILE	Failed to create specific handler file
1502	HY000	ER_BLOB_FIELD_IN_PART_FUNC_ERROR	A BLOB field is not allowed in partition function

Error Code	SQLSTATE	Error	Description
1503	HY000	ER_UNIQUE_KEY_NEED_ALL_FIELDS_IN_PF	A %s must include all columns in the table's partitioning function
1504	HY000	ER_NO_PARTS_ERROR	Number of %s = 0 is not an allowed value
1505	HY000	ER_PARTITION_MGMT_ON_NONPARTITIONED	Partition management on a not partitioned table is not possible
1506	HY000	ER_FOREIGN_KEY_ON_PARTITIONED	Foreign key clause is not yet supported in conjunction with partitioning
1507	HY000	ER_DROP_PARTITION_NON_EXISTENT	Error in list of partitions to %s
1508	HY000	ER_DROP_LAST_PARTITION	Cannot remove all partitions, use DROP TABLE instead
1509	HY000	ER_COALESCE_ONLY_ON_HASH_PARTITION	COALESCE PARTITION can only be used on HASH/KEY partitions
1510	HY000	ER_REORG_HASH_ONLY_ON_SAME_N	REORGANIZE PARTITION can only be used to reorganize partitions not to change their numbers
1511	HY000	ER_REORG_NO_PARAM_ERROR	REORGANIZE PARTITION without parameters can only be used on auto-partitioned tables using HASH PARTITIONs
1512	HY000	ER_ONLY_ON_RANGE_LIST_PARTITION	%s PARTITION can only be used on RANGE/LIST partitions
1513	HY000	ER_ADD_PARTITION_SUBPART_ERROR	Trying to Add partition(s) with wrong number of subpartitions
1514	HY000	ER_ADD_PARTITION_NO_NEW_PARTITION	At least one partition must be added
1515	HY000	ER_COALESCE_PARTITION_NO_PARTITION	At least one partition must be coalesced
1516	HY000	ER_REORG_PARTITION_NOT_EXIST	More partitions to reorganize than there are partitions
1517	HY000	ER_SAME_NAME_PARTITION	Duplicate partition name %s
1518	HY000	ER_NO_BINLOG_ERROR	It is not allowed to shut off binlog on this command
1519	HY000	ER_CONSECUTIVE_REORG_PARTITIONS	When reorganizing a set of partitions they must be in consecutive order
1520	HY000	ER_REORG_OUTSIDE_RANGE	Reorganize of range partitions cannot change total ranges except for last partition where it can extend the range
1521	HY000	ER_PARTITION_FUNCTION_FAILURE	Partition function not supported in this version for this handler
1522	HY000	ER_PART_STATE_ERROR	Partition state cannot be defined from CREATE/ALTER TABLE
1523	HY000	ER_LIMITED_PART_RANGE	The %s handler only supports 32 bit integers in VALUES
1524	HY000	ER_PLUGIN_IS_NOT_LOADED	Plugin '%s' is not loaded
1525	HY000	ER_WRONG_VALUE	Incorrect %s value: '%s'

Error Code	SQLSTATE	Error	Description
1526	HY000	ER_NO_PARTITION_FOR_GIVEN_VALUE	Table has no partition for value %s
1527	HY000	ER_FILEGROUP_OPTION_ONLY_ONCE	It is not allowed to specify %s more than once
1528	HY000	ER_CREATE_FILEGROUP_FAILED	Failed to create %s
1529	HY000	ER_DROP_FILEGROUP_FAILED	Failed to drop %s
1530	HY000	ER_TABLESPACE_AUTO_EXTEND_ERROR	The handler doesn't support autoextend of tablespaces
1531	HY000	ER_WRONG_SIZE_NUMBER	A size parameter was incorrectly specified, either number or on the form 10M
1532	HY000	ER_SIZE_OVERFLOW_ERROR	The size number was correct but we don't allow the digit part to be more than 2 billion
1533	HY000	ER_ALTER_FILEGROUP_FAILED	Failed to alter: %s
1534	HY000	ER_BINLOG_ROW_LOGGING_FAILED	Writing one row to the row-based binary log failed
1535	HY000	ER_BINLOG_ROW_WRONG_TABLE_DEF	Table definition on master and slave does not match: %s
1536	HY000	ER_BINLOG_ROW_RBR_TO_SBR	Slave running with –log-slave-updates must use row-based binary logging to be able to replicate row-based binary log events
1537	HY000	ER_EVENT_ALREADY_EXISTS	Event '%s' already exists
1538	HY000	ER_EVENT_STORE_FAILED	Failed to store event %s. Error code %d from storage engine.
1539	HY000	ER_EVENT_DOES_NOT_EXIST	Unknown event '%s'
1540	HY000	ER_EVENT_CANT_ALTER	Failed to alter event '%s'
1541	HY000	ER_EVENT_DROP_FAILED	Failed to drop %s
1542	HY000	ER_EVENT_INTERVAL_NOT_POSITIVE_OR_TOO_BIG	INTERVAL is either not positive or too big
1543	HY000	ER_EVENT_ENDS_BEFORE_STARTS	ENDS is either invalid or before STARTS
1544	HY000	ER_EVENT_EXEC_TIME_IN_THE_PAST	Event execution time is in the past. Event has been disabled
1545	HY000	ER_EVENT_OPEN_TABLE_FAILED	Failed to open mysql.event
1546	HY000	ER_EVENT_NEITHER_M_EXPR_NOR_M_AT	No datetime expression provided
1547	HY000	ER_COL_COUNT_DOESNT_MATCH_CORRUPTED	Column count of mysql.%s is wrong. Expected %d, found %d. The table is probably corrupted
1548	HY000	ER_CANNOT_LOAD_FROM_TABLE	Cannot load from mysql.%s. The table is probably corrupted

Error Code	SQLSTATE	Error	Description
1549	HY000	ER_EVENT_CANNOT_DELETE	Failed to delete the event from mysql.event
1550	HY000	ER_EVENT_COMPILE_ERROR	Error during compilation of event's body
1551	HY000	ER_EVENT_SAME_NAME	Same old and new event name
1552	HY000	ER_EVENT_DATA_TOO_LONG	Data for column '%s' too long
1553	HY000	ER_DROP_INDEX_FK	Cannot drop index '%s': needed in a foreign key constraint
1554	HY000	ER_WARN_DEPRECATED_SYNTAX_WITH_VER	The syntax '%s' is deprecated and will be removed in MariaDB %s. Please use %s instead
1555	HY000	ER_CANT_WRITE_LOCK_LOG_TABLE	You can't write-lock a log table. Only read access is possible
1556	HY000	ER_CANT_LOCK_LOG_TABLE	You can't use locks with log tables.
1557	23000	ER_FOREIGN_DUPLICATE_KEY	Upholding foreign key constraints for table '%s', entry '%s', key %d would lead to a duplicate entry
1558	HY000	ER_COL_COUNT_DOESNT_MATCH_PLEASE_UPDATE	Column count of mysql.%s is wrong. Expected %d, found %d. Created with MariaDB %d, now running %d. Please use mysql_upgrade to fix this error.
1559	HY000	ER_TEMP_TABLE_PREVENTS_SWITCH_OUT_OF_RBR	Cannot switch out of the row-based binary log format when the session has open temporary tables
1560	HY000	ER_STORED_FUNCTION_PREVENTS_SWITCH_BINLOG_FORMAT	Cannot change the binary logging format inside a stored function or trigger
1561		ER_UNUSED_13	You should never see it
1562	HY000	ER_PARTITION_NO_TEMPORARY	Cannot create temporary table with partitions
1563	HY000	ER_PARTITION_CONST_DOMAIN_ERROR	Partition constant is out of partition function domain
1564	HY000	ER_PARTITION_FUNCTION_IS_NOT_ALLOWED	This partition function is not allowed
1565	HY000	ER_DDL_LOG_ERROR	Error in DDL log
1566	HY000	ER_NULL_IN_VALUES_LESS_THAN	Not allowed to use NULL value in VALUES LESS THAN
1567	HY000	ER_WRONG_PARTITION_NAME	Incorrect partition name
1568	25001	ER_CANT_CHANGE_TX_ISOLATION	Transaction isolation level can't be changed while a transaction is in progress
1569	HY000	ER_DUP_ENTRY_AUTOINCREMENT_CASE	ALTER TABLE causes auto_increment resequencing, resulting in duplicate entry '%s' for key '%s'
1570	HY000	ER_EVENT_MODIFY_QUEUE_ERROR	Internal scheduler error %d
1571	HY000	ER_EVENT_SET_VAR_ERROR	Error during starting/stopping of the scheduler. Error code %u

Error Code	SQLSTATE	Error	Description
1572	HY000	ER_PARTITION_MERGE_ERROR	Engine cannot be used in partitioned tables
1573	HY000	ER_CANT_ACTIVATE_LOG	Cannot activate '%s' log
1574	HY000	ER_RBR_NOT_AVAILABLE	The server was not built with row-based replication
1575	HY000	ER_BASE64_DECODE_ERROR	Decoding of base64 string failed
1576	HY000	ER_EVENT_RECURSION_FORBIDDEN	Recursion of EVENT DDL statements is forbidden when body is present
1577	HY000	ER_EVENTS_DB_ERROR	Cannot proceed because system tables used by Event Scheduler were found damaged at server start
1578	HY000	ER_ONLY_INTEGERS_ALLOWED	Only integers allowed as number here
1579	HY000	ER_UNSUPORTED_LOG_ENGINE	This storage engine cannot be used for log tables"
1580	HY000	ER_BAD_LOG_STATEMENT	You cannot '%s' a log table if logging is enabled
1581	HY000	ER_CANT_RENAME_LOG_TABLE	Cannot rename '%s'. When logging enabled, rename to/from log table must rename two tables: the log table to an archive table and another table back to '%s'
1582	42000	ER_WRONG_PARAMCOUNT_TO_NATIVE_FCT	Incorrect parameter count in the call to native function '%s'
1583	42000	ER_WRONG_PARAMETERS_TO_NATIVE_FCT	Incorrect parameters in the call to native function '%s'
1584	42000	ER_WRONG_PARAMETERS_TO_STORED_FCT	Incorrect parameters in the call to stored function '%s'
1585	HY000	ER_NATIVE_FCT_NAME_COLLISION	This function '%s' has the same name as a native function
1586	23000	ER_DUP_ENTRY_WITH_KEY_NAME	Duplicate entry '%s' for key '%s'
1587	HY000	ER_BINLOG_PURGE_EMFILE	Too many files opened, please execute the command again
1588	HY000	ER_EVENT_CANNOT_CREATE_IN_THE_PAST	Event execution time is in the past and ON COMPLETION NOT PRESERVE is set. The event was dropped immediately after creation.
1589	HY000	ER_EVENT_CANNOT_ALTER_IN_THE_PAST	Event execution time is in the past and ON COMPLETION NOT PRESERVE is set. The event was dropped immediately after creation.
1590	HY000	ER_SLAVE_INCIDENT	The incident %s occurred on the master. Message: %s
1591	HY000	ER_NO_PARTITION_FOR_GIVEN_VALUE_SILENT	Table has no partition for some existing values
1592	HY000	ER_BINLOG_UNSAFE_STATEMENT	Unsafe statement written to the binary log using statement format since BINLOG_FORMAT = STATEMENT. %s
1593	HY000	ER_SLAVE_FATAL_ERROR	Fatal error: %s
1594	HY000	ER_SLAVE_RELAY_LOG_READ_FAILURE	Relay log read failure: %s

Error Code	SQLSTATE	Error	Description
1595	HY000	ER_SLAVE_RELAY_LOG_WRITE_FAILURE	Relay log write failure: %s
1596	HY000	ER_SLAVE_CREATE_EVENT_FAILURE	Failed to create %s
1597	HY000	ER_SLAVE_MASTER_COM_FAILURE	Master command %s failed: %s
1598	HY000	ER_BINLOG_LOGGING_IMPOSSIBLE	Binary logging not possible. Message: %s
1599	HY000	ER_VIEW_NO_CREATION_CTX	View `%s`.`%s` has no creation context

Error Code	SQLSTATE	Error	Description
1600	HY000	ER_VIEW_INVALID_CREATION_CTX	Creation context of view `%s`.`%s` is invalid
1601	HY000	ER_SR_INVALID_CREATION_CTX	Creation context of stored routine `%s`.`%s` is invalid
1602	HY000	ER_TRG_CORRUPTED_FILE	Corrupted TRG file for table `%s`.`%s`
1603	HY000	ER_TRG_NO_CREATION_CTX	Triggers for table `%s`.`%s` have no creation context
1604	HY000	ER_TRG_INVALID_CREATION_CTX	Trigger creation context of table `%s`.`%s` is invalid
1605	HY000	ER_EVENT_INVALID_CREATION_CTX	Creation context of event `%s`.`%s` is invalid
1606	HY000	ER_TRG_CANT_OPEN_TABLE	Cannot open table for trigger `%s`.`%s`
1607	HY000	ER_CANT_CREATE_SROUTINE	Cannot create stored routine `%s`. Check warnings
1608		ER_UNUSED_11	You should never see it
1609	HY000	ER_NO_FORMAT_DESCRIPTION_EVENT_BEFORE_BINLOG_STATEMENT	The BINLOG statement of type `%s` was not preceded by a format description BINLOG statement.
1610	HY000	ER_SLAVE_CORRUPT_EVENT	Corrupted replication event was detected
1611	HY000	ER_LOAD_DATA_INVALID_COLUMN	Invalid column reference (%s) in LOAD DATA
1612	HY000	ER_LOG_PURGE_NO_FILE	Being purged log %s was not found
1613	XA106	ER_XA_RBTIMEOUT	XA_RBTIMEOUT: Transaction branch was rolled back: took too long
1614	XA102	ER_XA_RBDEADLOCK	XA_RBDEADLOCK: Transaction branch was rolled back: deadlock was detected
1615	HY000	ER_NEED_REPREPARE	Prepared statement needs to be re-prepared
1616	HY000	ER_DELAYED_NOT_SUPPORTED	DELAYED option not supported for table '%s'

Error Code	SQLSTATE	Error	Description
1617	HY000	WARN_NO_MASTER_INF	The master info structure does not exist
1618	HY000	WARN_OPTION_IGNORED	<%s> option ignored
1619	HY000	WARN_PLUGIN_DELETE_BUILTIN	Built-in plugins cannot be deleted
1620	HY000	WARN_PLUGIN_BUSY	Plugin is busy and will be uninstalled on shutdown
1621	HY000	ER_VARIABLE_IS_READONLY	%s variable '%s' is read-only. Use SET %s to assign the value
1622	HY000	ER_WARN_ENGINE_TRANSACTION_ROLLBACK	Storage engine %s does not support rollback for this statement. Transaction rolled back and must be restarted
1623	HY000	ER_SLAVE_HEARTBEAT_FAILURE	Unexpected master's heartbeat data: %s
1624	HY000	ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE	The requested value for the heartbeat period is either negative or exceeds the maximum allowed (%s seconds).
1625		ER_UNUSED_14	You should never see it
1626	HY000	ER_CONFLICT_FN_PARSE_ERROR	Error in parsing conflict function. Message: %s
1627	HY000	ER_EXCEPTIONS_WRITE_ERROR	Write to exceptions table failed. Message: %s"
1628	HY000	ER_TOO_LONG_TABLE_COMMENT	Comment for table '%s' is too long (max = %lu)
1629	HY000	ER_TOO_LONG_FIELD_COMMENT	Comment for field '%s' is too long (max = %lu)
1630	42000	ER_FUNC_INEXISTENT_NAME_COLLISION	FUNCTION %s does not exist. Check the 'Function Name Parsing and Resolution' section in the Reference Manual
1631	HY000	ER_DATABASE_NAME	Database
1632	HY000	ER_TABLE_NAME	Table
1633	HY000	ER_PARTITION_NAME	Partition
1634	HY000	ER_SUBPARTITION_NAME	Subpartition
1635	HY000	ER_TEMPORARY_NAME	Temporary
1636	HY000	ER_RENAMED_NAME	Renamed
1637	HY000	ER_TOO_MANY_CONCURRENT_TRXS	Too many active concurrent transactions
1638	HY000	WARN_NON_ASCII_SEPARATOR_NOT_IMPLEMENTED	Non-ASCII separator arguments are not fully supported
1639	HY000	ER_DEBUG_SYNC_TIMEOUT	debug sync point wait timed out
1640	HY000	ER_DEBUG_SYNC_HIT_LIMIT	debug sync point hit limit reached

Error Code	SQLSTATE	Error	Description
1641	42000	ER_DUP_SIGNAL_SET	Duplicate condition information item ‘%s’
1642	01000	ER_SIGNAL_WARN	Unhandled user-defined warning condition
1643	02000	ER_SIGNAL_NOT_FOUND	Unhandled user-defined not found condition
1644	HY000	ER_SIGNAL_EXCEPTION	Unhandled user-defined exception condition
1645	0K000	ER_RESIGNAL_WITHOUT_ACTIVE_HANDLER	RESIGNAL when handler not active
1646	HY000	ER_SIGNAL_BAD_CONDITION_TYPE	SIGNAL/RESIGNAL can only use a CONDITION defined with SQLSTATE
1647	HY000	WARN_COND_ITEM_TRUNCATED	Data truncated for condition item ‘%s’
1648	HY000	ER_COND_ITEM_TOO_LONG	Data too long for condition item ‘%s’
1649	HY000	ER_UNKNOWN_LOCALE	Unknown locale: ‘%s’
1650	HY000	ER_SLAVE_IGNORE_SERVER_IDS	The requested server id %d clashes with the slave startup option –replicate-same-server-id
1651	HY000	ER_QUERY_CACHE_DISABLED	Query cache is disabled; restart the server with query_cache_type=1 to enable it
1652	HY000	ER_SAME_NAME_PARTITION_FIELD	Duplicate partition field name ‘%s’
1653	HY000	ER_PARTITION_COLUMN_LIST_ERROR	Inconsistency in usage of column lists for partitioning
1654	HY000	ER_WRONG_TYPE_COLUMN_VALUE_ERROR	Partition column values of incorrect type
1655	HY000	ER_TOO_MANY_PARTITION_FUNC_FIELDS_ERROR	Too many fields in ‘%s’
1656	HY000	ER_MAXVALUE_IN_VALUES_IN	Cannot use MAXVALUE as value in VALUES IN
1657	HY000	ER_TOO_MANY_VALUES_ERROR	Cannot have more than one value for this type of %s partitioning
1658	HY000	ER_ROW_SINGLE_PARTITION_FIELD_ERROR	Row expressions in VALUES IN only allowed for multi-field column partitioning
1659	HY000	ER_FIELD_TYPE_NOT_ALLOWED_AS_PARTITION_FIELD	Field ‘%s’ is of a not allowed type for this type of partitioning
1660	HY000	ER_PARTITION_FIELDS_TOO_LONG	The total length of the partitioning fields is too large
1661	HY000	ER_BINLOG_ROW_ENGINE_AND_STMT_ENGINE	Cannot execute statement: impossible to write to binary log since both row-incapable engines and statement-incapable engines are involved.
1662	HY000	ER_BINLOG_ROW_MODE_AND_STMT_ENGINE	Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = ROW and at least one table uses a storage engine limited to statement-based logging.

Error Code	SQLSTATE	Error	Description
1663	HY000	ER_BINLOG_UNSAFE_AND_STMT_ENGINE	Cannot execute statement: impossible to write to binary log since statement is unsafe, storage engine is limited to statement-based logging, and BINLOG_FORMAT = MIXED. %s
1664	HY000	ER_BINLOG_ROW_INJECTION_AND_STMT_ENGINE	Cannot execute statement: impossible to write to binary log since statement is in row format and at least one table uses a storage engine limited to statement-based logging.
1665	HY000	ER_BINLOG_STMT_MODE_AND_ROW_ENGINE	Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEMENT and at least one table uses a storage engine limited to row-based logging.%s
1666	HY000	ER_BINLOG_ROW_INJECTION_AND_STMT_MODE	Cannot execute statement: impossible to write to binary log since statement is in row format and BINLOG_FORMAT = STATEMENT.
1667	HY000	ER_BINLOG_MULTIPLE_ENGINES_AND_SELF_LOGGING_ENGINE	Cannot execute statement: impossible to write to binary log since more than one engine is involved and at least one engine is self-logging.
1668	HY000	ER_BINLOG_UNSAFE_LIMIT	The statement is unsafe because it uses a LIMIT clause. This is unsafe because the set of rows included cannot be predicted.
1669	HY000	ER_BINLOG_UNSAFE_INSERT_DELAYED	The statement is unsafe because it uses INSERT DELAYED. This is unsafe because the times when rows are inserted cannot be predicted.
1670	HY000	ER_BINLOG_UNSAFE_SYSTEM_TABLE	The statement is unsafe because it uses the general log, slow query log, or performance_schema table(s). This is unsafe because system tables may differ on slaves.
1671	HY000	ER_BINLOG_UNSAFE_AUTOINC_COLUMNS	Statement is unsafe because it invokes a trigger or a stored function that inserts into an AUTO_INCREMENT column. Inserted values cannot be logged correctly.
1672	HY000	ER_BINLOG_UNSAFE_UDF	Statement is unsafe because it uses a UDF which may not return the same value on the slave.
1673	HY000	ER_BINLOG_UNSAFE_SYSTEM_VARIABLE	Statement is unsafe because it uses a system variable that may have a different value on the slave.
1674	HY000	ER_BINLOG_UNSAFE_SYSTEM_FUNCTION	Statement is unsafe because it uses a system function that may return a different value on the slave.
1675	HY000	ER_BINLOG_UNSAFE_NONTRANS_AFTER_TRANS	Statement is unsafe because it accesses a non-transactional table after accessing a transactional table within the same transaction.
1676	HY000	ER_MESSAGE_AND_STATEMENT	%s Statement: %s
1677	HY000	ER_SLAVE_CONVERSION_FAILED	Column %d of table '%s.%s' cannot be converted from type '%s' to type '%s'
1678	HY000	ER_SLAVE_CANT_CREATE_CONVERSION	Can't create conversion table for table '%s.%s'
1679	HY000	ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_BINLOG_FORMAT	Cannot modify @@session.binlog_format inside a transaction
1680	HY000	ER_PATH_LENGTH	The path specified for %s is too long.
1681	HY000	ER_WARN_DEPRECATED_SYNTAX_NO_REPLACEMENT	'%s' is deprecated and will be removed in a future release.
1682	HY000	ER_WRONG_NATIVE_TABLE_STRUCTURE	Native table '%s'.'%s' has the wrong structure
1683	HY000	ER_WRONG_PERFSCHEMA_USAGE	Invalid performance_schema usage.

Error Code	SQLSTATE	Error	Description
1684	HY000	ER_WARN_I_S_SKIPPED_TABLE	Table '%s'.'%s' was skipped since its definition is being modified by concurrent DDL statement
1685	HY000	ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_BINLOG_DIRECT	Cannot modify @@session.binlog_direct_non_transactional_updates inside a transaction
1686	HY000	ER_STORED_FUNCTION_PREVENTS _SWITCH_BINLOG_DIRECT	Cannot change the binlog direct flag inside a stored function or trigger
1687	42000	ER_SPATIAL_MUST_HAVE_GEOM_COL	A SPATIAL index may only contain a geometrical type column
1688	HY000	ER_TOO_LONG_INDEX_COMMENT	Comment for index '%s' is too long (max = %lu)
1689	HY000	ER_LOCK_ABORTED	Wait on a lock was aborted due to a pending exclusive lock
1690	22003	ER_DATA_OUT_OF_RANGE	%s value is out of range in '%s'
1691	HY000	ER_WRONG_SPVAR_TYPE_IN_LIMIT	A variable of a non-integer based type in LIMIT clause
1692	HY000	ER_BINLOG_UNSAFE_MULTIPLE_ENGINES_AND_SELF_LOGGING_ENGINE	Mixing self-logging and non-self-logging engines in a statement is unsafe.
1693	HY000	ER_BINLOG_UNSAFE_MIXED_STATEMENT	Statement accesses nontransactional table as well as transactional or temporary table, and writes to any of them.
1694	HY000	ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_SQL_LOG_BIN	Cannot modify @@session.sql_log_bin inside a transaction
1695	HY000	ER_STORED_FUNCTION_PREVENTS_SWITCH_SQL_LOG_BIN	Cannot change the sql_log_bin inside a stored function or trigger
1696	HY000	ER_FAILED_READ_FROM_PAR_FILE	Failed to read from the .par file
1697	HY000	ER_VALUES_IS_NOT_INT_TYPE_ERROR	VALUES value for partition '%s' must have type INT
1698	28000	ER_ACCESS_DENIED_NO_PASSWORD_ERROR	Access denied for user '%s'@'%s'
1699	HY000	ER_SET_PASSWORD_AUTH_PLUGIN	SET PASSWORD has no significance for users authenticating via plugins

Error Code	SQLSTATE	Error	Description
1700	HY000	ER_GRANT_PLUGIN_USER_EXISTS	GRANT with IDENTIFIED WITH is illegal because the user %-.*s already exists
1701	42000	ER_TRUNCATE_ILLEGAL_FK	Cannot truncate a table referenced in a foreign key constraint (%s)
1702	HY000	ER_PLUGIN_IS_PERMANENT	Plugin '%s' is force_plus_permanent and can not be unloaded
1703	HY000	ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE_MIN	The requested value for the heartbeat period is less than 1 millisecond. The value is reset to 0, meaning that heartbeating will effectively be disabled.

Error Code	SQLSTATE	Error	Description
1704	HY000	ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE_MAX	The requested value for the heartbeat period exceeds the value of slave_net_timeout seconds. A sensible value for the period should be less than the timeout.
1705	HY000	ER_STMT_CACHE_FULL	Multi-row statements required more than 'max_binlog_stmt_cache_size' bytes of storage; increase this mysqld variable and try again
1706	HY000	ER_MULTI_UPDATE_KEY_CONFLICT	Primary key/partition key update is not allowed since the table is updated both as '%s' and '%s'.
1707	HY000	ER_TABLE_NEEDS_REBUILD	Table rebuild required. Please do "ALTER TABLE `%s` FORCE" or dump/reload to fix it!
1708	HY000	WARN_OPTION_BELOW_LIMIT	The value of '%s' should be no less than the value of '%s'
1709	HY000	ER_INDEX_COLUMN_TOO_LONG	Index column size too large. The maximum column size is %lu bytes.
1710	HY000	ER_ERROR_IN_TRIGGER_BODY	Trigger '%s' has an error in its body: '%s'
1711	HY000	ER_ERROR_IN_UNKNOWN_TRIGGER_BODY	Unknown trigger has an error in its body: '%s'
1712	HY000	ER_INDEX_CORRUPT	Index %s is corrupted
1713	HY000	ER_UNDO_RECORD_TOO_BIG	Undo log record is too big.
1714	HY000	ER_BINLOG_UNSAFE_INSERT_IGNORE_SELECT	INSERT IGNORE… SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are ignored. This order cannot be predicted and may differ on master and the slave.
1715	HY000	ER_BINLOG_UNSAFE_INSERT_SELECT_UPDATE	INSERT… SELECT… ON DUPLICATE KEY UPDATE is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are updated. This order cannot be predicted and may differ on master and the slave.
1716	HY000	ER_BINLOG_UNSAFE_REPLACE_SELECT	REPLACE… SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are replaced. This order cannot be predicted and may differ on master and the slave.
1717	HY000	ER_BINLOG_UNSAFE_CREATE_IGNORE_SELECT	CREATE… IGNORE SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are ignored. This order cannot be predicted and may differ on master and the slave.
1718	HY000	ER_BINLOG_UNSAFE_CREATE_REPLACE_SELECT	CREATE… REPLACE SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are replaced. This order cannot be predicted and may differ on master and the slave.
1719	HY000	ER_BINLOG_UNSAFE_UPDATE_IGNORE	UPDATE IGNORE is unsafe because the order in which rows are updated determines which (if any) rows are ignored. This order cannot be predicted and may differ on master and the slave.
1720		ER_UNUSED_15	You should never see it
1721		ER_UNUSED_16	You should never see it
1722	HY000	ER_BINLOG_UNSAFE_WRITE_AUTOINC_SELECT	Statements writing to a table with an auto-increment column after selecting from another table are unsafe because the order in which rows are retrieved determines what (if any) rows will be written. This order cannot be predicted and may differ on master and the slave.

Error Code	SQLSTATE	Error	Description
1723	HY000	ER_BINLOG_UNSAFE_CREATE_SELECT_AUTOINC	CREATE TABLE… SELECT… on a table with an auto-increment column is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are inserted. This order cannot be predicted and may differ on master and the slave.
1724	HY000	ER_BINLOG_UNSAFE_INSERT_TWO_KEYS	INSERT… ON DUPLICATE KEY UPDATE on a table with more than one UNIQUE KEY is unsafe
1725	HY000	ER_TABLE_IN_FK_CHECK	Table is being used in foreign key check.
1726	HY000	ER_UNSUPPORTED_ENGINE	Storage engine ‘%s’ does not support system tables. [%s.%s]
1727	HY000	ER_BINLOG_UNSAFE_AUTOINC_NOT_FIRST	INSERT into autoincrement field which is not the first part in the composed primary key is unsafe.
1728	HY000	ER_CANNOT_LOAD_FROM_TABLE_V2	Cannot load from %s.%s. The table is probably corrupted
1729	HY000	ER_MASTER_DELAY_VALUE_OUT_OF_RANGE	The requested value %s for the master delay exceeds the maximum %u
1730	HY000	ER_ONLY_FD_AND_RBR_EVENTS_ALLOWED_IN_BINLOG_STATEMENT	Only Format_description_log_event and row events are allowed in BINLOG statements (but %s was provided
1731	HY000	ER_PARTITION_EXCHANGE_DIFFERENT_OPTION	Non matching attribute ‘%s’ between partition and table
1732	HY000	ER_PARTITION_EXCHANGE_PART_TABLE	Table to exchange with partition is partitioned: ‘%s’
1733	HY000	ER_PARTITION_EXCHANGE_TEMP_TABLE	Table to exchange with partition is temporary: ‘%s’
1734	HY000	ER_PARTITION_INSTEAD_OF_SUBPARTITION	Subpartitioned table, use subpartition instead of partition
1735	HY000	ER_UNKNOWN_PARTITION	Unknown partition ‘%s’ in table ‘%s’
1736	HY000	ER_TABLES_DIFFERENT_METADATA	Tables have different definitions
1737	HY000	ER_ROW_DOES_NOT_MATCH_PARTITION	Found a row that does not match the partition
1738	HY000	ER_BINLOG_CACHE_SIZE_GREATER_THAN_MAX	Option binlog_cache_size (%lu) is greater than max_binlog_cache_size (%lu); setting binlog_cache_size equal to max_binlog_cache_size.
1739	HY000	ER_WARN_INDEX_NOT_APPLICABLE	Cannot use %s access on index ‘%s’ due to type or collation conversion on field ‘%s’
1740	HY000	ER_PARTITION_EXCHANGE_FOREIGN_KEY	Table to exchange with partition has foreign key references: ‘%s’
1741	HY000	ER_NO_SUCH_KEY_VALUE	Key value ‘%s’ was not found in table ‘%s.%s’
1742	HY000	ER_RPL_INFO_DATA_TOO_LONG	Data for column ‘%s’ too long
1743	HY000	ER_NETWORK_READ_EVENT_CHECKSUM_FAILURE	Replication event checksum verification failed while reading from network.

Error Code	SQL STATE	Error	Description
1744	HY000	ER_BINLOG_READ_EVENT_CHECKSUM_FAILURE	Replication event checksum verification failed while reading from a log file.
1745	HY000	ER_BINLOG_STMT_CACHE_SIZE_GREATER_THAN_MAX	Option binlog_stmt_cache_size (%lu) is greater than max_binlog_stmt_cache_size (%lu); setting binlog_stmt_cache_size equal to max_binlog_stmt_cache_size.
1746	HY000	ER_CANT_UPDATE_TABLE_IN_CREATE_TABLE_SELECT	Can't update table '%s' while '%s' is being created.
1747	HY000	ER_PARTITION_CLAUSE_ON_NONPARTITIONED	PARTITION () clause on non partitioned table
1748	HY000	ER_ROW_DOES_NOT_MATCH_GIVEN_PARTITION_SET	Found a row not matching the given partition set
1749	HY000	ER_NO_SUCH_PARTITION_UNUSED	partition '%s' doesn't exist
1750	HY000	ER_CHANGE_RPL_INFO_REPOSITORY_FAILURE	Failure while changing the type of replication repository: %s.
1751	HY000	ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_CREATED_TEMP_TABLE	The creation of some temporary tables could not be rolled back.
1752	HY000	ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_DROPPED_TEMP_TABLE	Some temporary tables were dropped, but these operations could not be rolled back.
1753	HY000	ER_MTS_FEATURE_IS_NOT_SUPPORTED	%s is not supported in multi-threaded slave mode. %s
1754	HY000	ER_MTS_UPDATED_DBS_GREATER_MAX	The number of modified databases exceeds the maximum %d; the database names will not be included in the replication event metadata.
1755	HY000	ER_MTS_CANT_PARALLEL	Cannot execute the current event group in the parallel mode. Encountered event %s, relay-log name %s, position %s which prevents execution of this event group in parallel mode. Reason: %s.
1756	HY000	ER_MTS_INCONSISTENT_DATA	%s
1757	HY000	ER_FULLTEXT_NOT_SUPPORTED_WITH_PARTITIONING	FULLTEXT index is not supported for partitioned tables.
1758	35000	ER_DA_INVALID_CONDITION_NUMBER	Invalid condition number
1759	HY000	ER_INSECURE_PLAIN_TEXT	Sending passwords in plain text without SSL/TLS is extremely insecure.
1760	HY000	ER_INSECURE_CHANGE_MASTER	Storing MariaDB user name or password information in the master.info repository is not secure and is therefore not recommended. Please see the MariaDB Manual for more about this issue and possible alternatives
1761	23000	ER_FOREIGN_DUPLICATE_KEY_WITH_CHILD_INFO	Foreign key constraint for table '%s', record '%s' would lead to a duplicate entry in table '%s', key '%s'
1762	23000	ER_FOREIGN_DUPLICATE_KEY_WITHOUT_CHILD_INFO	Foreign key constraint for table '%s', record '%s' would lead to a duplicate entry in a child table
1763	HY000	ER_SQLTHREAD_WITH_SECURE_SLAVE	Setting authentication options is not possible when only the Slave SQL Thread is being started.
1764	HY000	ER_TABLE_HAS_NO_FT	The table does not have FULLTEXT index to support this query

Error Code	SQLSTATE	Error	Description
1765	HY000	ER_VARIABLE_NOT_SETTABLE_IN_SF_OR_TRIGGER	The system variable %s cannot be set in stored functions or triggers.
1766	HY000	ER_VARIABLE_NOT_SETTABLE_IN_TRANSACTION	The system variable %s cannot be set when there is an ongoing transaction.
1767	HY000	ER_GTID_NEXT_IS_NOT_IN_GTID_NEXT_LIST	The system variable @@SESSION.GTID_NEXT has the value %s, which is not listed in @@SESSION.GTID_NEXT_LIST.
1768	HY000	ER_CANT_CHANGE_GTID_NEXT_IN_TRANSACTION_WHEN_GTID_NEXT_LIST_IS_NULL	The system variable @@SESSION.GTID_NEXT cannot change inside a transaction.
1769	HY000	ER_SET_STATEMENT_CANNOT_INVOKE_FUNCTION	The statement ‘SET %s’ cannot invoke a stored function.
1770	HY000	ER_GTID_NEXT_CANT_BE_AUTOMATIC_IF_GTID_NEXT_LIST_IS_NON_NULL	The system variable @@SESSION.GTID_NEXT cannot be ‘AUTOMATIC’ when @@SESSION.GTID_NEXT_LIST is non-NULL.
1771	HY000	ER_SKIPPING_LOGGED_TRANSACTION	Skipping transaction %s because it has already been executed and logged.
1772	HY000	ER_MALFORMED_GTID_SET_SPECIFICATION	Malformed GTID set specification ‘%s’.
1773	HY000	ER_MALFORMED_GTID_SET_ENCODING	Malformed GTID set encoding.
1774	HY000	ER_MALFORMED_GTID_SPECIFICATION	Malformed GTID specification ‘%s’.
1775	HY000	ER_GNO_EXHAUSTED	Impossible to generate Global Transaction Identifier: the integer component reached the maximal value. Restart the server with a new server_uuid.
1776	HY000	ER_BAD_SLAVE_AUTO_POSITION	Parameters MASTER_LOG_FILE, MASTER_LOG_POS, RELAY_LOG_FILE and RELAY_LOG_POS cannot be set when MASTER_AUTO_POSITION is active.
1777	HY000	ER_AUTO_POSITION_REQUIRES_GTID_MODE_ON	CHANGE MASTER TO MASTER_AUTO_POSITION = 1 can only be executed when @@GLOBAL.GTID_MODE = ON.
1778	HY000	ER_CANT_DO_IMPLICIT_COMMIT_IN_TRX_WHEN_GTID_NEXT_IS_SET	Cannot execute statements with implicit commit inside a transaction when @@SESSION.GTID_NEXT != AUTOMATIC.
1779	HY000	ER_GTID_MODE_2_OR_3_REQUIRES_DISABLE_GTID_UNSAFE_STATEMENTS_ON	GTID_MODE = ON or GTID_MODE = UPGRADE_STEP_2 requires DISABLE_GTID_UNSAFE_STATEMENTS = 1.
1779	HY000	ER_GTID_MODE_2_OR_3_REQUIRES_ENFORCE_GTID_CONSISTENCY_ON	@@GLOBAL.GTID_MODE = ON or UPGRADE_STEP_2 requires @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1.
1780	HY000	ER_GTID_MODE_REQUIRES_BINLOG	@@GLOBAL.GTID_MODE = ON or UPGRADE_STEP_1 or UPGRADE_STEP_2 requires –log-bin and –log-slave-updates.
1781	HY000	ER_CANT_SET_GTID_NEXT_TO_GTID_WHEN_GTID_MODE_IS_OFF	@@SESSION.GTID_NEXT cannot be set to UUID:NUMBER when @@GLOBAL.GTID_MODE = OFF.
1782	HY000	ER_CANT_SET_GTID_NEXT_TO_ANONYMOUS_WHEN_GTID_MODE_IS_ON	@@SESSION.GTID_NEXT cannot be set to ANONYMOUS when @@GLOBAL.GTID_MODE = ON.

Error Code	SQLSTATE	Error	Description
1783	HY000	ER_CANT_SET_GTID_NEXT_LIST_TO_NON_NULL_WHEN_GTID_MODE_IS_OFF	@@SESSION.GTID_NEXT_LIST cannot be set to a non-NULL value when @@GLOBAL.GTID_MODE = OFF.
1784	HY000	ER_FOUND_GTID_EVENT_WHEN_GTID_MODE_IS_OFF	Found a Gtid_log_event or Previous_gtids_log_event when @@GLOBAL.GTID_MODE = OFF.
1785	HY000	ER_GTID_UNSAFE_NON_TRANSACTIONAL_TABLE	When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, updates to non-transactional tables can only be done in either autocommitted statements or single-statement transactions, and never in the same statement as updates to transactional tables.
1786	HY000	ER_GTID_UNSAFE_CREATE_SELECT	CREATE TABLE … SELECT is forbidden when @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1.
1787	HY000	ER_GTID_UNSAFE_CREATE_DROP_TEMPORARY_TABLE_IN_TRANSACTION	When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, the statements CREATE TEMPORARY TABLE and DROP TEMPORARY TABLE can be executed in a non-transactional context only, and require that AUTOCOMMIT = 1.
1788	HY000	ER_GTID_MODE_CAN_ONLY_CHANGE_ONE_STEP_AT_A_TIME	The value of @@GLOBAL.GTID_MODE can only change one step at a time: OFF <-> UPGRADE_STEP_1 <-> UPGRADE_STEP_2 <-> ON. Also note that this value must be stepped up or down simultaneously on all servers; see the Manual for instructions.
1789	HY000	ER_MASTER_HAS_PURGED_REQUIRED_GTIDS	The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the master has purged binary logs containing GTIDs that the slave requires.
1790	HY000	ER_CANT_SET_GTID_NEXT_WHEN_OWNING_GTID	@@SESSION.GTID_NEXT cannot be changed by a client that owns a GTID. The client owns %s. Ownership is released on COMMIT or ROLLBACK.
1791	HY000	ER_UNKNOWN_EXPLAIN_FORMAT	Unknown EXPLAIN format name: ‘%s’
1792	25006	ER_CANT_EXECUTE_IN_READ_ONLY_TRANSACTION	Cannot execute statement in a READ ONLY transaction.
1793	HY000	ER_TOO_LONG_TABLE_PARTITION_COMMENT	Comment for table partition ‘%s’ is too long (max = %lu
1794	HY000	ER_SLAVE_CONFIGURATION	Slave is not configured or failed to initialize properly. You must at least set –server-id to enable either a master or a slave. Additional error messages can be found in the MariaDB error log
1795	HY000	ER_INNODB_FT_LIMIT	InnoDB presently supports one FULLTEXT index creation at a time
1796	HY000	ER_INNODB_NO_FT_TEMP_TABLE	Cannot create FULLTEXT index on temporary InnoDB table
1797	HY000	ER_INNODB_FT_WRONG_DOCID_COLUMN	Column ‘%s’ is of wrong type for an InnoDB FULLTEXT index
1798	HY000	ER_INNODB_FT_WRONG_DOCID_INDEX	Index ‘%s’ is of wrong type for an InnoDB FULLTEXT index
1799	HY000	ER_INNODB_ONLINE_LOG_TOO_BIG	Creating index ‘%s’ required more than ‘innodb_online_alter_log_max_size’ bytes of modification log. Please try again.

Error Code	SQLSTATE	Error	Description
1800	HY000	ER_UNKNOWN_ALTER_ALGORITHM	Unknown ALGORITHM ‘%s’

Error Code	SQLSTATE	Error	Description
1801	HY000	ER_UNKNOWN_ALTER_LOCK	Unknown LOCK type '%s'
1802	HY000	ER_MTS_CHANGE_MASTER_CANT_RUN_WITH_GAPS	CHANGE MASTER cannot be executed when the slave was stopped with an error or killed in MTS mode. Consider using RESET SLAVE or START SLAVE UNTIL.
1803	HY000	ER_MTS_RECOVERY_FAILURE	Cannot recover after SLAVE errored out in parallel execution mode. Additional error messages can be found in the MariaDB error log
1804	HY000	ER_MTS_RESET_WORKERS	Cannot clean up worker info tables. Additional error messages can be found in the MariaDB error log
1805	HY000	ER_COL_COUNT_DOESNT_MATCH_CORRUPTED_V2	Column count of %s.%s is wrong. Expected %d, found %d. The table is probably corrupted
1806	HY000	ER_SLAVE_SILENT_RETRY_TRANSACTION	Slave must silently retry current transaction
1807	HY000	ER_DISCARD_FK_CHECKS_RUNNING	There is a foreign key check running on table '%s'. Cannot discard the table.
1808	HY000	ER_TABLE_SCHEMA_MISMATCH	Schema mismatch (%s
1809	HY000	ER_TABLE_IN_SYSTEM_TABLESPACE	Table '%s' in system tablespace
1810	HY000	ER_IO_READ_ERROR	IO Read error: (%lu, %s) %s
1811	HY000	ER_IO_WRITE_ERROR	IO Write error: (%lu, %s) %s
1812	HY000	ER_TABLESPACE_MISSING	Tablespace is missing for table '%s'
1813	HY000	ER_TABLESPACE_EXISTS	Tablespace for table '%s' exists. Please DISCARD the tablespace before IMPORT.
1814	HY000	ER_TABLESPACE_DISCARDED	Tablespace has been discarded for table '%s'
1815	HY000	ER_INTERNAL_ERROR	Internal error: %s
1816	HY000	ER_INNODB_IMPORT_ERROR	ALTER TABLE '%s' IMPORT TABLESPACE failed with error %lu : '%s'
1817	HY000	ER_INNODB_INDEX_CORRUPT	Index corrupt: %s
1818	HY000	ER_INVALID_YEAR_COLUMN_LENGTH	YEAR(%lu) column type is deprecated. Creating YEAR(4) column instead.
1819	HY000	ER_NOT_VALID_PASSWORD	Your password does not satisfy the current policy requirements

Error Code	SQLSTATE	Error	Description
1820	HY000	ER_MUST_CHANGE_PASSWORD	You must SET PASSWORD before executing this statement
1821	HY000	ER_FK_NO_INDEX_CHILD	Failed to add the foreign key constraint. Missing index for constraint ‘%s’ in the foreign table ‘%s’
1822	HY000	ER_FK_NO_INDEX_PARENT	Failed to add the foreign key constraint. Missing index for constraint ‘%s’ in the referenced table ‘%s’
1823	HY000	ER_FK_FAIL_ADD_SYSTEM	Failed to add the foreign key constraint ‘%s’ to system tables
1824	HY000	ER_FK_CANNOT_OPEN_PARENT	Failed to open the referenced table ‘%s’
1825	HY000	ER_FK_INCORRECT_OPTION	Failed to add the foreign key constraint on table ‘%s’. Incorrect options in FOREIGN KEY constraint ‘%s’
1826	HY000	ER_FK_DUP_NAME	Duplicate foreign key constraint name ‘%s’
1827	HY000	ER_PASSWORD_FORMAT	The password hash doesn’t have the expected format. Check if the correct password algorithm is being used with the PASSWORD() function.
1828	HY000	ER_FK_COLUMN_CANNOT_DROP	Cannot drop column ‘%s’: needed in a foreign key constraint ‘%s’
1829	HY000	ER_FK_COLUMN_CANNOT_DROP_CHILD	Cannot drop column ‘%s’: needed in a foreign key constraint ‘%s’ of table ‘%s’
1830	HY000	ER_FK_COLUMN_NOT_NULL	Column ‘%s’ cannot be NOT NULL: needed in a foreign key constraint ‘%s’ SET NULL
1831	HY000	ER_DUP_INDEX	Duplicate index ‘%s’ defined on the table ‘%s.%s’. This is deprecated and will be disallowed in a future release.
1832	HY000	ER_FK_COLUMN_CANNOT_CHANGE	Cannot change column ‘%s’: used in a foreign key constraint ‘%s’
1833	HY000	ER_FK_COLUMN_CANNOT_CHANGE_CHILD	Cannot change column ‘%s’: used in a foreign key constraint ‘%s’ of table ‘%s’
1834	HY000	ER_FK_CANNOT_DELETE_PARENT	Cannot delete rows from table which is parent in a foreign key constraint ‘%s’ of table ‘%s’
1835	HY000	ER_MALFORMED_PACKET	Malformed communication packet.
1836	HY000	ER_READ_ONLY_MODE	Running in read-only mode
1837	HY000	ER_GTID_NEXT_TYPE_UNDEFINED_GROUP	When GTID_NEXT is set to a GTID, you must explicitly set it again after a COMMIT or ROLLBACK. If you see this error message in the slave SQL thread, it means that a table in the current transaction is transactional on the master and non-transactional on the slave. In a client connection, it means that you executed SET GTID_NEXT before a transaction and forgot to set GTID_NEXT to a different identifier or to ‘AUTOMATIC’ after COMMIT or ROLLBACK. Current GTID_NEXT is ‘%s’
1838	HY000	ER_VARIABLE_NOT_SETTABLE_IN_SP	The system variable %s cannot be set in stored procedures.

Error Code	SQLSTATE	Error	Description
1839	HY000	ER_CANT_SET_GTID_PURGED_WHEN_GTID_MODE_IS_OFF	@@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_MODE = ON.
1840	HY000	ER_CANT_SET_GTID_PURGED_WHEN_GTID_EXECUTED_IS_NOT_EMPTY	@@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty.
1841	HY000	ER_CANT_SET_GTID_PURGED_WHEN_OWNED_GTIDS_IS_NOT_EMPTY	@@GLOBAL.GTID_PURGED can only be set when there are no ongoing transactions (not even in other clients).
1842	HY000	ER_GTID_PURGED_WAS_CHANGED	@@GLOBAL.GTID_PURGED was changed from ‘%s’ to ‘%s’.
1843	HY000	ER_GTID_EXECUTED_WAS_CHANGED	@@GLOBAL.GTID_EXECUTED was changed from ‘%s’ to ‘%s’.
1844	HY000	ER_BINLOG_STMT_MODE_AND_NO_REPL_TABLES	Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEMENT, and both replicated and non replicated tables are written to.
1845	0A000	ER_ALTER_OPERATION_NOT_SUPPORTED	%s is not supported for this operation. Try %s.
1846	0A000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON	%s is not supported. Reason: %s. Try %s.
1847	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COPY	COPY algorithm requires a lock
1848	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_PARTITION	Partition specific operations do not yet support LOCK/ALGORITHM
1849	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_RENAME	Columns participating in a foreign key are renamed
1850	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COLUMN_TYPE	Cannot change column type INPLACE

Error Code	SQLSTATE	Error	Description
1851	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_CHECK	Adding foreign keys needs foreign_key_checks=OFF
1852	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_IGNORE	Creating unique indexes with IGNORE requires COPY algorithm to remove duplicate rows
1853	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOPK	Dropping a primary key is not allowed without also adding a new primary key
1854	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_AUTOINC	Adding an auto-increment column requires a lock
1855	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_HIDDEN_FTS	Cannot replace hidden FTS_DOC_ID with a user-visible one
1856	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_CHANGE_FTS	Cannot drop or rename FTS_DOC_ID
1857	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FTS	Fulltext index creation requires a lock
1858	HY000	ER_SQL_SLAVE_SKIP_COUNTER_NOT_SETTABLE_IN_GTID_MODE	sql_slave_skip_counter can not be set when the server is running with @@GLOBAL.GTID_MODE = ON. Instead, for each transaction that you want to skip, generate an empty transaction with the same GTID as the transaction
1859	23000	ER_DUP_UNKNOWN_IN_INDEX	Duplicate entry for key '%s'
1860	HY000	ER_IDENT_CAUSES_TOO_LONG_PATH	Long database name and identifier for object resulted in path length exceeding %d characters. Path: '%s'.
1861	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOT_NULL	cannot silently convert NULL values, as required in this SQL_MODE
1862	HY000	ER_MUST_CHANGE_PASSWORD_LOGIN	Your password has expired. To log in you must change it using a client that supports expired passwords.
1863	HY000	ER_ROW_IN_WRONG_PARTITION	Found a row in wrong partition %s

Error Code	SQLSTATE	Error	Description
1864	HY000	ER_MTS_EVENT_BIGGER_PENDING_JOBS_SIZE_MAX	Cannot schedule event %s, relay-log name %s, position %s to Worker thread because its size %lu exceeds %lu of slave_pending_jobs_size_max.
1865	HY000	ER_INNODB_NO_FT_USES_PARSER	Cannot CREATE FULLTEXT INDEX WITH PARSER on InnoDB table
1866	HY000	ER_BINLOG_LOGICAL_CORRUPTION	The binary log file '%s' is logically corrupted: %s
1867	HY000	ER_WARN_PURGE_LOG_IN_USE	file %s was not purged because it was being read by %d thread(s), purged only %d out of %d files.
1868	HY000	ER_WARN_PURGE_LOG_IS_ACTIVE	file %s was not purged because it is the active log file.
1869	HY000	ER_AUTO_INCREMENT_CONFLICT	Auto-increment value in UPDATE conflicts with internally generated values
1870	HY000	WARN_ON_BLOCKHOLE_IN_RBR	Row events are not logged for %s statements that modify BLACKHOLE tables in row format. Table(s): '%s'
1871	HY000	ER_SLAVE_MI_INIT_REPOSITORY	Slave failed to initialize master info structure from the repository
1872	HY000	ER_SLAVE_RLI_INIT_REPOSITORY	Slave failed to initialize relay log info structure from the repository
1873	28000	ER_ACCESS_DENIED_CHANGE_USER_ERROR	Access denied trying to change to user '%s'@'%s' (using password: %s). Disconnecting.
1874	HY000	ER_INNODB_READ_ONLY	InnoDB is in read only mode.
1875	HY000	ER_STOP_SLAVE_SQL_THREAD_TIMEOUT	STOP SLAVE command execution is incomplete: Slave SQL thread got the stop signal, thread is busy, SQL thread will stop once the current task is complete.
1876	HY000	ER_STOP_SLAVE_IO_THREAD_TIMEOUT	STOP SLAVE command execution is incomplete: Slave IO thread got the stop signal, thread is busy, IO thread will stop once the current task is complete.
1877	HY000	ER_TABLE_CORRUPT	Operation cannot be performed. The table '%s.%s' is missing, corrupt or contains bad data.
1878	HY000	ER_TEMP_FILE_WRITE_FAILURE	Temporary file write failure.
1879	HY000	ER_INNODB_FT_AUX_NOT_HEX_ID	Upgrade index name failed, please use create index(alter table) algorithm copy to rebuild index.
1880		ER_LAST_MYSQL_ERROR_MESSAGE	''

MariaDB-specific error codes

Error Code	SQLSTATE	Error	Description
1900		ER_UNUSED_18	”
1901		ER_GENERATED_COLUMN_FUNCTION_IS_NOT_ALLOWED	Function or expression ‘%s’ cannot be used in the %s clause of %`s
1902		ER_UNUSED_19	”
1903		ER_PRIMARY_KEY_BASED_ON_GENERATED_COLUMN	Primary key cannot be defined upon a generated column
1904		ER_KEY_BASED_ON_GENERATED_VIRTUAL_COLUMN	Key/Index cannot be defined on a virtual generated column
1905		ER_WRONG_FK_OPTION_FOR_GENERATED_COLUMN	Cannot define foreign key with %s clause on a generated column
1906		ER_WARNING_NON_DEFAULT_VALUE_FOR_GENERATED_COLUMN	The value specified for generated column ‘%s’ in table ‘%s’ has been ignored
1907		ER_UNSUPPORTED_ACTION_ON_GENERATED_COLUMN	This is not yet supported for generated columns
1908		ER_UNUSED_20	”
1909		ER_UNUSED_21	”
1910		ER_UNSUPPORTED_ENGINE_FOR_GENERATED_COLUMNS	%s storage engine does not support generated columns
1911		ER_UNKNOWN_OPTION	Unknown option ‘%-.64s’
1912		ER_BAD_OPTION_VALUE	Incorrect value ‘%-.64s’ for option ‘%-.64s’
1913		ER_UNUSED_6	You should never see it
1914		ER_UNUSED_7	You should never see it
1915		ER_UNUSED_8	You should never see it
1916		ER_DATA_OVERFLOW 22003	Got overflow when converting ‘%-.128s’ to %-.32s. Value truncated.
1917		ER_DATA_TRUNCATED 22003	Truncated value ‘%-.128s’ when converting to %-.32s
1918		ER_BAD_DATA 22007	Encountered illegal value ‘%-.128s’ when converting to %-.32s
1919		ER_DYN_COL_WRONG_FORMAT	Encountered illegal format of dynamic column string
1920		ER_DYN_COL_IMPLEMENTATION_LIMIT	Dynamic column implementation limit reached
1921		ER_DYN_COL_DATA 22007	Illegal value used as argument of dynamic column function
1922		ER_DYN_COL_WRONG_CHARSET	Dynamic column contains unknown character set

Error Code	SQLSTATE	Error	Description
1923		ER_ILLEGAL_SUBQUERY_OPTIMIZER_SWITCHES	At least one of the 'in_to_exists' or 'materialization' optimizer_switch flags must be 'on'.
1924		ER_QUERY_CACHE_IS_DISABLED	Query cache is disabled (resize or similar command in progress); repeat this command later
1925		ER_QUERY_CACHE_IS_GLOBALY_DISABLED	Query cache is globally disabled and you can't enable it only for this session
1926		ER_VIEW_ORDERBY_IGNORED	View '%-.192s'.'%-.192s' ORDER BY clause ignored because there is other ORDER BY clause already.
1927	70100	ER_CONNECTION_KILLED	Connection was killed
1928		ER_UNUSED_11	You should never see it
1929		ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_SKIP_REPLICATION	Cannot modify @@session.skip_replication inside a transaction
1930		ER_STORED_FUNCTION_PREVENTS_SWITCH_SKIP_REPLICATION	Cannot modify @@session.skip_replication inside a stored function or trigger
1931		ER_QUERY_RESULT_INCOMPLETE	Query execution was interrupted. The query exceeded %s %llu. The query result may be incomplete
1932		ER_NO_SUCH_TABLE_IN_ENGINE 42S02	Table '%-.192s.%-.192s' doesn't exist in engine
1933		ER_TARGET_NOT_EXPLAINABLE	Target is not running an EXPLAINable command
1934		ER_CONNECTION_ALREADY_EXISTS	Connection '%.*s' conflicts with existing connection '%.*s'
1935		ER_MASTER_LOG_PREFIX	Master '%.*s':
1936		ER_CANT_START_STOP_SLAVE	Can't %s SLAVE '%.*s'
1937		ER_SLAVE_STARTED	SLAVE '%.*s' started
1938		ER_SLAVE_STOPPED	SLAVE '%.*s' stopped
1939		ER_SQL_DISCOVER_ERROR	Engine %s failed to discover table % `-.192s.%` -.192s with '%s'
1940		ER_FAILED_GTID_STATE_INIT	Failed initializing replication GTID state
1941		ER_INCORRECT_GTID_STATE	Could not parse GTID list
1942		ER_CANNOT_UPDATE_GTID_STATE	Could not update replication slave gtid state
1943		ER_DUPLICATE_GTID_DOMAIN	GTID %u-%u-%llu and %u-%u-%llu conflict (duplicate domain id %u)
1944		ER_GTID_OPEN_TABLE_FAILED	Failed to open %s.%s
1945		ER_GTID_POSITION_NOT_FOUND_IN_BINLOG	Connecting slave requested to start from GTID %u-%u-%llu, which is not in the master's binlog

Error Code	SQL STATE	Error	Description
1946		ER_CANNOT_LOAD_SLAVE_GTID_STATE	Failed to load replication slave GTID position from table %s.%s
1947		ER_MASTER_GTID_POS_CONFLICTS_WITH_BINLOG	Specified GTID %u-%u-%llu conflicts with the binary log which contains a more recent GTID %u-%u-%llu. If MASTER_GTID_POS=CURRENT_POS is used, the binlog position will override the new value of @@gtid_slave_pos.
1948		ER_MASTER_GTID_POS_MISSING_DOMAIN	Specified value for @@gtid_slave_pos contains no value for replication domain %u. This conflicts with the binary log which contains GTID %u-%u-%llu. If MASTER_GTID_POS=CURRENT_POS is used, the binlog position will override the new value of @@gtid_slave_pos.
1949		ER_UNTIL_REQUIRES_USING_GTID	START SLAVE UNTIL master_gtid_pos requires that slave is using GTID
1950		ER_GTID_STRICT_OUT_OF_ORDER	An attempt was made to binlog GTID %u-%u-%llu which would create an out-of-order sequence number with existing GTID %u-%u-%llu, and gtid strict mode is enabled.
1951		ER_GTID_START_FROM_BINLOG_HOLE	The binlog on the master is missing the GTID %u-%u-%llu requested by the slave (even though a subsequent sequence number does exist), and GTID strict mode is enabled
1952		ER_SLAVE_UNEXPECTED_MASTER_SWITCH	Unexpected GTID received from master after reconnect. This normally indicates that the master server was replaced without restarting the slave threads. %s
1953		ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_GTID_DOMAIN_ID_SEQ_NO	Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a transaction
1954		ER_STORED_FUNCTION_PREVENTS_SWITCH_GTID_DOMAIN_ID_SEQ_NO	Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a stored function or trigger
1955		ER_GTID_POSITION_NOT_FOUND_IN_BINLOG2	Connecting slave requested to start from GTID %u-%u-%llu, which is not in the master's binlog. Since the master's binlog contains GTIDs with higher sequence numbers, it probably means that the slave has diverged due to executing extra erroneous transactions
1956		ER_BINLOG_MUST_BE_EMPTY	This operation is not allowed if any GTID has been logged to the binary log. Run RESET MASTER first to erase the log
1957		ER_NO_SUCH_QUERY	Unknown query id: %lld
1958		ER_BAD_BASE64_DATA	Bad base64 data as position %u
1959		ER_INVALID_ROLE	Invalid role specification %`s.
1960		ER_INVALID_CURRENT_USER	The current user is invalid.
1961		ER_CANNOT_GRANT_ROLE	Cannot grant role '%s' to: %s.
1962		ER_CANNOT_REVOKE_ROLE	Cannot revoke role '%s' from: %s.
1963		ER_CHANGE_SLAVE_PARALLEL_THREADS_ACTIVE	Cannot change @@slave_parallel_threads while another change is in progress
1964		ER_PRIOR_COMMIT_FAILED	Commit failed due to failure of an earlier commit on which this one depends
1965		ER_IT_IS_A_VIEW	'%-.192s' is a view
1966		ER_SLAVE_SKIP_NOT_IN_GTID	When using GTID, @@sql_slave_skip_counter can not be used. Instead, setting @@gtid_slave_pos explicitly can be used to skip to after a given GTID position.
1967		ER_TABLE_DEFINITION_TOO_BIG	The definition for table %`s is too big

Error Code	SQLSTATE	Error	Description
1968		ER_PLUGIN_INSTALLED	Plugin '%-.192s' already installed
1969		ER_STATEMENT_TIMEOUT	Query execution was interrupted (max_statement_time exceeded)
1970		ER_SUBQUERIES_NOT_SUPPORTED	%s does not support subqueries or stored functions.
1971		ER_SET_STATEMENT_NOT_SUPPORTED	The system variable %.200s cannot be set in SET STATEMENT.
1972		ER_UNUSED_9	You should never see it
1973		ER_USER_CREATE_EXISTS	Can't create user '%-.64s'@'%-.64s'; it already exists
1974		ER_USER_DROP_EXISTS	Can't drop user '%-.64s'@'%-.64s'; it doesn't exist
1975		ER_ROLE_CREATE_EXISTS	Can't create role '%-.64s'; it already exists
1976		ER_ROLE_DROP_EXISTS	Can't drop role '%-.64s'; it doesn't exist
1977		ER_CANNOT_CONVERT_CHARACTER	Cannot convert '%s' character 0x%-.64s to '%s'
1978		ER_INVALID_DEFAULT_VALUE_FOR_FIELD	Incorrect default value '%-.128s' for column '%.192s'
1979		ER_KILL_QUERY_DENIED_ERROR	You are not owner of query %lu
1980		ER_NO_EIS_FOR_FIELD	Engine-independent statistics are not collected for column '%s'
1981		ER_WARN_AGGFUNC_DEPENDENCE	Aggregate function '%-.192s)' of SELECT #%d belongs to SELECT #%d
1982		WARN_INNODB_PARTITION_OPTION_IGNORED	<%-.64s> option ignored for InnoDB partition

Error Code	SQLSTATE	Error	Description
3000		ER_FILE_CORRUPT	File %s is corrupted
3001		ER_ERROR_ON_MASTER	Query partially completed on the master (error on master: %d) and was aborted. There is a chance that your master is inconsistent at this point. If you are sure that your master is ok, run this query manually on the slave and then restart the slave with SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1; START SLAVE;. Query:'%s'"
3002		ER_INCONSISTENT_ERROR	Query caused different errors on master and slave. Error on master: message (format)='%s' error code=%d; Error on slave:actual message='%s', error code=%d. Default database:'%s'. Query:'%s'
3003		ER_STORAGE_ENGINE_NOT_LOADED	Storage engine for table '%s'.'%s' is not loaded.
3004		ER_GET_STACKED_DA_WITHOUT_ACTIVE_HANDLER 0Z002	GET STACKED DIAGNOSTICS when handler not active
3005		ER_WARN_LEGACY_SYNTAX_CONVERTED	%s is no longer supported. The statement was converted to %s.

Error Code	SQLSTATE	Error	Description
3006		ER_BINLOG_UNSAFE_FULLTEXT_PLUGIN	Statement is unsafe because it uses a fulltext parser plugin which may not return the same value on the slave.
3007		ER_CANNOT_DISCARD_TEMPORARY_TABLE	Cannot DISCARD/IMPORT tablespace associated with temporary table
3008		ER_FK_DEPTH_EXCEEDED	Foreign key cascade delete/update exceeds max depth of %d.
3009		ER_COL_COUNT_DOESNT_MATCH_PLEASE_UPDATE_V2	Column count of %s.%s is wrong. Expected %d, found %d. Created with MariaDB %d, now running %d. Please use mysql_upgrade to fix this error.
3010		ER_WARN_TRIGGER_DOESNT_HAVE_CREATED	Trigger %s.%s.%s does not have CREATED attribute.
3011		ER_REFERENCED_TRG_DOES_NOT_EXIST_MYSQL	Referenced trigger ‘%s’ for the given action time and event type does not exist.
3012		ER_EXPLAIN_NOT_SUPPORTED	EXPLAIN FOR CONNECTION command is supported only for SELECT/UPDATE/INSERT/DELETE/REPLACE
3013		ER_INVALID_FIELD_SIZE	Invalid size for column ‘%-.192s’.
3014		ER_MISSING_HA_CREATE_OPTION	Table storage engine ‘%-.64s’ found required create option missing
3015		ER_ENGINE_OUT_OF_MEMORY	Out of memory in storage engine ‘%-.64s’.
3016		ER_PASSWORD_EXPIRE_ANONYMOUS_USER	The password for anonymous user cannot be expired.
3017		ER_SLAVE_SQL_THREAD_MUST_STOP	This operation cannot be performed with a running slave sql thread; run STOP SLAVE SQL_THREAD first
3018		ER_NO_FT_MATERIALIZED_SUBQUERY	Cannot create FULLTEXT index on materialized subquery
3019		ER_INNODB_UNDO_LOG_FULL	Undo Log error: %s
3020		ER_INVALID_ARGUMENT_FOR_LOGARITHM	Invalid argument for logarithm
3021		ER_SLAVE_CHANNEL_IO_THREAD_MUST_STOP	This operation cannot be performed with a running slave io thread; run STOP SLAVE IO_THREAD FOR CHANNEL ‘%s’ first.
3022		ER_WARN_OPEN_TEMP_TABLES_MUST_BE_ZERO	This operation may not be safe when the slave has temporary tables. The tables will be kept open until the server restarts or until the tables are deleted by any replicated DROP statement. Suggest to wait until slave_open_temp_tables = 0.
3023		ER_WARN_ONLY_MASTER_LOG_FILE_NO_POS	CHANGE MASTER TO with a MASTER_LOG_FILE clause but no MASTER_LOG_POS clause may not be safe. The old position value may not be valid for the new binary log file.
3024		ER_QUERY_TIMEOUT	Query execution was interrupted, maximum statement execution time exceeded

Error Code	SQLSTATE	Error	Description
3025		ER_NON_RO_SELECT_DISABLE_TIMER	Select is not a read only statement, disabling timer
3026		ER_DUP_LIST_ENTRY	Duplicate entry '%-.192s'.
3027		ER_SQL_MODE_NO_EFFECT	'%s' mode no longer has any effect. Use STRICT_ALL_TABLES or STRICT_TRANS_TABLES instead.
3028		ER_AGGREGATE_ORDER_FOR_UNION	Expression #%u of ORDER BY contains aggregate function and applies to a UNION
3029		ER_AGGREGATE_ORDER_NON_AGG_QUERY	Expression #%u of ORDER BY contains aggregate function and applies to the result of a non-aggregated query
3030		ER_SLAVE_WORKER_STOPPED_PREVIOUS_THD_ERROR	Slave worker has stopped after at least one previous worker encountered an error when slave-preserve-commit-order was enabled. To preserve commit order, the last transaction executed by this thread has not been committed. When restarting the slave after fixing any failed threads, you should fix this worker as well.
3031		ER_DONT_SUPPORT_SLAVE_PRESERVE_COMMIT_ORDER	slave_preserve_commit_order is not supported %s.
3032		ER_SERVER_OFFLINE_MODE	The server is currently in offline mode
3033		ER_GIS_DIFFERENT_SRIDS	Binary geometry function %s given two geometries of different srids: %u and %u, which should have been identical.
3034		ER_GIS_UNSUPPORTED_ARGUMENT	Calling geometry function %s with unsupported types of arguments.
3035		ER_GIS_UNKNOWN_ERROR	Unknown GIS error occurred in function %s.
3036		ER_GIS_UNKNOWN_EXCEPTION	Unknown exception caught in GIS function %s.
3037		ER_GIS_INVALID_DATA	Invalid GIS data provided to function %s.
3038		ER_BOOST_GEOMETRY_EMPTY_INPUT_EXCEPTION	The geometry has no data in function %s.
3039		ER_BOOST_GEOMETRY_CENTROID_EXCEPTION	Unable to calculate centroid because geometry is empty in function %s.
3040		ER_BOOST_GEOMETRY_OVERLAY_INVALID_INPUT_EXCEPTION	Geometry overlay calculation error: geometry data is invalid in function %s.
3041		ER_BOOST_GEOMETRY_TURN_INFO_EXCEPTION	Geometry turn info calculation error: geometry data is invalid in function %s.
3042		ER_BOOST_GEOMETRY_SELF_INTERSECTION_POINT_EXCEPTION	Analysis procedures of intersection points interrupted unexpectedly in function %s.

Error Code	SQLSTATE	Error	Description
3043		ER_BOOST_GEOMETRY_UNKNOWN_EXCEPTION	Unknown exception thrown in function %s.
3044		ER_STD_BAD_ALLOC_ERROR	Memory allocation error: %-.256s in function %s.
3045		ER_STD_DOMAIN_ERROR	Domain error: %-.256s in function %s.
3046		ER_STD_LENGTH_ERROR	Length error: %-.256s in function %s.
3047		ER_STD_INVALID_ARGUMENT	Invalid argument error: %-.256s in function %s.
3048		ER_STD_OUT_OF_RANGE_ERROR	Out of range error: %-.256s in function %s.
3049		ER_STD_OVERFLOW_ERROR	Overflow error error: %-.256s in function %s.
3050		ER_STD_RANGE_ERROR	Range error: %-.256s in function %s.
3051		ER_STD_UNDERFLOW_ERROR	Underflow error: %-.256s in function %s.
3052		ER_STD_LOGIC_ERROR	Logic error: %-.256s in function %s.
3053		ER_STD_RUNTIME_ERROR	Runtime error: %-.256s in function %s.
3054		ER_STD_UNKNOWN_EXCEPTION	Unknown exception: %-.384s in function %s.
3055		ER_GIS_DATA_WRONG_ENDIANESS	Geometry byte string must be little endian.
3056		ER_CHANGE_MASTER_PASSWORD_LENGTH	The password provided for the replication user exceeds the maximum length of 32 characters
3057	42000	ER_USER_LOCK_WRONG_NAME	Incorrect user-level lock name '%-.192s'.
3058		ER_USER_LOCK_DEADLOCK	Deadlock found when trying to get user-level lock; try rolling back transaction/releasing locks and restarting lock acquisition.
3059		ER_REPLACE_INACCESSIBLE_ROWS	REPLACE cannot be executed as it requires deleting rows that are not in the view
3060		ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_GIS	Do not support online operation on table with GIS index

Error Code	SQLSTATE	Error	Description
4000	0A000	ER_COMMULTI_BADCONTEXT	COM_MULTI can't return a result set in the given context
4001		ER_BAD_COMMAND_IN_MULTI	Command '%s' is not allowed for COM_MULTI

Error Code	SQL STATE	Error	Description
4002		ER_WITH_COL_WRONG_LIST	WITH column list and SELECT field list have different column counts
4003		ER_TOO_MANY_DEFINITIONS_IN_WITH_CLAUSE	Too many WITH elements in WITH clause
4004		ER_DUP_QUERY_NAME	Duplicate query name %`-.64s in WITH clause
4005		ER_RECURSIVE_WITHOUT_ANCHORS	No anchors for recursive WITH element ‘%s’
4006		ER_UNACCEPTABLE_MUTUAL_RECURSION	Unacceptable mutual recursion with anchored table ‘%s’
4007		ER_REF_TO_RECURSIVE_WITH_TABLE_IN_DERIVED	Reference to recursive WITH table ‘%s’ in materialized derived
4008		ER_NOT_STANDARD_COMPLIANT_RECURSIVE	Restrictions imposed on recursive definitions are violated for table ‘%s’“R_WRONG_WINDOW_SPEC_NAME
4009		ER_WRONG_WINDOW_SPEC_NAME	Window specification with name ‘%s’ is not defined
4010		ER_DUP_WINDOW_NAME	Multiple window specifications with the same name ‘%s’
4011		ER_PARTITION_LIST_IN_REFERENCING_WINDOW_SPEC	Window specification referencing another one ‘%s’ cannot contain partition list
4012		ER_ORDER_LIST_IN_REFERENCING_WINDOW_SPEC	Referenced window specification ‘%s’ already contains order list
4013		ER_WINDOW_FRAME_IN_REFERENCED_WINDOW_SPEC	Referenced window specification ‘%s’ cannot contain window frame
4014		ER_BAD_COMBINATION_OF_WINDOW_FRAME_BOUND_SPECS	Unacceptable combination of window frame bound specifications
4015		ER_WRONG_PLACEMENT_OF_WINDOW_FUNCTION	Window function is allowed only in SELECT list and ORDER BY clause
4016		ER_WINDOW_FUNCTION_IN_WINDOW_SPEC	Window function is not allowed in window specification
4017		ER_NOT_ALLOWED_WINDOW_FRAME	Window frame is not allowed with ‘%s’
4018		ER_NO_ORDER_LIST_IN_WINDOW_SPEC	No order list in window specification for ‘%s’
4019		ER_RANGE_FRAME_NEEDS_SIMPLE_ORDERBY	RANGE-type frame requires ORDER BY clause with single sort key
4020		ER_WRONG_TYPE_FOR_ROWS_FRAME	Integer is required for ROWS-type frame
4021		ER_WRONG_TYPE_FOR_RANGE_FRAME	Numeric datatype is required for RANGE-type frame
4022		ER_FRAME_EXCLUSION_NOT_SUPPORTED	Frame exclusion is not supported yet
4023		ER_WINDOW_FUNCTION_DONT_HAVE_FRAME	This window function may not have a window frame
4024		ER_INVALID_NTILE_ARGUMENT	Argument of NTILE must be greater than 0

Error Code	SQL STATE	Error	Description
4025	23000	ER_CONSTRAINT_FAILED	CONSTRAINT %`s failed for %`-.192s.%`-.192s
4026		ER_EXPRESSION_IS_TOO_BIG	Expression in the %s clause is too big
4027		ER_ERROR_EVALUATING_EXPRESSION	Got an error evaluating stored expression %s
4028		ER_CALCULATING_DEFAULT_VALUE	Got an error when calculating default value for %`s
4029		ER_EXPRESSION_REFERS_TO_UNINIT_FIELD	Expression for field %`-.64s is referring to uninitialized field %`s
4030		ER_PARTITION_DEFAULT_ERROR	Only one DEFAULT partition allowed
4031		ER_REFERENCED_TRG_DOES_NOT_EXIST	Referenced trigger ‘%s’ for the given action time and event type does not exist
4032		ER_INVALID_DEFAULT_PARAM	Default/ignore value is not supported for such parameter usage
4033		ER_BINLOG_NON_SUPPORTED_BULK	Only row based replication supported for bulk operations
4034		ER_BINLOG_UNCOMPRESS_ERROR	Uncompress the compressed binlog failed
4035		ER_JSON_BAD_CHR	Broken JSON string in argument %d to function ‘%s’ at position %d
4036		ER_JSON_NOT_JSON_CHR	Character disallowed in JSON in argument %d to function ‘%s’ at position %d
4037		ER_JSON_EOS	Unexpected end of JSON text in argument %d to function ‘%s’
4038		ER_JSON_SYNTAX	Syntax error in JSON text in argument %d to function ‘%s’ at position %d
4039		ER_JSON_ESCAPING	Incorrect escaping in JSON text in argument %d to function ‘%s’ at position %d
4040		ER_JSON_DEPTH	Limit of %d on JSON nested structures depth is reached in argument %d to function ‘%s’ at position %d
4041		ER_JSON_PATH_EOS	Unexpected end of JSON path in argument %d to function ‘%s’
4042		ER_JSON_PATH_SYNTAX	Syntax error in JSON path in argument %d to function ‘%s’ at position %d
4043		ER_JSON_PATH_DEPTH	Limit of %d on JSON path depth is reached in argument %d to function ‘%s’ at position %d
4044		ER_JSON_PATH_NO_WILDCARD	Wildcards in JSON path not allowed in argument %d to function ‘%s’
4045		ER_JSON_PATH_ARRAY	JSON path should end with an array identifier in argument %d to function ‘%s’
4046		ER_JSON_ONE_OR_ALL	Argument 2 to function ‘%s’ must be “one” or “all”.
4047		ER_UNSUPPORT_COMPRESSED_TEMPORARY_TABLE	CREATE TEMPORARY TABLE is not allowed with ROW_FORMAT=COMPRESSED or KEY_BLOCK_SIZE.
4048		ER_GEOJSON_INCORRECT	Incorrect GeoJSON format specified for st_geomfromgeojson function.
4049		ER_GEOJSON_TOO_FEW_POINTS	Incorrect GeoJSON format - too few points for linestring specified.
4050		ER_GEOJSON_NOT_CLOSED	Incorrect GeoJSON format - polygon not closed.
4051		ER_JSON_PATH_EMPTY	Path expression ‘$’ is not allowed in argument %d to function ‘%s’.
4052		ER_SLAVE_SAME_ID	A slave with the same server_uuid/server_id as this slave has connected to the master

Error Code	SQL STATE	Error	Description
4053		ER_FLASHBACK_NOT_SUPPORTED	Flashback does not support %s %s
4054		ER_KEYS_OUT_OF_ORDER	Keys are out order during bulk load
4055		ER_OVERLAPPING_KEYS	Bulk load rows overlap existing rows
4056		ER_REQUIRE_ROW_BINLOG_FORMAT	Can't execute updates on master with binlog_format != ROW.
4057		ER_ISOLATION_MODE_NOT_SUPPORTED	MyRocks supports only READ COMMITTED and REPEATABLE READ isolation levels. Please change from current isolation level %s
4058		ER_ON_DUPLICATE_DISABLED	When unique checking is disabled in MyRocks, INSERT,UPDATE,LOAD statements with clauses that update or replace the key (i.e. INSERT ON DUPLICATE KEY UPDATE, REPLACE) are not allowed. Query: %s
4059		ER_UPDATES_WITH_CONSISTENT_SNAPSHOT	Can't execute updates when you started a transaction with START TRANSACTION WITH CONSISTENT [ROCKSDB] SNAPSHOT.
4060		ER_ROLLBACK_ONLY	This transaction was rolled back and cannot be committed. Only supported operation is to roll it back, so all pending changes will be discarded. Please restart another transaction.
4061		ER_ROLLBACK_TO_SAVEPOINT	MyRocks currently does not support ROLLBACK TO SAVEPOINT if modifying rows.
4062		ER_ISOLATION_LEVEL_WITH_CONSISTENT_SNAPSHOT	Only REPEATABLE READ isolation level is supported for START TRANSACTION WITH CONSISTENT SNAPSHOT in RocksDB Storage Engine.
4063		ER_UNSUPPORTED_COLLATION	Unsupported collation on string indexed column %s.%s Use binary collation (%s).
4064		ER_METADATA_INCONSISTENCY	Table '%s' does not exist, but metadata information exists inside MyRocks. This is a sign of data inconsistency. Please check if '%s.frm' exists, and try to restore it if it does not exist.
4065		ER_CF_DIFFERENT	Column family ('%s') flag (%d) is different from an existing flag (%d). Assign a new CF flag, or do not change existing CF flag.
4066		ER_RDB_TTL_DURATION_FORMAT	TTL duration (%s) in MyRocks must be an unsigned non-null 64-bit integer.
4067		ER_RDB_STATUS_GENERAL	Status error %d received from RocksDB: %s
4068		ER_RDB_STATUS_MSG	%s, Status error %d received from RocksDB: %s
4069		ER_RDB_TTL_UNSUPPORTED	TTL support is currently disabled when table has a hidden PK.
4070		ER_RDB_TTL_COL_FORMAT	TTL column (%s) in MyRocks must be an unsigned non-null 64-bit integer, exist inside the table, and have an accompanying ttl duration.
4071		ER_PER_INDEX_CF_DEPRECATED	The per-index column family option has been deprecated
4072		ER_KEY_CREATE_DURING_ALTER	MyRocks failed creating new key definitions during alter.
4073		ER_SK_POPULATE_DURING_ALTER	MyRocks failed populating secondary key during alter.
4074		ER_SUM_FUNC_WITH_WINDOW_FUNC_AS_ARG	Window functions can not be used as arguments to group functions.
4075		ER_NET_OK_PACKET_TOO_LARGE	OK packet too large

Error Code	SQL STATE	Error	Description
4076		ER_GEOJSON_EMPTY_COORDINATES	Incorrect GeoJSON format - empty 'coordinates' array.
4077		ER_MYROCKS_CANT_NOPAD_COLLATION	MyRocks doesn't currently support collations with "No pad" attribute.
4078		ER_ILLEGAL_PARAMETER_DATA_TYPES2_FOR_OPERATION	Illegal parameter data types %s and %s for operation '%s'
4079		ER_ILLEGAL_PARAMETER_DATA_TYPE_FOR_OPERATION	Illegal parameter data type %s for operation '%s'
4080		ER_WRONG_PARAMCOUNT_TO_CURSOR	Incorrect parameter count to cursor '%-.192s'
4081		ER_UNKNOWN_STRUCTURED_VARIABLE	Unknown structured system variable or ROW routine variable '%-.*s'
4082		ER_ROW_VARIABLE_DOES_NOT_HAVE_FIELD	Row variable '%-.192s' does not have a field '%-.192s'
4083		ER_END_IDENTIFIER_DOES_NOT_MATCH	END identifier '%-.192s' does not match '%-.192s'
4084		ER_SEQUENCE_RUN_OUT	Sequence '%-.64s.%-.64s' has run out
4085		ER_SEQUENCE_INVALID_DATA	Sequence '%-.64s.%-.64s' values are conflicting
4086		ER_SEQUENCE_INVALID_TABLE_STRUCTURE	Sequence '%-.64s.%-.64s' table structure is invalid (%s)
4087		ER_SEQUENCE_ACCESS_ERROR	Sequence '%-.64s.%-.64s' access error
4088		ER_SEQUENCE_BINLOG_FORMAT	Sequences requires binlog_format mixed or row
4089		ER_NOT_SEQUENCE	'%-.64s.%-.64s' is not a SEQUENCE
4090		ER_NOT_SEQUENCE2	'%-.192s' is not a SEQUENCE
4091		ER_UNKNOWN_SEQUENCES	Unknown SEQUENCE: '%-.300s'
4092		ER_UNKNOWN_VIEW	Unknown VIEW: '%-.300s'
4093		ER_WRONG_INSERT_INTO_SEQUENCE	Wrong INSERT into a SEQUENCE. One can only do single table INSERT into a sequence object (like with mysqldump). If you want to change the SEQUENCE, use ALTER SEQUENCE instead.
4094		ER_SP_STACK_TRACE	At line %u in %s
4095		ER_PACKAGE_ROUTINE_IN_SPEC_NOT_DEFINED_IN_BODY	Subroutine '%-.192s' is declared in the package specification but is not defined in the package body
4096		ER_PACKAGE_ROUTINE_FORWARD_DECLARATION_NOT_DEFINED	Subroutine '%-.192s' has a forward declaration but is not defined
4097		ER_COMPRESSED_COLUMN_USED_AS_KEY	Compressed column '%-.192s' can't be used in key specification
4098		ER_UNKNOWN_COMPRESSION_METHOD	Unknown compression method: %s

Error Code	SQL STATE	Error	Description
4099		ER_WRONG_NUMBER_OF_VALUES_IN_TVC	The used table value constructor has a different number of values

Error Code	SQLSTATE	Error	Description
4100		ER_FIELD_REFERENCE_IN_TVC	Field reference '%-.192s' can't be used in table value constructor
4101		ER_WRONG_TYPE_FOR_PERCENTILE_FUNC	Numeric datatype is required for %s function
4102		ER_ARGUMENT_NOT_CONSTANT	Argument to the %s function is not a constant for a partition
4103		ER_ARGUMENT_OUT_OF_RANGE	Argument to the %s function does not belong to the range [0,1]
4104		ER_WRONG_TYPE_OF_ARGUMENT	%s function only accepts arguments that can be converted to numerical types
4105		ER_NOT_AGGREGATE_FUNCTION	Aggregate specific instruction (FETCH GROUP NEXT ROW) used in a wrong context
4106		ER_INVALID_AGGREGATE_FUNCTION	Aggregate specific instruction(FETCH GROUP NEXT ROW) missing from the aggregate function
4107		ER_INVALID_VALUE_TO_LIMIT	Limit only accepts integer values
4108		ER_INVISIBLE_NOT_NULL_WITHOUT_DEFAULT	Invisible column %`s must have a default value
4109		ER_UPDATE_INFO_WITH_SYSTEM_VERSIONING	Rows matched: %ld Changed: %ld Inserted: %ld Warnings: %ld
4110		ER_VERS_FIELD_WRONG_TYPE	`% s must be of type %s for system-versioned table %` s
4111		ER_VERS_ENGINE_UNSUPPORTED	Transaction-precise system versioning for %`s is not supported
4112		ER_UNUSED_23	You should never see it
4113		ER_PARTITION_WRONG_TYPE	Wrong partitioning type, expected type: %`s
4114		WARN_VERS_PART_FULL	Versioned table % `s.%` s: last HISTORY partition (%`s) is out of %s, need more HISTORY partitions
4115		WARN_VERS_PARAMETERS	Maybe missing parameters: %s
4116		ER_VERS_DROP_PARTITION_INTERVAL	Can only drop oldest partitions when rotating by INTERVAL
4117		ER_UNUSED_25	You should never see it
4118		WARN_VERS_PART_NON_HISTORICAL	Partition %`s contains non-historical data
4119		ER_VERS_ALTER_NOT_ALLOWED	Not allowed for system-versioned % `s.%` s. Change @@system_versioning_alter_history to proceed with ALTER.
4120		ER_VERS_ALTER_ENGINE_PROHIBITED	Not allowed for system-versioned % `s.%` s. Change to/from native system versioning engine is not supported.
4121		ER_VERS_RANGE_PROHIBITED	SYSTEM_TIME range selector is not allowed
4122		ER_CONFLICTING_FOR_SYSTEM_TIME	Conflicting FOR SYSTEM_TIME clauses in WITH RECURSIVE
4123		ER_VERS_TABLE_MUST_HAVE_COLUMNS	Table %`s must have at least one versioned column
4124		ER_VERS_NOT_VERSIONED	Table %`s is not system-versioned
4125		ER_MISSING	Wrong parameters for %`s: missing '%s'
4126		ER_VERS_PERIOD_COLUMNS	PERIOD FOR SYSTEM_TIME must use columns % `s and %` s
4127		ER_PART_WRONG_VALUE	Wrong parameters for partitioned %`s: wrong value for '%s'

Error Code	SQLSTATE	Error	Description
4128		ER_VERS_WRONG_PARTS	Wrong partitions for %`s: must have at least one HISTORY and exactly one last CURRENT
4129		ER_VERS_NO_TRX_ID	TRX_ID %llu not found in `mysql.transaction_registry`
4130		ER_VERS_ALTER_SYSTEM_FIELD	Can not change system versioning field %`s
4131		ER_DROP_VERSIONING_SYSTEM_TIME_PARTITION	Can not DROP SYSTEM VERSIONING for table %`s partitioned BY SYSTEM_TIME
4132		ER_VERS_DB_NOT_SUPPORTED	System-versioned tables in the %`s database are not supported
4133		ER_VERS_TRT_IS_DISABLED	Transaction registry is disabled
4134		ER_VERS_DUPLICATE_ROW_START_END	Duplicate ROW %s column %`s
4135		ER_VERS_ALREADY_VERSIONED	Table %`s is already system-versioned
4136		ER_UNUSED_24	You should never see it
4137		ER_VERS_NOT_SUPPORTED	System-versioned tables do not support %s
4138		ER_VERS_TRX_PART_HISTORIC_ROW_NOT_SUPPORTED	Transaction-precise system-versioned tables do not support partitioning by ROW START or ROW END
4139		ER_INDEX_FILE_FULL	The index file for table '%-.192s' is full
4140		ER_UPDATED_COLUMN_ONLY_ONCE	The column % `s.%` s cannot be changed more than once in a single UPDATE statement
4141		ER_EMPTY_ROW_IN_TVC	Row with no elements is not allowed in table value constructor in this context
4142		ER_VERS_QUERY_IN_PARTITION	SYSTEM_TIME partitions in table %`s does not support historical query
4143		ER_KEY_DOESNT_SUPPORT	%s index %`s does not support this operation
4144		ER_ALTER_OPERATION_TABLE_OPTIONS_NEED_REBUILD	Changing table options requires the table to be rebuilt
4145		ER_BACKUP_LOCK_IS_ACTIVE	Can't execute the command as you have a BACKUP STAGE active
4146		ER_BACKUP_NOT_RUNNING	You must start backup with "BACKUP STAGE START"
4147		ER_BACKUP_WRONG_STAGE	Backup stage '%s' is same or before current backup stage '%s'
4148		ER_BACKUP_STAGE_FAILED	Backup stage '%s' failed
4149		ER_BACKUP_UNKNOWN_STAGE	Unknown backup stage: '%s'. Stage should be one of START, FLUSH, BLOCK_DDL, BLOCK_COMMIT or END
4150		ER_USER_IS_BLOCKED	User is blocked because of too many credential errors; unblock with 'FLUSH PRIVILEGES'
4151		ER_ACCOUNT_HAS_BEEN_LOCKED	Access denied, this account is locked
4152		ER_PERIOD_TEMPORARY_NOT_ALLOWED	Application-time period table cannot be temporary
4153		ER_PERIOD_TYPES_MISMATCH	Fields of PERIOD FOR %`s have different types
4154		ER_MORE_THAN_ONE_PERIOD	Cannot specify more than one application-time period
4155		ER_PERIOD_FIELD_WRONG_ATTRIBUTES	Period field %`s cannot be %s
4156		ER_PERIOD_NOT_FOUND	Period %`s is not found in table
4157		ER_PERIOD_COLUMNS_UPDATED	Column % `s used in period %` s specified in update SET list
4158		ER_PERIOD_CONSTRAINT_DROP	Can't DROP CONSTRAINT `%s` . Use DROP PERIOD `%s` for this

Error Code	SQLSTATE	Error	Description
4159	42000 S1009	ER_TOO_LONG_KEYPART	Specified key part was too long; max key part length is %u bytes
4160		ER_TOO_LONG_DATABASE_COMMENT	Comment for database '%-.64s' is too long (max = %u)
4161		ER_UNKNOWN_DATA_TYPE	Unknown data type: '%-.64s'
4162		ER_UNKNOWN_OPERATOR	Operator does not exists: '%-.128s'
4163		ER_WARN_HISTORY_ROW_START_TIME	Table `%s.%s` history row start '%s' is later than row end '%s'
4164		ER_PART_STARTS_BEYOND_INTERVAL	%`s: STARTS is later than query time, first history partition may exceed INTERVAL value
4165		ER_GALERA_REPLICATION_NOT_SUPPORTED	DDL-statement is forbidden as table storage engine does not support Galera replication
4166	HY000	ER_LOAD_INFILE_CAPABILITY_DISABLED	The used command is not allowed because the MariaDB server or client has disabled the local infile capability
4167		ER_NO_SECURE_TRANSPORTS_CONFIGURED	No secure transports are configured, unable to set –require_secure_transport=ON
4168		ER_SLAVE_IGNORED_SHARED_TABLE	Slave SQL thread ignored the '%s' because table is shared
4169		ER_NO_AUTOINCREMENT_WITH_UNIQUE	AUTO_INCREMENT column `%s cannot be used in the UNIQUE index %s`
4170		ER_KEY_CONTAINS_PERIOD_FIELDS	Key `%s cannot explicitly include column %s`
4171		ER_KEY_CANT_HAVE_WITHOUT_OVERLAPS	Key %`s cannot have WITHOUT OVERLAPS
4172		ER_NOT_ALLOWED_IN_THIS_CONTEXT	'%-.128s' is not allowed in this context
4173		ER_DATA_WAS_COMMITED_UNDER_ROLLBACK	Engine %s does not support rollback. Changes were committed during rollback call
4174		ER_PK_INDEX_CANT_BE_IGNORED	A primary key cannot be marked as IGNORE
4175		ER_BINLOG_UNSAFE_SKIP_LOCKED	SKIP LOCKED makes this statement unsafe
4176		ER_JSON_TABLE_ERROR_ON_FIELD	Field '%s' can't be set for JSON_TABLE '%s'.
4177		ER_JSON_TABLE_ALIAS_REQUIRED	Every table function must have an alias.
4178		ER_JSON_TABLE_SCALAR_EXPECTED	Can't store an array or an object in the scalar column '%s' of JSON_TABLE '%s'.
4179		ER_JSON_TABLE_MULTIPLE_MATCHES	Can't store multiple matches of the path in the column '%s' of JSON_TABLE '%s'.
4180		ER_WITH_TIES_NEEDS_ORDER	FETCH … WITH TIES requires ORDER BY clause to be present
4181		ER_REMOVED_ORPHAN_TRIGGER	Dropped orphan trigger '%-.64s', originally created for table: '%-.192s'
4182		ER_STORAGE_ENGINE_DISABLED	Storage engine %s is disabled
4183		WARN_SFORMAT_ERROR	SFORMAT error: %s
4184		ER_PARTITION_CONVERT_SUBPARTITIONED	Convert partition is not supported for subpartitioned table.
4185		ER_PROVIDER_NOT_LOADED	MariaDB tried to use the %s, but its provider plugin is not loaded
4186		ER_JSON_HISTOGRAM_PARSE_FAILED	Failed to parse histogram for table %s.%s: %s at offset %d.
4187		ER_SF_OUT_INOUT_ARG_NOT_ALLOWED	OUT or INOUT argument %d for function %s is not allowed here
4188		ER_INCONSISTENT_SLAVE_TEMP_TABLE	Replicated query '%s' table `%s.%s` can not be temporary

Error Code	SQLSTATE	Error	Description
4189		ER_VERS_HIST_PART_FAILED	Versioned table %`s.%`s: adding HISTORY partition(s) failed
4190		WARN_OPTION_CHANGING	%s is implicitly changing the value of '%s' from '%s' to '%s'
4191		ER_CM_OPTION_MISSING_REQUIREMENT	CHANGE MASTER TO option '%s=%s' is missing requirement %s
4192		ER_SLAVE_STATEMENT_TIMEOUT	Slave log event execution was interrupted (slave_max_statement_time exceeded)
4193		ER_JSON_INVALID_VALUE_FOR_KEYWORD	Invalid value for keyword %s
4194		ER_JSON_SCHEMA_KEYWORD_UNSUPPORTED	%s keyword is not supported
4195		ER_JSON_NO_VARIABLE_SCHEMA	Variable schema is not supported.
4196		ER_PSEUDO_THREAD_ID_OVERWRITE	Pseudo thread id should not be modified by the client as it will be overwritten
4197		ER_SEQUENCE_TABLE_HAS_WRONG_NUMBER_OF_COLUMNS	Wrong number of columns
4198		ER_SEQUENCE_TABLE_CANNOT_HAVE_ANY_KEYS	Sequence tables cannot have any keys
4199		ER_SEQUENCE_TABLE_CANNOT_HAVE_ANY_CONSTRAINTS	Sequence tables cannot have any constraints

Error Code	SQLSTATE	Error	Description
4200		ER_SEQUENCE_TABLE_ORDER_BY	ORDER BY
4201		ER_VARIABLE_IGNORED	The variable '%s' is ignored. It only exists for compatibility with old installations and will be removed in a future release
4202		ER_INCORRECT_COLUMN_NAME_COUNT	Incorrect column name count for derived table.
4203		WARN_SORTING_ON_TRUNCATED_LENGTH	Values were longer than max_sort_length. Sorting used only the first bytes.
4204		ER_VECTOR_BINARY_FORMAT_INVALID	Invalid binary vector format. Must use IEEE standard float representation in little-endian format. Use VEC_FromText() to generate it.
4205		ER_VECTOR_FORMAT_INVALID	Invalid vector format at offset: %d for '%-.100s'. Must be a valid JSON array of numbers.
4206		ER_VEC_DISTANCE_TYPE	Cannot determine distance type for VEC_DISTANCE, index is not found
4221		ER_UNRESOLVED_TABLE_HINT_NAME	Unresolved table name %s for %s hint
4222		ER_UNRESOLVED_INDEX_HINT_NAME	Unresolved index name %s for %s hint
4223		ER_HINTS_INSIDE_VIEWS_NOT_SUPPORTED	Optimizer hints are not supported inside view definitions
4224		ER_WARN_MALFORMED_HINT	Hint %s is ignored as malformed
4225		ER_WARN_HINTS_ON_INSERT_PART_OF_INSERT_SELECT	Optimizer hints at the INSERT part of a INSERT..SELECT statement are not supported
4226		ER_INIT_SLAVE_ERROR	Slave SQL thread aborted. Can't execute init_slave query due to error code

This page is licensed: CC BY-SA / Gnu FDL

Operating System Error Codes

Below is a partial list of more common Linux and Windows operating system error codes.

Linux Error Codes

The perror tool can be used to find the error message which is associated with a given error code.

Number	Error Code	Description
1	EPERM	Operation not permitted
2	ENOENT	No such file or directory
3	ESRCH	No such process
4	EINTR	Interrupted system call
5	EIO	I/O error
6	ENXIO	No such device or address
7	E2BIG	Argument list too long
8	ENOEXEC	Exec format error
9	EBADF	Bad file number
10	ECHILD	No child processes
11	EAGAIN	Try again
12	ENOMEM	Out of memory
13	EACCES	Permission denied
14	EFAULT	Bad address
15	ENOTBLK	Block device required
16	EBUSY	Device or resource busy
17	EEXIST	File exists
18	EXDEV	Cross-device link
19	ENODEV	No such device
20	ENOTDIR	Not a directory
21	EISDIR	Is a directory
22	EINVAL	Invalid argument
23	ENFILE	File table overflow
24	EMFILE	Too many open files
25	ENOTTY	Not a typewriter
26	ETXTBSY	Text file busy
27	EFBIG	File too large
28	ENOSPC	No space left on device
29	ESPIPE	Illegal seek
30	EROFS	Read-only file system
31	EMLINK	Too many links
32	EPIPE	Broken pipe
33	EDOM	Math argument out of domain of func

Number	Error Code	Description
34	ERANGE	Math result not representable
35	EDEADLK	Resource deadlock would occur
36	ENAMETOOLONG	File name too long
37	ENOLCK	No record locks available
38	ENOSYS	Function not implemented
39	ENOTEMPTY	Directory not empty
40	ELOOP	Too many symbolic links encountered
42	ENOMSG	No message of desired type
43	EIDRM	Identifier removed
44	ECHRNG	Channel number out of range
45	EL2NSYNC	Level 2 not synchronized
46	EL3HLT	Level 3 halted
47	EL3RST	Level 3 reset
48	ELNRNG	Link number out of range
49	EUNATCH	Protocol driver not attached
50	ENOCSI	No CSI structure available
51	EL2HLT	Level 2 halted
52	EBADE	Invalid exchange
53	EBADR	Invalid request descriptor
54	EXFULL	Exchange full
55	ENOANO	No anode
56	EBADRQC	Invalid request code
57	EBADSLT	Invalid slot
59	EBFONT	Bad font file format
60	ENOSTR	Device not a stream
61	ENODATA	No data available
62	ETIME	Timer expired
63	ENOSR	Out of streams resources
64	ENONET	Machine is not on the network
65	ENOPKG	Package not installed
66	EREMOTE	Object is remote
67	ENOLINK	Link has been severed
68	EADV	Advertise error
69	ESRMNT	Srmount error
70	ECOMM	Communication error on send
71	EPROTO	Protocol error
72	EMULTIHOP	Multihop attempted
73	EDOTDOT	RFS specific error
74	EBADMSG	Not a data message

Number	Error Code	Description
75	EOVERFLOW	Value too large for defined data type
76	ENOTUNIQ	Name not unique on network
77	EBADFD	File descriptor in bad state
78	EREMCHG	Remote address changed
79	ELIBACC	Can not access a needed shared library
80	ELIBBAD	Accessing a corrupted shared library
81	ELIBSCN	.lib section in a.out corrupted
82	ELIBMAX	Attempting to link in too many shared libraries
83	ELIBEXEC	Cannot exec a shared library directly
84	EILSEQ	Illegal byte sequence
85	ERESTART	Interrupted system call should be restarted
86	ESTRPIPE	Streams pipe error
87	EUSERS	Too many users
88	ENOTSOCK	Socket operation on non-socket
89	EDESTADDRREQ	Destination address required
90	EMSGSIZE	Message too long
91	EPROTOTYPE	Protocol wrong type for socket
92	ENOPROTOOPT	Protocol not available
93	EPROTONOSUPPORT	Protocol not supported
94	ESOCKTNOSUPPORT	Socket type not supported
95	EOPNOTSUPP	Operation not supported on transport endpoint
96	EPFNOSUPPORT	Protocol family not supported
97	EAFNOSUPPORT	Address family not supported by protocol
98	EADDRINUSE	Address already in use
99	EADDRNOTAVAIL	Cannot assign requested address
100	ENETDOWN	Network is down
101	ENETUNREACH	Network is unreachable
102	ENETRESET	Network dropped connection because of reset
103	ECONNABORTED	Software caused connection abort
104	ECONNRESET	Connection reset by peer
105	ENOBUFS	No buffer space available
106	EISCONN	Transport endpoint is already connected
107	ENOTCONN	Transport endpoint is not connected
108	ESHUTDOWN	Cannot send after transport endpoint shutdown
109	ETOOMANYREFS	Too many references: cannot splice
110	ETIMEDOUT	Connection timed out
111	ECONNREFUSED	Connection refused
112	EHOSTDOWN	Host is down
113	EHOSTUNREACH	No route to host

Number	Error Code	Description
114	EALREADY	Operation already in progress
115	EINPROGRESS	Operation now in progress
116	ESTALE	Stale NFS file handle
117	EUCLEAN	Structure needs cleaning
118	ENOTNAM	Not a XENIX named type file
119	ENAVAIL	No XENIX semaphores available
120	EISNAM	Is a named type file
121	EREMOTEIO	Remote I/O error
122	EDQUOT	Quota exceeded
123	ENOMEDIUM	No medium found
124	EMEDIUMTYPE	Wrong medium type
125	ECANCELED	Operation Canceled
126	ENOKEY	Required key not available
127	EKEYEXPIRED	Key has expired
128	EKEYREVOKED	Key has been revoked
129	EKEYREJECTED	Key was rejected by service
130	EOWNERDEAD	Owner died
131	ENOTRECOVERABLE	State not recoverable

Windows Error Codes

> **Note:**
>
> For a complete list, see ms681381.aspx

Number	Error Code	Description
1	ERROR_INVALID_FUNCTION	Incorrect function.
2	ERROR_FILE_NOT_FOUND	The system cannot find the file specified.
3	ERROR_PATH_NOT_FOUND	The system cannot find the path specified.
4	ERROR_TOO_MANY_OPEN_FILES	The system cannot open the file.
5	ERROR_ACCESS_DENIED	Access is denied.
6	ERROR_INVALID_HANDLE	The handle is invalid.
7	ERROR_ARENA_TRASHED	The storage control blocks were destroyed.
8	ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
9	ERROR_INVALID_BLOCK	The storage control block address is invalid.
10	ERROR_BAD_ENVIRONMENT	The environment is incorrect.
11	ERROR_BAD_FORMAT	An attempt was made to load a program with an incorrect format.
12	ERROR_INVALID_ACCESS	The access code is invalid.
13	ERROR_INVALID_DATA	The data is invalid.
14	ERROR_OUTOFMEMORY	Not enough storage is available to complete this operation.
15	ERROR_INVALID_DRIVE	The system cannot find the drive specified.

Number	Error Code	Description
16	ERROR_CURRENT_DIRECTORY	The directory cannot be removed.
17	ERROR_NOT_SAME_DEVICE	The system cannot move the file to a different disk drive.
18	ERROR_NO_MORE_FILES	There are no more files.
19	ERROR_WRITE_PROTECT	The media is write protected.
20	ERROR_BAD_UNIT	The system cannot find the device specified.
21	ERROR_NOT_READY	The device is not ready.
22	ERROR_BAD_COMMAND	The device does not recognize the command.
23	ERROR_CRC	Data error (cyclic redundancy check).
24	ERROR_BAD_LENGTH	The program issued a command but the command length is incorrect.
25	ERROR_SEEK	The drive cannot locate a specific area or track on the disk.
26	ERROR_NOT_DOS_DISK	The specified disk or diskette cannot be accessed.
27	ERROR_SECTOR_NOT_FOUND	The drive cannot find the sector requested.
28	ERROR_OUT_OF_PAPER	The printer is out of paper.
29	ERROR_WRITE_FAULT	The system cannot write to the specified device.
30	ERROR_READ_FAULT	The system cannot read from the specified device.
31	ERROR_GEN_FAILURE	A device attached to the system is not functioning.
32	ERROR_SHARING_VIOLATION	The process cannot access the file because it is being used by another process.
33	ERROR_LOCK_VIOLATION	The process cannot access the file because another process has locked a portion of the file.
34	ERROR_WRONG_DISK	The wrong diskette is in the drive. Insert %2 (Volume Serial Number: %3) into drive %1.
36	ERROR_SHARING_BUFFER_EXCEEDED	Too many files opened for sharing.
38	ERROR_HANDLE_EOF	Reached the end of the file.
39	ERROR_HANDLE_DISK_FULL	The disk is full.
87	ERROR_INVALID_PARAMETER	The parameter is incorrect.
112	ERROR_DISK_FULL	The disk is full.
123	ERROR_INVALID_NAME	The file name, directory name, or volume label syntax is incorrect.
1450	ERROR_NO_SYSTEM_RESOURCES	Insufficient system resources exist to complete the requested service.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1000 to 1099

- Error 1000: hashchk
- Error 1001: isamchk
- Error 1002: NO
- Error 1003: YES
- Error 1004: Can’t create file
- Error 1005: Can’t create table
- Error 1006: Can’t create database
- Error 1007: Can’t create database, database exists

- Error 1008: Can’t drop database; database doesn’t exist
- Error 1009: Error dropping database (can’t delete)
- Error 1010: Error dropping database (can’t rmdir)
- Error 1011: Error on delete of
- Error 1012: Can’t read record in system table
- Error 1013: Can’t get status of
- Error 1014: Can’t get working directory
- Error 1015: Can’t lock file
- Error 1016: Can’t open file
- Error 1017: Can’t find file
- Error 1018: Can’t read dir of
- Error 1019: Can’t change dir to
- Error 1020: Record has changed since last read in table
- Error 1021: Disk full (%s); waiting for someone to free some space
- Error 1022: Can’t write; duplicate key in table
- Error 1023: Error on close of
- Error 1024: Error reading file
- Error 1025: Error on rename of
- Error 1026: Error writing file
- Error 1027: is locked against change
- Error 1028: Sort aborted
- Error 1029: View doesn’t exist for
- Error 1030: Got error from storage engine
- Error 1031: Table storage engine doesn’t have this option
- Error 1032: Can’t find record in
- Error 1033: Incorrect information in file
- Error 1034: Incorrect key file for table; try to repair it
- Error 1035: Old key file for table; repair it!
- Error 1036: Table is read only
- Error 1037: Out of memory; restart server and try again
- Error 1038: Out of sort memory, consider increasing server sort buffer size
- Error 1039: Unexpected EOF found when reading file
- Error 1040: Too many connections
- Error 1041: Out of memory
- Error 1042: Can’t get hostname for your address
- Error 1043: Bad handshake
- Error 1044: Access denied for user
- Error 1045: Access denied for user (using password)
- Error 1046: No database selected
- Error 1047: Unknown command
- Error 1048: Column cannot be null
- Error 1049: Unknown database
- Error 1050: Table already exists
- Error 1051: Unknown table
- Error 1052: Column is ambiguous

- Error 1053: Server shutdown in progress
- Error 1054: Unknown column
- Error 1055: isn’t in GROUP BY
- Error 1056: Can’t group on
- Error 1057: Statement has sum functions and columns in same statement
- Error 1058: Column count doesn’t match value count
- Error 1059: Identifier name is too long
- Error 1060: Duplicate column name
- Error 1061: Duplicate key name
- Error 1062: Duplicate entry for key
- Error 1063: Incorrect column specifier for column
- Error 1064: near at line
- Error 1065: Query was empty
- Error 1066: Not unique table/alias
- Error 1067: Invalid default value for
- Error 1068: Multiple primary key defined
- Error 1069: Too many keys specified; max keys allowed
- Error 1070: Too many key parts specified; max parts allowed
- Error 1071: Specified key was too long; max key length is
- Error 1072: Key column doesn’t exist in table
- Error 1073: BLOB column can’t be used in key specification with the used table type
- Error 1074: Column length too big for column; use BLOB or TEXT instead
- Error 1075: Incorrect table definition; there can be only one auto column and it must be defined as
- Error 1076: ready for connections
- Error 1077: Normal shutdown
- Error 1078: Got signal. Aborting!
- Error 1079: Shutdown complete
- Error 1080: Forcing close of thread
- Error 1081: Can’t create IP socket
- Error 1082: Table has no index like the one used in CREATE INDEX; recreate the table
- Error 1083: Field separator argument is not what is expected; check the manual
- Error 1084: You can’t use fixed rowlength with BLOBs; please use ‘fields terminated by’
- Error 1085: The file must be in the database directory or be readable by all
- Error 1086: File already exists
- Error 1087: Records Deleted Skipped Warnings
- Error 1088: Records Duplicates
- Error 1089: Incorrect prefix key; the used key part isn’t a string, the used length is longer than t
- Error 1090: You can’t delete all columns with ALTER TABLE; use DROP TABLE instead
- Error 1091: Can’t DROP; check that column/key exists
- Error 1092: Records Duplicates Warnings
- Error 1093: You can’t specify target table for update in FROM clause
- Error 1094: Unknown thread id
- Error 1095: You are not owner of thread
- Error 1096: No tables used
- Error 1097: Too many strings for column and SET

- Error 1098: Can’t generate a unique log-filename
- Error 1099: Table was locked with a READ lock and can’t be updated

This page is licensed: CC BY-SA / Gnu FDL

Error 1000: hashchk

Error Code	SQLSTATE	Error	Description
1000	HY000	ER_HASHCHK	hashchk

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1001: isamchk

Error Code	SQLSTATE	Error	Description
1001	HY000	ER_NISAMCHK	isamchk

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1002: NO

Error Code	SQLSTATE	Error	Description
1002	HY000	ER_NO	NO

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1003: YES

Error Code	SQLSTATE	Error	Description
1003	HY000	ER_YES	YES

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1004: Can’t create file

Error Code	SQLSTATE	Error	Description
1004	HY000	ER_CANT_CREATE_FILE	Can’t create file ‘%s’ (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1005: Can’t create table

Error Code	SQLSTATE	Error	Description
1005	HY000	ER_CANT_CREATE_TABLE	Can’t create table ‘%s’ (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1006: Can’t create database

Error Code	SQLSTATE	Error	Description
1006	`HY000`	`ER_CANT_CREATE_DB`	Can’t create database ‘%s’ (errno: %d)

Possible Causes and Solutions

Insufficient Disk Space

The server has insufficient free disk space to create the database. Free up space and retry.

File permissions

There may be a file permission issue on the data directories. Check both the user and group permissions and set accordingly.

This page is licensed: CC BY-SA / Gnu FDL

Error 1007: Can’t create database, database exists

Error Code	SQLSTATE	Error	Description
1007	HY000	ER_DB_CREATE_EXISTS	Can’t create database ‘%s’; database exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1008: Can’t drop database; database doesn’t exist

Error Code	SQLSTATE	Error	Description
1008	HY000	ER_DB_DROP_EXISTS	Can’t drop database ‘%s’; database doesn’t exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1009: Error dropping database (can’t delete)

Error Code	SQLSTATE	Error	Description
1009	HY000	ER_DB_DROP_DELETE	Error dropping database (can’t delete ‘%s’, errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1010: Error dropping database (can’t rmdir)

Error Code	SQLSTATE	Error	Description
1010	HY000	ER_DB_DROP_RMDIR	Error dropping database (can’t rmdir ‘%s’, errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1011: Error on delete of

Error Code	SQLSTATE	Error	Description
1011	HY000	ER_CANT_DELETE_FILE	Error on delete of ‘%s’ (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1012: Can’t read record in system table

Error Code	SQLSTATE	Error	Description
1012	HY000	ER_CANT_FIND_SYSTEM_REC	Can’t read record in system table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1013: Can’t get status of

Error Code	SQLSTATE	Error	Description
1013	HY000	ER_CANT_GET_STAT	Can’t get status of ‘%s’ (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1014: Can’t get working directory

Error Code	SQLSTATE	Error	Description
1014	HY000	ER_CANT_GET_WD	Can’t get working directory (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1015: Can’t lock file

Error Code	SQLSTATE	Error	Description
1015	HY000	ER_CANT_LOCK	Can’t lock file (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1016: Can’t open file

Error Code	SQLSTATE	Error	Description
1016	HY000	ER_CANT_OPEN_FILE	Can’t open file: ‘%s’ (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1017: Can't find file

Error Code	SQLSTATE	Error	Description
1017	HY000	ER_FILE_NOT_FOUND	Can't find file: '%s' (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1018: Can't read dir of

Error Code	SQLSTATE	Error	Description
1018	HY000	ER_CANT_READ_DIR	Can't read dir of '%s' (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1019: Can't change dir to

Error Code	SQLSTATE	Error	Description
1019	HY000	ER_CANT_SET_WD	Can't change dir to '%s' (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1020: Record has changed since last read in table

Error Code	SQLSTATE	Error	Description
1020	HY000	ER_CHECKREAD	Record has changed since last read in table '%s'

> This error is more common in MariaDB 11.8 and later due to changes in snapshot isolation behavior. For more information, see the upgrade guide and innodb_snapshot_isolation pages.

Possible Causes and Solutions

When InnoDB returns this error, it results in a **full transaction rollback**, not just a statement rollback. For that reason, this error should be treated similarly to a deadlock (ER_LOCK_DEADLOCK) or a lock wait timeout (ER_LOCK_WAIT_TIMEOUT).

Try restarting the transaction.

This page is licensed: CC BY-SA / Gnu FDL

Error 1021: Disk full (%s); waiting for someone to free some space

Error Code	SQLSTATE	Error	Description
1021	HY000	ER_DISK_FULL	Disk full (%s); waiting for someone to free some space…

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1022: Can’t write; duplicate key in table

Error Code	SQLSTATE	Error	Description
1022	23000	ER_DUP_KEY	Can’t write; duplicate key in table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1023: Error on close of

Error Code	SQLSTATE	Error	Description
1023	HY000	ER_ERROR_ON_CLOSE	Error on close of ‘%s’ (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1024: Error reading file

Error Code	SQLSTATE	Error	Description
1024	HY000	ER_ERROR_ON_READ	Error reading file ‘%s’ (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1025: Error on rename of

Error Code	SQLSTATE	Error	Description
1025	HY000	ER_ERROR_ON_RENAME	Error on rename of ‘%s’ to ‘%s’ (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1026: Error writing file

Error Code	SQLSTATE	Error	Description
1026	HY000	ER_ERROR_ON_WRITE	Error writing file ‘%s’ (errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1027: is locked against change

Error Code	SQLSTATE	Error	Description
1027	HY000	ER_FILE_USED	‘%s’ is locked against change

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1028: Sort aborted

Error Code	SQLSTATE	Error	Description
1028	HY000	ER_FILSORT_ABORT	Sort aborted

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1029: View doesn't exist for

Error Code	SQLSTATE	Error	Description
1029	HY000	ER_FORM_NOT_FOUND	View '%s' doesn't exist for '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1030: Got error from storage engine

Error Code	SQLSTATE	Error	Description
1030	HY000	ER_GET_ERRNO	Got error %d from storage engine

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1031: Table storage engine doesn't have this option

Error Code	SQLSTATE	Error	Description
1031	HY000	ER_ILLEGAL_HA	Table storage engine for '%s' doesn't have this option

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1032: Can't find record in

Error Code	SQLSTATE	Error	Description
1032	HY000	ER_KEY_NOT_FOUND	Can't find record in '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1033: Incorrect information in file

Error Code	SQLSTATE	Error	Description
1033	HY000	ER_NOT_FORM_FILE	Incorrect information in file: '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1034: Incorrect key file for table; try to repair it

Error Code	SQLSTATE	Error	Description
1034	HY000	ER_NOT_KEYFILE	Incorrect key file for table '%s'; try to repair it

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1035: Old key file for table; repair it!

Error Code	SQLSTATE	Error	Description
1035	HY000	ER_OLD_KEYFILE	Old key file for table '%s'; repair it!

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1036: Table is read only

Error Code	SQLSTATE	Error	Description
1036	HY000	ER_OPEN_AS_READONLY	Table '%s' is read only

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1037: Out of memory; restart server and try again

Error Code	SQLSTATE	Error	Description
1038	HY001	ER_OUT_OF_SORTMEMORY	Out of sort memory, consider increasing server sort buffer size

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1038: Out of sort memory, consider increasing server sort buffer size

Error Code	SQLSTATE	Error	Description
1038	HY001	ER_OUT_OF_SORTMEMORY	Out of sort memory, consider increasing server sort buffer size

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1039: Unexpected EOF found when reading file

Error Code	SQLSTATE	Error	Description
1039	HY000	ER_UNEXPECTED_EOF	Unexpected EOF found when reading file '%s' (Errno: %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1040: Too many connections

Error Code	SQLSTATE	Error	Description
1040	08004	ER_CON_COUNT_ERROR	Too many connections

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1041: Out of memory

Error Code	SQLSTATE	Error	Description
1041	HY000	ER_OUT_OF_RESOURCES	Out of memory; check if mysqld or some other process uses all available memory; if not, you may have to use ‘ulimit’ to allow mysqld to use more memory or you can add more swap space

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1042: Can’t get hostname for your address

Error Code	SQLSTATE	Error	Description
1042	08S01	ER_BAD_HOST_ERROR	Can’t get hostname for your address

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1043: Bad handshake

Error Code	SQLSTATE	Error	Description
1043	08S01	ER_HANDSHAKE_ERROR	Bad handshake

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1044: Access denied for user

Error Code	SQLSTATE	Error	Description
1044	42000	ER_DBACCESS_DENIED_ERROR	Access denied for user ‘%s’@‘%s’ to database ‘%s’

Possible Causes and Solutions

The user is trying to access a database/schema they do not have access to. See Troubleshooting Connection Issues for more, as well as GRANT for details on setting permissions.

See Also

- When access_denied_errors status variable is incremented

This page is licensed: CC BY-SA / Gnu FDL

Error 1045: Access denied for user (using password)

Error Code	SQLSTATE	Error	Description
1045	28000	ER_ACCESS_DENIED_ERROR	Access denied for user ‘%s’@‘%s’ (using password: %s)

Possible Causes and Solutions

The user/password combination does not exist or the user does not have privileges to the given database. See Troubleshooting Connection Issues for more, as well as GRANT for details on setting permissions.

See Also

- When access_denied_errors status variable is incremented

This page is licensed: CC BY-SA / Gnu FDL

Error 1046: No database selected

Error Code	SQLSTATE	Error	Description
1046	3D000	ER_NO_DB_ERROR	No database selected

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1047: Unknown command

Error Code	SQLSTATE	Error	Description
1047	08S01	ER_UNKNOWN_COM_ERROR	Unknown command

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1048: Column cannot be null

Error Code	SQLSTATE	Error	Description
1048	23000	ER_BAD_NULL_ERROR	Column ‘%s’ cannot be null

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1049: Unknown database

Error Code	SQLSTATE	Error	Description
1049	42000	ER_BAD_DB_ERROR	Unknown database ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1050: Table already exists

Error Code	SQLSTATE	Error	Description
1050	42S01	ER_TABLE_EXISTS_ERROR	Table ‘%s’ already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1051: Unknown table

Error Code	SQLSTATE	Error	Description
1051	42S02	ER_BAD_TABLE_ERROR	Unknown table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1052: Column is ambiguous

Error Code	SQLSTATE	Error	Description
1052	23000	ER_NON_UNIQ_ERROR	Column ‘%s’ in %s is ambiguous

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1053: Server shutdown in progress

Error Code	SQLSTATE	Error	Description
1053	08S01	ER_SERVER_SHUTDOWN	Server shutdown in progress

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1054: Unknown column

Error Code	SQLSTATE	Error	Description
1054	42S22	ER_BAD_FIELD_ERROR	Unknown column ‘%s’ in ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1055: isn’t in GROUP BY

Error Code	SQLSTATE	Error	Description
1055	42000	ER_WRONG_FIELD_WITH_GROUP	‘%s’ isn’t in GROUP BY

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1056: Can’t group on

Error Code	SQLSTATE	Error	Description
1056	42000	ER_WRONG_GROUP_FIELD	Can’t group on ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1057: Statement has sum functions and columns in same statement

Error Code	SQLSTATE	Error	Description
1057	42000	ER_WRONG_SUM_SELECT	Statement has sum functions and columns in same statement

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1058: Column count doesn’t match value count

Error Code	SQLSTATE	Error	Description
1058	21S01	ER_WRONG_VALUE_COUNT	Column count doesn’t match value count

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1059: Identifier name is too long

Error Code	SQLSTATE	Error	Description
1059	42000	ER_TOO_LONG_IDENT	Identifier name ‘%s’ is too long

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1060: Duplicate column name

Error Code	SQLSTATE	Error	Description
1060	42S21	ER_DUP_FIELDNAME	Duplicate column name ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1061: Duplicate key name

Error Code	SQLSTATE	Error	Description
1061	42000	ER_DUP_KEYNAME	Duplicate key name ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1062: Duplicate entry for key

Error Code	SQLSTATE	Error	Description
1062	23000	ER_DUP_ENTRY	Duplicate entry ‘%s’ for key %d

Possible Causes and Solutions

This error occurs when a key that requires a unique value (unique or primary) instead receives a duplicate. For example:

```
CREATE TABLE t1 (
  id INT AUTO_INCREMENT PRIMARY KEY,
  f VARCHAR(10) UNIQUE
);

INSERT INTO t1 (f) VALUES ('a'),('b');

SELECT * FROM t1;
+----+------+
| id | f    |
+----+------+
|  1 | a    |
|  2 | b    |
+----+------+

INSERT INTO t1 (f) VALUES ('b'),('c');
ERROR 1062 (23000): Duplicate entry 'b' for key 'f'
```

Solve the error by either not attempting to insert a duplicate value, or not requiring the key to be unique. For example, the below replaces the unique index with an index permitting duplicates:

```
ALTER TABLE t1 DROP INDEX f, ADD INDEX (f);

INSERT INTO t1 (f) VALUES ('b'),('c');

SELECT * FROM t1;
+----+------+
| id | f    |
+----+------+
|  1 | a    |
|  2 | b    |
|  3 | b    |
|  4 | c    |
+----+------+
```

This page is licensed: CC BY-SA / Gnu FDL

Error 1063: Incorrect column specifier for column

Error Code	SQLSTATE	Error	Description
1063	42000	ER_WRONG_FIELD_SPEC	Incorrect column specifier for column ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1064: near at line

Error Code	SQLSTATE	Error	Description
1064	42000	ER_PARSE_ERROR	%s near ‘%s’ at line %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1065: Query was empty

Error Code	SQLSTATE	Error	Description
1065	42000	ER_EMPTY_QUERY	Query was empty

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1066: Not unique table/alias

Error Code	SQLSTATE	Error	Description
1066	42000	ER_NONUNIQ_TABLE	Not unique table/alias: ‘%s’

Possible Causes and Solutions

In order to avoid ambiguity, a table name or alias must uniquely identify an object. There are many situations where this could occur.

For example, HANDLER statements use unqualified table names, requiring the use of an alias in certain contexts. For example:

```
CREATE TABLE t1 (f1 INT);

INSERT INTO t1 VALUES (1),(2),(3);

HANDLER t1 OPEN;

HANDLER t1 READ NEXT;
+------+
| f1   |
+------+
|    1 |
+------+

HANDLER t1 READ NEXT;
+------+
| f1   |
+------+
|    2 |
+------+
```

In the previous example, the HANDLER was opened with the t1 table name. Since HANDLERs use unqualified table names, trying to access another table with this same name, even though it’s in another database, will result in ambiguity. An alias needs to be used to avoid the ambiguity:

```
CREATE DATABASE db_new;

CREATE TABLE db_new.t1 (id INT);

INSERT INTO db_new.t1 VALUES (4),(5),(6);

HANDLER db_new.t1 OPEN;
ERROR 1066 (42000): Not unique table/alias: 't1'

HANDLER db_new.t1 OPEN AS db_new_t1;

HANDLER db_new_t1 READ NEXT LIMIT 3;
+------+
| id   |
+------+
|    4 |
|    5 |
|    6 |
+------+
```

This page is licensed: CC BY-SA / Gnu FDL

Error 1067: Invalid default value for

Error Code	SQLSTATE	Error	Description
1067	42000	ER_INVALID_DEFAULT	Invalid default value for ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1068: Multiple primary key defined

Error Code	SQLSTATE	Error	Description
1068	42000	ER_MULTIPLE_PRI_KEY	Multiple primary key defined

Possible Causes and Solutions

No more than one primary key can be defined per table. Attempting to define more in a single table will result in this error. For example:

```
CREATE TABLE t1(
    c1 INT NOT NULL AUTO_INCREMENT,
    c2 INT NOT NULL,
    PRIMARY KEY(c1),
    PRIMARY KEY(c2)
);
ERROR 1068 (42000): Multiple primary key defined

CREATE TABLE t1(
    c1 INT NOT NULL AUTO_INCREMENT,
    c2 INT NOT NULL,
    PRIMARY KEY(c1)
);
```

It’s also possible that this error results from a mistaken attempt to define a multi-part primary key. This is an example of the correct definition for such a key.

```
CREATE TABLE t1(
    c1 INT NOT NULL AUTO_INCREMENT,
    c2 INT NOT NULL,
    PRIMARY KEY(c1, c2)
);
```

This page is licensed: CC BY-SA / Gnu FDL

Error 1069: Too many keys specified; max keys allowed

Error Code	SQLSTATE	Error	Description
1069	42000	ER_TOO_MANY_KEYS	Too many keys specified; max %d keys allowed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1070: Too many key parts specified; max parts allowed

Error Code	SQLSTATE	Error	Description
1070	42000	ER_TOO_MANY_KEY_PARTS	Too many key parts specified; max %d parts allowed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1071: Specified key was too long; max key length is

Error Code	SQLSTATE	Error	Description
1071	42000	ER_TOO_LONG_KEY	Specified key was too long; max key length is %d bytes

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1072: Key column doesn’t exist in table

Error Code	SQLSTATE	Error	Description
1072	42000	ER_KEY_COLUMN_DOES_NOT_EXITS	Key column ‘%s’ doesn’t exist in table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1073: BLOB column can’t be used in key specification with the used table type

Error Code	SQLSTATE	Error	Description
1073	42000	ER_BLOB_USED_AS_KEY	BLOB column ‘%s’ can’t be used in key specification with the used table type

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1074: Column length too big for column; use BLOB or TEXT instead

Error Code	SQLSTATE	Error	Description
1074	42000	ER_TOO_BIG_FIELDLENGTH	Column length too big for column ‘%s’ (max = %lu); use BLOB or TEXT instead

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1075: Incorrect table definition; there can be only one auto column and it must be defined as

Error Code	SQLSTATE	Error	Description
1075	42000	ER_WRONG_AUTO_KEY	Incorrect table definition; there can be only one auto column and it must be defined as a key

Possible Causes and Solutions

There can only be one AUTO_INCREMENT column per table, and the column must be defined as a key. Not adhering to either of these restrictions will result in this error. For example:

```
CREATE OR REPLACE TABLE animals (
    id1 MEDIUMINT NOT NULL AUTO_INCREMENT,
    id2 TINYINT NOT NULL AUTO_INCREMENT,
    name CHAR(30) NOT NULL,
    PRIMARY KEY (id1),
    KEY (id2)
);
ERROR 1075 (42000): Incorrect table definition; there can be only one auto column and it must be defined
as a key

CREATE OR REPLACE TABLE animals (
    id1 MEDIUMINT NOT NULL AUTO_INCREMENT,
    name CHAR(30) NOT NULL
);
ERROR 1075 (42000): Incorrect table definition; there can be only one auto column and it must be defined
as a key

CREATE OR REPLACE TABLE animals (
    id MEDIUMINT NOT NULL AUTO_INCREMENT,
    name CHAR(30) NOT NULL,
    PRIMARY KEY (id)
);
Query OK, 0 rows affected (0.017 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Error 1076: ready for connections

Error Code	SQLSTATE	Error	Description
1076	HY000	ER_READY	%s: ready for connections. Version: ‘%s’ socket: ‘%s’ port: %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1077: Normal shutdown

Error Code	SQLSTATE	Error	Description
1077	HY000	ER_NORMAL_SHUTDOWN	%s: Normal shutdown

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1078: Got signal. Aborting!

Error Code	SQLSTATE	Error	Description
1078	HY000	ER_GOT_SIGNAL	%s: Got signal %d. Aborting!

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1079: Shutdown complete

Error Code	SQLSTATE	Error	Description
1079	HY000	ER_SHUTDOWN_COMPLETE	%s: Shutdown complete

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1080: Forcing close of thread

Error Code	SQLSTATE	Error	Description
1080	08S01	ER_FORCING_CLOSE	%s: Forcing close of thread %ld user: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1081: Can’t create IP socket

Error Code	SQLSTATE	Error	Description
1081	08S01	ER_IPSOCK_ERROR	Can’t create IP socket

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1082: Table has no index like the one used in CREATE INDEX; recreate the table

Error Code	SQLSTATE	Error	Description
1082	42S12	ER_NO_SUCH_INDEX	Table ‘%s’ has no index like the one used in CREATE INDEX; recreate the table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1083: Field separator argument is not what is expected; check the manual

Error Code	SQLSTATE	Error	Description
1083	42000	ER_WRONG_FIELD_TERMINATORS	Field separator argument is not what is expected; check the manual

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1084: You can’t use fixed rowlength with BLOBs; please use ‘fields terminated by’

Error Code	SQLSTATE	Error	Description
1084	42000	ER_BLOBS_AND_NO_TERMINATED	You can’t use fixed rowlength with BLOBs; please use ‘fields terminated by’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1085: The file must be in the database directory or be readable by all

Error Code	SQLSTATE	Error	Description
1085	HY000	ER_TEXTFILE_NOT_READABLE	The file ‘%s’ must be in the database directory or be readable by all

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1086: File already exists

Error Code	SQLSTATE	Error	Description
1086	HY000	ER_FILE_EXISTS_ERROR	File ‘%s’ already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1087: Records Deleted Skipped Warnings

Error Code	SQLSTATE	Error	Description
1087	HY000	ER_LOAD_INF	Records: %ld Deleted: %ld Skipped: %ld Warnings: %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1088: Records Duplicates

Error Code	SQLSTATE	Error	Description
1088	HY000	ER_ALTER_INF	Records: %ld Duplicates: %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1089: Incorrect prefix key; the used key part isn't a string, the used length is longer than t

Error 1089: Incorrect prefix key; the used key part isn't a string, the used length is longer than the key part

Error Code	SQLSTATE	Error	Description
1089	HY000	ER_WRONG_SUB_KEY	Incorrect prefix key; the used key part isn't a string, the used length is longer than the key part, or the storage engine doesn't support unique prefix keys

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1090: You can't delete all columns with ALTER TABLE; use DROP TABLE instead

Error Code	SQLSTATE	Error	Description
1090	42000	ER_CANT_REMOVE_ALL_FIELDS	You can't delete all columns with ALTER TABLE; use DROP TABLE instead

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1091: Can't DROP; check that column/key exists

Error Code	SQLSTATE	Error	Description
1091	42000	ER_CANT_DROP_FIELD_OR_KEY	Can't DROP '%s'; check that column/key exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1092: Records Duplicates Warnings

Error Code	SQLSTATE	Error	Description
1092	HY000	ER_INSERT_INF	Records: %ld Duplicates: %ld Warnings: %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1093: You can't specify target table for update in FROM clause

Error Code	SQLSTATE	Error	Description
1093	HY000	ER_UPDATE_TABLE_USED	You can't specify target table '%s' for update in FROM clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1094: Unknown thread id

Error Code	SQLSTATE	Error	Description
1094	HY000	ER_NO_SUCH_THREAD	Unknown thread id: %lu

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1095: You are not owner of thread

Error Code	SQLSTATE	Error	Description
1095	HY000	ER_KILL_DENIED_ERROR	You are not owner of thread %lu

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1096: No tables used

Error Code	SQLSTATE	Error	Description
1096	HY000	ER_NO_TABLES_USED	No tables used

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1097: Too many strings for column and SET

Error Code	SQLSTATE	Error	Description
1097	HY000	ER_TOO_BIG_SET	Too many strings for column %s and SET

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1098: Can’t generate a unique log-filename

Error Code	SQLSTATE	Error	Description
1098	HY000	ER_NO_UNIQUE_LOGFILE	Can’t generate a unique log-filename %s.(1-999)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1099: Table was locked with a READ lock and can’t be updated

Error Code	SQLSTATE	Error	Description
1099	HY000	ER_TABLE_NOT_LOCKED_FOR_WRITE	Table ‘%s’ was locked with a READ lock and can’t be updated

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1100 to 1199

- Error 1100: Table was not locked with LOCK TABLES
- Error 1101: You should never see it
- Error 1102: Incorrect database name
- Error 1103: Incorrect table name

- Error 1104: The SELECT would examine more than MAX_JOIN_SIZE rows
- Error 1105: Unknown error
- Error 1106: Unknown procedure
- Error 1107: Incorrect parameter count to procedure
- Error 1108: Incorrect parameters to procedure
- Error 1109: Unknown table
- Error 1110: Column specified twice
- Error 1111: Invalid use of group function
- Error 1112: Table uses an extension that doesn't exist in this MariaDB version
- Error 1113: A table must have at least 1 column
- Error 1114: The table is full
- Error 1115: Unknown character set
- Error 1116: Too many tables; MariaDB can only use tables in a join
- Error 1117: Too many columns
- Error 1118: Row size too large
- Error 1119: Thread stack overrun
- Error 1120: Cross dependency found in OUTER JOIN; examine your ON conditions
- Error 1121: Table handler doesn't support NULL in given index
- Error 1122: Can't load function
- Error 1123: Can't initialize function
- Error 1124: No paths allowed for shared library
- Error 1125: Function already exists
- Error 1126: Can't open shared library
- Error 1127: Can't find symbol in library
- Error 1128: Function is not defined
- Error 1129: Host is blocked because of many connection errors
- Error 1130: Host is not allowed to connect to this MariaDB server
- Error 1131: You are using MariaDB as an anonymous user
- Error 1132: You must have privileges to update tables in the mysql database
- Error 1133: Can't find any matching row in the user table
- Error 1134: Rows matched Changed Warnings
- Error 1135: Can't create a new thread
- Error 1136: Column count doesn't match value count at row
- Error 1137: Can't reopen table
- Error 1138: Invalid use of NULL value
- Error 1139: Got error from regexp
- Error 1140: Mixing of GROUP columns with no GROUP columns
- Error 1141: There is no such grant defined for user on host
- Error 1142: command denied to user for table
- Error 1143: command denied to user for column in table
- Error 1144: Illegal GRANT/REVOKE command
- Error 1145: The host or user argument to GRANT is too long
- Error 1146: Table doesn't exist
- Error 1147: There is no such grant defined for user on host on table
- Error 1148: The used command is not allowed with this MariaDB version

- Error 1149: You have an error in your SQL syntax
- Error 1150: Delayed insert thread couldn’t get requested lock for table
- Error 1151: Too many delayed threads in use
- Error 1152: Aborted connection to db user
- Error 1153: Got a packet bigger than ‘max_allowed_packet’ bytes
- Error 1154: Got a read error from the connection pipe
- Error 1155: Got an error from fcntl()
- Error 1156: Got packets out of order
- Error 1157: Couldn’t uncompress communication packet
- Error 1158: Got an error reading communication packets
- Error 1159: Got timeout reading communication packets
- Error 1160: Got an error writing communication packets
- Error 1161: Got timeout writing communication packets
- Error 1162: Result string is longer than ‘max_allowed_packet’ bytes
- Error 1163: The used table type doesn’t support BLOB/TEXT columns
- Error 1164: The used table type doesn’t support AUTO_INCREMENT columns
- Error 1165: INSERT DELAYED can’t be used with table because it is locked
- Error 1166: Incorrect column name
- Error 1167: The used storage engine can’t index column
- Error 1168: Unable to open underlying table which is differently defined or of non-MyISAM type or do
- Error 1169: Can’t write, because of unique constraint, to table
- Error 1170: BLOB/TEXT column used in key specification without a key length
- Error 1171: All parts of a PRIMARY KEY must be NOT NULL
- Error 1172: Result consisted of more than one row
- Error 1173: This table type requires a primary key
- Error 1174: This version of MariaDB is not compiled with RAID support
- Error 1175: You are using safe update mode and you tried to update a table without a WHERE that uses
- Error 1176: Key doesn’t exist in table
- Error 1177: Can’t open table
- Error 1178: The storage engine for the table doesn’t support
- Error 1179: You are not allowed to execute this command in a transaction
- Error 1180: Got error during COMMIT
- Error 1181: Got error during ROLLBACK
- Error 1182: Got error during FLUSH_LOGS
- Error 1183: Got error during CHECKPOINT
- Error 1184: Aborted connection to db user host
- Error 1185: You should never see it
- Error 1186: Binlog closed, cannot RESET MASTER
- Error 1187: Failed rebuilding the index of dumped table
- Error 1188: Error from master
- Error 1189: Net error reading from master
- Error 1190: Net error writing to master
- Error 1191: Can’t find FULLTEXT index matching the column list
- Error 1192: Can’t execute the given command because you have active locked tables or an active trans
- Error 1193: Unknown system variable

- Error 1194: Table is marked as crashed and should be repaired
- Error 1195: Table is marked as crashed and last (automatic?) repair failed
- Error 1196: Some non-transactional changed tables couldn't be rolled back
- Error 1197: Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage
- Error 1198: This operation cannot be performed with a running slave; run STOP SLAVE first
- Error 1199: This operation requires a running slave; configure slave and do START SLAVE

This page is licensed: CC BY-SA / Gnu FDL

Error 1100: Table was not locked with LOCK TABLES

Error Code	SQLSTATE	Error	Description
1100	HY000	ER_TABLE_NOT_LOCKED	Table '%s' was not locked with LOCK TABLES

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1101: You should never see it

Error Code	SQLSTATE	Error	Description
1101	HY000	ER_UNUSED_17	You should never see it

Possible Causes and Solutions

Error 1102: Incorrect database name

Error Code	SQLSTATE	Error	Description
1102	42000	ER_WRONG_DB_NAME	Incorrect database name '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1103: Incorrect table name

Error Code	SQLSTATE	Error	Description
1103	42000	ER_WRONG_TABLE_NAME	Incorrect table name '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1104: The SELECT would examine more than MAX_JOIN_SIZE rows

Error Code	SQLSTATE	Error	Description
1104	42000	ER_TOO_BIG_SELECT	The SELECT would examine more than MAX_JOIN_SIZE rows; check your WHERE and use SET SQL_BIG_SELECTS=1 or SET MAX_JOIN_SIZE=# if the SELECT is okay

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1105: Unknown error

Error Code	SQLSTATE	Error	Description
1105	HY000	ER_UNKNOWN_ERROR	Unknown error

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1106: Unknown procedure

Error Code	SQLSTATE	Error	Description
1106	42000	ER_UNKNOWN_PROCEDURE	Unknown procedure ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1107: Incorrect parameter count to procedure

Error Code	SQLSTATE	Error	Description
1107	42000	ER_WRONG_PARAMCOUNT_TO_PROCEDURE	Incorrect parameter count to procedure ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1108: Incorrect parameters to procedure

Error Code	SQLSTATE	Error	Description
1108	HY000	ER_WRONG_PARAMETERS_TO_PROCEDURE	Incorrect parameters to procedure ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1109: Unknown table

Error Code	SQLSTATE	Error	Description
1109	42S02	ER_UNKNOWN_TABLE	Unknown table ‘%s’ in %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1110: Column specified twice

Error Code	SQLSTATE	Error	Description
1110	42000	ER_FIELD_SPECIFIED_TWICE	Column ‘%s’ specified twice

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1111: Invalid use of group function

Error Code	SQLSTATE	Error	Description
1111	HY000	ER_INVALID_GROUP_FUNC_USE	Invalid use of group function

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1112: Table uses an extension that doesn't exist in this MariaDB version

Error Code	SQLSTATE	Error	Description
1112	42000	ER_UNSUPPORTED_EXTENSION	Table '%s' uses an extension that doesn't exist in this MariaDB version

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1113: A table must have at least 1 column

Error Code	SQLSTATE	Error	Description
1113	42000	ER_TABLE_MUST_HAVE_COLUMNS	A table must have at least 1 column

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1114: The table is full

Error Code	SQLSTATE	Error	Description
1114	HY000	ER_RECORD_FILE_FULL	The table '%s' is full

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1115: Unknown character set

Error Code	SQLSTATE	Error	Description
1115	42000	ER_UNKNOWN_CHARACTER_SET	Unknown character set: '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1116: Too many tables; MariaDB can only use tables in a join

Error Code	SQLSTATE	Error	Description
1116	HY000	ER_TOO_MANY_TABLES	Too many tables; MariaDB can only use %d tables in a join

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1117: Too many columns

Error Code	SQLSTATE	Error	Description
1117	HY000	ER_TOO_MANY_FIELDS	Too many columns

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1118: Row size too large

Error Code	SQLSTATE	Error	Description
1118	42000	ER_TOO_BIG_ROWSIZE	Row size too large. The maximum row size for the used table type, not counting BLOBs, is %ld. You have to change some columns to TEXT or BLOBs

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1119: Thread stack overrun

Error Code	SQLSTATE	Error	Description
1119	HY000	ER_STACK_OVERRUN	Thread stack overrun: Used: %ld of a %ld stack. Use ‘mysqld –thread_stack=#’ to specify a bigger stack if needed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1120: Cross dependency found in OUTER JOIN; examine your ON conditions

Error Code	SQLSTATE	Error	Description
1120	42000	ER_WRONG_OUTER_JOIN	Cross dependency found in OUTER JOIN; examine your ON conditions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1121: Table handler doesn’t support NULL in given index

Error Code	SQLSTATE	Error	Description
1121	42000	ER_NULL_COLUMN_IN_INDEX	Table handler doesn’t support NULL in given index. Please change column ‘%s’ to be NOT NULL or use another handler

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1122: Can’t load function

Error Code	SQLSTATE	Error	Description
1122	HY000	ER_CANT_FIND_UDF	Can’t load function ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1123: Can’t initialize function

Error Code	SQLSTATE	Error	Description
1123	HY000	ER_CANT_INITIALIZE_UDF	Can’t initialize function ‘%s’; %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1124: No paths allowed for shared library

Error Code	SQLSTATE	Error	Description
1124	HY000	ER_UDF_NO_PATHS	No paths allowed for shared library

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1125: Function already exists

Error Code	SQLSTATE	Error	Description
1125	HY000	ER_UDF_EXISTS	Function ‘%s’ already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1126: Can’t open shared library

Error Code	SQLSTATE	Error	Description
1126	HY000	ER_CANT_OPEN_LIBRARY	Can’t open shared library ‘%s’ (Errno: %d %s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1127: Can’t find symbol in library

Error Code	SQLSTATE	Error	Description
1127	HY000	ER_CANT_FIND_DL_ENTRY	Can’t find symbol ‘%s’ in library

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1128: Function is not defined

Error Code	SQLSTATE	Error	Description
1128	HY000	ER_FUNCTION_NOT_DEFINED	Function ‘%s’ is not defined

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1129: Host is blocked because of many connection errors

Error Code	SQLSTATE	Error	Description
1129	HY000	ER_HOST_IS_BLOCKED	Host ‘%s’ is blocked because of many connection errors; unblock with ‘mysqladmin flush-hosts’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1130: Host is not allowed to connect to this MariaDB server

Error Code	SQLSTATE	Error	Description
1130	HY000	ER_HOST_NOT_PRIVILEGED	Host ‘%s’ is not allowed to connect to this MariaDB server

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1131: You are using MariaDB as an anonymous user

Error Code	SQLSTATE	Error	Description
1131	42000	ER_PASSWORD_ANONYMOUS_USER	You are using MariaDB as an anonymous user and anonymous users are not allowed to change passwords

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1132: You must have privileges to update tables in the mysql database

Error Code	SQLSTATE	Error	Description
1132	42000	ER_PASSWORD_NOT_ALLOWED	You must have privileges to update tables in the mysql database to be able to change passwords for others

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1133: Can’t find any matching row in the user table

Error Code	SQLSTATE	Error	Description
1133	42000	ER_PASSWORD_NO_MATCH	Can’t find any matching row in the user table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1134: Rows matched Changed Warnings

Error Code	SQLSTATE	Error	Description
1134	HY000	ER_UPDATE_INF	Rows matched: %ld Changed: %ld Warnings: %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1135: Can’t create a new thread

Error Code	SQLSTATE	Error	Description
1135	HY000	ER_CANT_CREATE_THREAD	Can’t create a new thread (Errno %d); if you are not out of available memory, you can consult the manual for a possible OS-dependent bug

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1136: Column count doesn’t match value count at row

Error Code	SQLSTATE	Error	Description
1136	21S01	ER_WRONG_VALUE_COUNT_ON_ROW	Column count doesn’t match value count at row %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1137: Can’t reopen table

Error Code	SQLSTATE	Error	Description
1137	HY000	ER_CANT_REOPEN_TABLE	Can’t reopen table: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1138: Invalid use of NULL value

Error Code	SQLSTATE	Error	Description
1138	22004	ER_INVALID_USE_OF_NULL	Invalid use of NULL value

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1139: Got error from regexp

Error Code	SQLSTATE	Error	Description
1139	42000	ER_REGEXP_ERROR	Got error ‘%s’ from regexp

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1140: Mixing of GROUP columns with no GROUP columns

Error Code	SQLSTATE	Error	Description
1140	42000	ER_MIX_OF_GROUP_FUNC_AND_FIELDS	Mixing of GROUP columns (MIN(),MAX(),COUNT(),…) with no GROUP columns is illegal if there is no GROUP BY clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1141: There is no such grant defined for user on host

Error Code	SQLSTATE	Error	Description
1141	42000	ER_NONEXISTING_GRANT	There is no such grant defined for user ‘%s’ on host ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1142: command denied to user for table

Error Code	SQLSTATE	Error	Description
1142	42000	ER_TABLEACCESS_DENIED_ERROR	%s command denied to user ‘%s’@‘%s’ for table ‘%s’

Possible Causes and Solutions

The user is trying to access a table or a column from a table they do not have access to. See GRANT for details on setting permissions.

See Also

- When access_denied_errors status variable is incremented

This page is licensed: CC BY-SA / Gnu FDL

Error 1143: command denied to user for column in table

Error Code	SQLSTATE	Error	Description
1143	42000	ER_COLUMNACCESS_DENIED_ERROR	%s command denied to user ‘%s’@‘%s’ for column ‘%s’ in table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1144: Illegal GRANT/REVOKE command

Error Code	SQLSTATE	Error	Description
1144	42000	ER_ILLEGAL_GRANT_FOR_TABLE	Illegal GRANT/REVOKE command; please consult the manual to see which privileges can be used

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1145: The host or user argument to GRANT is too long

Error Code	SQLSTATE	Error	Description
1145	42000	ER_GRANT_WRONG_HOST_OR_USER	The host or user argument to GRANT is too long

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1146: Table doesn't exist

Error Code	SQLSTATE	Error	Description
1146	42S02	ER_NO_SUCH_TABLE	Table '%s.%s' doesn't exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1147: There is no such grant defined for user on host on table

Error Code	SQLSTATE	Error	Description
1147	42000	ER_NONEXISTING_TABLE_GRANT	There is no such grant defined for user '%s' on host '%s' on table '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1148: The used command is not allowed with this MariaDB version

Error Code	SQLSTATE	Error	Description
1148	42000	ER_NOT_ALLOWED_COMMAND	The used command is not allowed with this MariaDB version

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1149: You have an error in your SQL syntax

Error Code	SQLSTATE	Error	Description
1149	42000	ER_SYNTAX_ERROR	You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use

Possible Causes and Solutions

This is one of the most common errors to see. It's caused by an SQL syntax error. The error message will give you a clue as to where the error could be, as MariaDB displays the text following where it picked up an error. There are a huge number of possible causes (and in most cases they're simple typos), but here are a few common ones;

Delimiters

Delimiters need to be present at the end of each statement. See Delimiters.
For example:

```
SELECT 1
SELECT 2;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds
  to your MariaDB server version for the right syntax to use near 'SELECT 2' at line 2
```

Here, there was no delimiter separating the first and the second line. MariaDB picked up the error at the start of the second line, pointing one to the end of the first line. The correct syntax is:

```
SELECT 1;
SELECT 2;
```

Delimiters in Stored Programs

When creating stored programs from the command-line, it is likely you will need to differentiate between the regular delimiter and a delimiter inside a BEGIN END block. See Delimiters.

For example, take the following, seemingly-valid declaration:

```
CREATE FUNCTION FortyTwo() RETURNS TINYINT DETERMINISTIC
BEGIN
  DECLARE x TINYINT;
  SET x = 42;
  RETURN x;
END;
```

Attempting to run will result in an error:

```
CREATE FUNCTION FortyTwo() RETURNS TINYINT DETERMINISTIC
    -> BEGIN
    ->   DECLARE x TINYINT;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds
  to your MariaDB server version for the right syntax to use near '' at line 3
```

The error was picked up at the end of the `DECLARE x TINYINT;` line, guiding one to the delimiter found there. The solution is to specify a distinct delimiter for the duration of the process, using the DELIMITER command, for example:

```
DELIMITER //

CREATE FUNCTION FortyTwo() RETURNS TINYINT DETERMINISTIC
BEGIN
  DECLARE x TINYINT;
  SET x = 42;
  RETURN x;
END

//

DELIMITER ;
```

Reserved Words

Reserved words cannot be used as identifiers unless they are quoted. For example:

```
CREATE TABLE accessible(id INT);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds
  to your MariaDB server version for the right syntax to use near 'accessible(id INT)' at line 1
```

The error was picked up after the `accessible` identifier, leading one to suspect that it's causing the problem. And indeed, it's a reserved word. There are a number of possible solutions.

Use a new, non-reserved, identifier:

```
CREATE TABLE accessible_status(id INT);
```

or quote the identifier:

```
CREATE TABLE `accessible`(id INT);
```

or fully qualify the identifier:

```
CREATE TABLE test.accessible(id INT);
```

Syntax Doesn't Match SQL_MODE

Some syntax may be valid in a particular SQL_MODE, but not in others. There are too many to list here, but the SQL_MODE page provides a full list.
For example:

```
CREATE TABLE "t" (id INT);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds
  to your MariaDB server version for the right syntax to use near '"t" (id INT)' at line 1
```

The error message directs one to the identifier `t`. Double quotes cannot be used to quote identifiers unless ANSI_MODE is set, which in this case it isn't:

```
select @@sql_mode;
+--------------------------------------------------------------------------------------------+
| @@sql_mode                                                                                 |
+--------------------------------------------------------------------------------------------+
| STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+--------------------------------------------------------------------------------------------+
```

Either quote with the standard quote character, the backtick, or set the appropriate SQL_MODE:

```
SET @@SQL_MODE = CONCAT(@@SQL_MODE, ',ANSI_QUOTES');
CREATE TABLE "t" (id INT);
```

Oracle Mode

Oracle mode in particular has a large number of differences to the default SQL_MODE, for example:

```
CREATE TABLE t (id VARCHAR2(10));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds
  to your MariaDB server version for the right syntax to use near '(10))' at line 1
```

The error is identified just after the VARCHAR2 type is specified, and this type is not valid in the default mode:

```
SELECT @@sql_mode;
+-------------------------------------------------------------------+
| @@sql_mode                                                        |
+-------------------------------------------------------------------+
| STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER |
+-------------------------------------------------------------------+
SET SQL_MODE='ORACLE';
CREATE TABLE t (id VARCHAR2(10));
```

See Oracle mode for a full list of syntax differences.

This page is licensed: CC BY-SA / Gnu FDL

Error 1150: Delayed insert thread couldn't get requested lock for table

Error Code	SQLSTATE	Error	Description
1150	HY000	ER_DELAYED_CANT_CHANGE_LOCK	Delayed insert thread couldn't get requested lock for table %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1151: Too many delayed threads in use

Error Code	SQLSTATE	Error	Description
1151	HY000	ER_TOO_MANY_DELAYED_THREADS	Too many delayed threads in use

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1152: Aborted connection to db user

Error Code	SQLSTATE	Error	Description
1152	08S01	ER_ABORTING_CONNECTION	Aborted connection %ld to db: '%s' user: '%s' (%s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1153: Got a packet bigger than 'max_allowed_packet' bytes

Error Code	SQLSTATE	Error	Description
1153	08S01	ER_NET_PACKET_TOO_LARGE	Got a packet bigger than 'max_allowed_packet' bytes

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1154: Got a read error from the connection pipe

Error Code	SQLSTATE	Error	Description
1154	08S01	ER_NET_READ_ERROR_FROM_PIPE	Got a read error from the connection pipe

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1155: Got an error from fcntl()

Error Code	SQLSTATE	Error	Description
1155	08S01	ER_NET_FCNTL_ERROR	Got an error from fcntl()

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1156: Got packets out of order

Error Code	SQLSTATE	Error	Description
1156	08S01	ER_NET_PACKETS_OUT_OF_ORDER	Got packets out of order

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1157: Couldn't uncompress communication packet

Error Code	SQLSTATE	Error	Description
1157	08S01	ER_NET_UNCOMPRESS_ERROR	Couldn't uncompress communication packet

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1158: Got an error reading communication packets

Error Code	SQLSTATE	Error	Description
1158	08S01	ER_NET_READ_ERROR	Got an error reading communication packets

Possible Causes

This error tells us that the connection between the server and client was aborted.
The most common cause is that the client hard-aborted the connection, without calling mysql_close().
It could also be a problem with the connection to the server, such as a wrong or lost package.

How to Find Out More

The error log may have more information about the cause of the error.
Setting the MariaDB server option log_warnings to a value of 4 or above can generate more diagnostic warnings in the error log when there is a problem reading packages:

```
2024-04-19 17:02:11 20090 [Warning] mariadbd: Could not read packet: fd: 4575  state: 1  read_length: 4
errno: 104  vio_errno: 1158  length: -1
```

How to interpret the above:

- `20090` is the connection id that got the warning/error.
- `fd: 4575` 4575 is the file descriptor that had a problem.
- `state: 1` shows that the file is open. Anything else means the file is not active. Note that in some cases MariaDB will try to abort a read on a file descriptor by closing it.
- `read_length: 4` is how many characters the server tried to read from the file.
- `errno 104` is the system error code (more below).
- `vio_errno: 1158` is our internal error code from the vio library, which stands for ER_NET_READ_ERROR "Got an error reading communication packets".
- `length: -1` is how many characters was reported to be read from the read() system call. -1 indicates failure. If `length` <= `read_length` then something broke in the communication.

You can use the perror utility to get a description of system and storage engine errors:

```
shell> perror 104
OS error code 104: Connection reset by peer
```

This means the connection was aborted by the application/user.

How to Fix

- Ensure you have a stable internet connection.
- Ensure that your applications calls mysql_close() for all open connections before exiting.

See Also

- perror Getting a description for an error number.

This page is licensed: CC BY-SA / Gnu FDL

Error 1159: Got timeout reading communication packets

Error Code	SQLSTATE	Error	Description
1159	08S01	ER_NET_READ_INTERRUPTED	Got timeout reading communication packets

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1160: Got an error writing communication packets

Error Code	SQLSTATE	Error	Description
1160	08S01	ER_NET_ERROR_ON_WRITE	Got an error writing communication packets

Possible Causes

This error tells us that the connection between the server and client was aborted.
The most common cause is that the client hard-aborted the connection, without calling mysql_close().
It could also be a problem with the connection to the server, such as a wrong or lost package.

How to Find Out More

The error log may have more information about the cause of the error.
Setting the MariaDB server option log_warnings to a value of 4 or above can generate more diagnostic warnings in the error log when there is a problem reading packages:

```
2024-02-17 12:32:45 5461 [ERROR] mariadbd: Could not write packet: fd: 406  state: 2  errno: 32
vio_errno: 1160  length: 34
```

How to interpret the above:

- `5461` is the connection id that got the warning/error.
- `fd: 406` 406 is the file descriptor that had a problem.
- `state: 2`
- `length: 34` is how many characters the server tried to wrote to the file.
- `errno 32` is the system error code (more below).
- `vio_errno: 1160` is our internal error code from the vio library, which stands for ER_NET_WRITE_ERROR “Got an error writing communication packets”.
- `length: -1` is how many characters was reported to be read from the read() system call. -1 indicates failure. If `length` <= `read_length` then something broke in the communication.

You can use the perror utility to get a description of system and storage engine errors:

```
shell> perror 104
OS error code 104: Connection reset by peer
```

This means the connection was aborted by the application/user.

How to Fix

- Ensure you have a stable internet connection.
- Ensure that your applications calls mysql_close() for all open connections before exiting.

See Also

- perror Getting a description for an error number.

This page is licensed: CC BY-SA / Gnu FDL

Error 1161: Got timeout writing communication packets

Error Code	SQLSTATE	Error	Description
1161	08S01	ER_NET_WRITE_INTERRUPTED	Got timeout writing communication packets

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1162: Result string is longer than ‘max_allowed_packet’ bytes

Error Code	SQLSTATE	Error	Description
1162	42000	ER_TOO_LONG_STRING	Result string is longer than ‘max_allowed_packet’ bytes

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1163: The used table type doesn’t support BLOB/TEXT columns

Error Code	SQLSTATE	Error	Description
1163	42000	ER_TABLE_CANT_HANDLE_BLOB	The used table type doesn’t support BLOB/TEXT columns

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1164: The used table type doesn’t support AUTO_INCREMENT columns

Error Code	SQLSTATE	Error	Description
1164	42000	ER_TABLE_CANT_HANDLE_AUTO_INCREMENT	The used table type doesn’t support AUTO_INCREMENT columns

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1165: INSERT DELAYED can’t be used with table because it is locked

Error Code	SQLSTATE	Error	Description
1165	HY000	ER_DELAYED_INSERT_TABLE_LOCKED	INSERT DELAYED can’t be used with table ‘%s’ because it is locked with LOCK TABLES

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1166: Incorrect column name

Error Code	SQLSTATE	Error	Description
1166	42000	ER_WRONG_COLUMN_NAME	Incorrect column name ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1167: The used storage engine can’t index column

Error Code	SQLSTATE	Error	Description
1167	42000	ER_WRONG_KEY_COLUMN	The used storage engine can’t index column ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1168: Unable to open underlying table which is differently defined or of non-MyISAM type or do

Error 1168: Unable to open underlying table which is differently defined or of non-MyISAM type or doesn’t exist

Error Code	SQLSTATE	Error	Description
1168	HY000	ER_WRONG_MRG_TABLE	Unable to open underlying table which is differently defined or of non-MyISAM type or doesn’t exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1169: Can’t write, because of unique constraint, to table

Error Code	SQLSTATE	Error	Description
1169	23000	ER_DUP_UNIQUE	Can’t write, because of unique constraint, to table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1170: BLOB/TEXT column used in key specification without a key length

Error Code	SQLSTATE	Error	Description
1170	42000	ER_BLOB_KEY_WITHOUT_LENGTH	BLOB/TEXT column ‘%s’ used in key specification without a key length

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1171: All parts of a PRIMARY KEY must be NOT NULL

Error Code	SQLSTATE	Error	Description
1171	42000	ER_PRIMARY_CANT_HAVE_NULL	All parts of a PRIMARY KEY must be NOT NULL; if you need NULL in a key, use UNIQUE instead

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1172: Result consisted of more than one row

Error Code	SQLSTATE	Error	Description
1172	42000	ER_TOO_MANY_ROWS	Result consisted of more than one row

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1173: This table type requires a primary key

Error Code	SQLSTATE	Error	Description
1173	42000	ER_REQUIRES_PRIMARY_KEY	This table type requires a primary key

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1174: This version of MariaDB is not compiled with RAID support

Error Code	SQLSTATE	Error	Description
1174	HY000	ER_NO_RAID_COMPILED	This version of MariaDB is not compiled with RAID support

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1175: You are using safe update mode and you tried to update a table without a WHERE that uses

Error Code	SQLSTATE	Error	Description
1175	HY000	ER_UPDATE_WITHOUT_KEY_IN_SAFE_MODE	You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1176: Key doesn't exist in table

Error Code	SQLSTATE	Error	Description
1176	42000	ER_KEY_DOES_NOT_EXITS	Key '%s' doesn't exist in table '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1177: Can't open table

Error Code	SQLSTATE	Error	Description
1177	42000	ER_CHECK_NO_SUCH_TABLE	Can't open table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1178: The storage engine for the table doesn't support

Error Code	SQLSTATE	Error	Description
1178	42000	ER_CHECK_NOT_IMPLEMENTED	The storage engine for the table doesn't support %s

Possible Causes and Solutions

MariaDB storage engines differ in many ways, and not all operations are supported by each engine. Perhaps you don't need the intended operation for the engine, or perhaps you need a different storage engine. For example, sequences don't support analyze table, as it's an unnecessary operation on a sequence table, which only contains one row.

```
CREATE SEQUENCE s START WITH 100 INCREMENT BY 10;

ANALYZE TABLE s;
+--------+---------+----------+-----------------------------------------------------------+
| Table  | Op      | Msg_type | Msg_text                                                  |
+--------+---------+----------+-----------------------------------------------------------+
| test.s | analyze | note     | The storage engine for the table doesn't support analyze |
+--------+---------+----------+-----------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Error 1179: You are not allowed to execute this command in a transaction

Error Code	SQLSTATE	Error	Description
1179	25000	ER_CANT_DO_THIS_DURING_AN_TRANSACTION	You are not allowed to execute this command in a transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1180: Got error during COMMIT

Error Code	SQLSTATE	Error	Description
1180	HY000	ER_ERROR_DURING_COMMIT	Got error %d during COMMIT

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1181: Got error during ROLLBACK

Error Code	SQLSTATE	Error	Description
1181	HY000	ER_ERROR_DURING_ROLLBACK	Got error %d during ROLLBACK

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1182: Got error during FLUSH_LOGS

Error Code	SQLSTATE	Error	Description
1182	HY000	ER_ERROR_DURING_FLUSH_LOGS	Got error %d during FLUSH_LOGS

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1183: Got error during CHECKPOINT

Error Code	SQLSTATE	Error	Description
1183	HY000	ER_ERROR_DURING_CHECKPOINT	Got error %d during CHECKPOINT

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1184: Aborted connection to db user host

Error Code	SQLSTATE	Error	Description
1184	08S01	ER_NEW_ABORTING_CONNECTION	Aborted connection %ld to db: ‘%s’ user: ‘%s’ host: ‘%s’ (%s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1185: You should never see it

Error Code	SQLSTATE	Error	Description
1185	HY000	ER_UNUSED_10	You should never see it

Possible Causes and Solutions

Error 1186: Binlog closed, cannot RESET MASTER

Error Code	SQLSTATE	Error	Description
1186	HY000	ER_FLUSH_MASTER_BINLOG_CLOSED	Binlog closed, cannot RESET MASTER

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1187: Failed rebuilding the index of dumped table

Error Code	SQLSTATE	Error	Description
1187	HY000	ER_INDEX_REBUILD	Failed rebuilding the index of dumped table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1188: Error from master

Error Code	SQLSTATE	Error	Description
1188	HY000	ER_MASTER	Error from master: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1189: Net error reading from master

Error Code	SQLSTATE	Error	Description
1189	08S01	ER_MASTER_NET_READ	Net error reading from master

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1190: Net error writing to master

Error Code	SQLSTATE	Error	Description
1190	08S01	ER_MASTER_NET_WRITE	Net error writing to master

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1191: Can’t find FULLTEXT index matching the column list

Error Code	SQLSTATE	Error	Description
1191	HY000	ER_FT_MATCHING_KEY_NOT_FOUND	Can’t find FULLTEXT index matching the column list

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1192: Can't execute the given command because you have active locked tables or an active trans

Error 1192: Can't execute the given command because you have active locked tables or an active transaction

Error Code	SQLSTATE	Error	Description
1192	HY000	ER_LOCK_OR_ACTIVE_TRANSACTION	Can't execute the given command because you have active locked tables or an active transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1193: Unknown system variable

Error Code	SQLSTATE	Error	Description
1193	HY000	ER_UNKNOWN_SYSTEM_VARIABLE	Unknown system variable '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1194: Table is marked as crashed and should be repaired

Error Code	SQLSTATE	Error	Description
1194	HY000	ER_CRASHED_ON_USAGE	Table '%s' is marked as crashed and should be repaired

Possible Causes and Solutions

This error occurs, in particular to MyISAM, Aria, Archive or CSV tables, if the table is marked as crashed. This can occur for a number of reasons, for example:

- running out of disk space
- sudden shutdown while the table was in use

Fixing the problem can usually be done by running a REPAIR TABLE statement, or one of the similar scripts, aria_chk, myisamchk or mariadb-check.

This page is licensed: CC BY-SA / Gnu FDL

Error 1195: Table is marked as crashed and last (automatic?) repair failed

Error Code	SQLSTATE	Error	Description
1195	HY000	ER_CRASHED_ON_REPAIR	Table '%s' is marked as crashed and last (automatic?) repair failed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1196: Some non-transactional changed tables couldn't be rolled back

Error Code	SQLSTATE	Error	Description
1196	HY000	ER_WARNING_NOT_COMPLETE_ROLLBACK	Some non-transactional changed tables couldn't be rolled back

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1197: Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage

Error Code	SQLSTATE	Error	Description
1197	HY000	ER_TRANS_CACHE_FULL	Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage; increase this mysqld variable and try again

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1198: This operation cannot be performed with a running slave; run STOP SLAVE first

Error Code	SQLSTATE	Error	Description
1198	HY000	ER_SLAVE_MUST_STOP	This operation cannot be performed with a running slave; run STOP SLAVE first

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1199: This operation requires a running slave; configure slave and do START SLAVE

Error Code	SQLSTATE	Error	Description
1199	HY000	ER_SLAVE_NOT_RUNNING	This operation requires a running slave; configure slave and do START SLAVE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1200 to 1299

- Error 1200: The server is not configured as slave
- Error 1201: Could not initialize master info structure
- Error 1202: Could not create slave thread; check system resources
- Error 1203: User already has more than 'max_user_connections' active connections
- Error 1204: You may only use constant expressions with SET
- Error 1205: Lock wait timeout exceeded; try restarting transaction
- Error 1206: The total number of locks exceeds the lock table size
- Error 1207: Update locks cannot be acquired during a READ UNCOMMITTED transaction
- Error 1208: DROP DATABASE not allowed while thread is holding global read lock
- Error 1209: CREATE DATABASE not allowed while thread is holding global read lock
- Error 1210: Incorrect arguments to
- Error 1211: is not allowed to create new users
- Error 1212: Incorrect table definition; all MERGE tables must be in the same database
- Error 1213: Deadlock found when trying to get lock; try restarting transaction
- Error 1214: The used table type doesn't support FULLTEXT indexes
- Error 1215: Cannot add foreign key constraint
- Error 1216: Cannot add or update a child row: a foreign key constraint fails

- Error 1217: Cannot delete or update a parent row: a foreign key constraint fails
- Error 1218: Error connecting to master
- Error 1219: Error running query on master
- Error 1220: Error when executing command
- Error 1221: Incorrect usage of
- Error 1222: The used SELECT statements have a different number of columns
- Error 1223: Can’t execute the query because you have a conflicting read lock
- Error 1224: Mixing of transactional and non-transactional tables is disabled
- Error 1225: Option used twice in statement
- Error 1226: User has exceeded the resource
- Error 1227: Access denied; you need (at least one of) the privilege(s) for this operation
- Error 1228: Variable is a SESSION variable and can’t be used with SET GLOBAL
- Error 1229: Variable is a GLOBAL variable and should be set with SET GLOBAL
- Error 1230: Variable doesn’t have a default value
- Error 1231: Variable can’t be set to the value of
- Error 1232: Incorrect argument type to variable
- Error 1233: Variable can only be set, not read
- Error 1234: Incorrect usage/placement
- Error 1235: This version of MariaDB doesn’t yet support
- Error 1236: Got fatal error from master when reading data from binary log
- Error 1237: Slave SQL thread ignored the query because of replicate-*-table rules
- Error 1238: Variable is a
- Error 1239: Incorrect foreign key definition
- Error 1240: Key reference and table reference don’t match
- Error 1241: Operand should contain column(s)
- Error 1242: Subquery returns more than 1 row
- Error 1243: Unknown prepared statement handler given
- Error 1244: Help database is corrupt or does not exist
- Error 1245: Cyclic reference on subqueries
- Error 1246: Converting column from to
- Error 1247: Reference not supported
- Error 1248: Every derived table must have its own alias
- Error 1249: Select was reduced during optimization
- Error 1250: Table from one of the SELECTs cannot be used
- Error 1251: Client does not support authentication protocol requested by server
- Error 1252: All parts of a SPATIAL index must be NOT NULL
- Error 1253: COLLATION is not valid for CHARACTER SET
- Error 1254: Slave is already running
- Error 1255: Slave already has been stopped
- Error 1256: Uncompressed data size too large
- Error 1257: ZLIB: Not enough memory
- Error 1258: ZLIB: Not enough room in the output buffer
- Error 1259: ZLIB: Input data corrupted
- Error 1260: Row was cut by GROUP_CONCAT()
- Error 1261: Row doesn’t contain data for all columns

- Error 1262: Row was truncated; it contained more data than there were input columns
- Error 1263: Column set to default value; NULL supplied to NOT NULL columnError 1263: Column set to d
- Error 1264: Out of range value for column
- Error 1265: Data truncated for column
- Error 1266: Using storage engine for table
- Error 1267: Illegal mix of collations for operation
- Error 1268: Cannot drop one or more of the requested users
- Error 1269: Can't revoke all privileges for one or more of the requested users
- Error 1270: Illegal mix of collations for operation
- Error 1271: Illegal mix of collations for operation
- Error 1272: Variable is not a variable component (can't be used as XXXX.variable_name)
- Error 1273: Unknown collation
- Error 1274: SSL parameters in CHANGE MASTER are ignored because this MariaDB slave was compiled with
- Error 1275: Server is running in –secure-auth mode, but has a password in the old format
- Error 1276: Field or reference of SELECT was resolved in SELECT
- Error 1277: Incorrect parameter or combination of parameters for START SLAVE UNTIL
- Error 1278: It is recommended to use –skip-slave-start when doing step-by-step replication with STA
- Error 1279: SQL thread is not to be started so UNTIL options are ignored
- Error 1280: Incorrect index name
- Error 1281: Incorrect catalog name
- Error 1282: Query cache failed to set size; new query cache size is
- Error 1283: Column cannot be part of FULLTEXT index
- Error 1284: Unknown key cache
- Error 1285: MariaDB is started in –skip-name-resolve mode
- Error 1286: Unknown storage engine
- Error 1287: is deprecated and will be removed in a future release. Please use instead
- Error 1288: The target table of the is not updatable
- Error 1289: The feature is disabled; you need MariaDB built with to have it working
- Error 1290: The MariaDB server is running with the option so it cannot execute this statement
- Error 1291: Column has duplicated value
- Error 1292: Truncated incorrect value
- Error 1293: Incorrect table definition; there can be only one TIMESTAMP column with CURRENT_TIMESTAM
- Error 1294: Invalid ON UPDATE clause
- Error 1295: This command is not supported in the prepared statement protocol yet
- Error 1296: Got error from
- Error 1297: Got temporary error from
- Error 1298: Unknown or incorrect time zone
- Error 1299: Invalid TIMESTAMP value in column at row

This page is licensed: CC BY-SA / Gnu FDL

Error 1200: The server is not configured as slave

Error Code	SQLSTATE	Error	Description
1200	HY000	ER_BAD_SLAVE	The server is not configured as slave; fix in config file or with CHANGE MASTER TO

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1201: Could not initialize master info structure

Error Code	SQLSTATE	Error	Description
1201	HY000	ER_MASTER_INFO	Could not initialize master info structure; more error messages can be found in the MariaDB error log

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1202: Could not create slave thread; check system resources

Error Code	SQLSTATE	Error	Description
1202	HY000	ER_SLAVE_THREAD	Could not create slave thread; check system resources

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1203: User already has more than ‘max_user_connections’ active connections

Error Code	SQLSTATE	Error	Description
1203	42000	ER_TOO_MANY_USER_CONNECTIONS	User %s already has more than ‘max_user_connections’ active connections

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1204: You may only use constant expressions with SET

Error Code	SQLSTATE	Error	Description
1204	HY000	ER_SET_CONSTANTS_ONLY	You may only use constant expressions with SET

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1205: Lock wait timeout exceeded; try restarting transaction

Error Code	SQLSTATE	Error	Description
1205	HY000	ER_LOCK_WAIT_TIMEOUT	Lock wait timeout exceeded; try restarting transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1206: The total number of locks exceeds the lock table size

Error Code	SQLSTATE	Error	Description
1206	HY000	ER_LOCK_TABLE_FULL	The total number of locks exceeds the lock table size

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1207: Update locks cannot be acquired during a READ UNCOMMITTED transaction

Error Code	SQLSTATE	Error	Description
1207	25000	ER_READ_ONLY_TRANSACTION	Update locks cannot be acquired during a READ UNCOMMITTED transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1208: DROP DATABASE not allowed while thread is holding global read lock

Error Code	SQLSTATE	Error	Description
1208	HY000	ER_DROP_DB_WITH_READ_LOCK	DROP DATABASE not allowed while thread is holding global read lock

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1209: CREATE DATABASE not allowed while thread is holding global read lock

Error Code	SQLSTATE	Error	Description
1209	HY000	ER_CREATE_DB_WITH_READ_LOCK	CREATE DATABASE not allowed while thread is holding global read lock

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1210: Incorrect arguments to

Error Code	SQLSTATE	Error	Description
1210	HY000	ER_WRONG_ARGUMENTS	Incorrect arguments to %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1211: is not allowed to create new users

Error Code	SQLSTATE	Error	Description
1211	42000	ER_NO_PERMISSION_TO_CREATE_USER	'%s'@'%s' is not allowed to create new users

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1212: Incorrect table definition; all MERGE tables must be in the same database

Error Code	SQLSTATE	Error	Description
1212	HY000	ER_UNION_TABLES_IN_DIFFERENT_DIR	Incorrect table definition; all MERGE tables must be in the same database

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1213: Deadlock found when trying to get lock; try restarting transaction

Error Code	SQLSTATE	Error	Description
1213	40001	ER_LOCK_DEADLOCK	Deadlock found when trying to get lock; try restarting transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1214: The used table type doesn’t support FULLTEXT indexes

Error Code	SQLSTATE	Error	Description
1214	HY000	ER_TABLE_CANT_HANDLE_FT	The used table type doesn’t support FULLTEXT indexes

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1215: Cannot add foreign key constraint

Error Code	SQLSTATE	Error	Description
1215	HY000	ER_CANNOT_ADD_FOREIGN	Cannot add foreign key constraint

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1216: Cannot add or update a child row: a foreign key constraint fails

Error Code	SQLSTATE	Error	Description
1216	23000	ER_NO_REFERENCED_ROW	Cannot add or update a child row: a foreign key constraint fails

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1217: Cannot delete or update a parent row: a foreign key constraint fails

Error Code	SQLSTATE	Error	Description
1217	23000	ER_ROW_IS_REFERENCED	Cannot delete or update a parent row: a foreign key constraint fails

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1218: Error connecting to master

Error Code	SQLSTATE	Error	Description
1218	08S01	ER_CONNECT_TO_MASTER	Error connecting to master: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1219: Error running query on master

Error Code	SQLSTATE	Error	Description
1219	HY000	ER_QUERY_ON_MASTER	Error running query on master: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1220: Error when executing command

Error Code	SQLSTATE	Error	Description
1220	HY000	ER_ERROR_WHEN_EXECUTING_COMMAND	Error when executing command %s: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1221: Incorrect usage of

Error Code	SQLSTATE	Error	Description
1221	HY000	ER_WRONG_USAGE	Incorrect usage of %s and %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1222: The used SELECT statements have a different number of columns

Error Code	SQLSTATE	Error	Description
1222	21000	ER_WRONG_NUMBER_OF_COLUMNS_IN_SELECT	The used SELECT statements have a different number of columns

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1223: Can’t execute the query because you have a conflicting read lock

Error Code	SQLSTATE	Error	Description
1223	HY000	ER_CANT_UPDATE_WITH_READLOCK	Can’t execute the query because you have a conflicting read lock

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1224: Mixing of transactional and non-transactional tables is disabled

Error Code	SQLSTATE	Error	Description
1224	HY000	ER_MIXING_NOT_ALLOWED	Mixing of transactional and non-transactional tables is disabled

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1225: Option used twice in statement

Error Code	SQLSTATE	Error	Description
1225	HY000	ER_DUP_ARGUMENT	Option ‘%s’ used twice in statement

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1226: User has exceeded the resource

Error Code	SQLSTATE	Error	Description
1226	42000	ER_USER_LIMIT_REACHED	User ‘%s’ has exceeded the ‘%s’ resource (current value: %ld)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1227: Access denied; you need (at least one of) the privilege(s) for this operation

Error Code	SQLSTATE	Error	Description
1227	42000	ER_SPECIFIC_ACCESS_DENIED_ERROR	Access denied; you need (at least one of) the %s privilege(s) for this operation

Possible Causes and Solutions

The user is trying to access a database/schema/object they do not have access to. The error shows which GRANT is needed to access the resource.

See Also

- When access_denied_errors status variable is incremented

This page is licensed: CC BY-SA / Gnu FDL

Error 1228: Variable is a SESSION variable and can’t be used with SET GLOBAL

Error Code	SQLSTATE	Error	Description
1228	HY000	ER_LOCAL_VARIABLE	Variable ‘%s’ is a SESSION variable and can’t be used with SET GLOBAL

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1229: Variable is a GLOBAL variable and should be set with SET GLOBAL

Error Code	SQLSTATE	Error	Description
1229	HY000	ER_GLOBAL_VARIABLE	Variable ‘%s’ is a GLOBAL variable and should be set with SET GLOBAL

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1230: Variable doesn’t have a default value

Error Code	SQLSTATE	Error	Description
1230	42000	ER_NO_DEFAULT	Variable ‘%s’ doesn’t have a default value

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1231: Variable can’t be set to the value of

Error Code	SQLSTATE	Error	Description
1231	42000	ER_WRONG_VALUE_FOR_VAR	Variable ‘%s’ can’t be set to the value of ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1232: Incorrect argument type to variable

Error Code	SQLSTATE	Error	Description
1232	42000	ER_WRONG_TYPE_FOR_VAR	Incorrect argument type to variable ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1233: Variable can only be set, not read

Error Code	SQLSTATE	Error	Description
1233	HY000	ER_VAR_CANT_BE_READ	Variable ‘%s’ can only be set, not read

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1234: Incorrect usage/placement

Error Code	SQLSTATE	Error	Description
1234	42000	ER_CANT_USE_OPTION_HERE	Incorrect usage/placement of ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1235: This version of MariaDB doesn't yet support

Error Code	SQLSTATE	Error	Description
1235	42000	ER_NOT_SUPPORTED_YET	This version of MariaDB doesn't yet support '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1236: Got fatal error from master when reading data from binary log

Error Code	SQLSTATE	Error	Description
1236	HY000	ER_MASTER_FATAL_ERROR_READING_BINLOG	Got fatal error %d from master when reading data from binary log: '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1237: Slave SQL thread ignored the query because of replicate-*-table rules

Error Code	SQLSTATE	Error	Description
1237	HY000	ER_SLAVE_IGNORED_TABLE	Slave SQL thread ignored the query because of replicate-*-table rules

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1238: Variable is a

Error Code	SQLSTATE	Error	Description
1238	HY000	ER_INCORRECT_GLOBAL_LOCAL_VAR	Variable '%s' is a %s variable

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1239: Incorrect foreign key definition

Error Code	SQLSTATE	Error	Description
1239	42000	ER_WRONG_FK_DEF	Incorrect foreign key definition for '%s': %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1240: Key reference and table reference don't match

Error Code	SQLSTATE	Error	Description
1240	HY000	ER_KEY_REF_DO_NOT_MATCH_TABLE_REF	Key reference and table reference don't match

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1241: Operand should contain column(s)

Error Code	SQLSTATE	Error	Description
1241	21000	ER_OPERAND_COLUMNS	Operand should contain %d column(s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1242: Subquery returns more than 1 row

Error Code	SQLSTATE	Error	Description
1242	21000	ER_SUBQUERY_NO_1_ROW	Subquery returns more than 1 row

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1243: Unknown prepared statement handler given

Error Code	SQLSTATE	Error	Description
1243	HY000	ER_UNKNOWN_STMT_HANDLER	Unknown prepared statement handler (%.*s) given to %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1244: Help database is corrupt or does not exist

Error Code	SQLSTATE	Error	Description
1244	HY000	ER_CORRUPT_HELP_DB	Help database is corrupt or does not exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1245: Cyclic reference on subqueries

Error Code	SQLSTATE	Error	Description
1245	HY000	ER_CYCLIC_REFERENCE	Cyclic reference on subqueries

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1246: Converting column from to

Error Code	SQLSTATE	Error	Description
1246	HY000	ER_AUTO_CONVERT	Converting column '%s' from %s to %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1247: Reference not supported

Error Code	SQLSTATE	Error	Description
1247	42S22	ER_ILLEGAL_REFERENCE	Reference ‘%s’ not supported (%s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1248: Every derived table must have its own alias

Error Code	SQLSTATE	Error	Description
1248	42000	ER_DERIVED_MUST_HAVE_ALIAS	Every derived table must have its own alias

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1249: Select was reduced during optimization

Error Code	SQLSTATE	Error	Description
1249	01000	ER_SELECT_REDUCED	Select %u was reduced during optimization

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1250: Table from one of the SELECTs cannot be used

Error Code	SQLSTATE	Error	Description
1250	42000	ER_TABLENAME_NOT_ALLOWED_HERE	Table ‘%s’ from one of the SELECTs cannot be used in %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1251: Client does not support authentication protocol requested by server

Error Code	SQLSTATE	Error	Description
1251	08004	ER_NOT_SUPPORTED_AUTH_MODE	Client does not support authentication protocol requested by server; consider upgrading MariaDB client

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1252: All parts of a SPATIAL index must be NOT NULL

Error Code	SQLSTATE	Error	Description
1252	42000	ER_SPATIAL_CANT_HAVE_NULL	All parts of a SPATIAL index must be NOT NULL

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1253: COLLATION is not valid for CHARACTER SET

Error Code	SQLSTATE	Error	Description
1253	42000	ER_COLLATION_CHARSET_MISMATCH	COLLATION ‘%s’ is not valid for CHARACTER SET ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1254: Slave is already running

Error Code	SQLSTATE	Error	Description
1254	HY000	ER_SLAVE_WAS_RUNNING	Slave is already running

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1255: Slave already has been stopped

Error Code	SQLSTATE	Error	Description
1255	HY000	ER_SLAVE_WAS_NOT_RUNNING	Slave already has been stopped

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1256: Uncompressed data size too large

Error Code	SQLSTATE	Error	Description
1256	HY000	ER_TOO_BIG_FOR_UNCOMPRESS	Uncompressed data size too large; the maximum size is %d (probably, length of uncompressed data was corrupted)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1257: ZLIB: Not enough memory

Error Code	SQLSTATE	Error	Description
1257	HY000	ER_ZLIB_Z_MEM_ERROR	ZLIB: Not enough memory

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1258: ZLIB: Not enough room in the output buffer

Error Code	SQLSTATE	Error	Description
1258	HY000	ER_ZLIB_Z_BUF_ERROR	ZLIB: Not enough room in the output buffer (probably, length of uncompressed data was corrupted)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1259: ZLIB: Input data corrupted

Error Code	SQLSTATE	Error	Description
1259	HY000	ER_ZLIB_Z_DATA_ERROR	ZLIB: Input data corrupted

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1260: Row was cut by GROUP_CONCAT()

Error Code	SQLSTATE	Error	Description
1260	HY000	ER_CUT_VALUE_GROUP_CONCAT	Row %u was cut by GROUP_CONCAT()

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1261: Row doesn’t contain data for all columns

Error Code	SQLSTATE	Error	Description
1261	01000	ER_WARN_TOO_FEW_RECORDS	Row %ld doesn’t contain data for all columns

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1262: Row was truncated; it contained more data than there were input columns

Error Code	SQLSTATE	Error	Description
1262	01000	ER_WARN_TOO_MANY_RECORDS	Row %ld was truncated; it contained more data than there were input columns

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1263: Column set to default value; NULL supplied to NOT NULL columnError 1263: Column set to d

Error Code	SQLSTATE	Error	Description
1263	22004	ER_WARN_NULL_TO_NOTNULL	Column set to default value; NULL supplied to NOT NULL column ‘%s’ at row %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1264: Out of range value for column

Error Code	SQLSTATE	Error	Description
1260	HY000	ER_CUT_VALUE_GROUP_CONCAT	Row %u was cut by GROUP_CONCAT()

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1265: Data truncated for column

Error Code	SQLSTATE	Error	Description
1265	01000	WARN_DATA_TRUNCATED	Data truncated for column ‘%s’ at row %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1266: Using storage engine for table

Error Code	SQLSTATE	Error	Description
1266	HY000	ER_WARN_USING_OTHER_HANDLER	Using storage engine %s for table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1267: Illegal mix of collations for operation

Error Code	SQLSTATE	Error	Description
1267	HY000	ER_CANT_AGGREGATE_2COLLATIONS	Illegal mix of collations (%s,%s) and (%s,%s) for operation ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1268: Cannot drop one or more of the requested users

Error Code	SQLSTATE	Error	Description
1268	HY000	ER_DROP_USER	Cannot drop one or more of the requested users

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1269: Can’t revoke all privileges for one or more of the requested users

Error Code	SQLSTATE	Error	Description
1269	HY000	ER_REVOKE_GRANTS	Can’t revoke all privileges for one or more of the requested users

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1270: Illegal mix of collations for operation

Error Code	SQLSTATE	Error	Description
1270	HY000	ER_CANT_AGGREGATE_3COLLATIONS	Illegal mix of collations (%s,%s), (%s,%s), (%s,%s) for operation '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1271: Illegal mix of collations for operation

Error Code	SQLSTATE	Error	Description
1271	HY000	ER_CANT_AGGREGATE_NCOLLATIONS	Illegal mix of collations for operation '%s'

Possible Causes and Solutions

This error occurs when attempting to combine (e.g., using UNION) two tables where the same column has different collation types. To resolve this, one of the column's collations must be adjusted to match the other.

Diagnosing Collation Types

To identify the collation type of a column, you can use the following query:

```
SELECT COLUMN_NAME, COLLATION_NAME
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'your_database_name'
AND TABLE_NAME = 'your_table_name';
```

For a comprehensive comparison, you might want to join `table1` and `table2` based on the `COLUMN_NAME` to visually inspect the collation differences.

Resolving Collation Conflicts

Assuming one column is using `utf8mb3_general_ci` and the other `utf8mb3_unicode_ci`, you can standardize the collation for a particular operation as follows:

```
FROM table;
SELECT COLUMN COLLATE utf8mb3_unicode_ci
```

This command temporarily adjusts the collation for `column` during the SELECT operation to `utf8mb3_unicode_ci`, ensuring consistent collation for the operation.

This page is licensed: CC BY-SA / Gnu FDL

Error 1272: Variable is not a variable component (can't be used as XXXX.variable_name)

Error Code	SQLSTATE	Error	Description
1272	HY000	ER_VARIABLE_IS_NOT_STRUCT	Variable '%s' is not a variable component (can't be used as XXXX.variable_name)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1273: Unknown collation

Error Code	SQLSTATE	Error	Description
1273	HY000	ER_UNKNOWN_COLLATION	Unknown collation: '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1274: SSL parameters in CHANGE MASTER are ignored because this MariaDB slave was compiled with

Error 1274: SSL parameters in CHANGE MASTER are ignored because this MariaDB slave was compiled without SSL support

Error Code	SQLSTATE	Error	Description
1274	HY000	ER_SLAVE_IGNORED_SSL_PARAMS	SSL parameters in CHANGE MASTER are ignored because this MariaDB slave was compiled without SSL support; they can be used later if MariaDB slave with SSL is started

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1275: Server is running in –secure-auth mode, but has a password in the old format

Error Code	SQLSTATE	Error	Description
1275	HY000	ER_SERVER_IS_IN_SECURE_AUTH_MODE	Server is running in –secure-auth mode, but '%s'@'%s' has a password in the old format; please change the password to the new format

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1276: Field or reference of SELECT was resolved in SELECT

Error Code	SQLSTATE	Error	Description
1276	HY000	ER_WARN_FIELD_RESOLVED	Field or reference '%s%s%s%s%s' of SELECT #%d was resolved in SELECT #%d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1277: Incorrect parameter or combination of parameters for START SLAVE UNTIL

Error Code	SQLSTATE	Error	Description
1277	HY000	ER_BAD_SLAVE_UNTIL_COND	Incorrect parameter or combination of parameters for START SLAVE UNTIL

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1278: It is recommended to use –skip-slave-start when doing step-by-step replication with STA

Error 1278: It is recommended to use –skip-slave-start when doing step-by-step replication with START SLAVE UNTIL

Error Code	SQLSTATE	Error	Description
1278	HY000	ER_MISSING_SKIP_SLAVE	It is recommended to use –skip-slave-start when doing step-by-step replication with START SLAVE UNTIL; otherwise, you will get problems if you get an unexpected slave's mysqld restart

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1279: SQL thread is not to be started so UNTIL options are ignored

Error Code	SQLSTATE	Error	Description
1279	HY000	ER_UNTIL_COND_IGNORED	SQL thread is not to be started so UNTIL options are ignored

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1280: Incorrect index name

Error Code	SQLSTATE	Error	Description
1280	42000	ER_WRONG_NAME_FOR_INDEX	Incorrect index name '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1281: Incorrect catalog name

Error Code	SQLSTATE	Error	Description
1281	42000	ER_WRONG_NAME_FOR_CATALOG	Incorrect catalog name '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1282: Query cache failed to set size; new query cache size is

Error Code	SQLSTATE	Error	Description
1282	HY000	ER_WARN_QC_RESIZE	Query cache failed to set size %lu; new query cache size is %lu

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1283: Column cannot be part of FULLTEXT index

Error Code	SQLSTATE	Error	Description
1283	HY000	ER_BAD_FT_COLUMN	Column '%s' cannot be part of FULLTEXT index

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1284: Unknown key cache

Error Code	SQLSTATE	Error	Description
1284	HY000	ER_UNKNOWN_KEY_CACHE	Unknown key cache ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1285: MariaDB is started in –skip-name-resolve mode

Error Code	SQLSTATE	Error	Description
1285	HY000	ER_WARN_HOSTNAME_WONT_WORK	MariaDB is started in –skip-name-resolve mode; you must restart it without this switch for this grant to work

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1286: Unknown storage engine

Error Code	SQLSTATE	Error	Description
1286	42000	ER_UNKNOWN_STORAGE_ENGINE	Unknown storage engine ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1287: is deprecated and will be removed in a future release. Please use instead

Error Code	SQLSTATE	Error	Description
1287	HY000	ER_WARN_DEPRECATED_SYNTAX	‘%s’ is deprecated and will be removed in a future release. Please use %s instead

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1288: The target table of the is not updatable

Error Code	SQLSTATE	Error	Description
1288	HY000	ER_NON_UPDATABLE_TABLE	The target table %s of the %s is not updatable

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1289: The feature is disabled; you need MariaDB built with to have it working

Error Code	SQLSTATE	Error	Description
1289	HY000	ER_FEATURE_DISABLED	The '%s' feature is disabled; you need MariaDB built with '%s' to have it working

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1290: The MariaDB server is running with the option so it cannot execute this statement

Error Code	SQLSTATE	Error	Description
1290	HY000	ER_OPTION_PREVENTS_STATEMENT	The MariaDB server is running with the %s option so it cannot execute this statement

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1291: Column has duplicated value

Error Code	SQLSTATE	Error	Description
1291	HY000	ER_DUPLICATED_VALUE_IN_TYPE	Column '%s' has duplicated value '%s' in %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1292: Truncated incorrect value

Error Code	SQLSTATE	Error	Description
1292	22007	ER_TRUNCATED_WRONG_VALUE	Truncated incorrect %s value: '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1293: Incorrect table definition; there can be only one TIMESTAMP column with CURRENT_TIMESTAM

Error Code	SQLSTATE	Error	Description
1293	HY000	ER_TOO_MUCH_AUTO_TIMESTAMP_COLS	Incorrect table definition; there can be only one TIMESTAMP column with CURRENT_TIMESTAMP in DEFAULT or ON UPDATE clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1294: Invalid ON UPDATE clause

Error Code	SQLSTATE	Error	Description
1294	HY000	ER_INVALID_ON_UPDATE	Invalid ON UPDATE clause for '%s' column

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1295: This command is not supported in the prepared statement protocol yet

Error Code	SQLSTATE	Error	Description
1295	HY000	ER_UNSUPPORTED_PS	This command is not supported in the prepared statement protocol yet

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1296: Got error from

Error Code	SQLSTATE	Error	Description
1296	HY000	ER_GET_ERRMSG	Got error %d '%s' from %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1297: Got temporary error from

Error Code	SQLSTATE	Error	Description
1297	HY000	ER_GET_TEMPORARY_ERRMSG	Got temporary error %d '%s' from %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1298: Unknown or incorrect time zone

Error Code	SQLSTATE	Error	Description
1298	HY000	ER_UNKNOWN_TIME_ZONE	Unknown or incorrect time zone: '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1299: Invalid TIMESTAMP value in column at row

Error Code	SQLSTATE	Error	Description
1299	HY000	ER_WARN_INVALID_TIMESTAMP	Invalid TIMESTAMP value in column '%s' at row %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1300 to 1399

- Error 1300: Invalid character string
- Error 1301: Result was larger than max_allowed_packet - truncated
- Error 1302: Conflicting declarations
- Error 1303: Can't create from within another stored routine

- Error 1304: already exists
- Error 1305: does not exist
- Error 1306: Failed to DROP
- Error 1307: Failed to CREATE
- Error 1308: with no matching label
- Error 1309: Redefining label
- Error 1310: End-label without match
- Error 1311: Referring to uninitialized variable
- Error 1312: PROCEDURE can’t return a result set in the given context
- Error 1313: RETURN is only allowed in a FUNCTION
- Error 1314: is not allowed in stored procedures
- Error 1315: The update log is deprecated and replaced by the binary log
- Error 1316: The update log is deprecated and replaced by the binary log; SET SQL_LOG_UPDATE has been
- Error 1317: Query execution was interrupted
- Error 1318: Incorrect number of arguments; expected, got
- Error 1319: Undefined CONDITION
- Error 1320: No RETURN found in FUNCTION
- Error 1321: FUNCTION ended without RETURN
- Error 1322: Cursor statement must be a SELECT
- Error 1323: Cursor SELECT must not have INTO
- Error 1324: Undefined CURSOR
- Error 1325: Cursor is already open
- Error 1326: Cursor is not open
- Error 1327: Undeclared variable
- Error 1328: Incorrect number of FETCH variables
- Error 1329: No data - zero rows fetched, selected, or processed
- Error 1330: Duplicate parameter
- Error 1331: Duplicate variable
- Error 1332: Duplicate condition
- Error 1333: Duplicate cursor
- Error 1334: Failed to ALTER
- Error 1335: Subquery value not supported
- Error 1336: is not allowed in stored function or trigger
- Error 1337: Variable or condition declaration after cursor or handler declaration
- Error 1338: Cursor declaration after handler declaration
- Error 1339: Case not found for CASE statement
- Error 1340: Configuration file is too big
- Error 1341: Malformed file type header in file
- Error 1342: Unexpected end of file while parsing comment
- Error 1343: Error while parsing parameter
- Error 1344: Unexpected end of file while skipping unknown parameter
- Error 1345: EXPLAIN/SHOW can not be issued; lacking privileges for underlying table
- Error 1346: File has unknown type in its header
- Error 1347: ‘%s.%s’ is not %s
- Error 1348: Column is not updatable

- Error 1349: View’s SELECT contains a subquery in the FROM clause
- Error 1350: View’s SELECT contains clause
- Error 1351: View’s SELECT contains a variable or parameter
- Error 1352: View’s SELECT refers to a temporary table’
- Error 1353: View’s SELECT and view’s field list have different column counts
- Error 1354: View merge algorithm can’t be used here for now
- Error 1355: View being updated does not have complete key of underlying table in it
- Error 1356: View references invalid table(s) or column(s) or function(s) or definer/invoker of view
- Error 1357: Can’t drop or alter from within another stored routine
- Error 1358: GOTO is not allowed in a stored procedure handler
- Error 1359: Trigger already exists
- Error 1360: Trigger does not exist
- Error 1361: Trigger’s is view or temporary table
- Error 1362: Updating of row is not allowed in trigger
- Error 1363: There is no row in trigger
- Error 1364: Field doesn’t have a default value
- Error 1365: Division by 0
- Error 1366: Incorrect value for column at row
- Error 1367: Illegal value found during parsing
- Error 1368: CHECK OPTION on non-updatable view
- Error 1369: CHECK OPTION failed
- Error 1370: command denied to user for routine
- Error 1371: Failed purging old relay logs
- Error 1372: Password hash should be a %d-digit hexadecimal number
- Error 1373: Target log not found in binlog index
- Error 1374: I/O error reading log index file
- Error 1375: Server configuration does not permit binlog purge
- Error 1376: Failed on fseek()
- Error 1377: Fatal error during log purge
- Error 1378: A purgeable log is in use, will not purge
- Error 1379: Unknown error during log purge
- Error 1380: Failed initializing relay log position
- Error 1381: You are not using binary logging
- Error 1382: The syntax is reserved for purposes internal to the MariaDB server
- Error 1383: WSAStartup Failed
- Error 1384: Can’t handle procedures with different groups yet
- Error 1385: Select must have a group with this procedure
- Error 1386: Can’t use ORDER clause with this procedure
- Error 1387: Binary logging and replication forbid changing the global server
- Error 1388: Can’t map file
- Error 1389: Wrong magic in
- Error 1390: Prepared statement contains too many placeholders
- Error 1391: Key part length cannot be 0
- Error 1392: View text checksum failed
- Error 1393: Can not modify more than one base table through a join view

- Error 1394: Can not insert into join view without fields list
- Error 1395: Can not delete from join view
- Error 1396: Operation failed for
- Error 1397: XAER_NOTA: Unknown XID
- Error 1398: XAER_INVAL: Invalid arguments (or unsupported command)
- Error 1399: XAER_RMFAIL: The command cannot be executed when global transaction is in the state

This page is licensed: CC BY-SA / Gnu FDL

Error 1300: Invalid character string

Error Code	SQLSTATE	Error	Description
1300	HY000	ER_INVALID_CHARACTER_STRING	Invalid %s character string: '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1301: Result was larger than max_allowed_packet - truncated

Error Code	SQLSTATE	Error	Description
1301	HY000	ER_WARN_ALLOWED_PACKET_OVERFLOWED	Result of %s() was larger than max_allowed_packet (%ld) - truncated

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1302: Conflicting declarations

Error Code	SQLSTATE	Error	Description
1302	HY000	ER_CONFLICTING_DECLARATIONS	Conflicting declarations: '%s%s' and '%s%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1303: Can't create from within another stored routine

Error Code	SQLSTATE	Error	Description
1303	2F003	ER_SP_NO_RECURSIVE_CREATE	Can't create a %s from within another stored routine

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1304: already exists

Error Code	SQLSTATE	Error	Description
1304	42000	ER_SP_ALREADY_EXISTS	%s %s already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1305: does not exist

Error Code	SQLSTATE	Error	Description
1305	42000	ER_SP_DOES_NOT_EXIST	%s %s does not exist

Possible Causes and Solutions

This error is returned when what is being called does not exist. For example:

```
CALL Reset_animal_count();
ERROR 1305 (42000): PROCEDURE test.Reset_animal_count does not exist

SELECT Reset_animal_count();
ERROR 1305 (42000): FUNCTION test.Reset_animal_count does not exist
```

There are a number of possible causes:

- The stored procedure or stored function has not yet been created. See CREATE PROCEDURE or CREATE FUNCTION. For example:

```
DELIMITER //
CREATE PROCEDURE Reset_animal_count()
    MODIFIES SQL DATA
    UPDATE animal_count SET animals = 0;
 //
Query OK, 0 rows affected (0.032 sec)

DELIMITER ;

CALL Reset_animal_count();
Query OK, 0 rows affected (0.001 sec)
```

- There was a typo in the name. Check also that the case is correct. See Identifier Case-sensitivity. For example:

```
CALL reset_animal_count();
ERROR 1305 (42000): PROCEDURE test.Reset_animal_count does not exist

CALL Reset_animal_count();
Query OK, 0 rows affected (0.001 sec)
```

- The database specified is not the same as the database containing the stored procedure or function. For example:

```
use test2
CALL Reset_animal_count();
ERROR 1305 (42000): PROCEDURE test2.Reset_animal_count does not exist
```

Either change the default (current) database with the USE statement, or specify the database in the call, for example:

```
CALL test.Reset_animal_count();
Query OK, 0 rows affected (0.001 sec)
```

or

```
use test
CALL Reset_animal_count();
Query OK, 0 rows affected (0.001 sec)
```

- One is trying to access a stored procedure instead of a stored function, or vice-versa. For example:

```
SELECT Reset_animal_count();
ERROR 1305 (42000): FUNCTION test.Reset_animal_count does not exist

CALL Reset_animal_count();
Query OK, 0 rows affected (0.001 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

Error 1306: Failed to DROP

Error Code	SQLSTATE	Error	Description
1306	HY000	ER_SP_DROP_FAILED	Failed to DROP %s %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1307: Failed to CREATE

Error Code	SQLSTATE	Error	Description
1307	HY000	ER_SP_STORE_FAILED	Failed to CREATE %s %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1308: with no matching label

Error Code	SQLSTATE	Error	Description
1308	42000	ER_SP_LILABEL_MISMATCH	%s with no matching label: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1309: Redefining label

Error Code	SQLSTATE	Error	Description
1309	42000	ER_SP_LABEL_REDEFINE	Redefining label %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1310: End-label without match

Error Code	SQLSTATE	Error	Description
1310	42000	ER_SP_LABEL_MISMATCH	End-label %s without match

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1311: Referring to uninitialized variable

Error Code	SQLSTATE	Error	Description
1311	01000	ER_SP_UNINIT_VAR	Referring to uninitialized variable %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1312: PROCEDURE can't return a result set in the given context

Error Code	SQLSTATE	Error	Description
1312	0A000	ER_SP_BADSELECT	PROCEDURE %s can't return a result set in the given context

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1313: RETURN is only allowed in a FUNCTION

Error Code	SQLSTATE	Error	Description
1313	42000	ER_SP_BADRETURN	RETURN is only allowed in a FUNCTION

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1314: is not allowed in stored procedures

Error Code	SQLSTATE	Error	Description
1314	0A000	ER_SP_BADSTATEMENT	%s is not allowed in stored procedures

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1315: The update log is deprecated and replaced by the binary log

Error Code	SQLSTATE	Error	Description
1315	42000	ER_UPDATE_LOG_DEPRECATED_IGNORED	The update log is deprecated and replaced by the binary log; SET SQL_LOG_UPDATE has been ignored. This option will be removed in MariaDB 5.6.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1316: The update log is deprecated and replaced by the binary log; SET SQL_LOG_UPDATE has been

Error Code	SQLSTATE	Error	Description
1316	42000	ER_UPDATE_LOG_DEPRECATED_TRANSLATED	The update log is deprecated and replaced by the binary log; SET SQL_LOG_UPDATE has been translated to SET SQL_LOG_BIN. This option will be removed in MariaDB 5.6.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1317: Query execution was interrupted

Error Code	SQLSTATE	Error	Description
1317	70100	ER_QUERY_INTERRUPTED	Query execution was interrupted

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1318: Incorrect number of arguments; expected, got

Error Code	SQLSTATE	Error	Description
1318	42000	ER_SP_WRONG_NO_OF_ARGS	Incorrect number of arguments for %s %s; expected %u, got %u

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1319: Undefined CONDITION

Error Code	SQLSTATE	Error	Description
1319	42000	ER_SP_COND_MISMATCH	Undefined CONDITION: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1320: No RETURN found in FUNCTION

Error Code	SQLSTATE	Error	Description
1320	42000	ER_SP_NORETURN	No RETURN found in FUNCTION %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1321: FUNCTION ended without RETURN

Error Code	SQLSTATE	Error	Description
1321	2F005	ER_SP_NORETURNEND	FUNCTION %s ended without RETURN

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1322: Cursor statement must be a SELECT

Error Code	SQLSTATE	Error	Description
1322	42000	ER_SP_BAD_CURSOR_QUERY	Cursor statement must be a SELECT

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1323: Cursor SELECT must not have INTO

Error Code	SQLSTATE	Error	Description
1323	42000	ER_SP_BAD_CURSOR_SELECT	Cursor SELECT must not have INTO

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1324: Undefined CURSOR

Error Code	SQLSTATE	Error	Description
1324	42000	ER_SP_CURSOR_MISMATCH	Undefined CURSOR: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1325: Cursor is already open

Error Code	SQLSTATE	Error	Description
1325	24000	ER_SP_CURSOR_ALREADY_OPEN	Cursor is already open

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1326: Cursor is not open

Error Code	SQLSTATE	Error	Description
1326	24000	ER_SP_CURSOR_NOT_OPEN	Cursor is not open

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1327: Undeclared variable

Error Code	SQLSTATE	Error	Description
1327	42000	ER_SP_UNDECLARED_VAR	Undeclared variable: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1328: Incorrect number of FETCH variables

Error Code	SQLSTATE	Error	Description
1328	HY000	ER_SP_WRONG_NO_OF_FETCH_ARGS	Incorrect number of FETCH variables

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1329: No data - zero rows fetched, selected, or processed

Error Code	SQLSTATE	Error	Description
1329	02000	ER_SP_FETCH_NO_DATA	No data - zero rows fetched, selected, or processed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1330: Duplicate parameter

Error Code	SQLSTATE	Error	Description
1330	42000	ER_SP_DUP_PARAM	Duplicate parameter: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1331: Duplicate variable

Error Code	SQLSTATE	Error	Description
1331	42000	ER_SP_DUP_VAR	Duplicate variable: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1332: Duplicate condition

Error Code	SQLSTATE	Error	Description
1332	42000	ER_SP_DUP_COND	Duplicate condition: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1333: Duplicate cursor

Error Code	SQLSTATE	Error	Description
1333	42000	ER_SP_DUP_CURS	Duplicate cursor: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1334: Failed to ALTER

Error Code	SQLSTATE	Error	Description
1334	HY000	ER_SP_CANT_ALTER	Failed to ALTER %s %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1335: Subquery value not supported

Error Code	SQLSTATE	Error	Description
1335	0A000	ER_SP_SUBSELECT_NYI	Subquery value not supported

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1336: is not allowed in stored function or trigger

Error Code	SQLSTATE	Error	Description
1336	0A000	ER_STMT_NOT_ALLOWED_IN_SF_OR_TRG	%s is not allowed in stored function or trigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1337: Variable or condition declaration after cursor or handler declaration

Error Code	SQLSTATE	Error	Description
1337	42000	ER_SP_VARCOND_AFTER_CURSHNDLR	Variable or condition declaration after cursor or handler declaration

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1338: Cursor declaration after handler declaration

Error Code	SQLSTATE	Error	Description
1338	42000	ER_SP_CURSOR_AFTER_HANDLER	Cursor declaration after handler declaration

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1339: Case not found for CASE statement

Error Code	SQLSTATE	Error	Description
1339	20000	ER_SP_CASE_NOT_FOUND	Case not found for CASE statement

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1340: Configuration file is too big

Error Code	SQLSTATE	Error	Description
1340	HY000	ER_FPARSER_TOO_BIG_FILE	Configuration file ‘%s’ is too big

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1341: Malformed file type header in file

Error Code	SQLSTATE	Error	Description
1341	HY000	ER_FPARSER_BAD_HEADER	Malformed file type header in file ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1342: Unexpected end of file while parsing comment

Error Code	SQLSTATE	Error	Description
1342	HY000	ER_FPARSER_EOF_IN_COMMENT	Unexpected end of file while parsing comment ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1343: Error while parsing parameter

Error Code	SQLSTATE	Error	Description
1343	HY000	ER_FPARSER_ERROR_IN_PARAMETER	Error while parsing parameter ‘%s’ (line: ‘%s’)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1344: Unexpected end of file while skipping unknown parameter

Error Code	SQLSTATE	Error	Description
1344	HY000	ER_FPARSER_EOF_IN_UNKNOWN_PARAMETER	Unexpected end of file while skipping unknown parameter ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1345: EXPLAIN/SHOW can not be issued; lacking privileges for underlying table

Error Code	SQLSTATE	Error	Description
1345	HY000	ER_VIEW_NO_EXPLAIN	EXPLAIN/SHOW can not be issued; lacking privileges for underlying table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1346: File has unknown type in its header

Error Code	SQLSTATE	Error	Description
1346	HY000	ER_FRM_UNKNOWN_TYPE	File ‘%s’ has unknown type ‘%s’ in its header

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1347: '%s.%s' is not %s

Error Code	SQLSTATE	Error	Description
1347	HY000	ER_WRONG_OBJECT	'%s.%s' is not %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1348: Column is not updatable

Error Code	SQLSTATE	Error	Description
1348	HY000	ER_NONUPDATEABLE_COLUMN	Column '%s' is not updatable

Possible Causes and Solutions

The usual cause is that an UPDATE, LOAD DATA or LOAD XML, is being performed on the column of a view where the view column is an expression.

This page is licensed: CC BY-SA / Gnu FDL

Error 1349: View's SELECT contains a subquery in the FROM clause

Error Code	SQLSTATE	Error	Description
1349	HY000	ER_VIEW_SELECT_DERIVED	View's SELECT contains a subquery in the FROM clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1350: View's SELECT contains clause

Error Code	SQLSTATE	Error	Description
1350	HY000	ER_VIEW_SELECT_CLAUSE	View's SELECT contains a '%s' clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1351: View's SELECT contains a variable or parameter

Error Code	SQLSTATE	Error	Description
1351	HY000	ER_VIEW_SELECT_VARIABLE	View's SELECT contains a variable or parameter

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1352: View's SELECT refers to a temporary table'

Error Code	SQLSTATE	Error	Description
1352	HY000	ER_VIEW_SELECT_TMPTABLE	View's SELECT refers to a temporary table '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1353: View's SELECT and view's field list have different column counts

Error Code	SQLSTATE	Error	Description
1353	HY000	ER_VIEW_WRONG_LIST	View's SELECT and view's field list have different column counts

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1354: View merge algorithm can't be used here for now

Error Code	SQLSTATE	Error	Description
1354	HY000	ER_WARN_VIEW_MERGE	View merge algorithm can't be used here for now (assumed undefined algorithm)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1355: View being updated does not have complete key of underlying table in it

Error Code	SQLSTATE	Error	Description
1355	HY000	ER_WARN_VIEW_WITHOUT_KEY	View being updated does not have complete key of underlying table in it

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1356: View references invalid table(s) or column(s) or function(s) or definer/invoker of view

Error Code	SQLSTATE	Error	Description
1356	HY000	ER_VIEW_INVALID	View '%s.%s' references invalid table(s) or column(s) or function(s) or definer/invoker of view lack rights to use them

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1357: Can't drop or alter from within another stored routine

Error Code	SQLSTATE	Error	Description
1357	HY000	ER_SP_NO_DROP_SP	Can't drop or alter a %s from within another stored routine

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1358: GOTO is not allowed in a stored procedure handler

Error Code	SQLSTATE	Error	Description
1358	HY000	ER_SP_GOTO_IN_HNDLR	GOTO is not allowed in a stored procedure handler

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1359: Trigger already exists

Error Code	SQLSTATE	Error	Description
1359	HY000	ER_TRG_ALREADY_EXISTS	Trigger already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1360: Trigger does not exist

Error Code	SQLSTATE	Error	Description
1360	HY000	ER_TRG_DOES_NOT_EXIST	Trigger does not exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1361: Trigger's is view or temporary table

Error Code	SQLSTATE	Error	Description
1361	HY000	ER_TRG_ON_VIEW_OR_TEMP_TABLE	Trigger's '%s' is view or temporary table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1362: Updating of row is not allowed in trigger

Error Code	SQLSTATE	Error	Description
1362	HY000	ER_TRG_CANT_CHANGE_ROW	Updating of %s row is not allowed in %strigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1363: There is no row in trigger

Error Code	SQLSTATE	Error	Description
1363	HY000	ER_TRG_NO_SUCH_ROW_IN_TRG	There is no %s row in %s trigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1364: Field doesn't have a default value

Error Code	SQLSTATE	Error	Description
1364	HY000	ER_NO_DEFAULT_FOR_FIELD	Field '%s' doesn't have a default value

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1365: Division by 0

Error Code	SQLSTATE	Error	Description
1365	22012	ER_DIVISION_BY_ZER	Division by 0

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1366: Incorrect value for column at row

Error Code	SQLSTATE	Error	Description
1366	HY000	ER_TRUNCATED_WRONG_VALUE_FOR_FIELD	Incorrect %s value: '%s' for column '%s' at row %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1367: Illegal value found during parsing

Error Code	SQLSTATE	Error	Description
1367	22007	ER_ILLEGAL_VALUE_FOR_TYPE	Illegal %s '%s' value found during parsing

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1368: CHECK OPTION on non-updatable view

Error Code	SQLSTATE	Error	Description
1368	HY000	ER_VIEW_NONUPD_CHECK	CHECK OPTION on non-updatable view '%s.%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1369: CHECK OPTION failed

Error Code	SQLSTATE	Error	Description
1369	HY000	ER_VIEW_CHECK_FAILED	CHECK OPTION failed '%s.%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1370: command denied to user for routine

Error Code	SQLSTATE	Error	Description
1370	42000	ER_PROCACCESS_DENIED_ERROR	%s command denied to user ‘%s’@‘%s’ for routine ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1371: Failed purging old relay logs

Error Code	SQLSTATE	Error	Description
1371	HY000	ER_RELAY_LOG_FAIL	Failed purging old relay logs: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1372: Password hash should be a %d-digit hexadecimal number

Error Code	SQLSTATE	Error	Description
1372	HY000	ER_PASSWD_LENGTH	Password hash should be a %d-digit hexadecimal number

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1373: Target log not found in binlog index

Error Code	SQLSTATE	Error	Description
1373	HY000	ER_UNKNOWN_TARGET_BINLOG	Target log not found in binlog index

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1374: I/O error reading log index file

Error Code	SQLSTATE	Error	Description
1374	HY000	ER_IO_ERR_LOG_INDEX_READ	I/O error reading log index file

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1375: Server configuration does not permit binlog purge

Error Code	SQLSTATE	Error	Description
1375	HY000	ER_BINLOG_PURGE_PROHIBITED	Server configuration does not permit binlog purge

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1376: Failed on fseek()

Error Code	SQLSTATE	Error	Description
1376	HY000	ER_FSEEK_FAIL	Failed on fseek()

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1377: Fatal error during log purge

Error Code	SQLSTATE	Error	Description
1377	HY000	ER_BINLOG_PURGE_FATAL_ERR	Fatal error during log purge

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1378: A purgeable log is in use, will not purge

Error Code	SQLSTATE	Error	Description
1378	HY000	ER_LOG_IN_USE	A purgeable log is in use, will not purge

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1379: Unknown error during log purge

Error Code	SQLSTATE	Error	Description
1379	HY000	ER_LOG_PURGE_UNKNOWN_ERR	Unknown error during log purge

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1380: Failed initializing relay log position

Error Code	SQLSTATE	Error	Description
1380	HY000	ER_RELAY_LOG_INIT	Failed initializing relay log position: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1381: You are not using binary logging

Error Code	SQLSTATE	Error	Description
1381	HY000	ER_NO_BINARY_LOGGING	You are not using binary logging

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1382: The syntax is reserved for purposes internal to the MariaDB server

Error Code	SQLSTATE	Error	Description
1382	HY000	ER_RESERVED_SYNTAX	The '%s' syntax is reserved for purposes internal to the MariaDB server

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1383: WSAStartup Failed

Error Code	SQLSTATE	Error	Description
1383	HY000	ER_WSAS_FAILED	WSAStartup Failed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1384: Can't handle procedures with different groups yet

Error Code	SQLSTATE	Error	Description
1384	HY000	ER_DIFF_GROUPS_PROC	Can't handle procedures with different groups yet

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1385: Select must have a group with this procedure

Error Code	SQLSTATE	Error	Description
1385	HY000	ER_NO_GROUP_FOR_PROC	Select must have a group with this procedure

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1386: Can't use ORDER clause with this procedure

Error Code	SQLSTATE	Error	Description
1386	HY000	ER_ORDER_WITH_PROC	Can't use ORDER clause with this procedure

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1387: Binary logging and replication forbid changing the global server

Error Code	SQLSTATE	Error	Description
1387	HY000	ER_LOGGING_PROHIBIT_CHANGING_OF	Binary logging and replication forbid changing the global server %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1388: Can’t map file

Error Code	SQLSTATE	Error	Description
1388	HY000	ER_NO_FILE_MAPPING	Can’t map file: %s, errno: %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1389: Wrong magic in

Error Code	SQLSTATE	Error	Description
1389	HY000	ER_WRONG_MAGIC	Wrong magic in %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1390: Prepared statement contains too many placeholders

Error Code	SQLSTATE	Error	Description
1390	HY000	ER_PS_MANY_PARAM	Prepared statement contains too many placeholders

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1391: Key part length cannot be 0

Error Code	SQLSTATE	Error	Description
1391	HY000	ER_KEY_PART_0	Key part ‘%s’ length cannot be 0

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1392: View text checksum failed

Error Code	SQLSTATE	Error	Description
1392	HY000	ER_VIEW_CHECKSUM	View text checksum failed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1393: Can not modify more than one base table through a join view

Error Code	SQLSTATE	Error	Description
1393	HY000	ER_VIEW_MULTIUPDATE	Can not modify more than one base table through a join view ‘%s.%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1394: Can not insert into join view without fields list

Error Code	SQLSTATE	Error	Description
1394	HY000	ER_VIEW_NO_INSERT_FIELD_LIST	Can not insert into join view ‘%s.%s’ without fields list

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1395: Can not delete from join view

Error Code	SQLSTATE	Error	Description
1395	HY000	ER_VIEW_DELETE_MERGE_VIEW	Can not delete from join view ‘%s.%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1396: Operation failed for

Error Code	SQLSTATE	Error	Description
1396	HY000	ER_CANNOT_USER	Operation %s failed for %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1397: XAER_NOTA: Unknown XID

Error Code	SQLSTATE	Error	Description
1397	XAE04	ER_XAER_NOTA	XAER_NOTA: Unknown XID

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1398: XAER_INVAL: Invalid arguments (or unsupported command)

Error Code	SQLSTATE	Error	Description
1398	XAE05	ER_XAER_INVAL	XAER_INVAL: Invalid arguments (or unsupported command)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1399: XAER_RMFAIL: The command cannot be executed when global transaction is in the state

Error Code	SQLSTATE	Error	Description
1399	XAE07	ER_XAER_RMFAIL	XAER_RMFAIL: The command cannot be executed when global transaction is in the %s state

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1400 to 1499

- Error 1400: XAER_OUTSIDE: Some work is done outside global transaction
- Error 1401: XAER_RMERR: Fatal error occurred in the transaction branch - check your data for consist
- Error 1402: XA_RBROLLBACK: Transaction branch was rolled back
- Error 1403: There is no such grant defined for user on host on routine
- Error 1404: Failed to grant EXECUTE and ALTER ROUTINE privileges
- Error 1405: Failed to revoke all privileges to dropped routine
- Error 1406: Data too long for column at row
- Error 1407: Bad SQLSTATE
- Error 1408: ready for connections. Version: socket: port:
- Error 1409: Can’t load value from file with fixed size rows to variable
- Error 1410: You are not allowed to create a user with GRANT
- Error 1411: Incorrect value for function
- Error 1412: Table definition has changed, please retry transaction
- Error 1413: Duplicate handler declared in the same block
- Error 1414: OUT or INOUT argument for routine is not a variable or NEW pseudo-variable in BEFORE tri
- Error 1415: Not allowed to return a result set from
- Error 1416: Cannot get geometry object from data you send to the GEOMETRY field
- Error 1417: A routine failed and has neither NO SQL nor READS SQL DATA in its declaration
- Error 1418: This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration an
- Error 1419: You do not have the SUPER privilege and binary logging is enabled
- Error 1420: You can’t execute a prepared statement which has an open cursor associated with it
- Error 1421: The statement has no open cursor
- Error 1422: Explicit or implicit commit is not allowed in stored function or trigger
- Error 1423: Field of view underlying table doesn’t have a default value
- Error 1424: Recursive stored functions and triggers are not allowed.
- Error 1425: Too big scale specified for column
- Error 1426: Too big precision specified for column
- Error 1427: For float(M,D, double(M,D or decimal(M,D, M must be >= D
- Error 1428: You can’t combine write-locking of system tables with other tables or lock types
- Error 1429: Unable to connect to foreign data source
- Error 1430: There was a problem processing the query on the foreign data source
- Error 1431: The foreign data source you are trying to reference does not exist
- Error 1432: Can’t create federated table. The data source connection string is not in the correct format
- Error 1433: The data source connection string is not in the correct format
- Error 1434: Can’t create federated table. Foreign data src error
- Error 1435: Trigger in wrong schema
- Error 1436: Thread stack overrun
- Error 1437: Routine body is too long
- Error 1438: Cannot drop default keycache
- Error 1439: Display width out of range for column
- Error 1440: XAER_DUPID: The XID already exists
- Error 1441: Datetime function: field overflow
- Error 1442: Can’t update table in stored function/trigger because it is already used by statement wh

- Error 1443: The definition of table prevents operation on table
- Error 1444: The prepared statement contains a stored routine call that refers to that same statement
- Error 1445: Not allowed to set autocommit from a stored function or trigger
- Error 1446: Definer is not fully qualified
- Error 1447: View has no definer information (old table format)
- Error 1448: You need the SUPER privilege for creation view with definer
- Error 1449: The user specified as a definer does not exist
- Error 1450: Changing schema from is not allowed.
- Error 1451: Cannot delete or update a parent row: a foreign key constraint fails
- Error 1452: Cannot add or update a child row: a foreign key constraint fails
- Error 1453: Variable must be quoted with …, or renamed
- Error 1454: No definer attribute for trigger
- Error 1455: has an old format, you should re-create the object(s)
- Error 1456: Recursive limit (as set by the max_sp_recursion_depth variable) was exceeded for routine
- Error 1457: Failed to load routine
- Error 1458: Incorrect routine name
- Error 1459: Table upgrade required. Please do “REPAIR TABLE” or dump/reload to fix it!
- Error 1460: AGGREGATE is not supported for stored functions
- Error 1461: Can’t create more than max_prepared_stmt_count statements
- Error 1462: contains view recursion
- Error 1463: Non-grouping field is used in clause
- Error 1464: The used table type doesn’t support SPATIAL indexes
- Error 1465: Triggers can not be created on system tables
- Error 1466: Leading spaces are removed from name
- Error 1467: Failed to read auto-increment value from storage engine
- Error 1468: user name
- Error 1469: host name
- Error 1470: String is too long for (should be no longer than)
- Error 1471: The target table of the is not insertable-into
- Error 1472: Table is differently defined or of non-MyISAM type or doesn’t exist
- Error 1473: Too high level of nesting for select
- Error 1474: Name has become ’’
- Error 1475: First character of the FIELDS TERMINATED string is ambiguous
- Error 1476: The foreign server you are trying to create already exists
- Error 1477: The foreign server name you are trying to reference does not exist. Data source error
- Error 1478: Table storage engine does not support the create option
- Error 1479: Syntax error: PARTITIONING requires definition of VALUES for each partition
- Error 1480: Only PARTITIONING can use VALUES in partition definition
- Error 1481: MAXVALUE can only be used in last partition definition
- Error 1482: Subpartitions can only be hash partitions and by key
- Error 1483: Must define subpartitions on all partitions if on one partition
- Error 1484: Wrong number of partitions defined, mismatch with previous setting
- Error 1485: Wrong number of subpartitions defined, mismatch with previous setting
- Error 1486: Constant/Random expression in (sub)partitioning function is not allowed
- Error 1487: Expression in RANGE/LIST VALUES must be constant

- Error 1488: Field in list of fields for partition function not found in table
- Error 1489: List of fields is only allowed in KEY partitions
- Error 1490: The partition info in the frm file is not consistent with what can be written into the f
- Error 1491: The function returns the wrong type
- Error 1492: For partitions each partition must be defined
- Error 1493: VALUES LESS THAN value must be strictly increasing for each partition
- Error 1494: VALUES value must be of same type as partition function
- Error 1495: Multiple definition of same constant in list partitioning
- Error 1496: Partitioning can not be used stand-alone in query
- Error 1497: The mix of handlers in the partitions is not allowed in this version of MariaDB
- Error 1498: For the partitioned engine it is necessary to define all
- Error 1499: Too many partitions (including subpartitions) were defined

This page is licensed: CC BY-SA / Gnu FDL

Error 1400: XAER_OUTSIDE: Some work is done outside global transaction

Error Code	SQLSTATE	Error	Description
1400	XAE09	ER_XAER_OUTSIDE	XAER_OUTSIDE: Some work is done outside global transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1401: XAER_RMERR: Fatal error occurred in the transaction branch - check your data for consist

Error Code	SQLSTATE	Error	Description
1401	XAE03	ER_XAER_RMERR	XAER_RMERR: Fatal error occurred in the transaction branch - check your data for consistency

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1402: XA_RBROLLBACK: Transaction branch was rolled back

Error Code	SQLSTATE	Error	Description
1402	XA100	ER_XA_RBROLLBACK	XA_RBROLLBACK: Transaction branch was rolled back

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1403: There is no such grant defined for user on host on routine

Error Code	SQLSTATE	Error	Description
1403	42000	ER_NONEXISTING_PROC_GRANT	There is no such grant defined for user ‘%s’ on host ‘%s’ on routine ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1404: Failed to grant EXECUTE and ALTER ROUTINE privileges

Error Code	SQLSTATE	Error	Description
1404	HY000	ER_PROC_AUTO_GRANT_FAIL	Failed to grant EXECUTE and ALTER ROUTINE privileges

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1405: Failed to revoke all privileges to dropped routine

Error Code	SQLSTATE	Error	Description
1405	HY000	ER_PROC_AUTO_REVOKE_FAIL	Failed to revoke all privileges to dropped routine

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1406: Data too long for column at row

Error Code	SQLSTATE	Error	Description
1406	22001	ER_DATA_TOO_LONG	Data too long for column ‘%s’ at row %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1407: Bad SQLSTATE

Error Code	SQLSTATE	Error	Description
1407	42000	ER_SP_BAD_SQLSTATE	Bad SQLSTATE: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1408: ready for connections. Version: socket: port:

Error Code	SQLSTATE	Error	Description
1408	HY000	ER_STARTUP	%s: ready for connections. Version: ‘%s’ socket: ‘%s’ port: %d %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1409: Can’t load value from file with fixed size rows to variable

Error Code	SQLSTATE	Error	Description
1409	HY000	ER_LOAD_FROM_FIXED_SIZE_ROWS_TO_VAR	Can’t load value from file with fixed size rows to variable

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1410: You are not allowed to create a user with GRANT

Error Code	SQLSTATE	Error	Description
1410	42000	ER_CANT_CREATE_USER_WITH_GRANT	You are not allowed to create a user with GRANT

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1411: Incorrect value for function

Error Code	SQLSTATE	Error	Description
1411	HY000	ER_WRONG_VALUE_FOR_TYPE	Incorrect %s value: '%s' for function %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1412: Table definition has changed, please retry transaction

Error Code	SQLSTATE	Error	Description
1412	HY000	ER_TABLE_DEF_CHANGED	Table definition has changed, please retry transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1413: Duplicate handler declared in the same block

Error Code	SQLSTATE	Error	Description
1413	42000	ER_SP_DUP_HANDLER	Duplicate handler declared in the same block

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1414: OUT or INOUT argument for routine is not a variable or NEW pseudo-variable in BEFORE tri

Error 1414: OUT or INOUT argument for routine is not a variable or NEW pseudo-variable in BEFORE trigger

Error Code	SQLSTATE	Error	Description
1414	42000	ER_SP_NOT_VAR_ARG	OUT or INOUT argument %d for routine %s is not a variable or NEW pseudo-variable in BEFORE trigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1415: Not allowed to return a result set from

Error Code	SQLSTATE	Error	Description
1415	0A000	ER_SP_NO_RETSET	Not allowed to return a result set from a %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1416: Cannot get geometry object from data you send to the GEOMETRY field

Error Code	SQLSTATE	Error	Description
1416	22003	ER_CANT_CREATE_GEOMETRY_OBJECT	Cannot get geometry object from data you send to the GEOMETRY field

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1417: A routine failed and has neither NO SQL nor READS SQL DATA in its declaration

Error Code	SQLSTATE	Error	Description
1417	HY000	ER_FAILED_ROUTINE_BREAK_BINLOG	A routine failed and has neither NO SQL nor READS SQL DATA in its declaration and binary logging is enabled; if non-transactional tables were updated, the binary log will miss their changes

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1418: This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration an

Error 1418: This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled

Error Code	SQLSTATE	Error	Description
1418	HY000	ER_BINLOG_UNSAFE_ROUTINE	This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled (you *might* want to use the less safe log_bin_trust_function_creators variable)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1419: You do not have the SUPER privilege and binary logging is enabled

Error Code	SQLSTATE	Error	Description
1419	HY000	ER_BINLOG_CREATE_ROUTINE_NEED_SUPER	You do not have the SUPER privilege and binary logging is enabled (you *might* want to use the less safe log_bin_trust_function_creators variable)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1420: You can't execute a prepared statement which has an open cursor associated with it

Error Code	SQLSTATE	Error	Description
1420	HY000	ER_EXEC_STMT_WITH_OPEN_CURSOR	You can't execute a prepared statement which has an open cursor associated with it. Reset the statement to re-execute it.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1421: The statement has no open cursor

Error Code	SQLSTATE	Error	Description
1421	HY000	ER_STMT_HAS_NO_OPEN_CURSOR	The statement (%lu) has no open cursor.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1422: Explicit or implicit commit is not allowed in stored function or trigger

Error Code	SQLSTATE	Error	Description
1422	HY000	ER_COMMIT_NOT_ALLOWED_IN_SF_OR_TRG	Explicit or implicit commit is not allowed in stored function or trigger.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1423: Field of view underlying table doesn't have a default value

Error Code	SQLSTATE	Error	Description
1423	HY000	ER_NO_DEFAULT_FOR_VIEW_FIELD	Field of view '%s.%s' underlying table doesn't have a default value

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1424: Recursive stored functions and triggers are not allowed.

Error Code	SQLSTATE	Error	Description
1424	HY000	ER_SP_NO_RECURSION	Recursive stored functions and triggers are not allowed.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1425: Too big scale specified for column

Error Code	SQLSTATE	Error	Description
1425	42000	ER_TOO_BIG_SCALE	Too big scale %d specified for column ‘%s’. Maximum is %lu.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1426: Too big precision specified for column

Error Code	SQLSTATE	Error	Description
1426	42000	ER_TOO_BIG_PRECISION	Too big precision %d specified for column ‘%s’. Maximum is %lu.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1427: For float(M,D, double(M,D or decimal(M,D, M must be >= D

Error Code	SQLSTATE	Error	Description
1427	42000	ER_M_BIGGER_THAN_D	For float(M,D, double(M,D or decimal(M,D, M must be >= D (column ‘%s’).

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1428: You can’t combine write-locking of system tables with other tables or lock types

Error Code	SQLSTATE	Error	Description
1428	HY000	ER_WRONG_LOCK_OF_SYSTEM_TABLE	You can’t combine write-locking of system tables with other tables or lock types

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1429: Unable to connect to foreign data source

Error Code	SQLSTATE	Error	Description
1429	HY000	ER_CONNECT_TO_FOREIGN_DATA_SOURCE	Unable to connect to foreign data source: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1430: There was a problem processing the query on the foreign data source

Error Code	SQLSTATE	Error	Description
1430	HY000	ER_QUERY_ON_FOREIGN_DATA_SOURCE	There was a problem processing the query on the foreign data source. Data source error: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1431: The foreign data source you are trying to reference does not exist

Error Code	SQLSTATE	Error	Description
1431	HY000	ER_FOREIGN_DATA_SOURCE_DOESNT_EXIST	The foreign data source you are trying to reference does not exist. Data source error: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1432: Can’t create federated table. The data source connection string is not in the correct format

Error Code	SQLSTATE	Error	Description
1432	HY000	ER_FOREIGN_DATA_STRING_INVALID_CANT_CREATE	Can’t create federated table. The data source connection string ‘%s’ is not in the correct format

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1433: The data source connection string is not in the correct format

Error Code	SQLSTATE	Error	Description
1433	HY000	ER_FOREIGN_DATA_STRING_INVALID	The data source connection string ‘%s’ is not in the correct format

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1434: Can’t create federated table. Foreign data src error

Error Code	SQLSTATE	Error	Description
1434	HY000	ER_CANT_CREATE_FEDERATED_TABLE	Can’t create federated table. Foreign data src error: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1435: Trigger in wrong schema

Error Code	SQLSTATE	Error	Description
1435	HY000	ER_TRG_IN_WRONG_SCHEMA	Trigger in wrong schema

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1436: Thread stack overrun

Error Code	SQLSTATE	Error	Description
1436	HY000	ER_STACK_OVERRUN_NEED_MORE	Thread stack overrun: %ld bytes used of a %ld byte stack, and %ld bytes needed. Use ‘mysqld –thread_stack=#’ to specify a bigger stack.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1437: Routine body is too long

Error Code	SQLSTATE	Error	Description
1437	42000	ER_TOO_LONG_BODY	Routine body for ‘%s’ is too long

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1438: Cannot drop default keycache

Error Code	SQLSTATE	Error	Description
1438	HY000	ER_WARN_CANT_DROP_DEFAULT_KEYCACHE	Cannot drop default keycache

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1439: Display width out of range for column

Error Code	SQLSTATE	Error	Description
1439	42000	ER_TOO_BIG_DISPLAYWIDTH	Display width out of range for column ‘%s’ (max = %lu)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1440: XAER_DUPID: The XID already exists

Error Code	SQLSTATE	Error	Description
1440	XAE08	ER_XAER_DUPID	XAER_DUPID: The XID already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1441: Datetime function: field overflow

Error Code	SQLSTATE	Error	Description
1441	22008	ER_DATETIME_FUNCTION_OVERFLOW	Datetime function: %s field overflow

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1442: Can't update table in stored function/trigger because it is already used by statement wh

Error Code	SQLSTATE	Error	Description
1442	HY000	ER_CANT_UPDATE_USED_TABLE_IN_SF_OR_TRG	Can't update table '%s' in stored function/trigger because it is already used by statement which invoked this stored function/trigger.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1443: The definition of table prevents operation on table

Error Code	SQLSTATE	Error	Description
1443	HY000	ER_VIEW_PREVENT_UPDATE	The definition of table '%s' prevents operation %s on table '%s'.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1444: The prepared statement contains a stored routine call that refers to that same statement

Error Code	SQLSTATE	Error	Description
1444	HY000	ER_PS_NO_RECURSION	The prepared statement contains a stored routine call that refers to that same statement. It's not allowed to execute a prepared statement in such a recursive manner

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1445: Not allowed to set autocommit from a stored function or trigger

Error Code	SQLSTATE	Error	Description
1445	HY000	ER_SP_CANT_SET_AUTOCOMMIT	Not allowed to set autocommit from a stored function or trigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1446: Definer is not fully qualified

Error Code	SQLSTATE	Error	Description
1446	HY000	ER_MALFORMED_DEFINER	Definer is not fully qualified

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1447: View has no definer information (old table format)

Error Code	SQLSTATE	Error	Description
1447	HY000	ER_VIEW_FRM_NO_USER	View '%s'.'%s' has no definer information (old table format). Current user is used as definer. Please recreate the view!

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1448: You need the SUPER privilege for creation view with definer

Error 1448: You should never see it

Error Code	SQLSTATE	Error	Description
1448	HY000	ER_UNUSED_30	You should never see it

Possible Causes and Solutions

Error 1449: The user specified as a definer does not exist

Error Code	SQLSTATE	Error	Description
1449	HY000	ER_NO_SUCH_USER	The user specified as a definer (‘%s’@‘%s’) does not exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1450: Changing schema from is not allowed.

Error Code	SQLSTATE	Error	Description
1450	HY000	ER_FORBID_SCHEMA_CHANGE	Changing schema from ‘%s’ to ‘%s’ is not allowed.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1451: Cannot delete or update a parent row: a foreign key constraint fails

Error Code	SQLSTATE	Error	Description
1451	23000	ER_ROW_IS_REFERENCED_2	Cannot delete or update a parent row: a foreign key constraint fails (%s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1452: Cannot add or update a child row: a foreign key constraint fails

Error Code	SQLSTATE	Error	Description
1452	23000	ER_NO_REFERENCED_ROW_2	Cannot add or update a child row: a foreign key constraint fails (%s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1453: Variable must be quoted with …, or renamed

Error Code	SQLSTATE	Error	Description
1453	42000	ER_SP_BAD_VAR_SHADOW	Variable ‘%s’ must be quoted with `...`, or renamed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1454: No definer attribute for trigger

Error Code	SQLSTATE	Error	Description
1454	HY000	ER_TRG_NO_DEFINER	No definer attribute for trigger ‘%s’.’%s’. The trigger will be activated under the authorization of the invoker, which may have insufficient privileges. Please recreate the trigger.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1455: has an old format, you should re-create the object(s)

Error Code	SQLSTATE	Error	Description
1455	HY000	ER_OLD_FILE_FORMAT	‘%s’ has an old format, you should re-create the ‘%s’ object(s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1456: Recursive limit (as set by the max_sp_recursion_depth variable) was exceeded for routine

Error Code	SQLSTATE	Error	Description
1456	HY000	ER_SP_RECURSION_LIMIT	Recursive limit %d (as set by the max_sp_recursion_depth variable) was exceeded for routine %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1457: Failed to load routine

Error Code	SQLSTATE	Error	Description
1457	HY000	ER_SP_PROC_TABLE_CORRUPT	Failed to load routine %s. The table mysql.proc is missing, corrupt, or contains bad data (internal code %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1458: Incorrect routine name

Error Code	SQLSTATE	Error	Description
1458	42000	ER_SP_WRONG_NAME	Incorrect routine name ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1459: Table upgrade required. Please do “REPAIR TABLE” or dump/reload to fix it!

Error Code	SQLSTATE	Error	Description
1459	HY000	ER_TABLE_NEEDS_UPGRADE	Table upgrade required. Please do “REPAIR TABLE `%s` ” or dump/reload to fix it!

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1460: AGGREGATE is not supported for stored functions

Error Code	SQLSTATE	Error	Description
1460	42000	ER_SP_NO_AGGREGATE	AGGREGATE is not supported for stored functions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1461: Can’t create more than max_prepared_stmt_count statements

Error Code	SQLSTATE	Error	Description
1461	42000	ER_MAX_PREPARED_STMT_COUNT_REACHED	Can’t create more than max_prepared_stmt_count statements (current value: %lu)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1462: contains view recursion

Error Code	SQLSTATE	Error	Description
1462	HY000	ER_VIEW_RECURSIVE	`%s`.`%s` contains view recursion

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1463: Non-grouping field is used in clause

Error Code	SQLSTATE	Error	Description
1463	42000	ER_NON_GROUPING_FIELD_USED	Non-grouping field ‘%s’ is used in %s clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1464: The used table type doesn’t support SPATIAL indexes

Error Code	SQLSTATE	Error	Description
1464	HY000	ER_TABLE_CANT_HANDLE_SPKEYS	The used table type doesn’t support SPATIAL indexes

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1465: Triggers can not be created on system tables

Error Code	SQLSTATE	Error	Description
1465	HY000	ER_NO_TRIGGERS_ON_SYSTEM_SCHEMA	Triggers can not be created on system tables

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1466: Leading spaces are removed from name

Error Code	SQLSTATE	Error	Description
1466	HY000	ER_REMOVED_SPACES	Leading spaces are removed from name ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1467: Failed to read auto-increment value from storage engine

Error Code	SQLSTATE	Error	Description
1467	HY000	ER_AUTOINC_READ_FAILED	Failed to read auto-increment value from storage engine

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1468: user name

Error Code	SQLSTATE	Error	Description
1468	HY000	ER_USERNAME	user name

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1469: host name

Error Code	SQLSTATE	Error	Description
1469	HY000	ER_HOSTNAME	host name

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1470: String is too long for (should be no longer than)

Error Code	SQLSTATE	Error	Description
1470	HY000	ER_WRONG_STRING_LENGTH	String ‘%s’ is too long for %s (should be no longer than %d)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1471: The target table of the is not insertable-into

Error Code	SQLSTATE	Error	Description
1471	HY000	ER_NON_INSERTABLE_TABLE	The target table %s of the %s is not insertable-into

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1472: Table is differently defined or of non-MyISAM type or doesn’t exist

Error Code	SQLSTATE	Error	Description
1472	HY000	ER_ADMIN_WRONG_MRG_TABLE	Table ‘%s’ is differently defined or of non-MyISAM type or doesn’t exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1473: Too high level of nesting for select

Error Code	SQLSTATE	Error	Description
1473	HY000	ER_TOO_HIGH_LEVEL_OF_NESTING_FOR_SELECT	Too high level of nesting for select

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1474: Name has become ”

Error Code	SQLSTATE	Error	Description
1474	HY000	ER_NAME_BECOMES_EMPTY	Name ‘%s’ has become ”

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1475: First character of the FIELDS TERMINATED string is ambiguous

Error Code	SQLSTATE	Error	Description
1475	HY000	ER_AMBIGUOUS_FIELD_TERM	First character of the FIELDS TERMINATED string is ambiguous; please use non-optional and non-empty FIELDS ENCLOSED BY

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1476: The foreign server you are trying to create already exists

Error Code	SQLSTATE	Error	Description
1476	HY000	ER_FOREIGN_SERVER_EXISTS	The foreign server, %s, you are trying to create already exists.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1477: The foreign server name you are trying to reference does not exist. Data source error

Error Code	SQLSTATE	Error	Description
1477	HY000	ER_FOREIGN_SERVER_DOESNT_EXIST	The foreign server name you are trying to reference does not exist. Data source error: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1478: Table storage engine does not support the create option

Error Code	SQLSTATE	Error	Description
1478	HY000	ER_ILLEGAL_HA_CREATE_OPTION	Table storage engine ‘%s’ does not support the create option ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1479: Syntax error: PARTITIONING requires definition of VALUES for each partition

Error Code	SQLSTATE	Error	Description
1479	HY000	ER_PARTITION_REQUIRES_VALUES_ERROR	Syntax error: %s PARTITIONING requires definition of VALUES %s for each partition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1480: Only PARTITIONING can use VALUES in partition definition

Error Code	SQLSTATE	Error	Description
1480	HY000	ER_PARTITION_WRONG_VALUES_ERROR	Only %s PARTITIONING can use VALUES %s in partition definition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1481: MAXVALUE can only be used in last partition definition

Error Code	SQLSTATE	Error	Description
1481	HY000	ER_PARTITION_MAXVALUE_ERROR	MAXVALUE can only be used in last partition definition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1482: Subpartitions can only be hash partitions and by key

Error Code	SQLSTATE	Error	Description
1482	HY000	ER_PARTITION_SUBPARTITION_ERROR	Subpartitions can only be hash partitions and by key

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1483: Must define subpartitions on all partitions if on one partition

Error Code	SQLSTATE	Error	Description
1483	HY000	ER_PARTITION_SUBPART_MIX_ERROR	Must define subpartitions on all partitions if on one partition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1484: Wrong number of partitions defined, mismatch with previous setting

Error Code	SQLSTATE	Error	Description
1484	HY000	ER_PARTITION_WRONG_NO_PART_ERROR	Wrong number of partitions defined, mismatch with previous setting

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1485: Wrong number of subpartitions defined, mismatch with previous setting

Error Code	SQLSTATE	Error	Description
1485	HY000	ER_PARTITION_WRONG_NO_SUBPART_ERROR	Wrong number of subpartitions defined, mismatch with previous setting

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1486: Constant/Random expression in (sub)partitioning function is not allowed

Error Code	SQLSTATE	Error	Description
1486	HY000	ER_CONST_EXPR_IN_PARTITION_FUNC_ERROR	Constant/Random expression in (sub)partitioning function is not allowed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1487: Expression in RANGE/LIST VALUES must be constant

Error Code	SQLSTATE	Error	Description
1487	HY000	ER_NO_CONST_EXPR_IN_RANGE_OR_LIST_ERROR	Expression in RANGE/LIST VALUES must be constant

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1488: Field in list of fields for partition function not found in table

Error Code	SQLSTATE	Error	Description
1488	HY000	ER_FIELD_NOT_FOUND_PART_ERROR	Field in list of fields for partition function not found in table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1489: List of fields is only allowed in KEY partitions

Error Code	SQLSTATE	Error	Description
1489	HY000	ER_LIST_OF_FIELDS_ONLY_IN_HASH_ERROR	List of fields is only allowed in KEY partitions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1490: The partition info in the frm file is not consistent with what can be written into the f

Error 1490: The partition info in the frm file is not consistent with what can be written into the frm file

Error Code	SQLSTATE	Error	Description
1490	HY000	ER_INCONSISTENT_PARTITION_INFO_ERROR	The partition info in the frm file is not consistent with what can be written into the frm file

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1491: The function returns the wrong type

Error Code	SQLSTATE	Error	Description
1491	HY000	ER_PARTITION_FUNC_NOT_ALLOWED_ERROR	The %s function returns the wrong type

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1492: For partitions each partition must be defined

Error Code	SQLSTATE	Error	Description
1492	HY000	ER_PARTITIONS_MUST_BE_DEFINED_ERROR	For %s partitions each partition must be defined

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1493: VALUES LESS THAN value must be strictly increasing for each partition

Error Code	SQLSTATE	Error	Description
1493	HY000	ER_RANGE_NOT_INCREASING_ERROR	VALUES LESS THAN value must be strictly increasing for each partition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1494: VALUES value must be of same type as partition function

Error Code	SQLSTATE	Error	Description
1494	HY000	ER_INCONSISTENT_TYPE_OF_FUNCTIONS_ERROR	VALUES value must be of same type as partition function

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1495: Multiple definition of same constant in list partitioning

Error Code	SQLSTATE	Error	Description
1495	HY000	ER_MULTIPLE_DEF_CONST_IN_LIST_PART_ERROR	Multiple definition of same constant in list partitioning

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1496: Partitioning can not be used stand-alone in query

Error Code	SQLSTATE	Error	Description
1496	HY000	ER_PARTITION_ENTRY_ERROR	Partitioning can not be used stand-alone in query

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1497: The mix of handlers in the partitions is not allowed in this version of MariaDB

Error Code	SQLSTATE	Error	Description
1497	HY000	ER_MIX_HANDLER_ERROR	The mix of handlers in the partitions is not allowed in this version of MariaDB

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1498: For the partitioned engine it is necessary to define all

Error Code	SQLSTATE	Error	Description
1498	HY000	ER_PARTITION_NOT_DEFINED_ERROR	For the partitioned engine it is necessary to define all %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1499: Too many partitions (including subpartitions) were defined

Error Code	SQLSTATE	Error	Description
1499	HY000	ER_TOO_MANY_PARTITIONS_ERROR	Too many partitions (including subpartitions) were defined

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1500 to 1599

- Error 1500: It is only possible to mix RANGE/LIST partitioning with HASH/KEY partitioning for subpar
- Error 1501: Failed to create specific handler file
- Error 1502: A BLOB field is not allowed in partition function
- Error 1503: must include all columns in the table's partitioning function
- Error 1504: Number of = 0 is not an allowed value
- Error 1505: Partition management on a not partitioned table is not possible
- Error 1506: Foreign key clause is not yet supported in conjunction with partitioning
- Error 1507: Error in list of partitions to
- Error 1508: Cannot remove all partitions, use DROP TABLE instead
- Error 1509: COALESCE PARTITION can only be used on HASH/KEY partitions
- Error 1510: REORGANIZE PARTITION can only be used to reorganize partitions not to change their numbe
- Error 1511: REORGANIZE PARTITION without parameters can only be used on auto-partitioned tables
- Error 1512: PARTITION can only be used on RANGE/LIST partitions
- Error 1513: Trying to Add partition(s) with wrong number of subpartitions
- Error 1514: At least one partition must be added
- Error 1515: At least one partition must be coalesced
- Error 1516: More partitions to reorganize than there are partitions
- Error 1517: Duplicate partition name
- Error 1518: It is not allowed to shut off binlog on this command
- Error 1519: When reorganizing a set of partitions they must be in consecutive order
- Error 1520: Reorganize of range partitions cannot change total ranges except for last partition
- Error 1521: Partition function not supported in this version for this handler
- Error 1522: Partition state cannot be defined from CREATE/ALTER TABLE
- Error 1523: The handler only supports 32 bit integers in VALUES
- Error 1524: Plugin is not loaded
- Error 1525: Incorrect value
- Error 1526: Table has no partition for value
- Error 1527: It is not allowed to specify more than once
- Error 1528: Failed to create

- Error 1529: Failed to drop
- Error 1530: The handler doesn't support autoextend of tablespaces
- Error 1531: A size parameter was incorrectly specified, either number or on the form 10M
- Error 1532: The size number was correct but we don't allow the digit part to be more than 2 billion
- Error 1533: Failed to alter
- Error 1534: Writing one row to the row-based binary log failed
- Error 1535: Table definition on master and slave does not match
- Error 1536: Slave running with –log-slave-updates must use row-based binary logging to be able to r
- Error 1537: Event already exists
- Error 1538: Failed to store event. Error code from storage engine.
- Error 1539: Unknown event
- Error 1540: Failed to alter event
- Error 1541: Failed to drop
- Error 1542: INTERVAL is either not positive or too big
- Error 1543: ENDS is either invalid or before STARTS
- Error 1544: Event execution time is in the past. Event has been disabled
- Error 1545: Failed to open mysql.event
- Error 1546: No datetime expression provided
- Error 1547: Column count is wrong. Expected found. The table is probably corrupted
- Error 1548: Cannot load from mysql. The table is probably corrupted
- Error 1549: Failed to delete the event from mysql.event
- Error 1550: Error during compilation of event's body
- Error 1551: Same old and new event name
- Error 1552: Data for column too long
- Error 1553: Cannot drop index needed in a foreign key constraint
- Error 1554: The syntax is deprecated and will be removed in MariaDB
- Error 1555: You can't write-lock a log table. Only read access is possible
- Error 1556: You can't use locks with log tables
- Error 1557: Upholding foreign key constraints for table, entry, key would lead to a duplicate entry
- Error 1558: Column count is wrong. Expected found. Created with MariaDB now running. Please use mysq
- Error 1559: Cannot switch out of the row-based binary log format when the session has open temporary
- Error 1560: Cannot change the binary logging format inside a stored function or trigger
- Error 1561: You should never see it
- Error 1562: Cannot create temporary table with partitions
- Error 1563: Partition constant is out of partition function domain
- Error 1564: This partition function is not allowed
- Error 1565: Error in DDL log
- Error 1566: Not allowed to use NULL value in VALUES LESS THAN
- Error 1567: Incorrect partition name
- Error 1568: Transaction isolation level can't be changed while a transaction is in progress
- Error 1569: ALTER TABLE causes auto_increment resequencing, resulting in duplicate entry for key
- Error 1570: Internal scheduler error
- Error 1571: Error during starting/stopping of the scheduler
- Error 1572: Engine cannot be used in partitioned tables
- Error 1573: Cannot activate log

- Error 1574: The server was not built with row-based replication
- Error 1575: Decoding of base64 string failed
- Error 1576: Recursion of EVENT DDL statements is forbidden when body is present
- Error 1577: Cannot proceed because system tables used by Event Scheduler were found damaged at serve
- Error 1578: Only integers allowed as number here
- Error 1579: This storage engine cannot be used for log tables
- Error 1580: You cannot '%s' a log table if logging is enabled
- Error 1581: Cannot rename. When logging enabled, rename to/from log table must rename two tables
- Error 1582: Incorrect parameter count in the call to native function
- Error 1583: Incorrect parameters in the call to native function
- Error 1584: Incorrect parameters in the call to stored function
- Error 1585: This function has the same name as a native function
- Error 1586: Duplicate entry for key
- Error 1587: Too many files opened, please execute the command again
- Error 1588: Event execution time is in the past and ON COMPLETION NOT PRESERVE is set
- Error 1589: Event execution time is in the past and ON COMPLETION NOT PRESERVE is set
- Error 1590: The incident occurred on the master
- Error 1591: Table has no partition for some existing values
- Error 1592: Unsafe statement written to the binary log using statement format since BINLOG_FORMAT =
- Error 1593: Fatal error
- Error 1594: Relay log read failure
- Error 1595: Relay log write failure
- Error 1596: Failed to create
- Error 1597: Master command failed
- Error 1598: Binary logging not possible
- Error 1599: View has no creation context

This page is licensed: CC BY-SA / Gnu FDL

Error 1500: It is only possible to mix RANGE/LIST partitioning with HASH/KEY partitioning for subpar

Error 1500: It is only possible to mix RANGE/LIST partitioning with HASH/KEY partitioning for subpartitioning

Error Code	SQLSTATE	Error	Description
1500	HY000	ER_SUBPARTITION_ERROR	It is only possible to mix RANGE/LIST partitioning with HASH/KEY partitioning for subpartitioning

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1501: Failed to create specific handler file

Error Code	SQLSTATE	Error	Description
1501	HY000	ER_CANT_CREATE_HANDLER_FILE	Failed to create specific handler file

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1502: A BLOB field is not allowed in partition function

Error Code	SQLSTATE	Error	Description
1502	HY000	ER_BLOB_FIELD_IN_PART_FUNC_ERROR	A BLOB field is not allowed in partition function

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1503: must include all columns in the table's partitioning function

Error Code	SQLSTATE	Error	Description
1503	HY000	ER_UNIQUE_KEY_NEED_ALL_FIELDS_IN_PF	A %s must include all columns in the table's partitioning function

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1504: Number of = 0 is not an allowed value

Error Code	SQLSTATE	Error	Description
1504	HY000	ER_NO_PARTS_ERROR	Number of %s = 0 is not an allowed value

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1505: Partition management on a not partitioned table is not possible

Error Code	SQLSTATE	Error	Description
1505	HY000	ER_PARTITION_MGMT_ON_NONPARTITIONED	Partition management on a not partitioned table is not possible

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1506: Foreign key clause is not yet supported in conjunction with partitioning

Error Code	SQLSTATE	Error	Description
1506	HY000	ER_FOREIGN_KEY_ON_PARTITIONED	Foreign key clause is not yet supported in conjunction with partitioning

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1507: Error in list of partitions to

Error Code	SQLSTATE	Error	Description
1507	HY000	ER_DROP_PARTITION_NON_EXISTENT	Error in list of partitions to %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1508: Cannot remove all partitions, use DROP TABLE instead

Error Code	SQLSTATE	Error	Description
1508	HY000	ER_DROP_LAST_PARTITION	Cannot remove all partitions, use DROP TABLE instead

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1509: COALESCE PARTITION can only be used on HASH/KEY partitions

Error Code	SQLSTATE	Error	Description
1509	HY000	ER_COALESCE_ONLY_ON_HASH_PARTITION	COALESCE PARTITION can only be used on HASH/KEY partitions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1510: REORGANIZE PARTITION can only be used to reorganize partitions not to change their numbe

Error 1510: REORGANIZE PARTITION can only be used to reorganize partitions not to change their numbers

Error Code	SQLSTATE	Error	Description
1510	HY000	ER_REORG_HASH_ONLY_ON_SAME_N	REORGANIZE PARTITION can only be used to reorganize partitions not to change their numbers

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1511: REORGANIZE PARTITION without parameters can only be used on auto-partitioned tables

Error 1511: REORGANIZE PARTITION without parameters can only be used on auto-partitioned tables using

Error Code	SQLSTATE	Error	Description
1511	HY000	ER_REORG_NO_PARAM_ERROR	REORGANIZE PARTITION without parameters can only be used on auto-partitioned tables using HASH PARTITIONs

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1512: PARTITION can only be used on RANGE/LIST partitions

Error Code	SQLSTATE	Error	Description
1512	HY000	ER_ONLY_ON_RANGE_LIST_PARTITION	%s PARTITION can only be used on RANGE/LIST partitions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1513: Trying to Add partition(s) with wrong number of subpartitions

Error Code	SQLSTATE	Error	Description
1513	HY000	ER_ADD_PARTITION_SUBPART_ERROR	Trying to Add partition(s) with wrong number of subpartitions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1514: At least one partition must be added

Error Code	SQLSTATE	Error	Description
1514	HY000	ER_ADD_PARTITION_NO_NEW_PARTITION	At least one partition must be added

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1515: At least one partition must be coalesced

Error Code	SQLSTATE	Error	Description
1515	HY000	ER_COALESCE_PARTITION_NO_PARTITION	At least one partition must be coalesced

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1516: More partitions to reorganize than there are partitions

Error Code	SQLSTATE	Error	Description
1516	HY000	ER_REORG_PARTITION_NOT_EXIST	More partitions to reorganize than there are partitions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1517: Duplicate partition name

Error Code	SQLSTATE	Error	Description
1517	HY000	ER_SAME_NAME_PARTITION	Duplicate partition name %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1518: It is not allowed to shut off binlog on this command

Error Code	SQLSTATE	Error	Description
1518	HY000	ER_NO_BINLOG_ERROR	It is not allowed to shut off binlog on this command

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1519: When reorganizing a set of partitions they must be in consecutive order

Error Code	SQLSTATE	Error	Description
1519	HY000	ER_CONSECUTIVE_REORG_PARTITIONS	When reorganizing a set of partitions they must be in consecutive order

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1520: Reorganize of range partitions cannot change total ranges except for last partition

Error 1520: Reorganize of range partitions cannot change total ranges except for last partition where

Error Code	SQLSTATE	Error	Description
1520	HY000	ER_REORG_OUTSIDE_RANGE	Reorganize of range partitions cannot change total ranges except for last partition where it can extend the range

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1521: Partition function not supported in this version for this handler

Error Code	SQLSTATE	Error	Description
1521	HY000	ER_PARTITION_FUNCTION_FAILURE	Partition function not supported in this version for this handler

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1522: Partition state cannot be defined from CREATE/ALTER TABLE

Error Code	SQLSTATE	Error	Description
1522	HY000	ER_PART_STATE_ERROR	Partition state cannot be defined from CREATE/ALTER TABLE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1523: The handler only supports 32 bit integers in VALUES

Error Code	SQLSTATE	Error	Description
1523	HY000	ER_LIMITED_PART_RANGE	The %s handler only supports 32 bit integers in VALUES

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1524: Plugin is not loaded

Error Code	SQLSTATE	Error	Description
1524	HY000	ER_PLUGIN_IS_NOT_LOADED	Plugin ‘%s’ is not loaded

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1525: Incorrect value

Error Code	SQLSTATE	Error	Description
1525	HY000	ER_WRONG_VALUE	Incorrect %s value: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1526: Table has no partition for value

Error Code	SQLSTATE	Error	Description
1526	HY000	ER_NO_PARTITION_FOR_GIVEN_VALUE	Table has no partition for value %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1527: It is not allowed to specify more than once

Error Code	SQLSTATE	Error	Description
1527	HY000	ER_FILEGROUP_OPTION_ONLY_ONCE	It is not allowed to specify %s more than once

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1528: Failed to create

Error Code	SQLSTATE	Error	Description
1528	HY000	ER_CREATE_FILEGROUP_FAILED	Failed to create %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1529: Failed to drop

Error Code	SQLSTATE	Error	Description
1529	HY000	ER_DROP_FILEGROUP_FAILED	Failed to drop %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1530: The handler doesn't support autoextend of tablespaces

Error Code	SQLSTATE	Error	Description
1530	HY000	ER_TABLESPACE_AUTO_EXTEND_ERROR	The handler doesn't support autoextend of tablespaces

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1531: A size parameter was incorrectly specified, either number or on the form 10M

Error Code	SQLSTATE	Error	Description
1531	HY000	ER_WRONG_SIZE_NUMBER	A size parameter was incorrectly specified, either number or on the form 10M

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1532: The size number was correct but we don't allow the digit part to be more than 2 billion

Error Code	SQLSTATE	Error	Description
1532	HY000	ER_SIZE_OVERFLOW_ERROR	The size number was correct but we don't allow the digit part to be more than 2 billion

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1533: Failed to alter

Error Code	SQLSTATE	Error	Description
1533	HY000	ER_ALTER_FILEGROUP_FAILED	Failed to alter: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1534: Writing one row to the row-based binary log failed

Error Code	SQLSTATE	Error	Description
1534	HY000	ER_BINLOG_ROW_LOGGING_FAILED	Writing one row to the row-based binary log failed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1535: Table definition on master and slave does not match

Error Code	SQLSTATE	Error	Description
1535	HY000	ER_BINLOG_ROW_WRONG_TABLE_DEF	Table definition on master and slave does not match: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1536: Slave running with –log-slave-updates must use row-based binary logging to be able to r

Error 1536: Slave running with –log-slave-updates must use row-based binary logging to be able to replicate row-based binary log events

Error Code	SQLSTATE	Error	Description
1536	HY000	ER_BINLOG_ROW_RBR_TO_SBR	Slave running with –log-slave-updates must use row-based binary logging to be able to replicate row-based binary log events

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1537: Event already exists

Error Code	SQLSTATE	Error	Description
1537	HY000	ER_EVENT_ALREADY_EXISTS	Event ‘%s’ already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1538: Failed to store event. Error code from storage engine.

Error Code	SQLSTATE	Error	Description
1538	HY000	ER_EVENT_STORE_FAILED	Failed to store event %s. Error code %d from storage engine.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1539: Unknown event

Error Code	SQLSTATE	Error	Description
1539	HY000	ER_EVENT_DOES_NOT_EXIST	Unknown event ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1540: Failed to alter event

Error Code	SQLSTATE	Error	Description
1540	HY000	ER_EVENT_CANT_ALTER	Failed to alter event ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1541: Failed to drop

Error Code	SQLSTATE	Error	Description
1541	HY000	ER_EVENT_DROP_FAILED	Failed to drop %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1542: INTERVAL is either not positive or too big

Error Code	SQLSTATE	Error	Description
1542	HY000	ER_EVENT_INTERVAL_NOT_POSITIVE_OR_TOO_BIG	INTERVAL is either not positive or too big

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1543: ENDS is either invalid or before STARTS

Error Code	SQLSTATE	Error	Description
1543	HY000	ER_EVENT_ENDS_BEFORE_STARTS	ENDS is either invalid or before STARTS

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1544: Event execution time is in the past. Event has been disabled

Error Code	SQLSTATE	Error	Description
1544	HY000	ER_EVENT_EXEC_TIME_IN_THE_PAST	Event execution time is in the past. Event has been disabled

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1545: Failed to open mysql.event

Error Code	SQLSTATE	Error	Description
1545	HY000	ER_EVENT_OPEN_TABLE_FAILED	Failed to open mysql.event

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1546: No datetime expression provided

Error Code	SQLSTATE	Error	Description
1546	HY000	ER_EVENT_NEITHER_M_EXPR_NOR_M_AT	No datetime expression provided

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1547: Column count is wrong. Expected found. The table is probably corrupted

Error 1547: You should never see it

Error Code	SQLSTATE	Error	Description
1547	HY000	ER_UNUSED_2	You should never see it

Possible Causes and Solutions

Error 1548: Cannot load from mysql. The table is probably corrupted

Error 1548: You should never see it

Error Code	SQLSTATE	Error	Description
1548	HY000	ER_UNUSED_3	You should never see it

Possible Causes and Solutions

Error 1549: Failed to delete the event from mysql.event

Error Code	SQLSTATE	Error	Description
1549	HY000	ER_EVENT_CANNOT_DELETE	Failed to delete the event from mysql.event

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1550: Error during compilation of event’s body

Error Code	SQLSTATE	Error	Description
1550	HY000	ER_EVENT_COMPILE_ERROR	Error during compilation of event’s body

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1551: Same old and new event name

Error Code	SQLSTATE	Error	Description
1551	HY000	ER_EVENT_SAME_NAME	Same old and new event name

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1552: Data for column too long

Error Code	SQLSTATE	Error	Description
1552	HY000	ER_EVENT_DATA_TOO_LONG	Data for column ‘%s’ too long

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1553: Cannot drop index needed in a foreign key constraint

Error Code	SQLSTATE	Error	Description
1553	HY000	ER_DROP_INDEX_FK	Cannot drop index ‘%s’: needed in a foreign key constraint

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1554: The syntax is deprecated and will be removed in MariaDB

Error Code	SQLSTATE	Error	Description
1554	HY000	ER_WARN_DEPRECATED_SYNTAX_WITH_VER	The syntax ‘%s’ is deprecated and will be removed in MariaDB %s. Please use %s instead

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1555: You can’t write-lock a log table. Only read access is possible

Error Code	SQLSTATE	Error	Description
1555	HY000	ER_CANT_WRITE_LOCK_LOG_TABLE	You can’t write-lock a log table. Only read access is possible

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1556: You can’t use locks with log tables

Error Code	SQLSTATE	Error	Description
1556	HY000	ER_CANT_LOCK_LOG_TABLE	You can’t use locks with log tables.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1557: Upholding foreign key constraints for table, entry, key would lead to a duplicate entry

Error 1557: You should never see it

Error Code	SQLSTATE	Error	Description
1557	HY000	ER_UNUSED_4	You should never see it

Possible Causes and Solutions

Error 1558: Column count is wrong. Expected found. Created with MariaDB now running. Please use mysq

Error Code	SQLSTATE	Error	Description
1558	HY000	ER_COL_COUNT_DOESNT_MATCH_PLEASE_UPDATE	Column count of mysql.%s is wrong. Expected %d, found %d. Created with MariaDB %d, now running %d. Please use mysql_upgrade to fix this error.

Possible Causes and Solutions

Not upgrading properly

This error likely means an upgrade hasn't been performed properly, as one of the system tables does not match the expected size for the running release. mariadb-upgrade usually takes care of this. See Upgrading Between Major MariaDB Versions.

This page is licensed: CC BY-SA / Gnu FDL

Error 1559: Cannot switch out of the row-based binary log format when the session has open temporary

Error Code	SQLSTATE	Error	Description
1559	HY000	ER_TEMP_TABLE_PREVENTS_SWITCH_OUT_OF_RBR	Cannot switch out of the row-based binary log format when the session has open temporary tables

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1560: Cannot change the binary logging format inside a stored function or trigger

Error Code	SQLSTATE	Error	Description
1560	HY000	ER_STORED_FUNCTION_PREVENTS_SWITCH_BINLOG_FORMAT	Cannot change the binary logging format inside a stored function or trigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1561: You should never see it

Error Code	SQLSTATE	Error	Description
1561	HY000	ER_UNUSED_13	You should never see it

Possible Causes and Solutions

Error 1562: Cannot create temporary table with partitions

Error Code	SQLSTATE	Error	Description
1562	HY000	ER_PARTITION_NO_TEMPORARY	Cannot create temporary table with partitions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1563: Partition constant is out of partition function domain

Error Code	SQLSTATE	Error	Description
1563	HY000	ER_PARTITION_CONST_DOMAIN_ERROR	Partition constant is out of partition function domain

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1564: This partition function is not allowed

Error Code	SQLSTATE	Error	Description
1564	HY000	ER_PARTITION_FUNCTION_IS_NOT_ALLOWED	This partition function is not allowed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1565: Error in DDL log

Error Code	SQLSTATE	Error	Description
1565	HY000	ER_DDL_LOG_ERROR	Error in DDL log

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1566: Not allowed to use NULL value in VALUES LESS THAN

Error Code	SQLSTATE	Error	Description
1566	HY000	ER_NULL_IN_VALUES_LESS_THAN	Not allowed to use NULL value in VALUES LESS THAN

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1567: Incorrect partition name

Error Code	SQLSTATE	Error	Description
1567	HY000	ER_WRONG_PARTITION_NAME	Incorrect partition name

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1568: Transaction isolation level can’t be changed while a transaction is in progress

Error Code	SQLSTATE	Error	Description
1568	25001	ER_CANT_CHANGE_TX_ISOLATION	Transaction isolation level can’t be changed while a transaction is in progress

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1569: ALTER TABLE causes auto_increment resequencing, resulting in duplicate entry for key

Error Code	SQLSTATE	Error	Description
1569	HY000	ER_DUP_ENTRY_AUTOINCREMENT_CASE	ALTER TABLE causes auto_increment resequencing, resulting in duplicate entry '%s' for key '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1570: Internal scheduler error

Error Code	SQLSTATE	Error	Description
1570	HY000	ER_EVENT_MODIFY_QUEUE_ERROR	Internal scheduler error %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1571: Error during starting/stopping of the scheduler

Error Code	SQLSTATE	Error	Description
1571	HY000	ER_EVENT_SET_VAR_ERROR	Error during starting/stopping of the scheduler. Error code %u

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1572: Engine cannot be used in partitioned tables

Error Code	SQLSTATE	Error	Description
1572	HY000	ER_PARTITION_MERGE_ERROR	Engine cannot be used in partitioned tables

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1573: Cannot activate log

Error Code	SQLSTATE	Error	Description
1573	HY000	ER_CANT_ACTIVATE_LOG	Cannot activate '%s' log

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1574: The server was not built with row-based replication

Error Code	SQLSTATE	Error	Description
1574	HY000	ER_RBR_NOT_AVAILABLE	The server was not built with row-based replication

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1575: Decoding of base64 string failed

Error Code	SQLSTATE	Error	Description
1575	HY000	ER_BASE64_DECODE_ERROR	Decoding of base64 string failed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1576: Recursion of EVENT DDL statements is forbidden when body is present

Error Code	SQLSTATE	Error	Description
1576	HY000	ER_EVENT_RECURSION_FORBIDDEN	Recursion of EVENT DDL statements is forbidden when body is present

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1577: Cannot proceed because system tables used by Event Scheduler were found damaged at serve

Error 1577: Cannot proceed because system tables used by Event Scheduler were found damaged at server start

Error Code	SQLSTATE	Error	Description
1577	HY000	ER_EVENTS_DB_ERROR	Cannot proceed because system tables used by Event Scheduler were found damaged at server start

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1578: Only integers allowed as number here

Error Code	SQLSTATE	Error	Description
1578	HY000	ER_ONLY_INTEGERS_ALLOWED	Only integers allowed as number here

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1579: This storage engine cannot be used for log tables

Error Code	SQLSTATE	Error	Description
1579	HY000	ER_UNSUPORTED_LOG_ENGINE	This storage engine cannot be used for log tables

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1580: You cannot ‘%s’ a log table if logging is enabled

Error Code	SQLSTATE	Error	Description
1580	HY000	ER_BAD_LOG_STATEMENT	You cannot ‘%s’ a log table if logging is enabled

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1581: Cannot rename. When logging enabled, rename to/from log table must rename two tables

Error Code	SQLSTATE	Error	Description
1581	HY000	ER_CANT_RENAME_LOG_TABLE	Cannot rename ‘%s’. When logging enabled, rename to/from log table must rename two tables: the log table to an archive table and another table back to ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1582: Incorrect parameter count in the call to native function

Error Code	SQLSTATE	Error	Description
1582	42000	ER_WRONG_PARAMCOUNT_TO_NATIVE_FCT	Incorrect parameter count in the call to native function ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1583: Incorrect parameters in the call to native function

Error Code	SQLSTATE	Error	Description
1583	42000	ER_WRONG_PARAMETERS_TO_NATIVE_FCT	Incorrect parameters in the call to native function ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1584: Incorrect parameters in the call to stored function

Error Code	SQLSTATE	Error	Description
1584	42000	ER_WRONG_PARAMETERS_TO_STORED_FCT	Incorrect parameters in the call to stored function ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1585: This function has the same name as a native function

Error Code	SQLSTATE	Error	Description
1585	HY000	ER_NATIVE_FCT_NAME_COLLISION	This function ‘%s’ has the same name as a native function

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1586: Duplicate entry for key

Error Code	SQLSTATE	Error	Description
1586	23000	ER_DUP_ENTRY_WITH_KEY_NAME	Duplicate entry ‘%s’ for key ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1587: Too many files opened, please execute the command again

Error Code	SQLSTATE	Error	Description
1587	HY000	ER_BINLOG_PURGE_EMFILE	Too many files opened, please execute the command again

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1588: Event execution time is in the past and ON COMPLETION NOT PRESERVE is set

Error Code	SQLSTATE	Error	Description
1588	HY000	ER_EVENT_CANNOT_CREATE_IN_THE_PAST	Event execution time is in the past and ON COMPLETION NOT PRESERVE is set. The event was dropped immediately after creation.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1589: Event execution time is in the past and ON COMPLETION NOT PRESERVE is set

Error Code	SQLSTATE	Error	Description
1589	HY000	ER_EVENT_CANNOT_ALTER_IN_THE_PAST	Event execution time is in the past and ON COMPLETION NOT PRESERVE is set. The event was dropped immediately after creation.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1590: The incident occurred on the master

Error Code	SQLSTATE	Error	Description
1590	HY000	ER_SLAVE_INCIDENT	The incident %s occurred on the master. Message: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1591: Table has no partition for some existing values

Error Code	SQLSTATE	Error	Description
1591	HY000	ER_NO_PARTITION_FOR_GIVEN_VALUE_SILENT	Table has no partition for some existing values

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1592: Unsafe statement written to the binary log using statement format since BINLOG_FORMAT =

Error Code	SQLSTATE	Error	Description
1592	HY000	ER_BINLOG_UNSAFE_STATEMENT	Unsafe statement written to the binary log using statement format since BINLOG_FORMAT = STATEMENT. %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1593: Fatal error

Error Code	SQLSTATE	Error	Description
1593	HY000	ER_SLAVE_FATAL_ERROR	Fatal error: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1594: Relay log read failure

Error Code	SQLSTATE	Error	Description
1594	HY000	ER_SLAVE_RELAY_LOG_READ_FAILURE	Relay log read failure: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1595: Relay log write failure

Error Code	SQLSTATE	Error	Description
1595	HY000	ER_SLAVE_RELAY_LOG_WRITE_FAILURE	Relay log write failure: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1596: Failed to create

Error Code	SQLSTATE	Error	Description
1596	HY000	ER_SLAVE_CREATE_EVENT_FAILURE	Failed to create %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1597: Master command failed

Error Code	SQLSTATE	Error	Description
1597	HY000	ER_SLAVE_MASTER_COM_FAILURE	Master command %s failed: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1598: Binary logging not possible

Error Code	SQLSTATE	Error	Description
1598	HY000	ER_BINLOG_LOGGING_IMPOSSIBLE	Binary logging not possible. Message: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1599: View has no creation context

Error Code	SQLSTATE	Error	Description
1599	HY000	ER_VIEW_NO_CREATION_CTX	View `%s`.`%s` has no creation context

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1600 to 1699

- Error 1600: Creation context of view is invalid
- Error 1601: Creation context of stored routine is invalid
- Error 1602: Corrupted TRG file for table
- Error 1603: Triggers for table have no creation context
- Error 1604: Trigger creation context of table is invalid
- Error 1605: Creation context of event is invalid
- Error 1606: Cannot open table for trigger
- Error 1607: Cannot create stored routine. Check warnings
- Error 1608: You should never see it
- Error 1609: The BINLOG statement of type was not preceded by a format description BINLOG statement.
- Error 1610: Corrupted replication event was detected
- Error 1611: Invalid column reference in LOAD DATA
- Error 1612: Being purged log was not found
- Error 1613: XA_RBTIMEOUT: Transaction branch was rolled back: took too long
- Error 1614: XA_RBDEADLOCK: Transaction branch was rolled back: deadlock was detected
- Error 1615: Prepared statement needs to be re-prepared
- Error 1616: DELAYED option not supported for table
- Error 1617: The master info structure does not exist
- Error 1618: option ignored
- Error 1619: Built-in plugins cannot be deleted
- Error 1620: Plugin is busy and will be uninstalled on shutdown
- Error 1621: variable is read-only. Use SET to assign the value
- Error 1622: Storage engine does not support rollback for this statement
- Error 1623: Unexpected master’s heartbeat data
- Error 1624: The requested value for the heartbeat period is either negative or exceeds the maximum a
- Error 1625: You should never see it

- Error 1626: Error in parsing conflict function
- Error 1627: Write to exceptions table failed
- Error 1628: Comment for table is too long
- Error 1629: Comment for field is too long
- Error 1630: FUNCTION does not exist. Check the ‘Function Name Parsing and Resolution’ section in the
- Error 1631: Database
- Error 1632: Table
- Error 1633: Partition
- Error 1634: Subpartition
- Error 1635: Temporary
- Error 1636: Renamed
- Error 1637: Too many active concurrent transactions
- Error 1638: Non-ASCII separator arguments are not fully supported
- Error 1639: debug sync point wait timed out
- Error 1640: debug sync point hit limit reached
- Error 1641: Duplicate condition information item
- Error 1642: Unhandled user-defined warning condition
- Error 1643: Unhandled user-defined not found condition
- Error 1644: Unhandled user-defined exception condition
- Error 1645: RESIGNAL when handler not active
- Error 1646: SIGNAL/RESIGNAL can only use a CONDITION defined with SQLSTATE
- Error 1647: Data truncated for condition item
- Error 1648: Data too long for condition item
- Error 1649: Unknown locale
- Error 1650: The requested server id clashes with the slave startup option –replicate-same-server-id
- Error 1651: Query cache is disabled; restart the server with query_cache_type=1 to enable it
- Error 1652: Duplicate partition field name
- Error 1653: Inconsistency in usage of column lists for partitioning
- Error 1654: Partition column values of incorrect type
- Error 1655: Too many fields
- Error 1656: Cannot use MAXVALUE as value in VALUES IN
- Error 1657: Cannot have more than one value for this type of partitioning
- Error 1658: Row expressions in VALUES IN only allowed for multi-field column partitioning
- Error 1659: Field is of a not allowed type for this type of partitioning
- Error 1660: The total length of the partitioning fields is too large
- Error 1661: Cannot execute statement: impossible to write to binary log since both row-incapable eng
- Error 1662: Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = ROW an
- Error 1663: Cannot execute statement: impossible to write to binary log since statement is unsafe, s
- Error 1664: Cannot execute statement: impossible to write to binary log since statement is in row format
- Error 1665: Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEM
- Error 1666: Cannot execute statement: impossible to write to binary log since statement is in row format
- Error 1667: Cannot execute statement: impossible to write to binary log since more than one engine i
- Error 1668: The statement is unsafe because it uses a LIMIT clause
- Error 1669: The statement is unsafe because it uses INSERT DELAYED. This is unsafe because the times
- Error 1670: The statement is unsafe because it uses the general log, slow query log, or performance_

- Error 1671: Statement is unsafe because it invokes a trigger or a stored function that inserts into
- Error 1672: Statement is unsafe because it uses a UDF which may not return the same value on the sla
- Error 1673: Statement is unsafe because it uses a system variable that may have a different value on
- Error 1674: Statement is unsafe because it uses a system function that may return a different value
- Error 1675: Statement is unsafe because it accesses a non-transactional table after accessing a tran
- Error 1676: Statement
- Error 1677: Column of table cannot be converted from type to type
- Error 1678: Can't create conversion table for table
- Error 1679: Cannot modify @@session.binlog_format inside a transaction
- Error 1680: The path specified is too long
- Error 1681: is deprecated and will be removed in a future release.
- Error 1682: Native table has the wrong structure
- Error 1683: Invalid performance_schema usage
- Error 1684: Table was skipped since its definition is being modified by concurrent DDL statement
- Error 1685: Cannot modify @@session.binlog_direct_non_transactional_updates inside a transaction
- Error 1686: Cannot change the binlog direct flag inside a stored function or trigger
- Error 1687: A SPATIAL index may only contain a geometrical type column
- Error 1688: Comment for index is too long
- Error 1689: Wait on a lock was aborted due to a pending exclusive lock
- Error 1690: value is out of range in
- Error 1691: A variable of a non-integer based type in LIMIT clause
- Error 1692: Mixing self-logging and non-self-logging engines in a statement is unsafe.
- Error 1693: Statement accesses nontransactional table as well as transactional or temporary table, a
- Error 1694: Cannot modify @@session.sql_log_bin inside a transaction
- Error 1695: Cannot change the sql_log_bin inside a stored function or trigger
- Error 1696: Failed to read from the .par file
- Error 1697: VALUES value for partition must have type INT
- Error 1698: Access denied for user
- Error 1699: SET PASSWORD has no significance for users authenticating via plugins

This page is licensed: CC BY-SA / Gnu FDL

Error 1600: Creation context of view is invalid

Error Code	SQLSTATE	Error	Description
1600	HY000	ER_VIEW_INVALID_CREATION_CTX	Creation context of view `%s`.`%s` is invalid

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1601: Creation context of stored routine is invalid

Error Code	SQLSTATE	Error	Description
1601	HY000	ER_SR_INVALID_CREATION_CTX	Creation context of stored routine `%s`.`%s` is invalid

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1602: Corrupted TRG file for table

Error Code	SQLSTATE	Error	Description
1602	HY000	ER_TRG_CORRUPTED_FILE	Corrupted TRG file for table `%s`.`%s`

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1603: Triggers for table have no creation context

Error Code	SQLSTATE	Error	Description
1603	HY000	ER_TRG_NO_CREATION_CTX	Triggers for table `%s`.`%s` have no creation context

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1604: Trigger creation context of table is invalid

Error Code	SQLSTATE	Error	Description
1604	HY000	ER_TRG_INVALID_CREATION_CTX	Trigger creation context of table `%s`.`%s` is invalid

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1605: Creation context of event is invalid

Error Code	SQLSTATE	Error	Description
1605	HY000	ER_EVENT_INVALID_CREATION_CTX	Creation context of event `%s`.`%s` is invalid

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1606: Cannot open table for trigger

Error Code	SQLSTATE	Error	Description
1606	HY000	ER_TRG_CANT_OPEN_TABLE	Cannot open table for trigger `%s`.`%s`

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1607: Cannot create stored routine. Check warnings

Error Code	SQLSTATE	Error	Description
1607	HY000	ER_CANT_CREATE_SROUTINE	Cannot create stored routine `%s`. Check warnings

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1608: You should never see it

Error Code	SQLSTATE	Error	Description
1608	HY000	ER_UNUSED_11	You should never see it

Possible Causes and Solutions

Error 1609: The BINLOG statement of type was not preceded by a format description BINLOG statement.

Error Code	SQLSTATE	Error	Description
1609	HY000	ER_NO_FORMAT_DESCRIPTION_EVENT_BEFORE_BINLOG_STATEMENT	The BINLOG statement of type `%s` was not preceded by a format description BINLOG statement.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1610: Corrupted replication event was detected

Error Code	SQLSTATE	Error	Description
1610	HY000	ER_SLAVE_CORRUPT_EVENT	Corrupted replication event was detected

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1611: Invalid column reference in LOAD DATA

Error Code	SQLSTATE	Error	Description
1611	HY000	ER_LOAD_DATA_INVALID_COLUMN	Invalid column reference (%s) in LOAD DATA

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1612: Being purged log was not found

Error Code	SQLSTATE	Error	Description
1612	HY000	ER_LOG_PURGE_NO_FILE	Being purged log %s was not found

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1613: XA_RBTIMEOUT: Transaction branch was rolled back: took too long

Error Code	SQLSTATE	Error	Description
1613	XA106	ER_XA_RBTIMEOUT	XA_RBTIMEOUT: Transaction branch was rolled back: took too long

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1614: XA_RBDEADLOCK: Transaction branch was rolled back: deadlock was detected

Error Code	SQLSTATE	Error	Description
1614	XA102	ER_XA_RBDEADLOCK	XA_RBDEADLOCK: Transaction branch was rolled back: deadlock was detected

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1615: Prepared statement needs to be re-prepared

Error Code	SQLSTATE	Error	Description
1615	HY000	ER_NEED_REPREPARE	Prepared statement needs to be re-prepared

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1616: DELAYED option not supported for table

Error Code	SQLSTATE	Error	Description
1616	HY000	ER_DELAYED_NOT_SUPPORTED	DELAYED option not supported for table '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1617: The master info structure does not exist

Error Code	SQLSTATE	Error	Description
1617	HY000	WARN_NO_MASTER_INF	The master info structure does not exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1618: option ignored

Error Code	SQLSTATE	Error	Description
1618	HY000	WARN_OPTION_IGNORED	<%s> option ignored

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1619: Built-in plugins cannot be deleted

Error Code	SQLSTATE	Error	Description
1619	HY000	WARN_PLUGIN_DELETE_BUILTIN	Built-in plugins cannot be deleted

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1620: Plugin is busy and will be uninstalled on shutdown

Error Code	SQLSTATE	Error	Description
1620	HY000	WARN_PLUGIN_BUSY	Plugin is busy and will be uninstalled on shutdown

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1621: variable is read-only. Use SET to assign the value

Error Code	SQLSTATE	Error	Description
1621	HY000	ER_VARIABLE_IS_READONLY	%s variable '%s' is read-only. Use SET %s to assign the value

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1622: Storage engine does not support rollback for this statement

Error Code	SQLSTATE	Error	Description
1622	HY000	ER_WARN_ENGINE_TRANSACTION_ROLLBACK	Storage engine %s does not support rollback for this statement. Transaction rolled back and must be restarted

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1623: Unexpected master's heartbeat data

Error Code	SQLSTATE	Error	Description
1623	HY000	ER_SLAVE_HEARTBEAT_FAILURE	Unexpected master's heartbeat data: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1624: The requested value for the heartbeat period is either negative or exceeds the maximum a

Error 1624: The requested value for the heartbeat period is either negative or exceeds the maximum allowed

Error Code	SQLSTATE	Error	Description
1624	HY000	ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE	The requested value for the heartbeat period is either negative or exceeds the maximum allowed (%s seconds).

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1625: You should never see it

Error Code	SQLSTATE	Error	Description
1625	HY000	ER_UNUSED_14	You should never see it

Possible Causes and Solutions

Error 1626: Error in parsing conflict function

Error Code	SQLSTATE	Error	Description
1626	HY000	ER_CONFLICT_FN_PARSE_ERROR	Error in parsing conflict function. Message: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1627: Write to exceptions table failed

Error Code	SQLSTATE	Error	Description
1627	HY000	ER_EXCEPTIONS_WRITE_ERROR	Write to exceptions table failed. Message: %s”

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1628: Comment for table is too long

Error Code	SQLSTATE	Error	Description
1628	HY000	ER_TOO_LONG_TABLE_COMMENT	Comment for table ‘%s’ is too long (max = %lu)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1629: Comment for field is too long

Error Code	SQLSTATE	Error	Description
1629	HY000	ER_TOO_LONG_FIELD_COMMENT	Comment for field ‘%s’ is too long (max = %lu)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1630: FUNCTION does not exist. Check the ‘Function Name Parsing and Resolution’ section in the

Error Code	SQLSTATE	Error	Description
1630	42000	ER_FUNC_INEXISTENT_NAME_COLLISION	FUNCTION %s does not exist. Check the ‘Function Name Parsing and Resolution’ section in the Reference Manual

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1631: Database

Error Code	SQLSTATE	Error	Description
1631	HY000	ER_DATABASE_NAME	Database

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1632: Table

Error Code	SQLSTATE	Error	Description
1632	HY000	ER_TABLE_NAME	Table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1633: Partition

Error Code	SQLSTATE	Error	Description
1633	HY000	ER_PARTITION_NAME	Partition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1634: Subpartition

Error Code	SQLSTATE	Error	Description
1634	HY000	ER_SUBPARTITION_NAME	Subpartition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1635: Temporary

Error Code	SQLSTATE	Error	Description
1635	HY000	ER_TEMPORARY_NAME	Temporary

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1636: Renamed

Error Code	SQLSTATE	Error	Description
1636	HY000	ER_RENAMED_NAME	Renamed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1637: Too many active concurrent transactions

Error Code	SQLSTATE	Error	Description
1637	HY000	ER_TOO_MANY_CONCURRENT_TRXS	Too many active concurrent transactions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1638: Non-ASCII separator arguments are not fully supported

Error Code	SQLSTATE	Error	Description
1638	HY000	WARN_NON_ASCII_SEPARATOR_NOT_IMPLEMENTED	Non-ASCII separator arguments are not fully supported

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1639: debug sync point wait timed out

Error Code	SQLSTATE	Error	Description
1639	HY000	ER_DEBUG_SYNC_TIMEOUT	debug sync point wait timed out

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1640: debug sync point hit limit reached

Error Code	SQLSTATE	Error	Description
1640	HY000	ER_DEBUG_SYNC_HIT_LIMIT	debug sync point hit limit reached

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1641: Duplicate condition information item

Error Code	SQLSTATE	Error	Description
1641	42000	ER_DUP_SIGNAL_SET	Duplicate condition information item ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1642: Unhandled user-defined warning condition

Error Code	SQLSTATE	Error	Description
1642	01000	ER_SIGNAL_WARN	Unhandled user-defined warning condition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1643: Unhandled user-defined not found condition

Error Code	SQLSTATE	Error	Description
1643	02000	ER_SIGNAL_NOT_FOUND	Unhandled user-defined not found condition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1644: Unhandled user-defined exception condition

Error Code	SQLSTATE	Error	Description
1644	HY000	ER_SIGNAL_EXCEPTION	Unhandled user-defined exception condition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1645: RESIGNAL when handler not active

Error Code	SQLSTATE	Error	Description
1645	0K000	ER_RESIGNAL_WITHOUT_ACTIVE_HANDLER	RESIGNAL when handler not active

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1646: SIGNAL/RESIGNAL can only use a CONDITION defined with SQLSTATE

Error Code	SQLSTATE	Error	Description
1646	HY000	ER_SIGNAL_BAD_CONDITION_TYPE	SIGNAL/RESIGNAL can only use a CONDITION defined with SQLSTATE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1647: Data truncated for condition item

Error Code	SQLSTATE	Error	Description
1647	HY000	WARN_COND_ITEM_TRUNCATED	Data truncated for condition item ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1648: Data too long for condition item

Error Code	SQLSTATE	Error	Description
1648	HY000	ER_COND_ITEM_TOO_LONG	Data too long for condition item ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1649: Unknown locale

Error Code	SQLSTATE	Error	Description
1649	HY000	ER_UNKNOWN_LOCALE	Unknown locale: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1650: The requested server id clashes with the slave startup option –replicate-same-server-id

Error Code	SQLSTATE	Error	Description
1650	HY000	ER_SLAVE_IGNORE_SERVER_IDS	The requested server id %d clashes with the slave startup option –replicate-same-server-id

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1651: Query cache is disabled; restart the server with query_cache_type=1 to enable it

Error Code	SQLSTATE	Error	Description
1651	HY000	ER_QUERY_CACHE_DISABLED	Query cache is disabled; restart the server with query_cache_type=1 to enable it

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1652: Duplicate partition field name

Error Code	SQLSTATE	Error	Description
1652	HY000	ER_SAME_NAME_PARTITION_FIELD	Duplicate partition field name ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1653: Inconsistency in usage of column lists for partitioning

Error Code	SQLSTATE	Error	Description
1653	HY000	ER_PARTITION_COLUMN_LIST_ERROR	Inconsistency in usage of column lists for partitioning

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1654: Partition column values of incorrect type

Error Code	SQLSTATE	Error	Description
1654	HY000	ER_WRONG_TYPE_COLUMN_VALUE_ERROR	Partition column values of incorrect type

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1655: Too many fields

Error Code	SQLSTATE	Error	Description
1655	HY000	ER_TOO_MANY_PARTITION_FUNC_FIELDS_ERROR	Too many fields in '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1656: Cannot use MAXVALUE as value in VALUES IN

Error Code	SQLSTATE	Error	Description
1656	HY000	ER_MAXVALUE_IN_VALUES_IN	Cannot use MAXVALUE as value in VALUES IN

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1657: Cannot have more than one value for this type of partitioning

Error Code	SQLSTATE	Error	Description
1657	HY000	ER_TOO_MANY_VALUES_ERROR	Cannot have more than one value for this type of %s partitioning

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1658: Row expressions in VALUES IN only allowed for multi-field column partitioning

Error Code	SQLSTATE	Error	Description
1658	HY000	ER_ROW_SINGLE_PARTITION_FIELD_ERROR	Row expressions in VALUES IN only allowed for multi-field column partitioning

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1659: Field is of a not allowed type for this type of partitioning

Error Code	SQLSTATE	Error	Description
1659	HY000	ER_FIELD_TYPE_NOT_ALLOWED_AS_PARTITION_FIELD	Field '%s' is of a not allowed type for this type of partitioning

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1660: The total length of the partitioning fields is too large

Error Code	SQLSTATE	Error	Description
1660	HY000	ER_PARTITION_FIELDS_TOO_LONG	The total length of the partitioning fields is too large

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1661: Cannot execute statement: impossible to write to binary log since both row-incapable eng

Error 1661: Cannot execute statement: impossible to write to binary log since both row-incapable engines and statement-incapable engines are involved

Error Code	SQLSTATE	Error	Description
1661	HY000	ER_BINLOG_ROW_ENGINE_AND_STMT_ENGINE	Cannot execute statement: impossible to write to binary log since both row-incapable engines and statement-incapable engines are involved.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1662: Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = ROW an

Error Code	SQLSTATE	Error	Description
1662	HY000	ER_BINLOG_ROW_MODE_AND_STMT_ENGINE	Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = ROW and at least one table uses a storage engine limited to statement-based logging.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1663: Cannot execute statement: impossible to write to binary log since statement is unsafe, s

Error 1663: Cannot execute statement: impossible to write to binary log since statement is unsafe, storage engine is limited to statement-based logging, and BINLOG_FORMAT = MIXED

Error Code	SQLSTATE	Error	Description
1663	HY000	ER_BINLOG_UNSAFE_AND_STMT_ENGINE	Cannot execute statement: impossible to write to binary log since statement is unsafe, storage engine is limited to statement-based logging, and BINLOG_FORMAT = MIXED. %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1664: Cannot execute statement: impossible to write to binary log since statement is in row format

Error Code	SQLSTATE	Error	Description
1664	HY000	ER_BINLOG_ROW_INJECTION_AND_STMT_ENGINE	Cannot execute statement: impossible to write to binary log since statement is in row format and at least one table uses a storage engine limited to statement-based logging.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1665: Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEM

Error Code	SQLSTATE	Error	Description
1665	HY000	ER_BINLOG_STMT_MODE_AND_ROW_ENGINE	Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEMENT and at least one table uses a storage engine limited to row-based logging.%s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1666: Cannot execute statement: impossible to write to binary log since statement is in row format

Error Code	SQLSTATE	Error	Description
1666	HY000	ER_BINLOG_ROW_INJECTION_AND_STMT_MODE	Cannot execute statement: impossible to write to binary log since statement is in row format and BINLOG_FORMAT = STATEMENT.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1667: Cannot execute statement: impossible to write to binary log since more than one engine i

Error 1667: Cannot execute statement: impossible to write to binary log since more than one engine is involved and at least one engine is self-logging

Error Code	SQLSTATE	Error	Description
1667	HY000	ER_BINLOG_MULTIPLE_ENGINES_AND_SELF_LOGGING_ENGINE	Cannot execute statement: impossible to write to binary log since more than one engine is involved and at least one engine is self-logging.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1668: The statement is unsafe because it uses a LIMIT clause

Error Code	SQLSTATE	Error	Description
1668	HY000	ER_BINLOG_UNSAFE_LIMIT	The statement is unsafe because it uses a LIMIT clause. This is unsafe because the set of rows included cannot be predicted.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1669: The statement is unsafe because it uses INSERT DELAYED. This is unsafe because the times

Error Code	SQLSTATE	Error	Description
1669	HY000	ER_BINLOG_UNSAFE_INSERT_DELAYED	The statement is unsafe because it uses INSERT DELAYED. This is unsafe because the times when rows are inserted cannot be predicted.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1670: The statement is unsafe because it uses the general log, slow query log, or performance_

Error Code	SQLSTATE	Error	Description
1670	HY000	ER_BINLOG_UNSAFE_SYSTEM_TABLE	The statement is unsafe because it uses the general log, slow query log, or performance_schema table(s). This is unsafe because system tables may differ on slaves.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1671: Statement is unsafe because it invokes a trigger or a stored function that inserts into

Error Code	SQLSTATE	Error	Description
1671	HY000	ER_BINLOG_UNSAFE_AUTOINC_COLUMNS	Statement is unsafe because it invokes a trigger or a stored function that inserts into an AUTO_INCREMENT column. Inserted values cannot be logged correctly.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1672: Statement is unsafe because it uses a UDF which may not return the same value on the sla

Error 1672: Statement is unsafe because it uses a UDF which may not return the same value on the slave

Error Code	SQLSTATE	Error	Description
1672	HY000	ER_BINLOG_UNSAFE_UDF	Statement is unsafe because it uses a UDF which may not return the same value on the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1673: Statement is unsafe because it uses a system variable that may have a different value on

Error Code	SQLSTATE	Error	Description
1673	HY000	ER_BINLOG_UNSAFE_SYSTEM_VARIABLE	Statement is unsafe because it uses a system variable that may have a different value on the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1674: Statement is unsafe because it uses a system function that may return a different value

Error Code	SQLSTATE	Error	Description
1674	HY000	ER_BINLOG_UNSAFE_SYSTEM_FUNCTION	Statement is unsafe because it uses a system function that may return a different value on the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1675: Statement is unsafe because it accesses a non-transactional table after accessing a tran

Error 1675: Statement is unsafe because it accesses a non-transactional table after accessing a transactional table within the same transaction

Error Code	SQLSTATE	Error	Description
1675	HY000	ER_BINLOG_UNSAFE_NONTRANS_AFTER_TRANS	Statement is unsafe because it accesses a non-transactional table after accessing a transactional table within the same transaction.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1676: Statement

Error Code	SQLSTATE	Error	Description
1676	HY000	ER_MESSAGE_AND_STATEMENT	%s Statement: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1677: Column of table cannot be converted from type to type

Error Code	SQLSTATE	Error	Description
1677	HY000	ER_SLAVE_CONVERSION_FAILED	Column %d of table ‘%s.%s’ cannot be converted from type ‘%s’ to type ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1678: Can’t create conversion table for table

Error Code	SQLSTATE	Error	Description
1678	HY000	ER_SLAVE_CANT_CREATE_CONVERSION	Can’t create conversion table for table ‘%s.%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1679: Cannot modify @@session.binlog_format inside a transaction

Error Code	SQLSTATE	Error	Description
1679	HY000	ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_BINLOG_FORMAT	Cannot modify @@session.binlog_format inside a transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1680: The path specified is too long

Error Code	SQLSTATE	Error	Description
1680	HY000	ER_PATH_LENGTH	The path specified for %s is too long.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1681: is deprecated and will be removed in a future release.

Error Code	SQLSTATE	Error	Description
1681	HY000	ER_WARN_DEPRECATED_SYNTAX_NO_REPLACEMENT	‘%s’ is deprecated and will be removed in a future release.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1682: Native table has the wrong structure

Error Code	SQLSTATE	Error	Description
1682	HY000	ER_WRONG_NATIVE_TABLE_STRUCTURE	Native table ‘%s’.’%s’ has the wrong structure

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1683: Invalid performance_schema usage

Error Code	SQLSTATE	Error	Description
1683	HY000	ER_WRONG_PERFSCHEMA_USAGE	Invalid performance_schema usage.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1684: Table was skipped since its definition is being modified by concurrent DDL statement

Error Code	SQLSTATE	Error	Description
1684	HY000	ER_WARN_I_S_SKIPPED_TABLE	Table ‘%s’.’%s’ was skipped since its definition is being modified by concurrent DDL statement

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1685: Cannot modify @@session.binlog_direct_non_transactional_updates inside a transaction

Error Code	SQLSTATE	Error	Description
1685	HY000	ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_BINLOG_DIRECT	Cannot modify @@session.binlog_direct_non_transactional_updates inside a transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1686: Cannot change the binlog direct flag inside a stored function or trigger

Error Code	SQLSTATE	Error	Description
1686	HY000	ER_STORED_FUNCTION_PREVENTS_SWITCH_BINLOG_DIRECT	Cannot change the binlog direct flag inside a stored function or trigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1687: A SPATIAL index may only contain a geometrical type column

Error Code	SQLSTATE	Error	Description
1687	42000	ER_SPATIAL_MUST_HAVE_GEOM_COL	A SPATIAL index may only contain a geometrical type column

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1688: Comment for index is too long

Error Code	SQLSTATE	Error	Description
1688	HY000	ER_TOO_LONG_INDEX_COMMENT	Comment for index ‘%s’ is too long (max = %lu)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1689: Wait on a lock was aborted due to a pending exclusive lock

Error Code	SQLSTATE	Error	Description
1689	HY000	ER_LOCK_ABORTED	Wait on a lock was aborted due to a pending exclusive lock

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1690: value is out of range in

Error Code	SQLSTATE	Error	Description
1690	22003	ER_DATA_OUT_OF_RANGE	%s value is out of range in ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1691: A variable of a non-integer based type in LIMIT clause

Error Code	SQLSTATE	Error	Description
1691	HY000	ER_WRONG_SPVAR_TYPE_IN_LIMIT	A variable of a non-integer based type in LIMIT clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1692: Mixing self-logging and non-self-logging engines in a statement is unsafe.

Error Code	SQLSTATE	Error	Description
1692	HY000	ER_BINLOG_UNSAFE_MULTIPLE_ENGINES_AND_SELF_LOGGING_ENGINE	Mixing self-logging and non-self-logging engines in a statement is unsafe.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1693: Statement accesses nontransactional table as well as transactional or temporary table, a

Error 1693: Statement accesses nontransactional table as well as transactional or temporary table, and writes to any of them

Error Code	SQLSTATE	Error	Description
1693	HY000	ER_BINLOG_UNSAFE_MIXED_STATEMENT	Statement accesses nontransactional table as well as transactional or temporary table, and writes to any of them.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1694: Cannot modify @@session.sql_log_bin inside a transaction

Error Code	SQLSTATE	Error	Description
1694	HY000	ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_SQL_LOG_BIN	Cannot modify @@session.sql_log_bin inside a transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1695: Cannot change the sql_log_bin inside a stored function or trigger

Error Code	SQLSTATE	Error	Description
1695	HY000	ER_STORED_FUNCTION_PREVENTS_SWITCH_SQL_LOG_BIN	Cannot change the sql_log_bin inside a stored function or trigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1696: Failed to read from the .par file

Error Code	SQLSTATE	Error	Description
1696	HY000	ER_FAILED_READ_FROM_PAR_FILE	Failed to read from the .par file

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1697: VALUES value for partition must have type INT

Error Code	SQLSTATE	Error	Description
1697	HY000	ER_VALUES_IS_NOT_INT_TYPE_ERROR	VALUES value for partition '%s' must have type INT

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1698: Access denied for user

Error Code	SQLSTATE	Error	Description
1698	28000	ER_ACCESS_DENIED_NO_PASSWORD_ERROR	Access denied for user '%s'@'%s'

Possible Causes and Solutions

The user/password combination may not exist.

There are a number of common problems that can occur when connecting to MariaDB. See Troubleshooting Connection Issues.

See Also

- When access_denied_errors status variable is incremented

This page is licensed: CC BY-SA / Gnu FDL

Error 1699: SET PASSWORD has no significance for users authenticating via plugins

Error Code	SQLSTATE	Error	Description
1699	HY000	ER_SET_PASSWORD_AUTH_PLUGIN	SET PASSWORD has no significance for users authenticating via plugins

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1700 to 1799

- Error 1700: GRANT with IDENTIFIED WITH is illegal because the user already exists
- Error 1701: Cannot truncate a table referenced in a foreign key constraint
- Error 1702: Plugin is force_plus_permanent and can not be unloaded
- Error 1703: The requested value for the heartbeat period is less than 1 millisecond
- Error 1704: The requested value for the heartbeat period exceeds the value of slave_net_timeout seco
- Error 1705: Multi-row statements required more than 'max_binlog_stmt_cache_size' bytes of storage; i
- Error 1706: Primary key/partition key update is not allowed since the table is updated both as and
- Error 1707: Table rebuild required. Please do "ALTER TABLE FORCE" or dump/reload to fix it!
- Error 1708: The value of should be no less than the value of
- Error 1709: Index column size too large. The maximum column size is bytes.
- Error 1710: Trigger has an error in its body

- Error 1711: Unknown trigger has an error in its body
- Error 1712: Index is corrupted
- Error 1713: Undo log record is too big
- Error 1714: INSERT IGNORE… SELECT is unsafe because the order in which rows are retrieved by the S
- Error 1715: INSERT… SELECT… ON DUPLICATE KEY UPDATE is unsafe because the order in which rows ar
- Error 1716: REPLACE… SELECT is unsafe because the order in which rows are retrieved by the SELECT
- Error 1717: CREATE… IGNORE SELECT is unsafe because the order in which rows are retrieved by the S
- Error 1718: CREATE… REPLACE SELECT is unsafe because the order in which rows are retrieved by the
- Error 1719: UPDATE IGNORE is unsafe because the order in which rows are updated determines which (if
- Error 1720: You should never see it
- Error 1721: You should never see it
- Error 1722: Statements writing to a table with an auto-increment column after selecting from another
- Error 1723: CREATE TABLE… SELECT… on a table with an auto-increment column is unsafe because the
- Error 1724: INSERT… ON DUPLICATE KEY UPDATE on a table with more than one UNIQUE KEY is unsafe
- Error 1725: Table is being used in foreign key check.
- Error 1726: Storage engine does not support system tables
- Error 1727: INSERT into autoincrement field which is not the first part in the composed primary key
- Error 1728: Cannot load from. The table is probably corrupted
- Error 1729: The requested value for the master delay exceeds the maximum
- Error 1730: Only Format_description_log_event and row events are allowed in BINLOG statements
- Error 1731: Non matching attribute between partition and table
- Error 1732: Table to exchange with partition is partitioned
- Error 1733: Table to exchange with partition is temporary
- Error 1734: Subpartitioned table, use subpartition instead of partition
- Error 1735: Unknown partition in table
- Error 1736: Tables have different definitions
- Error 1737: Found a row that does not match the partition
- Error 1738: Option binlog_cache_size is greater than max_binlog_cache_size; setting binlog_cache_siz
- Error 1739: Cannot use access on index due to type or collation conversion on field
- Error 1740: Table to exchange with partition has foreign key references
- Error 1741: Key value was not found in table
- Error 1742: Data for column too long
- Error 1743: Replication event checksum verification failed while reading from network.
- Error 1744: Replication event checksum verification failed while reading from a log file
- Error 1745: Option binlog_stmt_cache_size is greater than max_binlog_stmt_cache_size
- Error 1746: Can’t update table while is being created
- Error 1747: PARTITION () clause on non partitioned table
- Error 1748: Found a row not matching the given partition set
- Error 1749: partition doesn’t exist
- Error 1750: Failure while changing the type of replication repository
- Error 1751: The creation of some temporary tables could not be rolled back.
- Error 1752: Some temporary tables were dropped, but these operations could not be rolled back.
- Error 1753: is not supported in multi-threaded slave mode
- Error 1754: The number of modified databases exceeds the maximum
- Error 1755: Cannot execute the current event group in the parallel mode

- Error 1756: %s
- Error 1757: FULLTEXT index is not supported for partitioned tables
- Error 1758: Invalid condition number
- Error 1759: Sending passwords in plain text without SSL/TLS is extremely insecure
- Error 1760: Storing MariaDB user name or password information in the master.info repository is not s
- Error 1761: Foreign key constraint for table, record would lead to a duplicate entry in table
- Error 1762: Foreign key constraint for table, record would lead to a duplicate entry in a child table
- Error 1763: Setting authentication options is not possible when only the Slave SQL Thread is being s
- Error 1764: The table does not have FULLTEXT index to support this query
- Error 1765: The system variable cannot be set in stored functions or triggers
- Error 1766: The system variable cannot be set when there is an ongoing transaction
- Error 1767: The system variable @@SESSION.GTID_NEXT has the value which is not listed in @@SESSION.G
- Error 1768: The system variable @@SESSION.GTID_NEXT cannot change inside a transaction
- Error 1769: The statement ‘SET’ cannot invoke a stored function
- Error 1770: The system variable @@SESSION.GTID_NEXT cannot be ‘AUTOMATIC’ when @@SESSION.GTID_NEXT_L
- Error 1771: Skipping transaction because it has already been executed and logged
- Error 1772: Malformed GTID set specification
- Error 1773: Malformed GTID set encoding
- Error 1774: Malformed GTID specification
- Error 1775: Impossible to generate Global Transaction Identifier: the integer component reached the
- Error 1776: Parameters MASTER_LOG_FILE, MASTER_LOG_POS, RELAY_LOG_FILE and RELAY_LOG_POS cannot be s
- Error 1777: CHANGE MASTER TO MASTER_AUTO_POSITION = 1 can only be executed when @@GLOBAL.GTID_MODE =
- Error 1778: Cannot execute statements with implicit commit inside a transaction when @@SESSION.GTID_
- Error 1779: GTID_MODE = ON or GTID_MODE = UPGRADE_STEP_2 requires DISABLE_GTID_UNSAFE_STATEMENTS = 1
- Error 1780: @@GLOBAL.GTID_MODE = ON or UPGRADE_STEP_1 or UPGRADE_STEP_2 requires –log-bin and –log
- Error 1781: @@SESSION.GTID_NEXT cannot be set to UUID:NUMBER when @@GLOBAL.GTID_MODE = OFF
- Error 1782: @@SESSION.GTID_NEXT cannot be set to ANONYMOUS when @@GLOBAL.GTID_MODE = ON
- Error 1783: @@SESSION.GTID_NEXT_LIST cannot be set to a non-NULL value when @@GLOBAL.GTID_MODE = OFF
- Error 1784: Found a Gtid_log_event or Previous_gtids_log_event when @@GLOBAL.GTID_MODE = OFF
- Error 1785: When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, updates to non-transactional tables can only
- Error 1786: CREATE TABLE … SELECT is forbidden when @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1
- Error 1787: When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, the statements CREATE TEMPORARY TABLE and DR
- Error 1788: The value of @@GLOBAL.GTID_MODE can only change one step at a time
- Error 1789: The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the master

- Error 1790: @@SESSION.GTID_NEXT cannot be changed by a client that owns a GTID
- Error 1791: Unknown EXPLAIN format name
- Error 1792: Cannot execute statement in a READ ONLY transaction
- Error 1793: Comment for table partition is too long
- Error 1794: Slave is not configured or failed to initialize properly. You must at least set –server
- Error 1795: InnoDB presently supports one FULLTEXT index creation at a time
- Error 1796: Cannot create FULLTEXT index on temporary InnoDB table
- Error 1797: Column is of wrong type for an InnoDB FULLTEXT index
- Error 1798: Index ‘%s’ is of wrong type for an InnoDB FULLTEXT index
- Error 1799: Creating index required more than ‘innodb_online_alter_log_max_size’ bytes of modificati

This page is licensed: CC BY-SA / Gnu FDL

Error 1700: GRANT with IDENTIFIED WITH is illegal because the user already exists

Error Code	SQLSTATE	Error	Description
1700	HY000	ER_GRANT_PLUGIN_USER_EXISTS	GRANT with IDENTIFIED WITH is illegal because the user %-.*s already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1701: Cannot truncate a table referenced in a foreign key constraint

Error Code	SQLSTATE	Error	Description
1701	42000	ER_TRUNCATE_ILLEGAL_FK	Cannot truncate a table referenced in a foreign key constraint (%s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1702: Plugin is force_plus_permanent and can not be unloaded

Error Code	SQLSTATE	Error	Description
1702	HY000	ER_PLUGIN_IS_PERMANENT	Plugin ‘%s’ is force_plus_permanent and can not be unloaded

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1703: The requested value for the heartbeat period is less than 1 millisecond

Error Code	SQLSTATE	Error	Description
1703	HY000	ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE_MIN	The requested value for the heartbeat period is less than 1 millisecond. The value is reset to 0, meaning that heartbeating will effectively be disabled.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1704: The requested value for the heartbeat period exceeds the value of slave_net_timeout seco

Error Code	SQLSTATE	Error	Description
1704	HY000	ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE_MAX	The requested value for the heartbeat period exceeds the value of slave_net_timeout seconds. A sensible value for the period should be less than the timeout.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1705: Multi-row statements required more than ‘max_binlog_stmt_cache_size’ bytes of storage; i

Error Code	SQLSTATE	Error	Description
1705	HY000	ER_STMT_CACHE_FULL	Multi-row statements required more than ‘max_binlog_stmt_cache_size’ bytes of storage; increase this mysqld variable and try again

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1706: Primary key/partition key update is not allowed since the table is updated both as and

Error Code	SQLSTATE	Error	Description
1706	HY000	ER_MULTI_UPDATE_KEY_CONFLICT	Primary key/partition key update is not allowed since the table is updated both as ‘%s’ and ‘%s’.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1707: Table rebuild required. Please do “ALTER TABLE FORCE” or dump/reload to fix it!

Error Code	SQLSTATE	Error	Description
1707	HY000	ER_TABLE_NEEDS_REBUILD	Table rebuild required. Please do “ALTER TABLE `%s` FORCE” or dump/reload to fix it!

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1708: The value of should be no less than the value of

Error Code	SQLSTATE	Error	Description
1708	HY000	WARN_OPTION_BELOW_LIMIT	The value of ‘%s’ should be no less than the value of ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1709: Index column size too large. The maximum column size is bytes.

Error Code	SQLSTATE	Error	Description
1709	HY000	ER_INDEX_COLUMN_TOO_LONG	Index column size too large. The maximum column size is %lu bytes.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1710: Trigger has an error in its body

Error Code	SQLSTATE	Error	Description
1710	HY000	ER_ERROR_IN_TRIGGER_BODY	Trigger ‘%s’ has an error in its body: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1711: Unknown trigger has an error in its body

Error Code	SQLSTATE	Error	Description
1711	HY000	ER_ERROR_IN_UNKNOWN_TRIGGER_BODY	Unknown trigger has an error in its body: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1712: Index is corrupted

Error Code	SQLSTATE	Error	Description
1712	HY000	ER_INDEX_CORRUPT	Index %s is corrupted

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1713: Undo log record is too big

Error Code	SQLSTATE	Error	Description
1713	HY000	ER_UNDO_RECORD_TOO_BIG	Undo log record is too big.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1714: INSERT IGNORE… SELECT is unsafe because the order in which rows are retrieved by the S

Error 1714: INSERT IGNORE… SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are ignored

Error Code	SQL STATE	Error	Description
1714	HY000	ER_BINLOG_UNSAFE_INSERT_IGNORE_SELECT	INSERT IGNORE… SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are ignored. This order cannot be predicted and may differ on master and the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1715: INSERT… SELECT… ON DUPLICATE KEY UPDATE is unsafe because the order in which rows ar

Error 1715: INSERT… SELECT… ON DUPLICATE KEY UPDATE is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are updated

Error Code	SQL STATE	Error	Description
1715	HY000	ER_BINLOG_UNSAFE_INSERT_SELECT_UPDATE	INSERT… SELECT… ON DUPLICATE KEY UPDATE is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are updated. This order cannot be predicted and may differ on master and the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1716: REPLACE… SELECT is unsafe because the order in which rows are retrieved by the SELECT

Error Code	SQLSTATE	Error	Description
1716	HY000	ER_BINLOG_UNSAFE_REPLACE_SELECT	REPLACE… SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are replaced. This order cannot be predicted and may differ on master and the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1717: CREATE… IGNORE SELECT is unsafe because the order in which rows are retrieved by the S

Error 1717: CREATE… IGNORE SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are ignored

Error Code	SQL STATE	Error	Description
1717	HY000	ER_BINLOG_UNSAFE_CREATE_IGNORE_SELECT	CREATE… IGNORE SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are ignored. This order cannot be predicted and may differ on master and the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1718: CREATE… REPLACE SELECT is unsafe because the order in which rows are retrieved by the

Error Code	SQL STATE	Error	Description
1718	HY000	ER_BINLOG_UNSAFE_CREATE_REPLACE_SELECT	CREATE… REPLACE SELECT is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are replaced. This order cannot be predicted and may differ on master and the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1719: UPDATE IGNORE is unsafe because the order in which rows are updated determines which (if

Error Code	SQLSTATE	Error	Description
1719	HY000	ER_BINLOG_UNSAFE_UPDATE_IGNORE	UPDATE IGNORE is unsafe because the order in which rows are updated determines which (if any) rows are ignored. This order cannot be predicted and may differ on master and the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1720: You should never see it

Error Code	SQLSTATE	Error	Description
1720	HY000	ER_UNUSED_15	You should never see it

Possible Causes and Solutions

Error 1721: You should never see it

Error Code	SQLSTATE	Error	Description
1721	HY000	ER_UNUSED_16	You should never see it

Possible Causes and Solutions

Error 1722: Statements writing to a table with an auto-increment column after selecting from another

Error Code	SQLSTATE	Error	Description
1722	HY000	ER_BINLOG_UNSAFE_WRITE_AUTOINC_SELECT	Statements writing to a table with an auto-increment column after selecting from another table are unsafe because the order in which rows are retrieved determines what (if any) rows will be written. This order cannot be predicted and may differ on master and the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1723: CREATE TABLE… SELECT… on a table with an auto-increment column is unsafe because the

Error Code	SQLSTATE	Error	Description
1723	HY000	ER_BINLOG_UNSAFE_CREATE_SELECT_AUTOINC	CREATE TABLE… SELECT… on a table with an auto-increment column is unsafe because the order in which rows are retrieved by the SELECT determines which (if any) rows are inserted. This order cannot be predicted and may differ on master and the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1724: INSERT… ON DUPLICATE KEY UPDATE on a table with more than one UNIQUE KEY is unsafe

Error Code	SQLSTATE	Error	Description
1724	HY000	ER_BINLOG_UNSAFE_INSERT_TWO_KEYS	INSERT… ON DUPLICATE KEY UPDATE on a table with more than one UNIQUE KEY is unsafe

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1725: Table is being used in foreign key check.

Error 1725: You should never see it

Error Code	SQLSTATE	Error	Description
1725	HY000	ER_UNUSED_28	You should never see it

Possible Causes and Solutions

Error 1726: Storage engine does not support system tables

Error Code	SQLSTATE	Error	Description
1726	HY000	ER_UNSUPPORTED_ENGINE	Storage engine '%s' does not support system tables. [%s.%s]

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1727: INSERT into autoincrement field which is not the first part in the composed primary key

Error Code	SQLSTATE	Error	Description
1727	HY000	ER_BINLOG_UNSAFE_AUTOINC_NOT_FIRST	INSERT into autoincrement field which is not the first part in the composed primary key is unsafe.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1728: Cannot load from. The table is probably corrupted

Error Code	SQLSTATE	Error	Description
1728	HY000	ER_CANNOT_LOAD_FROM_TABLE_V2	Cannot load from %s.%s. The table is probably corrupted

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1729: The requested value for the master delay exceeds the maximum

Error Code	SQLSTATE	Error	Description
1729	HY000	ER_MASTER_DELAY_VALUE_OUT_OF_RANGE	The requested value %s for the master delay exceeds the maximum %u

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1730: Only Format_description_log_event and row events are allowed in BINLOG statements

Error Code	SQLSTATE	Error	Description
1730	HY000	ER_ONLY_FD_AND_RBR_EVENTS_ALLOWED_IN_BINLOG_STATEMENT	Only Format_description_log_event and row events are allowed in BINLOG statements (but %s was provided

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1731: Non matching attribute between partition and table

Error Code	SQLSTATE	Error	Description
1731	HY000	ER_PARTITION_EXCHANGE_DIFFERENT_OPTION	Non matching attribute ‘%s’ between partition and table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1732: Table to exchange with partition is partitioned

Error Code	SQLSTATE	Error	Description
1732	HY000	ER_PARTITION_EXCHANGE_PART_TABLE	Table to exchange with partition is partitioned: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1733: Table to exchange with partition is temporary

Error Code	SQLSTATE	Error	Description
1733	HY000	ER_PARTITION_EXCHANGE_TEMP_TABLE	Table to exchange with partition is temporary: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1734: Subpartitioned table, use subpartition instead of partition

Error Code	SQLSTATE	Error	Description
1734	HY000	ER_PARTITION_INSTEAD_OF_SUBPARTITION	Subpartitioned table, use subpartition instead of partition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1735: Unknown partition in table

Error Code	SQLSTATE	Error	Description
1735	HY000	ER_UNKNOWN_PARTITION	Unknown partition ‘%s’ in table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1736: Tables have different definitions

Error Code	SQLSTATE	Error	Description
1736	HY000	ER_TABLES_DIFFERENT_METADATA	Tables have different definitions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1737: Found a row that does not match the partition

Error Code	SQLSTATE	Error	Description
1737	HY000	ER_ROW_DOES_NOT_MATCH_PARTITION	Found a row that does not match the partition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1738: Option binlog_cache_size is greater than max_binlog_cache_size; setting binlog_cache_siz

Error Code	SQLSTATE	Error	Description
1738	HY000	ER_BINLOG_CACHE_SIZE_GREATER_THAN_MAX	Option binlog_cache_size (%lu) is greater than max_binlog_cache_size (%lu); setting binlog_cache_size equal to max_binlog_cache_size.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1739: Cannot use access on index due to type or collation conversion on field

Error Code	SQLSTATE	Error	Description
1739	HY000	ER_WARN_INDEX_NOT_APPLICABLE	Cannot use %s access on index ‘%s’ due to type or collation conversion on field ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1740: Table to exchange with partition has foreign key references

Error Code	SQLSTATE	Error	Description
1740	HY000	ER_PARTITION_EXCHANGE_FOREIGN_KEY	Table to exchange with partition has foreign key references: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1741: Key value was not found in table

Error Code	SQLSTATE	Error	Description
1741	HY000	ER_NO_SUCH_KEY_VALUE	Key value ‘%s’ was not found in table ‘%s.%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1742: Data for column too long

Error Code	SQLSTATE	Error	Description
1742	HY000	ER_RPL_INFO_DATA_TOO_LONG	Data for column ‘%s’ too long

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1743: Replication event checksum verification failed while reading from network.

Error Code	SQLSTATE	Error	Description
1743	HY000	ER_NETWORK_READ_EVENT_CHECKSUM_FAILURE	Replication event checksum verification failed while reading from network.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1744: Replication event checksum verification failed while reading from a log file

Error Code	SQLSTATE	Error	Description
1744	HY000	ER_BINLOG_READ_EVENT_CHECKSUM_FAILURE	Replication event checksum verification failed while reading from a log file.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1745: Option binlog_stmt_cache_size is greater than max_binlog_stmt_cache_size

Error Code	SQLSTATE	Error	Description
1745	HY000	ER_BINLOG_STMT_CACHE_SIZE_GREATER_THAN_MAX	Option binlog_stmt_cache_size (%lu) is greater than max_binlog_stmt_cache_size (%lu); setting binlog_stmt_cache_size equal to max_binlog_stmt_cache_size.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1746: Can’t update table while is being created

Error Code	SQLSTATE	Error	Description
1746	HY000	ER_CANT_UPDATE_TABLE_IN_CREATE_TABLE_SELECT	Can’t update table ‘%s’ while ‘%s’ is being created.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1747: PARTITION () clause on non partitioned table

Error Code	SQLSTATE	Error	Description
1747	HY000	ER_PARTITION_CLAUSE_ON_NONPARTITIONED	PARTITION () clause on non partitioned table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1748: Found a row not matching the given partition set

Error Code	SQLSTATE	Error	Description
1748	HY000	ER_ROW_DOES_NOT_MATCH_GIVEN_PARTITION_SET	Found a row not matching the given partition set

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1749: partition doesn't exist

Error 1749: You should never see it

Error Code	SQLSTATE	Error	Description
1749	HY000	ER_UNUSED_5	You should never see it

Possible Causes and Solutions

Error 1750: Failure while changing the type of replication repository

Error Code	SQLSTATE	Error	Description
1750	HY000	ER_CHANGE_RPL_INFO_REPOSITORY_FAILURE	Failure while changing the type of replication repository: %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1751: The creation of some temporary tables could not be rolled back.

Error Code	SQLSTATE	Error	Description
1751	HY000	ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_CREATED_TEMP_TABLE	The creation of some temporary tables could not be rolled back.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1752: Some temporary tables were dropped, but these operations could not be rolled back.

Error Code	SQLSTATE	Error	Description
1752	HY000	ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_DROPPED_TEMP_TABLE	Some temporary tables were dropped, but these operations could not be rolled back.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1753: is not supported in multi-threaded slave mode

Error Code	SQLSTATE	Error	Description
1753	HY000	ER_MTS_FEATURE_IS_NOT_SUPPORTED	%s is not supported in multi-threaded slave mode. %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1754: The number of modified databases exceeds the maximum

Error Code	SQLSTATE	Error	Description
1754	HY000	ER_MTS_UPDATED_DBS_GREATER_MAX	The number of modified databases exceeds the maximum %d; the database names will not be included in the replication event metadata.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1755: Cannot execute the current event group in the parallel mode

Error Code	SQLSTATE	Error	Description
1755	HY000	ER_MTS_CANT_PARALLEL	Cannot execute the current event group in the parallel mode. Encountered event %s, relay-log name %s, position %s which prevents execution of this event group in parallel mode. Reason: %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1756: %s

Error Code	SQLSTATE	Error	Description
1756	HY000	ER_MTS_INCONSISTENT_DATA	%s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1757: FULLTEXT index is not supported for partitioned tables

Error Code	SQLSTATE	Error	Description
1757	HY000	ER_FULLTEXT_NOT_SUPPORTED_WITH_PARTITIONING	FULLTEXT index is not supported for partitioned tables.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1758: Invalid condition number

Error Code	SQLSTATE	Error	Description
1758	35000	ER_DA_INVALID_CONDITION_NUMBER	Invalid condition number

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1759: Sending passwords in plain text without SSL/TLS is extremely insecure

Error Code	SQLSTATE	Error	Description
1759	HY000	ER_INSECURE_PLAIN_TEXT	Sending passwords in plain text without SSL/TLS is extremely insecure.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1760: Storing MariaDB user name or password information in the master.info repository is not s

Error 1760: Storing MariaDB user name or password information in the master.info repository is not secure and is therefore not recommended

Error Code	SQLSTATE	Error	Description
1760	HY000	ER_INSECURE_CHANGE_MASTER	Storing MariaDB user name or password information in the master.info repository is not secure and is therefore not recommended. Please see the MariaDB Manual for more about this issue and possible alternatives

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1761: Foreign key constraint for table, record would lead to a duplicate entry in table

Error Code	SQLSTATE	Error	Description
1761	23000	ER_FOREIGN_DUPLICATE_KEY_WITH_CHILD_INFO	Foreign key constraint for table ‘%s’, record ‘%s’ would lead to a duplicate entry in table ‘%s’, key ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1762: Foreign key constraint for table, record would lead to a duplicate entry in a child table

Error Code	SQLSTATE	Error	Description
1762	23000	ER_FOREIGN_DUPLICATE_KEY_WITHOUT_CHILD_INFO	Foreign key constraint for table ‘%s’, record ‘%s’ would lead to a duplicate entry in a child table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1763: Setting authentication options is not possible when only the Slave SQL Thread is being s

Error 1763: Setting authentication options is not possible when only the Slave SQL Thread is being started

Error Code	SQLSTATE	Error	Description
1763	HY000	ER_SQLTHREAD_WITH_SECURE_SLAVE	Setting authentication options is not possible when only the Slave SQL Thread is being started.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1764: The table does not have FULLTEXT index to support this query

Error Code	SQLSTATE	Error	Description
1764	HY000	ER_TABLE_HAS_NO_FT	The table does not have FULLTEXT index to support this query

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1765: The system variable cannot be set in stored functions or triggers

Error Code	SQLSTATE	Error	Description
1765	HY000	ER_VARIABLE_NOT_SETTABLE_IN_SF_OR_TRIGGER	The system variable %s cannot be set in stored functions or triggers.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1766: The system variable cannot be set when there is an ongoing transaction

Error Code	SQLSTATE	Error	Description
1766	HY000	ER_VARIABLE_NOT_SETTABLE_IN_TRANSACTION	The system variable %s cannot be set when there is an ongoing transaction.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1767: The system variable @@SESSION.GTID_NEXT has the value which is not listed in @@SESSION.G

Error Code	SQLSTATE	Error	Description
1767	HY000	ER_GTID_NEXT_IS_NOT_IN_GTID_NEXT_LIST	The system variable @@SESSION.GTID_NEXT has the value %s, which is not listed in @@SESSION.GTID_NEXT_LIST.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1768: The system variable @@SESSION.GTID_NEXT cannot change inside a transaction

Error Code	SQLSTATE	Error	Description
1768	HY000	ER_CANT_CHANGE_GTID_NEXT_IN_TRANSACTION_WHEN_GTID_NEXT_LIST_IS_NULL	The system variable @@SESSION.GTID_NEXT cannot change inside a transaction.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1769: The statement ‘SET’ cannot invoke a stored function

Error Code	SQLSTATE	Error	Description
1769	HY000	ER_SET_STATEMENT_CANNOT_INVOKE_FUNCTION	The statement ‘SET %s’ cannot invoke a stored function.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1770: The system variable @@SESSION.GTID_NEXT cannot be ‘AUTOMATIC’ when @@SESSION.GTID_NEXT_L

Error Code	SQLSTATE	Error	Description
1770	HY000	ER_GTID_NEXT_CANT_BE_AUTOMATIC_IF_GTID_NEXT_LIST_IS_NON_NULL	The system variable @@SESSION.GTID_NEXT cannot be ‘AUTOMATIC’ when @@SESSION.GTID_NEXT_LIST is non-NULL.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1771: Skipping transaction because it has already been executed and logged

Error Code	SQLSTATE	Error	Description
1771	HY000	ER_SKIPPING_LOGGED_TRANSACTION	Skipping transaction %s because it has already been executed and logged.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1772: Malformed GTID set specification

Error Code	SQLSTATE	Error	Description
1772	HY000	ER_MALFORMED_GTID_SET_SPECIFICATION	Malformed GTID set specification ‘%s’.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1773: Malformed GTID set encoding

Error Code	SQLSTATE	Error	Description
1773	HY000	ER_MALFORMED_GTID_SET_ENCODING	Malformed GTID set encoding.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1774: Malformed GTID specification

Error Code	SQLSTATE	Error	Description
1774	HY000	ER_MALFORMED_GTID_SPECIFICATION	Malformed GTID specification ‘%s’.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1775: Impossible to generate Global Transaction Identifier: the integer component reached the

Error Code	SQLSTATE	Error	Description
1775	HY000	ER_GNO_EXHAUSTED	Impossible to generate Global Transaction Identifier: the integer component reached the maximal value. Restart the server with a new server_uuid.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1776: Parameters MASTER_LOG_FILE, MASTER_LOG_POS, RELAY_LOG_FILE and RELAY_LOG_POS cannot be s

Error Code	SQLSTATE	Error	Description
1776	HY000	ER_BAD_SLAVE_AUTO_POSITION	Parameters MASTER_LOG_FILE, MASTER_LOG_POS, RELAY_LOG_FILE and RELAY_LOG_POS cannot be set when MASTER_AUTO_POSITION is active.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1777: CHANGE MASTER TO MASTER_AUTO_POSITION = 1 can only be executed when @@GLOBAL.GTID_MODE =

Error Code	SQLSTATE	Error	Description
1777	HY000	ER_AUTO_POSITION_REQUIRES_GTID_MODE_ON	CHANGE MASTER TO MASTER_AUTO_POSITION = 1 can only be executed when @@GLOBAL.GTID_MODE = ON.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1778: Cannot execute statements with implicit commit inside a transaction when @@SESSION.GTID_

Error Code	SQLSTATE	Error	Description
1778	HY000	ER_CANT_DO_IMPLICIT_COMMIT_IN_TRX_WHEN_GTID_NEXT_IS_SET	Cannot execute statements with implicit commit inside a transaction when @@SESSION.GTID_NEXT != AUTOMATIC.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1779: GTID_MODE = ON or GTID_MODE = UPGRADE_STEP_2 requires DISABLE_GTID_UNSAFE_STATEMENTS = 1

Error Code	SQLSTATE	Error	Description
1779	HY000	ER_GTID_MODE_2_OR_3_REQUIRES_DISABLE_GTID_UNSAFE_STATEMENTS_ON	GTID_MODE = ON or GTID_MODE = UPGRADE_STEP_2 requires DISABLE_GTID_UNSAFE_STATEMENTS = 1.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1780: @@GLOBAL.GTID_MODE = ON or UPGRADE_STEP_1 or UPGRADE_STEP_2 requires –log-bin and –log

Error Code	SQLSTATE	Error	Description
1780	HY000	ER_GTID_MODE_REQUIRES_BINLOG	@@GLOBAL.GTID_MODE = ON or UPGRADE_STEP_1 or UPGRADE_STEP_2 requires –log-bin and –log-slave-updates.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1781: @@SESSION.GTID_NEXT cannot be set to UUID:NUMBER when @@GLOBAL.GTID_MODE = OFF

Error Code	SQLSTATE	Error	Description
1781	HY000	ER_CANT_SET_GTID_NEXT_TO_GTID_WHEN_GTID_MODE_IS_OFF	@@SESSION.GTID_NEXT cannot be set to UUID:NUMBER when @@GLOBAL.GTID_MODE = OFF.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1782: @@SESSION.GTID_NEXT cannot be set to ANONYMOUS when @@GLOBAL.GTID_MODE = ON

Error Code	SQLSTATE	Error	Description
1782	HY000	ER_CANT_SET_GTID_NEXT_TO_ANONYMOUS_WHEN_GTID_MODE_IS_ON	@@SESSION.GTID_NEXT cannot be set to ANONYMOUS when @@GLOBAL.GTID_MODE = ON.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1783: @@SESSION.GTID_NEXT_LIST cannot be set to a non-NULL value when @@GLOBAL.GTID_MODE = OFF

Error Code	SQLSTATE	Error	Description
1783	HY000	ER_CANT_SET_GTID_NEXT_LIST_TO_NON_NULL_WHEN_GTID_MODE_IS_OFF	@@SESSION.GTID_NEXT_LIST cannot be set to a non-NULL value when @@GLOBAL.GTID_MODE = OFF.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1784: Found a Gtid_log_event or Previous_gtids_log_event when @@GLOBAL.GTID_MODE = OFF

Error Code	SQLSTATE	Error	Description
1784	HY000	ER_FOUND_GTID_EVENT_WHEN_GTID_MODE_IS_OFF	Found a Gtid_log_event or Previous_gtids_log_event when @@GLOBAL.GTID_MODE = OFF.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1785: When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, updates to non-transactional tables can only

Error Code	SQLSTATE	Error	Description
1785	HY000	ER_GTID_UNSAFE_NON_TRANSACTIONAL_TABLE	When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, updates to non-transactional tables can only be done in either autocommitted statements or single-statement transactions, and never in the same statement as updates to transactional tables.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1786: CREATE TABLE … SELECT is forbidden when @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1

Error Code	SQLSTATE	Error	Description
1786	HY000	ER_GTID_UNSAFE_CREATE_SELECT	CREATE TABLE … SELECT is forbidden when @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1787: When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, the statements CREATE TEMPORARY TABLE and DR

Error Code	SQLSTATE	Error	Description
1787	HY000	ER_GTID_UNSAFE_CREATE_DROP_TEMPORARY_TABLE_IN_TRANSACTION	When @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1, the statements CREATE TEMPORARY TABLE and DROP TEMPORARY TABLE can be executed in a non-transactional context only, and require that AUTOCOMMIT = 1.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1788: The value of @@GLOBAL.GTID_MODE can only change one step at a time

Error Code	SQLSTATE	Error	Description
1788	HY000	ER_GTID_MODE_CAN_ONLY_CHANGE_ONE_STEP_AT_A_TIME	The value of @@GLOBAL.GTID_MODE can only change one step at a time: OFF <-> UPGRADE_STEP_1 <-> UPGRADE_STEP_2 <-> ON. Also note that this value must be stepped up or down simultaneously on all servers; see the Manual for instructions.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1789: The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the master

Error Code	SQLSTATE	Error	Description
1789	HY000	ER_MASTER_HAS_PURGED_REQUIRED_GTIDS	The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the master has purged binary logs containing GTIDs that the slave requires.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1790: @@SESSION.GTID_NEXT cannot be changed by a client that owns a GTID

Error Code	SQLSTATE	Error	Description
1790	HY000	ER_CANT_SET_GTID_NEXT_WHEN_OWNING_GTID	@@SESSION.GTID_NEXT cannot be changed by a client that owns a GTID. The client owns %s. Ownership is released on COMMIT or ROLLBACK.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1791: Unknown EXPLAIN format name

Error Code	SQLSTATE	Error	Description
1791	HY000	ER_UNKNOWN_EXPLAIN_FORMAT	Unknown EXPLAIN format name: ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1792: Cannot execute statement in a READ ONLY transaction

Error Code	SQLSTATE	Error	Description
1792	25006	ER_CANT_EXECUTE_IN_READ_ONLY_TRANSACTION	Cannot execute statement in a READ ONLY transaction.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1793: Comment for table partition is too long

Error Code	SQLSTATE	Error	Description
1793	HY000	ER_TOO_LONG_TABLE_PARTITION_COMMENT	Comment for table partition ‘%s’ is too long (max = %lu

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1794: Slave is not configured or failed to initialize properly. You must at least set –server

Error Code	SQLSTATE	Error	Description
1794	HY000	ER_SLAVE_CONFIGURATION	Slave is not configured or failed to initialize properly. You must at least set –server-id to enable either a master or a slave. Additional error messages can be found in the MariaDB error log

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1795: InnoDB presently supports one FULLTEXT index creation at a time

Error Code	SQLSTATE	Error	Description
1795	HY000	ER_INNODB_FT_LIMIT	InnoDB presently supports one FULLTEXT index creation at a time

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1796: Cannot create FULLTEXT index on temporary InnoDB table

Error Code	SQLSTATE	Error	Description
1796	HY000	ER_INNODB_NO_FT_TEMP_TABLE	Cannot create FULLTEXT index on temporary InnoDB table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1797: Column is of wrong type for an InnoDB FULLTEXT index

Error Code	SQLSTATE	Error	Description
1797	HY000	ER_INNODB_FT_WRONG_DOCID_COLUMN	Column ‘%s’ is of wrong type for an InnoDB FULLTEXT index

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1798: Index ‘%s’ is of wrong type for an InnoDB FULLTEXT index

Error Code	SQLSTATE	Error	Description
1798	HY000	ER_INNODB_FT_WRONG_DOCID_INDEX	Index ‘%s’ is of wrong type for an InnoDB FULLTEXT index

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1799: Creating index required more than 'innodb_online_alter_log_max_size' bytes of modificati

Error Code	SQLSTATE	Error	Description
1799	HY000	ER_INNODB_ONLINE_LOG_TOO_BIG	Creating index '%s' required more than 'innodb_online_alter_log_max_size' bytes of modification log. Please try again.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1800 to 1899

- Error 1800: Unknown ALGORITHM
- Error 1801: Unknown LOCK type
- Error 1802: CHANGE MASTER cannot be executed when the slave was stopped with an error or killed in M
- Error 1803: Cannot recover after SLAVE errored out in parallel execution mode
- Error 1804: Cannot clean up worker info tables
- Error 1805: Column count of is wrong. Expected, found. The table is probably corrupted
- Error 1806: Slave must silently retry current transaction
- Error 1807: There is a foreign key check running on table
- Error 1808: Schema mismatch
- Error 1809: Table in system tablespace
- Error 1810: IO Read error
- Error 1811: IO Write error
- Error 1812: Tablespace is missing for table
- Error 1813: Tablespace for table exists. Please DISCARD the tablespace before IMPORT
- Error 1814: Tablespace has been discarded for table
- Error 1815: Internal error
- Error 1816: ALTER TABLE IMPORT TABLESPACE failed with error
- Error 1817: Index corrupt
- Error 1818: YEAR(%lu) column type is deprecated. Creating YEAR(4) column instead.
- Error 1819: Your password does not satisfy the current policy requirements
- Error 1820: You must SET PASSWORD before executing this statement
- Error 1821: Failed to add the foreign key constraint. Missing index for constraint in the foreign ta
- Error 1822: Failed to add the foreign key constraint. Missing index for constraint in the referenced
- Error 1823: Failed to add the foreign key constraint to system tables
- Error 1824: Failed to open the referenced table
- Error 1825: Failed to add the foreign key constraint on table. Incorrect options in FOREIGN KEY cons
- Error 1826: Duplicate foreign key constraint name
- Error 1827: The password hash doesn't have the expected format
- Error 1828: Cannot drop column: needed in a foreign key constraint
- Error 1829: Cannot drop column: needed in a foreign key constraint of table
- Error 1830: Column cannot be NOT NULL: needed in a foreign key constraint SET NULL
- Error 1831: Duplicate index defined on the table. This is deprecated and will be disallowed in a fut
- Error 1832: Cannot change column: used in a foreign key constraint
- Error 1833: Cannot change column: used in a foreign key constraint of table
- Error 1834: Cannot delete rows from table which is parent in a foreign key constraint of table

- Error 1835: Malformed communication packet.
- Error 1836: Running in read-only mode
- Error 1837: When GTID_NEXT is set to a GTID, you must explicitly set it again after a COMMIT or ROLL
- Error 1838: The system variable cannot be set in stored procedures
- Error 1839: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_MODE = ON
- Error 1840: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty
- Error 1841: @@GLOBAL.GTID_PURGED can only be set when there are no ongoing transactions (not even in
- Error 1842: @@GLOBAL.GTID_PURGED was changed from to
- Error 1843: @@GLOBAL.GTID_EXECUTED was changed from to
- Error 1844: Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEM
- Error 1845: is not supported for this operation. Try
- Error 1846: is not supported. Reason: Try
- Error 1847: COPY algorithm requires a lock
- Error 1848: Partition specific operations do not yet support LOCK/ALGORITHM
- Error 1849: Columns participating in a foreign key are renamed
- Error 1850: Cannot change column type INPLACE
- Error 1851: Adding foreign keys needs foreign_key_checks=OFF
- Error 1852: Creating unique indexes with IGNORE requires COPY algorithm to remove duplicate rows
- Error 1853: Dropping a primary key is not allowed without also adding a new primary key
- Error 1854: Adding an auto-increment column requires a lock
- Error 1855: Cannot replace hidden FTS_DOC_ID with a user-visible one
- Error 1856: Cannot drop or rename FTS_DOC_ID
- Error 1857: Fulltext index creation requires a lock
- Error 1858: sql_slave_skip_counter can not be set when the server is running with @@GLOBAL.GTID_MODE
- Error 1859: Duplicate entry for key
- Error 1860: Long database name and identifier for object resulted in path length exceeding maximum c
- Error 1861: cannot silently convert NULL values, as required in this SQL_MODE
- Error 1862: Your password has expired. To log in you must change it using a client that supports exp
- Error 1863: Found a row in wrong partition
- Error 1864: Cannot schedule event, relay-log name, position to Worker thread because its size exceed
- Error 1865: Cannot CREATE FULLTEXT INDEX WITH PARSER on InnoDB table
- Error 1866: The binary log file is logically corrupted
- Error 1867: file was not purged because it was being read by thread(s), purged only out of
- Error 1868: file was not purged because it is the active log file
- Error 1869: Auto-increment value in UPDATE conflicts with internally generated values
- Error 1870: Row events are not logged for statements that modify BLACKHOLE tables in row format
- Error 1871: Slave failed to initialize master info structure from the repository
- Error 1872: Slave failed to initialize relay log info structure from the repository
- Error 1873: Access denied trying to change to user (using password). Disconnecting.
- Error 1874: InnoDB is in read only mode
- Error 1875: STOP SLAVE command execution is incomplete: Slave SQL thread got the stop signal
- Error 1876: STOP SLAVE command execution is incomplete: Slave IO thread got the stop signal
- Error 1877: Operation cannot be performed. The table is missing, corrupt or contains bad data
- Error 1878: Temporary file write failure
- Error 1879: Upgrade index name failed, please use create index(alter table) algorithm copy to rebuild

- Error 1880: ”

This page is licensed: CC BY-SA / Gnu FDL

Error 1800: Unknown ALGORITHM

Error Code	SQLSTATE	Error	Description
1800	HY000	ER_UNKNOWN_ALTER_ALGORITHM	Unknown ALGORITHM ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1801: Unknown LOCK type

Error Code	SQLSTATE	Error	Description
1801	HY000	ER_UNKNOWN_ALTER_LOCK	Unknown LOCK type ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1802: CHANGE MASTER cannot be executed when the slave was stopped with an error or killed in M

Error 1802: CHANGE MASTER cannot be executed when the slave was stopped with an error or killed in MTS mode

Error Code	SQLSTATE	Error	Description
1802	HY000	ER_MTS_CHANGE_MASTER_CANT_RUN_WITH_GAPS	CHANGE MASTER cannot be executed when the slave was stopped with an error or killed in MTS mode. Consider using RESET SLAVE or START SLAVE UNTIL.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1803: Cannot recover after SLAVE errored out in parallel execution mode

Error Code	SQLSTATE	Error	Description
1803	HY000	ER_MTS_RECOVERY_FAILURE	Cannot recover after SLAVE errored out in parallel execution mode. Additional error messages can be found in the MariaDB error log

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1804: Cannot clean up worker info tables

Error Code	SQLSTATE	Error	Description
1804	HY000	ER_MTS_RESET_WORKERS	Cannot clean up worker info tables. Additional error messages can be found in the MariaDB error log

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1805: Column count of is wrong. Expected, found. The table is probably corrupted

Error Code	SQLSTATE	Error	Description
1805	HY000	ER_COL_COUNT_DOESNT_MATCH_CORRUPTED_V2	Column count of %s.%s is wrong. Expected %d, found %d. The table is probably corrupted

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1806: Slave must silently retry current transaction

Error Code	SQLSTATE	Error	Description
1806	HY000	ER_SLAVE_SILENT_RETRY_TRANSACTION	Slave must silently retry current transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1807: There is a foreign key check running on table

Error 1807: You should never see it

Error Code	SQLSTATE	Error	Description
1807	HY000	ER_UNUSED_22	You should never see it

Possible Causes and Solutions

Error 1808: Schema mismatch

Error Code	SQLSTATE	Error	Description
1808	HY000	ER_TABLE_SCHEMA_MISMATCH	Schema mismatch (%s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1809: Table in system tablespace

Error Code	SQLSTATE	Error	Description
1809	HY000	ER_TABLE_IN_SYSTEM_TABLESPACE	Table ‘%s’ in system tablespace

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1810: IO Read error

Error Code	SQLSTATE	Error	Description
1810	HY000	ER_IO_READ_ERROR	IO Read error: (%lu, %s) %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1811: IO Write error

Error Code	SQLSTATE	Error	Description
1811	HY000	ER_IO_WRITE_ERROR	IO Write error: (%lu, %s) %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1812: Tablespace is missing for table

Error Code	SQLSTATE	Error	Description
1812	HY000	ER_TABLESPACE_MISSING	Tablespace is missing for table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1813: Tablespace for table exists. Please DISCARD the tablespace before IMPORT

Error Code	SQLSTATE	Error	Description
1813	HY000	ER_TABLESPACE_EXISTS	Tablespace for table ‘%s’ exists. Please DISCARD the tablespace before IMPORT.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1814: Tablespace has been discarded for table

Error Code	SQLSTATE	Error	Description
1814	HY000	ER_TABLESPACE_DISCARDED	Tablespace has been discarded for table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1815: Internal error

Error Code	SQLSTATE	Error	Description
1815	HY000	ER_INTERNAL_ERROR	Internal error: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1816: ALTER TABLE IMPORT TABLESPACE failed with error

Error Code	SQLSTATE	Error	Description
1816	HY000	ER_INNODB_IMPORT_ERROR	ALTER TABLE ‘%s’ IMPORT TABLESPACE failed with error %lu : ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1817: Index corrupt

Error Code	SQLSTATE	Error	Description
1817	HY000	ER_INNODB_INDEX_CORRUPT	Index corrupt: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1818: YEAR(%lu) column type is deprecated. Creating YEAR(4) column instead.

Error Code	SQLSTATE	Error	Description
1818	HY000	ER_INVALID_YEAR_COLUMN_LENGTH	YEAR(%lu) column type is deprecated. Creating YEAR(4) column instead.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1819: Your password does not satisfy the current policy requirements

Error Code	SQLSTATE	Error	Description
1819	HY000	ER_NOT_VALID_PASSWORD	Your password does not satisfy the current policy requirements

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1820: You must SET PASSWORD before executing this statement

Error Code	SQLSTATE	Error	Description
1820	HY000	ER_MUST_CHANGE_PASSWORD	You must SET PASSWORD before executing this statement

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1821: Failed to add the foreign key constraint. Missing index for constraint in the foreign ta

Error 1821: Failed to add the foreign key constraint. Missing index for constraint in the foreign tab

Error Code	SQLSTATE	Error	Description
1821	HY000	ER_FK_NO_INDEX_CHILD	Failed to add the foreign key constraint. Missing index for constraint ‘%s’ in the foreign table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1822: Failed to add the foreign key constraint. Missing index for constraint in the referenced

Error Code	SQLSTATE	Error	Description
1822	HY000	ER_FK_NO_INDEX_PARENT	Failed to add the foreign key constraint. Missing index for constraint ‘%s’ in the referenced table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1823: Failed to add the foreign key constraint to system tables

Error Code	SQLSTATE	Error	Description
1823	HY000	ER_FK_FAIL_ADD_SYSTEM	Failed to add the foreign key constraint ‘%s’ to system tables

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1824: Failed to open the referenced table

Error Code	SQLSTATE	Error	Description
1824	HY000	ER_FK_CANNOT_OPEN_PARENT	Failed to open the referenced table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1825: Failed to add the foreign key constraint on table. Incorrect options in FOREIGN KEY cons

Error Code	SQLSTATE	Error	Description
1825	HY000	ER_FK_INCORRECT_OPTION	Failed to add the foreign key constraint on table ‘%s’. Incorrect options in FOREIGN KEY constraint ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1826: Duplicate foreign key constraint name

Error Code	SQLSTATE	Error	Description
1826	HY000	ER_FK_DUP_NAME	Duplicate foreign key constraint name ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1827: The password hash doesn’t have the expected format

Error Code	SQLSTATE	Error	Description
1827	HY000	ER_PASSWORD_FORMAT	The password hash doesn’t have the expected format. Check if the correct password algorithm is being used with the PASSWORD() function.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1828: Cannot drop column: needed in a foreign key constraint

Error Code	SQLSTATE	Error	Description
1828	HY000	ER_FK_COLUMN_CANNOT_DROP	Cannot drop column ‘%s’: needed in a foreign key constraint ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1829: Cannot drop column: needed in a foreign key constraint of table

Error Code	SQLSTATE	Error	Description
1829	HY000	ER_FK_COLUMN_CANNOT_DROP_CHILD	Cannot drop column ‘%s’: needed in a foreign key constraint ‘%s’ of table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1830: Column cannot be NOT NULL: needed in a foreign key constraint SET NULL

Error Code	SQLSTATE	Error	Description
1830	HY000	ER_FK_COLUMN_NOT_NULL	Column ‘%s’ cannot be NOT NULL: needed in a foreign key constraint ‘%s’ SET NULL

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1831: Duplicate index defined on the table. This is deprecated and will be disallowed in a fut

Error Code	SQLSTATE	Error	Description
1831	HY000	ER_DUP_INDEX	Duplicate index ‘%s’ defined on the table ‘%s.%s’. This is deprecated and will be disallowed in a future release.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1832: Cannot change column: used in a foreign key constraint

Error Code	SQLSTATE	Error	Description
1832	HY000	ER_FK_COLUMN_CANNOT_CHANGE	Cannot change column ‘%s’: used in a foreign key constraint ‘%s’

Possible Causes and Solutions

An `ALTER TABLE` statement tried to change a column that the table uses as a foreign key, in a way that would change how its values compare with the referenced column. Changing the column’s type, length, character set, or

collation all qualify. A common case is converting a table to a different collation, where the referenced table still uses the old one.

Setting `foreign_key_checks` to `0` does not lift this restriction, and neither does choosing a different `ALGORITHM`.

To make the change, drop the constraint in the same `ALTER TABLE` statement as the column change, apply the matching change to the referenced table, then re-create the constraint. For the full procedure, see Changing the Character Set or Collation of a Foreign Key Column.

If the referenced table is the one being altered, the server reports error 1833 instead.

This page is licensed: CC BY-SA / Gnu FDL

Error 1833: Cannot change column: used in a foreign key constraint of table

Error Code	SQLSTATE	Error	Description
1833	HY000	ER_FK_COLUMN_CANNOT_CHANGE_CHILD	Cannot change column '%s': used in a foreign key constraint '%s' of table '%s'

Possible Causes and Solutions

An `ALTER TABLE` statement tried to change a column that another table references through a foreign key, in a way that would change how its values compare with the referencing column. Changing the column's type, length, character set, or collation all qualify. A common case is converting a table to a different collation, where the referencing table still uses the old one. The third name in the message is the table holding the constraint.

Setting `foreign_key_checks` to `0` does not lift this restriction, and neither does choosing a different `ALGORITHM`.

A referencing table's constraint cannot be dropped from within the referenced table's `ALTER TABLE` statement, so drop it on the referencing table first, then apply the change to both tables and re-create the constraint. For the full procedure, see Changing the Character Set or Collation of a Foreign Key Column.

If the table being altered is the one holding the constraint, the server reports error 1832 instead.

This page is licensed: CC BY-SA / Gnu FDL

Error 1834: Cannot delete rows from table which is parent in a foreign key constraint of table

Error Code	SQLSTATE	Error	Description
1834	HY000	ER_FK_CANNOT_DELETE_PARENT	Cannot delete rows from table which is parent in a foreign key constraint '%s' of table '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1835: Malformed communication packet.

Error Code	SQLSTATE	Error	Description
1835	HY000	ER_MALFORMED_PACKET	Malformed communication packet.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1836: Running in read-only mode

Error Code	SQLSTATE	Error	Description
1836	HY000	ER_READ_ONLY_MODE	Running in read-only mode

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1837: When GTID_NEXT is set to a GTID, you must explicitly set it again after a COMMIT or ROLL

Error Code	SQLSTATE	Error	Description
1837	HY000	ER_GTID_NEXT_TYPE_UNDEFINED_GROUP	When GTID_NEXT is set to a GTID, you must explicitly set it again after a COMMIT or ROLLBACK. If you see this error message in the slave SQL thread, it means that a table in the current transaction is transactional on the master and non-transactional on the slave. In a client connection, it means that you executed SET GTID_NEXT before a transaction and forgot to set GTID_NEXT to a different identifier or to 'AUTOMATIC' after COMMIT or ROLLBACK. Current GTID_NEXT is '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1838: The system variable cannot be set in stored procedures

Error Code	SQLSTATE	Error	Description
1838	HY000	ER_VARIABLE_NOT_SETTABLE_IN_SP	The system variable %s cannot be set in stored procedures.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1839: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_MODE = ON

Error Code	SQLSTATE	Error	Description
1839	HY000	ER_CANT_SET_GTID_PURGED_WHEN_GTID_MODE_IS_OFF	@@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_MODE = ON.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1840: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty

Error Code	SQLSTATE	Error	Description
1840	HY000	ER_CANT_SET_GTID_PURGED_WHEN_GTID_EXECUTED_IS_NOT_EMPTY	@@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1841: @@GLOBAL.GTID_PURGED can only be set when there are no ongoing transactions (not even in

Error Code	SQLSTATE	Error	Description
1841	HY000	ER_CANT_SET_GTID_PURGED_WHEN_OWNED_GTIDS_IS_NOT_EMPTY	@@GLOBAL.GTID_PURGED can only be set when there are no ongoing transactions (not even in other clients).

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1842: @@GLOBAL.GTID_PURGED was changed from to

Error Code	SQLSTATE	Error	Description
1842	HY000	ER_GTID_PURGED_WAS_CHANGED	@@GLOBAL.GTID_PURGED was changed from '%s' to '%s'.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1843: @@GLOBAL.GTID_EXECUTED was changed from to

Error Code	SQLSTATE	Error	Description
1843	HY000	ER_GTID_EXECUTED_WAS_CHANGED	@@GLOBAL.GTID_EXECUTED was changed from '%s' to '%s'.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1844: Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEM

Error Code	SQLSTATE	Error	Description
1844	HY000	ER_BINLOG_STMT_MODE_AND_NO_REPL_TABLES	Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEMENT, and both replicated and non replicated tables are written to.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1845: is not supported for this operation. Try

Error Code	SQLSTATE	Error	Description
1845	0A000	ER_ALTER_OPERATION_NOT_SUPPORTED	%s is not supported for this operation. Try %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1846: is not supported. Reason: Try

Error Code	SQLSTATE	Error	Description
1846	0A000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON	%s is not supported. Reason: %s. Try %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1847: COPY algorithm requires a lock

Error Code	SQLSTATE	Error	Description
1847	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COPY	COPY algorithm requires a lock

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1848: Partition specific operations do not yet support LOCK/ALGORITHM

Error Code	SQLSTATE	Error	Description
1848	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_PARTITION	Partition specific operations do not yet support LOCK/ALGORITHM

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1849: Columns participating in a foreign key are renamed

Error Code	SQLSTATE	Error	Description
1849	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_RENAME	Columns participating in a foreign key are renamed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1850: Cannot change column type INPLACE

Error Code	SQLSTATE	Error	Description
1850	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COLUMN_TYPE	Cannot change column type INPLACE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1851: Adding foreign keys needs foreign_key_checks=OFF

Error Code	SQLSTATE	Error	Description
1851	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_CHECK	Adding foreign keys needs foreign_key_checks=OFF

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1852: Creating unique indexes with IGNORE requires COPY algorithm to remove duplicate rows

Error Code	SQLSTATE	Error	Description
1852	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_IGNORE	Creating unique indexes with IGNORE requires COPY algorithm to remove duplicate rows

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1853: Dropping a primary key is not allowed without also adding a new primary key

Error Code	SQLSTATE	Error	Description
1853	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOPK	Dropping a primary key is not allowed without also adding a new primary key

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1854: Adding an auto-increment column requires a lock

Error Code	SQLSTATE	Error	Description
1854	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_AUTOINC	Adding an auto-increment column requires a lock

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1855: Cannot replace hidden FTS_DOC_ID with a user-visible one

Error Code	SQLSTATE	Error	Description
1855	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_HIDDEN_FTS	Cannot replace hidden FTS_DOC_ID with a user-visible one

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1856: Cannot drop or rename FTS_DOC_ID

Error Code	SQLSTATE	Error	Description
1856	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_CHANGE_FTS	Cannot drop or rename FTS_DOC_ID

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1857: Fulltext index creation requires a lock

Error Code	SQLSTATE	Error	Description
1857	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FTS	Fulltext index creation requires a lock

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1858: sql_slave_skip_counter can not be set when the server is running with @@GLOBAL.GTID_MODE

Error Code	SQLSTATE	Error	Description
1858	HY000	ER_SQL_SLAVE_SKIP_COUNTER_NOT_SETTABLE_IN_GTID_MODE	sql_slave_skip_counter can not be set when the server is running with @@GLOBAL.GTID_MODE = ON. Instead, for each transaction that you want to skip, generate an empty transaction with the same GTID as the transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1859: Duplicate entry for key

Error Code	SQLSTATE	Error	Description
1859	23000	ER_DUP_UNKNOWN_IN_INDEX	Duplicate entry for key ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1860: Long database name and identifier for object resulted in path length exceeding maximum c

Error 1860: Long database name and identifier for object resulted in path length exceeding maximum characters

Error Code	SQLSTATE	Error	Description
1860	HY000	ER_IDENT_CAUSES_TOO_LONG_PATH	Long database name and identifier for object resulted in path length exceeding %d characters. Path: ‘%s’.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1861: cannot silently convert NULL values, as required in this SQL_MODE

Error Code	SQLSTATE	Error	Description
1861	HY000	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOT_NULL	cannot silently convert NULL values, as required in this SQL_MODE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1862: Your password has expired. To log in you must change it using a client that supports exp

Error 1862: Your password has expired. To log in you must change it using a client that supports expired passwords

Error Code	SQLSTATE	Error	Description
1862	HY000	ER_MUST_CHANGE_PASSWORD_LOGIN	Your password has expired. To log in you must change it using a client that supports expired passwords.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1863: Found a row in wrong partition

Error Code	SQLSTATE	Error	Description
1863	HY000	ER_ROW_IN_WRONG_PARTITION	Found a row in wrong partition %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1864: Cannot schedule event, relay-log name, position to Worker thread because its size exceed

Error Code	SQLSTATE	Error	Description
1864	HY000	ER_MTS_EVENT_BIGGER_PENDING_JOBS_SIZE_MAX	Cannot schedule event %s, relay-log name %s, position %s to Worker thread because its size %lu exceeds %lu of slave_pending_jobs_size_max.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1865: Cannot CREATE FULLTEXT INDEX WITH PARSER on InnoDB table

Error Code	SQLSTATE	Error	Description
1865	HY000	ER_INNODB_NO_FT_USES_PARSER	Cannot CREATE FULLTEXT INDEX WITH PARSER on InnoDB table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1866: The binary log file is logically corrupted

Error Code	SQLSTATE	Error	Description
1866	HY000	ER_BINLOG_LOGICAL_CORRUPTION	The binary log file ‘%s’ is logically corrupted: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1867: file was not purged because it was being read by thread(s), purged only out of

Error Code	SQLSTATE	Error	Description
1867	HY000	ER_WARN_PURGE_LOG_IN_USE	file %s was not purged because it was being read by %d thread(s), purged only %d out of %d files.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1868: file was not purged because it is the active log file

Error Code	SQLSTATE	Error	Description
1868	HY000	ER_WARN_PURGE_LOG_IS_ACTIVE	file %s was not purged because it is the active log file.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1869: Auto-increment value in UPDATE conflicts with internally generated values

Error Code	SQLSTATE	Error	Description
1869	HY000	ER_AUTO_INCREMENT_CONFLICT	Auto-increment value in UPDATE conflicts with internally generated values

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1870: Row events are not logged for statements that modify BLACKHOLE tables in row format

Error Code	SQLSTATE	Error	Description
1870	HY000	WARN_ON_BLOCKHOLE_IN_RBR	Row events are not logged for %s statements that modify BLACKHOLE tables in row format. Table(s): '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1871: Slave failed to initialize master info structure from the repository

Error Code	SQLSTATE	Error	Description
1871	HY000	ER_SLAVE_MI_INIT_REPOSITORY	Slave failed to initialize master info structure from the repository

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1872: Slave failed to initialize relay log info structure from the repository

Error Code	SQLSTATE	Error	Description
1872	HY000	ER_SLAVE_RLI_INIT_REPOSITORY	Slave failed to initialize relay log info structure from the repository

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1873: Access denied trying to change to user (using password). Disconnecting.

Error Code	SQLSTATE	Error	Description
1873	28000	ER_ACCESS_DENIED_CHANGE_USER_ERROR	Access denied trying to change to user ‘%s’@‘%s’ (using password: %s). Disconnecting.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1874: InnoDB is in read only mode

Error Code	SQLSTATE	Error	Description
1874	HY000	ER_INNODB_READ_ONLY	InnoDB is in read only mode.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1875: STOP SLAVE command execution is incomplete: Slave SQL thread got the stop signal

Error Code	SQLSTATE	Error	Description
1875	HY000	ER_STOP_SLAVE_SQL_THREAD_TIMEOUT	STOP SLAVE command execution is incomplete: Slave SQL thread got the stop signal, thread is busy, SQL thread will stop once the current task is complete.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1876: STOP SLAVE command execution is incomplete: Slave IO thread got the stop signal

Error Code	SQLSTATE	Error	Description
1876	HY000	ER_STOP_SLAVE_IO_THREAD_TIMEOUT	STOP SLAVE command execution is incomplete: Slave IO thread got the stop signal, thread is busy, IO thread will stop once the current task is complete.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1877: Operation cannot be performed. The table is missing, corrupt or contains bad data

Error Code	SQLSTATE	Error	Description
1877	HY000	ER_TABLE_CORRUPT	Operation cannot be performed. The table ‘%s.%s’ is missing, corrupt or contains bad data.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1878: Temporary file write failure

Error Code	SQLSTATE	Error	Description
1878	HY000	ER_TEMP_FILE_WRITE_FAILURE	Temporary file write failure.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1879: Upgrade index name failed, please use create index(alter table) algorithm copy to rebuild

Error Code	SQLSTATE	Error	Description
1879	HY000	ER_INNODB_FT_AUX_NOT_HEX_ID	Upgrade index name failed, please use create index(alter table) algorithm copy to rebuild index.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1880: ”

Error Code	SQLSTATE	Error	Description
1880		ER_LAST_MYSQL_ERROR_MESSAGE	”

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 1900 to 1999

- Error 1900: ”
- Error 1901: Function or expression cannot be used in the clause of
- Error 1902: ”
- Error 1903: Primary key cannot be defined upon a generated column
- Error 1904: Key/Index cannot be defined on a virtual generated column
- Error 1905: Cannot define foreign key with clause on a generated column
- Error 1906: The value specified for generated column in table has been ignored
- Error 1907: This is not yet supported for generated columns
- Error 1908: ”
- Error 1909: ”
- Error 1910: storage engine does not support generated columns
- Error 1911: Unknown option
- Error 1912: Incorrect value for option
- Error 1913: You should never see it
- Error 1914: You should never see it
- Error 1915: You should never see it
- Error 1916: Got overflow when converting to. Value truncated
- Error 1917: Truncated value when converting to
- Error 1918: Encountered illegal value when converting to
- Error 1919: Encountered illegal format of dynamic column string

- Error 1920: Dynamic column implementation limit reached
- Error 1921: Illegal value used as argument of dynamic column function
- Error 1922: Dynamic column contains unknown character set
- Error 1923: At least one of the ‘in_to_exists’ or ‘materialization’ optimizer_switch flags must be ’
- Error 1924: Query cache is disabled (resize or similar command in progress); repeat this command lat
- Error 1925: Query cache is globally disabled and you can’t enable it only for this session
- Error 1926: View ORDER BY clause ignored because there is other ORDER BY clause already
- Error 1927: Connection was killed
- Error 1928: You should never see it
- Error 1929: Cannot modify @@session.skip_replication inside a transaction
- Error 1930: Cannot modify @@session.skip_replication inside a stored function or trigger
- Error 1931: Query execution was interrupted. The query exceeded. The query result may be incomplete
- Error 1932: Table doesn’t exist in engine
- Error 1933: Target is not running an EXPLAINable command
- Error 1934: Connection conflicts with existing connection
- Error 1935: Master
- Error 1936: Can’t %s SLAVE
- Error 1937: SLAVE started
- Error 1938: SLAVE stopped
- Error 1939: Engine failed to discover table with
- Error 1940: Failed initializing replication GTID state
- Error 1941: Could not parse GTID list
- Error 1942: Could not update replication slave gtid state
- Error 1943: GTID conflict (duplicate domain id)
- Error 1944: Failed to open
- Error 1945: Connecting slave requested to start from GTID, which is not in the master’s binlog
- Error 1946: Failed to load replication slave GTID position from table
- Error 1947: Specified GTID conflicts with the binary log which contains a more recent GTID
- Error 1948: Specified value for @@gtid_slave_pos contains no value for replication domain
- Error 1949: START SLAVE UNTIL master_gtid_pos requires that slave is using GTID
- Error 1950: An attempt was made to binlog GTID which would create an out-of-order sequence number wi
- Error 1951: The binlog on the master is missing the GTID requested by the slave
- Error 1952: Unexpected GTID received from master after reconnect
- Error 1953: Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a transaction
- Error 1954: Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a stored function
- Error 1955: Connecting slave requested to start from GTID which is not in the master’s binlog
- Error 1956: This operation is not allowed if any GTID has been logged to the binary log
- Error 1957: Unknown query id
- Error 1958: Bad base64 data as position
- Error 1959: Invalid role specification
- Error 1960: The current user is invalid
- Error 1961: Cannot grant role to
- Error 1962: Cannot revoke role from
- Error 1963: Cannot change @@slave_parallel_threads while another change is in progress
- Error 1964: Commit failed due to failure of an earlier commit on which this one depends

- Error 1965: is a view
- Error 1966: When using GTID, @@sql_slave_skip_counter can not be used
- Error 1967: The definition for table is too big
- Error 1968: Plugin already installed
- Error 1969: Query execution was interrupted (max_statement_time exceeded)
- Error 1970: does not support subqueries or stored functions
- Error 1971: The system variable cannot be set in SET STATEMENT
- Error 1972: You should never see it
- Error 1973: Can't create user; it already exists
- Error 1974: Can't drop user; it doesn't exist
- Error 1975: Can't create role; it already exists
- Error 1976: Can't drop role; it doesn't exist
- Error 1977: Cannot convert character to
- Error 1978: Incorrect default value for column
- Error 1979: You are not owner of query
- Error 1980: Engine-independent statistics are not collected for column
- Error 1981: Aggregate function of SELECT belongs to SELECT
- Error 1982: option ignored for InnoDB partition

This page is licensed: CC BY-SA / Gnu FDL

Error 1900: ”

Error 1900: You should never see it

Error Code	SQLSTATE	Error	Description
1900	HY000	ER_UNUSED_18	You should never see it

Possible Causes and Solutions

Error 1901: Function or expression cannot be used in the clause of

Error Code	SQLSTATE	Error	Description
1901		ER_GENERATED_COLUMN_FUNCTION_IS_NOT_ALLOWED	Function or expression ‘%s’ cannot be used in the %s clause of %`s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1902: ”

Error 1902: You should never see it

Error Code	SQLSTATE	Error	Description
1902	HY000	ER_UNUSED_19	You should never see it

Possible Causes and Solutions

Error 1903: Primary key cannot be defined upon a generated column

Error Code	SQLSTATE	Error	Description
1903		ER_PRIMARY_KEY_BASED_ON_GENERATED_COLUMN	Primary key cannot be defined upon a generated column

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1904: Key/Index cannot be defined on a virtual generated column

Error Code	SQLSTATE	Error	Description
1904		ER_KEY_BASED_ON_GENERATED_VIRTUAL_COLUMN	Key/Index cannot be defined on a virtual generated column

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1905: Cannot define foreign key with clause on a generated column

Error Code	SQLSTATE	Error	Description
1905		ER_WRONG_FK_OPTION_FOR_GENERATED_COLUMN	Cannot define foreign key with %s clause on a generated column

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1906: The value specified for generated column in table has been ignored

Error Code	SQLSTATE	Error	Description
1906		ER_WARNING_NON_DEFAULT_VALUE_FOR_GENERATED_COLUMN	The value specified for generated column ‘%s’ in table ‘%s’ has been ignored

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1907: This is not yet supported for generated columns

Error Code	SQLSTATE	Error	Description
1907		ER_UNSUPPORTED_ACTION_ON_GENERATED_COLUMN	This is not yet supported for generated columns

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1908: ”

Error 1908: You should never see it

Error Code	SQLSTATE	Error	Description
1908	HY000	ER_UNUSED_20	You should never see it

Possible Causes and Solutions

Error 1909: ”

Error 1909: You should never see it

Error Code	SQLSTATE	Error	Description
1909	HY000	ER_UNUSED_21	You should never see it

Possible Causes and Solutions

Error 1910: storage engine does not support generated columns

Error Code	SQLSTATE	Error	Description
1910		ER_UNSUPPORTED_ENGINE_FOR_GENERATED_COLUMNS	%s storage engine does not support generated columns

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1911: Unknown option

Error Code	SQLSTATE	Error	Description
1911		ER_UNKNOWN_OPTION	Unknown option ‘%-.64s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1912: Incorrect value for option

Error Code	SQLSTATE	Error	Description
1912		ER_BAD_OPTION_VALUE	Incorrect value ‘%-.64s’ for option ‘%-.64s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1913: You should never see it

Error Code	SQLSTATE	Error	Description
1913	HY000	ER_UNUSED_6	You should never see it

Possible Causes and Solutions

Error 1914: You should never see it

Error Code	SQLSTATE	Error	Description
1914	HY000	ER_UNUSED_7	You should never see it

Possible Causes and Solutions

Error 1915: You should never see it

Error Code	SQLSTATE	Error	Description
1915	HY000	ER_UNUSED_8	You should never see it

Possible Causes and Solutions

Error 1916: Got overflow when converting to. Value truncated

Error Code	SQLSTATE	Error	Description
1916		ER_DATA_OVERFLOW 22003	Got overflow when converting '%-.128s' to %-.32s. Value truncated.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1917: Truncated value when converting to

Error Code	SQLSTATE	Error	Description
1917		ER_DATA_TRUNCATED 22003	Truncated value '%-.128s' when converting to %-.32s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1918: Encountered illegal value when converting to

Error Code	SQLSTATE	Error	Description
1918		ER_BAD_DATA 22007	Encountered illegal value '%-.128s' when converting to %-.32s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1919: Encountered illegal format of dynamic column string

Error Code	SQLSTATE	Error	Description
1919		ER_DYN_COL_WRONG_FORMAT	Encountered illegal format of dynamic column string

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1920: Dynamic column implementation limit reached

Error Code	SQLSTATE	Error	Description
1920		ER_DYN_COL_IMPLEMENTATION_LIMIT	Dynamic column implementation limit reached

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1921: Illegal value used as argument of dynamic column function

Error Code	SQLSTATE	Error	Description
1921		ER_DYN_COL_DATA 22007	Illegal value used as argument of dynamic column function

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1922: Dynamic column contains unknown character set

Error Code	SQLSTATE	Error	Description
1922		ER_DYN_COL_WRONG_CHARSET	Dynamic column contains unknown character set

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1923: At least one of the ‘in_to_exists’ or ‘materialization’ optimizer_switch flags must be ’

Error Code	SQLSTATE	Error	Description
1923		ER_ILLEGAL_SUBQUERY_OPTIMIZER_SWITCHES	At least one of the ‘in_to_exists’ or ‘materialization’ optimizer_switch flags must be ‘on’.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1924: Query cache is disabled (resize or similar command in progress); repeat this command lat

Error 1924: Query cache is disabled (resize or similar command in progress); repeat this command later

Error Code	SQLSTATE	Error	Description
1924		ER_QUERY_CACHE_IS_DISABLED	Query cache is disabled (resize or similar command in progress); repeat this command later

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1925: Query cache is globally disabled and you can’t enable it only for this session

Error Code	SQLSTATE	Error	Description
1925		ER_QUERY_CACHE_IS_GLOBALY_DISABLED	Query cache is globally disabled and you can’t enable it only for this session

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1926: View ORDER BY clause ignored because there is other ORDER BY clause already

Error Code	SQLSTATE	Error	Description
1926		ER_VIEW_ORDERBY_IGNORED	View ‘%-.192s’.’%-.192s’ ORDER BY clause ignored because there is other ORDER BY clause already.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1927: Connection was killed

Error Code	SQLSTATE	Error	Description
1927	70100	ER_CONNECTION_KILLED	Connection was killed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1928: You should never see it

Error Code	SQLSTATE	Error	Description
1928	HY000	ER_UNUSED_12	You should never see it

Possible Causes and Solutions

Error 1929: Cannot modify @@session.skip_replication inside a transaction

Error Code	SQLSTATE	Error	Description
1929		ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_SKIP_REPLICATION	Cannot modify @@session.skip_replication inside a transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1930: Cannot modify @@session.skip_replication inside a stored function or trigger

Error Code	SQLSTATE	Error	Description
1930		ER_STORED_FUNCTION_PREVENTS_SWITCH_SKIP_REPLICATION	Cannot modify @@session.skip_replication inside a stored function or trigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1931: Query execution was interrupted. The query exceeded. The query result may be incomplete

MariaDB starting with 10.5.22

> = MariaDB 11.1.2, MariaDB 10.11.5, MariaDB 10.6.15, MariaDB 10.5.22:

Error Code	SQLSTATE	Error	Description
1931		ER_QUERY_RESULT_INCOMPLETE	Query execution was interrupted. The query exceeded %s %llu. The query result may be incomplete

MariaDB until 10.6.14

<= MariaDB 11.1.1, MariaDB 10.11.4, MariaDB 10.6.14, MariaDB 10.5.21:

Error Code	SQLSTATE	Error	Description
1931		ER_QUERY_EXCEEDED_ROWS_EXAMINED_LIMIT	Query execution was interrupted. The query examined at least %llu rows, which exceeds LIMIT ROWS EXAMINED (%llu). The query result may be incomplete.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1932: Table doesn't exist in engine

Error Code	SQLSTATE	Error	Description
1932		ER_NO_SUCH_TABLE_IN_ENGINE 42S02	Table '%-.192s.%-.192s' doesn't exist in engine

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1933: Target is not running an EXPLAINable command

Error Code	SQLSTATE	Error	Description
1933		ER_TARGET_NOT_EXPLAINABLE	Target is not running an EXPLAINable command

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1934: Connection conflicts with existing connection

Error Code	SQLSTATE	Error	Description
1934		ER_CONNECTION_ALREADY_EXISTS	Connection '%.*s' conflicts with existing connection '%.*s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1935: Master

Error Code	SQLSTATE	Error	Description
1935		ER_MASTER_LOG_PREFIX	Master '%.*s':

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1936: Can't %s SLAVE

Error Code	SQLSTATE	Error	Description
1936		ER_CANT_START_STOP_SLAVE	Can't %s SLAVE '%.*s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1937: SLAVE started

Error Code	SQLSTATE	Error	Description
1937		ER_SLAVE_STARTED	SLAVE '%.*s' started

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1938: SLAVE stopped

Error Code	SQLSTATE	Error	Description
1938		ER_SLAVE_STOPPED	SLAVE '%.*s' stopped

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1939: Engine failed to discover table with

Error Code	SQLSTATE	Error	Description
1939		ER_SQL_DISCOVER_ERROR	Engine %s failed to discover table % `-.192s.%` -.192s with '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1940: Failed initializing replication GTID state

Error Code	SQLSTATE	Error	Description
1940		ER_FAILED_GTID_STATE_INIT	Failed initializing replication GTID state

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1941: Could not parse GTID list

Error Code	SQLSTATE	Error	Description
1941		ER_INCORRECT_GTID_STATE	Could not parse GTID list

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1942: Could not update replication slave gtid state

Error Code	SQLSTATE	Error	Description
1942		ER_CANNOT_UPDATE_GTID_STATE	Could not update replication slave gtid state

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1943: GTID conflict (duplicate domain id)

Error Code	SQLSTATE	Error	Description
1943		ER_DUPLICATE_GTID_DOMAIN	GTID %u-%u-%llu and %u-%u-%llu conflict (duplicate domain id %u)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1944: Failed to open

Error Code	SQLSTATE	Error	Description
1944		ER_GTID_OPEN_TABLE_FAILED	Failed to open %s.%s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1945: Connecting slave requested to start from GTID, which is not in the master’s binlog

Error Code	SQLSTATE	Error	Description
1945		ER_GTID_POSITION_NOT_FOUND_IN_BINLOG	Connecting slave requested to start from GTID %u-%u-%llu, which is not in the master’s binlog

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1946: Failed to load replication slave GTID position from table

Error Code	SQLSTATE	Error	Description
1946		ER_CANNOT_LOAD_SLAVE_GTID_STATE	Failed to load replication slave GTID position from table %s.%s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1947: Specified GTID conflicts with the binary log which contains a more recent GTID

Error Code	SQL STATE	Error	Description
1947		ER_MASTER_GTID_POS_CONFLICTS_WITH_BINLOG	Specified GTID %u-%u-%llu conflicts with the binary log which contains a more recent GTID %u-%u-%llu. If MASTER_GTID_POS=CURRENT_POS is used, the binlog position will override the new value of @@gtid_slave_pos.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1948: Specified value for @@gtid_slave_pos contains no value for replication domain

Error Code	SQL STATE	Error	Description
1948		ER_MASTER_GTID_POS_MISSING_DOMAIN	Specified value for @@gtid_slave_pos contains no value for replication domain %u. This conflicts with the binary log which contains GTID %u-%u-%llu. If MASTER_GTID_POS=CURRENT_POS is used, the binlog position will override the new value of @@gtid_slave_pos.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1949: START SLAVE UNTIL master_gtid_pos requires that slave is using GTID

Error Code	SQLSTATE	Error	Description
1949		ER_UNTIL_REQUIRES_USING_GTID	START SLAVE UNTIL master_gtid_pos requires that slave is using GTID

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1950: An attempt was made to binlog GTID which would create an out-of-order sequence number wi

Error Code	SQLSTATE	Error	Description
1950		ER_GTID_STRICT_OUT_OF_ORDER	An attempt was made to binlog GTID %u-%u-%llu which would create an out-of-order sequence number with existing GTID %u-%u-%llu, and gtid strict mode is enabled.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1951: The binlog on the master is missing the GTID requested by the slave

Error Code	SQLSTATE	Error	Description
1951		ER_GTID_START_FROM_BINLOG_HOLE	The binlog on the master is missing the GTID %u-%u-%llu requested by the slave (even though a subsequent sequence number does exist), and GTID strict mode is enabled

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1952: Unexpected GTID received from master after reconnect

Error Code	SQLSTATE	Error	Description
1952		ER_SLAVE_UNEXPECTED_MASTER_SWITCH	Unexpected GTID received from master after reconnect. This normally indicates that the master server was replaced without restarting the slave threads. %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1953: Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a transaction

Error Code	SQLSTATE	Error	Description
1953		ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_GTID_DOMAIN_ID_SEQ_NO	Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a transaction

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1954: Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a stored function

Error Code	SQLSTATE	Error	Description
1954		ER_STORED_FUNCTION_PREVENTS_SWITCH_GTID_DOMAIN_ID_SEQ_NO	Cannot modify @@session.gtid_domain_id or @@session.gtid_seq_no inside a stored function or trigger

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1955: Connecting slave requested to start from GTID which is not in the master’s binlog

Error Code	SQLSTATE	Error	Description
1955		ER_GTID_POSITION_NOT_FOUND_IN_BINLOG2	Connecting slave requested to start from GTID %u-%u-%llu, which is not in the master’s binlog. Since the master’s binlog contains GTIDs with higher sequence numbers, it probably means that the slave has diverged due to executing extra erroneous transactions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1956: This operation is not allowed if any GTID has been logged to the binary log

Error Code	SQLSTATE	Error	Description
1956		ER_BINLOG_MUST_BE_EMPTY	This operation is not allowed if any GTID has been logged to the binary log. Run RESET MASTER first to erase the log

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1957: Unknown query id

Error Code	SQLSTATE	Error	Description
1957		ER_NO_SUCH_QUERY	Unknown query id: %lld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1958: Bad base64 data as position

Error Code	SQLSTATE	Error	Description
1958		ER_BAD_BASE64_DATA	Bad base64 data as position %u

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1959: Invalid role specification

Error Code	SQLSTATE	Error	Description
1959		ER_INVALID_ROLE	Invalid role specification %`s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1960: The current user is invalid

Error Code	SQLSTATE	Error	Description
1960		ER_INVALID_CURRENT_USER	The current user is invalid.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1961: Cannot grant role to

Error Code	SQLSTATE	Error	Description
1961		ER_CANNOT_GRANT_ROLE	Cannot grant role ‘%s’ to: %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1962: Cannot revoke role from

Error Code	SQLSTATE	Error	Description
1962		ER_CANNOT_REVOKE_ROLE	Cannot revoke role ‘%s’ from: %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1963: Cannot change @@slave_parallel_threads while another change is in progress

Error Code	SQLSTATE	Error	Description
1963		ER_CHANGE_SLAVE_PARALLEL_THREADS_ACTIVE	Cannot change @@slave_parallel_threads while another change is in progress

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1964: Commit failed due to failure of an earlier commit on which this one depends

Error Code	SQLSTATE	Error	Description
1964		ER_PRIOR_COMMIT_FAILED	Commit failed due to failure of an earlier commit on which this one depends

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1965: is a view

Error Code	SQLSTATE	Error	Description
1965		ER_IT_IS_A_VIEW	‘%-.192s’ is a view

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1966: When using GTID, @@sql_slave_skip_counter can not be used

Error Code	SQLSTATE	Error	Description
1966		ER_SLAVE_SKIP_NOT_IN_GTID	When using GTID, @@sql_slave_skip_counter can not be used. Instead, setting @@gtid_slave_pos explicitly can be used to skip to after a given GTID position.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1967: The definition for table is too big

Error Code	SQLSTATE	Error	Description
1967		ER_TABLE_DEFINITION_TOO_BIG	The definition for table %`s is too big

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1968: Plugin already installed

Error Code	SQLSTATE	Error	Description
1968		ER_PLUGIN_INSTALLED	Plugin '%-.192s' already installed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1969: Query execution was interrupted (max_statement_time exceeded)

Error Code	SQLSTATE	Error	Description
1969		ER_STATEMENT_TIMEOUT	Query execution was interrupted (max_statement_time exceeded)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1970: does not support subqueries or stored functions

Error Code	SQLSTATE	Error	Description
1970		ER_SUBQUERIES_NOT_SUPPORTED	%s does not support subqueries or stored functions.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1971: The system variable cannot be set in SET STATEMENT

Error Code	SQLSTATE	Error	Description
1971		ER_SET_STATEMENT_NOT_SUPPORTED	The system variable %.200s cannot be set in SET STATEMENT.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1972: You should never see it

Error Code	SQLSTATE	Error	Description
1972	HY000	ER_UNUSED_9	You should never see it

Possible Causes and Solutions

Error 1973: Can't create user; it already exists

Error Code	SQLSTATE	Error	Description
1973		ER_USER_CREATE_EXISTS	Can't create user '%-.64s'@'%-.64s'; it already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1974: Can’t drop user; it doesn’t exist

Error Code	SQLSTATE	Error	Description
1974		ER_USER_DROP_EXISTS	Can’t drop user ‘%-.64s’@‘%-.64s’; it doesn’t exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1975: Can’t create role; it already exists

Error Code	SQLSTATE	Error	Description
1975		ER_ROLE_CREATE_EXISTS	Can’t create role ‘%-.64s’; it already exists

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1976: Can’t drop role; it doesn’t exist

Error Code	SQLSTATE	Error	Description
1976		ER_ROLE_DROP_EXISTS	Can’t drop role ‘%-.64s’; it doesn’t exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1977: Cannot convert character to

Error Code	SQLSTATE	Error	Description
1977		ER_CANNOT_CONVERT_CHARACTER	Cannot convert ‘%s’ character 0x%-.64s to ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1978: Incorrect default value for column

Error Code	SQLSTATE	Error	Description
1978		ER_INVALID_DEFAULT_VALUE_FOR_FIELD	Incorrect default value ‘%-.128s’ for column ‘%.192s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1979: You are not owner of query

Error Code	SQLSTATE	Error	Description
1979		ER_KILL_QUERY_DENIED_ERROR	You are not owner of query %lu

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1980: Engine-independent statistics are not collected for column

Error Code	SQLSTATE	Error	Description
1980		ER_NO_EIS_FOR_FIELD	Engine-independent statistics are not collected for column '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1981: Aggregate function of SELECT belongs to SELECT

Error Code	SQLSTATE	Error	Description
1981		ER_WARN_AGGFUNC_DEPENDENCE	Aggregate function '%-.192s)' of SELECT #%d belongs to SELECT #%d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 1982: option ignored for InnoDB partition

Error Code	SQLSTATE	Error	Description
1982		WARN_INNODB_PARTITION_OPTION_IGNORED	<%-.64s> option ignored for InnoDB partition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 3000 to 3099

- Error 3000: File is corrupted
- Error 3001: Query partially completed on the master (error on master) and was aborted
- Error 3002: Query caused different errors on master and slave
- Error 3003: Storage engine for table is not loaded
- Error 3004: GET STACKED DIAGNOSTICS when handler not active
- Error 3005: is no longer supported. The statement was converted to
- Error 3006: Statement is unsafe because it uses a fulltext parser plugin which may not return the sa
- Error 3007: Cannot DISCARD/IMPORT tablespace associated with temporary table
- Error 3008: Foreign key cascade delete/update exceeds max depth of
- Error 3009: Column count of is wrong
- Error 3010: Trigger does not have CREATED attribute
- Error 3011: Referenced trigger for the given action time and event type does not exist
- Error 3012: EXPLAIN FOR CONNECTION command is supported only for SELECT/UPDATE/INSERT/DELETE/REPLACE
- Error 3013: Invalid size for column
- Error 3014: Table storage engine found required create option missing
- Error 3015: Out of memory in storage engine

- Error 3016: The password for anonymous user cannot be expired
- Error 3017: This operation cannot be performed with a running slave sql thread
- Error 3018: Cannot create FULLTEXT index on materialized subquery
- Error 3019: Undo Log error
- Error 3020: Invalid argument for logarithm
- Error 3021: This operation cannot be performed with a running slave io thread
- Error 3022: This operation may not be safe when the slave has temporary tables
- Error 3023: CHANGE MASTER TO with a MASTER_LOG_FILE clause but no MASTER_LOG_POS clause may not be s
- Error 3024: Query execution was interrupted, maximum statement execution time exceeded
- Error 3025: Select is not a read only statement, disabling timer
- Error 3026: Duplicate entry
- Error 3027: mode no longer has any effect. Use STRICT_ALL_TABLES or STRICT_TRANS_TABLES instead.
- Error 3028: Expression of ORDER BY contains aggregate function and applies to a UNION
- Error 3029: Expression of ORDER BY contains aggregate function and applies to the result of a non-ag
- Error 3030: Slave worker has stopped after at least one previous worker encountered an error when sl
- Error 3031: slave_preserve_commit_order is not supported
- Error 3032: The server is currently in offline mode
- Error 3033: Binary geometry function given two geometries of different srids
- Error 3034: Calling geometry function with unsupported types of arguments
- Error 3035: Unknown GIS error occurred in function
- Error 3036: Unknown exception caught in GIS function
- Error 3037: Invalid GIS data provided to function
- Error 3038: The geometry has no data in function
- Error 3039: Unable to calculate centroid because geometry is empty in function
- Error 3040: Geometry overlay calculation error: geometry data is invalid in function
- Error 3041: Geometry turn info calculation error: geometry data is invalid in function
- Error 3042: Analysis procedures of intersection points interrupted unexpectedly in function
- Error 3043: Unknown exception thrown in function
- Error 3044: Memory allocation error
- Error 3045: Domain error
- Error 3046: Length error
- Error 3047: Invalid argument error
- Error 3048: Out of range error
- Error 3049: Overflow error error
- Error 3050: Range error
- Error 3051: Underflow error
- Error 3052: Logic error
- Error 3053: Runtime error
- Error 3054: Unknown exception
- Error 3055: Geometry byte string must be little endian
- Error 3056: The password provided for the replication user exceeds the maximum length of 32 characte
- Error 3057: Incorrect user-level lock name
- Error 3058: Deadlock found when trying to get user-level lock
- Error 3059: REPLACE cannot be executed as it requires deleting rows that are not in the view

- Error 3060: Do not support online operation on table with GIS index

This page is licensed: CC BY-SA / Gnu FDL

Error 3000: File is corrupted

Error Code	SQLSTATE	Error	Description
3000		ER_FILE_CORRUPT	File %s is corrupted

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3001: Query partially completed on the master (error on master) and was aborted

Error 3001: Query partially completed on the master (error on master: %d) and was aborted. There is a chance that your master is inconsistent at this point. If you are sure that your master is ok, run this query manually on the slave and then restart the slave with SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1; START SLAVE;. Query:'%s'"

Error Code	SQLSTATE	Error	Description
3001		ER_ERROR_ON_MASTER	Query partially completed on the master (error on master: %d) and was aborted. There is a chance that your master is inconsistent at this point. If you are sure that your master is ok, run this query manually on the slave and then restart the slave with SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1; START SLAVE;. Query:'%s'"

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3002: Query caused different errors on master and slave

Error Code	SQLSTATE	Error	Description
3002		ER_INCONSISTENT_ERROR	Query caused different errors on master and slave. Error on master: message (format)='%s' error code=%d; Error on slave:actual message='%s', error code=%d. Default database:'%s'. Query:'%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3003: Storage engine for table is not loaded

Error Code	SQLSTATE	Error	Description
3003		ER_STORAGE_ENGINE_NOT_LOADED	Storage engine for table '%s'.'%s' is not loaded.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3004: GET STACKED DIAGNOSTICS when handler not active

Error Code	SQLSTATE	Error	Description
3004		ER_GET_STACKED_DA_WITHOUT_ACTIVE_HANDLER 0Z002	GET STACKED DIAGNOSTICS when handler not active

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3005: is no longer supported. The statement was converted to

Error Code	SQLSTATE	Error	Description
3005		ER_WARN_LEGACY_SYNTAX_CONVERTED	%s is no longer supported. The statement was converted to %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3006: Statement is unsafe because it uses a fulltext parser plugin which may not return the sa

Error 3006: Statement is unsafe because it uses a fulltext parser plugin which may not return the same value on the slave

Error Code	SQLSTATE	Error	Description
3006		ER_BINLOG_UNSAFE_FULLTEXT_PLUGIN	Statement is unsafe because it uses a fulltext parser plugin which may not return the same value on the slave.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3007: Cannot DISCARD/IMPORT tablespace associated with temporary table

Error Code	SQLSTATE	Error	Description
3007		ER_CANNOT_DISCARD_TEMPORARY_TABLE	Cannot DISCARD/IMPORT tablespace associated with temporary table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3008: Foreign key cascade delete/update exceeds max depth of

Error Code	SQLSTATE	Error	Description
3008		ER_FK_DEPTH_EXCEEDED	Foreign key cascade delete/update exceeds max depth of %d.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3009: Column count of is wrong

Error Code	SQLSTATE	Error	Description
3009		ER_COL_COUNT_DOESNT_MATCH_PLEASE_UPDATE_V2	Column count of %s.%s is wrong. Expected %d, found %d. Created with MariaDB %d, now running %d. Please use mysql_upgrade to fix this error.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3010: Trigger does not have CREATED attribute

Error Code	SQLSTATE	Error	Description
3010		ER_WARN_TRIGGER_DOESNT_HAVE_CREATED	Trigger %s.%s.%s does not have CREATED attribute.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3011: Referenced trigger for the given action time and event type does not exist

Error Code	SQLSTATE	Error	Description
3011		ER_REFERENCED_TRG_DOES_NOT_EXIST_MYSQL	Referenced trigger ‘%s’ for the given action time and event type does not exist.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3012: EXPLAIN FOR CONNECTION command is supported only for SELECT/UPDATE/INSERT/DELETE/REPLACE

Error Code	SQLSTATE	Error	Description
3012		ER_EXPLAIN_NOT_SUPPORTED	EXPLAIN FOR CONNECTION command is supported only for SELECT/UPDATE/INSERT/DELETE/REPLACE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3013: Invalid size for column

Error Code	SQLSTATE	Error	Description
3013		ER_INVALID_FIELD_SIZE	Invalid size for column ‘%-.192s’.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3014: Table storage engine found required create option missing

Error Code	SQLSTATE	Error	Description
3014		ER_MISSING_HA_CREATE_OPTION	Table storage engine ‘%-.64s’ found required create option missing

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3015: Out of memory in storage engine

Error Code	SQLSTATE	Error	Description
3015		ER_ENGINE_OUT_OF_MEMORY	Out of memory in storage engine '%-.64s'.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3016: The password for anonymous user cannot be expired

Error Code	SQLSTATE	Error	Description
3016		ER_PASSWORD_EXPIRE_ANONYMOUS_USER	The password for anonymous user cannot be expired.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3017: This operation cannot be performed with a running slave sql thread

Error Code	SQLSTATE	Error	Description
3017		ER_SLAVE_SQL_THREAD_MUST_STOP	This operation cannot be performed with a running slave sql thread; run STOP SLAVE SQL_THREAD first

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3018: Cannot create FULLTEXT index on materialized subquery

Error Code	SQLSTATE	Error	Description
3018		ER_NO_FT_MATERIALIZED_SUBQUERY	Cannot create FULLTEXT index on materialized subquery

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3019: Undo Log error

Error Code	SQLSTATE	Error	Description
3019		ER_INNODB_UNDO_LOG_FULL	Undo Log error: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3020: Invalid argument for logarithm

Error Code	SQLSTATE	Error	Description
3020		ER_INVALID_ARGUMENT_FOR_LOGARITHM	Invalid argument for logarithm

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3021: This operation cannot be performed with a running slave io thread

Error Code	SQLSTATE	Error	Description
3021		ER_SLAVE_CHANNEL_IO_THREAD_MUST_STOP	This operation cannot be performed with a running slave io thread; run STOP SLAVE IO_THREAD FOR CHANNEL '%s' first.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3022: This operation may not be safe when the slave has temporary tables

Error Code	SQLSTATE	Error	Description
3022		ER_WARN_OPEN_TEMP_TABLES_MUST_BE_ZERO	This operation may not be safe when the slave has temporary tables. The tables will be kept open until the server restarts or until the tables are deleted by any replicated DROP statement. Suggest to wait until slave_open_temp_tables = 0.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3023: CHANGE MASTER TO with a MASTER_LOG_FILE clause but no MASTER_LOG_POS clause may not be s

Error Code	SQLSTATE	Error	Description
3023		ER_WARN_ONLY_MASTER_LOG_FILE_NO_POS	CHANGE MASTER TO with a MASTER_LOG_FILE clause but no MASTER_LOG_POS clause may not be safe. The old position value may not be valid for the new binary log file.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3024: Query execution was interrupted, maximum statement execution time exceeded

Error 3024: You should never see it

Error Code	SQLSTATE	Error	Description
3024	HY000	ER_UNUSED_1	You should never see it

Possible Causes and Solutions

The cause is that the executing statement took more than max_statement_time (primary) or slave_max_statement_time (replica) to execute.

To fix this, you have to either increase the value of max_statement_time or find out why the query is slow. Start by doing an EXPLAIN or ANALYZE on the query to find out how the query is executed.

See Also

- Aborting Statements that Exceed a Certain Time to Execute

Error 3025: Select is not a read only statement, disabling timer

Error Code	SQLSTATE	Error	Description
3025		ER_NON_RO_SELECT_DISABLE_TIMER	Select is not a read only statement, disabling timer

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3026: Duplicate entry

Error Code	SQLSTATE	Error	Description
3026		ER_DUP_LIST_ENTRY	Duplicate entry '%-.192s'.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3027: mode no longer has any effect. Use STRICT_ALL_TABLES or STRICT_TRANS_TABLES instead.

Error Code	SQLSTATE	Error	Description
3027		ER_SQL_MODE_NO_EFFECT	'%s' mode no longer has any effect. Use STRICT_ALL_TABLES or STRICT_TRANS_TABLES instead.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3028: Expression of ORDER BY contains aggregate function and applies to a UNION

Error Code	SQLSTATE	Error	Description
3028		ER_AGGREGATE_ORDER_FOR_UNION	Expression #%u of ORDER BY contains aggregate function and applies to a UNION

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3029: Expression of ORDER BY contains aggregate function and applies to the result of a non-ag

Error Code	SQLSTATE	Error	Description
3029		ER_AGGREGATE_ORDER_NON_AGG_QUERY	Expression #%u of ORDER BY contains aggregate function and applies to the result of a non-aggregated query

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3030: Slave worker has stopped after at least one previous worker encountered an error when sl

Error 3030: Slave worker has stopped after at least one previous worker encountered an error when slave-preserve-commit-order was enabled

Error Code	SQLSTATE	Error	Description
3030		ER_SLAVE_WORKER_STOPPED_PREVIOUS_THD_ERROR	Slave worker has stopped after at least one previous worker encountered an error when slave-preserve-commit-order was enabled. To preserve commit order, the last transaction executed by this thread has not been committed. When restarting the slave after fixing any failed threads, you should fix this worker as well.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3031: slave_preserve_commit_order is not supported

Error Code	SQLSTATE	Error	Description
3031		ER_DONT_SUPPORT_SLAVE_PRESERVE_COMMIT_ORDER	slave_preserve_commit_order is not supported %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3032: The server is currently in offline mode

Error Code	SQLSTATE	Error	Description
3032		ER_SERVER_OFFLINE_MODE	The server is currently in offline mode

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3033: Binary geometry function given two geometries of different srids

Error Code	SQLSTATE	Error	Description
3033		ER_GIS_DIFFERENT_SRIDS	Binary geometry function %s given two geometries of different srids: %u and %u, which should have been identical.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3034: Calling geometry function with unsupported types of arguments

Error Code	SQLSTATE	Error	Description
3034		ER_GIS_UNSUPPORTED_ARGUMENT	Calling geometry function %s with unsupported types of arguments.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3035: Unknown GIS error occurred in function

Error Code	SQLSTATE	Error	Description
3035		ER_GIS_UNKNOWN_ERROR	Unknown GIS error occurred in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3036: Unknown exception caught in GIS function

Error Code	SQLSTATE	Error	Description
3036		ER_GIS_UNKNOWN_EXCEPTION	Unknown exception caught in GIS function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3037: Invalid GIS data provided to function

Error Code	SQLSTATE	Error	Description
3037		ER_GIS_INVALID_DATA	Invalid GIS data provided to function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3038: The geometry has no data in function

Error Code	SQLSTATE	Error	Description
3038		ER_BOOST_GEOMETRY_EMPTY_INPUT_EXCEPTION	The geometry has no data in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3039: Unable to calculate centroid because geometry is empty in function

Error Code	SQLSTATE	Error	Description
3039		ER_BOOST_GEOMETRY_CENTROID_EXCEPTION	Unable to calculate centroid because geometry is empty in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3040: Geometry overlay calculation error: geometry data is invalid in function

Error Code	SQLSTATE	Error	Description
3040		ER_BOOST_GEOMETRY_OVERLAY_INVALID_INPUT_EXCEPTION	Geometry overlay calculation error: geometry data is invalid in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3041: Geometry turn info calculation error: geometry data is invalid in function

Error Code	SQLSTATE	Error	Description
3041		ER_BOOST_GEOMETRY_TURN_INFO_EXCEPTION	Geometry turn info calculation error: geometry data is invalid in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3042: Analysis procedures of intersection points interrupted unexpectedly in function

Error Code	SQLSTATE	Error	Description
3042		ER_BOOST_GEOMETRY_SELF_INTERSECTION_POINT_EXCEPTION	Analysis procedures of intersection points interrupted unexpectedly in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3043: Unknown exception thrown in function

Error Code	SQLSTATE	Error	Description
3043		ER_BOOST_GEOMETRY_UNKNOWN_EXCEPTION	Unknown exception thrown in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3044: Memory allocation error

Error Code	SQLSTATE	Error	Description
3044		ER_STD_BAD_ALLOC_ERROR	Memory allocation error: %-.256s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3045: Domain error

Error Code	SQLSTATE	Error	Description
3045		ER_STD_DOMAIN_ERROR	Domain error: %-.256s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3046: Length error

Error Code	SQLSTATE	Error	Description
3046		ER_STD_LENGTH_ERROR	Length error: %-.256s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3047: Invalid argument error

Error Code	SQLSTATE	Error	Description
3047		ER_STD_INVALID_ARGUMENT	Invalid argument error: %-.256s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3048: Out of range error

Error Code	SQLSTATE	Error	Description
3048		ER_STD_OUT_OF_RANGE_ERROR	Out of range error: %-.256s in function %s.

Possible Causes and Solutions

Function Arguments Outside the Expected Bounds

Functions arguments have an expected range, and this error is returned if this range is exceeded. Follow the documentation for the specific function indicated in the error message. For example, the ST_GeoHash function expects the longitude and latitude parameters to be between -180 and 180. In this case, the longitude argument is invalid:

```
SELECT ST_GeoHash(181,30,15);
ERROR 3048 (HY000): Out of range error: Longitude should be [-180,180] in function ST_GeoHash.
```

By providing a longitude argument within the expected range, the error is no longer returned:

```
SELECT ST_GeoHash(180,30,15);
+-----------------------+
| ST_GeoHash(180,30,15) |
+-----------------------+
| xvrfxvrfxvrfxvr       |
+-----------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Error 3049: Overflow error error

Error Code	SQLSTATE	Error	Description
3049		ER_STD_OVERFLOW_ERROR	Overflow error error: %-.256s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3050: Range error

Error Code	SQLSTATE	Error	Description
3050		ER_STD_RANGE_ERROR	Range error: %-.256s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3051: Underflow error

Error Code	SQLSTATE	Error	Description
3051		ER_STD_UNDERFLOW_ERROR	Underflow error: %-.256s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3052: Logic error

Error Code	SQLSTATE	Error	Description
3052		ER_STD_LOGIC_ERROR	Logic error: %-.256s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3053: Runtime error

Error Code	SQLSTATE	Error	Description
3053		ER_STD_RUNTIME_ERROR	Runtime error: %-.256s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3054: Unknown exception

Error Code	SQLSTATE	Error	Description
3054		ER_STD_UNKNOWN_EXCEPTION	Unknown exception: %-.384s in function %s.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3055: Geometry byte string must be little endian

Error Code	SQLSTATE	Error	Description
3055		ER_GIS_DATA_WRONG_ENDIANESS	Geometry byte string must be little endian.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3056: The password provided for the replication user exceeds the maximum length of 32 characte

Error 3056: The password provided for the replication user exceeds the maximum length of 32 characters

Error Code	SQLSTATE	Error	Description
3056		ER_CHANGE_MASTER_PASSWORD_LENGTH	The password provided for the replication user exceeds the maximum length of 32 characters

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3057: Incorrect user-level lock name

Error Code	SQLSTATE	Error	Description
3057	42000	ER_USER_LOCK_WRONG_NAME	Incorrect user-level lock name ‘%-.192s’.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3058: Deadlock found when trying to get user-level lock

Error Code	SQLSTATE	Error	Description
3058		ER_USER_LOCK_DEADLOCK	Deadlock found when trying to get user-level lock; try rolling back transaction/releasing locks and restarting lock acquisition.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3059: REPLACE cannot be executed as it requires deleting rows that are not in the view

Error Code	SQLSTATE	Error	Description
3059		ER_REPLACE_INACCESSIBLE_ROWS	REPLACE cannot be executed as it requires deleting rows that are not in the view

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 3060: Do not support online operation on table with GIS index

Error Code	SQLSTATE	Error	Description
3060		ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_GIS	Do not support online operation on table with GIS index

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 4000 to 4099

- Error 4000: COM_MULTI can't return a result set in the given context
- Error 4001: Command is not allowed for COM_MULTI
- Error 4002: WITH column list and SELECT field list have different column counts
- Error 4003: Too many WITH elements in WITH clause
- Error 4004: Duplicate query name in WITH clause
- Error 4005: No anchors for recursive WITH element
- Error 4006: Unacceptable mutual recursion with anchored table
- Error 4007: Reference to recursive WITH table in materialized derived
- Error 4008: Restrictions imposed on recursive definitions are violated for table
- Error 4009: Window specification with name is not defined
- Error 4010: Multiple window specifications with the same name
- Error 4011: Window specification referencing another one cannot contain partition list
- Error 4012: Referenced window specification already contains order list
- Error 4013: Referenced window specification cannot contain window frame
- Error 4014: Unacceptable combination of window frame bound specifications
- Error 4015: Window function is allowed only in SELECT list and ORDER BY clause
- Error 4016: Window function is not allowed in window specification
- Error 4017: Window frame is not allowed with
- Error 4018: No order list in window specification for
- Error 4019: RANGE-type frame requires ORDER BY clause with single sort key
- Error 4020: Integer is required for ROWS-type frame
- Error 4021: Numeric datatype is required for RANGE-type frame
- Error 4022: Frame exclusion is not supported yet
- Error 4023: This window function may not have a window frame
- Error 4024: Argument of NTILE must be greater than 0
- Error 4025: CONSTRAINT failed for
- Error 4026: Expression in the clause is too big
- Error 4027: Got an error evaluating stored expression
- Error 4028: Got an error when calculating default value for
- Error 4029: Expression for field is referring to uninitialized field
- Error 4030: Only one DEFAULT partition allowed
- Error 4031: Referenced trigger for the given action time and event type does not exist
- Error 4032: Default/ignore value is not supported for such parameter usage
- Error 4033: Only row based replication supported for bulk operations
- Error 4034: Uncompress the compressed binlog failed
- Error 4035: Broken JSON string in argument to function at position
- Error 4036: Character disallowed in JSON in argument to function at position
- Error 4037: Unexpected end of JSON text in argument to function
- Error 4038: Syntax error in JSON text in argument to function at position
- Error 4039: Incorrect escaping in JSON text in argument to function at position

- Error 4040: Limit on JSON nested structures depth is reached in argument to function at position
- Error 4041: Unexpected end of JSON path in argument to function
- Error 4042: Syntax error in JSON path in argument to function at position
- Error 4043: Limit on JSON path depth is reached in argument to function at position
- Error 4044: Wildcards in JSON path not allowed in argument to function
- Error 4045: JSON path should end with an array identifier in argument to function
- Error 4046: Argument 2 to function must be “one” or “all”.
- Error 4047: CREATE TEMPORARY TABLE is not allowed with ROW_FORMAT=COMPRESSED or KEY_BLOCK_SIZE.
- Error 4048: Incorrect GeoJSON format specified for st_geomfromgeojson function
- Error 4049: Incorrect GeoJSON format - too few points for linestring specified
- Error 4050: Incorrect GeoJSON format - polygon not closed
- Error 4051: Path expression ‘$’ is not allowed in argument to function
- Error 4052: A slave with the same server_uuid/server_id as this slave has connected to the master
- Error 4053: Flashback does not support
- Error 4054: Keys are out order during bulk load
- Error 4055: Bulk load rows overlap existing rows
- Error 4056: Can’t execute updates on master with binlog_format != ROW
- Error 4057: MyRocks supports only READ COMMITTED and REPEATABLE READ isolation levels
- Error 4058: When unique checking is disabled in MyRocks, INSERT,UPDATE,LOAD statements with clauses
- Error 4059: Can’t execute updates when you started a transaction with START TRANSACTION WITH CONSIST
- Error 4060: This transaction was rolled back and cannot be committed. Only supported operation is to
- Error 4061: MyRocks currently does not support ROLLBACK TO SAVEPOINT if modifying rows
- Error 4062: Only REPEATABLE READ isolation level is supported for START TRANSACTION WITH CONSISTENT
- Error 4063: Unsupported collation on string indexed column
- Error 4064: Table ‘%s’ does not exist, but metadata information exists inside MyRocks. This is a sig
- Error 4065: Column family flag is different from an existing flag
- Error 4066: TTL duration in MyRocks must be an unsigned non-null 64-bit integer
- Error 4067: Status error received from RocksDB
- Error 4068: Status error received from RocksDB
- Error 4069: TTL support is currently disabled when table has a hidden PK
- Error 4070: TTL column in MyRocks must be an unsigned non-null 64-bit integer, exist inside the table
- Error 4071: The per-index column family option has been deprecated
- Error 4072: MyRocks failed creating new key definitions during alter
- Error 4073: MyRocks failed populating secondary key during alter
- Error 4074: Window functions can not be used as arguments to group functions
- Error 4075: OK packet too large
- Error 4076: Incorrect GeoJSON format - empty ‘coordinates’ array
- Error 4077: MyRocks doesn’t currently support collations with “No pad” attribute
- Error 4078: Illegal parameter data types for operation
- Error 4079: Illegal parameter data type for operation
- Error 4080: Incorrect parameter count to cursor
- Error 4081: Unknown structured system variable or ROW routine variable
- Error 4082: Row variable does not have a field

- Error 4083: END identifier does not match
- Error 4084: Sequence has run out
- Error 4085: Sequence values are conflicting
- Error 4086: Sequence table structure is invalid
- Error 4087: Sequence access error
- Error 4088: Sequences requires binlog_format mixed or row
- Error 4089: is not a SEQUENCE
- Error 4090: is not a SEQUENCE
- Error 4091: Unknown SEQUENCE
- Error 4092: Unknown VIEW
- Error 4093: Wrong INSERT into a SEQUENCE. One can only do single table INSERT into a sequence object
- Error 4094: At line in
- Error 4095: Subroutine is declared in the package specification but is not defined in the package bo
- Error 4096: Subroutine has a forward declaration but is not defined
- Error 4097: Compressed column can't be used in key specification
- Error 4098: Unknown compression method
- Error 4099: The used table value constructor has a different number of values

This page is licensed: CC BY-SA / Gnu FDL

Error 4000: COM_MULTI can't return a result set in the given context

Error 4000: This error never happens

Error Code	SQLSTATE	Error	Description
4000	0A000	ER_UNUSED_26	This error never happens

Possible Causes and Solutions

Error 4001: Command is not allowed for COM_MULTI

Error 4001: This error never happens

Error Code	SQLSTATE	Error	Description
4001	HY000	ER_UNUSED_27	This error never happens

Possible Causes and Solutions

Error 4002: WITH column list and SELECT field list have different column counts

Error Code	SQLSTATE	Error	Description
4002		ER_WITH_COL_WRONG_LIST	WITH column list and SELECT field list have different column counts

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4003: Too many WITH elements in WITH clause

Error Code	SQLSTATE	Error	Description
4003		ER_TOO_MANY_DEFINITIONS_IN_WITH_CLAUSE	Too many WITH elements in WITH clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4004: Duplicate query name in WITH clause

Error Code	SQLSTATE	Error	Description
4004		ER_DUP_QUERY_NAME	Duplicate query name %`-.64s in WITH clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4005: No anchors for recursive WITH element

Error Code	SQLSTATE	Error	Description
4005		ER_RECURSIVE_WITHOUT_ANCHORS	No anchors for recursive WITH element ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4006: Unacceptable mutual recursion with anchored table

Error Code	SQLSTATE	Error	Description
4006		ER_UNACCEPTABLE_MUTUAL_RECURSION	Unacceptable mutual recursion with anchored table ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4007: Reference to recursive WITH table in materialized derived

Error Code	SQLSTATE	Error	Description
4007		ER_REF_TO_RECURSIVE_WITH_TABLE_IN_DERIVED	Reference to recursive WITH table ‘%s’ in materialized derived

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4008: Restrictions imposed on recursive definitions are violated for table

Error Code	SQLSTATE	Error	Description
4008		ER_NOT_STANDARD_COMPLIANT_RECURSIVE	Restrictions imposed on recursive definitions are violated for table ‘%s’“R_WRONG_WINDOW_SPEC_NAME

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4009: Window specification with name is not defined

Error Code	SQLSTATE	Error	Description
4009		ER_WRONG_WINDOW_SPEC_NAME	Window specification with name ‘%s’ is not defined

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4010: Multiple window specifications with the same name

Error Code	SQLSTATE	Error	Description
4010		ER_DUP_WINDOW_NAME	Multiple window specifications with the same name ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4011: Window specification referencing another one cannot contain partition list

Error Code	SQLSTATE	Error	Description
4011		ER_PARTITION_LIST_IN_REFERENCING_WINDOW_SPEC	Window specification referencing another one ‘%s’ cannot contain partition list

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4012: Referenced window specification already contains order list

Error Code	SQLSTATE	Error	Description
4012		ER_ORDER_LIST_IN_REFERENCING_WINDOW_SPEC	Referenced window specification ‘%s’ already contains order list

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4013: Referenced window specification cannot contain window frame

Error Code	SQLSTATE	Error	Description
4013		ER_WINDOW_FRAME_IN_REFERENCED_WINDOW_SPEC	Referenced window specification ‘%s’ cannot contain window frame

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4014: Unacceptable combination of window frame bound specifications

Error Code	SQLSTATE	Error	Description
4014		ER_BAD_COMBINATION_OF_WINDOW_FRAME_BOUND_SPECS	Unacceptable combination of window frame bound specifications

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4015: Window function is allowed only in SELECT list and ORDER BY clause

Error Code	SQLSTATE	Error	Description
4015		ER_WRONG_PLACEMENT_OF_WINDOW_FUNCTION	Window function is allowed only in SELECT list and ORDER BY clause

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4016: Window function is not allowed in window specification

Error Code	SQLSTATE	Error	Description
4016		ER_WINDOW_FUNCTION_IN_WINDOW_SPEC	Window function is not allowed in window specification

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4017: Window frame is not allowed with

Error Code	SQLSTATE	Error	Description
4017		ER_NOT_ALLOWED_WINDOW_FRAME	Window frame is not allowed with ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4018: No order list in window specification for

Error Code	SQLSTATE	Error	Description
4018		ER_NO_ORDER_LIST_IN_WINDOW_SPEC	No order list in window specification for ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4019: RANGE-type frame requires ORDER BY clause with single sort key

Error Code	SQLSTATE	Error	Description
4019		ER_RANGE_FRAME_NEEDS_SIMPLE_ORDERBY	RANGE-type frame requires ORDER BY clause with single sort key

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4020: Integer is required for ROWS-type frame

Error Code	SQLSTATE	Error	Description
4020		ER_WRONG_TYPE_FOR_ROWS_FRAME	Integer is required for ROWS-type frame

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4021: Numeric datatype is required for RANGE-type frame

Error Code	SQLSTATE	Error	Description
4021		ER_WRONG_TYPE_FOR_RANGE_FRAME	Numeric datatype is required for RANGE-type frame

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4022: Frame exclusion is not supported yet

Error Code	SQLSTATE	Error	Description
4022		ER_FRAME_EXCLUSION_NOT_SUPPORTED	Frame exclusion is not supported yet

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4023: This window function may not have a window frame

Error Code	SQLSTATE	Error	Description
4023		ER_WINDOW_FUNCTION_DONT_HAVE_FRAME	This window function may not have a window frame

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4024: Argument of NTILE must be greater than 0

Error Code	SQLSTATE	Error	Description
4024		ER_INVALID_NTILE_ARGUMENT	Argument of NTILE must be greater than 0

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4025: CONSTRAINT failed for

Error Code	SQLSTATE	Error	Description
4025	23000	ER_CONSTRAINT_FAILED	CONSTRAINT %`s failed for %`-.192s.%`-.192s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4026: Expression in the clause is too big

Error Code	SQLSTATE	Error	Description
4026		ER_EXPRESSION_IS_TOO_BIG	Expression in the %s clause is too big

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4027: Got an error evaluating stored expression

Error Code	SQLSTATE	Error	Description
4027		ER_ERROR_EVALUATING_EXPRESSION	Got an error evaluating stored expression %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4028: Got an error when calculating default value for

Error Code	SQLSTATE	Error	Description
4028		ER_CALCULATING_DEFAULT_VALUE	Got an error when calculating default value for %`s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4029: Expression for field is referring to uninitialized field

Error Code	SQLSTATE	Error	Description
4029		ER_EXPRESSION_REFERS_TO_UNINIT_FIELD	Expression for field % `-.64s is referring to uninitialized field %` s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4030: Only one DEFAULT partition allowed

Error Code	SQLSTATE	Error	Description
4030		ER_PARTITION_DEFAULT_ERROR	Only one DEFAULT partition allowed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4031: Referenced trigger for the given action time and event type does not exist

Error Code	SQLSTATE	Error	Description
4031		ER_REFERENCED_TRG_DOES_NOT_EXIST	Referenced trigger ‘%s’ for the given action time and event type does not exist

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4032: Default/ignore value is not supported for such parameter usage

Error Code	SQLSTATE	Error	Description
4032		ER_INVALID_DEFAULT_PARAM	Default/ignore value is not supported for such parameter usage

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4033: Only row based replication supported for bulk operations

Error Code	SQLSTATE	Error	Description
4033		ER_BINLOG_NON_SUPPORTED_BULK	Only row based replication supported for bulk operations

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4034: Uncompress the compressed binlog failed

Error Code	SQLSTATE	Error	Description
4034		ER_BINLOG_UNCOMPRESS_ERROR	Uncompress the compressed binlog failed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4035: Broken JSON string in argument to function at position

Error Code	SQLSTATE	Error	Description
4035		ER_JSON_BAD_CHR	Broken JSON string in argument %d to function ‘%s’ at position %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4036: Character disallowed in JSON in argument to function at position

Error Code	SQLSTATE	Error	Description
4036		ER_JSON_NOT_JSON_CHR	Character disallowed in JSON in argument %d to function ‘%s’ at position %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4037: Unexpected end of JSON text in argument to function

Error Code	SQLSTATE	Error	Description
4037		ER_JSON_EOS	Unexpected end of JSON text in argument %d to function '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4038: Syntax error in JSON text in argument to function at position

Error Code	SQLSTATE	Error	Description
4038		ER_JSON_SYNTAX	Syntax error in JSON text in argument %d to function '%s' at position %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4039: Incorrect escaping in JSON text in argument to function at position

Error Code	SQLSTATE	Error	Description
4039		ER_JSON_ESCAPING	Incorrect escaping in JSON text in argument %d to function '%s' at position %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4040: Limit on JSON nested structures depth is reached in argument to function at position

Error Code	SQLSTATE	Error	Description
4040		ER_JSON_DEPTH	Limit of %d on JSON nested structures depth is reached in argument %d to function '%s' at position %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4041: Unexpected end of JSON path in argument to function

Error Code	SQLSTATE	Error	Description
4041		ER_JSON_PATH_EOS	Unexpected end of JSON path in argument %d to function '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4042: Syntax error in JSON path in argument to function at position

Error Code	SQLSTATE	Error	Description
4042		ER_JSON_PATH_SYNTAX	Syntax error in JSON path in argument %d to function '%s' at position %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4043: Limit on JSON path depth is reached in argument to function at position

Error Code	SQLSTATE	Error	Description
4043		ER_JSON_PATH_DEPTH	Limit of %d on JSON path depth is reached in argument %d to function ‘%s’ at position %d

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4044: Wildcards in JSON path not allowed in argument to function

Error Code	SQLSTATE	Error	Description
4044		ER_JSON_PATH_NO_WILDCARD	Wildcards in JSON path not allowed in argument %d to function ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4045: JSON path should end with an array identifier in argument to function

Error Code	SQLSTATE	Error	Description
4045		ER_JSON_PATH_ARRAY	JSON path should end with an array identifier in argument %d to function ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4046: Argument 2 to function must be “one” or “all”.

Error Code	SQLSTATE	Error	Description
4046		ER_JSON_ONE_OR_ALL	Argument 2 to function ‘%s’ must be “one” or “all”.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4047: CREATE TEMPORARY TABLE is not allowed with ROW_FORMAT=COMPRESSED or KEY_BLOCK_SIZE.

Error Code	SQLSTATE	Error	Description
4047		ER_UNSUPPORT_COMPRESSED_TEMPORARY_TABLE	CREATE TEMPORARY TABLE is not allowed with ROW_FORMAT=COMPRESSED or KEY_BLOCK_SIZE.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4048: Incorrect GeoJSON format specified for st_geomfromgeojson function

Error Code	SQLSTATE	Error	Description
4048		ER_GEOJSON_INCORRECT	Incorrect GeoJSON format specified for st_geomfromgeojson function.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4049: Incorrect GeoJSON format - too few points for linestring specified

Error Code	SQLSTATE	Error	Description
4049		ER_GEOJSON_TOO_FEW_POINTS	Incorrect GeoJSON format - too few points for linestring specified.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4050: Incorrect GeoJSON format - polygon not closed

Error Code	SQLSTATE	Error	Description
4050		ER_GEOJSON_NOT_CLOSED	Incorrect GeoJSON format - polygon not closed.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4051: Path expression '$' is not allowed in argument to function

Error Code	SQLSTATE	Error	Description
4051		ER_JSON_PATH_EMPTY	Path expression '$' is not allowed in argument %d to function '%s'.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4052: A slave with the same server_uuid/server_id as this slave has connected to the master

Error Code	SQLSTATE	Error	Description
4052		ER_SLAVE_SAME_ID	A slave with the same server_uuid/server_id as this slave has connected to the master

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4053: Flashback does not support

Error Code	SQLSTATE	Error	Description
4053		ER_FLASHBACK_NOT_SUPPORTED	Flashback does not support %s %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4054: Keys are out order during bulk load

Error Code	SQLSTATE	Error	Description
4054		ER_KEYS_OUT_OF_ORDER	Keys are out order during bulk load

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4055: Bulk load rows overlap existing rows

Error Code	SQLSTATE	Error	Description
4055		ER_OVERLAPPING_KEYS	Bulk load rows overlap existing rows

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4056: Can’t execute updates on master with binlog_format != ROW

Error Code	SQLSTATE	Error	Description
4056		ER_REQUIRE_ROW_BINLOG_FORMAT	Can’t execute updates on master with binlog_format != ROW.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4057: MyRocks supports only READ COMMITTED and REPEATABLE READ isolation levels

Error Code	SQLSTATE	Error	Description
4057		ER_ISOLATION_MODE_NOT_SUPPORTED	MyRocks supports only READ COMMITTED and REPEATABLE READ isolation levels. Please change from current isolation level %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4058: When unique checking is disabled in MyRocks, INSERT,UPDATE,LOAD statements with clauses

Error Code	SQLSTATE	Error	Description
4058		ER_ON_DUPLICATE_DISABLED	When unique checking is disabled in MyRocks, INSERT,UPDATE,LOAD statements with clauses that update or replace the key (i.e. INSERT ON DUPLICATE KEY UPDATE, REPLACE) are not allowed. Query: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4059: Can’t execute updates when you started a transaction with START TRANSACTION WITH CONSIST

Error 4059: Can’t execute updates when you started a transaction with START TRANSACTION WITH CONSISTENT [ROCKSDB] SNAPSHOT

Error Code	SQLSTATE	Error	Description
4059		ER_UPDATES_WITH_CONSISTENT_SNAPSHOT	Can’t execute updates when you started a transaction with START TRANSACTION WITH CONSISTENT [ROCKSDB] SNAPSHOT.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4060: This transaction was rolled back and cannot be committed. Only supported operation is to

Error Code	SQLSTATE	Error	Description
4060		ER_ROLLBACK_ONLY	This transaction was rolled back and cannot be committed. Only supported operation is to roll it back, so all pending changes will be discarded. Please restart another transaction.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4061: MyRocks currently does not support ROLLBACK TO SAVEPOINT if modifying rows

Error Code	SQLSTATE	Error	Description
4061		ER_ROLLBACK_TO_SAVEPOINT	MyRocks currently does not support ROLLBACK TO SAVEPOINT if modifying rows.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4062: Only REPEATABLE READ isolation level is supported for START TRANSACTION WITH CONSISTENT

Error Code	SQLSTATE	Error	Description
4062		ER_ISOLATION_LEVEL_WITH_CONSISTENT_SNAPSHOT	Only REPEATABLE READ isolation level is supported for START TRANSACTION WITH CONSISTENT SNAPSHOT in RocksDB Storage Engine.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4063: Unsupported collation on string indexed column

Error Code	SQLSTATE	Error	Description
4063		ER_UNSUPPORTED_COLLATION	Unsupported collation on string indexed column %s.%s Use binary collation (%s).

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4064: Table ‘%s’ does not exist, but metadata information exists inside MyRocks. This is a sig

Error Code	SQLSTATE	Error	Description
4064		ER_METADATA_INCONSISTENCY	Table ‘%s’ does not exist, but metadata information exists inside MyRocks. This is a sign of data inconsistency. Please check if ‘%s.frm’ exists, and try to restore it if it does not exist.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4065: Column family flag is different from an existing flag

Error Code	SQLSTATE	Error	Description
4065		ER_CF_DIFFERENT	Column family (‘%s’) flag (%d) is different from an existing flag (%d). Assign a new CF flag, or do not change existing CF flag.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4066: TTL duration in MyRocks must be an unsigned non-null 64-bit integer

Error Code	SQLSTATE	Error	Description
4066		ER_RDB_TTL_DURATION_FORMAT	TTL duration (%s) in MyRocks must be an unsigned non-null 64-bit integer.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4067: Status error received from RocksDB

Error Code	SQLSTATE	Error	Description
4067		ER_RDB_STATUS_GENERAL	Status error %d received from RocksDB: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4068: Status error received from RocksDB

Error Code	SQLSTATE	Error	Description
4068		ER_RDB_STATUS_MSG	%s, Status error %d received from RocksDB: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4069: TTL support is currently disabled when table has a hidden PK

Error Code	SQLSTATE	Error	Description
4069		ER_RDB_TTL_UNSUPPORTED	TTL support is currently disabled when table has a hidden PK.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4070: TTL column in MyRocks must be an unsigned non-null 64-bit integer, exist inside the table

Error Code	SQLSTATE	Error	Description
4070		ER_RDB_TTL_COL_FORMAT	TTL column (%s) in MyRocks must be an unsigned non-null 64-bit integer, exist inside the table, and have an accompanying ttl duration.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4071: The per-index column family option has been deprecated

Error Code	SQLSTATE	Error	Description
4071		ER_PER_INDEX_CF_DEPRECATED	The per-index column family option has been deprecated

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4072: MyRocks failed creating new key definitions during alter

Error Code	SQLSTATE	Error	Description
4072		ER_KEY_CREATE_DURING_ALTER	MyRocks failed creating new key definitions during alter.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4073: MyRocks failed populating secondary key during alter

Error Code	SQLSTATE	Error	Description
4073		ER_SK_POPULATE_DURING_ALTER	MyRocks failed populating secondary key during alter.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4074: Window functions can not be used as arguments to group functions

Error Code	SQLSTATE	Error	Description
4074		ER_SUM_FUNC_WITH_WINDOW_FUNC_AS_ARG	Window functions can not be used as arguments to group functions.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4075: OK packet too large

Error Code	SQLSTATE	Error	Description
4075		ER_NET_OK_PACKET_TOO_LARGE	OK packet too large

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4076: Incorrect GeoJSON format - empty ‘coordinates’ array

Error Code	SQLSTATE	Error	Description
4076		ER_GEOJSON_EMPTY_COORDINATES	Incorrect GeoJSON format - empty ‘coordinates’ array.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4077: MyRocks doesn’t currently support collations with “No pad” attribute

Error Code	SQLSTATE	Error	Description
4077		ER_MYROCKS_CANT_NOPAD_COLLATION	MyRocks doesn’t currently support collations with “No pad” attribute.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4078: Illegal parameter data types for operation

Error Code	SQLSTATE	Error	Description
4078		ER_ILLEGAL_PARAMETER_DATA_TYPES2_FOR_OPERATION	Illegal parameter data types %s and %s for operation ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4079: Illegal parameter data type for operation

Error Code	SQLSTATE	Error	Description
4079		ER_ILLEGAL_PARAMETER_DATA_TYPE_FOR_OPERATION	Illegal parameter data type %s for operation ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4080: Incorrect parameter count to cursor

Error Code	SQLSTATE	Error	Description
4080		ER_WRONG_PARAMCOUNT_TO_CURSOR	Incorrect parameter count to cursor ‘%-.192s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4081: Unknown structured system variable or ROW routine variable

Error Code	SQLSTATE	Error	Description
4081		ER_UNKNOWN_STRUCTURED_VARIABLE	Unknown structured system variable or ROW routine variable '%-.*s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4082: Row variable does not have a field

Error Code	SQLSTATE	Error	Description
4082		ER_ROW_VARIABLE_DOES_NOT_HAVE_FIELD	Row variable '%-.192s' does not have a field '%-.192s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4083: END identifier does not match

Error Code	SQLSTATE	Error	Description
4083		ER_END_IDENTIFIER_DOES_NOT_MATCH	END identifier '%-.192s' does not match '%-.192s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4084: Sequence has run out

Error Code	SQLSTATE	Error	Description
4084		ER_SEQUENCE_RUN_OUT	Sequence '%-.64s.%-.64s' has run out

Possible Causes and Solutions

Sequence has Completed Normally

In a sequence, once the MAXVALUE has been reached, and the CYCLE argument has not been used, further attempts to set the next value will fail:

```
CREATE OR REPLACE SEQUENCE s START WITH 1 INCREMENT BY 1 MAXVALUE=2;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          1 |
+------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          2 |
+------------+

SELECT NEXTVAL(s);
ERROR 4084 (HY000): Sequence 'test.s' has run out
```

This is usually intended behaviour, but there are a number of ways to prevent this:

Choose a larger MAXVALUE

Prior to MariaDB 11.5, this defaulted to the maximum, BIGINT. From MariaDB 11.5, it can be BIGINT UNSIGNED

```
CREATE OR REPLACE SEQUENCE s AS BIGINT UNSIGNED START WITH 1 INCREMENT BY 1;

SELECT SEQUENCE_NAME,MAXIMUM_VALUE FROM INFORMATION_SCHEMA.SEQUENCES\G
*************************** 1. row ***************************
SEQUENCE_NAME: s
MAXIMUM_VALUE: 18446744073709551614
```

Cycling

If reuse sequence values is not a problem, you can cycle back to the start once the maximum value has been reached:

```
CREATE OR REPLACE SEQUENCE s START WITH 1 INCREMENT BY 1 MAXVALUE=2 CYCLE;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          1 |
+------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          2 |
+------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          1 |
+------------+
```

Sequence has Completed Due to Cache

The underlying table is only updated with a new sequence value the first time the value is created, and then each time the cache expires. By default the cache is set to `1000`. For example:

```
CREATE OR REPLACE SEQUENCE s START WITH 1 INCREMENT BY 1 MAXVALUE=10 CACHE=5;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          1 |
+------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          2 |
+------------+

FLUSH TABLES s;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          6 |
+------------+

FLUSH TABLES s;

SELECT NEXTVAL(s);
ERROR 4084 (HY000): Sequence 'test.s' has run out
```

This behaviour can be prevented by setting the cache to zero, so that the value stored in the underlying table is written each time:

```
CREATE OR REPLACE SEQUENCE s START WITH 1 INCREMENT BY 1 MAXVALUE=10 CACHE=0;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          1 |
+------------+

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          2 |
+------------+

FLUSH TABLES s;

SELECT NEXTVAL(s);
+------------+
| NEXTVAL(s) |
+------------+
|          3 |
+------------+
```

See Also

- Sequence Overview
- CREATE SEQUENCE
- ALTER SEQUENCE
- DROP SEQUENCE

- NEXT VALUE FOR
- PREVIOUS VALUE FOR
- SETVAL(). Set next value for the sequence.
- AUTO INCREMENT
- SHOW CREATE SEQUENCE
- Information Schema SEQUENCES Table

This page is licensed: CC BY-SA / Gnu FDL

Error 4085: Sequence values are conflicting

Error 4085: Sequence Has Out of Range Value for Options

Error Code	SQLSTATE	Error	Description
4085		ER_SEQUENCE_INVALID_DATA	Sequence '%-.64s.%-.64s' has out of range value for options

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4086: Sequence table structure is invalid

Error Code	SQLSTATE	Error	Description
4086		ER_SEQUENCE_INVALID_TABLE_STRUCTURE	Sequence '%-.64s.%-.64s' table structure is invalid (%s)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4087: Sequence access error

Error Code	SQLSTATE	Error	Description
4087		ER_SEQUENCE_ACCESS_ERROR	Sequence '%-.64s.%-.64s' access error

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4088: Sequences requires binlog_format mixed or row

Error Code	SQLSTATE	Error	Description
4088		ER_SEQUENCE_BINLOG_FORMAT	Sequences requires binlog_format mixed or row

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4089: is not a SEQUENCE

Error Code	SQLSTATE	Error	Description
4089		ER_NOT_SEQUENCE	'%-.64s.%-.64s' is not a SEQUENCE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4090: is not a SEQUENCE

Error Code	SQLSTATE	Error	Description
4090		ER_NOT_SEQUENCE2	'%-.192s' is not a SEQUENCE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4091: Unknown SEQUENCE

Error Code	SQLSTATE	Error	Description
4091		ER_UNKNOWN_SEQUENCES	Unknown SEQUENCE: '%-.300s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4092: Unknown VIEW

Error Code	SQLSTATE	Error	Description
4092		ER_UNKNOWN_VIEW	Unknown VIEW: '%-.300s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4093: Wrong INSERT into a SEQUENCE. One can only do single table INSERT into a sequence object

Error Code	SQLSTATE	Error	Description
4093		ER_WRONG_INSERT_INTO_SEQUENCE	Wrong INSERT into a SEQUENCE. One can only do single table INSERT into a sequence object (like with mysqldump). If you want to change the SEQUENCE, use ALTER SEQUENCE instead.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4094: At line in

Error Code	SQLSTATE	Error	Description
4094		ER_SP_STACK_TRACE	At line %u in %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4095: Subroutine is declared in the package specification but is not defined in the package bo

Error Code	SQLSTATE	Error	Description
4095		ER_PACKAGE_ROUTINE_IN_SPEC_NOT_DEFINED_IN_BODY	Subroutine '%-.192s' is declared in the package specification but is not defined in the package body

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4096: Subroutine has a forward declaration but is not defined

Error Code	SQLSTATE	Error	Description
4096		ER_PACKAGE_ROUTINE_FORWARD_DECLARATION_NOT_DEFINED	Subroutine '%-.192s' has a forward declaration but is not defined

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4097: Compressed column can't be used in key specification

Error Code	SQLSTATE	Error	Description
4097		ER_COMPRESSED_COLUMN_USED_AS_KEY	Compressed column '%-.192s' can't be used in key specification

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4098: Unknown compression method

Error Code	SQLSTATE	Error	Description
4098		ER_UNKNOWN_COMPRESSION_METHOD	Unknown compression method: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4099: The used table value constructor has a different number of values

Error Code	SQLSTATE	Error	Description
4099		ER_WRONG_NUMBER_OF_VALUES_IN_TVC	The used table value constructor has a different number of values

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 4100 to 4199

- Error 4100: Field reference can't be used in table value constructor
- Error 4101: Numeric datatype is required for function

- Error 4102: Argument to the function is not a constant for a partition
- Error 4104: function only accepts arguments that can be converted to numerical types
- Error 4105: Aggregate specific instruction (FETCH GROUP NEXT ROW) used in a wrong context
- Error 4106: Aggregate specific instruction(FETCH GROUP NEXT ROW) missing from the aggregate function
- Error 4107: Limit only accepts integer values
- Error 4108: Invisible column must have a default value
- Error 4109: Rows matched: Changed: Inserted: Warnings:
- Error 4110: must be of type for system-versioned table
- Error 4111: Transaction-precise system versioning for is not supported
- Error 4112: You should never see it
- Error 4113: Wrong partitioning type, expected type
- Error 4114: Versioned table last HISTORY partition is out of, need more HISTORY partitions
- Error 4115: Maybe missing parameters
- Error 4116: Can only drop oldest partitions when rotating by INTERVAL
- Error 4117: You should never see it
- Error 4118: Partition contains non-historical data
- Error 4119: Not allowed for system-versioned. Change @@system_versioning_alter_history to proceed wi
- Error 4120: Not allowed for system-versioned. Change to/from native system versioning engine is not
- Error 4121: SYSTEM_TIME range selector is not allowed
- Error 4122: Conflicting FOR SYSTEM_TIME clauses in WITH RECURSIVE
- Error 4123: Table must have at least one versioned column
- Error 4124: Table is not system-versioned
- Error 4125: Wrong parameters for: missing
- Error 4126: PERIOD FOR SYSTEM_TIME must use columns
- Error 4127: Wrong parameters for partitioned: wrong value for
- Error 4128: Wrong partitions for: must have at least one HISTORY and exactly one last CURRENT
- Error 4129: TRX_ID not found in mysql.transaction_registry
- Error 4130: Can not change system versioning field
- Error 4131: Can not DROP SYSTEM VERSIONING for table partitioned BY SYSTEM_TIME
- Error 4132: System-versioned tables in the database are not supported
- Error 4133: Transaction registry is disabled
- Error 4134: Duplicate ROW column
- Error 4135: Table is already system-versioned
- Error 4136: You should never see it
- Error 4137: System-versioned tables do not support
- Error 4138: Transaction-precise system-versioned tables do not support partitioning by ROW START or
- Error 4139: The index file for table is full
- Error 4140: The column cannot be changed more than once in a single UPDATE statement
- Error 4141: Row with no elements is not allowed in table value constructor in this context
- Error 4142: SYSTEM_TIME partitions in table does not support historical query
- Error 4143: index does not support this operation
- Error 4144: Changing table options requires the table to be rebuilt
- Error 4145: Can’t execute the command as you have a BACKUP STAGE active
- Error 4146: You must start backup with “BACKUP STAGE START”
- Error 4147: Backup stage is same or before current backup stage

- Error 4148: Backup stage failed
- Error 4149: Unknown backup stage: Stage should be one of START, FLUSH, BLOCK_DDL, BLOCK_COMMIT or EN
- Error 4150: User is blocked because of too many credential errors; unblock with ‘FLUSH PRIVILEGES’
- Error 4151: Access denied, this account is locked
- Error 4152: Application-time period table cannot be temporary
- Error 4153: Fields of PERIOD FOR have different types
- Error 4154: Cannot specify more than one application-time period
- Error 4155: Period field cannot be
- Error 4156: Period is not found in table
- Error 4157: Column used in period specified in update SET list
- Error 4158: Can’t DROP CONSTRAINT. Use DROP PERIOD for this
- Error 4159: Specified key part was too long; max key part length is
- Error 4160: Comment for database is too long
- Error 4161: Unknown data type
- Error 4162: Operator does not exists
- Error 4163: Table history row start is later than row end
- Error 4164: STARTS is later than query time, first history partition may exceed INTERVAL value
- Error 4165: DDL-statement is forbidden as table storage engine does not support Galera replication
- Error 4166: The used command is not allowed because the MariaDB server or client has disabled the lo
- Error 4167: No secure transports are configured, unable to set –require_secure_transport=ON
- Error 4168: Slave SQL thread ignored the ‘%s’ because table is shared
- Error 4169: AUTO_INCREMENT column cannot be used in the UNIQUE index
- Error 4170: Key cannot explicitly include column
- Error 4171: Key cannot have WITHOUT OVERLAPS
- Error 4172: is not allowed in this context
- Error 4173: Engine does not support rollback. Changes where committed during rollback call
- Error 4174: A primary key cannot be marked as IGNORE
- Error 4175: SKIP LOCKED makes this statement unsafe
- Error 4176: Field can’t be set for JSON_TABLE
- Error 4177: Every table function must have an alias
- Error 4178: Can’t store an array or an object in the scalar column of JSON_TABLE
- Error 4179: Can’t store multiple matches of the path in the column of JSON_TABLE
- Error 4180: FETCH … WITH TIES requires ORDER BY clause to be present
- Error 4181: Dropped orphan trigger, originally created for table
- Error 4182: Storage engine is disabled
- Error 4183: SFORMAT error
- Error 4184: Convert partition is not supported for subpartitioned table
- Error 4185: MariaDB tried to use the %s, but its provider plugin is not loaded
- Error 4186: Failed to parse histogram for table at offset
- Error 4187: OUT or INOUT argument for function is not allowed here
- Error 4188: Replicated query table can not be temporary
- Error 4189: Versioned table: adding HISTORY partition(s) failed
- Error 4190: is implicitly changing the value of from to
- Error 4191: CHANGE MASTER TO option is missing requirement

- Error 4192: Slave log event execution was interrupted (slave_max_statement_time exceeded)
- Error 4193: Invalid value for keyword
- Error 4194: keyword is not supported
- Error 4195: Variable schema is not supported
- Error 4196: Pseudo thread id should not be modified by the client as it will be overwritten
- Error 4197: Wrong number of columns
- Error 4198: Sequence tables cannot have any keys
- Error 4199: Sequence tables cannot have any constraints

This page is licensed: CC BY-SA / Gnu FDL

Error 4100: Field reference can't be used in table value constructor

Error Code	SQLSTATE	Error	Description
4100		ER_FIELD_REFERENCE_IN_TVC	Field reference '%-.192s' can't be used in table value constructor

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4101: Numeric datatype is required for function

Error Code	SQLSTATE	Error	Description
4101		ER_WRONG_TYPE_FOR_PERCENTILE_FUNC	Numeric datatype is required for %s function

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4102: Argument to the function is not a constant for a partition

Error Code	SQLSTATE	Error	Description
4102		ER_ARGUMENT_NOT_CONSTANT	Argument to the %s function is not a constant for a partition

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4103: Argument to the function does not belong to the range [0,1]

Error Code	SQLSTATE	Error	Description
4103		ER_ARGUMENT_OUT_OF_RANGE	Argument to the %s function does not belong to the range [0,1]

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4104: function only accepts arguments that can be converted to numerical types

Error Code	SQLSTATE	Error	Description
4104		ER_WRONG_TYPE_OF_ARGUMENT	%s function only accepts arguments that can be converted to numerical types

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4105: Aggregate specific instruction (FETCH GROUP NEXT ROW) used in a wrong context

Error Code	SQLSTATE	Error	Description
4105		ER_NOT_AGGREGATE_FUNCTION	Aggregate specific instruction (FETCH GROUP NEXT ROW) used in a wrong context

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4106: Aggregate specific instruction(FETCH GROUP NEXT ROW) missing from the aggregate function

Error Code	SQLSTATE	Error	Description
4106		ER_INVALID_AGGREGATE_FUNCTION	Aggregate specific instruction(FETCH GROUP NEXT ROW) missing from the aggregate function

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4107: Limit only accepts integer values

Error Code	SQLSTATE	Error	Description
4107		ER_INVALID_VALUE_TO_LIMIT	Limit only accepts integer values

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4108: Invisible column must have a default value

Error Code	SQLSTATE	Error	Description
4108		ER_INVISIBLE_NOT_NULL_WITHOUT_DEFAULT	Invisible column %`s must have a default value

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4109: Rows matched: Changed: Inserted: Warnings:

Error Code	SQLSTATE	Error	Description
4109		ER_UPDATE_INFO_WITH_SYSTEM_VERSIONING	Rows matched: %ld Changed: %ld Inserted: %ld Warnings: %ld

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4110: must be of type for system-versioned table

Error Code	SQLSTATE	Error	Description
4110		ER_VERS_FIELD_WRONG_TYPE	% `s must be of type %s for system-versioned table %` s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4111: Transaction-precise system versioning for is not supported

Error Code	SQLSTATE	Error	Description
4111		ER_VERS_ENGINE_UNSUPPORTED	Transaction-precise system versioning for %`s is not supported

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4112: You should never see it

Error Code	SQLSTATE	Error	Description
4112		ER_UNUSED_23	You should never see it

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4113: Wrong partitioning type, expected type

Error Code	SQLSTATE	Error	Description
4113		ER_PARTITION_WRONG_TYPE	Wrong partitioning type, expected type: %`s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4114: Versioned table last HISTORY partition is out of, need more HISTORY partitions

Error Code	SQLSTATE	Error	Description
4114		WARN_VERS_PART_FULL	Versioned table % `s.%` s: last HISTORY partition (%`s) is out of %s, need more HISTORY partitions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4115: Maybe missing parameters

Error Code	SQLSTATE	Error	Description
4115		WARN_VERS_PARAMETERS	Maybe missing parameters: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4116: Can only drop oldest partitions when rotating by INTERVAL

Error Code	SQLSTATE	Error	Description
4116		ER_VERS_DROP_PARTITION_INTERVAL	Can only drop oldest partitions when rotating by INTERVAL

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4117: You should never see it

Error Code	SQLSTATE	Error	Description
4117	HY000	ER_UNUSED_25	You should never see it

Possible Causes and Solutions

Error 4118: Partition contains non-historical data

Error Code	SQLSTATE	Error	Description
4118		WARN_VERS_PART_NON_HISTORICAL	Partition %`s contains non-historical data

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4119: Not allowed for system-versioned. Change @@system_versioning_alter_history to proceed wi

Error Code	SQLSTATE	Error	Description
4119		ER_VERS_ALTER_NOT_ALLOWED	Not allowed for system-versioned % `s.%` s. Change @@system_versioning_alter_history to proceed with ALTER.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4120: Not allowed for system-versioned. Change to/from native system versioning engine is not

Error Code	SQLSTATE	Error	Description
4120		ER_VERS_ALTER_ENGINE_PROHIBITED	Not allowed for system-versioned % `s.%` s. Change to/from native system versioning engine is not supported.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4121: SYSTEM_TIME range selector is not allowed

Error Code	SQLSTATE	Error	Description
4121		ER_VERS_RANGE_PROHIBITED	SYSTEM_TIME range selector is not allowed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4122: Conflicting FOR SYSTEM_TIME clauses in WITH RECURSIVE

Error Code	SQLSTATE	Error	Description
4122		ER_CONFLICTING_FOR_SYSTEM_TIME	Conflicting FOR SYSTEM_TIME clauses in WITH RECURSIVE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4123: Table must have at least one versioned column

Error Code	SQLSTATE	Error	Description
4123		ER_VERS_TABLE_MUST_HAVE_COLUMNS	Table %`s must have at least one versioned column

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4124: Table is not system-versioned

Error Code	SQLSTATE	Error	Description
4124		ER_VERS_NOT_VERSIONED	Table %`s is not system-versioned

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4125: Wrong parameters for: missing

Error Code	SQLSTATE	Error	Description
4125		ER_MISSING	Wrong parameters for %`s: missing '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4126: PERIOD FOR SYSTEM_TIME must use columns

Error Code	SQLSTATE	Error	Description
4126		ER_VERS_PERIOD_COLUMNS	PERIOD FOR SYSTEM_TIME must use columns %`s and %`s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4127: Wrong parameters for partitioned: wrong value for

Error Code	SQLSTATE	Error	Description
4127		ER_PART_WRONG_VALUE	Wrong parameters for partitioned %`s: wrong value for '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4128: Wrong partitions for: must have at least one HISTORY and exactly one last CURRENT

Error Code	SQLSTATE	Error	Description
4128		ER_VERS_WRONG_PARTS	Wrong partitions for %`s: must have at least one HISTORY and exactly one last CURRENT

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4129: TRX_ID not found in mysql.transaction_registry

Error Code	SQLSTATE	Error	Description
4129		ER_VERS_NO_TRX_ID	TRX_ID %llu not found in `mysql.transaction_registry`

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4130: Can not change system versioning field

Error Code	SQLSTATE	Error	Description
4130		ER_VERS_ALTER_SYSTEM_FIELD	Can not change system versioning field %`s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4131: Can not DROP SYSTEM VERSIONING for table partitioned BY SYSTEM_TIME

Error Code	SQLSTATE	Error	Description
4131		ER_DROP_VERSIONING_SYSTEM_TIME_PARTITION	Can not DROP SYSTEM VERSIONING for table %`s partitioned BY SYSTEM_TIME

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4132: System-versioned tables in the database are not supported

Error Code	SQLSTATE	Error	Description
4132		ER_VERS_DB_NOT_SUPPORTED	System-versioned tables in the %`s database are not supported

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4133: Transaction registry is disabled

Error Code	SQLSTATE	Error	Description
4133		ER_VERS_TRT_IS_DISABLED	Transaction registry is disabled

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4134: Duplicate ROW column

Error Code	SQLSTATE	Error	Description
4134		ER_VERS_DUPLICATE_ROW_START_END	Duplicate ROW %s column %`s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4135: Table is already system-versioned

Error Code	SQLSTATE	Error	Description
4135		ER_VERS_ALREADY_VERSIONED	Table %`s is already system-versioned

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4136: You should never see it

Error Code	SQLSTATE	Error	Description
4136	HY000	ER_UNUSED_24	You should never see it

Possible Causes and Solutions

Error 4137: System-versioned tables do not support

Error Code	SQLSTATE	Error	Description
4137		ER_VERS_NOT_SUPPORTED	System-versioned tables do not support %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4138: Transaction-precise system-versioned tables do not support partitioning by ROW START or

Error Code	SQLSTATE	Error	Description
4138		ER_VERS_TRX_PART_HISTORIC_ROW_NOT_SUPPORTED	Transaction-precise system-versioned tables do not support partitioning by ROW START or ROW END

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4139: The index file for table is full

Error Code	SQLSTATE	Error	Description
4139		ER_INDEX_FILE_FULL	The index file for table '%-.192s' is full

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4140: The column cannot be changed more than once in a single UPDATE statement

Error Code	SQLSTATE	Error	Description
4140		ER_UPDATED_COLUMN_ONLY_ONCE	The column % `s.%` s cannot be changed more than once in a single UPDATE statement

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4141: Row with no elements is not allowed in table value constructor in this context

Error Code	SQLSTATE	Error	Description
4141		ER_EMPTY_ROW_IN_TVC	Row with no elements is not allowed in table value constructor in this context

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4142: SYSTEM_TIME partitions in table does not support historical query

Error Code	SQLSTATE	Error	Description
4142		ER_VERS_QUERY_IN_PARTITION	SYSTEM_TIME partitions in table %`s does not support historical query

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4143: index does not support this operation

Error Code	SQLSTATE	Error	Description
4143		ER_KEY_DOESNT_SUPPORT	%s index %`s does not support this operation

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4144: Changing table options requires the table to be rebuilt

Error Code	SQLSTATE	Error	Description
4144		ER_ALTER_OPERATION_TABLE_OPTIONS_NEED_REBUILD	Changing table options requires the table to be rebuilt

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4145: Can’t execute the command as you have a BACKUP STAGE active

Error Code	SQLSTATE	Error	Description
4145		ER_BACKUP_LOCK_IS_ACTIVE	Can’t execute the command as you have a BACKUP STAGE active

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4146: You must start backup with “BACKUP STAGE START”

Error Code	SQLSTATE	Error	Description
4146		ER_BACKUP_NOT_RUNNING	You must start backup with “BACKUP STAGE START”

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4147: Backup stage is same or before current backup stage

Error Code	SQLSTATE	Error	Description
4147		ER_BACKUP_WRONG_STAGE	Backup stage ‘%s’ is same or before current backup stage ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4148: Backup stage failed

Error Code	SQLSTATE	Error	Description
4148		ER_BACKUP_STAGE_FAILED	Backup stage ‘%s’ failed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4149: Unknown backup stage: Stage should be one of START, FLUSH, BLOCK_DDL, BLOCK_COMMIT or EN

Error Code	SQLSTATE	Error	Description
4149		ER_BACKUP_UNKNOWN_STAGE	Unknown backup stage: ‘%s’. Stage should be one of START, FLUSH, BLOCK_DDL, BLOCK_COMMIT or END

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4150: User is blocked because of too many credential errors; unblock with ‘FLUSH PRIVILEGES’

Error Code	SQLSTATE	Error	Description
4150		ER_USER_IS_BLOCKED	User is blocked because of too many credential errors; unblock with ‘FLUSH PRIVILEGES’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4151: Access denied, this account is locked

Error Code	SQLSTATE	Error	Description
4151		ER_ACCOUNT_HAS_BEEN_LOCKED	Access denied, this account is locked

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4152: Application-time period table cannot be temporary

Error Code	SQLSTATE	Error	Description
4152		ER_PERIOD_TEMPORARY_NOT_ALLOWED	Application-time period table cannot be temporary

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4153: Fields of PERIOD FOR have different types

Error Code	SQLSTATE	Error	Description
4153		ER_PERIOD_TYPES_MISMATCH	Fields of PERIOD FOR %`s have different types

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4154: Cannot specify more than one application-time period

Error Code	SQLSTATE	Error	Description
4154		ER_MORE_THAN_ONE_PERIOD	Cannot specify more than one application-time period

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4155: Period field cannot be

Error Code	SQLSTATE	Error	Description
4155		ER_PERIOD_FIELD_WRONG_ATTRIBUTES	Period field %`s cannot be %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4156: Period is not found in table

Error Code	SQLSTATE	Error	Description
4156		ER_PERIOD_NOT_FOUND	Period %`s is not found in table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4157: Column used in period specified in update SET list

Error Code	SQLSTATE	Error	Description
4157		ER_PERIOD_COLUMNS_UPDATED	Column %`s used in period %`s specified in update SET list

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4158: Can’t DROP CONSTRAINT. Use DROP PERIOD for this

Error Code	SQLSTATE	Error	Description
4158		ER_PERIOD_CONSTRAINT_DROP	Can’t DROP CONSTRAINT `%s`. Use DROP PERIOD `%s` for this

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4159: Specified key part was too long; max key part length is

Error Code	SQLSTATE	Error	Description
4159	42000 S1009	ER_TOO_LONG_KEYPART	Specified key part was too long; max key part length is %u bytes

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4160: Comment for database is too long

Error Code	SQLSTATE	Error	Description
4160		ER_TOO_LONG_DATABASE_COMMENT	Comment for database ‘%-.64s’ is too long (max = %u)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4161: Unknown data type

Error Code	SQLSTATE	Error	Description
4161		ER_UNKNOWN_DATA_TYPE	Unknown data type: ‘%-.64s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4162: Operator does not exists

Error Code	SQLSTATE	Error	Description
4162		ER_UNKNOWN_OPERATOR	Operator does not exists: ‘%-.128s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4163: Table history row start is later than row end

Error Code	SQLSTATE	Error	Description
4163		ER_WARN_HISTORY_ROW_START_TIME	Table `%s.%s` history row start '%s' is later than row end '%s'

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4164: STARTS is later than query time, first history partition may exceed INTERVAL value

Error Code	SQLSTATE	Error	Description
4164		ER_PART_STARTS_BEYOND_INTERVAL	%`s: STARTS is later than query time, first history partition may exceed INTERVAL value

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4165: DDL-statement is forbidden as table storage engine does not support Galera replication

Error Code	SQLSTATE	Error	Description
4165		ER_GALERA_REPLICATION_NOT_SUPPORTED	DDL-statement is forbidden as table storage engine does not support Galera replication

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4166: The used command is not allowed because the MariaDB server or client has disabled the lo

Error 4166: The used command is not allowed because the MariaDB server or client has disabled the local infile capability

Error Code	SQLSTATE	Error	Description
4166	HY000	ER_LOAD_INFILE_CAPABILITY_DISABLED	The used command is not allowed because the MariaDB server or client has disabled the local infile capability

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4167: No secure transports are configured, unable to set –require_secure_transport=ON

Error Code	SQLSTATE	Error	Description
4167		ER_NO_SECURE_TRANSPORTS_CONFIGURED	No secure transports are configured, unable to set –require_secure_transport=ON

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4168: Slave SQL thread ignored the '%s' because table is shared

Error Code	SQLSTATE	Error	Description
4168		ER_SLAVE_IGNORED_SHARED_TABLE	Slave SQL thread ignored the '%s' because table is shared

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4169: AUTO_INCREMENT column cannot be used in the UNIQUE index

Error Code	SQLSTATE	Error	Description
4169		ER_NO_AUTOINCREMENT_WITH_UNIQUE	AUTO_INCREMENT column %`s cannot be used in the UNIQUE index %`s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4170: Key cannot explicitly include column

Error Code	SQLSTATE	Error	Description
4170		ER_KEY_CONTAINS_PERIOD_FIELDS	Key %`s cannot explicitly include column %`s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4171: Key cannot have WITHOUT OVERLAPS

Error Code	SQLSTATE	Error	Description
4171		ER_KEY_CANT_HAVE_WITHOUT_OVERLAPS	Key %`s cannot have WITHOUT OVERLAPS

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4172: is not allowed in this context

Error Code	SQLSTATE	Error	Description
4172		ER_NOT_ALLOWED_IN_THIS_CONTEXT	'%-.128s' is not allowed in this context

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4173: Engine does not support rollback. Changes where committed during rollback call

Error Code	SQLSTATE	Error	Description
4173		ER_DATA_WAS_COMMITED_UNDER_ROLLBACK	Engine %s does not support rollback. Changes were committed during rollback call

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4174: A primary key cannot be marked as IGNORE

Error Code	SQLSTATE	Error	Description
4174		ER_PK_INDEX_CANT_BE_IGNORED	A primary key cannot be marked as IGNORE

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4175: SKIP LOCKED makes this statement unsafe

Error Code	SQLSTATE	Error	Description
4175		ER_BINLOG_UNSAFE_SKIP_LOCKED	SKIP LOCKED makes this statement unsafe

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4176: Field can't be set for JSON_TABLE

Error Code	SQLSTATE	Error	Description
4176		ER_JSON_TABLE_ERROR_ON_FIELD	Field '%s' can't be set for JSON_TABLE '%s'.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4177: Every table function must have an alias

Error Code	SQLSTATE	Error	Description
4177		ER_JSON_TABLE_ALIAS_REQUIRED	Every table function must have an alias.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4178: Can't store an array or an object in the scalar column of JSON_TABLE

Error Code	SQLSTATE	Error	Description
4178		ER_JSON_TABLE_SCALAR_EXPECTED	Can't store an array or an object in the scalar column '%s' of JSON_TABLE '%s'.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4179: Can’t store multiple matches of the path in the column of JSON_TABLE

Error Code	SQLSTATE	Error	Description
4179		ER_JSON_TABLE_MULTIPLE_MATCHES	Can’t store multiple matches of the path in the column ‘%s’ of JSON_TABLE ‘%s’.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4180: FETCH … WITH TIES requires ORDER BY clause to be present

Error Code	SQLSTATE	Error	Description
4180		ER_WITH_TIES_NEEDS_ORDER	FETCH … WITH TIES requires ORDER BY clause to be present

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4181: Dropped orphan trigger, originally created for table

Error Code	SQLSTATE	Error	Description
4181		ER_REMOVED_ORPHAN_TRIGGER	Dropped orphan trigger ‘%-.64s’, originally created for table: ‘%-.192s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4182: Storage engine is disabled

Error Code	SQLSTATE	Error	Description
4182		ER_STORAGE_ENGINE_DISABLED	Storage engine %s is disabled

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4183: SFORMAT error

Error Code	SQLSTATE	Error	Description
4183		WARN_SFORMAT_ERROR	SFORMAT error: %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4184: Convert partition is not supported for subpartitioned table

Error Code	SQLSTATE	Error	Description
4184		ER_PARTITION_CONVERT_SUBPARTITIONED	Convert partition is not supported for subpartitioned table.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4185: MariaDB tried to use the %s, but its provider plugin is not loaded

Error Code	SQLSTATE	Error	Description
4185		ER_PROVIDER_NOT_LOADED	MariaDB tried to use the %s, but its provider plugin is not loaded

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4186: Failed to parse histogram for table at offset

Error Code	SQLSTATE	Error	Description
4186		ER_JSON_HISTOGRAM_PARSE_FAILED	Failed to parse histogram for table %s.%s: %s at offset %d.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4187: OUT or INOUT argument for function is not allowed here

Error Code	SQLSTATE	Error	Description
4187		ER_SF_OUT_INOUT_ARG_NOT_ALLOWED	OUT or INOUT argument %d for function %s is not allowed here

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4188: Replicated query table can not be temporary

Error Code	SQLSTATE	Error	Description
4188		ER_INCONSISTENT_SLAVE_TEMP_TABLE	Replicated query ‘%s’ table `%s.%s` can not be temporary

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4189: Versioned table: adding HISTORY partition(s) failed

Error Code	SQLSTATE	Error	Description
4189		ER_VERS_HIST_PART_FAILED	Versioned table % `s.%` s: adding HISTORY partition(s) failed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4190: is implicitly changing the value of from to

Error Code	SQLSTATE	Error	Description
4190		WARN_OPTION_CHANGING	%s is implicitly changing the value of ‘%s’ from ‘%s’ to ‘%s’

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4191: CHANGE MASTER TO option is missing requirement

Error Code	SQLSTATE	Error	Description
4191		ER_CM_OPTION_MISSING_REQUIREMENT	CHANGE MASTER TO option '%s=%s' is missing requirement %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4192: Slave log event execution was interrupted (slave_max_statement_time exceeded)

Error Code	SQLSTATE	Error	Description
4192		ER_SLAVE_STATEMENT_TIMEOUT	Slave log event execution was interrupted (slave_max_statement_time exceeded)

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4193: Invalid value for keyword

Error Code	SQLSTATE	Error	Description
4193		ER_JSON_INVALID_VALUE_FOR_KEYWORD	Invalid value for keyword %s

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4194: keyword is not supported

Error Code	SQLSTATE	Error	Description
4194		ER_JSON_SCHEMA_KEYWORD_UNSUPPORTED	%s keyword is not supported

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4195: Variable schema is not supported

Error Code	SQLSTATE	Error	Description
4195		ER_JSON_NO_VARIABLE_SCHEMA	Variable schema is not supported.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4196: Pseudo thread id should not be modified by the client as it will be overwritten

Error Code	SQLSTATE	Error	Description
4196		ER_PSEUDO_THREAD_ID_OVERWRITE	Pseudo thread id should not be modified by the client as it will be overwritten.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4197: Wrong number of columns

Error Code	SQLSTATE	Error	Description
4197		ER_SEQUENCE_TABLE_HAS_WRONG_NUMBER_OF_COLUMNS	Wrong number of columns.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4198: Sequence tables cannot have any keys

Error Code	SQLSTATE	Error	Description
4198		ER_SEQUENCE_TABLE_CANNOT_HAVE_ANY_KEYS	Sequence tables cannot have any keys

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4199: Sequence tables cannot have any constraints

Error Code	SQLSTATE	Error	Description
4199		ER_SEQUENCE_TABLE_CANNOT_HAVE_ANY_CONSTRAINTS	Sequence tables cannot have any constraints

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Codes 4200 to 4299

- Error 4200: ORDER BY
- Error 4201: The variable is ignored
- Error 4202: Incorrect column name count for derived table
- Error 4203: Values were longer than max_sort_length. Sorting used only the first bytes
- Error 4204: Invalid binary vector format
- Error 4205: Invalid vector format at offset
- Error 4206: Cannot determine distance type for VEC_DISTANCE, index is not found
- Error 4221: Unresolved table name for hint
- Error 4222: Unresolved index name for hint
- Error 4223: Optimizer hints are not supported inside view definitions
- Error 4224: Hint is ignored as malformed
- Error 4225: Optimizer hints at the INSERT part of a INSERT..SELECT statement are not supported

- Error 4226: Slave SQL thread aborted

This page is licensed: CC BY-SA / Gnu FDL

Error 4200: ORDER BY

Error Code	SQLSTATE	Error	Description
4200		ER_SEQUENCE_TABLE_ORDER_BY	ORDER BY

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4201: The variable is ignored

Error Code	SQLSTATE	Error	Description
4201		ER_VARIABLE_IGNORED	The variable '%s' is ignored. It only exists for compatibility with old installations and will be removed in a future release

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4202: Incorrect column name count for derived table

Error Code	SQLSTATE	Error	Description
4202		ER_INCORRECT_COLUMN_NAME_COUNT	Incorrect column name count for derived table

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4203: Values were longer than max_sort_length. Sorting used only the first bytes

Error Code	SQLSTATE	Error	Description
4203		WARN_SORTING_ON_TRUNCATED_LENGTH	%llu values were longer than max_sort_length. Sorting used only the first %lu bytes

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4204: Invalid binary vector format

Error Code	SQLSTATE	Error	Description
4204		ER_VECTOR_BINARY_FORMAT_INVALID	Invalid binary vector format. Must use IEEE standard float representation in little-endian format. Use VEC_FromText() to generate it.

Possible Causes and Solutions

Argument is not a Valid Binary Vector

For example, the VEC_ToText function expects a binary vector. If not given one, this error will result:

```
SELECT VEC_ToText(x'aabbcc');
+-----------------------+
| VEC_ToText(x'aabbcc') |
+-----------------------+
| NULL                  |
+-----------------------+
1 row in set, 1 warning (0.000 sec)

Warning (Code 4204): Invalid binary vector format. Must use IEEE standard float
  representation in little-endian format. Use VEC_FromText() to generate it.
```

The solution is to provide a valid binary vector:

```
SELECT VEC_ToText(x'e360d63ebe554f3fcdbc523f4522193f5236083d');
+-----------------------------------------------------------+
| VEC_ToText(x'e360d63ebe554f3fcdbc523f4522193f5236083d') |
+-----------------------------------------------------------+
| [0.418708,0.809902,0.823193,0.598179,0.033255]          |
+-----------------------------------------------------------+
```

This page is licensed: CC BY-SA / Gnu FDL

Error 4205: Invalid vector format at offset

Error Code	SQLSTATE	Error	Description
4205		ER_VECTOR_FORMAT_INVALID	Invalid vector format at offset: %d for '%-.100s'. Must be a valid JSON array of numbers.

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4206: Cannot determine distance type for VEC_DISTANCE, index is not found

Error Code	SQLSTATE	Error	Description
4206		ER_VEC_DISTANCE_TYPE	Cannot determine distance type for VEC_DISTANCE, index is not found

Possible Causes and Solutions

Table is Built Without a Vector Index

```
CREATE OR REPLACE TABLE v (id INT PRIMARY KEY, v VECTOR(5) NOT NULL);

INSERT INTO v VALUES
    (1, x'e360d63ebe554f3fcdbc523f4522193f5236083d'),
    (2, x'f511303f72224a3fdd05fe3eb22a133ffae86a3f'),
    (3,x'f09baa3ea172763f123def3e0c7fe53e288bf33e'),
    (4,x'b97a523f2a193e3eb4f62e3f2d23583e9dd60d3f'),
    (5,x'f7c5df3e984b2b3e65e59d3d7376db3eac63773e'),
    (6,x'de01453ffa486d3f10aa4d3fdd66813c71cb163f'),
    (7,x'76edfc3e4b57243f10f8423fb158713f020bda3e'),
    (8,x'56926c3fdf098d3e2c8c5e3d1ad4953daa9d0b3e'),
    (9,x'7b713f3e5258323f80d1113d673b2b3f66e3583f'),
    (10,x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e');

SELECT id FROM v ORDER BY VEC_DISTANCE(v, x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e');
ERROR 4206 (HY000): Cannot determine distance type for VEC_DISTANCE, index is not found
```

To solve, add a vector index to the table:

```
ALTER TABLE v ADD VECTOR INDEX(v);

SELECT id FROM v ORDER BY VEC_DISTANCE(v, x'6ca1d43e9df91b3fe580da3e1c247d3f147cf33e');
+----+
| id |
+----+
| 10 |
|  7 |
|  3 |
|  9 |
|  2 |
|  1 |
|  5 |
|  4 |
|  6 |
|  8 |
+----+
```

This page is licensed: CC BY-SA / Gnu FDL

Error 4221: Unresolved table name for hint

Error Code	SQLSTATE	Error	Description
4221		ER_UNRESOLVED_TABLE_HINT_NAME	Unresolved table name %s for %s hint

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4222: Unresolved index name for hint

Error Code	SQLSTATE	Error	Description
4222		ER_UNRESOLVED_INDEX_HINT_NAME	Unresolved index name %s for %s hint

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4223: Optimizer hints are not supported inside view definitions

Error Code	SQLSTATE	Error	Description
4223		ER_HINTS_INSIDE_VIEWS_NOT_SUPPORTED	Optimizer hints are not supported inside view definitions

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4224: Hint is ignored as malformed

Error Code	SQLSTATE	Error	Description
4224		ER_WARN_MALFORMED_HINT	Hint %s is ignored as malformed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4225: Optimizer hints at the INSERT part of a INSERT..SELECT statement are not supported

Error Code	SQLSTATE	Error	Description
4225		ER_WARN_HINTS_ON_INSERT_PART_OF_INSERT_SELECT	Optimizer hints at the INSERT part of a INSERT..SELECT statement are not supported

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4226: Slave SQL thread aborted

Error Code	SQLSTATE	Error	Description
4226		ER_INIT_SLAVE_ERROR	Slave SQL thread aborted. Can't execute init_slave query due to error code %u: %s

Possible Causes and Solutions

Prior to the introduction of this error, there were multiple error codes reported for the issue 'Can't execute init_slave query'. Those errors are the underlying reason the init_slave query cannot be executed, but this made it difficult to detect the issue in an automated way.

This error code, ER_INIT_SLAVE_ERROR, unifies all the errors related to the init_slave query. The ER_INIT_SLAVE_ERROR error is raised for any issue related to the init_slave query, and the underlying error code and message are included in the Last_SQL_Error field. See those specific errors for more details.

This page is licensed: CC BY-SA / Gnu FDL

Error 4265: LIST partition type does not support INTERVAL

Error Code	SQLSTATE	Error	Description
4265	HY000	ER_PARTITION_INTERVAL_NOT_LIST	LIST partition type does not support INTERVAL

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4266: RANGE COLUMN partition by a DATE with INTERVAL smaller than date

Error Code	SQLSTATE	Error	Description
4266	HY000	ER_PARTITION_INTERVAL_FINER_THAN_DATE	RANGE COLUMN partition by a DATE with INTERVAL smaller than date

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4267: MAXVALUE is not allowed in range partitioning with interval

Error Code	SQLSTATE	Error	Description
4267	HY000	ER_PARTITION_INTERVAL_MAXVALUE	MAXVALUE is not allowed in range partitioning with interval

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4268: Range partition table %s.%s: adding INTERVAL partition(s) failed

Error Code	SQLSTATE	Error	Description
4268	HY000	ER_RANGE_INTERVAL_PART_FAILED	Range partition table % `s.%` s: adding INTERVAL partition(s) failed

Possible Causes and Solutions

This page is licensed: CC BY-SA / Gnu FDL

Error 4269: Failed to parse saved optimizer context: %s at offset %d.

Error Code	SQLSTATE	Error	Description
4269	HY000	ER_JSON_OPTIMIZER_REPLAY_CONTEXT_PARSE_FAILED	Failed to parse saved optimizer context: %s at offset %d.

Possible Causes and Solutions

Error 4270: Failed to match the stats from replay context with the optimizer stats: %s doesn't exist in %s

Error Code	SQLSTATE	Error	Description
4270	HY000	ER_JSON_OPTIMIZER_REPLAY_CONTEXT_MATCH_FAILED	Failed to match the stats from replay context with the optimizer stats: %s doesn't exist in %s

Possible Causes and Solutions

Client/Server Protocol

Client/Server Protocol Overview

See Protocol Data Types.

This page defines the fundamental data types used in the MariaDB client/server protocol, including integers, strings, and binary representations.

See MariaDB Protocol Differences with MySQL.

Learn about the specific extensions and differences in the MariaDB protocol compared to MySQL, such as extended capability flags and metadata.

Client/Server Protocol Documentation

See 0 - Packet.

Understand the standard packet structure in the MariaDB protocol, including the packet header, length, sequence number, and payload handling.

See 1 - Connecting.

Learn about the connection phase in the client/server protocol. This section details how clients establish initial communication with the server, including handshaking and authentication processes.

See 2 - Text Protocol.

Understand the text protocol in the Server's client/server communication. This section details how SQL commands and results are exchanged as plain text, including command types and packet structures.

See 3 - Binary Protocol (Prepared Statements).

Understand the binary protocol for prepared statements. This section details how prepared statements are exchanged efficiently between client and server, optimizing performance and security.

See 4 - Server Response Packets.

Understand server response packets in MariaDB's client/server protocol. This section details the various types of packets sent by the server, including OK, Error, and Result Set packets.

Replication Protocol Documentation

See Replication Protocol.

Understand the replication protocol. This section details how primary and replica servers communicate, exchanging binary log events to ensure data consistency and enable high availability.

This page is licensed: CC BY-SA / Gnu FDL

Protocol Data Types

List of Possible Types

Unknown type:

byte<1>	Fixed-length bytes
byte<lenenc>	Length-encoded bytes
byte<EOF>	End-of-file length bytes

Integer type:

int<1>	Fixed-length integers
int<lenenc>	Length-encoded integers

String type:

string<fix>	Fixed-length strings
string<NUL>	Null-terminated strings
string<lenenc>	Length-encoded strings
string<EOF>	End-of-file length strings

Fixed-Length Bytes

The notation is `byte <n>`, where `<n>` is a positive integer. A fixed-length byte stores the value in a series of `n` bytes.

Length-Encoded Bytes

The notation is `byte<lenenc>`. Length-encoded bytes are prefixed by a length-encoded integer which describes the length of the byte value, followed by the bytes value.

End of File Length Bytes

The notation is `byte<EOF>`. Bytes whose length is calculated by the packet remaining length.

Fixed-Length Integers

Notation is `int <n>`, where `<n>` is a positive integer. A fixed-length integer stores the value in a series of `n` bytes. The least significant byte is always the first byte (little-endian format).

Example

An `int<4>` with value of `2` is stored as `02 00 00 00` .

Length-Encoded Integers

The notation is `int<lenenc>`. An integer which depending on its value is represented by `n` bytes.

The first byte represents the size of the integer, depending on the value of the first byte:

- < `0xFB` - Integer value is this a 1-byte integer
- `0xFB` - `NULL` value
- `0xFC` - Integer value is encoded in the next 2 bytes (3 bytes total)
- `0xFD` - Integer value is encoded in the next 3 bytes (4 bytes total)
- `0xFE` - Integer value is encoded in the next 8 bytes (9 bytes total)

Fixed-Length Strings

The notation is `string<fix>`. Fixed-length strings have a known hardcoded length.

Null-Terminated Strings

The notation is `string<NUL>`. Null-terminated strings have a variable size and are terminated by a `0x00` character.

Length-Encoded Strings

The notation is `string<lenenc>`. Length-encoded strings are prefixed by a length-encoded integer which describes the length of the string, followed by the string value.

Example

An string of 512 “a” is encoded in 515 bytes:

```
fc 00 02 97 97 97 97 97 97 97 97 97 97 97 97 ² .. a a a a a a a a a a a a a
```

The `NULL` value is encoded using null (`0xfb`) length.

An empty value is encoded with a 0 (`0x00`) length.

End of File Length Strings

The notation is `string<EOF>`.

Strings whose length is calculated by the packet remaining length. For an example, see the COM_STMT_PREPARE packet.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Protocol Differences with MySQL

Here is a list of the differences between MariaDB and MySQL in terms of protocol, in order to help community driver maintainers.

MariaDB Capabilities Extension

MariaDB/MySQL servers can advertise feature support using capabilities. To expand the capabilities beyond the original 4 bytes, MariaDB utilizes 4 bytes, unused by MySQL, in the Initial handshake packet (server capabilities 3rd part). In order to avoid incompatibility in the future, those 4 bytes have to be read only if capability `CLIENT_MYSQL` is not set (server then being MariaDB).

Enhanced Capabilities

- `MARIADB_CLIENT_PROGRESS` : Client progress reporting.
- `MARIADB_CLIENT_CACHE_METADATA` : Enables clients to cache metadata and avoid repeated network transmissions (since MariaDB 10.6.0).
- `MARIADB_CLIENT_EXTENDED_METADATA` : Provides more detailed column metadata information for specific data types (since MariaDB 10.5.2).
- `MARIADB_CLIENT_STMT_BULK_OPERATIONS` : Introduces a dedicated command, COM_STMT_BULK_EXECUTE, for efficient batch execution of statements.
- MARIADB_CLIENT_BULK_UNIT_RESULTS: Allows for individual result sets for each bulk operation (since MariaDB 11.5.1).

See Connection Capabilities.

Prepare Statement Skipping Metadata

> **Note:**
>
> This feature is available from *MariaDB 10.6.0.*

Prepared statement metadata, which typically remains unchanged except during table alterations, can be cached by clients when the `MARIADB_CLIENT_CACHE_METADATA` capability is enabled. The server won't then send them again, unless they change. This significantly improves the performance of subsequent executions, especially for large metadata sets.

When `MARIADB_CLIENT_CACHE_METADATA` capability is set, the result set Column count packet format indicates if metadata follows or is skipped:

- `int<lenenc>` column count,
- if (`MARIADB_CLIENT_CACHE_METADATA` capability set) `int<1>` metadata follows (0 / 1).

Example

Java code:

```
stmt.execute("CREATE TABLE test_table (id int, val varchar(32))");
stmt.execute("INSERT INTO test_table VALUES (1, 'a'), (2, 'b')");
try (PreparedStatement prep = sharedConnBinary.prepareStatement("SELECT * FROM test_table WHERE id = ?"))
{
    prep.setInt(1, 1);
    prep.executeQuery();
}
```

Results with metadata caching:

```
Column count packet:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 02 00 00 01 02 00                               | ......           |
+------+-------------------------------------------------+------------------+
row:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 08 00 00 02 00 00 01 00  00 00 01 61            | ...........a     |
+------+-------------------------------------------------+------------------+

OK_Packet with a 0xFE header:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 07 00 00 03 FE 00 00 22  00 00 00               | .......".....    |
+------+-------------------------------------------------+------------------+
```

The same, without metadata caching:

```
Column count packet:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 01 00 00 01 02                                  | .....            |
+------+-------------------------------------------------+------------------+

Column Definition packet:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 33 00 00 02 03 64 65 66  05 74 65 73 74 6A 0A 74 | 3....def.testj.t |
|000010| 65 73 74 5F 74 61 62 6C  65 0A 74 65 73 74 5F 74 | est_table.test_t |
|000020| 61 62 6C 65 02 69 64 02  69 64 0C 3F 00 0B 00 00 | able.id.id.?.... |
|000030| 00 03 00 00 00 00 00                            | .......          |
+------+-------------------------------------------------+------------------+

Column Definition packet:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 35 00 00 03 03 64 65 66  05 74 65 73 74 6A 0A 74 | 5....def.testj.t |
|000010| 65 73 74 5F 74 61 62 6C  65 0A 74 65 73 74 5F 74 | est_table.test_t |
|000020| 61 62 6C 65 03 76 61 6C  03 76 61 6C 0C FF 00 80 | able.val.val.... |
|000030| 00 00 00 FD 00 00 00 00  00                     | .........        |
+------+-------------------------------------------------+------------------+

row:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 08 00 00 04 00 00 01 00  00 00 01 61            | ...........a     |
+------+-------------------------------------------------+------------------+

OK_Packet with a 0xFE header:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 07 00 00 05 FE 00 00 22  00 00 00               | .......".....    |
+------+-------------------------------------------------+------------------+
```

Extended Column Information

> **Note:**
>
> This feature is available from MariaDB 10.5.2.

When the `MARIADB_CLIENT_EXTENDED_METADATA` capability is set, column definition packet can include additional type and format information.

- For geometric fields: Detailed geometric data type (for instance, `point`, `polygon`).
- For JSON fields: Type `json`.
- For UUID fields: Type `uuid`.

Bulk

> **Note:**
>
> This feature is available for unit results from MariaDB 10.2.0 or MariaDB 11.5.1, respectively.

The `MARIADB_CLIENT_STMT_BULK_OPERATIONS` capability enables the COM_STMT_BULK_EXECUTE command for efficient batch processing. However, note that only one result (`OK` or `ERROR`) is returned per batch, containing the total affected rows and the first auto-generated ID. For individual results, the `MARIADB_CLIENT_BULK_UNIT_RESULTS` capability can be set. The server will then return a result set containing for each unitary results (containing auto generated ids and affected rows).

Example

Java code:

```
Statement stmt = connection.createStatement();
stmt.execute("CREATE TABLE test_table (id int, val varchar(32))");

try (PreparedStatement prep = connection.prepareStatement("INSERT INTO test_table VALUES (?, ?)")) {

    prep.setInt(1, 1);
    prep.setString(2, "a");
    prep.addBatch();

    prep.setInt(1, 2);
    prep.setString(2, "b");
    prep.addBatch();

    prep.executeBatch();
}
```

Client send:

```
MARIADB_CLIENT_STMT_BULK_OPERATIONS:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 1B 00 00 00 FA FF FF FF  FF 80 00 03 00 FD 00 00 | ................ |
|000010| 01 00 00 00 00 01 61 00  02 00 00 00 00 01 62    | ......a.......b  |
+------+-------------------------------------------------+------------------+
```

Server response:

```
OK_Packet:
       +-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| 2E 00 00 01 00 02 00 02  00 00 00 26 52 65 63 6F | ...........&Reco |
|000010| 72 64 73 3A 20 32 20 20  44 75 70 6C 69 63 61 74 | rds: 2  Duplicate |
|000020| 65 73 3A 20 30 20 20 57  61 72 6E 69 6E 67 73 3A | es: 0  Warnings: |
|000030| 20 30                                           |  0               |
+------+-------------------------------------------------+------------------+
```

Authentication Plugins

MariaDB has specific authentication methods.

- ED25519 plugin
- PARSEC plugin (from MariaDB 11.6.1)
- GSSAPI plugin

Redirection

> **Note:**
>
> This feature is available from MariaDB 11.3.1 or MaxScale 25.08.0, respectively.

MariaDB permits connection redirection.

Use Cases

- Proxy Scenarios: Connection redirection is particularly beneficial when multiple servers share a single proxy.
- Server Management: This feature can also be used during planned server shutdowns or restarts, allowing for a graceful transition to a new server.

Connectors can support 2 different levels:

- On Connection Creation only: The redirection information is included in the initial OK_Packet sent by the server to the client. This allows the client to connect directly to the target server immediately.
- Anytime Redirection: If redirection information becomes available later, the connector can handle it based on the existing transaction state.
  - No Transaction: If no transaction is in progress, the connector can redirect the connection directly.
  - Transaction in Progress: If a transaction is ongoing, the redirection information is stored until the transaction is completed. The transaction state is determined using server status flags like `SERVER_STATUS_IN_TRANS` in the `OK_Packet`, `ERR_Packet`, or `EOF_Packet`.

Zero-Configuration SSL

> **Note:**
>
> This feature is available from MariaDB 11.4.1.

A feature that enables TLS certificate validation without requiring client-side certificate configuration.

Limitations

- Requires a nonempty password.
- Only supports the following authentication methods:
  - `mysql_native_password`
  - `client_ed25519`

- `parsec`

Operational Mechanism

Server-Side Process

1. When no SSL certificates are pre-configured, the server automatically generates a temporary self-signed certificate.
2. During connection establishment, the server embeds a special validation hash in the connection's `OK_Packet` information field.

Client-Side Process

1. The client connector must postpone SSL error handling until the connection phase is complete.
2. The client captures and stores the SHA256 fingerprint of the server's certificate.
3. If SSL errors occur, the client can only use specific authentication plugins (`mysql_native_password` / `ed25519` / `parsec`) to prevent potential password exposure.
4. At connection conclusion, the server sends an `OK_Packet` with a validation hash.
5. The client generates a hash using:

- The password hash;
- The server's seed;
- Stored certificate fingerprint.

> **Warning:**
>
> The SSL-error connection proceeds only if the client-generated hash matches the server-provided hash.

Password Hash Generation Methods

- `mysql_native_password`:
  - Hash generation: `SHA1`(`SHA1`(*password*)).
- `ed25519`:
  - Uses the Ed25519 cryptographic algorithm for hash generation.
- `parsec`:
  - Hash generation involves combining
    - `P` character;
    - Number of iterations;
    - Salt;
    - Raw public key.

Initial Session Tracking

MySQL and MariaDB support session tracking when the `CLIENT_SESSION_TRACK` capability is set.

> **Note:**
>
> One difference is that, since MariaDB 11.5.1, connection ending `OK_Packet` lists all the current variables of tracked variable.

This is useful for connectors which have a method to set the transaction type, retrieving database for example to always have the server current value when changed. This permits to avoid executing some queries when not needed.

Example of ending connection `OK_Packet` :

```
+-------------------------------------------------+
       |  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+------+-------------------------------------------------+------------------+
|000000| A6 00 00 02 00 00 00 02  40 00 00 00 9D 00 0E 0A | ........@....... |
|000010| 61 75 74 6F 63 6F 6D 6D  69 74 02 4F 4E 00 11 09 | autocommit.ON... |
|000020| 74 69 6D 65 5F 7A 6F 6E  65 06 53 59 53 54 45 4D | time_zone.SYSTEM |
|000030| 00 1D 14 63 68 61 72 61  63 74 65 72 5F 73 65 74 | ...character_set |
|000040| 5F 63 6C 69 65 6E 74 07  75 74 66 38 6D 62 34 00 | _client.utf8mb4. |
|000050| 21 18 63 68 61 72 61 63  74 65 72 5F 73 65 74 5F | !.character_set_ |
|000060| 63 6F 6E 6E 65 63 74 69  6F 6E 07 75 74 66 38 6D | connection.utf8m |
|000070| 62 34 00 1E 15 63 68 61  72 61 63 74 65 72 5F 73 | b4...character_s |
|000080| 65 74 5F 72 65 73 75 6C  74 73 07 75 74 66 38 6D | et_results.utf8m |
|000090| 62 34 00 0E 0C 72 65 64  69 72 65 63 74 5F 75 72 | b4...redirect_ur |
|0000a0| 6C 00 01 06 05 74 65 73  74 6A                   | l....testj       |
+------+-------------------------------------------------+------------------+
```

It indicates:

- autocommit = ON
- time_zone = SYSTEM
- character_set_client = utf8mb4
- character_set_connection = utf8mb4
- character_set_results = utf8mb4
- redirect_url =

A connector knows that, if `character_set_client` is set to `utf8mb4`, it can avoid executing a statement like `SET NAMES utf8mb4`.

MySQL Features Not Supported

- The X protocol is not supported.

Unsupported features and associate capabilities:

- `CLIENT_OPTIONAL_RESULTSET_METADATA`: permits setting no `METADATA` at all for a connection. See Prepare statement skipping metadata's MariaDB implementation choice.
- `CLIENT_QUERY_ATTRIBUTES` adds some metadata attributes.
- `CLIENT_ZSTD_COMPRESSION_ALGORITHM` permits zstd compression.
- `MULTI_FACTOR_AUTHENTICATION` Multifactor Authentication capability.

TIPS

Identifying MariaDB Server

MariaDB connectors use specific criteria to determine if a server is a MariaDB instance during the initial handshake process.

The two key indicators used are:

- Missing `CLIENT_MYSQL` capability: MariaDB does not set the `CLIENT_MYSQL` capability flag in the initial handshake packet.
- Server version string: The server's version string is examined for the presence of the word *mariadb* (ignoring case sensitivity).

The reason is that some features like using `COM_RESET_CONNECTION` have no capability, and depend on the MySQL or MariaDB server version.

Pipelining Prepare Execute

Connectors usually follow a two-step process for prepared statements:

1. Prepare: Send a COM_STMT_PREPARE command to the server, receiving a statement ID in response.
2. Execute: Send a COM_STMT_EXECUTE command, using the statement ID obtained in the previous step.

When the server support `MARIADB_CLIENT_STMT_BULK_OPERATIONS` capability, a specific statement ID value of `-1` (or `0xffffffff` in hexadecimal) can be used to indicate that the previously prepared statement can be reused. This enables connectors to pipeline the preparation and execution steps into a single request:

- Send a COM_STMT_PREPARE, then a COM_STMT_EXECUTE with statement ID `-1` (`0xffffffff`) command, to the server.
- Read the prepare and execute responses.

If the COM_STMT_PREPARE command returns an error (`ERR_Packet`), the subsequent COM_STMT_EXECUTE with statement ID `-1` also fails and returns an error.

By eliminating the round trip for the separate `COM_STMT_EXECUTE` command, this approach improves performance for the first execution.

Traditionally, connectors send COM_STMT_PREPARE, wait for results, then execute COM_STMT_EXECUTE with `statement_id` received from the prepare result.

This description is for COM_STMT_EXECUTE, but COM_STMT_BULK_EXECUTE works exactly the same way.

Query Timeout

A timeout for all commands can be set using `SET max_statement_time=` *`XXX`* with *`XXX`* in seconds.

Setting it for a specific query can be done using `SET STATEMENT max_statement_time=XXX FOR ...`

Collations

Connectors don't care about collations, but normally want to ensure charset in connection exchanges.

The only good solution is to use `SET NAMES utf8mb4` or `SET NAMES utf8mb4 COLLATE someUtf8mb4collation`.

If they support session tracking, connectors can check if the character set of initially tracked variable `character_set_connection` corresponds to the expected value, then permit skipping this `SET NAMES` statement ('server default collation' from initial handshake packet cannot be trusted, since truncated to one byte. Recent mysql and mariadb collation can go on 2 bytes).

This page is licensed: CC BY-SA / Gnu FDL

0 - Packet

Client/server exchanges are done using the following format:

Standard Packet

The standard MySQL/MariaDB packet has a 4-byte header and a packet body:

- int<3> packet length;
- int<1> sequence number;
- byte<n> packet body.

Packet length is the length of the packet body. Packet length size cannot be more than 3 bytes length value. The actual length of the packet is calculated as from the 3 bytes as length:

```
byte[0] + (byte[1]<<8) + (byte[2]<<16)
```

The maximum size of a packet (with all 3 bytes `0xff`) is 16777215 , or 2^24-1 or `0xffffff`, or 16MB-1 byte.

The sequence number indicates the exchange number when an exchange demands different exchanges. Whenever the client sends a query, the sequence number is set to 0 initially, and is incremented if clients need to split packets.

In more complex situations, when the client and server exchange several packets, for instance, authentication handshake, the rule of thumb for clients is to set sequence number = (last seq.number from received server packet + 1).

Example: Sending a COM_PING packet COM_PING body has only one byte (`0x10`):

```
01 00 00 00 10
```

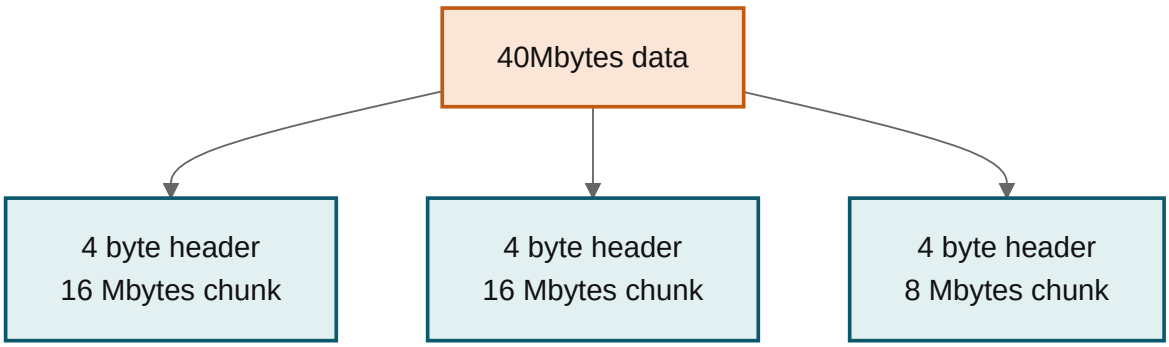
The server then returns an OK_Packet response with a sequence number of `1`.

Packet Splitting

The packet length is 3 bytes, making a maximum size of (2^24 -1) bytes or 16Mbytes-1 byte. But the protocol allows sending and receiving larger data. For those cases, the client can send many packets for the same data, incrementing the sequence number for each packet.

The principle is to split data by chunks of 16M bytes. When the server receives a packet with `0xffffff` length, it will continue to read the next packet. In case of a length of exactly 16M bytes, an empty packet must terminate the sequence.

Example: `max_allowed_packet` is set to a value > to 40M bytes. Sending a 40M bytes packet body:



A 40 Mbyte packet body split across three standard packets: 4-byte header + 16 Mbytes, 4-byte header + 16 Mbytes, 4-byte header + 8 Mbytes.

First packet:

```
ff ff ff 00 ...
```

Second packet:

```
ff ff ff 01 ...
```

Third packet:

```
02 00 80 02 ...
```

The client must be aware of the max_allowed_packet variable value. The server has a buffer to store the body with a maximum size corresponding to this `max_allowed_packet` value. If the client sends more data than

`max_allowed_packet` size, the socket is closed.

> **Note:**
>
> Note that data of exact size 2^24 -1 byte must be sent in 2 packets, the first one with length prefix `0xffffff`, and the second one with length 0 (length byte `0x000000`, seqno incremented). Generally, if data length is an exact multiple of 2^24-1, it must always be followed by an empty packet.

Compressed Packet

For slow connections, the packet can be compressed. This is activated after the handshake-response-packet when the client indicates a `[[1-connecting-connecting#capabilities|COMPRESS]` capability with the server having this functionality, too.

When activated, packets are composed of 7 bytes, a compress header and data. The compression algorithm used is ZLIB, widely available and supported by many languages and runtimes.

- int<3> compressed packet length;
- int<1> compression protocol sequence number;
- int<3> uncompress body length;
- byte<n> compressed body;
  - compressed body contains one or many standard packets but partial packets can also be sent:
    - one or many standard packets :
      - int<3> packet length
      - int<1> sequence number
      - byte<n> packet body

Since compress body can contain many “standard packets”, compress sequence number is incremented separately from sequence number. If the length of the uncompressed payload exceeds `0xffffff` bytes, the uncompressed payload must be sent in several compressed protocol packets.

For small packets, using compression isn’t efficient, so the client can choose to send uncompressed data. That is done by setting the compressed packet length to the real length and the uncompressed packet length to `0`. (Data must then be uncompressed).

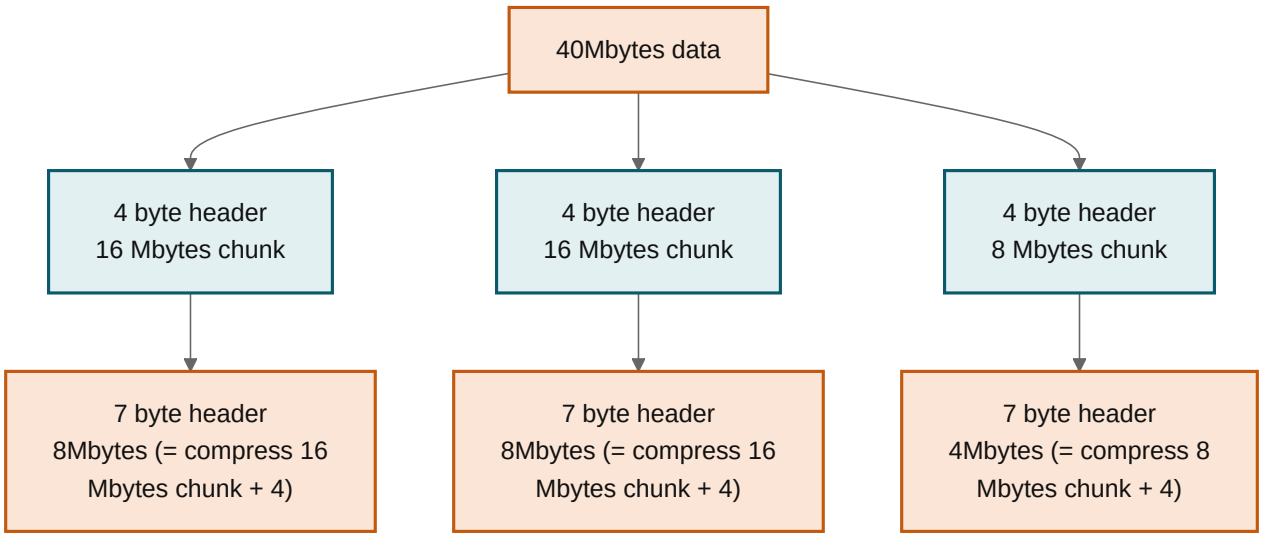
Example: Sending a COM_PING packet `COM_PING` body when `COMPRESS` is enabled. This is 1 byte of data that has then no reason to be compressed, so:

```
01 00 00 00 00 00 00 01 00 00 00 10
```

The server returns an OK_Packet response with a compress sequence number of `1`, and a sequence number of `1`.

Compression Packet Splitting

The server uncompresses data, and then must have the same packet as if there was no compression. If data size needs splitting, it’s better to separate compress packet.



Each standard packet from the 40 Mbyte split is re-wrapped in a 7-byte-header compressed packet whose body size equals the compressed chunk plus its 4-byte header.

This page is licensed: CC BY-SA / Gnu FDL

1 - Connecting

See Connecting.

The connection phase involves an initial handshake where the client and server exchange capabilities, default settings, and authentication data to establish a session.

See Connecting via sha256_password.

The sha256_password plugin manages authentication using SHA-256 encryption, supporting both clear text passwords over SSL and RSA encrypted password exchange.

See Connecting via caching_sha2_password.

This plugin implements the caching_sha2_password authentication method, using an in-memory cache for fast authentication or RSA encryption for full verification.

This page is licensed: CC BY-SA / Gnu FDL

Connecting

Overview

Connection is done by many exchanges:

- (Create socket)
- If first byte from server is `0xFF`:
  - Packet is an ERR_Packet, socket has to be closed.
- Else:
  - Packet is an Initial handshake packet.
  - If SSL/TLS connection:
    - Client sends SSLRequest packet and switches to SSL mode for sending and receiving the following messages:
  - Client sends Handshake response packet.

- Server sends either:
  - An `OK` packet in case of success OK_Packet.
  - An error packet in case of error ERR_Packet.
  - Further authentication data, if requested by the authentication plugin.
    - The content of this authentication data is defined by the authentication plugin.
    - The server *may* send `0x01` byte first to escape the authentication data, particularly if the data starts with the `0x00` or `0xFE` or `0XFF` byte.
    - This optional first `0x01` byte must always be skipped by the client.
  - Authentication switch:
    - If the client or server doesn't have `PLUGIN_AUTH` capability:
      - Server sends `0xFE` byte.
      - Client sends `old_password`.
    - Else:
      - Server sends Authentication switch request.
      - Client may have many exchanges with the server according to the Plugin.
    - Authentication switch ends with server sending either OK_Packet or ERR_Packet.

Initial Handshake Packet

- int<1> protocol version.
- string<NUL> server version:
  - MariaDB Server 10.X versions are by default prefixed "5.5.5-".
  - MariaDB 11.0 and later versions do not have a "5.5.5-" default prefix.
- int<4> connection id.
- string<8> authentication plugin data (1st part).
- string<1> reserved byte.
- int<2> server capabilities (1st part).
- int<1> server default collation.
- int<2> status flags.
- int<2> server capabilities (2nd part).
- If (`server_capabilities` & `PLUGIN_AUTH`):
  - int<1> plugin data length.
- Else:
  - int<1> `0x00`.
- string<6> filler.
- If (`server_capabilities` & `CLIENT_MYSQL`):
  - string<4> filler.
- Else:
  - int<4> server capabilities 3rd part. MariaDB specific flags `/* MariaDB 10.2 or later */`.
- If (`server_capabilities` & `CLIENT_SECURE_CONNECTION`):
  - string<n> authentication plugin data 2nd part. Length = max(12, plugin data length - 9).
  - string<1> reserved byte.
- If (server_capabilities & PLUGIN_AUTH):
  - string<NUL> authentication plugin name.

Client Handshake Response

If the client requests a TLS/SSL connection, the first response is an SSL connection request packet, followed by a handshake response packet. If no TLS is required, client directly sends a handshake response packet.

SSLRequest Packet

- int<4> client capabilities.
- int<4> max packet size.
- int<1> client's default character set and collation.
- string<19> reserved.
- If not (`server_capabilities` & `CLIENT_MYSQL`):
  - int<4> extended client capabilities
- Else:
  - string<4> reserved.

Zero-Configuration SSL Encryption

Automatic Encrypted Connections (MariaDB 11.4+):

Previously, failed SSL connections due to self-signed certificates prevented communication. MariaDB 11.4+ introduces a secondary validation method that works for all servers.

What Happens When SSL Validation Fails?

Even without a valid SSL certificate, the connector can still authenticate by remembering the server's fingerprint (unique identifier). However, it needs to confirm the connection is secure.

Verifying a Secure Connection

The confirmation method depends on the connection type. When using secure MitM-proof methods, like Unix sockets, connector can automatically validate the connection. Otherwise, a shared secret is used.

Shared Secret for Secure Connection

The shared secret is only used if the authentication plugin password is hashable (for instance, `mysql_native_password` , `client_ed25519` , or `parsec`) and not empty.

It's calculated by hashing the user's hash password with the authentication seed and the server fingerprint.

Password hash is generated depending on authentication plugin:

- `ed25519` : identical to password encryption.
- `mysql_native_password` : identical to password encryption.
- `parsec` : ext-salt + raw ed25519 public key.

Server 11.4+ Confirmation Details

For servers running MariaDB 11.4 or later, the final confirmation packet contains:

- int<1> encryption (actually only 0x01 = SHA256 encryption)
- byte<EOF> shared secret.

Matching the Shared Secret

If the calculated shared secret matches the received one, the SSL connection is considered valid (host validation is not needed). Otherwise, the connection must be closed for security reasons.

Handshake Response Packet

- int<4> client capabilities.
- int<4> max packet size.
- int<1> client's default character set and collation.
- string<19> reserved.
- If not (`server_capabilities` & `CLIENT_MYSQL`):
  - int<4> extended client capabilities.
- Else:
  - string<4> reserved.
- string<NUL> username.
- If (password):
  - If (`server_capabilities` & `PLUGIN_AUTH_LENENC_CLIENT_DATA`):
    - string<lenenc> authentication data.
  - Else if (`server_capabilities` & `CLIENT_SECURE_CONNECTION`):
    - int<1> length of authentication response.
    - string<fix> authentication response (length is indicated by previous field).
  - Else:
    - string<NUL> authentication response null ended.
- Else:
  - `string<1>\0` (empty password).
- If (`server_capabilities` & `CLIENT_CONNECT_WITH_DB`):
  - string<NUL> default database name.
- If (`server_capabilities` & `CLIENT_PLUGIN_AUTH`):
  - string<NUL> authentication plugin name.
- If (`server_capabilities` & `CLIENT_CONNECT_ATTRS`):
  - int<lenenc> size of connection attributes.
  - While packet has remaining data:
    - string<lenenc> key.
    - string<lenenc> value.

Server Response to Handshake Response Packet

If the authentication plugin needs further rounds of data exchange (like `parsec`), the server sends additional plugin authentication data (optionally prefixed with `0x01`) to which the client sends an additional response. This can be repeated in multiple rounds. It ends with one of the following:

The server responds with an OK_packet, an ERR_packet, or an Authentication Switch Request packet.

Authentication Switch Request

(If client and server support `CLIENT_AUTH` capability):

- int<1> `0xFE` : Authentication switch request header.
- string<NUL> authentication plugin name.
- byte<EOF> authentication plugin data.

Plugin List

mysql_old_password Plugin

deprecated — send an 8 byte encrypted password.

Authentication plugin data format:

- byte<8> 8-byte seed.

Client response:

- string<NUL> old format encrypted password.

mysql_clear_password Plugin

> **Important:**
>
> Since the password is transmitted in clear, this must be used only when using SSL connections.

Send clear password to server.

Client response:

- string<NUL> password without encryption.

mysql_native_password Plugin

SHA-1 encrypted password with server seed.

Authentication plugin data format:

- string<NUL> seed.

Client response:

- byte<EOF> `SHA1`-encrypted password.

The password is encrypted with: `SHA1( password ) ^ SHA1( seed + SHA1( SHA1( password ) ) )`.

dialog Plugin (PAM)

Interactive exchanges to permit fill passwords — for example for 2-step authentication.

Authentication plugin data format:

- byte<1> password type.
- string<EOF> prompt message.

The server can send one or many requests. For each of them, the client must display this prompt message to the user, to permit the user to type requested information, then send it to the server in string<NUL> format. Password type indicates the answer format (`2` means “read the input with the echo enabled”, `4` means “password-like input, echo disabled”).

First authentication format (from authentication switch packet) can be empty.

This end when the server sends an EOF_Packet, OK_Packet or ERROR_packet.

auth_gssapi_client Plugin

GSSAPI implementation.

Authentication plugin data format:

- string<NUL> serverPrincipalName (UTF-8 format).
- string<NUL> mechanisms (UTF-8 format).

Client must exchange packet with server until having a mutual GSSAPI authentication.
The only difference compared to standard client-server GSSAPI authentication is that exchanges contain standard protocol with packet headers.

client_ed25519 Plugin

The `ed25519` plugin uses the Elliptic Curve Digital Signature Algorithm to securely store users' passwords and to authenticate users.

See plugin description.

The server sends a random nonce that the client signs.

authentication plugin data format:

- byte<EOF> seed.

Client response:

- byte<EOF> `ed25519` encrypted password.

parsec Plugin

Authentication plugin data format:

- string<32> server nonce.

Client has to send an empty packet to request "ext-salt".

Format of ext-salt is:

- string<1> 'P' (denotes KDF algorithm = PBKDF2).
- byte<1> iteration factor. number of iterations correspond to 1024 << iteration factor (0x0 means 1024, 0x1 means 2048, etc.).
- byte<EOF> salt.

The client must then:

- Generate derived key = hash password with PBKDF2 (sha512 digest) with iteration number and salt from ext-salt.
- Generate a client 32 bytes nonce.
- Generate the signature with `ed25519` of an array concatenation of server nonce + client nonce with the generated derived key as private key.

Client response:

- byte<32> client nonce.
- byte<64> signature.

Capabilities

Server and Client have different capabilities, here is the possibles values.
client with capabilities CLIENT_MYSQL + CONNECT_WITH_DB will have a value of 9 (1 + 8).

Capability	Value	Details
CLIENT_MYSQL	1	Set by older MariaDB versions. MySQL named this `CLIENT_LONG_PASSWORD` .

Capability	Value	Details
FOUND_ROWS	2	
CONNECT_WITH_DB	8	You can specify database on connect.
COMPRESS	32	Can use compression protocol
LOCAL_FILES	128	Can use LOAD DATA LOCAL.
IGNORE_SPACE	256	Ignore spaces before `(`.
CLIENT_PROTOCOL_41	1 << 9	4.1 protocol.
CLIENT_INTERACTIVE	1 << 10	
SSL	1 << 11	Can use SSL.
TRANSACTIONS	1 << 13	
SECURE_CONNECTION	1 << 15	4.1 authentication.
MULTI_STATEMENTS	1 << 16	Enable/disable multi-statement support.
MULTI_RESULTS	1 << 17	Enable/disable multi-results.
PS_MULTI_RESULTS	1 << 18	Enable/disable multi-results for PrepareStatement.
PLUGIN_AUTH	1 << 19	Client supports plugin authentication.
CONNECT_ATTRS	1 << 20	Client sends connection attributes.
PLUGIN_AUTH_LENENC_CLIENT_DATA	1 << 21	Enable authentication response packet to be larger than 255 bytes.
CLIENT_CAN_HANDLE_EXPIRED_PASSWORDS	1 << 22	Client can handle expired passwords.
CLIENT_SESSION_TRACK	1 << 23	Enable/disable session tracking in `OK_Packet`.
CLIENT_DEPRECATE_EOF	1 << 24	`EOF_Packet` deprecation: `OK_Packet` replace `EOF_Packet` at the end of the result set when in text format. `EOF_Packet` between columns definition and `resultsetRows` is deleted.
CLIENT_OPTIONAL_RESULTSET_METADATA	1 << 25	Not in use for MariaDB.
CLIENT_ZSTD_COMPRESSION_ALGORITHM	1 << 26	Support `zstd` protocol compression.
CLIENT_CAPABILITY_EXTENSION	1 << 29	Reserved for future use.
CLIENT_SSL_VERIFY_SERVER_CERT	1 << 30	Client verify server certificate. Deprecated, client has options to indicate if server certificate must be verified.
CLIENT_REMEMBER_OPTIONS	1 << 31	
MARIADB_CLIENT_PROGRESS	1 << 32	Client support progress indicator.

Capability	Value	Details
MARIADB_CLIENT_COM_MULTI	1 << 33	deprecated - did permit `COM_MULTI` protocol.
MARIADB_CLIENT_STMT_BULK_OPERATIONS	1 << 34	Permit bulk insert.
MARIADB_CLIENT_EXTENDED_METADATA	1 << 35	Add extended metadata information.
MARIADB_CLIENT_CACHE_METADATA	1 << 36	Permit skipping metadata.
MARIADB_CLIENT_BULK_UNIT_RESULTS	1 << 37	When enabled, indicate that bulk command can use `STMT_BULK_FLAG_SEND_UNIT_RESULTS` flag that permits to return a result set of all affected rows and auto-increment values.

Native Password Authentication

The 20-byte string 'seed' is calculated by concatenating scramble first part (8 bytes) and scramble second part from Initial handshake packet. After that, the client calculates a password hash using the password and seed by using `^` (bitwise xor), `+` (string concatenation), and SHA1 as follows:

```
SHA1( passwd) ^ SHA1( seed + SHA1( SHA1( passwd ) ) )
```

This page is licensed: CC BY-SA / Gnu FDL

Connecting via sha256_password

Overview

`SHA256` authentication possible exchanges:

- If connection uses SSL (SSLRequest Packet sent):
  - Client sends a clear password answer.
- Else:
  - If client doesn't know server RSA public key:
    - Client sends a public key request.
    - Server sends a public key response.
  - Client sends an RSA encrypted password.
  - Ends with server sending either OK_Packet , ERR_Packet.

Authentication

Client Clear Password Answer

- string<NUL> password without encryption.

Public Key Request

- byte<1> fixed `0x01` value.

Public Key Response

- byte<1> fixed `0x01` value.
- byte<EOF> public key data.

RSA Encrypted Password

- byte<256> RSA encrypted password.

RSA encrypted value of `XOR`(password, seed) using server public key (`RSA_PKCS1_OAEP_PADDING`).

This page is licensed: CC BY-SA / Gnu FDL

Connecting via caching_sha2_password

Overview

Caching `SHA256` first sends an `SHA256` -encrypted password. MySQL server has an in-memory cache of `SHA256` key for successful authentication. When a cache hit occurs, the connection is validated, if not, using some more steps to a process similar to sha256_password.

Caching SHA256 authentication possible exchanges:

- Client sends an SHA-2 encrypted password.
- Server result is either OK_Packet , ERR_Packet, or “fast” authentication result.
- If fast authentication result:
  - If connection uses SSL (SSLRequest Packet sent):
    - Client sends a clear password answer.
  - Else:
    - If client doesn’t know server RSA public key:
      - Client sends a public key request.
      - Server sends a public key response.
    - Client sends an RSA encrypted password.
    - Ends with server sending either OK_Packet , ERR_Packet.

Authentication

SHA-2 Encrypted Password

Encryption is `XOR` (`SHA256` (password), `SHA256` (seed, `SHA256` (`SHA256` (password)))).

- byte<32> encrypted password.

“Fast” Authentication Result

Result of fast authentication.

- byte<lenenc> authentication result.

`0x03` value means success authentication.
`0x04` value means continue.

Client Clear Password Answer

- string<NUL> password without encryption.

Public Key Request

> **Warning:**
>
> Value send is not `0x01` like sha256_password use, but `0x02` .

- byte<1> fixed 0x02 value.

Public Key Response

- byte<1> fixed 0x01 value.
- byte<EOF> public key data.

RSA Encrypted Password

- byte<256> RSA encrypted password.

RSA encrypted value of `XOR` (password, seed) using server public key (`RSA_PKCS1_OAEP_PADDING`).

This page is licensed: CC BY-SA / Gnu FDL

2 - Text Protocol

See COM_CHANGE_USER.

This command allows a connected client to re-authenticate as a different user without closing and reopening the connection.

See COM_CREATE_DB.

This command creates a new database on the server, functionally equivalent to the SQL statement CREATE DATABASE.

See COM_DEBUG.

This command forces the server to dump debug information to the standard output or log, requiring the SUPER privilege.

See COM_DROP_DB.

This command drops an existing database from the server, functionally equivalent to the SQL statement DROP DATABASE.

See COM_FIELD_LIST.

This command retrieves a list of fields (columns) for a specified table, similar to the SHOW COLUMNS SQL statement.

See COM_INIT_DB.

This command selects the default database for the current connection, functionally equivalent to the USE statement.

See COM_PING.

This command checks if the server is alive and reachable, the server responds with an OK packet if it is running.

See COM_PROCESSLIST.

This command retrieves a list of active threads and their current status, similar to the SHOW PROCESSLIST statement.

See COM_PROCESS_KILL.

This command asks the server to terminate a specific connection thread, functionally equivalent to the KILL statement.

See COM_QUERY.

This command sends an SQL statement to the server for execution immediately, without the prepare/execute steps.

See COM_QUIT.

This command instructs the server to close the connection and release associated resources.

See COM_RESET_CONNECTION.

This command resets the session state (variables, tables, etc.) to its initial values without closing the connection.

See COM_SET_OPTION.

This command configures client-specific options for the current connection, such as enabling or disabling multi-statements.

See COM_SHUTDOWN.

This command requests the server to shut down, it requires the SHUTDOWN privilege to be executed successfully.

See COM_SLEEP.

This is an internal command used to represent a sleeping connection that is waiting for a new command from the client.

See COM_STATISTICS.

This command retrieves a human-readable string containing internal server statistics like uptime and thread counts.

This page is licensed: CC BY-SA / Gnu FDL

COM_CHANGE_USER

`COM_CHANGE_USER` resets the connection and re-authenticates with the given credentials. The packet is identical to the authentication packet in the connection handshake.

Fields

- int<1> `0x11` : `COM_CHANGE_USER` header.
- string<NUL> username.
- If (`server_capabilities` & `CLIENT_SECURE_CONNECTION`):
  - int<1> length of authentication response.
  - string authentication response.
- Else:
  - string<NUL> authentication response.
- string<NUL> default schema name.
- int<2> client character collation.
- If (`server_capabilities` & `CLIENT_PLUGIN_AUTH`):
  - string<NUL> authentication plugin name.
- If (`server_capabilities` & `CLIENT_CONNECT_ATTRS`):
  - int<lenenc> size of connection attributes.
  - Loop:
    - string<lenenc> key.
    - string<lenenc> value.

Response

Server response is like connection authentication :

- An `OK` packet in case of success OK_Packet.
- An error packet in case of error ERR_Packet.
- Authentication switch:
  - If the client or server doesn't have `PLUGIN_AUTH` capability:

        - Server sends `0xFE` byte.
        - Client sends `old_password`.
    - Else:
        - Server sends Authentication switch request.
        - Client may have many exchanges with the server according to the Plugin.
    - Authentication switch ends with server sending either OK_Packet or ERR_Packet.

If the authentication fails more than three times, all future `COM_CHANGE_USER` commands on the connection will return the `#08S01 Unknown command` error. This is an anti-brute-force mechanism designed to prevent rapid guessing of passwords.

This page is licensed: CC BY-SA / Gnu FDL

COM_CREATE_DB

> **Note:**
>
> This command is deprecated and not used by MariaDB connectors any more. Use the SQL statements CREATE SCHEMA or CREATE DATABASE instead.

This page is licensed: CC BY-SA / Gnu FDL

COM_DEBUG

The `COM_DEBUG` command forces the server to dump debug information to stdout. It requires the SUPER privilege.

Fields

- int<1> `0xOD` : `COM_DEBUG` header.

Response

EOF Packet.

This page is licensed: CC BY-SA / Gnu FDL

COM_DROP_DB

> **Note:**
>
> This command is deprecated and not used by MariaDB connectors any more. Use the SQL statements DROP SCHEMA or DROP DATABASE instead.

This page is licensed: CC BY-SA / Gnu FDL

COM_FIELD_LIST

> **Note:**
>
> This command is deprecated and not used by MariaDB connectors any more. Use the SQL statements SHOW COLUMNS or SELECT FROM INFORMATION_SCHEMA.COLUMNS instead.

Fields

- int<1> `0x04` : `COM_FIELD_LIST` header.

Response

- Zero or more result set rows.
- EOF packet.

This page is licensed: CC BY-SA / Gnu FDL

COM_INIT_DB

`COM_INIT_DB` is used to specify the default schema for the connection.

Fields

- int<1> `0x02` : `COM_INIT_DB` Header.
- string<NUL> schema name.

Response

ERR_Packet or OK_Packet.

Example

```
06 00 00 00 02 74 65 73 74 63                  .....testc
```

This page is licensed: CC BY-SA / Gnu FDL

COM_PING

`COM_PING` permits sending a packet containing one byte to check that the connection is active.

Fields

- int<1> `0x0e` : `COM_PING` header.

Response

OK_Packet.

This page is licensed: CC BY-SA / Gnu FDL

COM_PROCESSLIST

> **Note:**
>
> This command is deprecated and not used by MariaDB connectors any more. Use the SQL statement SHOW PROCESSLIST instead.

This page is licensed: CC BY-SA / Gnu FDL

COM_PROCESS_KILL

Forces the server to terminate a specified connection.

Fields

- int<1> `0xC` `COM_PROCESS_KILL`.
- int<4> process ID.

Response

OK Packet or ERR Packet.

This page is licensed: CC BY-SA / Gnu FDL

COM_QUERY

With the `COM_QUERY` command, the client sends the server an SQL statement to be executed immediately.

Fields

- int<1> `0x03` : `COM_QUERY` header.
- string<EOF> SQL statement.

The SQL statement should be properly escaped. The escape character is usually a backslash (`\`) = `0x5c` . However, if the status flag returned by the last OK Packet had the `NO_BACKSLASH_ESCAPES` bit set, the escape character is a single quote (`'` = `0x60`).

If the escape character is a backslash, the following characters are escaped:

- Single quote (`'` = `0x60`).
- Backslash (`\` = `0x5c`).
- Double quote (`"` = `0x22`).
- Null character (`0x00`).

If the escape character is a single quote, only the single quote (`'` = `0x60`) can be escaped.

Response

The server can answer with 4 different responses that can be differentiated by the first byte (packet header):

- `0xFF` - ERR_Packet if any error occurs.
- `0x00` - OK_Packet when query execution works without result set.
- `0xFB` - LOCAL_INFILE Packet if the query was `LOCAL INFILE ...` .
- Or a result set, when the query returns results (in case of a `SELECT` query, for example).

Example

1b 00 00 00 03 44 52 4f 50 20 54 41 42 4c 45 20	.....DROP TABLE
49 46 20 45 58 49 53 54 53 20 62 75 6c 6b 31	IF EXISTS bulk1

This page is licensed: CC BY-SA / Gnu FDL

COM_QUIT

Using the `COM_QUIT` command, the client tells the server that the connection should be terminated.

Fields

- int<1> `0x01` : `COM_QUIT` header.

Response

Server terminates connection.

This page is licensed: CC BY-SA / Gnu FDL

COM_RESET_CONNECTION

`COM_RESET_CONNECTION` resets a connection without reauthentication.

The command does this:

- Roll back any open transactions.
- Reset transaction isolation level.
- Rset session variables.
- Delete user variables.
- Remove temporary tables.
- Remove all `PREPARE` statements.

Database will *not* be reset to initial value.

Fields

- int<1> `0x1f` : `COM_RESET_CONNECTION` Header

Response

ERR_Packet or OK_Packet

This page is licensed: CC BY-SA / Gnu FDL

COM_SET_OPTION

Enables or disables server option.

Fields

- int<1> `0x1B` `COM_SET_OPTION` .
- int<2> option.

Options

Constant	Value
`MYSQL_OPTION_MULTI_STATEMENTS_ON`	0
`MYSQL_OPTION_MULTI_STATEMENTS_OFF`	1

Response

EOF Packet on success or ERR packet.

This page is licensed: CC BY-SA / Gnu FDL

COM_SHUTDOWN

Shuts down the server. To execute this command, the SHUTDOWN privilege is required.

Fields

- int<1> 0x0A COM_SHUTDOWN
- int<1> option

Options

Constant	Value
`SHUTDOWN_DEFAULT`	0

Response

OK Packet or ERR packet.

This page is licensed: CC BY-SA / Gnu FDL

COM_SLEEP

This command is used inside the server only.

Direction

Used inside server only.

Fields

- int<1> `0x00` : `COM_SLEEP` header.

Response

ERR_Packet.

This page is licensed: CC BY-SA / Gnu FDL

COM_STATISTICS

Get internal server statistics.

Fields

- int<1> `0x09` : `COM_STATISTICS` header.

Response

string<EOF> human-readable string.

This page is licensed: CC BY-SA / Gnu FDL

3 - Binary Protocol (Prepared Statements)

See COM_STMT_BULK_EXECUTE.

This command executes a bulk insert for a previously prepared statement, using a compact binary format for efficiency.

See COM_STMT_CLOSE.

This command deallocates a prepared statement on the server, freeing up associated resources.

See COM_STMT_EXECUTE.

This command executes a prepared statement using parameter values provided in the binary protocol format.

See COM_STMT_FETCH.

This command fetches rows from an existing result set of a prepared statement that was executed with a cursor.

See COM_STMT_PREPARE.

This command prepares an SQL statement on the server, returning a statement ID and metadata about parameters and columns.

See COM_STMT_RESET.

This command resets the data of a prepared statement on the server, clearing any buffers or previous parameter values.

See COM_STMT_SEND_LONG_DATA.

This command sends long data, such as BLOB or TEXT values, in chunks for a specific parameter of a prepared statement.

See Server Response Packets (Binary Protocol).

This section details the structure of response packets sent by the server when using the binary protocol, particularly for result sets.

This page is licensed: CC BY-SA / Gnu FDL

COM_STMT_BULK_EXECUTE

Executes a bulk insert of a previously prepared statement.

A command that returns a result set returns an error (Error packet).

Direction

Client to server.

Fields

- int<1> `0xfa` : `COM_STMT_BULK_EXECUTE` header.
- int<4> statement id.
- int<2> bulk flags.
- If (`bulk_flag` & `SEND_TYPES_TO_SERVER`):
  - For each parameter:
    - byte<1>: field type.
    - byte<1>: parameter type flag.
- Until end of packet:
  - For each parameter (i.e `param_count` times):
    - byte<1>: parameter indicator.
    - If indicator == `NONE` :
      - byte<n> : binary parameter value.

Flags

Bulk Flags

Flag	Value	Details
`SEND_UNIT_RESULTS`	64	Return generated affected rows and auto-increment IDs as a result set (only when server supports `MARIADB_CLIENT_BULK_UNIT_RESULTS` capability).
`SEND_TYPES_TO_SERVER`	128	Send types to server.

Parameter Type Flag

128	unsigned

Parameter Indicator

0	`NONE`	Value follow
1	`NULL`	Value is null
2	`DEFAULT`	For `INSERT` / `UPDATE`, value is default
3	`IGNORE`	Value is default for insert, Is ignored for update

Response

The server can answer with 3 different responses:

- `0xff`: ERR_Packet if any errors occur.
- `0x00`: OK_packet when query execution works without result set.
- A result set containing affected rows and auto-increment IDs when bulk flag `SEND_UNIT_RESULTS` is set.

This page is licensed: CC BY-SA / Gnu FDL

COM_STMT_CLOSE

Closes a previously prepared statement.

Direction

Client to server.

Implemented by

- mysql_stmt_close().
- mysql_stmt_prepare().

Fields

- int<1> `0x19` `COM_STMT_CLOSE` header.
- int<4> statement id.

Example

```
05 00 00 00 19 04 00 00 00
```

Response

No response from server.

This page is licensed: CC BY-SA / Gnu FDL

COM_STMT_EXECUTE

Executes a previously prepared statement.

If specific data is large, it can be sent separately prior to this command (see COM_STMT_SEND_LONG_DATA).

If a statement is re-executed without changing the bind types, the types do not need to be sent to the server again.

Direction

Client to server.

Fields

- int<1> `0x17` : `COM_STMT_EXECUTE` header.
- int<4> statement ID.
- int<1> flags.
- int<4> Iteration count (always 1).
- If (`param_count > 0`)
  - byte<(param_count + 7)/8> null bitmap.
  - byte<1>: send type to server (0 / 1).
  - If (send type to server):
    - For each parameter:
      - byte<1>: field type.
      - byte<1>: parameter flag.
  - For each parameter (for instance, `param_count` times):
    - If parameter is not null:
      - byte binary parameter value.

Statement ID

Statement ID is the identifier of the prepared statement (from COM_STMT_PREPARE answer).

Specific “-1” Statement ID Value

Value `-1` (`0xFFFFFFFF`) can be used to indicate to use the last statement prepared on current connection if no `COM_STMT_PREPARE` has failed since.

This permit pipelining :

- Send `COM_STMT_PREPARE` + `COM_STMT_EXECUTE` with statement ID `-1`.
- Read `COM_STMT_PREPARE` + `COM_STMT_EXECUTE` response.

In case `COM_STMT_PREPARE` returns an error, `COM_STMT_EXECUTE` returns an error that statement ID `-1` is unknown. This permits to avoid much of the network latency.

Flag

0	no cursor
1	read only
2	cursor for update
4	scrollable cursor

Cursors

If the flags of the `COM_STMT_EXECUTE` request a cursor to be opened, the returned result only contains the column definitions and the `EOF` that terminates it, and the result set rows are fetched using separate `COM_STMT_FETCH` commands.

Whether a cursor is actually opened is indicated by the `SERVER_STATUS_CURSOR_EXISTS` bit in the first EOF packet in the response to the `COM_STMT_EXECUTE`. If it is **not** set in, no cursor is opened and a normal result set is returned.

Parameter Flag

Parameter type flag byte:

128	**unsigned**

NULL-Bitmap

The `NULL-Bitmap` indicates if parameters are null (one bit per parameter). If the parameter is `NULL`, the bit is set in the bitmap and the parameter value is not sent.

The size in bytes of the `NULL-bitmap` can be calculated with: `(parameter number + 7) / 8`

Binary Parameter Encoding

The encoding of the `COM_STMT_EXECUTE` parameters are the same as the encoding of the binary result sets.

Response

The server can answer with 3 different responses:

- `0xff` : ERR_Packet if any errors occur.
- `0x00` : OK_packet when query execution works without result set.
- One (or more) result set, when query execution return rows (in case of `SELECT` query, for example).

This page is licensed: CC BY-SA / Gnu FDL

COM_STMT_FETCH

Fetch rows from a prepared statement.

A COM_STMT_EXECUTE with a non-zero cursor flag must have been successfully executed before any `COM_STMT_FETCH` commands can be executed.

Fields

- int<1> 0x1C COM_STMT_FETCH header.
- int<4> statement id.
- int<4> number of rows to fetch.

Response

Returns one or more binary result set rows followed by an EOF packet.

This page is licensed: CC BY-SA / Gnu FDL

COM_STMT_PREPARE

Prepares a statement on the server.

> **Note:**
>
> Not all statements can be prepared. See PREPARE for a list of statements that can be prepared.

Implemented by

- mariadb_stmt_execute_direct()
- mysql_stmt_prepare()

Fields

- int<1> `0x16` `COM_STMT_PREPARE` header
- string<EOF> SQL Statement

Example

```
1F 00 00 00 16 53 45 4C 45 43 54 20 2A 20 46 52  .....SELECT * FR
1F 4D 20 74 65 73 74 5F 62 69 6E 64 5F 72 65 73  OM test_bind_res
75 6C 74                                         ult
```

Response

If something goes wrong, the server sends an ERR_Packet. If the command succeeds, different packets are received:

- COM_STMT_PREPARE_OK.
- If number of parameters (count of `?` placeholders) > `0`:
  - For each parameter:
    - column definition packet.
  - If ! `DEPRECATE_EOF` eof_packet.
- If prepared statement returns result set and number of result set columns > `0`:
  - For each column:
    - column definition packet.
  - If ! `DEPRECATE_EOF` eof_packet.

COM_STMT_PREPARE_OK

- int<1> `0x00` `COM_STMT_PREPARE_OK` header.
- int<4> statement ID.
- int<2> number of columns in the returned result set (or 0 if statement does not return result set).
- int<2> number of prepared statement parameters (‘?’ placeholders).
- string<1> -not used-
- int<2> number of warnings.

This page is licensed: CC BY-SA / Gnu FDL

COM_STMT_RESET

Resets a prepared statement on the client and server to state after preparing.

Fields

- int<1> `0x1A` `COM_STMT_RESET` header.
- int<4> Statement ID.

Response

ERR_Packet or OK_Packet.

Example

```
05 00 00 00 1A 04 00 00 00
```

This page is licensed: CC BY-SA / Gnu FDL

COM_STMT_SEND_LONG_DATA

When data for a specific column is large, it can be sent separately to avoid the limitation of `max_allowed_packet` (see packet splitting).

Multiple `COM_STMT_SEND_LONG_DATA` commands with the same column ID append the data. `COM_STMT_SEND_LONG_DATA` is sent before COM_STMT_EXECUTE.

Fields

- int<1> `0x18` `COM_STMT_SEND_LONG_DATA` header.
- int<4> statement id.
- int<2> parameter number.
- byte<EOF> data.

Response

Server doesn't send response.

This page is licensed: CC BY-SA / Gnu FDL

Server Response Packets (Binary Protocol)

The following response packets will be sent from the server to client in binary protocol only. They may be mixed with packets from the text protocol, like OK_PACKET, `PACKET_METADATA` or ERR_PACKET.

See PACKET_BINDATA.

This page is licensed: CC BY-SA / Gnu FDL

PACKET_BINDATA

In contrast to the text protocol, the binary protocol transfers data according to the format of the field types returned in `PACKET_METADATA`.

Field type	Representation
MYSQL_TYPE_BIT	str_LEC
MYSQL_TYPE_BLOB	str_LEC
MYSQL_TYPE_DATE MYSQL_TYPE_DATETIME MYSQL_TYPE_TIMESTAMP	int_11 (default)int_7 (no microseconds)int_4 (no time values)int_0 (no values)
MYSQL_TYPE_DECIMAL	str_LEC
MYSQL_TYPE_DOUBLE	int_8
MYSQL_TYPE_ENUM	str_LEC
MYSQL_TYPE_FLOAT	int_4
MYSQL_TYPE_GEOMETRY	str_LEC
MYSQL_TYPE_INT24	int_4
MYSQL_TYPE_JSON	str_LEC
MYSQL_TYPE_LONGLONG	int_8
MYSQL_TYPE_LONG_BLOB	str_LEC
MYSQL_TYPE_LONG	int_4
MYSQL_TYPE_MEDIUM_BLOB	str_LEC
MYSQL_TYPE_NEWDECIMAL	str_LEC

Field type	Representation
MYSQL_TYPE_NULL	stored in bitmap
MYSQL_TYPE_SET	str_LEC
MYSQL_TYPE_STRING	str_LEC
MYSQL_TYPE_SHORT	int_2
MYSQL_TYPE_TINY_BLOB	str_LEC
MYSQL_TYPE_TINY	int_1
MYSQL_TYPE_VARCHAR	str_LEC
MYSQL_TYPE_VAR_STRING	str_LEC
MYSQL_TYPE_YEAR	int_4

Fields

```
(column_count + 7)/8   null bitmap
while (!eof) {
  for (i=0; i < column_count; i++)
  {
    data (length depends on the data type)
  }
}
```

This page is licensed: CC BY-SA / Gnu FDL

4 - Server Response Packets

For most commands which the client sends to the server, the server returns the following response packets:

See EOF_Packet.

Marks the end of a result set and returns status and warnings.

See ERR_Packet.

Indicates that an error occurred.

See LOCAL_INFILE_Packet.

If the client sends a `LOAD DATA LOCAL INFILE` statement via com_query, the server responds with `LOCAL_INFILE_Packet` to tell the client to send a specified file to the server.

See OK_Packet.

Sent by the server to the client. Indicates a successful completion of a command sent by the client before.

See Result Set Packets.

The server sends multiple packets as part of a result set.

See Result Set Row.

Data representing a database result set unit.

This page is licensed: CC BY-SA / Gnu FDL

EOF_Packet

The `EOF_Packet` marks the end of a result set, and returns status and warnings.

When testing for an EOF packet, the packet size must be less than 9 bytes in length. The result set can send data that begin with a `0xfe` byte, but then the packet length will be greater than 9.

Fields

- int<1> `0xfe` : EOF header.
- int<2> warning count.
- int<2> server status.

This page is licensed: CC BY-SA / Gnu FDL

ERR_Packet

`ERR_Packet` indicates that an error occurred.

Fields

- int<1> `ERR_Packet` header = `0xFF`.
- int<2> error code. see error list.
- If (error code == `0xFFFF`) /* progress reporting */:
  - int<1> stage.
  - int<1> max_stage.
  - int<3> progress.
  - string<lenenc> progress_info.
- Else:
  - If (next byte = '#'):
    - string<1> sql state marker '#'.
    - string<5> sql state.
    - string<EOF> human-readable error message.
  - Else:
    - string<EOF> human-readable error message.

> **Warning:**
>
> Note that the `ERR_Packet` is supposed to send a server error to the client. In particular, all error codes in the range 2000 to 2999 and 5000 to 5999 (inclusive) are reserved for client errors, and an ERR packet with such an error code will be considered malformed.

See also

- Progress reporting

This page is licensed: CC BY-SA / Gnu FDL

LOCAL_INFILE_Packet

If the client sends a `LOAD DATA LOCAL INFILE` statement via com_query, the server responds with `LOCAL_INFILE_Packet` to tell the client to send a specified file to the server.

Fields

- int<1> `0xFB` : `LOCAL_INFILE` header.
- string<EOF> filename.

Client Response

The client sends the file as the packet body. If the file is large, the contents are sent in multiple separate packets. After the file is sent, the client must send an empty packet to indicate that no more data will follow.

Once the client has finished sending the file, the server will respond with an OK_packet or an ERR_packet.

This page is licensed: CC BY-SA / Gnu FDL

OK_Packet

`OK_Packet` is sent by the server to the client and indicates a successful completion of a command sent by the client before.

Direction

Server to client.

Fields

- int<1> `0x00` : `OK_Packet` header or (`0xFE` if `CLIENT_DEPRECATE_EOF` is set).
- int<lenenc> affected rows.
- int last insert id.
- int<2> server status.
- int<2> warning count.
- If packet has more data:
  - string info.
  - If (status flags & `SERVER_SESSION_STATE_CHANGED`) and `session_tracking_supported` (see CLIENT_SESSION_TRACK):
    - string session state info.

The length-encoded info string is not always included in the packet. Check the length of the packet to detect if there is data after the warning count. For the first `OK_Packet` in the connection it contains (if present) the SSL certificate verification signature. For the following `OK_Packets` it contains (if present) various human-readable information.

Server status flag

Values of server status flags:

Flag	Value	Details
SERVER_STATUS_IN_TRANS	1	A transaction is currently active.
SERVER_STATUS_AUTOCOMMIT	2	Autocommit mode is set.
SERVER_MORE_RESULTS_EXISTS	8	More results exists (more packets will follow).
SERVER_QUERY_NO_GOOD_INDEX_USED	16	Set if EXPLAIN would've shown Range checked for each record.
SERVER_QUERY_NO_INDEX_USED	32	The query did not use an index.
SERVER_STATUS_CURSOR_EXISTS	64	When using `COM_STMT_FETCH` , indicate that current cursor still has result.
SERVER_STATUS_LAST_ROW_SENT	128	When using `COM_STMT_FETCH` , indicate that current cursor has finished to send results.
SERVER_STATUS_DB_DROPPED	1<<8	Database has been dropped.
SERVER_STATUS_NO_BACKSLASH_ESCAPES	1<<9	Current escape mode is “no backslash escape”.

Flag	Value	Details
SERVER_STATUS_METADATA_CHANGED	1<<10	A DDL change did have an impact on an existing `PREPARE` (an automatic re-prepare has been executed).
SERVER_QUERY_WAS_SLOW	1<<11	The query was slower than long_query_time.
SERVER_PS_OUT_PARAMS	1<<12	This result set contains stored procedure output parameter.
SERVER_STATUS_IN_TRANS_READONLY	1<<13	Current transaction is a read-only transaction.
SERVER_SESSION_STATE_CHANGED	1<<14	Session state change. See session change type for more information.

Session state info

- While packet has remaining data:
  - int<1> session change type.
  - If (session-change-type != `SESSION_TRACK_STATE_CHANGE`):
    - int total length.
  - string session data change.

Session change type

Change Type	Flag
0	SESSION_TRACK_SYSTEM_VARIABLES
1	SESSION_TRACK_SCHEMA
2	SESSION_TRACK_STATE_CHANGE
3	SESSION_TRACK_GTIDS
4	SESSION_TRACK_TRANSACTION_CHARACTERISTICS
5	SESSION_TRACK_TRANSACTION_STATE

Session Data Change

Each type of data has its own kind of format:

SESSION_TRACK_SCHEMA

- string new current schema.

SESSION_TRACK_SYSTEM_VARIABLES

While there is remaining data:

- string variable data.

For each variable data:

- string variable name.
- string variable value.

Note:

Possible tracked variables list is tracked by session_track_system_variables special variable value description:

- redirect_url: format is `mariadb/mysql:[<user>[:<password>]@]<host>[:<port>]/[<db>[?<opt1>=<value1>[&<opt2>=<value2>]]]`. Possible options:
  - `ttl` : cache timeout in ms to remember redirection, in order to reconnect directly to new host. 0=no caching

SESSION_TRACK_STATE_CHANGE

Indicates if session state changes occurred. The value is represented as `1`.

- string `1` if session state tracking was enabled.

SESSION_TRACK_GTIDS

This tracker is not implemented by MariaDB.

SESSION_TRACK_TRANSACTION_CHARACTERISTICS

- string Transaction characteristics.

Transaction characteristics is the set of SQL statements that reproduce the type and state of the current transaction. It can consist of the following SQL statements:

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION;
START TRANSACTION WITH CONSISTENT SNAPSHOT;
START TRANSACTION WITH CONSISTENT SNAPSHOT, READ ONLY;
START TRANSACTION WITH CONSISTENT SNAPSHOT, READ WRITE;
SET TRANSACTION READ ONLY;
SET TRANSACTION READ WRITE;
XA START <XA specification>;
```

SESSION_TRACK_TRANSACTION_STATE

- string Transaction state string.

The transaction state string is always 8 characters long. The characters, in order, are:

1. No transaction: `_` Explicit transaction: `T` Implicit transaction: `I`
2. Transaction read safe: `_` Transaction read unsafe: `r`
3. Unknown transaction type: `_` Read-only transaction: `R`
4. Transaction write safe: `_` Transaction write unsafe: `w`
5. Unknown transaction type: `_` Read-write transaction: `W`
6. Transaction statement safe: `_` Transaction statement unsafe: `s`
7. Transaction does not have resultsets: `_` Transaction with result sets: `S`
8. No locked tables: `_` Tables have been locked: `L`

Notes

- To determine if session tracking is enabled, check if the CLIENT_SESSION_TRACK flag is set in `server_capabilities`.

This page is licensed: CC BY-SA / Gnu FDL

Result Set Packets

MariaDB Server sends the following packets as part of a result set.

A result set consists of different packets:

- Result set metadata.
  - 1 column count packet.
- If not (`MARIADB_CLIENT_CACHE_METADATA` capability set) `OR` (send metadata == 1):
  - For each column (for instance, `column_count` times):
    - Column Definition packet.
- If not (`CLIENT_DEPRECATE_EOF` capability set) EOF_Packet.
- n result set row.
- If error:
  - ERR_Packet.
- Else:
  - If `CLIENT_DEPRECATE_EOF` capability:
    - OK_Packet with a `0xFE` header.
  - Else EOF_Packet.

> **Warning:**
>
> It is unsafe to assume that any packet with a `0xFE` header is an OK packet (OK_Packet) or an EOF packet (EOF_Packet), because result set row packets (ResultsetRow) can also begin with `0xFE` when using the text protocol with a field length greater than `0xFFFFFF`.
> To safely confirm that a packet with a `0xFE` header is an OK packet (OK_Packet) or an EOF packet (EOF_Packet), you must also check that the packet length is less than `0xFFFFFF`.

Column Count Packet

The column count packet describes the number of columns in the result set. It uses the following format:

- int<lenenc> column count.
- If (`MARIADB_CLIENT_CACHE_METADATA` capability set):
  - int<1> metadata follows (0 / 1).

The metadata indicator byte is only present if both the client and the server declare the `MARIADB_CLIENT_CACHE_METADATA` capability.

If the metadata byte is set to `1`, the normal metadata follows the column definitions. If the metadata byte is set to `0`, the Column Count Packet is immediately followed by the second EOF packet (EOF_Packet) or the result set rows if the `CLIENT_DEPRECATE_EOF` capability is set.

Column Definition Packet

A column definition packet describes a column in the result set. It uses the following format:

- string<lenenc> catalog (always 'def').
- string<lenenc> schema.
- string<lenenc> table alias.
- string<lenenc> table.
- string<lenenc> column alias.
- string<lenenc> column.
- If extended type supported (see `MARIADB_CLIENT_EXTENDED_METADATA`):

    - string<lenenc> extended metadata.
- int<lenenc> length of fixed fields (=0xC).
- int<2> character set number.
- int<4> max. column size.
- int<1> Field types.
- int<2> Field detail flag.
- int<1> decimals.
- int<2> - unused -

Field Types

The column type field in the column definition packet describes the base type of the column. It also indicates how the values are encoded for `COM_STMT_EXECUTE` parameters and binary result set rows.

Value	Protocol Column Type	Encoding
0	MYSQL_TYPE_DECIMAL	byte encoding
1	MYSQL_TYPE_TINY	TINYINT Binary encoding
2	MYSQL_TYPE_SHORT	SMALLINT Binary encoding
3	MYSQL_TYPE_LONG	INTEGER Binary encoding
4	MYSQL_TYPE_FLOAT	FLOAT Binary encoding
5	MYSQL_TYPE_DOUBLE	DOUBLE Binary encoding
6	MYSQL_TYPE_NULL	Not used, being NULL is indicated by the NULL-bitmap in the result
7	MYSQL_TYPE_TIMESTAMP	TIMESTAMP Binary encoding
8	MYSQL_TYPE_LONGLONG	BIGINT Binary encoding
9	MYSQL_TYPE_INT24	INTEGER Binary encoding
10	MYSQL_TYPE_DATE	TIMESTAMP Binary encoding
11	MYSQL_TYPE_TIME	TIME Binary encoding
12	MYSQL_TYPE_DATETIME	TIMESTAMP Binary encoding
13	MYSQL_TYPE_YEAR	SMALLINT Binary encoding
14	MYSQL_TYPE_NEWDATE	byte encoding
15	MYSQL_TYPE_VARCHAR	byte encoding
16	MYSQL_TYPE_BIT	byte encoding
17	MYSQL_TYPE_TIMESTAMP2	Used only in the replication protocol
18	MYSQL_TYPE_DATETIME2	Used only in the replication protocol
19	MYSQL_TYPE_TIME2	Used only in the replication protocol
245	MYSQL_TYPE_JSON	byte encoding (only used with MySQL, MariaDB uses MYSQL_TYPE_STRING for JSON)
246	MYSQL_TYPE_NEWDECIMAL	byte encoding
247	MYSQL_TYPE_ENUM	byte encoding
248	MYSQL_TYPE_SET	byte encoding
249	MYSQL_TYPE_TINY_BLOB	byte encoding
250	MYSQL_TYPE_MEDIUM_BLOB	byte encoding
251	MYSQL_TYPE_LONG_BLOB	byte encoding
252	MYSQL_TYPE_BLOB	byte encoding

Value	Protocol Column Type	Encoding
253	MYSQL_TYPE_VAR_STRING	byte encoding
254	MYSQL_TYPE_STRING	byte encoding
255	MYSQL_TYPE_GEOMETRY	byte encoding

Field Details Flag

The column details flag describes certain column attributes and whether certain column options are set.

It is a bitmask with the following flags:

Flag Value	Flag Name	Flag Description
1	NOT_NULL	field cannot be null
2	PRIMARY_KEY	field is a primary key
4	UNIQUE_KEY	field is unique
8	MULTIPLE_KEY	field is in a multiple key
16	BLOB	is this field a Blob
32	UNSIGNED	is this field unsigned
64	ZEROFILL_FLAG	is this field a zerofill
128	BINARY_COLLATION	whether this field has a binary collation
256	ENUM	Field is an enumeration
512	AUTO_INCREMENT	field auto-increment
1024	TIMESTAMP	field is a timestamp value
2048	SET	field is a SET
4096	NO_DEFAULT_VALUE_FLAG	field doesn't have default value
8192	ON_UPDATE_NOW_FLAG	field is set to NOW on UPDATE
32768	NUM_FLAG	field is num

> **Note:**
>
> The `BLOB` flag cannot be used to determine if a column has binary data, because BINARY and VARBINARY columns are treated as strings, instead of blobs.
>
> The `BINARY_COLLATION` flag can be used to determine if a string column has binary data.

Extended Metadata

This extended column type information can be used to find out more specific details about the column type:

- For a POINT column, the column type field is `MYSQL_TYPE_GEOMETRY`, but the extended type indicates 'point'.
- For a JSON column, the column type field is `MYSQL_TYPE_STRING`, but the extended type indicates 'json'.
- For an XML (`XMLTYPE`) column, the column type field is `MYSQL_TYPE_LONG_BLOB`, but the extended type indicates 'xml'. This was added in MariaDB 12.3.2.
- While string has data:
  - int<1> data type: 0x00:type, 0x01: format.
  - string<lenenc> value.

This page is licensed: CC BY-SA / Gnu FDL

Result Set Row

A result set row represents a database result set unit, which is usually generated by executing a statement that queries the database. Using COM_STMT_EXECUTE, the result set row is in binary format, otherswise in text format.

Text Result Set Row

- For each column:
  - string<lenenc> column data.

The byte representation of the string according to client character collation.

Binary Result Set Row

- byte<1> `0x00` header.
- byte<(number_of_columns + 7) / 8> NULL-Bitmap.
- For each column:
  - If column value is not null:
    - If `MYSQL_TYPE_DOUBLE` type : DOUBLE Binary encoding
    - If `MYSQL_TYPE_LONGLONG` type : BIGINT Binary encoding
    - If `MYSQL_TYPE_INTEGER` type : INTEGER Binary encoding
    - If `MYSQL_TYPE_MEDIUMINT` type : MEDIUMINT Binary encoding
    - If `MYSQL_TYPE_FLOAT` type : FLOAT Binary encoding
    - If `MYSQL_TYPE_SMALLINT` type : SMALLINTBinary encoding
    - If `MYSQL_TYPE_YEAR` type : YEAR Binary encoding
    - If `MYSQL_TYPE_TINYINT` type : TINYINT Binary encoding
    - If `MYSQL_TYPE_DATE` type : DATE Binary encoding
    - If `MYSQL_TYPE_TIMESTAMP` type : TIMESTAMP Binary encoding
    - If `MYSQL_TYPE_DATETIME` type : TIMESTAMP Binary encoding
    - If `MYSQL_TYPE_TIME` type : TIME Binary encoding
    - If `MYSQL_TYPE_NEWDECIMAL` type : DECIMAL Binary encoding
    - If `MYSQL_TYPE_TINY_BLOB`, `MYSQL_TYPE_MEDIUM_BLOB`, `MYSQL_TYPE_LONG_BLOB`, `MYSQL_TYPE_BLOB`, `MYSQL_TYPE_GEOMETRY`, `MYSQL_TYPE_STRING`, `MYSQL_TYPE_VARCHAR`, `MYSQL_TYPE_VAR_STRING`): byte<lenenc> value

NULL-Bitmap Values

The `NULL`-Bitmap indicates if a parameter for a column is null (one bit per parameter) beginning with the 3rd bit. `NULL`-bitmap size is (number_of_columns + 7) / 8.

DECIMAL Binary Encoding

`DECIMAL` has no fixed size, so will be encoded as string<lenenc>. A `DECIMAL(10,2)` with a value of `-15.5` is stored as:

```
06 45 49 53 46 53 48      . - 1 5 . 5 0
```

DOUBLE Binary Encoding

`DOUBLE` is the IEEE 754 floating-point value in Little-endian format on 8 bytes.

BIGINT Binary Encoding

`BIGINT` is the value in Little-endian format on 8 bytes. Signed is defined by the Column field detail flag.

INTEGER Binary Encoding

`INTEGER` is the value in Little-endian format on 4 bytes. Signed is defined by the Column field detail flag.

MEDIUMINT Binary Encoding

`MEDIUMINT` is similar to `INTEGER` binary encoding, even if `MEDIUM` int is 3-bytes encoded server side. (Last byte will always be 0x00).

FLOAT Binary Encoding

`FLOAT` is the IEEE 754 floating-point value in Little-endian format on 4 bytes.

SMALLINT Binary Encoding

`SMALLINT` is the value in Little-endian format on 2 bytes. Signed is defined by the Column field detail flag.

YEAR Binary Encoding

`YEAR` uses the same format as `SMALLINT`.

TINYINT Binary Encoding

`TINYINT` is the value of 1 byte. Signed is defined by the Column field detail flag.

DATE Binary Encoding

`DATE` uses the same format as `TIMESTAMP` binary encoding, with a data length of `0` for the special '`0000-00-00`' value and `4` for the standard year/month/day format.

TIMESTAMP Binary Encoding

Data is encoded in 8 bytes without fractional seconds, 12 bytes with fractional seconds.

Byte Position	Description
1	data length : 0 for special '0000-00-00 00:00:00' value. 4 with year + month + day of month only 7 for timestamps without fractional seconds 11 with fractional seconds
2-3	year on 2 bytes little-endian format
4	Month (1=january)
5	days of month
6	hour of day (0 if DATE type)
7	minutes (0 if DATE type)
8	seconds (0 if DATE type)
9-12	micro-second on 4 bytes little-endian format (only if data-length is > 7)

TIME Binary Encoding

Data is encoded in 9 bytes without fractional seconds, 13 bytes with fractional seconds.

- int<1> data length: 0 for special '00:00:00' value, 8 without fractional seconds, 12 with fractional seconds.
- If data length > `0`:
  - int<1> `0` for positive time, `1` for negative time.

- int<4> days.
- int<1> hours.
- int<1> minutes.
- int<1> seconds.
- if data length > `8` :
  - int<4> microseconds.

This page is licensed: CC BY-SA / Gnu FDL

Replication Protocol

See 1-Binlog Events.

This section provides an overview of the various events recorded in the binary log, which are the core units of replication data transmission.

See 2-Binlog Event Header.

Every binary log event starts with a standardized header containing metadata such as the timestamp, event type, server ID, and event size.

See 3-Binlog Network Stream.

Describes the continuous packet stream format used to transmit binary log events from the primary server to the replica over the network.

See 4-Semi-Sync Replication.

Explains the handshake and acknowledgement process for semi-synchronous replication, ensuring data is committed on at least one replica.

See 5-Replica Registration.

Details the initialization phase where a replica connects to the primary, authenticates, sends capabilities, and registers for updates.

See ANNOTATE_ROWS_EVENT.

This event accompanies row-based events to provide the original SQL query text, which is useful for auditing and debugging replication.

See BEGIN_LOAD_QUERY_EVENT.

Used during LOAD DATA INFILE operations, this event marks the beginning of the data load and contains the initial query information.

See BINLOG_CHECKPOINT_EVENT.

A marker event indicating a checkpoint in the binary log, used to ensure consistency and safe rotation of log files.

See COM_BINLOG_DUMP.

This command is sent by a replica to the primary server to request the start of the binary log event stream from a specific file and position.

See COM_REGISTER_SLAVE.

This command is used by a replica to register its details, such as server ID, hostname, and port, with the primary server.

See EXECUTE_LOAD_QUERY_EVENT.

This event is used for LOAD DATA INFILE operations, managing the execution phase similar to a QUERY_EVENT but with extra static fields for file handling.

See Fake GTID_LIST Event.

A synthetic event sent by the master after the initial handshake to inform the replica of its current GTID state, it is not written to the binary log.

See Fake ROTATE_EVENT.

An artificial event sent to the replica to indicate the name of the binary log file on the master, ensuring the replica knows which file is being read.

See FORMAT_DESCRIPTION_EVENT.

This descriptor event appears at the start of every binary log file, defining the server version, binlog version, and header lengths for all event types.

See GTID_EVENT.

The GTID_EVENT marks the start of a new transaction event group, associating it with a Global Transaction ID (GTID) and providing commit flags.

See GTID_LIST_EVENT.

Logged during binlog rotation or checkpoints, this event lists the GTIDs present in the binary log to help replicas determine their replication state.

See HEARTBEAT_LOG_EVENT.

A heartbeat event sent over the network by the master when there are no binlog events, ensuring the replica knows the connection is still active.

See INTVAR_EVENT.

This event records integer values for auto-increment columns or the LAST_INSERT_ID function, ensuring that these values are replicated deterministically.

See QUERY_EVENT.

The QUERY_EVENT records text-based SQL statements for statement-based replication, capturing the query string and execution context like the default database.

See RAND_EVENT.

The RAND_EVENT records the two seed values used for the random number generator, ensuring that calls to the RAND() function produce identical results on replicas.

See ROTATE_EVENT.

The ROTATE_EVENT indicates a log rotation, specifying the name of the next binary log file and the position where writing will continue.

See ROWS_EVENT_V1/V2, ROWS_COMPRESSED_EVENT_V1.

These events record row-level changes (WRITE, UPDATE, DELETE) for replication, with versions supporting different column counts and compression.

See START_ENCRYPTION_EVENT.

This event marks the beginning of encrypted data in the binary log, defining the encryption scheme and key version for subsequent events.

See STOP_EVENT.

The STOP_EVENT is written to the binary log when the server shuts down, serving as a marker for a clean stop.

See TABLE_MAP_EVENT.

This event provides a mapping between a table ID and its table definition, preceding row events to interpret the row data correctly.

See USER_VAR_EVENT.

The USER_VAR_EVENT logs the value of a user-defined variable, ensuring that statements using variables replicate consistently.

See XA_PREPARE_LOG_EVENT.

This event records the preparation phase of an XA transaction, storing the XID to support two-phase commit and recovery.

See XID_EVENT.

The XID_EVENT signifies the commit of a transaction, containing the transaction ID (XID) to ensure atomicity across replication.

This page is licensed: CC BY-SA / Gnu FDL

1-Binlog Events

Binary log events, or binlog events, are information about data modification made to a MariaDB server instance stored in the binary log files.

Log File Structure

Each log file contains:

- A 4-byte magic number, followed by a series of events describing data modifications:
  The magic number bytes are `0xfe` `0x62` `0x69` `0x6e` = `0xfe 'b''i''n'` (this is the `BINLOG_MAGIC` constant).
- A series of binlog events.

Event Content

Each event contains the 'header' followed by 'data bytes':

The header bytes provide information about:

- The event type;
- The creation time;
- Which server created the event;
- Flags and so forth.

The data bytes provide information specific to the type of event.

> **Note:**
>
> The first event, FORMAT_DESCRIPTION_EVENT at 'position' 4, is a descriptor event that describes the format used to write events in the file.

The remaining events are interpreted according to the version.

The final event is usually a log-rotation event ROTATE_EVENT that specifies the next binary log filename or a STOP_EVENT written during server shutdown.

Note: in case of a server crash there is no terminating event (no `ROTATE` nor `STOP`).

Example Binlog File (Hexdump -C $file_name)

```
fe 62 69 6e a4 85 9e 59  0f 8c 27 00 00 f5 00 00  |.bin...Y..'.....|
00 f9 00 00 00 00 00 04  00 31 30 2e 31 2e 32 34  |.........10.1.24|
2d 4d 61 72 69 61 44 42  00 6c 6f 67 00 00 00 00  |-MariaDB.log....|
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00 00 00 00 00 00 00 00  00 00 00 a4 85 9e 59 13  |..............Y.|
38 0d 00 08 00 12 00 04  04 04 04 12 00 00 dd 00  |8...............|
04 1a 08 00 00 00 08 08  08 02 00 00 00 0a 0a 0a  |................|
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
04 13 04 00 01 ab 5b a2  e0 a4 85 9e 59 a3 8c 27  |......[.....Y..'|
00 00 2b 00 00 00 24 01  00 00 00 00 01 00 00 00  |..+...$.........|
00 00 00 00 8c 27 00 00  00 0e 00 00 00 00 00 00  |.....'..........|
...
...
```

Example From mysqlbinlog

```
DELIMITER /*!*/;
# at 4
#170824  9:52:04 server id 10124  end_log_pos 249 CRC32 0xe0a25bab  Start: binlog v 4, server v 10.1.24-
MariaDB created 170824  9:52:04 at startup
ROLLBACK/*!*/;
BINLOG '
pIWeWQ+MJwAA9QAAAPkAAAAAAQAMTAuMS4yNC1NYXJpYURCAGxvZwAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAACkhZ5ZEzgNAAgAEgAEBAQEEgAA3QAEGggAAAAICAgCAAAACgoKAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAEEwQAAatbouA=
'/*!*/;
# at 249
#170824  9:52:04 server id 10124  end_log_pos 292 CRC32 0xb6d8f0a8  Gtid list [0-10124-3584]
# at 292
#170824  9:52:04 server id 10124  end_log_pos 334 CRC32 0xf2dc685f  Binlog checkpoint log-bin.000011
# at 334
#170824  9:52:13 server id 10124  end_log_pos 376 CRC32 0xe958a0ae  GTID 0-10124-3585 trans
...
...
```

This page is licensed: CC BY-SA / Gnu FDL

2-Binlog Event Header

All the binlog events stored in a binary log file have a common structure:

- An event header;
- Event data.

Event Header Structure, 19 Bytes

- uint<4> Timestamp (creation time).
- uint<1> Event Type (type_code).
- uint<4> Server_id (server which created the event).
- uint<4> Event Length (header + data).
- uint<4> Next Event position.
- uint<2> Event flags.

> **Note:**
>
> **Note**: if `CRC32` is in use, the Event Length is 4 bytes bigger in size. The 4 bytes `CRC32` are written at the end of the event (just after the last 'data' byte).

Encrypted Binlog Events

For encrypted binlog events, only the event length is in plaintext, and everything else is encrypted.

To decrypt the binlog event:

- Store the event length in memory;
- Move the timestamp into the event length position;
- Decrypt the whole payload except the first four bytes;
- Move the timestamp back to its original position;
- Copy the original event length back to its position.

Regardless of the cipher used to encrypt the binlogs, the encrypted data are the same size as the original unencrypted event. For events that are encrypted in CBC mode, and whose length is not a multiple of the cipher block size, the final partial block is encrypted using a form of residual block termination:

- Encrypt the current IV of the binlog file in ECB mode;
- `XOR` the remaining bytes with the encrypted IV.

Event Type

Hex	Event type description
0x02	QUERY_EVENT
0x03	STOP_EVENT
0x04	ROTATE_EVENT
0x10	XID_EVENT
0x0d	RAND_EVENT
0x0e	USER_VAR_EVENT
0x0f	FORMAT_DESCRIPTION_EVENT
0x13	TABLE_MAP_EVENT
0x1b	HEARTBEAT_LOG_EVENT
0xa0	ANNOTATE_ROWS_EVENT
0xa1	BINLOG_CHECKPOINT_EVENT
0xa2	GTID_EVENT
0xa3	GTID_LIST_EVENT
0xa4	START_ENCRYPTION_EVENT
0xa5	QUERY_COMPRESSED_EVENT
0xa6	WRITE_ROWS_COMPRESSED_V1
0xa7	UPDATE_ROWS_COMPRESSED_V1
0xa8	DELETE_ROWS_COMPRESSED_V1
0xa9	WRITE_ROWS_V1
0xaa	UPDATE_ROWS_V1
0xab	DELETE_ROWS_V1

Fake Events

These are generated on the fly, never written:

Hex	Event type description
0x04	FAKE_ROTATE_EVENT
0xa3	FAKE_GTID_LIST_EVENT

Event Flag

Hex	Event flag description
0x0001	`LOG_EVENT_BINLOG_IN_USE_F` This flag only makes sense for `Format_description_log_event`. It is set when the event is written, and *reset* when a binlog file is closed (yes, it's the only case when MySQL modifies already written part of binlog). Thus it is a reliable indicator that binlog was closed correctly.
0x0002	`LOG_EVENT_FORCED_ROTATE_F` (unused).
0x0004	`LOG_EVENT_THREAD_SPECIFIC_F` If the query depends on the thread (for example: `TEMPORARY TABLE`).
0x0008	`LOG_EVENT_SUPPRESS_USE_F` Suppress the generation of 'USE' statements before the actual statement. This flag should be set for any events that does not need the current database set to function correctly. Most notable cases are '`CREATE DATABASE`' and '`DROP DATABASE`'.
0x0010	`LOG_EVENT_UPDATE_TABLE_MAP_VERSION_F` (unused).
0x0020	`LOG_EVENT_ARTIFICIAL_F` Artificial events are created arbitrarily and not written to binary log.These events should not update the master log position when slave SQL thread executes them.
0x0040	`LOG_EVENT_RELAY_LOG_F` Events with this flag set are created by slave IO thread and written to relay log.
0x0080	`LOG_EVENT_IGNORABLE_F` For an event, 'e', carrying a type code, that a slave,'s', does not recognize, 's' will check 'e' for `LOG_EVENT_IGNORABLE_F`, and if the flag is set, then 'e'is ignored. Otherwise, 's' acknowledges that it has found an unknown event in the relay log.
0x0100	`LOG_EVENT_NO_FILTER_F` (no description yet).
0x0200	`LOG_EVENT_MTS_ISOLATE_F` (no description yet).
0x8000	`LOG_EVENT_SKIP_REPLICATION_F` Flag set by application creating the event (with `@@skip_replication`);the replica skips replication of such events, if `--replicate-events-marked-for-skip` is not set to `REPLICATE`. This is a MariaDB flag; we allocate it from the end of the available values to reduce risk of conflict with new MySQL flags.

Event Header Example of FORMAT_DESCRIPTION_EVENT

This is the first event in the binlog file at pos 4:

```
a4 85 9e 59 0f 8c 27 00  00 f5 00 00 00 f9 00 00  ...Y..'.........
00 00 00 04 00 31 30 2e  31 2e 32 34 2d 4d 61 72  .....10.1.24-Mar
69 61 44 42 00 6c 6f 67  00 00 00 00 00 00 00 00  iaDB.log....
...
...
```

Interpretation of First 19 Bytes of the Event (the Event Header)

- a4 85 9e 59 [4] Timestamp => 59 9e 85 a4 => 1503561124 = 2017-08-24 09:52:04
- 0f [1] Event Type = 0x0f = FORMAT_DESCRIPTION_EVENT
- 8c 27 00 00 [4] Server_id => 00 00 27 8c = 10124
- f5 00 00 00 [4] Event length => 00 00 00 f5 => 245
- f9 00 00 00 [4] Next Event pos => 00 00 00 f9 => 249 (pos 4 + event size)

- 00 00 [2] Event flags = 0

This page is licensed: CC BY-SA / Gnu FDL

3-Binlog Network Stream

The binary log events stored in a binary log file can be sent over the network in order to replicate data changes from the master server (where data changes are written in binary logs) to replica servers which apply data changes to their own databases.

The MariaDB replica replication protocol consists of:

- A registration phase to master;
- Events receiving (the master sending data when changes are available).

> **Note:**
>
> This section is related to **events sending only**.

Binlog Network streams are requested with `COM_BINLOG_DUMP` , and each Binlog Event is prepended with a status byte. The data sent over network is then MariaDB network protocol (4 bytes) + 1 byte status flag + event data.

MariaDB network protocol 4 bytes are:

- uint<3> packet length (the sent binlog event can be up to 2^24 - 1 - 1 data bytes).
- uint<1> packet sequence byte<1>(0 to 255).

Replication protocol status byte:

- uint<1> OK (0) or ERR (ff) or End of File, EOF, (fe).

> **Warning:**
>
> Due to the 1 byte status flag, the effective data payload is event_size + 1. This means than an event of exactly 16M bytes (2 ^ 24 - 1) cannot be sent in one transmission: it requires 2 packets instead.

```
packet #n:   3 bytes length + sequence + status + [event_header + (event data - 1)]
packet #n+1: 3 bytes length + sequence + last byte of the event data.
```

> **Note:**
>
> The remaining bytes of a large event transmission are always sent **without** a status flag and binlog event header, but rather just as network packet header + data.

Example of an Event Transmission HEARTBEAT_LOG_EVENT

```
T 127.0.0.1:8808 -> 127.0.0.1:57157 [AP]
  23 00 00 04 00 00 00 00    00 1b 67 2b 00 00 22 00    '.........g+..&.
  00 00 ed 01 00 00 20 00    66 6f 6f 2d 62 69 6e 2e    ...... .log-bin.
  31 30 30 30 31 33 39                                  1000139
```

Network Replication Protocol, 5 Bytes

- packet size [3] = 23 00 00 => 00 00 23 => 35 (ok byte + event size).
- pkt sequence [1] = 04.
- `OK` indicator [1] = 0 (`OK`).

Heartbeat event

- Header, 19 bytes.
- Content, string<EOF>.

This page is licensed: CC BY-SA / Gnu FDL

4-Semi-Sync Replication

Regular MariaDB replication is asynchronous. MariaDB also includes semisynchronous replication semi-synchronous Binlog Event.

Event Header Changes

If the user variable `@rpl_semi_sync_slave` is set, 2 extra bytes are added after the status byte of a binlog network stream and before the normal binlog event header.

- uint<1> semi-sync indicator, always `0xef`.
- uint<1> semi-sync flags, either `0x00` (no `ACK`) or `0x01` (`ACK`).

Note : The packet size, as in the network protocol header, is then: `event_size + 1 byte status + 2 bytes semi-sync replication`.

The MariaDB server sets the user variable whenever it is starting replication. For MariaDB Connector/C , the following query must be executed before the call to `mariadb_rpl_open()` is made to enable semi-sync replication.

```
SET @rpl_semi_sync_slave=1
```

If the semi-sync flag is set to `0x01`, the master waits for a Semi Sync `ACK` packet from the slave and when the Semi Sync `ACK` is seen, the master acknowledges the client which has issued the transaction with a standard `OK_Packet` or a `ERR_Packet`.

The master can then write the transaction to the binary log and send the next events to the replica.

Note : The master only requests Semi Sync ACKs if rpl_semi_sync_master_enabled is enabled. If it is not enabled, the semi-sync flag will always be `0x00`.

Semi Sync ACK Details

This event is sent by the replica **only if** the semi-sync flag is set to `0x01`.

- uint<1> semi-sync indicator, always 0xef;
- uint<8> the next position of received event;
- string<EOF> binlog file name.

> **Note:**
>
> This packet sent by the replica never includes the `CRC32`.

Sending an `ACK` when the semi-sync flag is set to `0x0` causes an error, and the connection is closed.

Example of Heartbeat Event With Semi-Sync Protocol and CRC32

We can see:

- 2a 00 00 [3 bytes] packet size.
- 06 [1] sequence.
- 00 [1] status byte = 00 => OK.
- ef 00 [2] bytes => semi sync indicator (0xef) and semi-sync flag (00).

- The ef 00 2 bytes after the OK byte 00.

```
T 127.0.0.1:23240 -> 127.0.0.1:41054 [AP]
  2a 00 00 06 00 ef 00 00    00 00 00 1b d9 27 00 00    *............'..
  27 00 00 00 79 04 00 00    00 00 6d 79 73 71 6c 2d    '...y.....mysql-
  62 69 6e 2e 30 30 30 30    33 34 ed ef e1 f0          bin.000034....
```

Example of XID_EVENT, With CRC32

The master sets the Semi-Sync `ACK` request in the `XID_EVENT` event:

We see the 2 semi sync bytes: **ef** and **01**. The latter, being 1, means the replica server must send the Semi Sync `ACK` packet.

We also see in the binlog event header:

- Event Type [1] = 10 `XID_EVENT`.
- Next Event pos [4] = 4a 05 00 00 => 1354.

```
22 00 00 0c 00 ef 01 17  d0 37 5a 17 d0 37 5a 10  ".........7Z.
d9 27 00 00 1f 00 00 00  4a 05 00 00 00 00 6f 00  .?......J.....o.
00 00 00 00 00 00 44 30  aa fc                    ......D0..
```

Example of Semi-Sync ACK

This is sent by the replica server after the `XID_EVENT` receiving.

We see:

- The semi sync indicator [1] = 0xef, sent before anything else.
- The Next Event position [8] = 4a 05 00 00 00 00 00 00 => 1354 which is the next position of the XID_EVENT above.
- The binlog filename = mysql-bin.000034.

Please note:

- There is no terminating `CRC32`.
- The packet sequence now start starts from 0.

```
19 00 00 00 ef 4a 05 00    00 00 00 00 00 6d 79 73    .....J.......mys
  71 6c 2d 62 69 6e 2e 30    30 30 30 33 34             ql-bin.000034
```

This page is licensed: CC BY-SA / Gnu FDL

5-Replica Registration

Overview

The replica server, when properly configured with `CHANGE MASTER TO ...` , can start MariaDB replication with the START REPLICA statement.

After authentication, some COM_QUERY packets are exchanged before sending COM_REGISTER_SLAVE and COM_BINLOG_DUMP.

The following `COM_QUERY` packets come from MariaDB 10.X replicas using GTID:

- `SELECT UNIX_TIMESTAMP()`;
- `SHOW VARIABLES LIKE 'SERVER_ID'`;
- `SET @master_heartbeat_period` = 30000001024;

- `SET @master_binlog_checksum= @@global.binlog_checksum;`
- `SELECT @master_binlog_checksum;`
- `SET @mariadb_slave_capability=4;`
- `SELECT @@GLOBAL.gtid_domain_id GTID` registration: domain ID;
- `SET @slave_connect_state='0-10201-9868'` GTID registration: the requested GTID;
- `SET @slave_gtid_strict_mode=0` GTID registration: strict_mode;
- `SET @slave_gtid_ignore_duplicates=0` GTID registration: ignore_duplicates.

Then `COM_REGISTER_SLAVE` completes the registration.

The `COM_BINLOG_DUMP` marks the request of binlog events stream.

> **Note:**
>
> If semi-sync is in use, the request for the network protocol change is sent between `COM_REGISTER_SLAVE` and `COM_BINLOG_DUMP`.

Example Using 'ngrep'

COM_REGISTER_SLAVE, Semi-Sync and COM_BINLOG_DUMP:

```
T 127.0.0.1:42158 -> 127.0.0.1:23240 [AP]
  1a 00 00 00 15 75 27 00    00 08 53 42 73 6c 61 76    .....u'...SBslav
  65 31 00 00 c9 5a 00 00    00 00 00 00 00 00          e1...Z........

T 127.0.0.1:23240 -> 127.0.0.1:42158 [AP]
  07 00 00 01 00 00 00 02    00 00 00                   ...........

T 127.0.0.1:42158 -> 127.0.0.1:23240 [AP]
  33 00 00 00 03 53 48 4f    57 20 56 41 52 49 41 42    3....SHOW VARIAB
  4c 45 53 20 4c 49 4b 45    20 27 72 70 6c 5f 73 65    LES LIKE 'rpl_se
  6d 69 5f 73 79 6e 63 5f    6d 61 73 74 65 72 5f 65    mi_sync_master_e
  6e 61 62 6c 65 64 27                                  nabled'

T 127.0.0.1:23240 -> 127.0.0.1:42158 [AP]
  01 00 00 01 02 64 00 00    02 03 64 65 66 12 69 6e    .....d....def.in
  66 6f 72 6d 61 74 69 6f    6e 5f 73 63 68 65 6d 61    formation_schema
  11 53 45 53 53 49 4f 4e    5f 56 41 52 49 41 42 4c    .SESSION_VARIABL
  45 53 11 53 45 53 53 49    4f 4e 5f 56 41 52 49 41    ES.SESSION_VARIA
  42 4c 45 53 0d 56 61 72    69 61 62 6c 65 5f 6e 61    BLES.Variable_na
  6d 65 0d 56 41 52 49 41    42 4c 45 5f 4e 41 4d 45    me.VARIABLE_NAME
  0c 08 00 40 00 00 00 fd    01 00 00 00 00 5d 00 00    ...@.........]..
  03 03 64 65 66 12 69 6e    66 6f 72 6d 61 74 69 6f    ..def.informatio
  6e 5f 73 63 68 65 6d 61    11 53 45 53 53 49 4f 4e    n_schema.SESSION
  5f 56 41 52 49 41 42 4c    45 53 11 53 45 53 53 49    _VARIABLES.SESSI
  4f 4e 5f 56 41 52 49 41    42 4c 45 53 05 56 61 6c    ON_VARIABLES.Val
  75 65 0e 56 41 52 49 41    42 4c 45 5f 56 41 4c 55    ue.VARIABLE_VALU
  45 0c 08 00 00 08 00 00    fd 01 00 00 00 00 05 00    E...............
  00 04 fe 00 00 22 00 20    00 00 05 1c 72 70 6c 5f    .....". ....rpl_
  73 65 6d 69 5f 73 79 6e    63 5f 6d 61 73 74 65 72    semi_sync_master
  5f 65 6e 61 62 6c 65 64    02 4f 4e 05 00 00 06 fe    _enabled.ON.....
  00 00 22 00                                           ..".

T 127.0.0.1:42158 -> 127.0.0.1:23240 [AP]
  1c 00 00 00 03 53 45 54    20 40 72 70 6c 5f 73 65    .....SET @rpl_se
  6d 69 5f 73 79 6e 63 5f    73 6c 61 76 65 3d 20 31    mi_sync_slave= 1

T 127.0.0.1:23240 -> 127.0.0.1:42158 [AP]
  07 00 00 01 00 00 00 02    00 00 00                   ...........

T 127.0.0.1:42158 -> 127.0.0.1:23240 [AP]
  1b 00 00 00 12 34 06 00    00 02 00 75 27 00 00 6d    .....4.....u'..m
  79 73 71 6c 2d 62 69 6e    2e 30 30 30 30 33 34       ysql-bin.000034
```

In the example, we see that these two `COM_QUERY` commands are sent just after `COM_REGISTER_SLAVE` and before `COM_BINLOG_DUMP`:

- `SHOW VARIABLES LIKE` 'rpl_semi_sync_master_enabled';
- `SET @rpl_semi_sync_slave` = 1.

Complete Example with GTID Registration

The example shows output up to `COM_BINLOG_DUMP` request, No Semi-Sync:

```
T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  5d 00 00 00 0a 35 2e 35    2e 35 2d 31 30 2e 32 2e    ]....5.5.5-10.2.
  31 30 2d 4d 61 72 69 61    44 42 2d 6c 6f 67 00 22    10-MariaDB-log."
  00 00 00 7d 2e 6a 4f 2c    2c 36 6a 00 fe f7 08 02    ...}.jO,,6j.....
  00 bf 81 15 00 00 00 00    00 00 07 00 00 00 38 74    ..............8t
  60 64 54 59 44 28 38 24    48 7c 00 6d 79 73 71 6c    `dTYD(8$H|.mysql
  5f 6e 61 74 69 76 65 5f    70 61 73 73 77 6f 72 64    _native_password
  00                                                    .

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  a9 00 00 01 05 a2 38 80    70 03 00 40 08 00 00 00    ......8.p..@....
  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 6d 73 61 6e    64 62 6f 78 00 14 52 42    ....msandbox..RB
  0b e8 ae 56 ec ff ef 1f    1f 14 51 1d 4a 47 f4 32    ...V......Q.JG.2
  56 74 6d 79 73 71 6c 5f    6e 61 74 69 76 65 5f 70    Vtmysql_native_p
  61 73 73 77 6f 72 64 00    54 03 5f 6f 73 05 4c 69    assword.T._os.Li
  6e 75 78 0c 5f 63 6c 69    65 6e 74 5f 6e 61 6d 65    nux._client_name
  08 6c 69 62 6d 79 73 71    6c 04 5f 70 69 64 05 33    .libmysql._pid.3
  30 30 31 33 0f 5f 63 6c    69 65 6e 74 5f 76 65 72    0013._client_ver
  73 69 6f 6e 07 31 30 2e    32 2e 31 30 09 5f 70 6c    sion.10.2.10._pl
  61 74 66 6f 72 6d 06 78    38 36 5f 36 34             atform.x86_64

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  07 00 00 02 00 00 00 02    00 00 00                   ...........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  18 00 00 00 03 53 45 4c    45 43 54 20 55 4e 49 58    .....SELECT UNIX
  5f 54 49 4d 45 53 54 41    4d 50 28 29                _TIMESTAMP()

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  01 00 00 01 01 26 00 00    02 03 64 65 66 00 00 00    .....&....def...
  10 55 4e 49 58 5f 54 49    4d 45 53 54 41 4d 50 28    .UNIX_TIMESTAMP(
  29 00 0c 3f 00 11 00 00    00 08 80 00 00 00 00 05    )..?............
  00 00 03 fe 00 00 02 00    0b 00 00 04 0a 31 35 31    .............151
  33 36 38 34 33 38 36 05    00 00 05 fe 00 00 02 00    3684386.........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  20 00 00 00 03 53 48 4f    57 20 56 41 52 49 41 42     ....SHOW VARIAB
  4c 45 53 20 4c 49 4b 45    20 27 53 45 52 56 45 52    LES LIKE 'SERVER
  5f 49 44 27                                           _ID'

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  01 00 00 01 02 64 00 00    02 03 64 65 66 12 69 6e    .....d....def.in
  66 6f 72 6d 61 74 69 6f    6e 5f 73 63 68 65 6d 61    formation_schema
  11 53 45 53 53 49 4f 4e    5f 56 41 52 49 41 42 4c    .SESSION_VARIABL
  45 53 11 53 45 53 53 49    4f 4e 5f 56 41 52 49 41    ES.SESSION_VARIA
  42 4c 45 53 0d 56 61 72    69 61 62 6c 65 5f 6e 61    BLES.Variable_na
  6d 65 0d 56 41 52 49 41    42 4c 45 5f 4e 41 4d 45    me.VARIABLE_NAME
  0c 08 00 40 00 00 00 fd    01 00 00 00 00 5d 00 00    ...@.........]..
  03 03 64 65 66 12 69 6e    66 6f 72 6d 61 74 69 6f    ..def.informatio
  6e 5f 73 63 68 65 6d 61    11 53 45 53 53 49 4f 4e    n_schema.SESSION
  5f 56 41 52 49 41 42 4c    45 53 11 53 45 53 53 49    _VARIABLES.SESSI
  4f 4e 5f 56 41 52 49 41    42 4c 45 53 05 56 61 6c    ON_VARIABLES.Val
  75 65 0e 56 41 52 49 41    42 4c 45 5f 56 41 4c 55    ue.VARIABLE_VALU
  45 0c 08 00 00 08 00 00    fd 01 00 00 00 00 05 00    E...............
  00 04 fe 00 00 22 00 10    00 00 05 09 73 65 72 76    ....."......serv
  65 72 5f 69 64 05 31 30    32 30 31 05 00 00 06 fe    er_id.10201.....
  00 00 22 00                                           ..".

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  2a 00 00 00 03 53 45 54    20 40 6d 61 73 74 65 72    *....SET @master
  5f 68 65 61 72 74 62 65    61 74 5f 70 65 72 69 6f    _heartbeat_perio
  64 3d 20 33 30 30 30 30    30 30 31 30 32 34          d= 30000001024

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  07 00 00 01 00 00 00 02    00 00 00                   ...........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
```

```
  36 00 00 00 03 53 45 54    20 40 6d 61 73 74 65 72    6....SET @master
  5f 62 69 6e 6c 6f 67 5f    63 68 65 63 6b 73 75 6d    _binlog_checksum
  3d 20 40 40 67 6c 6f 62    61 6c 2e 62 69 6e 6c 6f    = @@global.binlo
  67 5f 63 68 65 63 6b 73    75 6d                      g_checksum

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  07 00 00 01 00 00 00 02    00 00 00                   ...........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  1f 00 00 00 03 53 45 4c    45 43 54 20 40 6d 61 73    .....SELECT @mas
  74 65 72 5f 62 69 6e 6c    6f 67 5f 63 68 65 63 6b    ter_binlog_check
  73 75 6d                                              sum

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  01 00 00 01 01 2d 00 00    02 03 64 65 66 00 00 00    .....-....def...
  17 40 6d 61 73 74 65 72    5f 62 69 6e 6c 6f 67 5f    .@master_binlog_
  63 68 65 63 6b 73 75 6d    00 0c 08 00 ff ff ff 00    checksum........
  fa 00 00 27 00 00 05 00    00 03 fe 00 00 02 00 06    ...'............
  00 00 04 05 43 52 43 33    32 05 00 00 05 fe 00 00    ....CRC32.......
  02 00                                                 ..

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  20 00 00 00 03 53 45 54    20 40 6d 61 72 69 61 64     ....SET @mariad
  62 5f 73 6c 61 76 65 5f    63 61 70 61 62 69 6c 69    b_slave_capabili
  74 79 3d 34                                           ty=4

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  07 00 00 01 00 00 00 02    00 00 00                   ...........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  1f 00 00 00 03 53 45 4c    45 43 54 20 40 40 47 4c    .....SELECT @@GL
  4f 42 41 4c 2e 67 74 69    64 5f 64 6f 6d 61 69 6e    OBAL.gtid_domain
  5f 69 64                                              _id

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  01 00 00 01 01 2d 00 00    02 03 64 65 66 00 00 00    .....-....def...
  17 40 40 47 4c 4f 42 41    4c 2e 67 74 69 64 5f 64    .@@GLOBAL.gtid_d
  6f 6d 61 69 6e 5f 69 64    00 0c 3f 00 15 00 00 00    omain_id..?.....
  08 a0 00 00 00 00 05 00    00 03 fe 00 00 02 00 02    ................
  00 00 04 01 30 05 00 00    05 fe 00 00 02 00          ....0.........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  28 00 00 00 03 53 45 54    20 40 73 6c 61 76 65 5f    (....SET @slave_
  63 6f 6e 6e 65 63 74 5f    73 74 61 74 65 3d 27 30    connect_state='0
  2d 31 30 32 30 31 2d 39    38 36 38 27                -10201-9868'

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  07 00 00 01 00 00 00 02    00 00 00                   ...........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  1e 00 00 00 03 53 45 54    20 40 73 6c 61 76 65 5f    .....SET @slave_
  67 74 69 64 5f 73 74 72    69 63 74 5f 6d 6f 64 65    gtid_strict_mode
  3d 30                                                 =0

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  07 00 00 01 00 00 00 02    00 00 00                   ...........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  24 00 00 00 03 53 45 54    20 40 73 6c 61 76 65 5f    $....SET @slave_
  67 74 69 64 5f 69 67 6e    6f 72 65 5f 64 75 70 6c    gtid_ignore_dupl
  69 63 61 74 65 73 3d 30                               icates=0

T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  07 00 00 01 00 00 00 02    00 00 00                   ...........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  1b 00 00 00 15 75 27 00    00 09 73 6c 61 76 65 5f    .....u'...slave_
  6e 5f 31 00 00 c9 5a 00    00 00 00 00 00 00 00       n_1...Z........
```

```
T 127.0.0.1:23240 -> 127.0.0.1:42367 [AP]
  07 00 00 01 00 00 00 02    00 00 00                    ...........

T 127.0.0.1:42367 -> 127.0.0.1:23240 [AP]
  1b 00 00 00 12 34 06 00    00 02 00 75 27 00 00 6d    .....4.....u'..m
  79 73 71 6c 2d 62 69 6e    2e 30 30 30 30 33 34       ysql-bin.000034
```

Events Transmission After COM_BINLOG_DUMP

The MariaDB master always sends this, after the `COM_BINLOG_DUMP`:

- FAKE_ROTATE_EVENT;
- FORMAT_DESCRIPTION_EVENT: Next Pos in the header is set to 0 if not requesting binlog file from the beginning, and the GTID is not in use; otherwise, Next Pos is related to next event after FDE.
- FAKE_GTID_LIST_EVENT with latest GTID information.

After those first events, the master sends events related to changes in database to the connected replica binlog. The replica is just waiting for new events from master.

Complete Example of Event Transmission With CRC32

```
T 127.0.0.1:23240 -> 127.0.0.1:42219 [AP]
  30 00 00 01 00 00 00 00    00 04 d9 27 00 00 2f 00    0.........'../.
  00 00 00 00 00 00 20 00    04 00 00 00 00 00 00 00    ...... .........
  6d 79 73 71 6c 2d 62 69    6e 2e 30 30 30 30 33 34    mysql-bin.000034
  d5 3f ea d7 fd 00 00 02    00 fb cc 37 5a 0f d9 27    .?.........7Z..'
  00 00 fc 00 00 00 00 01    00 00 00 00 04 00 31 30    ..............10
  2e 32 2e 31 30 2d 4d 61    72 69 61 44 42 2d 6c 6f    .2.10-MariaDB-lo
  67 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    g...............
  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 13 38 0d 00    08 00 12 00 04 04 04 04    .....8..........
  12 00 00 e4 00 04 1a 08    00 00 00 08 08 08 02 00    ................
  00 00 0a 0a 0a 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
  00 00 00 00 00 04 13 04    00 0d 08 08 08 0a 0a 0a    ................
  01 17 0b 12 63 3c 00 00    03 00 fb cc 37 5a a3 d9    ....c<......7Z..
  27 00 00 3b 00 00 00 3b    01 00 00 00 00 02 00 00    '..;...;........
  00 00 00 00 00 01 00 00    00 1e 00 00 00 00 00 00    ................
  00 00 00 00 00 d9 27 00    00 86 26 00 00 00 00 00    ......'...&.....
  00 b6 33 8a 22 2c 00 00    04 00 fb cc 37 5a a1 d9    ..3.",......7Z..
  27 00 00 2b 00 00 00 66    01 00 00 00 00 10 00 00    '..+...f........
  00 6d 79 73 71 6c 2d 62    69 6e 2e 30 30 30 30 33    .mysql-bin.00003
  34 16 1f fe 3f 2c 00 00    05 00 00 00 00 00 a3 d9    4...?,..........
  27 00 00 2b 00 00 00 34    06 00 00 20 00 01 00 00    '..+...4... ....
  00 00 00 00 00 d9 27 00    00 8c 26 00 00 00 00 00    ......'...&.....
  00 4a 01 94 22 2b 00 00    06 00 94 fd 38 5a a2 d9    .J.."+......8Z..
  27 00 00 2a 00 00 00 5e    06 00 00 08 00 8d 26 00    '..*...^......&.
  00 00 00 00 00 00 00 00    00 29 00 00 00 00 00 00    .........)......
  22 87 c0 61 4c 00 00 07    00 94 fd 38 5a 02 d9 27    "..aL......8Z..'
  00 00 4b 00 00 00 a9 06    00 00 00 00 21 00 00 00    ..K.........!...
  00 00 00 00 00 00 00 1a    00 00 00 00 00 00 01 00    ................
  00 20 54 00 00 00 00 06    03 73 74 64 04 08 00 08    . T......std....
  00 08 00 00 66 6c 75 73    68 20 74 61 62 6c 65 73    ....flush tables
  6e c8 89 60                                           n..`
```

We can see:

1. `FAKE_ROTATE_EVENT` packet: 30 00 00 01 … d5 3f ea d7

2. `FORMAT_DESCRIPTION_EVENT` packet: fd 00 00 02 00 … 17 0b 12 63
   FDE size is fc 00 00 00 (252)
   Next Pos in FDE is 00 01 00 00 = >256 = 4 + FDE size (252)
3. `FAKE_GTID_LIST_EVENT` packet : 3c 00 00 03 00 … b6 33 8a 22
4. `BINLOG_CHECKPOINT` EVENT packet: 2c 00 00 04 … 16 1f fe 3f
5. `GTID_LIST_EVENT` packet: 2c 00 00 05 … 4a 01 94 22
6. `GTID_EVENT` packet: 2b 00 00 06 … 22 87 c0 61
7. `QUERY_EVENT` packet: 4c 00 00 07 … 6e c8 89 60

This page is licensed: CC BY-SA / Gnu FDL

ANNOTATE_ROWS_EVENT

`ANNOTATE_ROWS_EVENT` events accompany row events and describe the query which caused the row event.

You can enable this with –binlog-annotate-row-events (default on).

In the binary log, each `ANNOTATE_ROWS` event precedes the corresponding Table map event.

For additional information refer to the annotate_rows_log_event documentation.

> **Note:**
>
> The master server sends `ANNOTATE_ROWS_EVENT` events only if the replica server connects with the `BINLOG_SEND_ANNOTATE_ROWS_EVENT` flag (value is 2) in the `COM_BINLOG_DUMP` replica registration phase.

Header

- Event Type is 160 (`0xa0`).

Fields

- string<EOF> The SQL statement (not null-terminated).

Complete Example with CRC32

```
ee b7 15 5a a0 01 00 00  00 36 00 00 00 80 0b 00  ...Z.....6......
00 00 00 69 6e 73 65 72  74 20 69 6e 74 6f 20 74  ...insert into t
65 73 74 2e 74 34 20 76  61 6c 75 65 73 28 31 30  est.t4 values(10
30 29 6d 4c 42 33                                 0)mLB3
```

This page is licensed: CC BY-SA / Gnu FDL

BEGIN_LOAD_QUERY_EVENT

This event is written to the binary log file for LOAD DATA INFILE events if the server variable `binlog_mode` is set to "`STATEMENT`".

Header

- Event Type = 0x11.

Fields

Fixed data part:

- uint<4> The ID of the file.

Variable data part:

- byte<NULL> Null terminated data block.

Example

This page is licensed: CC BY-SA / Gnu FDL

BINLOG_CHECKPOINT_EVENT

Binlog Checkpoint Event, Event Type is 161 (0xa1). This event specifies a binlog file such that XA crash recovery can start from that file.

> **Note:**
>
> There can be more than one event in a binlog file.

Header

- Event type is 161 (0xa1).

Fields

- uint<4> Log filename length.
- string<EOF> Log filename.

Example Without CRC32

```
12 ad 26 5a a1 84 27 00  00 27 00 00 00 47 01 00  ..&Z..'..'...G..
00 00 00 10 00 00 00 6d  79 73 71 6c 2d 62 69 6e  .......mysql-bin
2e 30 30 30 30 36 32                              .000062
```

Header, 19 Bytes

- Event Timestamp = 12 ad 26 5a.
- Event Type = a1 => 161.
- Server_id = 84 27 00 00 => 00 00 27 84 = 10116.
- Event Size = 27 00 00 00 => 00 00 00 27 = 39.
- Next Pos = 47 01 00 00 => 00 00 01 47 = 327.
- Flags = 00 => 0.

Content, Variable Size

- Filename length = 10 00 00 00 = >00 00 00 10 => 16.
- Filename = mysql-bin.000062.

This page is licensed: CC BY-SA / Gnu FDL

COM_BINLOG_DUMP

This is a command the replica sends to the master after COM_REGISTER_SLAVE.

The master server sends the binlog events from the requested file and position, or if GTID registration is in use, from the GTID value set in the earlier registration phase.

The payload is:

- uint<1> command (COM_BINLOG_DUMP = 0x12).
- uint<4> The requested binlog position.
- uint<2> Flags.

- uint<4> Slave server_id.
- string<EOF> The requested binlog file name.

Note

- Flags, usually set to `0`. It can be set to `BINLOG_SEND_ANNOTATE_ROWS_EVENT` (`0x02`) if the replica server wants to receive the MariaDB 10 ANNOTATE_ROWS events.
  It can also be set to `BINLOG_DUMP_NON_BLOCK` (`1`), in the case the replica is receiving an EOF packet after the last event sent by the master.
- Requested binlog position can be `4` when registering to master server for the very first time or when requesting events from a particular binlog file from the beginning of it.
- The requested binlog file can empty when registering for the very first time if master log file is unknown or with GTID registration (not required).
- After sending events to the replica the server kills the connection.

When replication resumes or it is restarted (`STOP REPLICA`; `START REPLICA`), the replica server always sends the latest binlog file name and position, even if GTID registration is in place.

Example of COM_BINLOG_DUMP

```
1b 00 00 00 12 34 06 00    00 02 00 75 27 00 00 6d    .....4.....u'..m
  79 73 71 6c 2d 62 69 6e    2e 30 30 30 30 33 34       ysql-bin.000034
```

After 4 bytes network protocol header we can see:

- Command [1] = 12.
- Requested binlog position [4] = 34 06 00 00 => 00 00 06 34 = 1588.
- Flags [2] = 02 00 => 2 = `BINLOG_SEND_ANNOTATE_ROWS_EVENT`.
- Binlog file[n] = mysql-bin.000034.

This page is licensed: CC BY-SA / Gnu FDL

COM_REGISTER_SLAVE

This command is sent by the replica server to start MariaDB replication, and should be sent before requesting binlog events with COM_BINLOG_DUMP.

The payload is:

- uint<1> command (`COM_REGISTER_SLAVE` = `0x15`).
- uint<4> Replica server ID.
- uint<1> Replica hostname length.
- string<n> Hostname.
- uint<1> Replica username length.
- string<n> Username.
- uint<1> Replica password length.
- string<n> Replica password.
- uint<2> Replica connection port.
- uint<4> Replication rank.
- uint<4> Master server ID.

Note:

- Replica hostname, replica user, replica password, and replica port are usually not set. Some replica replication parameters can be used for such settings (`report_host` , `report_port` , and so forth).
- Replication rank is not set.
- Master server ID is not set either.

Example of COM_REGISTER_SLA

The replica server is configured with:

- server-id=10101;
- report-host=slave_n_1;
- report-port=23241.

```
1b 00 00 00 15 75 27 00    00 09 73 6c 61 76 65 5f    .....u'...slave_
 6e 5f 31 00 00 c9 5a 00    00 00 00 00 00 00 00       n_1...Z.......
```

We can see from the example:

- server_id [4] = 75 27 00 00 => 10101.
- hostname_len [1] = 09.
- hostname[n] = slave_n_1 (9 bytes).
- username len [1] = 0 (not set).
- password len [1] = 0 (not set).
- slave port [2] = c9 5a => 23241.
- rank [4] = 0.
- master server id = 0.

This page is licensed: CC BY-SA / Gnu FDL

EXECUTE_LOAD_QUERY_EVENT

This event is written to the binary log file for LOAD DATA INFILE events. The event format is similar to a QUERY_EVENT, except that it has extra static fields.

Header

- Event Type = 0x12.

Fields

Fixed Data Part

- uint<4> The ID of the thread that issued this statement on the master.
- uint<4> The time in seconds that the statement took to execute.
- uint<1> The length of the name of the database which was the default database when the statement was executed. This name appears later, in the variable data part. It is necessary for statements such as `INSERT INTO t VALUES(1)` that don't specify the database and rely on the default database previously selected by `USE` .
- uint<2> The error code resulting from execution of the statement on the master.
- uint<2> The length of the status variable block.
- uint<4> The ID of the loaded file.
- uint<4> Offset from the start of the statement to the beginning of the filename.
- uint<4> Offset from the start of the statement to the end of the filename.
- uint<1> How `LOAD DATA INFILE` handles duplicates (0x0: error, 0x1: ignore, 0x2: replace).

Variable Data Part

- byte<n> Zero or more status variables. Each status variable consists of one byte code identifying the variable stored, followed by the value of the variable. The format of the value is variable-specific.
  The number of bytes 'n' is the length of the status variable block (read in fixed data part)
- string<NUL> The default database name (null-terminated).
- string<EOF> The SQL statement. By subtraction the size of the statement can be known.

This page is licensed: CC BY-SA / Gnu FDL

Fake GTID_LIST Event

This event is sent by master server to the registering replica. It is sent only once, after the Format Description Event.

> **Note:**
>
> The fake `GTID_LIST` event is not written in the binlog file. It's created by the master and sent to newly connected replica before any "real" binlog event.

Header

- Event type is set to `163` (`0xa3`).
- Time stamp set to `0`.
- NextPos tells which is the binlog position of next event.
- Flags are set to `ARTIFICIAL` (`0x20`).

Content

The content is the same as the "real" GTID_LIST.

- **of GTIDs**
- domain_id
- server_id
- sequence
- …

This page is licensed: CC BY-SA / Gnu FDL

Fake ROTATE_EVENT

When a slave server connects to a MariaDB master server, the first binlog event sent is Fake `ROTATE_EVENT`. This event is similar to ROTATE_EVENT, but it's artificial and its purpose is to tell the replica server which the binlog file name of the master is.

This matters when the replica connects with the GTID option (no filename is given) or when using file and pos with empty file name (usually file='' and pos = 4).

The Event Type is `ROTATE_EVENT` (`0x4`).

> **Note:**
>
> The fake `ROTATE_EVENT` event is not written in the binlog file. It's created by the master and sent to newly connected replica before FORMAT_DESCRIPTION_EVENT

Header

- Timestamp set to `0`.

- Event Type is `ROTATE_EVENT`.
- Next Pos is set to `0`.
- Flags are set to `LOG_ARTIFICIAL_F` (`0x20`).

Content

The content is the same as ROTATE_EVENT.

- pos = the requested pos from slave, usually `4`.
- filename = the master binlog filename.

If it is the first fake rotate event, and the global server variable `@@binlog_checksum` is set to `CRC32`:

- crc32_checksum (4 Bytes).

This page is licensed: CC BY-SA / Gnu FDL

FORMAT_DESCRIPTION_EVENT

This is a descriptor event that is written to the beginning of a binary log file, at position 4 (after the 4 magic number bytes). The whole event written to disk is byte<19> event header + data fields.

Header

- The Event Type is 15 (0x0f).

Fields

- uint<2> The binary log format version. This is `4` .
- string<50> The MariaDB server version (example: 10.8.1-debug-log), padded with 0x00 bytes on the right.
- uint<4> Timestamp in seconds when this event was created (this is the moment when the binary log was created). This value is redundant; the same value occurs in the timestamp header field.
- uint<1> The header length. This length `-19` gives the size of the extra headers field at the end of the header for other events.
- byte<n> Variable-sized. An array that indicates the post-header lengths for all event types. There is one byte per event type that the server knows about. The value 'n' comes from the following formula:

```
n = event_size - header length - offset (2 + 50 + 4 + 1) - checksum_algo - checksum
```

- uint<1> Checksum Algorithm Type
- uint<4> CRC32 4 bytes (value matters only if checksum algo is CRC32)

Example FDE of MariaDB With CRC32

```
4d af 15 5a 0f 01 00 00  00 fc 00 00 00 00 01 00  M..Z............
00 00 00 04 00 31 30 2e  32 2e 31 30 2d 4d 61 72  .....10.2.10-Mar
69 61 44 42 2d 6c 6f 67  00 00 00 00 00 00 00 00  iaDB-log........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 4d af 15 5a 13  ...........M..Z.
00 12 00 04 04 04 04 12  00 00 e4 00 04 1a 08 00  ................
00 00 08 08 08 02 00 00  00 0a 0a 0a 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 04 13 04 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 04 13 04 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 04 13 04 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 04 13 04 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 04 13 04 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 04 13 04 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 04 13 04 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 04 13 04 00  ................
0d 08 08 08 0a 0a 0a 01  d6 ce 13 e2              ............
```

Header, 19 Bytes

- timestamp => 4d af 15 5a.
- type = 0f => 15.
- server_id = 1.
- Event Size = fc => 252.
- Next Pos = 00 01 00 00 => 00 00 01 00 => 256.
- Flags = 00 => 0.

Content, Variable Size Depending on MariaDB Versions

- format version = 04 00 => 4.
- server’s version = 10.2.10-MariaDB-log …. [50 bytes].
- create time = 4d af 15 5a.
- header_length = 13 => 19.
- event_types array[252 - 19 - (2 + 50 + 4 +1) - 1 - 4] = 171 supported events.
- checksum_algo = 01 => 1 (CRC32).
- CRC32 bytes = d6 ce 13 e2.

Example FDE of MariaDB With CRC32

```
12 ad 26 5a 0f 84 27 00  00 f5 00 00 00 f9 00 00  ..&Z..'.........
00 01 00 04 00 31 30 2e  31 2e 31 36 2d 4d 61 72  .....10.1.16-Mar
69 61 44 42 00 6c 6f 67  00 00 00 00 00 00 00 00  iaDB.log........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 12  ad 26 5a 13 38 0d 00 08  .........&Z.8...
00 12 00 04 04 04 04 12  00 00 dd 00 04 1a 08 00  ................
00 00 08 08 08 02 00 00  00 0a 0a 0a 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 04 13 04 00  ................
00 2b 91 c2 91
```

Header, 19 Bytes

- timestamp => 12 ad 26 5aa.
- type = 0f => 15.
- server_id = 84 27 00 00 => 00 00 27 84 => 10116.
- Event Size = fc => 245.

- Next Pos = f9 00 00 00 => 00 00 00 f9 => 249.
- Flags = 00 => 0.

Content, Variable Size Depending on MariaDB Versions

- format version = 04 00 => 4.
- server's version = 10.1.16-MariaDB.log …. [50 bytes].
- create time = 12 ad 26 5a.
- header_length = 13 => 19.
- event_types array[245 - 19 - (2 + 50 + 4 +1) - 1 - 4] = 164 supported events.
- checksum_algo = 0 => 0 (NONE).
- CRC32 bytes = 2b 91 c2 91 (useless).

This page is licensed: CC BY-SA / Gnu FDL

GTID_EVENT

For global transaction ID, used to start a new transaction event group, instead of the old `BEGIN` query event, and also to mark stand-alone (DDL).

`GTID_EVENT` event type is `162` (`0xa2`).

Event Header

- Type[1] = `0xa2`.
- Flags[2] = 08 00 => `LOG_EVENT_SUPPRESS_USE_F`.

Fields

- uint<8> GTID sequence.
- uint<4> Replication Domain ID.
- uint<1> Flags.

If flag & `FL_GROUP_COMMIT_ID`:

- uint<8> commit_id.

Else if flag & (`FL_PREPARED_XA` or `FL_COMPLETED_XA`):

- uint<4> format_id.
- uint<1> gtid_length.
- uint<1> bqual_length.
- byte<n> xid, where n is sum of gtrid and bqual lengths.

Else:

- uint<6> 0.

Flags

Flag	Value	Details
FL_STANDALONE	1	Set when there is no terminating `COMMIT` event.
FL_GROUP_COMMIT_ID	2	Set when event group is part of a group commit on the master. Groups with same commit_id are part of the same group commit.

Flag	Value	Details
FL_TRANSACTIONAL	4	Set for an event group that can be safely rolled back (no MyISAM, for instance).
FL_ALLOW_PARALLEL	8	Reflects the (negation of the) value of `@@SESSION.skip_parallel_replication` at the time of commit.
FL_WAITED	16	Set if a row lock wait (or other wait) is detected during the execution of the transaction.
FL_DDL	32	Set for event group containing DDL.
FL_PREPARED_XA	64	Set for XA transaction.
FL_COMPLETED_XA	128	XA transaction completed (committed or rolled back).

Transaction Example From mysqlbinlog Utility

```
BEGIN
#171205 18:22:52 server id 10124  end_log_pos 652 CRC32 0x23c8d337  GTID 0-10124-9884 trans
TBALE_MAP
#171205 18:22:52 server id 10124  end_log_pos 752 CRC32 0x52601513  Table_map: `test`.`t4` mapped to
number 92
WRITE
#171205 18:22:52 server id 10124  end_log_pos 790 CRC32 0x8869c123  Write_rows: table id 92 flags:
STMT_END_F
COMMIT
#171205 18:22:52 server id 10124  end_log_pos 821 CRC32 0x15517636  Xid = 42004
```

Standalone Event DDL (FLUSH TABLES) From mysqlbinlog Utility

```
#171205 17:44:27 server id 10124  end_log_pos 535 CRC32 0x309a668e  GTID 0-10124-9883 ddl
#171205 17:44:27 server id 10124  end_log_pos 610 CRC32 0xda151470  Query   thread_id=819   ...
```

Example GTID_EVENT with DDL and CRC32

```
eb cc 26 5a a2 8c 27 00  00 2a 00 00 00 17 02 00  ...&Z..'..*.....
00 08 00 9b 26 00 00 00  00 00 00 00 00 00 00 29  ...&.........).
00 00 00 00 00 00 8e 66  9a 30                    ......f.0.
```

Content

- GTID seq[8] = 9b 26 00 00 00 00 00 00 => 9883.
- domain id[4] = 00 00 00 00 00 => 0.
- flags[1] = 29 => 41 (`FL_DDL` =32 + `FL_ALLOW_PARALLEL` =8 + `FL_STANDALONE` =1).
- commit_id[6] = 00 00 00 00 00 00 = 0.
- CRC32[4] = 8e 66 9a 30.

Example GTID_EVENT With a Transaction and CRC32

```
ec d5 26 5a a2 8c 27 00  00 2a 00 00 00 8c 02 00 ..&Z..'..*......
00 08 00 9c 26 00 00 00  00 00 00 00 00 00 00 0c ....&...........
00 00 00 00 00 00 37 d3  c8 23                   ......7..#
```

Content

- GTID seq[8] = 9c 26 00 00 00 00 00 00 => 9884.
- domain id[4] = 00 00 00 00 00 => 0.
- flags[1] = 0c => 12 (`FL_ALLOW_PARALLEL` =8 + `FL_TRANSACTIONAL` =4).

- commit_id[6] = 00 00 00 00 00 00 = 0.
- CRC32[4] = 37 d3 c8 23.

This page is licensed: CC BY-SA / Gnu FDL

GTID_LIST_EVENT

Logged in every binlog to record the current replication state. Consists of the last GTID seen for each replication domain.

The Global Transaction ID, GTID for short, consists of three components:

- Replication domain ID;
- Master server ID;
- Sequence ID.

It's represented as three numbers separated with dashes (`-`); for example: `1-1222-1011` .

It's usually written after the Format Description Event. If binary log encryption is enabled, it is written after the Start Encryption Event.

> **Note:**
>
> In case of encrypted binlog files (encrypt_binlog is set to `ON`), this event is written just after the START_ENCRYPTION_EVENT.

Header

- Event type is 163 (0xa3).

Fields

- uint<4> Number of GTIDs.
- For (i=0; i < gtid_count; i++):
  - uint<4> Replication Domain ID.
  - uint<4> Server_ID.
  - uint<8> GTID sequence.

The minimum content size for one GTID is `4 + (4 + 4 + 8) * 1 = 20 bytes` .

Example With 1 GTID, With CRC32

From the mysqlbinlog utility:

170824 9:52:04 server id 10124 end_log_pos 292 CRC32 **0xb6d8f0a8** Gtid list **[0-10124-3584]**

```
a4 85 9e 59 a3 8c 27 00  00 2b 00 00 00 24 01 00  ...Y..'..+...$..
00 00 00 01 00 00 00 00  00 00 00 8c 27 00 00 00  ............'...
0e 00 00 00 00 00 00 a8  f0 d8 b6                 ..........
```

Header, 19 Bytes

- Event Time = a4 85 9e 59 ===> 2017-08-24 9:52:04.
- Event Type = a3 => 163.
- Server_id = 8c 27 00 00 => 00 00 27 8c => 10124.
- Event Size = 2b => 43 (header[19] + 1 GTID(20 bytes) + CRC32[4].
- Next Pos = 24 01 00 00 => 00 00 01 24 => 292.

- Flags = 00 => 0.

Content, Variable Size, is (4 + (4 + 4 + 8) * n_GTIDs) Bytes

The content example with one GTID is 20 bytes + 4 bytes CRC32:

- Number of GTIDs[4] = 01 00 00 00 => 1.
- GTID[0] replication_domain[4] = 00 00 00 00 => 0.
- GTID[0] Server_id[4] = 8c 27 00 00 => 00 00 27 8c => 10124.
- GTID[0] Sequence[8] = 00 0e 00 00 00 00 00 00 ===> 3584.
- crc32[4] = a8 f0 d8 b6 => b6 d8 f0 a8 => 0xb6d8f0a8.

This page is licensed: CC BY-SA / Gnu FDL

HEARTBEAT_LOG_EVENT

This event does not appear in the binary log. It's only sent over the network by a master to a replica server to let it know that the master is still alive, and is only sent when the master has no binlog events to send to replica servers.

> **Note:**
>
> This event is never written to the binary log file.

Header

- Timestamp is set to `0`.
- Next position is set to last position.
- Type is set to `HEARTBEAT_EVENT` (`0x1b`).

Fields

- string<EOF> The current master binary log name.

Example of Transmission (Without CRC32)

```
T 127.0.0.1:8808 -> 127.0.0.1:57157 [AP]
  23 00 00 04 00 00 00 00    00 1b 67 2b 00 00 22 00    '.........g+..&.
  00 00 ed 01 00 00 20 00    66 6f 6f 2d 62 69 6e 2e    ......  .log-bin.
  31 30 30 30 31 33 39                                  1000139
```

Network Replication protocol, 5 bytes

- packet size [3] = 23 00 00 => 00 00 23 => 35 (ok byte + event size).
- pkt sequence [1] = 04.
- `OK` indicator [1] = 0 (`OK`).

Heartbeat Event

Header

- Timestamp [4] = 00 00 00 00 => 0.
- Event Type [1] = 1b => 27.
- Server_id [4] = 67 2b 00 00 => 00 00 2b 67 => 111111.
- Event Size [4] = 22 00 00 00 => 00 00 00 26 => 34 (header + data).
- Next_pos [4] = ed 01 00 00 => 00 00 01 ed => 493.
- Flags [2] == 20 00 => 00 20 = > 32.

Content, String<EOF>

- log-bin.1000139.

This page is licensed: CC BY-SA / Gnu FDL

INTVAR_EVENT

An `INTVAR_EVENT` is written every time a statement uses an auto increment column or the `LAST_INSERT_ID()` function.

Header

- Event Type is `5` (`0x05`).

Fields

- uint<1> Type.
- uint<8> Value.

Type

0x00	Invalid value.
0x01	LAST_INSERT_ID.
0x02	Insert id (auto_increment).

Example From mysqlbinlog Utility, CRC32

```
# at 738
#180610 11:20:56 server id 1  end_log_pos 770 CRC32 0xf5a23f2d  Intvar
SET LAST_INSERT_ID=1/*!*/;
```

Example Event As It's Written In The Binlog File

```
78 ed 1c 5b 05 01 00 00 00 20 00       x..[..... .
00 00 02 03 00 00 00 00 01 01 00 00 00 00 00 00  ................
00 2d 3f a2 f5                                   .-?..
```

This page is licensed: CC BY-SA / Gnu FDL

QUERY_EVENT

This event is written into the binary log file for:

- Statement-based replication (updating statements).
- DDLs.
- COMMIT related to nontransactional engines (like MyISAM or BLACKHOLE).

Header

- Event Type = 0x02.

Fields

Fixed data part:

- uint<4> The ID of the thread that issued this statement on the master.
- uint<4> The time in seconds that the statement took to execute.

- uint<1> The length of the name of the database which was the default database when the statement was executed. This name appears later, in the variable data part. It is necessary for statements such as `INSERT INTO t VALUES(1)` that don't specify the database and rely on the default database previously selected by `USE`.
- uint<2> The error code resulting from execution of the statement on the master.
- uint<2> The length of the status variable block.

Variable data part:

- byte<n> Zero or more status variables. Each status variable consists of one byte code identifying the variable stored, followed by the value of the variable. The format of the value is variable-specific.
  The number of bytes 'n' is the length of the status variable block (read in fixed data part).
- string<NUL> The default database name (null-terminated).
- string<EOF> The SQL statement. By subtraction the size of the statement can be known.

Status Variables

Q_FLAGS2_CODE (0x00)

- uint<4> bitmask.

Value	Option
0x00004000	OPTION_AUTO_IS_NULL
0x00080000	OPTION_NOT_AUTOCOMMIT
0x04000000	OPTION_NO_FOREIGN_KEY_CHECKS
0x08000000	OPTION_RELAXED_UNIQUE_CHECKS

Q_SQL_MODE_CODE (0x01)

- uint<8> 8-byte bitmask.

Value	Option
0x00000001	MODE_REAL_AS_FLOAT
0x00000002	MODE_PIPES_AS_CONCAT
0x00000004	MODE_ANSI_QUOTES
0x00000008	MODE_IGNORE_SPACE
0x00000010	MODE_NOT_USED
0x00000020	MODE_ONLY_FULL_GROUP_BY
0x00000040	MODE_NO_UNSIGNED_SUBTRACTION
0x00000080	MODE_NO_DIR_IN_CREATE
0x00000100	MODE_POSTGRESQL
0x00000200	MODE_ORACLE
0x00000400	MODE_MSSQL
0x00000800	MODE_DB2
0x00001000	MODE_MAXDB
0x00002000	MODE_NO_KEY_OPTIONS
0x00004000	MODE_NO_TABLE_OPTIONS
0x00008000	MODE_NO_FIELD_OPTIONS
0x00010000	MODE_MYSQL323

Value	Option
0x00020000	MODE_MYSQL40
0x00040000	MODE_ANSI
0x00080000	MODE_NO_AUTO_VALUE_ON_ZERO
0x00100000	MODE_NO_BACKSLASH_ESCAPES
0x00200000	MODE_STRICT_TRANS_TABLES
0x00400000	MODE_STRICT_ALL_TABLES
0x00800000	MODE_NO_ZERO_IN_DATE
0x01000000	MODE_NO_ZERO_DATE
0x02000000	MODE_INVALID_DATES
0x04000000	MODE_ERROR_FOR_DIVISION_BY_ZERO
0x08000000	MODE_TRADITIONAL
0x10000000	MODE_NO_AUTO_CREATE_USER
0x20000000	MODE_HIGH_NOT_PRECEDENCE
0x40000000	MODE_NO_ENGINE_SUBSTITUTION
0x80000000	MODE_PAD_CHAR_TO_FULL_LENGTH

Q_CATALOG_NZ_CODE (0x02)

- uint<1> length
- string<length +1> catalog name + '\0'

Q_AUTO_INCREMENT (0x03)

- uint<2> auto_increment increment
- uint<2> auto_increment offset

Q_CHARSET_CODE (0x04)

- uint<2> client character set
- uint<2> collation_connection
- uint<2> collation_server

Q_TIMEZONE_CODE (0x05)

- uint<1> length
- string<length> time zone

Q_CATALOG_NZ_CODE (0x06)

- uint<1> length
- string<length> catalog

Q_LC_TIME_NAMES_CODE (0x07)

- uint<2> code
  The mapping between code and names are defined in sql_locale.cc.

Q_CHARSET_DATABASE_CODE (0x08)

- uint<2> database collation

Q_TABLE_MAP_FOR_UPDATE_CODE (0x09)

- uint<8> table bittmask
  Every bit of this variable represents a table, and is set to 1 if the corresponding table is to be updated by this statement.

Q_MASTER_DATA_WRITTEN_CODE (0x0A)

- Not in use any more
- uint<4> original event length

Q_INVOKER (0x0B|)

- uint<1> user name length
- string<length> user name
- uint<1> host name length
- string<length> host name

Q_UPDATED_DB_NAMES (0x0C)

- MySQL only
- uint<1> count
- for (i=0;i < count; i++)
  - string Null terminated database name

Q_MICROSECONDS (0x0D)

- MySQL only
- uint<3> microsecond part

Q_HRNOW (0x80)

- MariaDB only
- uint<3> microsecond part

Q_XID (0x81)

- MariaDB only
- uint<8> xid

Example With CRC32

```
71 17 28 5a 02 8c 27 00  00 55 00 00 00 01 09 00  q.(Z..'..U......
00 00 00 66 01 00 00 00  00 00 00 00 00 00 1a 00  ...f............
00 00 00 00 00 01 00 00  00 50 00 00 00 00 06 03  .........P......
73 74 64 04 08 00 08 00  08 00 00 54 52 55 4e 43  std........TRUNC
41 54 45 20 54 41 42 4c  45 20 74 65 73 74 2e 74  ATE TABLE test.t
34 4a 69 9e ed                                    4Ji..
```

Header, 19 Bytes

- Event Time[4] = 71 17 28 5a ===> 1512576881.
- Event Type[1] = 2.
- Server_id[4] = 8c 27 00 00 ===> 10124.
- Event Size = 55 00 00 00 ===> 85.
- Next Pos = 01 09 00 00 ===> 2305.

- Flags = 00 00 => 0.

Content, Variable Data

- Thread ID[4] = 66 01 00 00 ===> 358.
- Execution Time[4] = 00 00 00 00 => 0 seconds.
- Statement default database name len[1] = 00 => 0 (no default database).
- Error code[2] = 00 00 => 0 (no errors).
- Status variable block len[2] = 1a 00 => 26.
- Status variables[n] = 00 … 08 00.
- The default database[string] = 00 = 0 (no default database).
- The SQL statement[string] = `TRUNCATE TABLE` test.t4.

CRC32, 4 Bytes

- 4a 69 9e ed.

Example With Default Database and CRC32

```
MariaDB []> use test;
Database changed
MariaDB [test]> TRUNCATE TABLE t4;
...

ce 22 28 5a 02 8c 27 00  00 54 00 00 00 87 0c 00  ."(Z..'..T......
00 00 00 66 01 00 00 01  00 00 00 04 00 00 1a 00  ...f............
00 00 00 00 00 01 00 00  00 50 00 00 00 00 06 03  .........P......
73 74 64 04 08 00 08 00  08 00 74 65 73 74 00 54  std.......test.T
52 55 4e 43 41 54 45 20  54 41 42 4c 45 20 74 34  RUNCATE TABLE t4
08 f1 09 16                                       ....
```

Content, Variable Data

- Thread ID[4] = 66 01 00 00 ===> 358.
- Execution Time[4] = 10 00 00 00 => 1 second.
- Statement default database name len[1] = 04 => 4 (default database is “test”).
- Error code[2] = 00 00 => 0 (no errors).
- Status variable block len[2] = 1a 00 => 26.
- Status variables[n] = 00 … 08 00.
- The default database[string] = 74 65 73 74 00 ⇒ test.
- The SQL statement[string] = `TRUNCATE TABLE` test.t4.

This page is licensed: CC BY-SA / Gnu FDL

RAND_EVENT

The SQL function RAND() generates a random number.

A `RAND_EVENT` contains two seed values that set the rand_seed1 and rand_seed2 system variables that are used to compute the random number.

> **Note:**
>
> It is written only before a `QUERY_EVENT` , and not used with row-based logging.

Header

- Event Type is `13` (`0x0d`).

Fields

- uint<8> The value for the first seed.
- uint<8> The value for the second seed.

Example From mysqlbinlog Utility, No CRC32

```
# at 389
#171206 13:46:56 server id 10116  end_log_pos 424   Rand
SET @@RAND_SEED1=685157301, @@RAND_SEED2=758850369/*!*/;
# at 424
```

Example Event as Written to the Binlog File

`c0 e6 27 5a 0d 84 27 00 00 23 00 00 00 a8 01 00 ..'Z..'..#..... 00 00 00 b5 ab d6 28 00 00 00 00 41 23 3b 2d 00 ......(..... 00 00 00 ....`

This page is licensed: CC BY-SA / Gnu FDL

ROTATE_EVENT

When a binary log file exceeds the configured size limit, a `ROTATE_EVENT` is written at the end of the file, pointing to the next file in the sequence.

`ROTATE_EVENT` is generated locally and written to the binary log on the master. It is also written when a FLUSH LOGS statement occurs on the master server.

The `ROTATE_EVENT` is sent to the connected replica servers.

Header

- The Event Type is `ROTATE_EVENT` (`0x4`).

Fields

- uint<8> The position of the first event in the next log file. It always contains the number `4` (meaning the next event starts at position 4 in the next binary log).
- string<EOF> The next binary log name. The filename is not null-terminated.

Example of Transmission With CRC32 (The Last 4 Bytes)

```
T 127.0.0.1:8808 -> 127.0.0.1:57157 [AP]
  30 00 00 4d 00 bc 4e 21    5a 04 d9 27 00 00 2f 00    0..M..N!Z..'../.
  00 00 c0 01 00 00 00 00    04 00 00 00 00 00 00 00    ................
  6d 79 73 71 6c 2d 62 69    6e 2e 30 30 30 30 31 39    mysql-bin.000019
  b2 bc db bf                                           ....
```

This page is licensed: CC BY-SA / Gnu FDL

ROWS_EVENT_V1/V2, ROWS_COMPRESSED_EVENT_V1

A `ROWS_EVENT_V1` is written for row-based replication if data is inserted, deleted or updated.

A `ROWS_EVENT` (version 2) is written for row based replication if data is inserted, deleted or updated. MariaDB Server doesn't send version 2 row events.

Event types

Event	Details
WRITE_ROWS_EVENT_V1	Insert new row.
UPDATE_ROWS_EVENT_V1	Update existing row.
DELETE_ROWS_EVENT_V1	Delete existing row.
WRITE_ROWS_COMPRESSED_EVENT_V1	Insert new row.
UPDATE_ROWS_COMPRESSED_EVENT_V1	Update existing row.
DELETE_ROWS_COMPRESSED_EVENT_V1	Delete existing row.
WRITE_ROWS_EVENT	Insert new row (version 2, MySQL only).
UPDATE_ROWS_EVENT	Update existing row (version 2, MySQL only).
DELETE_ROWS_EVENT	Delete existing row (version 2, MySQL only).

Header

- `WRITE_ROWS_EVENT_V1` : Event Type is 23 (0x17).
- `UPDATE_ROWS_EVENT_V1` : Event Type is 24 (0x18).
- `DELETE_ROWS_EVENT_V1` : Event Type is 25 (0x19).
- `WRITE_ROWS_EVENT` : Event Type is 30 (0x1E).
- `UPDATE_ROWS_EVENT` : Event Type is 31 (0x1F).
- `DELETE_ROWS_EVENT` : Event Type is 32 (0x20).
- `WRITE_ROWS_COMPRESSED_EVENT_V1` : Event Type is 166 (0xA6).
- `UPDATE_ROWS_COMPRESSED_EVENT_V1` : Event Type is 167 (0xA7).
- `DELETE_ROWS_COMPRESSED_EVENT_V1` : Event Type is 168 (0xA8).

Fields

- uint<6> The table id.
- uint<2> Flags.
- If `rows_event` is version 2:
  - uint<2> Extra data length.
  - string<len> Extra data.
- uint Number of columns.
- byte<n> Columns used. n = (number_of_columns + 7)/8.
- If (event_type == `UPDATE_ROWS_EVENT_v1`):
  - byte Columns used (Update). n = (number_of_columns + 7)/8.
- If `*_COMPRESSED_EVENT_V1` :
  - byte<1> header.
    - algorithm: (header & 0x07) >> 4 (always 0=zlib).
    - header_size: header & 0x07.
  - byte<header_size>uncompressed length, stored in MyISAM format.
- byte<n> Null Bitmap (n = (number_of_columns + 7)/8).
- string<len> Column data. The length needs to be calculated by checking the column types from referring `TABLE_MAP_EVENT` .
- If (event_type == `UPDATE_ROWS_EVENT_v1`):
  - byte<n> Null Bitmap_Update. n = (number_of_columns + 7)/8.

- string<len>
- g Update Column data. The length needs to be calculated by checking the used columns bitmap and column types from referring `TABLE_MAP_EVENT`.

Table ID

Table id refers to a table defined by TABLE_MAP_EVENT. The special value `0xFFFFFF` should have “end of statement flag” (`0x0001`) set and indicates that table maps can be freed.

Flags

Flag	Details
0x0001	End of statement.
0x0002	No foreign key checks.
0x0004	No unique key checks.
0x0008	Indicates that rows in this event are complete.
0x0010	No check constraints.

Extra data length (version 2)

The length of extra data.

Extra data (version 2)

Extra data, length is extra data length `-2`.

Column Data Formats

The row data is stored in a packed format where each field is encoded in a particular format. The encoding is almost identical to the binary protocol but there are a few differences.

The field metadata is stored in the metadata block of the TABLE_MAP_EVENT. The metadata is required to decode the events. The following list shows number of bytes a field uses from the metadata block.

- 2 bytes
  - MYSQL_TYPE_BIT
  - MYSQL_TYPE_ENUM
  - MYSQL_TYPE_SET
  - MYSQL_TYPE_NEWDECIMAL
  - MYSQL_TYPE_DECIMAL
  - MYSQL_TYPE_VARCHAR
  - MYSQL_TYPE_VAR_STRING
  - MYSQL_TYPE_STRING
- 1 byte
  - MYSQL_TYPE_TINY_BLOB
  - MYSQL_TYPE_MEDIUM_BLOB
  - MYSQL_TYPE_LONG_BLOB
  - MYSQL_TYPE_BLOB
  - MYSQL_TYPE_TIMESTAMP2
  - MYSQL_TYPE_DATETIME2
  - MYSQL_TYPE_TIME2

- MYSQL_TYPE_FLOAT
- MYSQL_TYPE_DOUBLE

The types that aren't listed here do not store data in the metadata block.

Simple Types

Type	Details
MYSQL_TYPE_NULL	Bit set in null bitmap, no value in row data.
MYSQL_TYPE_TINY	1 byte integer.
MYSQL_TYPE_YEAR	1 byte integer (year = value + 1900).
MYSQL_TYPE_SHORT	2 byte integer.
MYSQL_TYPE_INT24	3 byte integer.
MYSQL_TYPE_LONG	4 byte integer.
MYSQL_TYPE_LONGLONG	8 byte integer.
MYSQL_TYPE_FLOAT	4 byte floating-point value (stored as a C float type).
MYSQL_TYPE_DOUBLE	8 byte floating-point value (stored as a C double type).

`MYSQL_TYPE_BLOB` and other blob types

Stored as a length-encoded string where the string is preceded by a variable-sized integer that stores the length of the blob. The size of the preceding integer in bytes is stored as a one byte integer in the table metadata that is a part of the table map event.

For example, if the value 4 is stored in the table metadata, the length is stored as a 4 byte integer (for instance, `uint32_t`) followed by the data.

The exact `column_type` can be determined by the metadata length:

Length	Type
1	MYSQL_TYPE_TINY_BLOB
2	MYSQL_TYPE_BLOB
3	MYSQL_TYPE_MEDIUM_BLOB
4	MYSQL_TYPE_LONG_BLOB

`MYSQL_TYPE_STRING`, `MYSQL_TYPE_SET` and `MYSQL_TYPE_ENUM`

Stored as a fixed-length string with the length of the string stored in the second byte of the table metadata. All three of these types are stored as `MYSQL_TYPE_STRING` in the binlog and the real type of the field is stored in the first byte of the metadata.

`MYSQL_TYPE_VARCHAR` and other variable length string types

Stored as a length-encoded string where the string is preceded by a variable-sized integer that stores the length of the string. The field length is stored as a two byte integer in the table metadata.

If the field length is larger than 255, the string length is stored as a two byte integer. If the value is equal to or less than 255, the string length is stored as a one byte integer.

`MYSQL_TYPE_DATETIME`

This field type is only used in MariaDB if the global variable `mysql56_temporal_format` is set to `OFF`.
Stored as an 8 byte value with the values stored as multiples of 100. This means that the stored value is in the

format `YYYYMMDDHHMMSS` , and can be extracted by repeatedly calculating the remainder of dividing the value by 100 and dividing the value by 100. The following pseudo-code demonstrates extracting the value.

```
value = read_8_byte_value(row_data)
date_val= value / 1000000
time_val= value % 1000000

year = (date_val / 100) / 100
month = (date_val / 100) % 100
day = date_val % 100
hour= (time_val / 100) / 100
minute = (time_val / 100) % 100
second = time_val % 100
```

`MYSQL_TYPE_TIME`

This field type is only used in MariaDB if the global variable `mysql56_temporal_format` is set to `OFF`.
Stored as a 3 byte value with the values stored as multiples of 100. This means that the stored value is in the format `HHMMSS` and can be extracted in the same way a `MYSQL_TYPE_DATETIME` is extracted. The following pseudo-code demonstrates extracting the value.

```
time_val = read_3_byte_value(row_data); /* myisam pack format */
hour= (time_val / 100) / 100;
minute = (time_val / 100) % 100;
second = time_val % 100;
```

`MYSQL_TYPE_DATETIME2`

Stored as 4-byte value The number of decimals for the fractional part is stored in the table metadata as a one byte value. The number of bytes that follow the 5 byte datetime value can be calculated with the following formula: `(decimals + 1) / 2` .

```
val = read_5_byte_value(row_data) - 0x8000000000
d_val= val >> 17;
t_val = val % (1 << 17);

day= d_val % (1 << 5);
month= (d_val >> 5) % 13;
year= (d_val >> 5) / 13;
second= t_val % (1 << 6);
minute= (t_val >> 6) % (1 << 6);
hour= (t_val)(time_part >> 12);
```

`MYSQL_TYPE_TIME2`

Stored as 3-byte value. The number of decimals for the fractional part is stored in the table metadata as a one byte value. The number of bytes that follow the 3 byte time value can be calculated with the following formula: `(decimals + 1) / 2` .

```
t_val = read_3_byte_value(row_data) - 0x800000
if (t_val < 0)
{
  signed= 1;
  t_val= - tval;
}
hour= (t_val >> 12) % (1 << 10);
minute= (t_val >> 6) % (1 << 6);
second= t_val % (1 << 6);
```

`MYSQL_TYPE_TIMESTAMP2`

Stored as a 4 byte UNIX timestamp (number of seconds since 00:00, Jan 1 1970 UTC) followed by the fractional second parts. The number of decimals for the fractional part is stored in the table metadata as a one byte value. The number of bytes that follow the 4 byte timestamp can be calculated with the following formula: `(decimals + 1) / 2`

Microseconds for `MYSQL_TYPE_DATETIME2`, `MYSQL_TYPE_TIME2`, and `MYSQL_TYPE_TIMESTAMP`.

```
len = (decimals + 1) / 2
  val= read_len_byte_value(row_data)
  llen= 0: microseconds= 0
  len = 1 or 2: microseconds = val * 10000
  len = 3 or 4: microseconds = 2-bytes val in myisam pack format
  len = 5 or 6: microseconds = 3-bytes val in myisam pack format
```

`MYSQL_TYPE_DATE`

Stored as a 3 byte value where bits 1 to 5 store the day, bits 6 to 9 store the month and the remaining bits store the year.

`MYSQL_TYPE_TIMESTAMP`

Stored as a 4 byte UNIX timestamp (number of seconds since 00:00, Jan 1 1970 UTC).

Example From mysqlbinlog Utility, CRC32

```
# at 1680
#180611  9:50:51 server id 1  end_log_pos 1754 CRC32 0x5415a8fb     Write_rows: table id 23 flags:
STMT_END_F

BINLOG '
2ykeWxMBAAAAPgAAAJAGAAAAABcAAAAAAAEABHRlc3QACWJ1bGtfbnVsbAAFDwMFE/YGFAAIAAMB
H1bULg8=
2ykeWxcBAAAASgAAANoGAAAAABcAAAAAAAEABf/gATMDAAAAAAAAAAAACECAAACDAP/gATMDAAAA
AAAAAAACECAAACDAPuoFVQ=
'/*!*/;
```

Example Event as Written to the Binlog File

```
db 29 1e 5b 17 01 00 00 00 4a 00       .).[.....J.
00 00 da 06 00 00 00 00 17 00 00 00 00 00 01 00  ................
05 ff e0 01 33 03 00 00 00 00 00 00 00 00 08  ....3...........
40 80 00 00 83 00 ff e0 01 33 03 00 00 00 00 00  @........3......
00 00 00 00 08 40 80 00 00 83 00 fb a8 15 54     .....@........T                                  ....
```

This page is licensed: CC BY-SA / Gnu FDL

START_ENCRYPTION_EVENT

The `START_ENCRYPTION` event is written to every binary log file if encrypt_binlog is set to `ON`.

This event is written just once, after the Format Description event (which is the first event of a binlog file at pos 4).

The event has the 19 bytes event header with EventType set to value `164` (`0xa4`) + 17 bytes data.

Header

- Event type is `164` (`0xa4`).

Fields

- uint<1> The Encryption scheme, always set to 1 for system files.
- uint<4> The Encryption key version.
- byte<12> Nonce (12 random bytes) of current binlog file.

Decryption of Following Events

All data of following events in the binlog file are encrypted, except for the `event_length` field.

The 16 byte encryption IV is generated from the 12 byte nonce (uint<12>) in the binlog plus the current position of the event being encrypted (uint<4>). This means the last four bytes of the IV change for every event and the first 12 bytes change for every binlog file.

Since the `event_length` is always unencrypted, the encrypted data block has to be modified before it can be decrypted:

- Store event_length.
- Copy the first four bytes (encrypted timestamp) to event_length position (offset=9).
- Decrypt starting from offset 4 and store result at offset 4 of decrypted buffer.

The unencrypted block now also needs to be modified:

- Move unencrypted timestamp value from offset 9 to the beginning (offset=0).
- Store event_length at position 9.

Complete example with CRC32 from a binary log.

```
b8 5f 5a 59 a4 5d 00 00  00 28 00 00 00 21 01 00 ._ZY.]...(...!..
00 00 00 01 01 00 00 00  65 57 50 26 63 59 37 46 ........eWP&cY7F
2f 3b 33 23 06 bb da 62                          /;3#...b
```

Header, 19 Bytes

- timestamp [4] = b8 5f 5a 59 => 59 5a 5f b8 => 1499094968 [2017-07-03 17:16:08].
- type [1} = a4 => 164.
- server_id [4} = 5d 00 00 00 => 00 00 00 5d => 93.
- event_size [4] = 28 00 00 00 => 00 00 00 28 => 40 [header + content + crc32(header + content)].
- next_pos [4] = 21 01 00 00 => 00 00 01 21 => 289.
- flags [2] = 00 00 => 0.

Content, 17 Bytes

- Enc scheme [1] = 01 => 1.
- Enc key ver [4] = 01 00 00 00 => 00 00 00 01 => 1.
- Nonce [12] = eWP&cY7F/;3#.

CRC32, 4 bytes, of the Whole Event (Header[19] + Content[17])

- 06 bb da 62 => 62 da bb 06 => 1658501894.

This page is licensed: CC BY-SA / Gnu FDL

STOP_EVENT

The master server writes the event to the binary log when it shuts down, or when resuming after a mariadbd process crash.

A new binary log file is always created, but there is no `ROTATE_EVENT`.

`STOP_EVENT` is the last written event after a clean shutdown, or when resuming after a crash.

> **Warning:**
>
> This event is never sent to replica servers.

Header

- Event header with EventType set to `STOP_EVENT` (`0x03`).
- Event header NextPos set to `EOF`.
- No special flags added.

Fields

- The event has no data.

Example With CRC32 (Last 4 Bytes)

Event size = header[19] + 0 bytes data + 4 CRC32 = 23.

```
3a b8 15 5a 03 01 00 00  00 17 00 00 00 09 0c 00  ..Z............
00 00 00 4e 99 ee 2c                              ...N..,
```

This page is licensed: CC BY-SA / Gnu FDL

TABLE_MAP_EVENT

Used for row-based binary logging beginning (binlog_format=`ROW` or `MIXED`).

This event precedes each row operation event and maps a table definition to a number, where the table definition consists of database and table names.

Header

- Event Type is `19` (`0x13`).

Fields

Fixed Data Part

- uint<6> The table ID.
- uint<2> Reserved for future use.

Variable Data Part

- uint<1> Database name length.
- string<NUL> The database name (null-terminated).
- uint<1> Table name length.
- string<NUL> The table name (null-terminated).
- int<lenenc> The number of columns in the table.
- byte<n> An array of ‘n’ column types, one byte per column.
- int<lenenc> The length of the metadata block.
- byte<n> The metadata block;
- byte<n> Bit-field indicating whether each column can be `NULL`, one bit per column.

- If (more_data_available):
  - byte<n> Optional metadata block.

Metadata Block

The metadata block contains type specific metadata information for each column.

Type	Length	Description
MYSQL_TYPE_BLOB	1	Number of bytes for length: e.g. 4 bytes means length is stored in a 4 byte integer).
MYSQL_TYPE_DATETIME2	1	Length of microseconds.
MYSQL_TYPE_DECIMAL	2	Not in use anymore.
MYSQL_TYPE_DOUBLE ,MYSQL_TYPE_FLOAT	1	length (4 or 8 bytes).
MYSQL_TYPE_STRING	2	The first byte contains type (`MYSQL_TYPE_STRING` , `MYSQL_TYPE_ENUM` , or `MYSQL_TYPE_SET`). The second byte contains the length.
MYSQL_TYPE_NEWDECIMAL	2	Precision, Scale.
MYSQL_TYPE_TIME2	1	Length in microseconds.
MYSQL_TYPE_TIMESTAMP2	1	Length in microseconds.
MYSQL_TYPE_VARCHAR, MYSQL_TYPE_VAR_STRING	2	Defined varchar length. If the value is > 255, the length is stored in 2 bytes, otherwise in 1 byte.

Optional Metadata Block

Optional metadata are available if the global server variable `BINLOG_ROW_METADATA` is set to `MIN` or `FULL` .

The metadata block consists of one or more of the following blocks:

- byte<1> Optional metadata type.
- int<lenenc> Length.
- byte<len> Data.

Optional metadata types:

Name	Value	Mode	Description
SIGNEDNESS	1	MIN	Data contains a bitmap indicating which integer columns are signed.
DEFAULT_CHARSET	2	MIN	Character set of string columns, used if most columns have the same result. Columns with other character sets will follow as pair (column_index, collation number).
COLUMN_CHARSET	3	MIN	Character set of columns, used if columns have different character sets. Returned as a sequence of collation numbers.
COLUMN_NAME	4	FULL	List of Column names, the first byte specifies the length of the column name.
SET_STR_VALUE	5	FULL	List of set values: First byte is the number of different values, followed by length/value pairs.
ENUM_STR_VALUE	6	FULL	Same as SET_STR_VALUE. Since ENUM values might have up to 0xFFFF members, the number of values is a length encoded integer.
GEOMETRY_TYPE	7	FULL	A sequence of bytes representing the type of `GEOMETRY` columns: 0 = `GEOMETRY` , 1 = `POINT` , 2 = `LINESTRING` , 3 = `POLYGON` , 4= `MULTIPOINT` , 5 = `MULTILINESTRING` , 6 = `MULTIPOLYGON` , 7 = `GEOMETRYCOLLECTION` .

Name	Value	Mode	Description
SIMPLE_PRIMARY_KEY	8	FULL	A sequence of length encoded column indexes.
PRIMARY_KEY_WITH_PREFIX	9	FULL	A sequence of length encoded column indexes and prefix lengths.
ENUM_AND_SET_DEFAULT_CHARSET	10	FULL	The default character set number used for `ENUM` and `SET` columns.
ENUM_AND_SET_COLUMN_CHARSET	11	FULL	Character set of `ENUM` and `SET` columns, used if these columns have different character sets. Returned as a sequence of collation numbers.

Example From mysqlbinlog

```
# at 847
#171206 13:43:00 server id 10124  end_log_pos 892 CRC32 0xbe3c6b05  Table_map: `test`.`t4` mapped to
number 33
# at 892
```

Complete Event

```
d4 e5 27 5a 13 8c 27 00  00 2d 00 00 00 7c 03 00  ..'Z..'..-...|..
00 00 00 21 00 00 00 00  00 01 00 04 74 65 73 74  ...!........test
00 02 74 34 00 01 03 01  01 05 6b 3c be           ..t4......k<.
```

This page is licensed: CC BY-SA / Gnu FDL

USER_VAR_EVENT

A `USER_VAR_EVENT` is written every time a statement uses a user defined variable.

Header

- Event Type is `14` (`0x0e`).

Fields

- uint<4> The length of the user variable name.
- string<len> The name of the user variable.
- uint<1> `NULL` indicator.
- If (not null indicator):
  - uint<1> variable type.
  - uint<4> collation number.
  - uint<4> The length of value.
  - string<len> value.
  - uint<1> flags.

Variable type

Value	Type	Example
0x00	STRING_RESULT	set @a:=“foo”
0x01	REAL_RESULT	set @a:= @@timestamp
0x02	INT_RESULT	set @a:= 4

Value	Type	Example
0x03	ROW_RESULT	(not in use)
0x04	DECIMAL_RESULT	set @a:=1.2345

Flag

0x01	unsigned

Example for `SET @foo:="bar"` From mysqlbinlog Utility, CRC32

```
# at 511
#180610 10:26:43 server id 1  end_log_pos 554 CRC32 0x7dd93d6b  User_var
SET @`foo`:=_utf8 X'626172' COLLATE `utf8_general_ci`/*!*/;
```

Example Event as Written to the Binlog File

```
c3 e0 1c 5b 0e 01 00 00 00 2b 00       ...[.....+.
00 00 2a 02 00 00 00 00 03 00 00 00 66 6f 6f 00  ..*.........foo.
00 21 00 00 00 03 00 00 00 62 61 72 6b 3d d9 7d  .!.......bark=.}
....
```

This page is licensed: CC BY-SA / Gnu FDL

XA_PREPARE_LOG_EVENT

An `XA_PREPARE_LOG_EVENT` records the prepare phase of a distributed transaction using the XA log. It is used to ensure atomicity and consistency of transactions.

Header

- Event Type is `XA_PREPARE_LOG_EVENT` (`0x26`).

Fields

- uint<1> One Phase Commit.
- uint<4> Format ID.
- uint<4> Length of gtrid.
- uint<1> Length of bqual.

Payload:

- byte<n> xid, where n is sum of gtrid and bqual lengths.

This page is licensed: CC BY-SA / Gnu FDL

XID_EVENT

An XID event is generated for a `COMMIT` of a transaction that modifies one or more tables of an XA-capable storage engine.

Header

- Event Type is `XID_EVENT` (`0x10`).

Fields

uint<8> The XID transaction number.

Complete Example With CRC32

```
ee b7 15 5a 10 01 00 00  00 1f 00 00 00 f2 0b 00  ...Z............
00 00 00 66 00 00 00 00  00 00 00 09 30 45 a8     ...f........0E.
```

Header, 19 Bytes

Event size is: header[19] + XID[8] + CRC32[4] of (header + xid).

- Event Time ee b7 15 5a => 5a 15 b7 ee => 1511372782 [2017-11-22 18:46:22].
- Event Type = 10.
- Server_id 01 00 00 00 => 1.
- Event Size 1f 00 00 00 => 31.
- Next Pos f2 0b 00 00 => 00 00 0b f2 => 3058.
- Flags 00 00 = 0.

Content, 8 Bytes

- XID 66 00 00 00 00 00 00 00 => 102.

CRC32, 4 Bytes

- 09 30 45 a8 => a8 45 30 09 => 2823106569.

This page is licensed: CC BY-SA / Gnu FDL

Product Development

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See MariaDB Quality Development Rules.

Outlines strict quality standards for new features, including requirements for design specifications, testability, and mandatory Worklog quality checklists.

See Uploading Package to PPA.

Provides instructions for developers to upload MariaDB source packages to a Personal Package Archive (PPA) using tools like dput for Ubuntu-based distributions.

See Debugging & Troubleshooting.

Offers deep-dive technical guides for diagnosing server issues, including trace file generation, debugger usage, and analyzing core dumps or memory usage.

See Plugin Development.

Details the APIs and processes for extending MariaDB functionality through custom plugins, such as authentication, logging, or specialized server enhancements.

See Server Development.

Covers foundational engineering topics, including the server roadmap, GitHub collaboration workflows, and quality assurance protocols for core contributions.

See MariaDB Internals.

Explores the low-level architecture of MariaDB, documenting source code internals, merging procedures, and how to use internal APIs in external programs.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Quality Development Rules

Those are quality-improving rules that everyone with a write access to the MariaDB repository is expected to follow:

- Respect previews
  - A feature can be pushed into an RC release X.Y.1 only after it was in an earlier preview. Normally, in X.Y.0, but generally any earlier preview will do.
- Tester sign-off for all new features
  - A feature being in the preview is a necessary, but not a sufficient condition. It needs to be tested (by a dedicated tester, not a developer) and the tester has to say it's good enough
  - Testing might discover bugs, that's normal, they have to be fixed before the feature is pushed (or — at the tester's discretion — they could be fixed after the push, if they're minor)
  - For tester's sake, develop in a dedicated branch with the version and the issue number in the branch name, for example, `bb-13.1-MDEV-11111`, and let the release master know when it's ready, so that they could cherry-pick it into a preview.
- Features must not be pushed directly into the GA release bypassing the above
  - Keep an eye on the release schedule (jira.mariadb.org) to know when the next release is due
  - Or simply remember that preview happen in mid-March/mid-June/mid-September/mid-December, innovation releases — in early February, early May, early August, early November, see mariadb-release-model.
- Get your commit reviewed.
  - If you don't know who would be the best reviewer for your PR, assign whoever github suggests — this person can reassign, if github was wrong
- Don't push into the red (in buildbot) branch
  - Fix failures first (or make sure they're fixed)
  - Eventually buildbot will evolve to simply not let you to
- Blocker issues block a release
  - we don't release if there's a Blocker bug open, that's why they're called blockers
  - so fix them asap, as your first priority, you don't want all the users to wait specifically for you
- There's no penalty for breaking these rules, we hope that everyone wants quality bug-free releases anyway
  - but if there will be serial violators, some kind of a penalty can be introduced later

This page is licensed: CC BY-SA / Gnu FDL

Uploading Package to PPA

After creating a Launchpad account:

Docker build, cloning the MariaDB repository and mapping it to the docker container

1. mkdir mariadb-source
2. cd mariadb-source
3. vi Dockerfile
4. Copy the following contents to Dockerfile:

```
# MariaDB 10.3 Ubuntu 17.10 build environment
# Published as mariadb-10-3-ubuntu-17.10-build-env
FROM ubuntu:17.10
RUN DEBIAN_FRONTEND=noninteractive apt-get update && \
apt-get install -y --no-install-recommends \
systemd \
devscripts \
build-essential \
lsb-release \
equivs \
git \
curl \
git-buildpackage \
nano \
vim \
pristine-tar

RUN curl -skO https://raw.githubusercontent.com/ottok/mariadb-10.1/ubuntu-17.10/debian/control

ENV GIT_SSL_NO_VERIFY true

RUN mk-build-deps -t 'apt-get -y -o Debug::pkgProblemResolver=yes --no-install-recommends' -i control

ENV container docker
ENV DEBIAN_FRONTEND noninteractive
ENV LANG C.UTF-8
ENV LC_ALL C.UTF-8
```

5. Run `docker build . -t ubuntu-17.10-packaging`
6. Do git clone of the latest repository:

`cd && mkdir debian-packaging && cd debian-packaging && git clone https://salsa.debian.org/mariadb-team/mariadb-10.1.git`

Generate, publish and upload PGP key

7. Generate OpenPGP key with the following command:

`$ gpg --gen-key`

- select (1) RSA and RSA
- Keysize: accept 2048
- Valid for: accept 0
- Type name, email and comment (comment is optional)
- Type 0
- Type passphrase (twice)
- Follow instructions to help generate a random key
- Keep the Key ID and fingerprint text, they are needed in the next step

Set generated key as default in `~/.bashrc`

`$ nano ~/.bashrc`

[.. add this ..]

`export GPGKEY=<key_id>`

Restart GPG-agent and source '/.bashrc', or restart session

8. Publish the key to the key server:

`gpg --keyserver keyserver.ubuntu.com --send-keys 12345678` and substitute `12345678` with your key's id

- If this gives timeout error, keep re-trying after a while

9. Upload the key's fingerprint here:
   Upload ImportingYourPGPKey fingerprint here +editpgpkeys
10. `gpg --export [your-key-id] > ~/debian-packaging/pub.key`
11. `gpg --export-secret-key [your-key-id] > ~/debian-packaging/secret.key`

`gpg -k`

^Should show the key

12. How to upload:How-to-upload-to-Launchpad-PPA-repository-(.deb-packages)
13. Open `/etc/devscripts.conf`

And look for this line:

`DEBSIGN_MAINT`

Uncomment it and add your name there

`export DEBEMAIL=[your-email-id]`

From inside the container

14. `docker run -v ~/debian-packaging/:/repo -it ubuntu-17.10-packaging bash`
15. `apt-get install devscripts`
16. `gpg --import pub.key`
17. `gpg --import secret.key`
18. `gpg -k`
19. `cd /repo/mariadb-10.1 && git fetch && git checkout pristine-tar && git checkout ubuntu-17.10`
20. `git clean -dffx && git reset --hard HEAD`
21. `export DEB_BUILD_OPTIONS="parallel=10 nocheck"` or `export DEB_BUILD_OPTIONS="parallel=5 nocheck"`
22. Go to `/repo` folder (Inside docker) and delete all the files except mariadb-10.1 folder:

`rm *`

23.

`gbp buildpackage`

For re-running the set up container

24. To generate an ID, run: `docker commit <container-id>` This will generate an ID.

For restarting the same container again use this ID: `docker run -v ~/debian-packaging/:/repo -it <ID> bash`

25. Last command for uploading package to PPA:

`backportpackage -u <your-ppa-address> -d <ubuntu-version-to-backport-to> -S ~<a-version-suffix-name-for-this-package> <the-most-recent-dsc-file>`

Example:

`backportpackage -u ppa:cvicentiu/mariadb-10.0-dev2 -d bionic -S ~testtry mariadb-10.1_10.1.30-0ubuntu0.17.10.1.dsc`

Run this command in the `/repo` folder, where the `.dsc` file is located
It should ask for the gpg key password again

- Docker tutorial available here:edit#slide=id.p4

This page is licensed: CC BY-SA / Gnu FDL

Debugging & Troubleshooting

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

Troubleshooting Guidance

Guidance for DBAs and developers, focused on diagnosis.

See Troubleshooting Overview.

See MariaDB Crashes.

See MariaDB Error Messages.

See Debugging Memory Usage.

Debugging Guidance

Guidance for developers working on MariaDB, focused on tools and workflows.

See Compiling MariaDB for Debugging.

See Creating a Trace File.

See Debugging a Running Server (on Linux).

See Debugging MariaDB With a Debugger.

See Enabling Core Dumps.

See Producing a Full Stack Trace for mariadbd.

See Using procmon to Trace mysqld.exe Filesystem Access.

See Limitations/Differences with a MariaDB Server Compiled for Debugging.

This page is licensed: CC BY-SA / Gnu FDL

Troubleshooting Overview

> **Note:**
>
> This page is focused on system-level development: It is geared toward developers or DBAs trying to diagnose why a specific instance is failing or crashing in a production or testing environment (e.g., interpreting error messages, handling crashes).

There are a number of ways MariaDB can result in a fault for you as the user. Getting a good clear understanding of the problem is the perfect way to get a good bug report to developers to allow them to fix your problem.

Like the MariaDB Community Bug Reporting says, start by searching for an existing bug. If its not found, or it not 100% clear that it applies to you, gain some additional information an include those in a new or perhaps and existing bug report.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Crashes

> **Note:**
>
> This page is focused on system-level development: It is geared toward developers or DBAs trying to diagnose why a specific instance is failing or crashing in a production or testing environment (e.g., interpreting error messages, handling crashes).

There are a number of causes of MariaDB crashes. Most of these are MariaDB encountering a situation where it cannot safely proceed. Crashes on Linux/Unix systems are fatal signals. On Windows these are similar looking exceptions.

InnoDB Semaphore wait

These are indicated by the following text in the error log:

```
[ERROR] [FATAL] InnoDB: Semaphore wait has lasted > 600 seconds. We intentionally crash the server
because it appears to be hung.
```

For this its essential that a full backtrace of [how-to-produce-a-full-stack-trace-for-mysqld/#getting-full-backtraces-for-all-threads-from-a-running-mariadbd-process|all the running threads].

Assertions

These look like `[ERROR] mariadbd got signal 6`. There might be associated text in the error log like:

```
InnoDB: Assertion failure in file ./storage/innobase/os/os0file.cc line 3540
InnoDB: Failing assertion: cb->m_err == DB_SUCCESS
```

On OSX it might show up as `Exception Type: EXC_CRASH (SIGABRT)`.

There maybe a stack trace, and perhaps a user SQL query (depending on the location). These are usually program errors that MariaDB got into a state where the developers did not expect to happen. Occasionally like the above, there are operating system factors that the developers haven't written handling routines for. In all cases this should be reported as a bug.

SEGV / Segment Violation

A SEGV, or Segment Violation, is an operating system concept in this case MariaDB accessing a location in memory it doesn't own. It will occur in the logs like:

`[ERROR] mysqld got signal 11 ;`

A full stack trace from the [how-to-produce-a-full-stack-trace-for-mysqld/#analyzing-a-core-file-with-gdb-on-linux|core file generated] is needed to resolve this form of error.

If there is a SQL query in the log file, include this in the bug report, along with `EXPLAIN query` and `SHOW CREATE TABLE tblname` for tables involved to enable a test case for this crash.

SIGBUS

This occurs when the alignment of memory doesn't correspond to the code accessing it.

`[ERROR] mysqld got signal 7`

Threat like SEGV.

SIGILL / Illegal Instruction

This indicates the MariaDB you have has been compiled for a CPU hardware feature that isn't available.

One cause is distributions occasionally target a minimum CPU version and if you are running on an earlier version it will crash in this way.

MariaDB does use optimizations in a number of critical paths under feature detection. Its possible that these aren't comprehensive for you your hardware. Very precise reporting of the hardware CPU model is required when reporting these bugs.

SIGKILL

This is an indicator that a user or OS killed of MariaDB. User invoked `SIGKILL` termination can be indirect using `podman kill {container}`.

The OS will kill MariaDB in an out of memory scenario in an attempt to regain memory. As MariaDB is a big memory user its usually fairly high on the list of processes killed by the OS.

If MariaDB is shutting down, it might be possible under some service manager that if it takes too long, it will get a SIGKILL. The default MariaDB systemd service has `SendSIGKILL=No`.

Galera SST scripts also have a `kill -9` in them.

Generally the SIGKILL won't be a bug, however the slow time that resulted in a service manager taking this drastic step might be.

Hardware faults

This can be disk corruption where MariaDB reads data and that data wasn't what was written. This will result in MariaDB doing unexpected things, including potentially accessing unallocated memory.

RAM corruptions can also result in undefined behaviour that is unfixable with software.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Error Messages

> **Note:**
>
> This page is focused on system-level development: It is geared toward developers or DBAs trying to diagnose why a specific instance is failing or crashing in a production or testing environment (e.g., interpreting error messages, handling crashes).

MariaDB has a number of error messages. If the meaning is not clear, use the following table to guide you to the desired actions.

Can't lock aria control file, error 11

`[ERROR] mysqld: Can't lock aria control file '/var/lib/mysql/aria_log_control' for exclusive use, error: 11. Will retry for 30 seconds`

The `perror 11` is 'Resource temporarily unavailable'. This is an indication that an operating system lock couldn't be obtained on the ariia_log_control file. This is a strong indication that an existing MariaDB server is already running on the datadir.

Look at the operating system process list. Identify if the existing mysqld/mariadbd process is shutting down (look at the logs, and if its listening to new connections). MariaDB will stop accepting new SQL connections immediately once the shutdown starts.

If its shutting down, its best to let it continue. Force killing off the process will cause a crash recovery on the next restart. If you are about to upgrade there is a possibility your next upgraded version will not complete the crash recovery and you will need the current version to complete the log recovery.

If it completing the shutdown, take two [how-to-produce-a-full-stack-trace-for-mysqld/#getting-full-backtraces-for-all-threads-from-a-running-mariadbd-process|backtraces] with 30 seconds to a minute between them.

Can't create/write/read to file …. .. Errcode X

`[ERROR] mariadbd: Can't create/write to file '/var/run/mariadb/mariadb.pid' (Errcode: 2 "No such file or directory")`.

The errcode can be look up with `[perror](https://mariadb.com/docs/server/clients-and-utilities/perror) X` to return a text description of the code if not already there. In may cases there are directly the same as operating system errors. Common ones are:

- Error 2 - No such file or directory. MariaDB is expecting to read/write/create a file somewhere and it doesn't exist. If creating, the directory doesn't exist.
- Error 13 - EPERM, permission denied. Either file permissions or an access control mechanism like selinux or apparmor or systemd is preventing access to this location. Alternately you are running MariaDB as the incorrect user for the data it is trying to access.
- Error 28 - ENOSPC - no space left on device. The filesystem where writing is occurring is full and cannot fit any more data.

[ERROR] InnoDB: Invalid flags 0x4800 in ./ibdata1

This indicates that MariaDB is starting on a MySQL-8.0 data directory. Other flags are possible if a different page size or different features (see MDEV-27882).

MariaDB cannot startup on MySQL-8.0 data directory. MySQL should be run on the MySQL data directory and if the objective is to migrate, take logical dumps from MySQL before recreating a MariaDB database, perhaps in a different location, or after the MySQL data directory has been moved.

This page is licensed: CC BY-SA / Gnu FDL

Debugging Memory Usage

> **Note:**
>
> This page is focused on system-level development: It is geared toward developers or DBAs trying to diagnose why a specific instance is failing or crashing in a production or testing environment (e.g., interpreting error messages, handling crashes).

Troubleshooting Memory Usage

This page describes how to debug MariaDB's memory usage. It uses CentOS 7 but can be applied to other systems as well.

Use Google PerfTools for debugging memory usage: heapprofile.html

On CentOS, install PerfTools like this:

```
sudo yum install gperftools
service mariadb stop
systemctl edit mariadb
```

In the editor that opens, add and save this content:

```
[Service]
Environment="HEAPPROFILE=/tmp/heap-prof-1"
Environment="HEAP_PROFILE_ALLOCATION_INTERVAL=10737418240"
Environment="HEAP_PROFILE_INUSE_INTERVAL=1073741824"
Environment="LD_PRELOAD=/usr/lib64/libtcmalloc.so.4"
```

Then run this command:

```
service mariadb start
```

Now, run the workload. When memory consumption becomes large enough, run this command:

```
ls -la /tmp/heap-prof-*
```

This should show several files. Copy the last file from the list:

```
cp /tmp/heap-prof-1.0007.heap .
```

Then, run this command:

```
pprof --dot /usr/sbin/mysqld heap-prof-1.0007.heap  > 7.dot
```

> **Note:**
>
> This produces a lot of lines, like `/bin/addr2line: Dwarf Error: ...`.

Then, please send us the `7.dot` file.

Memory Leak Overview

Operating system memory can be assigned, giving MariaDB a virtual address space, and allocated (paged in), once MariaDB starts to use this memory.

When looking at something that might be a memory leak, pay particular attention between the virtual size and the resident size.

When MariaDB has the same virtual size, but the resident size increases over time, this is a good indication that the memory buffers allocated to MariaDB are slowly being used. In this case its unlikely to be a memory leak.

Because memory leaks are hard to track down, documenting a known good version and a version that is leaking memory can make it easier to examine code changes in those versions against some other criteria like those obtained from the sections below to identify the cause.

Performance Schema

The memory instrumentation of the performance schema can be enable with the following:

```
performance_schema=on
performance-schema-instrument='memory/%=ON'
```

This is available since MariaDB 10.5.2. With this enabled, performance_schema.memory_summary_global_by_event_name can start to show where the memory leak is occurring.

Memleak

`memleak` is a BPF program for Linux that is frequently packaged like bcc-tools or similar (bpfcc-tools on Debian).

The upstream documentation provides this example.

The important aspect on measurement is to let MariaDB startup, and commence an normal workload before measurement. Starting too early or before the caches are populated is likely to result in false recording of memory allocations that will be eventually released.

Increasing the time interval of `memleak` is important to run longer than the usual query. If your memory is leaking over hours, then a high interval of 10 minutes so will reduce memory temporary allocated for a query. Recording for a longer time or multiple times should provide enough information to narrow down the location of the leak.

> **Note:**
>
> Very old Linux kernels like Centos/RHEL 7 might not have sufficient hooks to measure this correctly.

This page is licensed: CC BY-SA / Gnu FDL

Compiling MariaDB for Debugging

> **Note:**
>
> This page is focused on lower-level development: It contains technical “how-to” documentation for engineers who are actually modifying the MariaDB source code (e.g., compiling for debug, using GDB, and memory leak detection in the code).

Compiling MariaDB for Debugging Using the `CMAKE_BUILD_TYPE` Option

This option enables multiple debug instrumentation aspects within the MariaDB server that provided more detailed information around complex parts of the server and can be used to implement and run tests where the concurrent execution of multiple threads must be controlled to achieve a specific state. If you are not doing this, the following option is sufficient.

Compiling MariaDB with full debug information includes all code symbols and also new code to do internal testing of structures and allow one to trace MariaDB execution. A full debug binary will be notably slower than a normal binary (30%). Most of this overhead can be removed by disabling `-DWITH_DBUG_TRACE=OFF`

You can configure build a debug build by executing `cmake` and by setting the `CMAKE_BUILD_TYPE` option to `Debug`. For example:

```
cmake -DCMAKE_BUILD_TYPE=Debug source_directory
```

To compile:

```
cmake --build .
```

You can find a list of the needed packages/libraries for building on Linux here.

Building Optimized Build With Debug Symbols

To build MariaDB with debug symbols, to get better stack traces and to be able to debug the binary with `gdb`, you need to supply the -Og and -g3 options to the `gcc` compiler for an debuggable, with limited optimization tradeoffs..

Pass these are options to cmake like:

```
cmake -DCMAKE_CXX_COMPILE_FLAGS='-Og -g3' -DCMAKE_C_COMPILE_FLAGS='-Og -g3' source_directory
```

To compile:

```
cmake --build .
```

Doing a Debug Build on Debian/Ubuntu

To build a “mariadbd” binary with debugging enabled that uses the same parameters as the ones used in Debian/Ubuntu binary packages, you must do as follows (you must have a deb-src line of one of the MariaDB repositories on your /etc/apt/sources.list in order to do that):

```
apt install build-essential devscripts fakeroot dpkg-dev
apt-get build-dep mariadb-server
apt-get source mariadb-server
cd mariadb-*
 DEB_CFLAGS_APPEND='-Og -g3' DEB_CXXFLAGS_APPEND='-Og -g3' debuild -us -uc
```

The packages created will have these flags set.

Temporarily Installing your Debug Build

The commands shown below replace the release `mariadbd` binary with the debug `mariadbd` binary that you compiled. Most importantly, they replace the binary in a way which makes it trivial to revert back to the original release `mariadbd` binary.

First, stop MariaDB.

Then, use the `mv` utility to rename the release `mariadbd` binary:

```
sudo mv /usr/sbin/mariadbd /usr/sbin/mariadbd-orig
```

Note: Do not use the `cp` utility because that will change the file modification timestamp.

Then, install the debug `mariadbd` binary from your source tree:

```
sudo install ~/mariadb-*/sql/mariadbd /usr/sbin/mariadbd
```

Then, start MariaDB.

Be sure to replace `/usr/sbin/mariadbd` with the path to your `mariadbdd` binary and to also replace `~/mariadb-*/sql/mariadbd` with the path to your debug #mariadbd `binary.`

Reinstalling your Release Build

If you want to restore your original `mariadbd` binary, you can do it with the following process::

First, stop MariaDB.

Then, execute the following command to delete the symbolic link:

```
sudo mv /usr/sbin/mariadbd /usr/sbin/mariadbd-debug
```

Then, execute the following command to move the original `mariadbd` release binary back into place:

```
sudo mv /usr/sbin/mariadbd-orig /usr/sbin/mariadbd
```

Then, start MariaDB.

Be sure to replace `/usr/sbin/mariadbd` with the path to your `mariadbd` binary

The debug `mariadb-debug` binary is still present if it is needed again in the future.

Different Compilation Options

Changing DBUG_ASSERT to Print to Error Log

A debug binary has lots of code checks and asserts, that are not checked in production. This is done to get more performance when running in production.
In some cases, when one is trying to find a hard-to-repeat bug, it could be beneficial to have these checks in production builds too.

Compiling with `-DDBUG_ASSERT_AS_PRINTF` will change DBUG_ASSERT() to print any failed check to the error log.

```
cmake . -DDBUG_ASSERT_AS_PRINTF
```

Enabling the above option should not have any notable impact on performance (probably < 1% slowdown).
This is achieved by grouping asserts in MariaDB server code into two groups:

- Fast checks, using `DBUG_ASSERT()`: These are converted to printing to error log.
- Slow checks, using `DBUG_SLOW_ASSERT()`. These will always be removed in production builds.

See Also

- Build environment setup for Linux
- Debugging MariaDB with a debugger
- Creating a trace file
- Using ASAN with MariaDB

This page is licensed: CC BY-SA / Gnu FDL

Creating a Trace File

> **Note:**
>
> This page is focused on lower-level development: It contains technical “how-to” documentation for engineers who are actually modifying the MariaDB source code (e.g., compiling for debug, using GDB, and memory leak detection in the code).

If `mysqld` is crashing, creating a trace file is a good way to find the issue.

A `mysqld` binary that has been compiled with debugging support can create trace files using the DBUG package created by Fred Fish. To find out if your `mysqld` binary has debugging support, run `mysqld -V` on the command line. If the version number ends in `-debug` then your `mysqld` binary was compiled with debugging support.

See Compiling MariaDB for debugging for instructions on how to create your own `mysqld` binary with debugging enabled.

To create the trace log, start `mysqld` like so:

```
mysqld --debug
```

Without options for –debug, the trace file will be named `/tmp/mysqld.trace` in MySQL and older versions of MariaDB before 10.5 and `/tmp/mariadbd.trace` starting from MariaDB 10.5.

On Windows, the debug `mysqld` is called `mysqld-debug` and you should also use the `--standalone` option. So the command on Windows will look like:

```
mysqld-debug --debug --standalone
```

Once the server is started, use the regular `mysql` command-line client (or another client) to connect and work with the server.

After you are finished debugging, stop the server with:

```
mysqladmin shutdown
```

DBUG Options

Trace files can grow to a significant size. You can reduce their size by telling the server to only log certain items.

The `--debug` flag can take extra options in the form of a colon (:) delimited string of options. Individual options can have comma-separated sub-options.

For example:

```
mysqld --debug=d,info,error,query:o,/tmp/mariadbd.trace
```

The '`d`' option limits the output to the named DBUG_ macros. In the above example, the `/tmp/mariadbd.trace` tracefile will contain output from the info, error, and query DBUG macros. A '`d`' by itself (with no sub-options) will select all DBUG_ macros.

The '`o`' option redirects the output to a file (`/tmp/mariadbd.trace` in the example above) and overwrites the file if it exists.

See Also

- Options for –debug

This page is licensed: CC BY-SA / Gnu FDL

Debugging a Running Server (on Linux)

> **Note:**
>
> This page is focused on lower-level development: It contains technical "how-to" documentation for engineers who are actually modifying the MariaDB source code (e.g., compiling for debug, using GDB, and memory leak detection in the code).

Even if you don't have a server that is compiled for debugging, there are still ways to get more information out of it if things go wrong.

When things go wrong, it's always better to have a version of the mysqld daemon that is not stripped.

```
shell> file /usr/sbin/mysqld
```

If this doesn't say 'stripped', then you are fine. If not, you should either download a binary with debugging information or compile it, without stripping the binary.

Debugging Memory Consumption With tcmalloc

Read the Profiling Memory Usage page for more information on how to debug high memory consumption.

If you have a problem with a mysqld process that keeps on growing, you can use tcmalloc to find out what is allocating memory:

Depending on the system, you have to install the `tcmalloc` (openSUSE) or the `google-perftools-lib` (RedHat, CentOS) package.

The following set of commands starts mysqld with memory profiling, and if you kill it with SIGABRT, you will get a core dump that you can examine:

```
HEAPPROFILE=/tmp/mysqld.prof /usr/sbin/mysqld_safe --malloc-lib=tcmalloc --core-file-size=unlimited --
core-file
```

or if you prefer to invoke mysqld directly:

```
ulimit -c unlimited
LD_PRELOAD=/usr/lib64/libtcmalloc.so.4 HEAPPROFILE=/tmp/mysqld.prof /usr/sbin/mysqld --core-file
```

You can, of course, add other mysqld options to the end of the above line.

Now start your client/application that uses MariaDB. You can find where memory is allocated in the `/tmp/mysqld.prof` file. If you find any memory issues, please report them in the MariaDB bug tracker!

ptrace Protection and Attaching GDB to a mysqld Instance

New Ubuntu releases do not allow one process to examine the memory of an arbitrary user's process. As a result, when trying to attach GDB to a running
MariaDB (or any other process) instance, one gets the following error in GDB:

```
ptrace: Operation not permitted
```

More details are available in the Ubuntu Wiki.

To allow GDB to attach, one needs to edit the value of the `/proc/sys/kernel/yama/ptrace_scope` sysctl value.

- To change it temporarily, open a root shell and issue:

```
echo 0 > /proc/sys/kernel/yama/ptrace_scope
```

- To change it permanently, edit as root:

```
/etc/sysctl.d/10-ptrace.conf
```

and set the value to `0`.

Debugging a Server That Hangs

If your mysqld server hangs, you may want to debug it to know what happened.

Preferably, the server should be compiled for debugging, but it's not strictly necessary:

```
cmake -DCMAKE_BUILD_TYPE=Debug -DWITH_VALGRIND=ON .
make -j4
```

To know what the server is doing:

- Find out the process number of mysqld

```
ps -edalf | grep mysqld
```

- Attach to the process and get a back trace:

```
gdb -p 'pid of mysqld' path-to-mysqld
set height 0
set logging file /tmp/mysqld.log
set logging on
thread apply all backtrace full
```

After the above, you have a full backtrace, including all local variables, in the `mysqld.log` file. Note that you will only get all variables if the server is not stripped.

This page is licensed: CC BY-SA / Gnu FDL

Debugging MariaDB With a Debugger

> **Note:**
>
> This page is focused on lower-level development: It contains technical "how-to" documentation for engineers who are actually modifying the MariaDB source code (e.g., compiling for debug, using GDB, and memory leak detection in the code).

If you have MariaDB compiled for debugging you can both use it in a debugger, like ddd or gdb, and get comprehensive trace files of the execution of MariaDB. The trace files allow you to both see the flow of the code and to see the differences in execution by comparing two trace files.

Core dumps are also much easier to investigate if they come from a debug binary.

Note that a binary compiled for debugging and tracing is about 10-20% slower than a normal binary. If you just compile a binary for debugging (option `-g` with gcc) the speed difference compared to a normal binary is negligible.

Checking That MariaDB is Compiled For Debugging

Execute:

```
mariadbd --debug --help
```

If you are using MariaDB before 10.5, then you should use `mysqld` instead of `mariadbd`!

If you get an error `unknown option '--debug`, then MariaDB is not compiled for debugging and tracing.

Building MariaDB for Debugging Starting from 5.5

On Unix you need to pass `-DCMAKE_BUILD_TYPE=Debug` to cmake to compile with debug information.

Building MariaDB 5.3 and Older

Here is how you compile with debug on older versions:

Use the scripts in the BUILD directory that will compile MariaDB with most common debug options and plugins, for example:

```
./BUILD/compile-pentium64-debug-max
```

For the most common configurations there exists a fine-tuned script in the BUILD directory.

If you want to use valgrind, a very good memory instrumentation tool and memory overrun checker, you should use

```
./BUILD/compile-pentium64-valgrind-max
```

Some recommended debugging scripts for Intel/AMD are:

```
BUILD/compile-pentium64-debug-max
BUILD/compile-pentium64-valgrind-max
```

This is an example of how to compile MariaDB for debugging in your home directory with MariaDB 5.2.9 as an example:

```
cd ~
mkdir mariadb
cd mariadb
tar xvf mariadb-5.2.9.tar.gz
ln -s mariadb-5.2.9 current
cd current
./BUILD/compile-pentium64-debug-max
```

The last command will produce a debug version of `sql/mysqld`.

Debugging MariaDB From the Source Directory

Creating the MariaDB Database Directory

The following example creates the MariaDB databases in `/data`.

```
./scripts/mariadb-install-db --srcdir=. --datadir=/data
```

Running MariaDB in a Debugger

The following example is using `ddd`, an excellent graphical debugger in Linux. If you don't have `ddd` installed, you can use `gdb` instead.

```
cd sql
ddd ./mariadbd &
```

In `ddd` or `gdb`

```
run --datadir=/data --language=./share/english --gdb
```

You can set the options in your /.my.cnf file so as not to have to repeat them on the `run` line.

If you run `mysqld` with `--debug`, you will get a trace file in /tmp/mysqld.trace that shows what is happening.

Note that you can have different options in the configuration file for each MariaDB version (like having a specific language directory).

Debugging MariaDB Server with mariadb-test-run

If you get a crash while running `mariadb-test-run` you can debug this in a debugger by using one of the following options:

```
mariadb-test-run --gdb failing-test-name
```

or if you prefer the `ddd` debugger:

```
mariadb-test-run --ddd failing-test-name
```

Sample .my.cnf file to Make Debugging Easier

```
[client-server]
socket=/tmp/mysql-dbug.sock
port=3307

[mariadb]
datadir=/my/data
loose-innodb_file_per_table
server_id= 1
log-basename=master
loose-debug-mutex-deadlock-detector
max-connections=20
lc-messages=en_us

[mariadb-10.0]
lc-messages-dir=/my/maria-10.0/sql/share

[mariadb-10.1]
lc-messages-dir=/my/maria-10.1/sql/share

[mariadb-10.2]
lc-messages-dir=/my/maria-10.2/sql/share

[mariadb-10.3]
lc-messages-dir=/my/maria-10.3/sql/share
```

The above `.my.cnf` file:

- Uses an explicit socket for both client and server.
- Assumes the server source is in /my/maria-xxx. You should change this to point to where your sources are located.
- Has a unique patch for each MariaDB version so that one doesn't have to specify –lc-messages-dir or –language even if one switches between debugging different MariaDB versions.

See Also

- Creating a trace file
- Configuring MariaDB with my.cnf
- Running mariadbd from the build director

This page is licensed: CC BY-SA / Gnu FDL

Enabling Core Dumps

> **Note:**
>
> This page is focused on lower-level development: It contains technical "how-to" documentation for engineers who are actually modifying the MariaDB source code (e.g., compiling for debug, using GDB, and memory leak detection in the code).

Enabling in an Option File

Core dumps are enabled by default on **Windows**, so **this step can be skipped on Windows** in those versions. See MDEV-18439 for more information.

In order to enable core dumps, you need to set the core_file system variable either on the command-line or in a relevant server option group in an option file. For example:

```
[mariadb]
...
core_file
```

You can check your current value by executing:

```
my_print_defaults --mariadbd
```

core_file is a system variable. Its value can be checked at runtime by executing the following:

```
SHOW GLOBAL VARIABLES LIKE 'core_file';
```

Core Files on Linux

There are some additional details related to using core files on Linux.

Disabling Core File Size Restrictions on Linux

On some systems there is a limit on the sizes of core files that can be dumped. You can check the system's current system-wide limit by executing the following:

```
ulimit -c
```

You can check the current limit of the `mariadbd` process specifically by executing the following:

```
sudo cat /proc/$(pidof mariadbd)/limits | grep "core file"
```

If you need to change the core size limit, the method you use depends on how you start `mariadbd`. See the sections below for more details.

The resource limits for the `mariadbd` process are printed to the error log when the `mariadbd` process crashes. That way, users can confirm whether the process may have been allowed to dump a core file. See MDEV-15051 for more information.

Running mariadbd Using mysqld_safe

If you are starting MariaDB by running mysqld_safe, then configuring the following in the `[mysqld_safe]` option group in an option file should allow for unlimited sized core files:

```
[mysqld_safe]
...
core_file_size=unlimited
```

You can check your current values by executing:

```
my_print_defaults mysqld_safe
```

See mysqld_safe: Configuring the Core File Size for more details.

Note: If you are using mysqld_safe and running `mariadbd` as the `root` user, then no core file is created on some systems. The solution is to run `mariadbd` as another user.

Running mariadbd Manually

If you are starting mariadbd manually or in a custom script, then you can allow for unlimited sized core files by executing the following in the same shell or script in which mariadbd is executed:

```
ulimit -c unlimited
```

Running mariadbd Using systemd

If you are starting `mariadbd` using systemd, then you may need to customize the MariaDB service to allow for unlimited size core files. For example, you could execute the following:

Using `sudo systemctl edit mariadb.service` add the contents:

```
[Service]

LimitCORE=infinity
```

See systemd: Configuring the Core File Size for more details.

Running MariaDB Containers

To get a core dump in a `mariadb` container requires setting the path on Linux to not include a `sysctl kernel.core_pattern` beginning with a pipe to an executable that doesn't exist in the container. Setting to a straight `core` is recommended.

Also see Container with Debug Symbols.

Changing the System-Wide Limit

If you want to change the system-wide limit to allow for unlimited size core files for the `mysql` user account, then you can do so by adding the following lines to a file in /etc/security/limits.d/. For example:

```
sudo tee /etc/security/limits.d/mariadb_core.conf <<EOF
mysql soft core unlimited
mysql hard core unlimited
EOF
```

The system would have to be restarted for this change to take effect.

See Configuring Linux for MariaDB: Configuring the Core File Size for more details.

Setting the Path on Linux

If you are using Linux, then it can be helpful to change a few settings to alter where the core files is written and what file name is used. This is done by setting the `kernel.core_pattern` and `kernel.core_uses_pid` attributes. You can check the current values by executing the following:

```
sysctl kernel.core_pattern
sysctl kernel.core_uses_pid
```

If you are using `mysql-test-run` and want to have the core as part of the test result, the optimal setting is probably the following (store cores in the current directory as `core.number-of-process-id`):

```
sudo sysctl kernel.core_pattern=core.%p kernel.core_uses_pid=0
```

If you are using a production system, you probably want to have the core files in a specific directory, not in the data directory. They place to store cores can be temporarily altered using the sysctl utility, but it is often more common to alter them via the /proc file system. See the following example:

```
sudo mkdir /tmp/corefiles
sudo chmod 777 /tmp/corefiles
sudo sysctl kernel.core_pattern=/tmp/corefiles/core
sudo sysctl kernel.core_uses_pid=1
```

The above commands will tell the system to put core files in `/tmp/corefiles`, and it also tells the system to put the process ID in the file name.

If you want to make these changes permanent, then you can add the following to a file in /etc/sysctl.conf.d/. For example:

```
sudo tee /etc/sysctl.d/mariadb_core.conf <<EOF
kernel.core_pattern=/tmp/corefiles/core
kernel.core_uses_pid=1
EOF
```

Note: if you are using containers, the pid is always going to be 1, so this may not be a useful setting. Appending an identifier like %t to the kernel.core_pattern will generate more unique files.

The value of `kernel.core_pattern` is printed to the error log when the `mariadbd` process crashes. That way, users can determine where the process may have dumped a core file. See MDEV-15051 for more information.

Note: Ensure that you have enough free disk space in the path pointed to by `kernel.core_pattern`.

Extracting Linux core dumps with systemd-coredump

Core dump management can be automated using systemd, which then centrally manages all core dump files and provides information about detected core dumps and access to collected core files using the coredumpctl utility.

This is enabled per default on Red Hat Enterprise Linux 8 and CentOS 8, and maybe other contemporary Linux distribution releases by now, too. It can be easily checked for by looking at the `kernel.core_pattern setting`. If it looks like this systemd-coredump is enabled:

```
# sysctl kernel.core_pattern
kernel.core_pattern = |/usr/lib/systemd/systemd-coredump %P %u %g %s %t %c %h %e
```

On other distributions like Ubuntu (at least up to 21.10) it is not enabled by default, but can be set up manually.

To see all recent core dumps on the system you can then simply run

```
# coredumpctl list
```

Or you can check for MariaDB Server core dumps specifically with:

```
# coredumpctl list /usr/sbin/mariadbd
```

If an actual core file got stored you'll see `present` in the COREFILE column of the output, you can then extract the core file with:

```
# coredumpctl dump -o mariadbd.core ...PID...
```

using the process id number from the PID column, or when you just want to retrieve the latest MariaDB Server related entry:

```
# coredumpctl dump -o mariadb.core /usr/sbin/mariadbd
```

Starting with `systemd` 248 it is also possible to invoke the `gdb` debugger directly using the new `--debugger-arguments=...` option, e.g. making the extraction of all thread backtraces from the most recent MariaDB server crash a one liner without even having to extract the core dump file first (requires `gdb` to be installed):

```
# coredumpctl debug --debugger-arguments="-batch -ex 'thread apply all bt full'" /usr/sbin/mariadbd
```

So far none of the long term support Linux distribution releases have a new enough `systemd` version for this, the (as of this writing) still upcoming Ubuntu 22.04 "Jammy Jellyfish" will probably the first to support this.

Core Dumps and setuid on Linux

Since `mariadbd` executes setuid, you may have to set `fs.suid_dumpable=2` to allow core dumps on Linux. You can check the current `fs.suid_dumpable` value by using the sysctl utility. For example:

```
sysctl fs.suid_dumpable
```

You can temporarily set it to `2` by using the sysctl utility. For example:

```
sudo sysctl -w fs.suid_dumpable=2
```

Or you can temporarily set it to `2` by writing to the /proc file system. For example:

```
sudo echo 2 > /proc/sys/fs/suid_dumpable
```

If you want to permanently set it to `2` then you can add the following to a file in /etc/sysctl.conf.d/:

```
sudo tee /etc/sysctl.d/mariadb_fs_suid_dumpable.conf <<EOF
fs.suid_dumpable=2
EOF
```

Note: If you don't want to change `fs.suid_dumpable`, then another solution is to start `mariadbd` directly as the `mysql` user, so that the setuid call is not needed.

Forcing a Core File on Linux

You can get a core dump from a running server with:

```
sudo gcore -o filename $(pidof mariadbd)
```

This will store a core dump in `filename.pid` where pid is the process ID of `mariadbd`. `mariadbd` will continue to be running and will not be affected by `gcore`.

Another method is to force a core file for `mariadbd` by sending the process the `sigabrt` signal, which has the signal code `6`. This is very useful to get the state of the unresponsive `mariadbd` process. However, this will cause `mariadbd` to crash, and crash recovery will be run on restart.

You can send the signal with the kill command. For example:

```
sudo kill -6 $(pidof mariadbd)
```

As an alternative to `$(pidof mariadbd)`, you can find the process ID either by using the ps utility or by checking the file defined by the pid_file system variable.

Core Files on Windows

Core dumps are enabled by default on Windows. See MDEV-18439 for more information.

There are some additional details related to using core files on Windows.

Minidump Files on Windows

On Windows, the core file is created as a minidump file.

For details on how to configure and read the minidump file, see How to read the small memory dump file that is created by Windows if a crash occurs.

Core Files on Kubernetes

See the IBM Core Dump Handler project.

Core Files and Address Sanitizer (ASAN)

If your `mariadbd` binary is built with Address Sanitizer (ASAN) then it will disable core dumps by default. A core dump can be created if `mariadbd` is started with the environment variable `ASAN_OPTIONS` set with `disable_coredump=0`.

What's Included in Core Files

Core files usually contain a dump of all memory in the process's full address space. This means that if a server has some large buffers configured (such as a large InnoDB buffer pool), then the server's core files can get very large.

Some large buffers have been excluded from core files on some systems as a way to reduce the size.

The following buffers are excluded:

- InnoDB buffer pool
- InnoDB log buffer
- InnoDB Redo log buffer (fixed 2M)
- Query cache

The buffers are only excluded on Linux when using kernel version 3.4 and above and when using a non-debug build of `mariadbd`. Some Linux kernel versions have a bug which would cause the following warning to be printed to the log:

```
Sep 25 10:41:19 srv1 mysqld: 2018-09-25 10:41:19 0 [Warning] InnoDB: Failed to set memory to DODUMP:
Invalid argument ptr 0x2aaac3400000 size 33554432
```

In those cases, the core dump may exclude some additional data. If that is not a concern, then the warning can be ignored. The problem can be fixed by upgrading to a Linux kernel version in which the bug is fixed.

See Also

- How to Produce a Full Stack Trace for mariadbd
- HowTo: Debug Crashed Linux Application Core Files Like A Pro
- A Nice Feature in MariaDB 10.3: no InnoDB Buffer Pool in Core Dumps
- Getting MySQL Core file on Linux

This page is licensed: CC BY-SA / Gnu FDL

Producing a Full Stack Trace for mariadbd

> **Note:**
>
> This page is focused on lower-level development: It contains technical "how-to" documentation for engineers who are actually modifying the MariaDB source code (e.g., compiling for debug, using GDB, and memory leak detection in the code).

Partial Stack Traces in the Error Log

When `mariadbd` crashes, it will write a stack trace in the error log by default. This is because the stack_trace option defaults to `ON`. With a normal release build, this stack trace in the error log may look something like this:

```
2019-03-28 23:31:08 0x7ff4dc62d700  InnoDB: Assertion failure in file
/home/buildbot/buildbot/build/mariadb-10.2.23/storage/innobase/rem/rem0rec.cc line 574
InnoDB: We intentionally generate a memory trap.
InnoDB: Submit a detailed bug report to https://jira.mariadb.org/
InnoDB: If you get repeated assertion failures or crashes, even
InnoDB: immediately after the mysqld startup, there may be
InnoDB: corruption in the InnoDB tablespace. Please refer to
InnoDB: https://mariadb.com/kb/en/library/innodb-recovery-modes/
InnoDB: about forcing recovery.
190328 23:31:08 [ERROR] mysqld got signal 6 ;
This could be because you hit a bug. It is also possible that this binary
or one of the libraries it was linked against is corrupt, improperly built,
or misconfigured. This error can also be caused by malfunctioning hardware.

To report this bug, see https://mariadb.com/kb/en/reporting-bugs

We will try our best to scrape up some info that will hopefully help
diagnose the problem, but since we have already crashed,
something is definitely wrong and this may fail.

Server version: 10.2.23-MariaDB-10.2.23+maria~stretch
key_buffer_size=134217728
read_buffer_size=131072
max_used_connections=234
max_threads=752
thread_count=273
It is possible that mysqld could use up to
key_buffer_size + (read_buffer_size + sort_buffer_size)*max_threads = 1783435 K  bytes of memory
Hope that's ok; if not, decrease some variables in the equation.

Thread pointer: 0x7ff4d8001f28
Attempting backtrace. You can use the following information to find out
where mysqld died. If you see no messages after this, something went
terribly wrong...
stack_bottom = 0x7ff4dc62ccc8 thread_stack 0x49000
*** buffer overflow detected ***: /usr/sbin/mysqld terminated
======= Backtrace: =========
/lib/x86_64-linux-gnu/libc.so.6(+0x70bfb)[0x7ffa09af5bfb]
/lib/x86_64-linux-gnu/libc.so.6(__fortify_fail+0x37)[0x7ffa09b7e437]
/lib/x86_64-linux-gnu/libc.so.6(+0xf7570)[0x7ffa09b7c570]
/lib/x86_64-linux-gnu/libc.so.6(+0xf93aa)[0x7ffa09b7e3aa]
/usr/sbin/mysqld(my_addr_resolve+0xe2)[0x55ca42284922]
/usr/sbin/mysqld(my_print_stacktrace+0x1bb)[0x55ca4226b1eb]
/usr/sbin/mysqld(handle_fatal_signal+0x41d)[0x55ca41d0a01d]
/lib/x86_64-linux-gnu/libpthread.so.0(+0x110e0)[0x7ffa0b4180e0]
/lib/x86_64-linux-gnu/libc.so.6(gsignal+0xcf)[0x7ffa09ab7fff]
/lib/x86_64-linux-gnu/libc.so.6(abort+0x16a)[0x7ffa09ab942a]
/usr/sbin/mysqld(+0x40f971)[0x55ca41ab8971]
/usr/sbin/mysqld(+0x887df6)[0x55ca41f30df6]
/usr/sbin/mysqld(+0x863673)[0x55ca41f0c673]
/usr/sbin/mysqld(+0x96648e)[0x55ca4200f48e]
/usr/sbin/mysqld(+0x89b559)[0x55ca41f44559]
/usr/sbin/mysqld(+0x8a15e4)[0x55ca41f4a5e4]
/usr/sbin/mysqld(+0x8a2187)[0x55ca41f4b187]
/usr/sbin/mysqld(+0x8b1a20)[0x55ca41f5aa20]
/usr/sbin/mysqld(+0x7f5c04)[0x55ca41e9ec04]
/usr/sbin/mysqld(_ZN7handler12ha_write_rowEPh+0x107)[0x55ca41d140d7]
/usr/sbin/mysqld(_Z12write_recordP3THDP5TABLEP12st_copy_info+0x72)[0x55ca41b4b992]
/usr/sbin/mysqld(_Z12mysql_insertP3THDP10TABLE_LISTR4ListI4ItemERS3_IS5_ES6_S6_15enum_duplicatesb+0x1206)
[0x55ca41b560f6]
/usr/sbin/mysqld(_Z21mysql_execute_commandP3THD+0x3f68)[0x55ca41b6bee8]
/usr/sbin/mysqld(_Z11mysql_parseP3THDPcjP12Parser_statebb+0x28a)[0x55ca41b70e4a]
/usr/sbin/mysqld(+0x4c864f)[0x55ca41b7164f]
/usr/sbin/mysqld(_Z16dispatch_command19enum_server_commandP3THDPcjbb+0x1a7c)[0x55ca41b737fc]
/usr/sbin/mysqld(_Z10do_commandP3THD+0x176)[0x55ca41b748a6]
/usr/sbin/mysqld(_Z24do_handle_one_connectionP7CONNECT+0x25a)[0x55ca41c3ec0a]
/usr/sbin/mysqld(handle_one_connection+0x3d)[0x55ca41c3ed7d]
/usr/sbin/mysqld(+0xb75791)[0x55ca4221e791]
```

```
/lib/x86_64-linux-gnu/libpthread.so.0(+0x74a4)[0x7ffa0b40e4a4]
/lib/x86_64-linux-gnu/libc.so.6(clone+0x3f)[0x7ffa09b6dd0f]
```

If you plan to report a bug about the problem, then this information can be very useful for MariaDB's developers to track down the root cause. However, notice that some of the function names in the call stack are missing. In some cases, this partial stack trace may not be enough to find out exactly where the problem is.

A full stack trace can only be produced if you have debugging symbols for your `mariadbd` binary.

Obtaining Debugging Symbols for Your mariadbd executable

Debug information is used by debugging tools to produce a meaningful stack trace. Importantly these packages do not replace any executables or any existing production executables or in any way interfere with the way the production server ran before these packages where installed.

If you are obtaining a backtrace for a coredump, you can move the core dump to a difference server that has the identical mariadb-server and debug info packages, and perform the backtrace there with no loss of information.

Installing Debug Info Packages on Linux

On some Linux distributions, you may be able to install `debuginfo` packages that contain debugging symbols.

Currently, `debuginfo` packages may not allow the server to print a nice stack trace in the error log. They also allow users to extract full stack traces from core dumps. See MDEV-20738 for more information.

Installing Debug Info Packages with yum/dnf

The MariaDB `yum` repository contains debuginfo packages.

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using yum or dnf. Starting with RHEL 8 and Fedora 22, `yum` has been replaced by `dnf`, which is the next major version of `yum`. However, `yum` commands still work on many systems that use `dnf`. For example:

```
sudo yum install MariaDB-server-debuginfo
```

See Installing MariaDB with yum/dnf: Installing Debug Info Packages with YUM for more information.

Installing Debug Info Packages with zypper

The MariaDB `zypper` repository contains debuginfo packages.

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant RPM package from MariaDB's repository using zypper. For example:

```
sudo zypper install MariaDB-server-debuginfo
```

See Installing MariaDB with zypper: Installing Debug Info Packages with ZYpp for more information.

Installing Debug Info Packages from MariaDB's Debian or Ubuntu repository

These are for when you already installed MariaDB from a MariaDB mirror.

For Ubuntu an additional repository step is needed:

```
sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el,s390x]
https://ftp.osuosl.org/pub/mariadb/repo/10.5/ubuntu focal main/debug'
```

Adjust `10.5` to the major version you are debugging and `focal` to the required distribution.

```
apt-get update && apt-get install -y mariadb-server-core-10.5-dbgsym
```

From MariaDB 10.9 the version isn't included in the package name and `mariadb-server-core-dbgsym` can be used as the package name.

Installing Debug Info Packages packaged by Ubuntu or Debian

If you used the MariaDB versions provided by Debian or Ubuntu see the following links.

For Debian see AutomaticDebugPackages

For Ubuntu see Debug%20Symbol%20Packages

Installing Debugging Symbols on Windows

Debugging symbols are available to install on Windows.

Installing Debugging Symbols with the MSI Installer on Windows

Debugging symbols can be installed with the MSI installer. Debugging symbols are not installed by default. You must perform a custom installation and explicitly choose to install debugging symbols.

The MSI installer can be downloaded from the MariaDB downloads page.

Installing Debugging Symbols with the ZIP Package on Windows

MariaDB also provides a ZIP package that contains debugging symbols on Windows.

The ZIP package that contains debugging symbol can be downloaded from the MariaDB downloads page.

Containers with Debug Symbols

Prebuilt Debug Containers

These are currently only per major release version and are generated out of CI. They are always the latest version in the main branch on GitHub.

There are available on at mariadb-debug?tab=tags.

Use the container name `quay.io/mariadb-foundation/mariadb-debug:VERSION`.

Where `VERSION` corresponds to the major version you wish to test with.

Creating a Container with Debugging Symbols

Build using a Containerfile/Dockerfile:

```
ARG VERSION=10.11
FROM mariadb:$VERSION

RUN apt-get update \
        && apt-get install -y gdb mariadb-server-core-dbgsym=$(dpkg -s mariadb-server-core |  awk
'/Version/{print $NF}') ; \
        rm -rf /var/lib/apt/lists/*
```

Build with:

```
buildah bud --tag mariadb_debug --build-arg VERSION=10.11.6 -f Containerfile .
```

Then you will have a `mariadb_debug` container.

Example use:

```
podman run --entrypoint gdb --user mysql --volume mariadb_data:/var/lib/mysql mariadb_debug -ex r --args
/usr/sbin/mariadbd
```

Enabling Core Dumps

To enable core dumps, see Enabling Core Dumps for details.

Where is the Core File on Linux?

At the bottom of the error log there will be some text about the core location including:

```
Writing a core file...
Working directory at /var/lib/mariadb
Resource Limits:
Limit                     Soft Limit           Hard Limit           Units
...
Max core file size        unlimited            unlimited            bytes
...
Core pattern: |/usr/lib/systemd/systemd-coredump %P %u %g %s %t %c %h

Kernel version: Linux version 6.0.0-0.rc2.19.fc38.x86_64 (mockbuild@bkernel01.iad2.fedoraproject.org)
(gcc (GCC) 12.2.1 20220819 (Red Hat 12.2.1-1), GNU ld version 2.39-2.fc38) #1 SMP PREEMPT_DYNAMIC Mon Aug
22 12:52:40 UTC 2022
```

If the was a core limit in the resource limits there may be limited or no core file information.

If the core pattern begins with a **|**, then the following is the executable that handled the core file during the crash. The following show a few techniques to access the core depending on the pattern. If another program is used, look at its manual page to see how to get access to the core file.

If a plain core filename is in the “Core pattern” there’s a good chance it will be in the Working directory location. It might have a `.{process number}` suffix on the filename.

Extracting a core file from a container

If you are running MariaDB in a container, the locations where the core dump can be generated are limited. Looking at the container log, this will likely be where the error log information is. The “Core pattern” of a Linux system is currently a global fixed value. The consequence is if this core pattern refers to a program, that program isn’t likely to be in the container and won’t be executed on the crash.

The system wide crash handler can be changed with `sysctl kernel.core_pattern=core` to set this back to a file based crash. With this, the crash should occur in the working directory, normally the `/var/lib/mysql` data directory of the container volume.

Extracting a core file from systemd-coredump

For `systemd-coredump`, there is a program `coredumpctl` to manage access.

```
coredumpctl list
TIME                            PID   UID   GID SIG     COREFILE EXE
>
Fri 2022-09-09 14:16:37 AEST 213571  1000  1000 SIGSEGV present  /usr/sbin/mariadbd
```

To access the program using `gdb`, `coredumpctl debug` (defaults to last crash), will load the core dump in gdb. The instructions in the next section for extracting information.

See also: extracting core dumps with systemd-coredump.

Extract a core file from abrt

A core pattern of `|/usr/libexec/abrt-hook-ccpp` indicates `abrt` system is used.

abrt-cli is a command line user interface for access the core file.

Extract a core file from apport

A core pattern of `[|/usr/share/apport/apport` indicates `apport`.

For more information see Apport Project Wiki.

apport-retrace allows you to “Examine Locally” and run a `gdb` session. One you have gdb started instructions in the next section can be used for extracting information.

Analyzing a Core File with `gdb` on Linux

To analyze the core file on Linux, you can use gdb.

For example, to open a core file with gdb, you could execute the following:

```
sudo gdb /usr/sbin/mariadbd  /var/lib/mysql/core.932
```

Be sure to replace `/usr/sbin/mariadbd` with the path to your `mariadbd` binary (might be `mysqld` on MariaDB 10.4 and earlier) and to also replace `/var/lib/mysql/core.932` with the path to your core file.

Once gdb has opened the core file, if you want to log all output to a file, then you could execute the following commands:

```
set logging file /tmp/gdb_output.log
set logging on
```

If you do not execute `set logging file`, then the `set logging on` command creates a `gdb.txt` in your current working directory. Redirecting the output to a file is useful, because it can make it easier to analyze. It also makes it easier to send the information to a MariaDB developer, if that becomes necessary.

Do any commands that you would like to do. For example, you could get the backtraces.

Once you are done, you can exit gdb by executing the quit command.

Getting Backtraces with `gdb` on Linux

On Linux, once you have debugging symbols for your `mariadbd` binary, you can use the gdb utility to get backtraces, which are what `gdb` calls stack traces. Backtraces can be obtained from a core file or from a running `mariadbd` process.

Full backtraces are preferred and will contain function arguments, which can contain useful information such as query strings, so it can make the information easier to analyze.

To get a **full** backtrace of the main thread, then you could execute the following:

```
bt -frame-arguments all full
```

If you want to get a **full** backtrace of **all** threads, then you could execute the following:

```
thread apply all bt -frame-arguments all full
```

If you want to get a full backtrace to a file to report a bug, the recommended way is to use gdb:

```
set logging on
set pagination off
set print frame-arguments all
thread apply all bt full
set logging off
```

This will write the full backtrace into the file `gdb.txt`.

Getting Full Backtraces For All Threads From a Core File

Sometimes it can be helpful to get **full** backtraces for all threads. The full backtraces will contain function arguments, which can contain useful information such as query strings, so it can make the information easier to analyze.

To get **full** backtraces for all threads from a `mariadbd` core file, execute a command like the following:

```
sudo gdb --batch --eval-command="set print frame-arguments all" --eval-command="thread apply all bt full"
/usr/sbin/mariadbd /var/lib/mysql/core.932  > mariadbd_full_bt_all_threads.txt
```

Be sure to replace `/usr/sbin/mariadbd` with the path to your `mariadbd` binary and to also replace `/var/lib/mysql/core.932` with the path to your core dump.

The backtraces will be output to the file `mariadbd_full_bt_all_threads.txt`.

Getting Full Backtraces For All Threads From a Running `mariadbd` Process

Sometimes it can be helpful to get **full** backtraces for all threads. The full backtraces will contain function arguments, which can contain useful information such as query strings, so it can make the information easier to analyze.

To get **full** backtraces for all threads from a running `mariadbd` process, execute a command like the following:

```
sudo gdb --batch --eval-command="set print frame-arguments all"  --eval-command="thread apply all bt
full" /usr/sbin/mariadbd $(pgrep -xn mariadbd)  > mariadbd_full_bt_all_threads.txt
```

Be sure to replace `/usr/sbin/mariadbd` with the path to your `mariadbd` binary.

The backtraces will be output to the file `mariadbd_full_bt_all_threads.txt`.

Sometimes very busy systems are too busy to batch obtain the backtrace. If this is the case, `gcore $(pidof mariadbd)` can save the core and then obtain the backtrace out of the dumped core.

Getting a Full Backtrace out of a Container

If the crash or assertion is repeatable it could be easiest to run `mariadbd` under #gdb `.`

The container image name here can be a prebuilt one from `quay.io/mariadb-foundation/mariadb-debug` or an explicit version built yourself as above.

```
docker run -v datavolume:/var/lib/mysql/ --rm --user mysql  quay.io/mariadb-foundation/mariadb-
debug:10.11 gdb -ex r  -ex 'thread apply all bt -frame-arguments all full'  --args mariadbd
```

In docker-compose.yml form this looks like:

```
services:
  mariadb:
    image: quay.io/mariadb-foundation/mariadb-debug:10.11
    volumes:
      - mariadbdata:/var/lib/mysql
    environment:
      - MARIADB_ROOT_PASSWORD=bob
    command: gdb -ex r  -ex 'thread apply all bt -frame-arguments all full'  --args mariadbd
    user: mysql
volumes:
  mariadbdata: {}
```

Note, the initialization of data is assumed. Omit `command` and `user` if it isn't.

If you wish to attach to and existing process in a container, the container needs to be started with the SYS_PTRACE capability. The sysctl `kernel.yama.ptrace_scope` that allows this should also be set to 0.

```
$ podman run -v data:/var/lib/mysql/ --cap-add SYS_PTRACE --name mtest -d quay.io/mariadb-
foundation/mariadb-debug:11.2
$ podman exec --user mysql mtest gdb -p 1 -ex  'thread apply all bt -frame-arguments all full'
```

Note: `podman` has the same arguments and behaviour as `docker` if you'd rather use that.

or in compose:

```
cap_add:
    - SYS_PTRACE
```

The container process is always pid one, and here we use `c` then `thread apply all bt -frame-arguments all full` as the pre-loaded `gdb` commands. When a particular signal like assert or SEGV is triggered, the backtrace will be displayed.

Letting a Container coredump

First, the `sysctl kernel.core_pattern` needs to be `core`. If it starts with a pipe character it try to execute this within the container. This is a kernel wide setting and cannot be applied to a specific container. The bottom of a crash will show you want it is set to.

With this set, just running a container with debug info is sufficient. The core should be dumped on the datadir volume. With the `core` dump here, analysis can occur like:

```
podman run --rm --user mysql --volume data:/var/lib/mysql -i mariadb_debug gdb --batch --eval-
command="set print frame-arguments all" --eval-command="thread apply all bt full" /usr/sbin/mariadbd
/var/lib/mysql/core | tee mariadbd_full_bt_all_threads.txt
```

Running a Copy of the Database Directory

If you are concerned with debuggers running on your production database you can also copy the database to another location.

This is useful when you know which statement crashed the server.

Just start mariadbd with the options `--datadir=/copy-of-original-data-directory --core-file --stack-trace --socket=/tmp/mariadbd-alone.sock --skip-networking`

Disabling Stack Traces in the Error Log

In order to disable stack traces in the error log, you can configure the skip_stack_trace option either on the command-line or in a relevant server option group in an option file. For example:

```
[mariadb]
...
skip_stack_trace
```

Reporting the Problem

If you encounter some problem in MariaDB, then MariaDB's developers would appreciate if you would report a bug at the MariaDB JIRA bug tracker. Please include the following information:

- Your full stack trace.
- Your error log.
- Your option files.

- How to reproduce the problem.
- SHOW ENGINE INNODB STATUS
- SHOW CREATE TABLE {table (for each table in query) and EXPLAIN {query} if a query related crash.

A MariaDB FTP server is available for large and/or sensitive information. Please upload in `.tar.gz` or `.zip` archive.

For very difficult or critical errors, you should consider uploading the following information to the MariaDB FTP server the following:

- Your build of `mariadbd` (if you compiled it), otherwise version information on the mariadb-server package.
- Your core file.
- Your contact information.
- The associated JIRA issue identifier for the bug, if you reported a bug.

This information will allow MariaDB developers to analyze it and try to create a fix.

This page is licensed: CC BY-SA / Gnu FDL

Using procmon to Trace mysqld.exe Filesystem Access

> **Note:**
>
> This page is focused on lower-level development: It contains technical “how-to” documentation for engineers who are actually modifying the MariaDB source code (e.g., compiling for debug, using GDB, and memory leak detection in the code).

This article provides a walkthrough on using the Process Monitor on Windows, tracing file system access by mysqld.exe during the “install plugin” call.

Download

Process Monitor is an advanced monitoring tool for Windows that shows real-time file system, registry and process/thread activity. It is a part of sysinternals suite developed by Mark Russinovich and Bryce Cogswell. Process Monitor can be directly downloaded from ProcessMonitor.zip . More description can be found at https://learn.microsoft.com/en-us/sysinternals/downloads/procmon

Installation

There is no installation necessary; the single executable can be used after unpacking. I suggest putting procmon into some directory in the PATH environment variable.

Example of taking a mysqld.exe trace

The purpose of the following exercise is to learn how to use procmon to trace mysqld.exe calls to the filesystem.

We assume that mysqld.exe is already started.

1. Start procmon.exe . Dialog will pop up that offers to set filter. Use this dialog to set filter to “Process name” “is” “mysqld.exe”, as shown in the screenshot below.

Click on “Add” button to mysqld.exe to include it in the filter, “Apply” and “OK”.

2. Capture events (Menu File=>Capture Events (Ctrl+E)
3. Start mysql command line client and connect to the server.
   Execute

```
mysql> install plugin blackhole soname 'ha_blackhole.dll';
Query OK, 0 rows affected (0.03 sec)
```

4. Saving the trace

Back to Process Monitor Windows, you should see the filesystem events initiated by the “INSTALL PLUGIN” operation

To save it, choose File/Save.

(Advanced) Seeing stack traces corresponding to events

It is also possible to see stacktraces corresponding to the events. For this to work, symbols support needs to be configured. This needs to be only done once.

1. Install Debugging Tools for Windows (google on how to do that).
2. Switch to Process Monitor’s menu Options => Configure symbols.
3. Add dbghelp.dll from your installation of Debugging Tools into “dbghelp.dll path” input field . On my system it is C:\Program Files\Debugging Tools for Windows (x64)\dbghelp.dll
4. In “symbol path” input field, add `srv*C:\symbols*https://msdl.microsoft.com/download/symbols;<path\to\your\installation\bin>` (substitute last path element with real path to your installation)

This is how it looks on my machine:

Once symbols are configured, you’ll get a stack trace corresponding to a filesystem event by simply doubleclicking on the line corresponding to the event. This is what I see after clicking on the first event of my tracing session (corresponds to opening my.ini file)

It is also possible to save the whole trace with callstacks as text (File/Save, choose XML, include callstack + resolve callstack).

This page is licensed: CC BY-SA / Gnu FDL

Limitations/Differences with a MariaDB Server Compiled for Debugging

> **Note:**
>
> This page is focused on lower-level development: It contains technical “how-to” documentation for engineers who are actually modifying the MariaDB source code (e.g., compiling for debug, using GDB, and memory leak detection in the code).

A MariaDB server configured with `--with-debug=full` has the following differences from a normal MariaDB server:

- You can have maximum of 1000 tables locked at the same time in one statement. (Define `MAX_LOCKS` in mysys/thrlock.c). This is to detect loops in the used lists.
- You can have maximum of 1000 threads locking the same table. (Define `MAX_THREADS` in mysys/thrlock.c). This is to detect loops in the used lists.
- Deadlock detection of mutex will be done at runtime. If wrong mutex handling is found an error will be printed to the error log. (Define `SAFE_MUTEX`)
- Memory overrun/underrun and not freed memory will be reported to the error log (Define `SAFEMALLOC`)
- You can get a trace of what `mysqld` (and most other binaries) is doing by starting it with the `--debug` option. The trace is usually put in `/tmp` or `C:\`

This page is licensed: CC BY-SA / Gnu FDL

Plugin Development

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See Writing Plugins for MariaDB.

A comprehensive guide on the basic structure and necessary components for creating a new plugin from scratch.

See Encryption Plugin Development.

Details the API used to develop encryption plugins for protecting data at rest, including key management and encryption schemes.

See Audit Plugin Development.

Provides instructions for developing plugins that track and log server activity for security and compliance monitoring.

See Authentication Plugin Development.

Details how to implement custom authentication methods, allowing users to connect using external credentials or protocols.

See Information Schema Plugins: SHOW and FLUSH Statements.

Explains how to create plugins that expose internal server data through virtual tables in the Information Schema.

See Replication & Cluster Plugin Development.

Focuses on building plugins that interact with the server's replication stream or manage cluster-level synchronization.

See Password Validation Plugin Development.

Implementation guide specifically for developers creating logic to intercept and validate password changes in the server.

See Storage Engine Development.

A dedicated resource for engineers to learn how to build or modify storage engines, detailing the pluggable API and data handling at the physical level.

This page is licensed: CC BY-SA / Gnu FDL

Writing Plugins for MariaDB

> **Note:**
>
> This page contains background information for MariaDB developers.

About

Generally speaking, writing plugins for MariaDB is very similar to writing plugins for MySQL.

Authentication Plugins

See Pluggable Authentication.

Storage Engine Plugins

Storage engines can extend `CREATE TABLE` syntax with optional index, field, and table attribute clauses. See Extending CREATE TABLE for more information. See also Storage Engine Development.

Information Schema Plugins

Information Schema plugins can have their own FLUSH and SHOW statements. See FLUSH and SHOW for Information Schema plugins.

Encryption Plugins

Encryption plugins in MariaDB are used for the data at rest encryption feature. They are responsible for both key management and for the actual encryption and decryption of data.

Function Plugins

Function plugins add new SQL functions to MariaDB. Unlike the old UDF API, function plugins can do almost anything that a built-function can.

Plugin Declaration Structure

The MariaDB plugin declaration differs from the MySQL plugin declaration in the following ways:

1. it has no useless 'reserved' field (the very last field in the MySQL plugin declaration)
2. it has a 'maturity' declaration
3. it has a field for a text representation of the version field

MariaDB can load plugins that only have the MySQL plugin declaration but both `PLUGIN_MATURITY` and `PLUGIN_AUTH_VERSION` will show up as 'Unknown' in the INFORMATION_SCHEMA.PLUGINS table.

For compiled-in (not dynamically loaded) plugins, the presence of the MariaDB plugin declaration is mandatory.

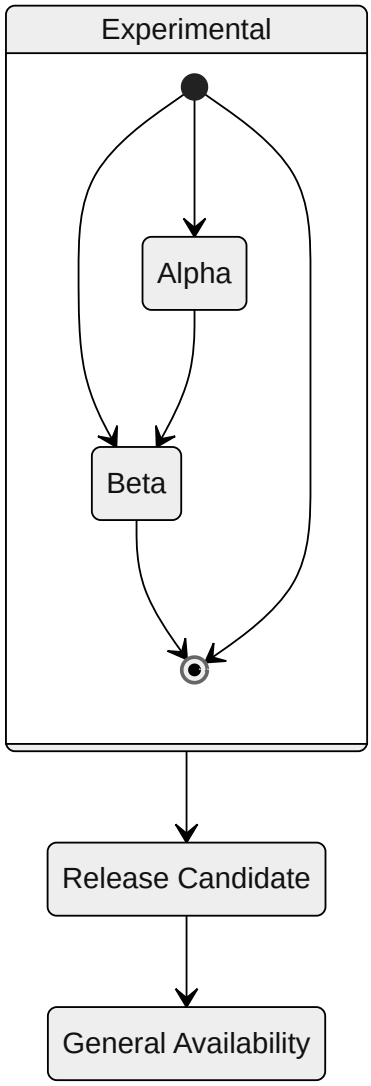
Example Plugin Declaration

The MariaDB plugin declaration looks like this:

```
/* MariaDB plugin declaration */
maria_declare_plugin(example)
{
   MYSQL_STORAGE_ENGINE_PLUGIN, /* the plugin type (see include/mysql/plugin.h) */
   &example_storage_engine_info, /* pointer to type-specific plugin descriptor   */
   "EXAMPLEDB", /* plugin name */
   "John Smith",  /* plugin author */
   "Example of plugin interface", /* the plugin description */
   PLUGIN_LICENSE_GPL, /* the plugin license (see include/mysql/plugin.h) */
   example_init_func,   /* Pointer to plugin initialization function */
   example_deinit_func,  /* Pointer to plugin deinitialization function */
   0x0001 /* Numeric version 0xAABB means AA.BB version */,
   example_status_variables,  /* Status variables */
   example_system_variables,  /* System variables */
   "0.1 example",  /* String version representation */
   MariaDB_PLUGIN_MATURITY_EXPERIMENTAL /* Maturity (see include/mysql/plugin.h)*/
}
maria_declare_plugin_end;
```

Maturity Guidelines For Plugins

Plugin Maturity At a Glance



Plugin Maturity Guidelines

> **Note:**
>
> New features or sufficiently big code changes to a plugin should lead to:
>
> - an increase of the plugin version
> - knocking the maturity level down.

- First a plugin is Alpha
- After at least 1.5 months (or more, in 3 month increments, until the quality is deemed acceptable) it becomes Gamma. Meanwhile it can transition through Beta as the maintainer wants (e,g. it can start directly from beta)
- After at least 3 more months (or more, in 3 month increments — depending on the bug inflow) it becomes Stable

If a plugin must be statically compiled into the server, it is a subject to an additional limitation — its maturity cannot be lower than the server's.

Beta Plugins In GA Releases

If a plugin contribution is added directly at beta maturity to a GA release series, it will not be removed from that release series. However, we reserve the right to exclude a plugin from future release series if it is unmaintained (for example, if it stops receiving bug fixes).

This page is licensed: CC BY-SA / Gnu FDL

Encryption Plugin Development

> **Note:**
>
> This page contains background information for MariaDB developers.

MariaDB's data-at-rest encryption requires the use of a key management and encryption plugin. These plugins are responsible both for the management of encryption keys and for the actual encryption and decryption of data.

MariaDB supports the use of multiple encryption keys. Each encryption key uses a 32-bit integer as a key identifier. If the specific plugin supports key rotation, encryption keys can also be rotated, which creates a new version of the encryption key.

See Data at Rest Encryption and Encryption Key Management for more information.

Encryption Plugin API

The Encryption plugin API was created to allow a plugin to:

- implement key management, provide encryption keys to the server on request and change them according to internal policies.
- implement actual data encryption and decryption with the algorithm defined by the plugin.

This is how the API reflects that:

```
/* Returned from get_latest_key_version() */
#define ENCRYPTION_KEY_VERSION_INVALID (~(unsigned int)0)
#define ENCRYPTION_KEY_NOT_ENCRYPTED (0)

#define ENCRYPTION_KEY_SYSTEM_DATA 1
#define ENCRYPTION_KEY_TEMPORARY_DATA 2

/* Returned from get_key()  */
#define ENCRYPTION_KEY_BUFFER_TOO_SMALL (100)

#define ENCRYPTION_FLAG_DECRYPT 0
#define ENCRYPTION_FLAG_ENCRYPT 1
#define ENCRYPTION_FLAG_NOPAD 2

struct st_mariadb_encryption {
  int interface_version; /**< version plugin uses */

  /******************** KEY MANAGEMENT ***********************************/

  /**
    Function returning latest key version for a given key id.

    @return A version or ENCRYPTION_KEY_VERSION_INVALID to indicate an error.
  */
  unsigned int (*get_latest_key_version)(unsigned int key_id);

  /**
    Function returning a key for a key version

    @param key_id       The requested key id
    @param version      The requested key version
    @param key          The key will be stored there. Can be NULL -
                        in which case no key will be returned
    @param key_length   in: key buffer size
                        out: the actual length of the key

    This method can be used to query the key length - the required
    buffer size - by passing key==NULL.

    If the buffer size is less than the key length the content of the
    key buffer is undefined (the plugin is free to partially fill it with
    the key data or leave it untouched).

    @return 0 on success, or
            ENCRYPTION_KEY_VERSION_INVALID, ENCRYPTION_KEY_BUFFER_TOO_SMALL
            or any other non-zero number for errors
  */
  unsigned int (*get_key)(unsigned int key_id, unsigned int version,
                          unsigned char *key, unsigned int *key_length);

  /******************** ENCRYPTION ***********************************/
  /*
    The caller uses encryption as follows:
      1. Create the encryption context object of the crypt_ctx_size() bytes.
      2. Initialize it with crypt_ctx_init().
      3. Repeat crypt_ctx_update() until there are no more data to encrypt.
      4. Write the remaining output bytes and destroy the context object
         with crypt_ctx_finish().
  */

  /**
    Returns the size of the encryption context object in bytes
  */
  unsigned int (*crypt_ctx_size)(unsigned int key_id, unsigned int key_version);
  /**
    Initializes the encryption context object.
  */
  int (*crypt_ctx_init)(void *ctx, const unsigned char *key, unsigned int klen,
```

```
                        const unsigned char *iv, unsigned int ivlen, int flags,
                        unsigned int key_id, unsigned int key_version);
  /**
    Processes (encrypts or decrypts) a chunk of data

    Writes the output to th dst buffer. note that it might write
    more bytes that were in the input. or less. or none at all.
  */
  int (*crypt_ctx_update)(void *ctx, const unsigned char *src,
                          unsigned int slen, unsigned char *dst,
                          unsigned int *dlen);
  /**
    Writes the remaining output bytes and destroys the encryption context

    crypt_ctx_update might've cached part of the output in the context,
    this method will flush these data out.
  */
  int (*crypt_ctx_finish)(void *ctx, unsigned char *dst, unsigned int *dlen);
  /**
    Returns the length of the encrypted data

    It returns the exact length, given only the source length.
    Which means, this API only supports encryption algorithms where
    the length of the encrypted data only depends on the length of the
    input (a.k.a. compression is not supported).
  */
  unsigned int (*encrypted_length)(unsigned int slen, unsigned int key_id,
                                   unsigned int key_version);
};
```

The first method is used for key rotation. A plugin that doesn't support key rotation — for example, **file_key_management** — can return a fixed version for any valid key id. Note that it still has to return an error for an invalid key id. The version `ENCRYPTION_KEY_NOT_ENCRYPTED` means that the data should not be encrypted.

The second method is used for key management, the server uses it to retrieve the key corresponding to a specific key identifier and a specific key version.

The last five methods deal with encryption. Note that they take the key to use and key identifier and version. This is needed because the server can derive a session-specific, user-specific, or a tablespace-specific key from the original encryption key as returned by `get_key()`, so the `key` argument doesn't have to match the encryption key as the plugin knows it. On the other hand, the encryption algorithm may depend on the key identifier and version (and in the **example_key_management** plugin it does) so the plugin needs to know them to be able to encrypt the data.

Encryption methods are optional — if unset (as in the **debug_key_management** plugin), the server will fall back to AES_CBC.

Current Encryption Plugins

The MariaDB source tree has four encryption plugins. All these plugins are fairly simple and can serve as good examples of the Encryption plugin API.

file_key_management

It reads encryption keys from a plain-text file. It supports two different encryption algorithms. It supports multiple encryption keys. It does not support key rotation.
See File Key Management Plugin for details.

aws_key_management

The AWS Key Management plugin uses the Amazon Web Services (AWS) Key Management Service (KMS) to generate and store AES keys on disk, in encrypted form, using the Customer Master Key (CMK) kept in AWS KMS.

When MariaDB Server starts, the plugin will decrypt the encrypted keys, using the AWS KMS “Decrypt” API function. MariaDB data will then be encrypted and decrypted using the AES key. It supports multiple encryption keys. It supports key rotation.

See AWS Key Management Plugin for details.

example_key_management

Uses random time-based generated keys, ignores key identifiers, supports key versions and key rotation. Uses AES_ECB and AES_CBC as encryption algorithms and changes them automatically together with key versions.

debug_key_management

Key is generated from the version, user manually controls key rotation. Only supports key identifier 1, uses only AES_CBC.

Encryption Service

Encryption is generally needed on the very low level **inside the storage engine**. That is, the storage engine needs to support encryption and have access to the encryption and key management functionality. The usual way for a plugin to access some functionality in the server is via a *service*. In this case the server provides the Encryption Service for storage engines (and other interested plugins) to use. These service functions are directly hooked into encryption plugin methods (described above).

Service functions are declared as follows:

```
unsigned int encryption_key_get_latest_version(unsigned int key_id);
unsigned int encryption_key_get(unsigned int key_id, unsigned int key_version,
                                unsigned char *buffer, unsigned int *length);
unsigned int encryption_ctx_size(unsigned int key_id, unsigned int key_version);
int encryption_ctx_init(void *ctx, const unsigned char *key, unsigned int klen,
                        const unsigned char *iv, unsigned int ivlen, int flags,
                        unsigned int key_id, unsigned int key_version);
int encryption_ctx_update(void *ctx, const unsigned char *src,
                          unsigned int slen, unsigned char *dst,
                          unsigned int *dlen);
int encryption_ctx_finish(void *ctx, unsigned char *dst, unsigned int *dlen);
unsigned int encryption_encrypted_length(unsigned int slen, unsigned int key_id,
                                         unsigned int key_version);
```

There are also convenience helpers to check for a key or key version existence and to encrypt or decrypt a block of data with one function call.

```
unsigned int encryption_key_id_exists(unsigned int id);
unsigned int encryption_key_version_exists(unsigned int id,
                                           unsigned int version);
int encryption_crypt(const unsigned char *src, unsigned int slen,
                     unsigned char *dst, unsigned int *dlen,
                     const unsigned char *key, unsigned int klen,
                     const unsigned char *iv, unsigned int ivlen, int flags,
                     unsigned int key_id, unsigned int key_version);
```

This page is licensed: CC BY-SA / Gnu FDL

Audit Plugin Development

> **Note:**
>
> This page contains background information for MariaDB developers.

This page is licensed: CC BY-SA / Gnu FDL

Authentication Plugin Development

> **Note:**
>
> This page contains background information for MariaDB developers.

Authentication Plugin API

The authentication plugin API is extensively documented in the source code in the following files:

- `mysql/plugin_auth.h` (server part)
- `mysql/client_plugin.h` (client part)
- `mysql/plugin_auth_common.h` (common parts)

The MariaDB source code also contains some authentication plugins that are intended explicitly to be examples for developers. They are located in `plugin/auth_examples`.

The definitions of two example authentication plugins called `two_questions` and `three_attempts` can be seen in `plugin/auth_examples/dialog_examples.c`. These authentication plugins demonstrate how to communicate with the user using the dialog client authentication plugin.

The `two_questions` authentication plugin asks the user for a password and a confirmation ("Are you sure?").

The `three_attempts` authentication plugin gives the user three attempts to enter a correct password.

The password for both of these plugins should be specified in the plain text in the `USING` clause:

```
CREATE USER insecure IDENTIFIED VIA two_questions USING 'notverysecret';
```

Dialog Client Authentication Plugin - Client Library Extension

The dialog client authentication plugin, strictly speaking, is not part of the client-server or authentication plugin API. But it can be loaded into any client application that uses the `libmysqlclient` or MariaDB Connector/C libraries. This authentication plugin provides a way for the application to customize the UI of the dialog function.

In order to use the dialog client authentication plugin to communicate with the user in a customized way, the application will need to implement a function with the following signature:

```
extern "C" char *mysql_authentication_dialog_ask(
  MYSQL *mysql, int type, const char *prompt, char *buf, int buf_len)
```

The function takes the following arguments:

- The connection handle.
- A question "type", which has one of the following values:
  - `1` - Normal question
  - `2` - Password (no echo)
- A prompt.
- A buffer.
- The length of the buffer.

The function returns a pointer to a string of characters, as entered by the user. It may be stored in `buf` or allocated with `malloc()`.

By using this function, a GUI application can open a dialog window, and a network application can send the question over the network, as required. If no `mysql_authentication_dialog_ask` function is provided by the application, the dialog client authentication plugin falls back to fputs() and fgets().

Providing this callback is particularly important on Windows, because Windows GUI applications have no associated console and the default dialog function will not be able to reach the user. An example of Windows GUI client that does it correctly is HeidiSQL.

This page is licensed: CC BY-SA / Gnu FDL

Information Schema Plugins: SHOW and FLUSH Statements

> **Note:**
>
> This page contains background information for MariaDB developers.

Information Schema plugins support SHOW and FLUSH statements.

SHOW

`SHOW` statement support is enabled by default. A plugin only has to specify column names for the `SHOW` statement in the `old_name` member of the field declaration structure. Columns with the `old_name` set to `0` are hidden from the `SHOW` statement. If all columns are hidden, the `SHOW` statement doesn't work for this plugin.

Note that `SHOW`` *statement*` is a user-friendly shortcut – it's easier to type and view. If the Information Schema table contains many columns, the `SHOW` statement is supposed to display only the most important columns, so that the output fits nicely on the 80x25 terminal screen.

Consider an example, the LOCALES plugin:

```
static ST_FIELD_INFO locale_info_locale_fields_info[]=
{
  {"ID", 4, MYSQL_TYPE_LONGLONG, 0, 0, "Id", 0},
  {"NAME", 255, MYSQL_TYPE_STRING, 0, 0, "Name", 0},
  {"DESCRIPTION", 255,  MYSQL_TYPE_STRING, 0, 0, "Description", 0},
  {"MAX_MONTH_NAME_LENGTH", 4, MYSQL_TYPE_LONGLONG, 0, 0, 0, 0},
  {"MAX_DAY_NAME_LENGTH", 4, MYSQL_TYPE_LONGLONG, 0, 0, 0, 0},
  {"DECIMAL_POINT", 2, MYSQL_TYPE_STRING, 0, 0, 0, 0},
  {"THOUSAND_SEP", 2, MYSQL_TYPE_STRING, 0, 0, 0, 0},
  {"ERROR_MESSAGE_LANGUAGE", 64, MYSQL_TYPE_STRING, 0, 0, "Error_Message_Language", 0},
  {0, 0, MYSQL_TYPE_STRING, 0, 0, 0, 0}
};
```

While the INFORMATION_SCHEMA.LOCALES table has 8 columns, the SHOW LOCALES statement only displays 4 of them:

```
MariaDB [test]> SHOW LOCALES;
+-----+-------+-------------------------------------+-------------------------+
| Id  | Name  | Description                         | Error_Message_Language |
+-----+-------+-------------------------------------+-------------------------+
|   0 | en_US | English - United States             | english                |
|   1 | en_GB | English - United Kingdom            | english                |
|   2 | ja_JP | Japanese - Japan                    | japanese               |
|   3 | sv_SE | Swedish - Sweden                    | swedish                |
...
```

FLUSH

To support the `FLUSH` statement, a plugin must declare the `reset_table` callback. For example, in the QUERY_RESPONSE_TIME plugin:

```
static int query_response_time_info_init(void *p)
{
  ST_SCHEMA_TABLE *i_s_query_response_time= (ST_SCHEMA_TABLE *) p;
  i_s_query_response_time->fields_info= query_response_time_fields_info;
  i_s_query_response_time->fill_table= query_response_time_fill;
  i_s_query_response_time->reset_table= query_response_time_flush;
  query_response_time_init();
  return 0;
}
```

This page is licensed: CC BY-SA / Gnu FDL

Replication & Cluster Plugin Development

> **Note:**
>
> This page contains background information for MariaDB developers.

This page is licensed: CC BY-SA / Gnu FDL

Password Validation Plugin Development

> **Note:**
>
> This page contains background information for MariaDB developers.

Plugin API

Password validation plugin API is very simple. A plugin must implement only one method — `validate_password()`. This method takes two arguments — user name and the plain-text password. It returns `0` when the password has passed the validation; otherwise, `1`.

See also `mysql/plugin_password_validation.h` and password validation plugins in `plugin/simple_password_check/` and `plugins/cracklib_password_check/`.

This page is licensed: CC BY-SA / Gnu FDL

Storage Engine Development

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See Engine-defined New Table/Field/Index Attributes.

See Table Discovery.

This page is licensed: CC BY-SA / Gnu FDL

Engine-defined New Table/Field/Index Attributes

> **Note:**
>
> This page contains background information for MariaDB developers.

In MariaDB, a storage engine can allow the user to specify additional attributes per index, field, or table. The engine needs to declare what attributes it introduces.

API

There are three new members in the `handlerton` structure, they can be set in the engine's initialization function as follows:

```
example_hton->table_options= example_table_option_array;
example_hton->field_options= example_field_option_array;
example_hton->index_options= example_index_option_array;
```

The arrays are declared statically, as in the following example:

```
static MYSQL_THDVAR_ULONG(varopt_default, PLUGIN_VAR_RQCMDARG,
  "default value of the VAROPT table option", NULL, NULL, 5, 0, 100, 0);

struct ha_table_option_struct
{
  char *strparam;
  ulonglong ullparam;
  uint enumparam;
  bool boolparam;
  ulonglong varparam;
};

ha_create_table_option example_table_option_list[]=
{
  HA_TOPTION_NUMBER("NUMBER", ullparam, UINT_MAX32, 0, UINT_MAX32, 10),
  HA_TOPTION_STRING("STR", strparam),
  HA_TOPTION_ENUM("ONE_OR_TWO", enumparam, "one,two", 0),
  HA_TOPTION_BOOL("YESNO", boolparam, 1),
  HA_TOPTION_SYSVAR("VAROPT", varopt, varparam),
  HA_TOPTION_END
};
```

The engine declares a structure `ha_table_option_struct` that will hold values of these new attributes.

And it describes these attributes to MySQL by creating an array of `HA_TOPTION_*` macros. Note a detail: these macros expect a structure called `ha_table_option_struct`, if the structure is called differently, a `#define` are needed.

There are five supported kinds of attributes:

macro name	attribute value type	corresponding C type	additional parameters of a macro
macro name	attribute value type	corresponding C type	additional parameters of a macro
HA_TOPTION_NUMBER	an integer number	unsigned long long	a default value, minimal allowed value, maximal allowed value, a factor, that any allowed should be a multiple of.
HA_TOPTION_STRING	a string	char *	none. The default value is a null pointer.
HA_TOPTION_ENUM	one value from a list of allowed values	unsigned int	a string with a comma-separated list of allowed values, and a default value as a number, starting from 0.
HA_TOPTION_BOOL	a boolean	bool	a default value
HA_TOPTION_SYSVAR	defined by the system variable	defined by the system variable	system variable name

Do not use `enum` for your `HA_TOPTION_ENUM` C structure members, the size of the `enum` depends on the compiler, and even on the compilation options, and the plugin API uses only types with known storage sizes.

In all macros the first two parameters are name of the attribute as should be used in SQL in the `CREATE TABLE` statement, and the name of the corresponding member of the `ha_table_option_struct` structure.

The `HA_TOPTION_SYSVAR` stands aside a bit. It does not specify the attribute type or the default value, instead it binds the attribute to a system variable. The attribute type and the range of allowed values are the same as of the corresponding system variable. The attribute **default value** are the **current value** of its system variable. And unlike other attribute types that are only stored in the `.frm` file if explicitly set in the `CREATE TABLE` statement, the `HA_TOPTION_SYSVAR` attributes are always stored. If the system variable value is changed, it will not affect existing tables. Note that for this very reason, if a table was created in the old version of a storage engine, and a new version has introduced a `HA_TOPTION_SYSVAR` attribute, the attribute value in the old tables are the **default** value of the system variable, not its **current** value.

The array ends with a `HA_TOPTION_END` macro.

Field and index (key) attributes are declared similarly using `HA_FOPTION_*` and `HA_IOPTION_*` macros.

> **CURRENT**
>
> When in a `CREATE TABLE` statement, the `::create()` handler method is called, the table attributes are available in the `option_struct` member of the handler, field attributes - in the `option_struct` member of the individual fields (objects of the `Field` class), index attributes - in the `option_struct` member of the individual keys (objects of the `KEY` class).
>
> **< 12.3**
>
> When in a `CREATE TABLE` statement, the `::create()` handler method is called, the table attributes are available in the `table_arg->s->option_struct`, field attributes - in the `option_struct` member of the individual fields (objects of the `Field` class), index attributes - in the `option_struct` member of the individual keys (objects of the `KEY` class).

Additionally, they are available in most other handler methods: the attributes are stored in the `.frm` file and on every open MySQL makes them available to the engine by filling the corresponding `option_struct` members of the table, fields, and keys.

The `ALTER TABLE` needs a special support from the engine. MySQL compares old and new table definitions to decide whether it needs to rebuild the table or not. As the semantics of the engine declared attributes is unknown, MySQL cannot make this decision by analyzing attribute values - this is delegated to the engine. The `HA_CREATE_INFO` structure has three new members:

```
ha_table_option_struct *option_struct;          ///< structure with parsed table options
ha_field_option_struct **fields_option_struct;  ///< array of field option structures
ha_index_option_struct **indexes_option_struct; ///< array of index option structures
```

The engine (in the `::check_if_incompatible_data()` method) is responsible for comparing new values of the attributes from the `HA_CREATE_INFO` structure with the old values from the table and returning `COMPATIBLE_DATA_NO` if they were changed in such a way that requires the table to be rebuild.

The example of declaring the attributes and comparing the values for the `ALTER TABLE` can be found in the EXAMPLE engine.

SQL

The engine declared attributes can be specified per field, index, or table in the `CREATE TABLE` or `ALTER TABLE`. The syntax is the conventional:

```
CREATE TABLE ... (
  field ... [attribute=value [attribute=value ...]],
  ...
  index ... [attribute=value [attribute=value ...]],
  ...
) ...  [attribute=value [attribute=value ...]]
```

All values must be specified as literals, not expressions. The value of a boolean option may be specified as one of YES, NO, ON, OFF, TRUE, FALSE, 1, or 0. A string value may be specified either quoted or not, as an identifier (if it is a valid identifier, of course — for example `ON` is a reserved keyword and must always be quoted, when used as a value). Additionally, when an enum option expects values of YES or NO, all other boolean aliases will automatically be recognized as well.

When an attribute is set, it is stored with the table definition and shown in the `SHOW CREATE TABLE;`. To remove an attribute from a table definition use `ALTER TABLE` to set its value to a `DEFAULT`.

The values of unknown attributes or attributes with the illegal values cause an error by default. But with ALTER TABLE one can change the storage engine and some previously valid attributes may become unknown — to the new engine. They are not removed automatically, though, because the table might be altered back to the first engine, and these attributes are valid again. Still SHOW CREATE TABLE will comment these unknown attributes out in the output, otherwise they would make a generated CREATE TABLE statement invalid.

With the `IGNORE_BAD_TABLE_OPTIONS` sql mode this behavior changes. Unknown attributes do not cause an error, they only result in a warning. And SHOW CREATE TABLE will not comment them out. This mode is implicitly enabled in the replication slave thread.

See Also

- Writing Plugins for MariaDB
- Storage Engines
- Storage Engine Development

This page is licensed: CC BY-SA / Gnu FDL

Table Discovery

> **Note:**
>
> This page contains background information for MariaDB developers.

In MariaDB, it is not always necessary to run an explicit `CREATE TABLE` statement for a table to appear. Sometimes a table may already exist in the storage engine, but the server does not know about it, because there is no `.frm` file for this table. This can happen for various reasons; for example, for a cluster engine the table might have been created in the cluster by another MariaDB server node. Or for the engine that supports table shipping a table file might have been simply copied into the MariaDB data directory. But no matter what the reason is, there is a mechanism for an engine to tell the server that the table exists. This mechanism is called **table discovery** and if an engine wants the server to discover its tables, the engine should support the table discovery API.

There are two different kinds of table discovery — a fully automatic discovery and a user-assisted one. In the former, the engine can automatically discover the table whenever an SQL statement needs it. In MariaDB, the Archive and Sequence engines support this kind of discovery. For example, one can copy a `t1.ARZ` file into the database directory and immediately start using it — the corresponding `.frm` file are created automatically. Or one can select from say, the `seq_1_to_10` table without any explicit `CREATE TABLE` statement.

In the latter, user-assisted, discovery the engine does not have enough information to discover the table all on its own. But it can discover the table structure if the user provides certain additional information. In this case, an explicit

`CREATE TABLE` statement is still necessary, but it should contain no table structure — only the table name and the table attributes. In MariaDB, the FederatedX storage engine supports this. When creating a table, one only needs to specify the `CONNECTION` attribute and the table structure — fields and indexes — are provided automatically by the engine.

Automatic Discovery

As far as automatic table discovery is concerned, the tables, from the server point of view, may appear, disappear, or change structure anytime. Thus the server needs to be able to ask whether a given table exists and what its structure is. It needs to be notified when a table structure changes outside of the server. And it needs to be able to get a list of all (unknown to the server) tables, for statements like `SHOW TABLES`. The server does all that by invoking specific methods of the `handlerton`:

```
const char **tablefile_extensions;
int (*discover_table_names)(handlerton *hton, LEX_STRING *db, MY_DIR *dir,
                            discovered_list *result);
int (*discover_table_existence)(handlerton *hton, const char *db,
                                const char *table_name);
int (*discover_table)(handlerton *hton, THD* thd, TABLE_SHARE *share);
```

handlerton::tablefile_extensions

Engines that store tables in separate files (one table might occupy many files with different extensions, but having the same base file name) should store the list of possible extensions in the **tablefile_extensions** member of the `handlerton` (earlier this list was returned by the `handler::bas_ext()` method). This will significantly simplify the discovery implementation for these engines, as you will see below.

handlerton::discover_table_names()

When a user asks for a list of tables in a specific database — for example, by using `SHOW TABLES` or by selecting from `INFORMATION_SCHEMA.TABLES` — the server invokes **discover_table_names()** method of the `handlerton`. For convenience this method, besides the database name in question, gets the list of all files in this database directory, so that the engine can look for table files without doing any filesystem i/o. All discovered tables should be added to the `result` collector object. It is defined as

```
class discovered_list
{
  public:
  bool add_table(const char *tname, size_t tlen);
  bool add_file(const char *fname);
};
```

and the engine should call `result->add_table()` or `result->add_file()` for every discovered table (use `add_file()` if the name to add is in the MariaDB file name encoding, and `add_table()` if it's a true table name, as shown in `SHOW TABLES`).

If the engine is file-based, that is, it has non-empty list in the `tablefile_extensions`, this method is optional. For any file-based engine that does not implement `discover_table_names()`, MariaDB will automatically discover the list of all tables of this engine, by looking for files with the extension `tablefile_extensions[0]`.

handlerton::discover_table_existence()

In some rare cases MariaDB needs to know whether a given table exists, but does not particularly care about this table structure (for example, when executing a `DROP TABLE` statement). In these cases, the server uses the **discover_table_existence()** method to find out whether a table with the given name exists in the engine.

This method is optional. For the engine that does not implement it, MariaDB will look for files with the `tablefile_extensions[0]`, if possible. But if the engine is not file-based, MariaDB will use the `discover_table()` method to perform a full table discovery. While this will allow determining correctly whether a table exists, a full discovery is usually slower than the simple existence check. In other words, engines that are not file-based might want to support `discover_table_existence()` method as a useful optimization.

handlerton::discover_table()

This is the main method of table discovery, the heart of it. The server invokes it when it wants to use the table. The **discover_table()** method gets the `TABLE_SHARE` structure, which is not completely initialized — only the table and the database name (and a path to the table file) are filled in. It should initialize this `TABLE_SHARE` with the desired table structure.

MariaDB provides convenient and easy to use helpers that allow the engine to initialize the `TABLE_SHARE` with minimal efforts. They are the `TABLE_SHARE` methods `init_from_binary_frm_image()` and `init_from_sql_statement_string()`.

TABLE_SHARE::init_from_binary_frm_image()

This method is used by engines that use “frm shipping” — such as Archive or NDB Cluster in MySQL. An frm shipping engine reads the frm file for a given table, exactly as it was generated by the server, and stores it internally. Later it can discover the table structure by using this very frm image. In this sense, a separate frm file in the database directory becomes redundant, because a copy of it is stored in the engine.

TABLE_SHARE::init_from_sql_statement_string()

This method allows initializing the `TABLE_SHARE` using a conventional SQL `CREATE TABLE` syntax.

TABLE_SHARE::read_frm_image()

Engines that use frm shipping need to get the frm image corresponding to a particular table (typically in the `handler::create()` method). They do it via the **read_frm_image()** method. It returns an allocated buffer with the binary frm image, that the engine can use the way it needs.

TABLE_SHARE::free_frm_image()

The frm image that was returned by `read_frm_image()` must be freed with the **free_frm_image()**.

HA_ERR_TABLE_DEF_CHANGED

One of the consequences of automatic discovery is that the table definition might change when the server doesn’t expect it to. Between two `SELECT` queries, for example. If this happens, if the engine detects that the server is using an outdated version of the table definition, it should return a **HA_ERR_TABLE_DEF_CHANGED** handler error. Depending on when in the query processing this error has happened, MariaDB will either re-discover the table and execute the query with the correct table structure, or abort the query and return an error message to the user.

TABLE_SHARE::tabledef_version

The previous paragraph doesn’t cover one important question — how can the engine know that the server uses an outdated table definition? The answer is — by checking the **tabledef_version**, the table definition version. Every table gets a unique `tabledef_version` value. Normally it is generated automatically when a table is created. When a table is discovered the engine can force it to have a specific `tabledef_version` value (simply by setting it in the `TABLE_SHARE` before calling the `init_from_binary_frm_image()` or `init_from_sql_statement_string()` methods).

Now the engine can compare the table definition version that the server is using (from any handler method it can be accessed as `this->table->s->tabledef_version`) with the version of the actual table definition. If they differ — it is `HA_ERR_TABLE_DEF_CHANGED`.

Assisted discovery

Assisted discovery is a lot simpler from the server point of view, a lot more controlled. The table cannot appear or disappear at will, one still needs explicit DDL statements to manipulate it. There is only one new handlerton method that the server uses to discover the table structure when a user has issued an explicit `CREATE TABLE` statement without declaring any columns or indexes.

```
int (*discover_table_structure)(handlerton *hton, THD* thd,
                                TABLE_SHARE *share, HA_CREATE_INFO *info);
```

The assisted discovery API is pretty much independent from the automatic discovery API. An engine can implement either of them or both (or none); there is no requirement to support automatic discovery if only assisted discovery is needed.

handlerton::discover_table_structure()

Much like the `discover_table()` method, the **discover_table_structure()** handlerton method gets a partially initialized `TABLE_SHARE` with the table name, database name, and a path to table files filled in, but without a table structure. Unlike `discover_table()`, here the `TABLE_SHARE` has all the engine-defined table attributes in the `TABLE_SHARE::option_struct` structure. Based on the values of these attributes the `discover_table_structure()` method should initialize the `TABLE_SHARE` with the desired set of fields and keys. It can use `TABLE_SHARE` helper methods `init_from_binary_frm_image()` and `init_from_sql_statement_string()` for that.

The role of `.frm` files

Before table discovery was introduced, MariaDB used `.frm` files to store the table definition. But now the engine can store the table definition (if the engine supports automatic discovery, of course), and `.frm` files become redundant. Still, the server *can* use `.frm` files for such an engine — but they are no longer the only source of the table definition. Now `.frm` files are merely a *cache* of the table definition, while the original authoritative table definition is stored in the engine. Like any cache, its purpose is to reduce discovery attempts for a table. The engine decides whether it makes sense to cache table definition in the `.frm` file or not (see the second argument for the `TABLE_SHARE::init_from_binary_frm_image()`). For example, the Archive engine uses `.frm` cache, while the Sequence engine does not. In other words, MariaDB creates `.frm` files for Archive tables, but not for Sequence tables.

The cache is completely transparent for a user; MariaDB makes sure that it always stores the actual table definition and invalidates the `.frm` file automatically when it becomes out of date. This can happen, for example, if a user copies a new Archive table into the datadir and forgets to delete the `.frm` file of the old table with the same name.

This page is licensed: CC BY-SA / Gnu FDL

Server Development

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See MariaDB Server Roadmap.

Provides a high-level view of upcoming features, version release schedules, and the strategic direction for MariaDB Server development.

See Working with GitHub.

Outlines the workflow for contributing to MariaDB via GitHub, including branch management, pull requests, and coding standards for developers.

See Debugging & Troubleshooting.

Offers deep-dive technical guides for diagnosing server issues, including trace file generation, debugger usage, and analyzing core dumps or memory usage.

See Quality Assurance.

Details the processes and methodologies used to ensure code quality, focusing on testing frameworks and bug verification during development.

See MariaDB Development Tools.

A guide to the essential software and utilities recommended for building, testing, and optimizing MariaDB Server code.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Server Roadmap

> **Note:**
>
> This page contains background information for MariaDB developers.

This page talks in general about the MariaDB Server roadmap, and how it is formed.The roadmap it describes is located in our JIRA issue tracker: **jira.mariadb.org**

MariaDB was designed as a drop-in replacement of MySQL with more features, new storage engines, fewer bugs, and better performance. We aim to make upgrading from MySQL to MariaDB extremely easy.

The roadmap for MariaDB Server is formed by its engineers and product managers, with input from the community.

MariaDB Server developers work with several storage engine vendors and developers to get the best storage engines into MariaDB Server.

The collected information is then turned into specific tasks which are then prioritized and added to the roadmap in the JIRA issue tracker at **jira.mariadb.org**

Goals and Quality Standards

The primary goal of MariaDB Server is for it to be a practical database developed in the direction users and developers want it to be developed. Each feature should ideally be developed by or for users who want to test and put the feature into production ASAP — this helps ensure we don't implement features no one wants or needs.

We are also putting a lot of effort into speeding up MariaDB Server, and to keep it stable and easy to use!

The MariaDB Server source tree is maintained by MariaDB the company and its developers. They are the primary contributors to the MariaDB Server project and the ones who are ultimately responsible for the quality of the code.

MariaDB Server 5.1, MariaDB Server 5.2, and MariaDB Server 5.3 were built off of MySQL 5.1.

MariaDB Server 5.5 was a combination of MariaDB Server 5.3 and MySQL 5.5.

MariaDB Server 10.0, and later build off of the previous MariaDB Server releases with backported features from MySQL and entirely new features not found anywhere else.

Short descriptions of the various MariaDB Server releases and their main new features can be found on the MariaDB Server Releases page.

Plans

MariaDB 12.3 is an upcoming major development release.

JIRA

We manage our development plans in JIRA, so the definitive list will be there. This search shows what we **currently** plan for 12.3. It shows all tasks with the **Fix-Version** being 12.3. Not all these tasks will really end up in the release but tasks with the “red” priorities have a much higher chance of being done in time. Practically, you can think of these tasks as “features that **will** be in 12.3”. Tasks with the “green” priorities probably won’t be in. Think of them as “bonus features that would be **nice to have** in 12.3”.

Contributing

If you want to be part of developing any of these features, see Contributing to the MariaDB Project. You can also add new features to this list or to JIRA.

See Also

- Current tasks for 12.3
- 12.3 Features/fixes by vote

This page is licensed: CC BY-SA / Gnu FDL

Working with GitHub

> **Note:**
>
> This page contains background information for MariaDB developers.

Benefits of a Pull Request

A pull request created on GitHub provides visibility to the entire userbase on the changes that are occurring in MariaDB Server.

The pull request created automatically triggers the required Buildbot builds associated with it.

The created pull request is associated with a MariaDB Server branch where it is to be merged. There have been cases where a passing protected branch build has been incorrectly pushed to a former branch by mistake.

A pull request on GitHub can be reviewed, and those reviewing it might have valuable insight. By putting all the reviews in one location, reviewers can learn off each other and not repeat works already review. The reviews are there for future reference if additional though is requires as to why a particular change was made.

A pull request has a clear indicator of the reviews conducted, by who, and who you are expecting further reviews from.

When a developer with commit access, either the author or a reviewer has completed and passed a review, they can mark the pull request as auto-merge, enabling the pull request to be merge once the [[branch-protection-using-buildbot|protected branch builds] have passed.

Mechanism of Creating a Pull Request

A pull request can be created on the MariaDB GitHub web page, server, or any fork created using GitHub.

Alternately GitHub has an open source [|command line client] that has RPM and Deb packages, and a static compiled executable. This is actively maintained.

Creating a Branch

So to make a GitHub pull request you already have some code/changes that you have committed locally. Ideally, this will be on its own branch, rebased onto the MariaDB server branch where it is intended to merge.

A pull request starts from a public branch. You can either create a pull request from a public fork of the MariaDB Server, or if you have access, a branch on the MariaDB Server. Both will achieve the same outcome.

If you are creating on the MariaDB server, a branch name of the MDEV associated with the issue is recommended. If there are targeted branch names that need specific fixes, then prepending the target branch name to the newly created branch is recommended.

Previous documentation suggested a bb-*prefix. This will trigger duplicate runs through Buildbot so your assistance in saving it some work would be appreciated.

This gives other server developers a reasonable visibility of to why the branch is there, a way to check if it can be deleted, and who to ask about in a single branch name.

Creating a Pull Request

If you push with a command line client over a ssh transport remote, the output from GitHub will provide a link to create the pull request. This can be opened in your browser to complete the process.

To use the *gh cli*, use [[gh_pr_create|`gh pr create`] to interactively create it, or use a variety of the command line options to increase the automation.

Common options you may wish to use:

- --base BRANCH - the branch to commit to
- --assignee GITHUB_HANDLE - the primary person you wish to review this
- --review GITHUB_HANDLE - multiple people you wish to review this
- --fill - to create the PR content base on your commit(s) in your branch
- --draft - if you have a partial concept or unfinished work for a review. This gives a clear indication to not merge as is.
- --label MariaDB Corporation - a label to make it easy to distinguish your affiliation to make the MariaDB Foundations job easier to identify external contributions.

Wait for Review

While waiting for a review it might be possible that someone introduces changes in the code what conflict with your code. It might also be the case that you have some improvements to make. If this is the case, rebase or adjust your commits locally, and then *git push –force-with-lease* to the same branch as before. This will automaticity update the pull request and retrigger buildbot.

Examining Buildbot Results.

From the pull request number created, substitute that into the ID of the URL below will access all the builds based on that PR.

grid?branch=refs%2Fpull%2FID%2Fmerge

Related Articles

(when they are written)

- Reviews
- Automerge
- Rebasing your PR on a different branch (quick version - rebase, force push and then edit the title of the PR to the base branch).

This page is licensed: CC BY-SA / Gnu FDL

Quality Assurance

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See Code Coverage.

See Code Coverage With dgcov.

See Optimizer Quality.

See QA - Aria Recovery.

See QA Metrics.

See QA Tests.

See QA Tools.

See Random Query Generator Tests.

See RQG Extensions for MariaDB.

See Worklog Quality Checklist Template.

See Benchmarks and Long Running Tests.

See InnoDB Upgrade Tests.

See QA Datasets.

This page is licensed: CC BY-SA / Gnu FDL

Code Coverage

> **Note:**
>
> This page contains background information for MariaDB developers.

Code Coverage

We are working on getting more of the MariaDB source covered by our mysql-test-run (MTR) test suite. This is an ongoing task.

Goals for new code

For new code in MariaDB, we aim much higher:

The goals are:

1. All new lines of code should ideally be tested by MTR.
2. Code which cannot reasonably be tested by MTR needs to be tested by another tool and those code lines marked with `/* purecov: tested */`.

    - In this case the tool used for testing should be documented in the worklog entry for the code or in the commit message.

3. Code that can't reasonably be tested (such as error conditions) should be marked with `/* purecov: inspected */` so that a reviewer of the code can easily spot this code.
4. Code that is suspected to be deadcode should have a `DBUG_ASSERT(0)` or be marked with `/* purecov: deadcode */` so that we have a chance to notice if the code is ever executed.

The reason we are using 'purecov' to mark lines is an attribution to the purecov tool we originally used for code coverage in the early years of MySQL.

Markers

The recommended markers are:

`/* purecov: tested */`

- For code lines that are tested by something *other* than mysql-test-run:

`/* purecov: inspected */`

- For code lines that are hard to test but for which one has read the line multiple times to ensure it is correct. A code reviewer should also inspect these lines with care as they have not been properly tested.

`/* purecov: deadcode */`

- For code lines that one suspects will never be called. Having this marker allows us to generate a warning during mysql-test-run code coverage if this line is executed.

The comment must be placed on the line/lines that are affected.

For code blocks larger than 1 line one can use the block syntax:

```
/* purecov: begin tested */
....
/* purecov: end */
```

Running mysql-test-run With gcov

Prerequisites

1. First make sure that gcov 4.9 is installed.Older versions of the gocv library (lgcov) can't handle running several instances of a program in parallel. This causes the generated .gov files to not contain all executed lines when running `mysql-test-run` with the `--parallel` option or running test that starts several mysqld servers, like replication or spider tests.
2. Compile MariaDB with `BUILD/compile-pentium64-gcov` (if your machine does not have a pentium CPU, hack this script, or just live with the pentium-specific stuff).

Running mysql-test-run

To be able to see the level of coverage within the current test suite, do the following:

1. In the mysql-test directory, run this command: `./mysql-test-run -gcov`
2. To see which lines are not yet covered, look at `source_file_name.gcov` in the source tree. In MariaDB 10.1 or below it's in the CMakeFiles directory where the object files are stored. In MariaDB 10.2 it's stored together with the source files.
3. Think hard about a test case which will cover those lines that are not tested, and write one.

Tools

- You can use the code-coverage-with-dgcov/dgcov tool to check the coverage for the new code. This is especially written and maintained for MariaDB.
- For code coverage you also use the lcov tool.

Code coverage in buildbot

buildbot, the MariaDB build system, is doing automatic coverage testing for each push.

This page is licensed: CC BY-SA / Gnu FDL

Code Coverage With dgcov

> **Note:**
>
> This page contains background information for MariaDB developers.

The `dgcov` tool helps you check the coverage for new code. The `dgcov.pl` script is part of the mariadb-test framework (and any packages that include mariadb-test).

Overview

The `dgcov` program runs `gcov` for code coverage analysis aggregates the coverage data, and optionally reports coverage only for those lines that are changed by commits. Commits are specified in the `git diff` format.

If no commits are specified, the default is to work on all uncommitted changes, if any, otherwise on the last commit (in other words, on `git diff HEAD` or `git diff HEAD^`).

It's recommended that a developer runs dgcov on their new code before pushing it into a MariaDB repository.

Usage

```
./dgcov.pl --help
./dgcov.pl [options] [<commit> [<commit>]]
```

Options and Variables

Short Option	Long Option	Description
`-h`	`--help`	Print help and exit
`-v`	`--verbose`	Show commands run.
`-p`	`--purge`	Delete all test coverage information, to prepare for a new coverage test.
`-o`	`--only-gcov`	Stop after running gcov, don't run git
`-s`	`--skip-gcov`	Do not run `gcov`, assume `.gcov` files are already in place
`-g`	`--generate`	Create `.dgcov` files for all source files

How to Prepare Code for dgcov

Prior to running this tool, MariaDB should be built like this:

```
cmake -DENABLE_GCOV=ON
```

In addition, the test suite should be run. `dgcov` reports the coverage for all lines modified in the specified commits.

Output

Output `.dgcov` files have a conventional `gcov` format: Lines not covered are prefixed with `#####`, lines without generated code are prefixed with `-`, and other lines are prefixed with the number of times they were executed. See `info gcov` for more information.

The patch-like coverage for commits uses `gcov` format (as above) for lines, changed in these commits, and no prefix at all for lines that were not changed.

Examples

Checking the coverage for all unpushed commits:

```
dgcov.pl @{u} HEAD
```

Checking the coverage for all uncommitted changes:

```
dgcov.pl HEAD
```

Checking the coverage for a specific commit *1234567*:

```
dgcov.pl 1234567^ 1234567
```

mariadb-test-run can invoke `dgcov` automatically:

```
./mtr --gcov
```

In the latter case, the coverage for uncommitted changes (or for the last commit) is not printed to `stdout`, but to the `var/last_changes.dgcov` file.

Caveats

> **Note:**
>
> To be able to run `gcov` with the mariadb-test framework, you must have `gcc` version 4.8 or newer.

References

- `dgcov` was created by Kristian Nielsen and announced here.
- `dgcov` was reimplemented to aggregate the data, and to work for `git` and `cmake`, by Sergei Golubchik.

This page is licensed: CC BY-SA / Gnu FDL

QA Tests

> **Note:**
>
> This page contains background information for MariaDB developers.

Optimizer and the Random Query Generator

The RQG[47] is used to test various Optimizer features. See Optimizer Quality for more information.

Aria Engine Recovery

The QA - Aria Recovery page contains a plan on how to test it.

Upgrade and Installer Testing

- Upgrades using `.deb` and RPM packages are tested using very simple tests in BuildBot by the various `bld_kvm*` builders.

TODO

- More complex tests around `.deb`, RPM and tarballs.
- Decide on specific upgrade/downgrade paths (e.g. MySQL 5.5 to MariaDB 2.2?) and methods (`mysqldump`, `mysql_upgrade`) that we support and test each individually.
- Test the Windows installer and service NSIS allows for scripted unattended installs by providing an `/SD` argument to functions such as `MessageBox`.
- Test the contents of the Windows package, e.g. if `HELP`, `.test`, and other files are properly placed and runnable.

Linking Testing

The purpose of those tests is to check that various applications that use `libmysql` can be compiled, linked and run with MariaDB. They are run by the `compile-connectors` builder in BuildBot.

- Perl `DBD::mysql`
  - We configure and compile the Perl DBI MySQL driver. Then we run the test suite provided with it.
- PHP
  - We configure and compile both the `mysql` and `mysqli` PHP drivers without `mysql-nd`. For each, we run those tests from the PHP test suite that are known to be good (other tests fail for both MySQL and MariaDB).
- TODO: Perl and PHP with the embedded library.

Connectors Testing

The purpose of those tests is to check that the libraries that implement the MySQL protocol can work with MariaDB.

- The `libmysql` library/connector is tested both by the MTR test suite (since `mysqltest` links with it).

TODO

- PHP with the `mysql-nd` driver
- Connector C++
- JDBC

Replication Testing

Individual applications:

- group commit:

```
perl runall.pl \
  --engine=InnoDB \
  --grammar=conf/replication/rpl_transactions.yy \
  --gendata=conf/replication/rpl_transactions.zz \
  --mysqld=--sync_binlog=1 \
  --mysqld=--innodb-flush_log_at_trx_commit=1 \
  --mysqld=--binlog-dbug_fsync_sleep=100000 \
  --mysqld=--default-storage-engine=InnoDB \
  --threads=15 \
  --queries=1M \
  --duration=600 \
  --validator=None
```

This page is licensed: CC BY-SA / Gnu FDL

QA Metrics

> **Note:**
>
> This page contains background information for MariaDB developers.

Code Coverage

The `gcov` or `lcov` reports are run by BuildBot. The tests that are run to obtain the coverage percentage are listed under the `f_gcov` Factory in the BuildBot configuration file.

Bugs

The list of MariaDB bugs is available on JIRA.

This page is licensed: CC BY-SA / Gnu FDL

QA Tools

> **Note:**
>
> This page contains background information for MariaDB developers.

Tools used for quality assurance testing include:

- Worklog Quality Checklist Template is a template for creating a checklist for testing individual features or WorkLogs.
- MTR/mysqltest reference
- The Random Query Generator
- BuildBot Manual

See Also

- RQG Performance Comparisons
- RQG Extensions for MariaDB Features
- Optimizer Quality

This page is licensed: CC BY-SA / Gnu FDL

Optimizer Quality

> **Note:**
>
> This page contains background information for MariaDB developers.

Generating the Queries

There are 5 primary SQL grammars available for testing the Optimizer:

- `optimizer_nosubquery.yy` generates random queries with no subselects, with up to 3-way join and with various SQL clauses such as aggregates, `GROUP BY`, `HAVING`, and `LIMIT`.
- `optimizer_subquery.yy` generates queries with subselects with up to 2 levels of nesting. Subqueries are placed in various locations, such as in the `SELECT` list, in the `ON` clause, etc. Aggregates, `LIMIT`, `HAVING`, etc. are used if allowed by the server.
- `optimizer_subquery.yy` generates outer joins of various depths.

- `range_optimizer.yy` tests the range optimizer by joining several tables and generating various conditions on which range optimization is likely to be applied.
- `range_optimizer2.yy` tests the range optimizer by generating single-table queries that contain a lot of range-optimizable clauses. Avoiding joins allows the single table to be arbitrarily large, this allowing for more interesting range overlaps.

Validating the Results

The RQG[48] has two primary modes for validating the results:

- Using a reference implementation. This can be a PostgreSQL, JavaDB or another version or flavor of MySQL/Maria/Drizzle. Testing this way requires trusting external software that is not under our control. Also, it is sometimes difficult to determine which implementation has returned the correct result. Technically, 3 implementations can "vote" as to which is the correct result, but this is not reliable if the implementations all derive from one another.
- Executing the generated query using a different execution plan. This is usually achieved by disabling particular optimizations and thus "downgrading" the plan to a more basic, presumed stable one. It is assumed that a nested-loop-join that takes no advantage of indexes would always provide the correct result. The advantage of this approach is that there is no need for a reference implementation and the source of the wrong result can be obtained by diffing the original and the downgraded execution plan.

In addition to result set validation, there is a module which executes each generated `SELECT` in various contexts, such as part of a union, stored procedure, trigger, etc. and makes sure that the query returns a correct result. This is most often used for testing subselects.

Running a Complete Test Cycle

A test cycle is described in a configuration file called the CC file. The CC file contains a list of mysqld options to use, the list of grammars to use and other settings (e.g. storage engines, table sizes, etc.). The testing framework will then take a random permutation from the settings described in the file and run them as a RQG test for a predefined time, such as 10 minutes. This is repeated up to 100 times, each with a different random permutation. The PRNG seed for each run will also be different, so different queries will be generated for each run, in addition to using different mysqld options, engine, etc.

By default, all cycles include MyISAM, Aria and InnoDB, and some percentage are run under Valgrind. Cycles run with both `NULL` and `NOT NULL` fields and with and without simple views.

Configuration for Join Cache Testing

`outer_join_with_cache` is always `ON`. `--join_cache_level` varies from `0` to `8`. `--join_buffer_size` varies between 1, 100, 1K, 10K and 100K. The `optimizer_no_subquery.yy`, `outer_join.yy` and `range_access.yy` grammars are used. Once semijoin is stable, `join_cache` + `semijoin` are tested with `optimizer_subquery.yy`.

Configuration for MRR/ICP/DS-MRR-CPK Testing

`--optimizer_use_mrr` is ON, `mrr_sort_keys` is both `ON` and `OFF`, `index_condition_pushdown` is both ON and OFF, `join_cache_level` is between 0 and 8, `join_buffer_size` and `mrr_buffer_size` are 1, 100, 1K, 10K and 100K. `optimizer_no_subquery.yy`, `outer_join.yy`, `range_access.yy` and `range_access2.yy` grammars are used.

Configuration for Subquery Testing

The `optimizer_no_subquery.yy` grammar is used. Each individual `optimizer_switch` related to subquery optimization may be disabled so that the "second best" plan is generated.

See Also

- RQG Documentation
- RQG Performance Comparisons
- RQG Extensions for MariaDB Features
- QA Tools
- Worklog Quality Checklist Template

This page is licensed: CC BY-SA / Gnu FDL

QA - Aria Recovery

> **Note:**
>
> This page contains background information for MariaDB developers.

General Principles

Recovery is tested via the RQG, which provides a random workload against the server, and then uses kill -9 to kill the process. After that, recovery is attempted both by using `maria_read_log` and by restarting the `mysqld` process. Once the server has started up, the tables are verified in various ways, including `ALTER|OPTIMIZE|ANALYZE|REPAIR TABLE` as well `SELECT` queries that read the table back and forth using various access methods.

A combinations `.cc` file named `lp:randgen/conf/engines/maria/maria_recovery.cc` is used to define various `mysql` options and RQG parameters that are relevant to recovery. Then, RQG's `combinations.pl` script is used to run hundreds of individual test runs. Each run uses a random permutation from the settings in the `.cc` file in order to generate a unique workload that is then validated via the `Recovery` RQG Reporter.

Individual tests

The following are the individual tests or test runs that must be completed or created in order to ensure that Aria recovery is solid.

Standard kill -9 Testing

Done 2011-02-28 The standard `conf/engines/maria/maria_recovery.cc` passes with no failures when run with hundreds of trials.

Testing With Small Block Sizes

On hold pending 2 bug fixes related to –maria-block-size=1K and –maria-block-size=2K

Testing With Small Page Cache Size

Done 2011-03-04 Completed 400 rounds with

```
'--mysqld=--maria-block-size=4K --mysqld=--maria-pagecache-buffer-size=128K',
'--mysqld=--maria-block-size=16K --mysqld=--maria-pagecache-buffer-size=256K',
'--mysqld=--maria-block-size=32K --mysqld=--maria-pagecache-buffer-size=512K'
```

Two pre-recovery crashes were filed, no recovery issues.

Killing and Restarting the Recovery Process

In Progress

The AriaDoubleRecovery reporter currently attempts double recovery via `maria_read_log`. The first invocation of `maria_read_log` is killed halfway through the process, and the second invocation is left to complete the recovery.

Future testing involves doing the same with the `mysqld` server in place of `maria_read_log`.

Another Realistic Workload

The usefulness of the SMF workload, derived from the SimpleMachines forum application means that another such workload is required in order to make sure no residual recovery bugs remain. Hopefully something can be cooked up using Wikipedia so that longer fields and blobs are exercised.

Transactional consistency

A transactional grammar that simulates transactions using `LOCK TABLE` statements is required. The `RecoveryConsistency Reporter` can then be used to validate that no partial transactions appear in the database after recovery.

See Also

- RQG Performance Comparisons
- RQG Extensions for MariaDB Features
- Optimizer Quality
- QA Tools
- Worklog Quality Checklist Template

This page is licensed: CC BY-SA / Gnu FDL

Worklog Quality Checklist Template

> **Note:**
>
> This page contains background information for MariaDB developers.

The purpose of this template is to produce individual checklists for testing individual Worklogs. Using a checklist ensures that all WLs[49] are treated equally and that all important points are addressed and tested, or at least considered when clearing the WL[50] for merging into a stable release.

Inputs

This page lists all sources of information which can be used as inputs to the testing process. The goal is to gather comprehensive information before creating a testing plan.

Usage Scenario

TODO: Summarize here the intended use for the feature.

Design, Architecture, Specification

- **WorkLog number and title:**
  - *If there is no Worklog, create one or ask one to be created*
- **Are HLS and LLS present?**
  - *If the HLS or LLS sections are missing, add them or ask that they are created*

- **Reviewer name:**
  - *Testing suggestions and concerns should be obtained by the reviewer*
- **Customer requirements:**
  - *Make yourself familiar with any requirements or documents supplied by the customer*
- **Other implementations:**
  - *Make yourself familiar with other implementations of the feature in other database products or flavors of MySQL*

Testability and Observability

- **Does the feature have an on/off switch?**
  - *An on/off switch is required in at least the following situations:*
    - *A RQG comparison test will be performed in order to validate the feature;*
    - *The feature may cause a performance degradation for certain queries or usage scenarios;*
    - *The feature is experimental or of interest to a small number of users;*
- **Is the operation of the feature observable?**
  - *Does the feature leave any traces in* `SHOW STATUS` *or* `EXPLAIN`*, or in the logs?*

The Patch

- **List of modified files:**
  - *Obtain a list of all modified files from the version control system, to serve as a starting point for identifying areas of risk within the server or potential interactions with other features. A larger number of modified files may indicate larger risk of unforeseen interactions*
- **Review comments:**
  - *Paste a link to the review and read it for any concerns that the reviewer may have raised*
- **MTR tests:**
  - *Does the patch include enough MTR tests?*
- **DGCov output:**
  - *Generate a report or obtain it from a developer and make sure it is acceptable for all parties*

Feedback From Interested Parties

- **Feedback from developer:**
  - *Paste here any useful information provided by the developer in informal discussions, emails or chat*
  - *Make sure any concerns raised by the developer are recorded in this document*
- **Feedback from reviewer:**
  - *Paste here any discussions with the reviewer related to concerns, risks or testing, if not included in the formal review comments for the patch itself*
- **Feedback from customer:**
  - *Summarize here any relevant discussions with the customer*

Query Patterns

If the feature relates to the optimization of a particular set of queries, provide here examples that were extracted from the sources of information listed above, in particular the Worklog and any MTR test cases provided with the patch.

Risks

This section lists any risks inherent to the Worklog, as identified by reviewing the inputs mentioned

Refactoring

- **Areas that have been refactored:**
  - *List areas that have been refactored in order to accommodate the new functionality*

Interactions

- **Areas of potential interaction:**
  - *List areas in the server that may have an interaction with the feature*
- **Does the feature contain engine-specific code, hooks or hacks?**
  - *List any engine-specific considerations there may be*
- **Does the feature define a new API?**
- **Does the feature relate to transactions or transactional integrity?**
  - *If yes, is testing required on all storage engines, or using cross-engine transactions or binlog involvement?*
- **Does the feature relate to any of the core server features?**
  - *Do we need to test with views, triggers, stored procedures, functions, events, partitions, foreign keys, time zones*

Upgrade or Downgrade

- **Does the feature change any on-disk formats?**
  - *If yes, testing would be required that upgrading to the new format works properly*
- **Does the feature change the format of the system tables in the `mysql` database?**
  - *If yes, testing would be required to make sure that any upgrade/downgrade scripts are able to handle the change gracefully*
- **Does the feature change or retire or deprecate any mysqld configuration options?**
  - *List options here*
- **Is downgrade to be supported?**
  - *If using the new feature makes the database non-downgradable, this must be a conscious design decision*

User-Visible Changes

- **Does the feature change the mysql protocol?**
  - *If yes, full connectors testing is required to make sure the change is compatible*
- **Does the feature change any result formatting or data types?**
  - *If yes, testing via the `mysql` client is required to verify that the new format or data type properly arrives at the client*
- **Does the feature change the syntax or the output of a `SHOW` command?**
  - *If yes, clients, connectors and tools that issue that `SHOW` command and process its output should be tested*
- **Does the feature change anything with respect to `INFORMATION_SCHEMA` and `PERFORMANCE_SCHEMA` ?**

Performance and Optimizer

- **Does the feature have any performance implications?**
  - *Is the feature expected to deliver any performance improvement?*
  - *Is the feature feared to have a performance impact?*

- **Does the feature interact with the Optimizer or indexes?**
  - *Does the feature change anything related to indexes or data stored in them?*
  - *Does the feature change anything with respect to any optimizations?*

Replication

Does the feature generate or transmit data that may be sent or processed by the slave or tools such as `mysqlbinlog`*?*

- **Which binary log formats should be tested?**
  - *Certain optimizations or scenarios may only trigger for a particular binary log mode*
- **Are existing replication RQG grammars sufficient to trigger the new code?**
- **Do we need to test DDL in replication, log rotation and such?**
- **Does the change affect the master as well, or just the slave? Is a transactional integrity test indicated?**
- **Do we need to simulate replication failures, server restarts and such?**
- **Do we need to test cloning new slaves from scratch, binary log replay and other replication maintenance activities?**
- **Do we need to cover exotic data types, overly large records, statements that are only safe in a particular binlog mode, etc?**

Extreme Conditions and Corner Cases

- **Does the feature deal with extreme conditions and corner cases?**
  - *e.g. huge values, huge tables, huge row lengths, blobs, high concurrencies*
- **Is any form of stress testing indicated for the feature?**
  - *Does the feature have any thread interactions, locking or concurrency mechanisms?*

Testing Plan

This section identifies the testing that will be performed based on the risks identified above

Additional MTR Tests

Describe any MTR tests that will be implemented in addition to the ones already present in the patch

RQG Tests

- **Are new Random Query Generator grammars required?**
  - *If existing grammars do not cover the new feature*
- **Are new Validators or Reporters required?**
  - *If the existing RQG validation code can not verify the proper operation of the feature*
- **RQG tests lines to run:**
  - *provide actual RQG command lines, along with a description of the desired effect*

Manual Testing

Describe any manual testing that will be performed involving connectors, clients or installers

Benchmarks and Stress Tests

Describe any benchmarks and stress tests that will be performed. Include the exact parameters and command lines that will be used

Hand-Off

This section identifies final checks to be performed on the test content when it is delivered.

Requirements for MTR Tests

- Are all new mysqld options, `SHOW STATUS` variables, `I_S`[51] tables and such covered with an MTR test?
- Are all new error messages covered with a test?
- Is code coverage acceptable to all parties involved?

Requirements for Random Query Generator Tests

- Has the proper set of RQG grammars been chosen or created?
- Do we have a .CC file listing all interesting mysqld options and RQG variants, in order to run a combinations test? Was the CC file pushed into RQG's LP repository?
- Has an at least 24-hour combination test been run?
- Was code coverage run on the RQG test?
- Were all new grammars and Perl modules pushed into the RQG LP repository?
- Were the RQG tests prepared to be buildbot-ready, as opposed to manually-monitored?
- Does the test include any safeguards against false negatives?
  - *e.g. all queries generated are bad, which causes no actual replication, which is still reported as test success*
- Are the tests running stably and green in the feature tree's BuildBot?
- Are any bug reports missing reproducible test cases, core files, RQG command lines and the like?

Requirements for Code Coverage

This makes it much faster to review code as one knows what it tested and what one needs to review more carefully

- Ideally all code should be tested by MTR;
- If code can only be reproduced by an RQG test or other external test, sufficient information must be provided in the Worklog for the test to be reproducible;
- All code should either be tested or reviewed and marked with `/* purecov inspected */`
- Code that is suspected to be unreachable should have an DBUG_ASSERT(0) and be marked with /* purecov: deadcode */.
- Was an MTR code coverage report for the feature tree generated by i7.stoev.org/lcov?
- Has the dgcov coverage tool been run?
- Are there any notable regressions in the code coverage?
- Was code coverage run on the RQG tests alone, along with an effort to increase it?
- Is the code covered by comments?

Requirements for Code Trees

- Was the stable tree merged **into** the feature tree recently?
- Were all RQG tests pushed into lp:randgen and documented on the RQG manual pages?
- Were all RQG or other automatable tests installed in BuildBot?
- Is the feature tree BuildBot green?

Requirements for Closing the Worklog as Complete

- Was the Worklog updated with information on the testing performed?
- Was the QA check box in the Worklog ticked?

- Were all related bugs on Launchpad closed? This includes A) bugs that were tagged to belong to the feature tree, B) bugs assigned to the developer regardless of tree, C) bugs assigned to the tester and D) and orphan bugs just hanging around the same theme or time frame;
- Did the WL complete the entire review procedure, including any back-and-forth emails?
- Has the new feature been documented?

See Also

- RQG Documentation
- RQG Performance Comparisons
- RQG Extensions for MariaDB Features
- Optimizer Quality
- QA Tools

This page is licensed: CC BY-SA / Gnu FDL

Random Query Generator Tests

> **Note:**
>
> This page contains background information for MariaDB developers.

PQuery is currently in development to perform random query generation.

The documentation for the Random Query Generator is available here.

The latest RQG used by MariaDB can be found at:

- https://github.com/elenst/rqg/tree/master
- https://github.com/MariaDB/randgen/tree/master

The list of automatic tests is located in the `buildbot.conf` file in the `mariadb.org-tools` repository.

See Also

- RQG Performance Comparisons
- RQG Extensions for MariaDB Features
- Optimizer Quality
- QA Tools
- Worklog Quality Checklist Template

This page is licensed: CC BY-SA / Gnu FDL

RQG Extensions for MariaDB

> **Note:**
>
> This page contains background information for MariaDB developers.

All recent reasonably stable changes can be found in our Github repository. The link points to the master branch; there are other branches created for more specific cases.

While all described changes were made in order to test MariaDB code, many of them are applicable to other MySQL flavors as well.

Galera Mode

A set of changes to support running RQG tests with multi-master Galera replication, implemented to test MariaDB Galera Cluster.

The top-level script `runall-new.pl` got a new option `--galera`, which takes a string value. The string can be a combination of `'m'` or `'s'`, where each symbol represents a Galera node. `'m'` stands for `'master'`, and `'s'` stands for `'slave'`.

The new module `GaleraMySQLd.pm` implements cluster initialization and startup. It is only interested in the number of nodes.

Internals are adapted to the multi-master or mixed mode:

- Instead of executing data generation on each started server, it is only executed on the first master.
- Test flow is executed on each master (but not on slaves).
- No result comparison is performed after each query, the flow runs independently, but, in the end, if no other errors were encountered, `runall-new.pl` performs dump comparison in the usual manner.

Also, a sample grammar and data template are provided in `conf/galera`, they are slightly modified versions of `conf/engines/engine_stress.yy` and `engine_stress.zz`, adapted for Galera taking into account its limitations (e.g. no `LOCK`).

Note: `GaleraMySQLd.pm` has defaults for everything mandatory but `wsrep_provider`. If you run the test with more than 1 node, you need to provide the value on the command line in the usual manner (`--mysqld=--wsrep-provider=...`), otherwise the test fails with the corresponding error message.

So, if you provide the following option:

```
--galera=mms --mysqld=--wsrep-provider=<path to Galera library>
```

3 nodes will be started, and Galera replication will be set up between them; the data will be generated on the first node; the test flow will be executed on the first and second nodes; at the end of the test, data dump from each node will be taken and compared; node vardirs will be placed under `<vardir>/node0`, `<vardir>/node1`, `<vardir>/node2`.

CheckFieldValue Validator

A grammar can set requirements on results of a query through a specifically formatted comment. If the validator finds a comment which matches the template, it performs the requested check. The validation is defined in the comment itself: it says which field in which row in the result set should be checked, and provides the condition (currently simple numeric comparisons: `=`, `<`, `>`, `<=`, `>=`).

It allows to do simple verification without implementing a special validator.

Example:

A query in the grammar file

```
  SELECT COUNT(*), MAX(`pk`) FROM _table /* Validate 2 > 10 for row all*/;
```

The comment means “Check that the value in field 2 is greater than 10 for all rows” (for this query “all rows” is overkill, it could just as well be “row 1”).

Assuming that there is a table with not more than 10 rows, the validator will produce the error:

```
ERROR: For query 'SELECT COUNT(*), MAX(`pk`) FROM `BB` /* Validate 2 > 10 for row 1*/' on row 1 result 10
does not meet the condition > 10
Full row: [1] : 1; [2] : 10;
```

The exit code for this failure is `STATUS_REQUIREMENT_UNMET`.

Usage example: investigation of the sporadic failure MDEV-4578.

MariadbGtidCrashSafety Reporter

The reporter was created to test slave crash-safety with MariaDB implementation of GTID. It is similar to SlaveCrashRecovery reporter, but is adjusted to check GTID-specific aspects:

- It restarts the slave with `--skip-slave-start`, and executes these statements:

```
CHANGE MASTER .. MASTER_USE_GTID=current_pos;
STOP SLAVE;
```

So, regardless how the replication was set up initially, after the first crash or restart, it starts using GTID for sure, while the slave is running, and also before a crash and after restart the reporter checks `gtid*` variables and the contents of the system table `mysql.gtid_slave_pos` and verifies basic consistency of the data, e.g. watches that GTID `seq_no` constantly grows.

SlaveCrashRecovery Reporter

The reporter can be used to test crash safety of replication.

It is a periodic reporter, every 30 seconds it kills the slave server using `SIGKILL`, and immediately restarts it on the old data directory, with the same parameters as before. On server restart, the reporter checks that the server and the replication started all right.

The reporter itself does not check consistency of the data, but it can be used together with `ReplicationConsistency` reporter.

It is supposed to be used with `runall-new.pl`, so that the server is started without MTR involvement.

Usage example: testing of GTID in MariaDB.

BinlogConsistency Reporter

The reporter checks that the contents of the binary log correctly reflects the contents of the server.

After the main test flow is finished, the reporter creates a data dump of the server, stores its binary logs, shuts down the server, starts a new one with the same parameters, but on a clean data directory, replays the binary logs into it, creates a dump of the new server, and compares two dumps.

It is to be used with `runall-new.pl`.

Usage example: testing of binlog changes in MariaDB 10.0 (e.g. MDEV-181, MDEV-232)

CrashRecovery Reporter

The idea is very much the same as in the old Recovery reporter: crash the server at the end of the test, restart it and make sure it started all right, and the data is not corrupted. The main difference is that the old reporter restarts the server in a hardcoded pre-defined manner, which limits its use. Instead, CrashRecovery reporter starts the server with the same set of options as the initial one, which imitates a more realistic scenario, and also allows to use it on non-default InnoDB configurations.

It is to be used with `runall-new.pl` .

LimitRowsExamined Transformer

The transformer was developed for testing the LIMIT ROWS EXAMINED functionality added in MariaDB 5.5. It can be used in the usual way, by providing its name in the `--transformers` list.

The transformer checks whether the original query already contains a `ROWS EXAMINED` clause. If it does not, it adds the clause either after the `LIMIT` clause, or at the end of the query. In any case (even if `ROWS EXAMINED` was already there), the transformer returns the following sequence of statements:

- `FLUSH STATUS`
- <the query with `ROWS EXAMINED`>
- <a query which sums up status variables related to examined rows>

The result of the main query is checked to be a subset of the original query's result set. The sum of status variables is checked to be not greater than the limit provided in the `ROWS EXAMINED` clause, plus a margin. The margin is configured in the transformer.

If the result of the transformed query appears not to be a subset of the original result set, `STATUS_LENGTH_MISMATCH` is returned.

If the sum of status variables is greater than the maximum allowed value, `STATUS_REQUIREMENT_UNMET` is returned.

Note: Status values `STATUS_REQUIREMENT_UNMET` and `STATUS_REQUIREMENT_UNMET_SELECT` were added to `Constants.pm`.

ShowExplain Validator

The validator was developed for testing SHOW EXPLAIN.

The validator checks that the output of `SHOW EXPLAIN` correlates with the output of traditional `EXPLAIN` executed for the same query. It also tries to filter out known expected mismatches between the produced plans.

Comparison of VIEW Algorithms

RQG already provided `--views[=<view type>]` option, which means that in addition to the normal data generation, views of the requested types will be added and used in the test flow. However, you could only create views of the same type on servers that you were comparing. Now there are `--views1` and `--views2` options, which work the same way as `--basedir1` / `--basedir2`, `--mysqld1` / `--mysqld2`, etc. The backward compatibility is preserved, so you can still use `--views` which will be applied to both servers unless there is `--views1` or `--views2` to override it.

The change was made in `gentest.pl` and both `runall.pl` and `runall-new.pl`.

Usage example: testing of MERGE view extension in MariaDB 10.0 (MDEV-3862)

Multiple Redefining Grammars

RQG allows to provide `--redefine=<grammar file>` option to override or add rules to the main grammar. Sometimes you have a redefining grammar for a feature, and another redefining grammar for an engine, and yet another redefining grammar for a specific version, and combining them makes the number of grammars to grow uncontrollably. The simple change has been made to allow providing several redefining grammars at once, for instance:

```
runall.pl ... --redefine=memory_engine_redefine.yy --redefine=binlog_format_stmt_redefine.yy --
redefine=legacy_version_redefine.yy
```

> **Warning:**
>
> It is better not to allow them to override each other, since the result can be unexpected (that is, they should define different sets of rules).

The change was made in `gentest.pl` and both `runall.pl` and `runall-new.pl`.

Keywords _basetable and _view

The RQG keyword `_table` means any table or view from the test dataset. Often it's good enough, but sometimes it badly increases the number of erroneous statements. Now tables and views can be differentiated in a grammar depending on what the goal is: `_table` can still be used for both tables and views, while _basetable will only pick tables, and _view will only pick views.

Example: If you create the following rule in your grammar:

```
alter:
        ALTER TABLE _table ENGINE=MyISAM;
```

When you are running the test with views, you get lots of `ER_WRONG_OBJECT` errors ("is not BASE TABLE"). This is hardly the goal, so you can improve the grammar by changing it to this:

```
alter:
        ALTER TABLE _basetable ENGINE=MyISAM;
```

That way, the rule only picks "real" tables. Of course, if you still want to pick views sometimes, as a sanity check that there is no crash or anything, you can make the rule a bit more complicated, but much better tuned than it was:

```
alter:
        ALTER TABLE alter_object ENGINE=MyISAM;

alter_object:
        _basetable | _basetable | _basetable | _basetable | _basetable | _view ;
```

See Also

- RQG Performance Comparisons
- RQG Extensions for MariaDB Features
- Optimizer Quality
- QA Tools
- Worklog Quality Checklist Template

This page is licensed: CC BY-SA / Gnu FDL

Benchmarks and Long Running Tests

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See Benchmark Results.

See Benchmarks.

This page is licensed: CC BY-SA / Gnu FDL

Benchmark Results

- sysbench v0.5 - Single Five Minute Runs on T500 Laptop
- sysbench v0.5 - Three Times Five Minutes Runs on work with 5.1.42
- sysbench v0.5 - 3x Five Minute Runs on work with 5.2-wl86
- sysbench v0.5 - 3x Five Minute Runs on work with 5.1 vs. 5.2-wl86
- sysbench v0.5 - 3x 15 Minute Runs on perro with 5.2-wl86 a
- sysbench v0.5 - 3x 15 Minute Runs on perro with 5.2-wl86 b

- Select Random Ranges and Select Random Point
- Sysbench Results
- sysbench v0.5 - Single Five Minute Runs on perro
- sysbench v0.5 - Single Five Minute Runs on work
- Threadpool Benchmarks

This page is licensed: CC BY-SA / Gnu FDL

sysbench v0.5 - Single Five Minute Runs on T500 Laptop

sysbench v0.5 - Three Times Five Minutes Runs on work with 5.1.42

sysbench v0.5 - 3x Five Minute Runs on work with 5.2-wl86

sysbench v0.5 - 3x Five Minute Runs on work with 5.1 vs. 5.2-wl86

sysbench v0.5 - 3x 15 Minute Runs on perro with 5.2-wl86 a

sysbench v0.5 - 3x 15 Minute Runs on perro with 5.2-wl86 b

Select Random Ranges and Select Random Point

Sysbench Results

sysbench v0.5 - Single Five Minute Runs on perro

sysbench v0.5 - Single Five Minute Runs on work

Threadpool Benchmarks

Benchmarks

- Benchmark Builds
- Benchmarking Aria
- DBT3 Automation Scripts
- DBT3 Benchmark Results InnoDB
- DBT3 Benchmark Results MyISAM
- DBT3 Example Preparation Time
- MariaDB 5.3 - Asynchronous I/O on Windows with InnoDB
- MariaDB 5.3/MySQL 5.5 Windows performance patches
- mariadb-tools
- Performance of MEMORY Tables
- Recommended Settings for Benchmarks
- RQG Performance Comparisons
- run-sql-bench.pl
- Segmented Key Cache Performance
- sysbench Benchmark Setup

This page is licensed: CC BY-SA / Gnu FDL

Benchmark Builds

Benchmarking Aria

We have not yet had time to benchmark Aria properly. Here follows some things that have been discussed on the maria-discuss email list.

Aria used for internal temporary tables

By default Aria (instead of MyISAM) is used for the internal temporary tables when MEMORY tables overflows to disk or MEMORY tables can't be used (for example when you are using temporary results with BLOB's).
In most cases Aria should give you better performance than using MyISAM, but this is not always the case.

```
CREATE TABLE `t1` (`id` INT(11) DEFAULT NULL, `tea` TEXT)
  ENGINE=MyISAM DEFAULT CHARSET=latin1;
INSERT t1 SELECT rand()*2e8, repeat(rand(), rand()*64) FROM t1;
```

Repeat the last row until you get 2097152 rows.

The queries tested

```
Q1: SELECT id, tea from t1 group by left(id,1) order by null;
Q2: SELECT id, count(*), tea from t1 group by left(id,1) order by null;
Q3: SELECT id, tea from t1 group by left(id,2) order by null;
Q4: SELECT id, count(*), tea from t1 group by left(id,2) order by null;
Q5: SELECT id, tea from t1 group by id % 100 order by null;
Q6: SELECT id, count(*), tea from t1 group by id % 100 order by null;
```

Results (times in seconds, lower is better):

Test	Aria 8K page size	Aria 2K page size	MyISAM
Q1	3.08	2.41	2.17
Q2	6.24	5.21	12.89
Q3	4.87	4.05	4.04
Q4	8.20	7.04	15.14
Q5	7.10	6.37	6.28
Q6	10.38	9.09	17.00

The good news is that for common group by queries that is using summary functions there is a close to 50 % speedup of using Aria for internal temporary tables.

Note that queries Q1,Q3 and Q5 are not typical queries as there is no sum functions involved. In this case rows are just written to the tmp tables and there is no updates. As soon as there are summary functions and updates the new row format in Aria gives a close to 50 % speedup.

The above table also shows how the page size (determined by the aria_block_size system variable) affects the performance.
The reason for the difference is that there is more data to move back/from the page cache for inserting of keys. (When reading data we are normally not copying pages). The bigger page size however allows longer keys and fewer index levels so for bigger data sets the different should be smaller. It's possible to in the future optimize Aria to not copy pages from the page cache also for index writes and then this difference should disappear.

The default page size for Aria is 8K.

If you want to run MariaDB with MyISAM for temporary tables, don't use the configure option '–with-aria-tmp-tables' when building MariaDB.

This page is licensed: CC BY-SA / Gnu FDL

DBT3 Automation Scripts

DBT-3 (OSDL Database Test 3) is a workload tool for the Linux kernel that OSDL
(Open Source Development Labs, inc) developed based on TPC-H which is provided by the Transaction Performance Processing Council (TPC).

DBT-3, like TPC-H, simulates an actual decision-making support system and models complex business analysis applications that perform data processing jobs for making better business decisions. By running the workload that DBT-3 simulates, it is possible to verify and measure the performances of the Linux kernel in an actual decision-making support system.

DBT-3 uses the “scale factor (SF)” as a stress indicator of the system. By varying the SF, it becomes possible to make the size of a database the SF times its size.

The tests performed by DBT-3 comprise the three tests listed below. DBT-3 obtains the execution times of these three tests as well as the system status information and database statistics information.

1. Load test

- Enters the data to be used for the Power and Throughput tests into the database. Makes a bulk insert of the huge CSV data corresponding to the scale factor into the database.

1. Power test

- Performs 22 complex queries.

1. Throughput test

- Performs the same 22 queries as in the Power test simultaneously in more than one process.

For the purpose of this task, only the Power test is performed over preliminary
prepared database with various Scale factors. The time for each query execution
will be measured and stored into a database. Later the results of one whole
test with all 22 queries will be rendered into a histogram graphics comparing
it to different configurations.

Benchmark environment preparation

sudo rights

The user that will run the benchmark must have sudo rights on the machine.

For clearing the system caches between query runs, the automation script uses the following command:

```
sudo /sbin/sysctl vm.drop_caches=3
```

This command must be run with superuser rights. Even if a user supplies a password to sudo, this password expires after some timeout. In order for this command to be run without requiring password, the following line should be added to the sudoers file (edit it with the `"sudo visudo"` command):

```
'your_username' ALL=NOPASSWD:/sbin/sysctl
```

…where `'your_username'` is the user that will run the benchmark.

Required software

The automated DBT3 benchmark requires the following software:

- Perl
  - Project home:
- mariadb-tools
  - Project home: mariadb-tools
  - The project folder is called "`dbt3_benchmark`" and is under `mariadb-tools`.
- dbt3-1.9
  - Download location:
- Gnuplot 4.4 — graphics output program.
  - Project home:
- Config::Auto — a Perl module that reads configuration files. To install it use the following command:

```
sudo cpan Config::Auto
```

- DBD::mysql — a Perl module to connect to MariaDB/MySQL and PostgreSQL. To install it use the following command:

```
sudo cpan DBD::mysql
```

NOTE: You may receive an error saying that CPAN could not find `mysql_config`. In this case you have to install the mysql client development library. In OpenSuse the command is:

```
sudo zypper install libmysqlclient-devel
```

Alternatively this module can be installed manually by following these steps:

1. Download DBD-mysql-4.020.tar.gz from mysql.pm and unpack it
2. Run the perl script PerlMake.pl under the unzipped dir:

```
perl Makefile.PL --mysql_config=/path/to/some/mysql_binary_distribution/bin/mysql_config
```

1. Run `make` to compile `DBD::mysql`:

```
make
```

1. Add the necessary paths in order to run `DBD::mysql`:

```
export PERL5LIB="/path/to/unzipped_DBD_mysql/DBD-mysql-4.020/lib"
export LD_LIBRARY_PATH="/path/to/unzipped_DBD_mysql/DBD-mysql-
4.020/blib/arch/auto/DBD/mysql/:/path/to/some/mysql_binary_distribution/lib/"
```

Tested DBMS

- MySQL 5.5.x
  - Download location: #downloads →
    Generally Available (GA) Releases → Linux - Generic 2.6 (x86, 64-bit),
    Compressed TAR Archive - downloads mysql-5.5.x-linux2.6-x86_64.tar.gz - gzipped tar file for Linux x86
- MySQL 5.6.x
  - Download location: #downloads →
    Development Releases → Linux - Generic 2.6 (x86, 64-bit), Compressed TAR
    Archive - downloads mysql-5.6.x-m5-linux2.6-x86_64.tar.gz - gzipped tar file for Linux x86
- MariaDB 5.3.x

    - Download location: 5.3 , downloaded with Bazaar:

```
bzr branch lp:maria/5.3
```

- MariaDB 5.5.x
    - Download location: 5.5 , downloaded with Bazaar:

```
bzr branch lp:maria/5.5
```

- PostgreSQL
    - Download location:

NOTE: The DBT3 benchmark requires a lot of disk space (for example MySQL 5.5.x + MyISAM database with scale factor 30 takes about 50 GB). Also some queries require the utilization of temp tables under the directory set by the `--tmpdir` startup parameter passed to `mysqld`. In the prepared configuration files the temp directory is pointed to the `mysql` system directory of the binary distribution, but one should reassure that there is enough free space available for the temp directory.

Installation instructions

NOTE: The directory where all the files will be downloaded or installed will be referred as `$PROJECT_HOME`. This could be for example `~/benchmark/dbt3`.

Download **mariadb-tools**

1. Go to your project folder

```
cd $PROJECT_HOME
```

1. Get the latest branch from LaunchPad with Bazaar:

```
bzr branch lp:mariadb-tools
```

Now the project for the dbt3 benchmark test will be in the following dir:

```
$PROJECT_HOME/mariadb-tools/dbt3_benchmark/
```

The project `dbt3_benchmark` has the following directories and files:

- config — a folder where the configuration files for MariaDB, MySQL and PostgreSQL are stored. They are divided into subfolders named ‘`sXX`’, where `XX` is the scale factor.
- dbt3_mysql — a folder with all the necessary files for preparing DBT3 databases and queries for the tests with MySQL and
  MariaDB
- tests — a folder where the different test configurations are stored. It contains the following directories:
    - db_conf — here are stored the database configuration files
    - queries_conf — here are stored the different queries configuration files
    - results_db_conf — here is stored the configuration of the results database
    - test_conf — here are the test configurations
    - launcher.pl — a perl script that automates the test. Details about calling and functionality of this file are listed later on this page.

Prepare benchmark workload and queries

For the purpose of the benchmark from DBT3-1.9 we will only need **DBGEN** and **QGEN**. DBGEN is a tool that generates a workload for the test and QGEN is a tool that generates the queries used for the test.

1. Go to
2. Download the archive for DBT3 1.9 into your project folder $PROJECT_HOME
3. Unzip the archive into your project folder

```
cd $PROJECT_HOME
tar -zxf dbt3-1.9.tar.gz
```

1. Copy the file tpcd.h into the dbt3 folder. This step includes the necessary labels for MySQL/MariaDB when building queries.

```
cp $PROJECT_HOME/mariadb-tools/dbt3_benchmark/dbt3_mysql/tpcd.h $PROJECT_HOME/dbt3-1.9/src/dbgen/
```

1. Copy the file Makefile under `$PROJECT_HOME/mariadb-tools/dbt3_benchmark/dbt3_mysql/` into the dbt3 folder

- NOTE: This step is executed only if you want to overwrite the default behavior of PostgreSQL settings. After copying this Makefile and building the project, QGEN will be set to generate queries for MariaDB/MySQL. If you skip this step, QGEN will generate queries for PostgreSQL by default.

```
cp $PROJECT_HOME/mariadb-tools/dbt3_benchmark/dbt3_mysql/Makefile $PROJECT_HOME/dbt3-1.9/src/dbgen/
```

1. Go to $PROJECT_HOME/dbt3-1.9/src/dbgen and build the project

```
cd $PROJECT_HOME/dbt3-1.9/src/dbgen
make
```

1. Set the variable DSS_QUERY to the folder with template queries for MariaDB/MySQL or for PostgreSQL
2. If you want to build the queries that fit MariaDB/MySQL dialect execute the following command:

```
export DSS_QUERY=$PROJECT_HOME/mariadb-tools/dbt3_benchmark/dbt3_mysql/mysql_queries/
```

1. If you want to use the default PostgreSQL templates, execute the following command:

```
export DSS_QUERY=$PROJECT_HOME/dbt3-1.9/queries/pgsql/
```

1. Create a directory to store the generated queries in

```
mkdir $PROJECT_HOME/gen_query
```

1. Generate the queries

NOTE: The examples use scale factor 30. If you want different scale, change the value of `-s` parameter

-

cd $PROJECT_HOME/dbt3-1.9/src/dbgen
./qgen -s 30 1 > $PROJECT_HOME/gen_query/s30-m/1.sql
./qgen -s 30 2 > $PROJECT_HOME/gen_query/s30-m/2.sql
./qgen -s 30 3 > $PROJECT_HOME/gen_query/s30-m/3.sql
./qgen -s 30 4 > $PROJECT_HOME/gen_query/s30-m/4.sql
./qgen -s 30 5 > $PROJECT_HOME/gen_query/s30-m/5.sql
./qgen -s 30 6 > $PROJECT_HOME/gen_query/s30-m/6.sql

./qgen -s 30 7 > $PROJECT_HOME/gen_query/s30-m/7.sql
./qgen -s 30 8 > $PROJECT_HOME/gen_query/s30-m/8.sql
./qgen -s 30 9 > $PROJECT_HOME/gen_query/s30-m/9.sql
./qgen -s 30 10 > $PROJECT_HOME/gen_query/s30-m/10.sql
./qgen -s 30 11 > $PROJECT_HOME/gen_query/s30-m/11.sql
./qgen -s 30 12 > $PROJECT_HOME/gen_query/s30-m/12.sql
./qgen -s 30 13 > $PROJECT_HOME/gen_query/s30-m/13.sql
./qgen -s 30 14 > $PROJECT_HOME/gen_query/s30-m/14.sql
./qgen -s 30 15 > $PROJECT_HOME/gen_query/s30-m/15.sql
./qgen -s 30 16 > $PROJECT_HOME/gen_query/s30-m/16.sql
./qgen -s 30 17 > $PROJECT_HOME/gen_query/s30-m/17.sql
./qgen -s 30 18 > $PROJECT_HOME/gen_query/s30-m/18.sql
./qgen -s 30 19 > $PROJECT_HOME/gen_query/s30-m/19.sql
./qgen -s 30 20 > $PROJECT_HOME/gen_query/s30-m/20.sql
./qgen -s 30 21 > $PROJECT_HOME/gen_query/s30-m/21.sql
./qgen -s 30 22 > $PROJECT_HOME/gen_query/s30-m/22.sql

```
1. Generate the explain queries
```

./qgen -s 30 -x 1 > $PROJECT_HOME/gen_query/s30-m/1_explain.sql
./qgen -s 30 -x 2 > $PROJECT_HOME/gen_query/s30-m/2_explain.sql
./qgen -s 30 -x 3 > $PROJECT_HOME/gen_query/s30-m/3_explain.sql
./qgen -s 30 -x 4 > $PROJECT_HOME/gen_query/s30-m/4_explain.sql
./qgen -s 30 -x 5 > $PROJECT_HOME/gen_query/s30-m/5_explain.sql
./qgen -s 30 -x 6 > $PROJECT_HOME/gen_query/s30-m/6_explain.sql
./qgen -s 30 -x 7 > $PROJECT_HOME/gen_query/s30-m/7_explain.sql
./qgen -s 30 -x 8 > $PROJECT_HOME/gen_query/s30-m/8_explain.sql
./qgen -s 30 -x 9 > $PROJECT_HOME/gen_query/s30-m/9_explain.sql
./qgen -s 30 -x 10 > $PROJECT_HOME/gen_query/s30-m/10_explain.sql
./qgen -s 30 -x 11 > $PROJECT_HOME/gen_query/s30-m/11_explain.sql
./qgen -s 30 -x 12 > $PROJECT_HOME/gen_query/s30-m/12_explain.sql
./qgen -s 30 -x 13 > $PROJECT_HOME/gen_query/s30-m/13_explain.sql
./qgen -s 30 -x 14 > $PROJECT_HOME/gen_query/s30-m/14_explain.sql
./qgen -s 30 -x 15 > $PROJECT_HOME/gen_query/s30-m/15_explain.sql
./qgen -s 30 -x 16 > $PROJECT_HOME/gen_query/s30-m/16_explain.sql
./qgen -s 30 -x 17 > $PROJECT_HOME/gen_query/s30-m/17_explain.sql
./qgen -s 30 -x 18 > $PROJECT_HOME/gen_query/s30-m/18_explain.sql
./qgen -s 30 -x 19 > $PROJECT_HOME/gen_query/s30-m/19_explain.sql
./qgen -s 30 -x 20 > $PROJECT_HOME/gen_query/s30-m/20_explain.sql
./qgen -s 30 -x 21 > $PROJECT_HOME/gen_query/s30-m/21_explain.sql
./qgen -s 30 -x 22 > $PROJECT_HOME/gen_query/s30-m/22_explain.sql

```
Now the generated queries for MariaDB/MySQL test are ready and are stored into
the folder `$PROJECT_HOME/gen_query/s30-m/` (-m is for MariaDB/MySQL).

Additional reorganization of directories is up to the user.

1. Create a directory for the generated workload
```

mkdir $PROJECT_HOME/gen_data/s30

```
1. Set the variable DSS_PATH to the folder with the generated table data. The generated workload for the
test will be generated there.
```

export DSS_PATH=$PROJECT_HOME/gen_data/s30/

```
1. Generate the table data

* NOTE: The example uses scale factor = `30`. If you want to change it, you should change the parameter
`-s`.
```

./dbgen -vfF -s 30

```
* Now the generated data load is stored into the folder set in `$DSS_PATH = $PROJECT_HOME/gen_data/`

For the purpose of this benchmark these steps have been performed for scale
factor 30 and are stored on facebook-maria1 in the following locations:

* `/benchmark/dbt3/gen_data/s30` – the data load for scale factor 30
* `/benchmark/dbt3/gen_query/s30-m` – generated queries for MariaDB/MySQL with scale factor 30
* `/benchmark/dbt3/gen_query/s30-p` – generated queries for PostgreSQL with scale factor 30

See [DBT3 example preparation time](#pg-long-running-tests-benchmarks-dbt3-example-preparation-time-
fbaf8d27) to see how long it would take you to
prepare the databases for the test.

###### Download [MySQL 5.5.x](https://dev.mysql.com/downloads/mysql/#downloads)

1. Download the tar.gz file into your project folder `$PROJECT_HOME/bin/` for example

1. Unzip the archive with the following command:
```

gunzip < mysql-5.5.x-linux2.6-x86_64.tar.gz |tar xf -

```
Now the server could be started with the following command:
```

$PROJECT_HOME/bin/mysql-5.5.x-linux2.6-x86_64/bin/mysqld_safe –datadir=some/data/dir &

```
###### Download [MySQL 5.6.x](https://dev.mysql.com/downloads/mysql/#downloads)

1. Download the tar.gz file into your project folder `$PROJECT_HOME/bin/` for example

1. Unzip the archive with the following command:
```

gunzip < mysql-5.6.x-m5-linux2.6-x86_64.tar.gz |tar xf -

```
Now the server could be started with the following command:
```

$PROJECT_HOME/bin/mysql-5.6.x-m5-linux2.6-x86_64/bin/mysqld_safe –datadir=some/data/dir &

```
###### Download and build [MariaDB 5.3](https://mariadb.com/docs/release-notes/community-server/old-
releases/5.3/changes-improvements-in-mariadb-5-3).x / [MariaDB 5.5](https://mariadb.com/docs/release-
notes/community-server/old-releases/5.5/changes-improvements-in-mariadb-5-5).x

**NOTE:** These steps are the same for [MariaDB 5.5](https://mariadb.com/docs/release-notes/community-
server/old-releases/5.5/changes-improvements-in-mariadb-5-5).x with properly replaced
version numbers

1. Download with Bazaar the [mariadb 5.3](https://mariadb.com/docs/release-notes/community-server/old-
releases/5.3/changes-improvements-in-mariadb-5-3) project
```

bzr branch lp:maria/5.3
mv 5.3/ mariadb-5.3

```
1. Build MariaDB
```

cd mariadb-5.3/
./BUILD/compile-amd64-max

```
1. Build a binary distribution tar.gz file
```

./scripts/make_binary_distribution

```
1. Move the generated tar.gz file and unzip it to $PROJECT_HOME/bin from where
 it will be used by the automation script
```

mv mariadb-5.3.x-beta-linux-x86_64.tar.gz $PROJECT_HOME/bin/
cd $PROJECT_HOME/bin/
tar -xf mariadb-5.3.x-beta-linux-x86_64.tar.gz

```
Now the server could be started with the following command:
```

$PROJECT_HOME/bin/mariadb-5.3.x-beta-linux-x86_64/bin/mysqld_safe –datadir=some/data/dir &

```
###### Prepare the databases for the benchmark

**NOTE:** These instructions are the same for MariaDB, MySQL 5.5.x and MySQL
5.6.x with changing only the database home folders, noted here as $DB_HOME (for
example for MySQL 5.5.x $DB_HOME is
`$PROJECT_HOME/bin/mysql-5.5.x-linux2.6-x86_64`). Also you can prepare InnoDB
storage engine test databases. Instructions for preparing PostgreSQL could be
found in the section for downloading, building and preparing PostgreSQL later
on this page.

1. Open the file `$PROJECT_HOME/mariadb-tools/dbt3_benchmark/dbt3_mysql/make-dbt3-db_innodb.sql` and edit
the values for the call of the sql commands that look like this one:
```

LOAD DATA INFILE ‘/some/path/to/gen_data/nation.tbl’ into table nation fields terminated by ‘|’;

```
* They all look the same but operate with different tables.

* Replace "/some/path/to/gen_data/" with the proper directory where the
 generated data load is stored. At the end the same command could look like
 this:
```

LOAD DATA INFILE '~/benchmark/dbt3/gen_data/s30/nation.tbl' into table nation fields terminated by '|';

```
1. Create an empty MySQL database into a folder that will be used for the benchmark
```

cd $DB_HOME
./scripts/mysql_install_db –defaults-file=$PROJECT_HOME/mariadb-tools/dbt3_benchmark/config/s30/load_mysql_myisam_my.cnf –basedir=$DB_HOME –datadir=$PROJECT_HOME/db_data/myisam-s30/

```
* NOTE: For InnoDB change the defaults-file to `load_mysql_innodb_my.cnf`.

1. Start the mysqld process`./bin/mysqld_safe --defaults-file=$PROJECT_HOME/mariadb-
tools/dbt3_benchmark/config/s30/load_mysql_myisam_my.cnf --tmpdir=$PROJECT_HOME/temp/ --
socket=$PROJECT_HOME/temp/mysql.sock --datadir=$PROJECT_HOME/db_data/myisam-s30/ &`

* NOTE: For InnoDB change the defaults-file to `load_mysql_innodb_my.cnf`.
 Also make sure that you have enough space in the directory set by the
 parameter `--tmpdir`, since loading the database could take a
 lot of temporary space.

1. Load the data into the database by executing the file `make-dbt3-db_pre_create_PK.sql` (for InnoDB) or
`make-dbt3-db_post_create_PK.sql` (for MyISAM)`./bin/mysql -u root -S $PROJECT_HOME/temp/mysql.sock <
$PROJECT_HOME/mariadb-tools/dbt3_benchmark/dbt3_mysql/make-dbt3-db_post_create_PK.sql`

* NOTE: For faster creation, it is recommended to
 use `make-dbt3-db_pre_create_PK.sql` for loading InnoDB
 and `make-dbt3-db_post_create_PK.sql` for loading MyISAM databases.

1. Shutdown the database server:
```

./bin/mysqladmin –user=root –socket=$PROJECT_HOME/temp/mysql.sock shutdown 0

```
Now you have a database loaded with scale 30. Its datadir is
`$PROJECT_HOME/db_data/myisam-s30/`

The same steps can be reproduced for different scale factors and for different
storage engines.

###### Download, build and prepare [PostgreSQL](https://www.postgresql.org/ftp/source/)

1. Go to [](https://www.postgresql.org/ftp/source/v9.1rc1/)

1. Download the file under the link postgresql-9.1rc1.tar.gz

1. Unzip the archive to your project folder
```

gunzip < postgresql-9.1rc1.tar.gz |tar xf -

```
1. Execute the following commands into the shell to install PostgreSQL:
```

mkdir $PROJECT_HOME/PostgreSQL_bin
cd $PROJECT_HOME/postgresql-9.1rc1
./configure –prefix=$PROJECT_HOME/bin/PostgreSQL_bin
make
make install

```
* NOTE: Configure script may not find the following libraries: readline
 and zlib. In that case you can run configure without these libraries by
 adding the following parameters to the command line:
 `--without-readline --without-zlib`

1. Prepare the database to test with:
```

mkdir $PROJECT_HOME/db_data/postgre_s30
cd $PROJECT_HOME/bin/PostgreSQL_bin
./bin/initdb -D $PROJECT_HOME/db_data/postgre_s30

```
1. Start the server:
```

./bin/postgres -D $PROJECT_HOME/db_data/postgre_s30 -p 54322 &

```
1. Load the dataload into the DB
```

./bin/createdb -O {YOUR_USERNAME} dbt3 -p 54322
./bin/psql -p 54322 -d dbt3 -f $PROJECT_HOME/mariadb-tools/dbt3_benchmark/dbt3_mysql/make-dbt3-db_pg.sql

```
* NOTE: Here under `{YOUR_USERNAME}` you should put the
 database owner.

1. Stop the server:
```

./bin/pg_ctl -D $PROJECT_HOME/db_data/postgre_s30/ -p 54322 stop

```
The steps for preparing the workload for the benchmark on facebook-maria1 are
already made for MariaDB, MySQL and PostgreSQL. Here are the directories for
the different DBMS, storage engines and scale factors that are prepared on
facebook-maria1:

* `~/benchmark/dbt3/db_data/myisam_s30`
— datadir for MariaDB/MySQL + MyISAM with scale
 factor 30

* `~/benchmark/dbt3/db_data/innodb_mariadb_s30`
— datadir for MariaDB + InnoDB with scale factor 30
 (TODO)

* `~/benchmark/dbt3/db_data/innodb_mysql_s30`
— datadir for
 MySQL + InnoDB with scale factor 30 (TODO)

* `~/benchmark/dbt3/db_data/postgre_s30`
— datadir for
 PostgreSQL with scale factor 30 (TODO)

###### Prepare the results database

The results of the benchmark will be stored in a separate database that will be
run by [MariaDB 5.3](https://mariadb.com/docs/release-notes/community-server/old-releases/5.3/changes-
improvements-in-mariadb-5-3).x.

**NOTE:** The results database will be a subject to change in future versions
of the DBT3 benchmarking project.

The database is created by the file
`$PROJECT_HOME/mariadb-tools/dbt3_benchmark/dbt3_mysql/make-results-db.sql`.
In that file you can find details about every table and column in the database.

To prepare the database for work follow these steps:

1. Go to [MariaDB 5.3](https://mariadb.com/docs/release-notes/community-server/old-releases/5.3/changes-
improvements-in-mariadb-5-3).x installation directory
```

cd $PROJECT_HOME/bin/mariadb-5.3.x-beta-linux-x86_64

```
1. Install the system database tables into the datadir for the results (for
example `$PROJECT_HOME/db_data/dbt3_results_db`)
```

./scripts/mysql_install_db –datadir=$PROJECT_HOME/db_data/dbt3_results_db

```
1. Start mysqld for results db
```

./bin/mysqld_safe –defaults-file=$PROJECT_HOME/mariadb-tools/dbt3_benchmark/config/results_mariadb_my.cnf
–port=12340 –socket=$PROJECT_HOME/temp/mysql_results.sock
–datadir=$PROJECT_HOME/db_data/dbt3_results_db/ &

```
1. Install the database
```

./bin/mysql -u root -P 12340 -S $PROJECT_HOME/temp/mysql_results.sock < $PROJECT_HOME/mariadb-
tools/dbt3_benchmark/dbt3_mysql/make-results-db.sql

```
1. Shutdown the results db server:
```

```
./bin/mysqladmin –user=root –port=12340 –socket=$PROJECT_HOME/temp/mysql_results.sock shutdown 0
```

```
###### Automation script

###### Configuring and running a benchmark

In order to run a benchmark, one should have:

* 5 configuration files:

  1. DBMS server configuration (see [#dbms-server-configuration](#dbms-server-configuration))
  1. Test configuration (see [#test-configuration](#test-configuration))
  1. Queries configuration (see [#queries-configuration](#queries-configuration))
  1. Results database configuration (see [#results-database-configuration](#results-database-
configuration))
  1. Top-level configuration file that combines all of the above (see
 [#top-level-configuration](#top-level-configuration))
* an automation script `launcher.pl` that could be found
 under `mariadb-tools/dbt3_benchmark/`
* startup parameters that should be passed to the automation script (see
 [#script-startup-parameters](#script-startup-parameters)).

Details about each of these is given in the following sections.

Each benchmark is configured by a set of configuration files. One can find
example (default) configuration files under the directory
'mariadb-tools/dbt3_benchmark/tests'. Each configuration file has an 'ini'
configuration syntax and is parsed by the perl automation script with the CPAN
module [Config::Auto](https://search.cpan.org/~simon/Config-Auto-0.03/Auto.pm)

###### Configuration keywords

Every configuration file could contain keywords that will be replaced by the
script with particular values. They are used for convenience when you want to
make your configuration files more common to the environment that you have
prepared for the benchmark. These keywords are:

* `$PROJECT_HOME` – used as the directory where the
 project ``mariadb-tools`' is located or as a base path for the whole project
 (e.g. "`DBMS_HOME = $PROJECT_HOME/bin/mariadb-5.3.x-beta-linux-x86_64`").
 It is replaced by the value set with the startup parameter 'project-home'
 passed to launcher.pl,

* `$DATADIR_HOME` – used as the directory where the
 datadir folders are located for the benchmark (e.g.
 "`$DATADIR_HOME/myisam-s30`"). It is replaced by the value set
 with the startup parameter 'datadir-home' passed to launcher.pl.

* `$QUERIES_HOME` – used as the directory where the
 queries are located (e.g. "`$QUERIES_HOME/s30-m`"
 – queries for MariaDB/MySQL for scale factor 30). It
 is replaced by the value set with the startup parameter 'queries-home'
 passed to launcher.pl.

* `$SCALE_FACTOR` – the scale factor that will be
 used. It is usually a part of the name of the datadir directory (e.g.
 "`$DATADIR_HOME/myisam-s$SCALE_FACTOR`"), the queries directory (e.g.
 "`$QUERIES_HOME/s$SCALE_FACTOR-m`") or the database configuration directory
 (e.g. `$PROJECT_HOME/mariadb-tools/dbt3_benchmark/config/s$SCALE_FACTOR`).
 It is replaced by the value set with the startup parameter 'scale-factor'
 passed to launcher.pl.

Note that if any of the configuration files contains such keyword, the
corresponding startup parameter passed to **launcher.pl** will become required.

###### Top-level configuration

A top-level configuration file provides paths to the **Test, DBMS, Queries**
and **Results database** configurations files
```

```
There are default configuration files in the directory
`mariadb-tools/dbt3_benchmark/tests/` and contain the following settings:

| Parameter | Description |
| --- | --- |
| RESULTS_DB_CONFIG | The configuration file for results DB settings |
| TEST_CONFIG | The configuration file for the test settings |
| QUERIES_CONFIG | The configuration file for the queries settings |
| DB_CONFIG | The configuration file for the DBMS server settings |

This file has the following format:
```

[common]
RESULTS_DB_CONFIG = $PROJECT_HOME/mariadb-tools/dbt3_benchmark/tests/results_db_conf/results_db.conf
TEST_CONFIG = $PROJECT_HOME/mariadb-tools/dbt3_benchmark/tests/test_conf/test_myisam.conf

[mariadb_5_3]
QUERIES_CONFIG = $PROJECT_HOME/mariadb-tools/dbt3_benchmark/tests/queries_conf/queries.conf
DB_CONFIG = $PROJECT_HOME/mariadb-tools/dbt3_benchmark/tests/db_conf/db_mariadb_5_3_myisam.conf

[mysql_5_5]
QUERIES_CONFIG = $PROJECT_HOME/mariadb-tools/dbt3_benchmark/tests/queries_conf/queries_mysql.conf
DB_CONFIG = $PROJECT_HOME/mariadb-tools/dbt3_benchmark/tests/db_conf/db_mysql_5_5_myisam.conf
…

NOTE: The settings `RESULTS_DB_CONFIG` and `TEST_CONFIG` should be set
under the `[common]` section. They are common for the whole test (although
some settings from `TEST_CONFIG` could be overridden in the
`QUERIES_CONFIG` file). All settings that combine `QUERIES_CONFIG` and
`DB_CONFIG` should be in a separate section (e.g. `[mariadb_5_3]`).

A test configuration is passed as an input parameter to the automation script
with the parameter
`--test=/path/to/some_test_configuration.conf` (see
#script-startup-parameters)

DBMS server configuration

These configuration files contain settings that describe the benchmarked DBMS.
They are usually contained into the folder
`mariadb-tools/dbt3_benchmark/tests/db_conf`.

Here is the list of parameters that could be set into this configuration file:

Parameter	Description
DBMS_HOME	Where the installation folder of MariaDB / MySQL / PostgreSQL is located.NOTE: The automation script uses "./bin/mysqld_safe" to start the mysqld process. So the versions of MariaDB and MySQL should be a "binary distribution" ones.
DBMS_USER	The database user that will be used.
CONFIG_FILE	The config file that mysqld or postgres will use when starting
SOCKET	The socket that will be used to start the server
PORT	The port that the server will be started on
HOST	The host where the server is located
DATADIR	Where the datadir for mysqld or postgres is located
TMPDIR	Where the temp tables will be created while sorting and grouping.
DBNAME	The database (schema) name where the benchmark tables are located.
KEYWORD	This text will be stored into the results database as a keyword. Also will be used as a name for a subfolder with results and statistics.
DBMS	Database Management System that will be used. Possible values: "MySQL", "MariaDB" and "PostgreSQL"
STORAGE_ENGINE	The storage engine that was used (MyISAM, InnoDB, etc.)
STARTUP_PARAMS	Any startup parameters that will be used while starting the mysqld process or postgres process. Same format as given on the command line.
GRAPH_HEADING	The heading of the graphic for that particular test.
MYSQL_SYSTEM_DIR	See "MYSQL_SYSTEM_DIR note", below.
READ_ONLY	If set to 1, mysqld process will be started with '--read-only' startup parameter
PRE_RUN_SQL	SQL commands that are run prior each query run
POST_RUN_SQL	SQL commands that are run after each query run
PRE_TEST_SQL	SQL commands that are run prior the whole test with that database settings
POST_TEST_SQL	SQL commands that are run after the whole test with that database settings

`MYSQL_SYSTEM_DIR` note:
This option is added for convenience when you want to save time and disk space
for generating databases for different DBMS (and different versions) and use a
single data directory for all of them. When running different versions of
MariaDB/MySQL over a single datadir, one should run
[mysql-upgrade](https://dev.mysql.com/doc/refman/5.5/en/mysql-upgrade.html) in
order to fix the system tables. So in one data directory, you could prepare the
following directories for different MariaDB/MySQL system directories:

* `mysql_mysql_5_5` — a copy of the system directory
 ''mysql`' upgraded by MySQL 5.5.x
* `mysql_mariadb_5_3` — a copy of the system
 directory ''mysql`' upgraded by [MariaDB 5.3](https://mariadb.com/docs/release-notes/community-server/old-releases/5.3/changes-improvements-in-mariadb-5-3).x
* `mysql_mariadb_5_5` — a copy of the system
 directory ''mysql`' upgraded by [MariaDB 5.5](https://mariadb.com/docs/release-notes/community-server/old-releases/5.5/changes-improvements-in-mariadb-5-5).x

If `MYSQL_SYSTEM_DIR` is set to one of these directories, the automation
script will unlink the current system directory 'mysql' and make a new symbolic

> link with that name to the one in the setting.
> Here is an example command that will be executed:

unlink /path/to/datadir/mysql
ln -s /path/to/value/in/MYSQL_SYSTEM_DIR/mysql_mariadb_5_3 /path/to/datadir/mysql

> **NOTE:** This approach is suitable for MyISAM tests.
>
> The configuration file looks like this:

[db_settings]
DBMS_HOME = $PROJECT_HOME/bin/mariadb-5.3.2-beta-linux-x86_64
DBMS_USER = root
…

```
Note that the section `[db_settings]` is required for the file to be properly
parsed by the automation script.

###### Test configuration

These configuration files contain settings describing the test. They are
usually contained into the folder
**mariadb-tools/dbt3_benchmark/tests/test_conf**.

Here is the list of parameters that could be set into this configuration file:

| Parameter | Description |
| --- | --- |
| QUERIES_AT_ONCE | If set to 1, then all the queries are executed sequentially without restarting the
server or clearing the caches between queries. |
| CLEAR_CACHES | If set to 1, the disk caches will be cleared before each query test. |
| WARMUP | Perform a warm-up runs before running the query. |
| EXPLAIN | Run an Explain command prior the run of the query. The explain results will be stored in a
file under the results output directory. |
| RUN | Perform the actual test |
| ANALYZE_EXPLAIN | A result extraction mechanism where only the best execution plan (results from
EXPLAIN select) will be measured. It is designed to be used when benchmarking InnoDB storage engine where
execution plan is changing between server restarts (see [#results-extraction-mechanisms](#results-
extraction-mechanisms)). |
| MIN_MAX_OUT_OF_N | A result extraction mechanism where the minimal and maximal values out of N (set by
the parameter NUM_TESTS) tests are taken as a result. This could be used when InnoDB storage engine is
tested (see [#results-extraction-mechanisms](#results-extraction-mechanisms)). |
| SIMPLE_AVERAGE | A result extraction mechanism where the final result is the average time taken for the
tests. The number of tests is per query is set by the NUM_TESTS parameter. Note that if even one test has
timed out, the result is 'time-out'. This is used when testing MyISAM storage engine since there the
execution plan is constant (see [#results-extraction-mechanisms](#results-extraction-mechanisms)). |
| NUM_TESTS | How many tests should be performed for each query. When ANALYZE_EXPLAIN is set, this value
could be set to 0, meaning that the tests will continue until enough results are extracted (see setting
CLUSTER_SIZE). This parameter is very important when MIN_MAX_OUT_OF_N or SIMPLE_AVERAGE is selected. |
| MAX_SKIPPED_TESTS | When ANALYZE_EXPLAIN is set and an execution plan that is slower is selected, the
execution of the query is skipped and the server is restarted in order to change the execution plan. If
the server is restarted more than MAX_SKIPPED_TESTS, there are obviously no more different execution
plans and the script continues to the next query benchmark. |
| WARMUPS_COUNT | How many warmup runs will be performed prior the actual benchmark run. |
| CLUSTER_SIZE | How big a cluster with results for a query should be in order to extract the final
result. It is used when ANALYZE_EXPLAIN is selected as a result extraction method. |
| MAX_QUERY_TIME | The maximum time that one query will be tested. Currently it is applicable only when
ANALYZE_EXPLAIN is selected. |
| TIMEOUT | The maximum time that one query could run. Currently timeout is applicable only for MySQL and
MariaDB. |
| OS_STATS_INTERVAL | What is the time interval between extraction of OS statistics for CPU, memory, etc.
|
| PRE_RUN_OS | OS commands that should be executed prior each query run |
| POST_RUN_OS | OS commands that should be executed after each query run |
| PRE_TEST_OS | OS commands that should be executed prior the whole test |
| POST_TEST_OS | OS commands that should be executed after the whole test is complete |

The configuration file looks like this:
```

QUERIES_AT_ONCE = 0
CLEAR_CACHES = 1
WARMUP = 0
…

```
###### Queries configuration

These configuration files contain the list of all the queries that will be
benchmarked against each database. Some settings
from `DBMS server configuration` and `Test configuration` could be
overridden into the `Queries configuration` files. The folder that contains
such configurations is `mariadb-tools/dbt3_benchmark/tests/queries_conf`.

Here is the list of parameters that could be set into this configuration file:

| Parameter | Description |
| --- | --- |
| QUERIES_HOME | Where the queries are located on disk. This value is concatenated to the QUERY setting
and this makes the path to the particular query. NOTE: This setting should be set under the section
[queries_settings]. |
| CONFIG_FILE | This overrides the startup setting CONFIG_FILE from DMBS server configuration file and
sets the database configuration file that is used. It could be used if some configuration file without
any optimizations should be set for this particular queries configuration file.NOTE: This setting should
be set under the section [queries_settings]. |
| QUERY | The name of the query located into QUERIES_HOME folder. E.g. "1.sql" |
| EXPLAIN_QUERY | The name of the explain query into QUERIES_HOME folder. E.g. "1_explain.sql" |
| TMPDIR | This overrides the setting TMPDIR from the DMBS server configuration. |
| STARTUP_PARAMS | This overrides the setting STARTUP_PARAMS from the DMBS server configuration. Using
this setting one could change the particular startup parameters (like optimizations and buffers) for the
DB server. |
| PRE_RUN_SQL | This overrides the setting PRE_RUN_SQL from the DMBS server configuration. |
| POST_RUN_SQL | This overrides the setting POST_RUN_SQL from the DMBS server configuration. |
| RUN | This overrides the setting RUN from the test configuration. |
| EXPLAIN | This overrides the setting EXPLAIN from the test configuration. |
| TIMEOUT | This overrides the setting TIMEOUT from the test configuration. |
| NUM_TESTS | This overrides the setting NUM_TESTS from the test configuration. |
| MAX_SKIPPED_TESTS | This overrides the setting MAX_SKIPPED_TESTS from the test configuration. |
| WARMUP | This overrides the setting WARMUP from the test configuration. |
| WARMUPS_COUNT | This overrides the setting WARMUPS_COUNT from the test configuration. |
| MAX_QUERY_TIME | This overrides the setting MAX_QUERY_TIME from the test configuration. |
| CLUSTER_SIZE | This overrides the setting CLUSTER_SIZE from the test configuration. |
| PRE_RUN_OS | This overrides the setting PRE_RUN_OS from the test configuration. |
| POST_RUN_OS | This overrides the setting POST_RUN_OS from the test configuration. |
| OS_STATS_INTERVAL | This overrides the setting OS_STATS_INTERVAL from the test configuration. |

The queries configuration file could look like this:
```

[queries_settings]
QUERIES_HOME = /path/to/queries

[query1]
QUERY=1.sql
EXPLAIN_QUERY=1_explain.sql
STARTUP_PARAMS=

[query2]
QUERY=2.sql
EXPLAIN_QUERY=2_explain.sql
STARTUP_PARAMS=–optimizer_switch=‘mrr=on’ –mrr_buffer_size=8M –some_startup_parmas
…

```
...where "`QUERIES_HOME = /path/to/queries`" could be replaced with
"`QUERIES_HOME = $QUERIES_HOME/s$SCALE_FACTOR-m`" for example and thus
`$QUERIES_HOME` and $SCALE_FACTOR will be replaced by the script startup
parameters passed to `launcher.pl` (see [#script-startup-parameters](#script-startup-parameters))

**NOTE:** The section `[queries_settings]` is required for the configuration
file to be parsed correctly. Also each query settings should be set under an
uniquely named configuration section (e.g. `[query1]` or `[1.sql]`)

###### Results database configuration

These configuration files contain settings about the database where the results
will be stored. They are usually contained into the folder
`mariadb-tools/dbt3_benchmark/tests/results_db_conf`.

Here is the list of parameters that could be set into this configuration file:

| Parameter | Description |
| --- | --- |
| DBMS_HOME | Where the database directory is located. E.g. "$PROJECT_HOME/mariadb-5.3.x-beta-linux-
x86_64". This should be a binary distribution of MariaDB or MySQL. |
| DBMS_USER | The user that will be used by the DBMS |
| DATADIR | Where the data directory is located for the results database |
| CONFIG_FILE | What the configuration file used by the database is. |
| SOCKET | The socket that will be used by the results database. This should be different socket than the
one provided for the testing databases. |
| PORT | The port that the results database will use. This should be different port than the one provided
for the testing databases. |
| STARTUP_PARAMS | Any startup parameters that should be set to start the server. |
| DBNAME | The database name to use. |
| HOST | The host where the results database is. |

The results database configuration could look like this:
```

DBMS_HOME = $PROJECT_HOME/mariadb-5.3.x-beta-linux-x86_64
DBMS_USER = root
...

```
###### Script startup parameters

`launcher.pl` could accept startup parameters called in the manner
"`--some-param`". Note that these startup parameters are
case-sensitive. The ones that are with upper-case are used when overriding a
setting in some of the configuration files.

Here is a list of the startup parameters:

| Parameter | Description |
| --- | --- |
| test | The top-level benchmark configuration file that will be run. This is a required startup
parameter. |
| results-output-dir | Where the results of the benchmark will be stored. A timestamp is automatically
attached to the directory name so that it keeps track of time and date of the benchmark. This is a
required parameter. |
| dry-run | If set, no benchmark will be performed. Instead only messages will be displayed for the
actions that were intended to be done. |
| project-home | Required if any configuration file uses the variable '$PROJECT_HOME'. If all
configuration files use absolute paths, not used. |
| datadir-home | The value in this parameter will replace any occurrences of the string '$DATADIR_HOME'
into the configuration files. If there are no such occurrences, it is not a required parameter. |
| queries-home | The value in this parameter will replace any occurrences of the string '$QUERIES_HOME'
into the configuration files. If there are no such occurrences, it is not a required parameter. |
| scale-factor | The value in this parameter will replace any occurrences of the string '$SCALE_FACTOR'
into the configuration files. If there are no such occurrences, it is not a required parameter. |
| CLEAR_CACHES | If set. this overrides the default setting set into the test configuration file. |
| QUERIES_AT_ONCE | If set. this overrides the default setting set into the test configuration file. |
| RUN | If set. this overrides the default setting set into the test configuration file. |
| EXPLAIN | If set. this overrides the default setting set into the test configuration file. |
| TIMEOUT | If set. this overrides the default setting set into the test configuration file. |
| NUM_TESTS | If set. this overrides the default setting set into the test configuration file. |
| MAX_SKIPPED_TESTS | If set. this overrides the default setting set into the test configuration file. |
| WARMUP | If set. this overrides the default setting set into the test configuration file. |
| WARMUPS_COUNT | If set. this overrides the default setting set into the test configuration file. |
| MAX_QUERY_TIME | If set. this overrides the default setting set into the test configuration file. |
| CLUSTER_SIZE | If set. this overrides the default setting set into the test configuration file. |
| PRE_RUN_OS | If set. this overrides the default setting set into the test configuration file. |
| POST_RUN_OS | If set. this overrides the default setting set into the test configuration file. |
| OS_STATS_INTERVAL | If set. this overrides the default setting set into the test configuration file. |

###### Results extraction mechanisms

There are three possible result extraction mechanisms. They are set by the
parameters set into the test configuration file:

* `ANALYZE_EXPLAIN`
* `MIN_MAX_OUT_OF_N`
* `SIMPLE_AVERAGE`

Only one of these should be set to true (1).

**`ANALYZE_EXPLAIN`** is used for benchmarking InnoDB storage engine where the
execution plan could change for the same query when the server is restarted. It
is designed to run the query only with the fastest execution plan. This means
that the server is restarted if the current execution plan is proven slower
than the other. As a final result is taken the result for the query plan that
turns out to be fastest and there are at least `CLUSTER_SIZE` tests with it for
that query. By setting the configuration parameter `NUM_TESTS` you can set a
maximum test runs that when reached will get the best cluster's average time
(even if it is less than `CLUSTER_SIZE`). Also when a timeout for that query
(`MAX_QUERY_TIME`) is reached, the scoring mechanism will return the best
available cluster result.

**`MIN_MAX_OUT_OF_N`** is also used for benchmarking InnoDB storage engine.
As a result are stored the values for the fastest and the slowest run. It is
assumed that when the execution plan has changed it has different execution
```

```
plan and we are interested only in the min and max time.

**`SIMPLE_AVERAGE`** is used for benchmarking storage engines that do not
change the execution plan between restarts like MyISAM. The final result is the
average execution time from all the test runs for the query.

###### Results graphics

After each query test run, the result is stored into a file named
**results.dat** located into `{RESULTS_OUTPUT_DIR}`. This file
is designed to be easy to be read by the plotting program Gnuplot 4.4. It is
divided into blocks, separated by several new lines. Each block starts with a
comment line containing details for the current block of results.

Queries that have timed out have a value of `100000` so that they run out of
the graphics and are cut off. Other queries have their real times (in seconds)
starting from `0`. The graphics is cut off on the y-axis on the longest time
for `completed test + 20%`. For example if the longest time is `100`
seconds, the graphics is cut-off to `120` seconds. Thus the timed out queries
will be truncated by this limitation and will seem as really timed out.

During the test run, a gnuplot script file is generated with the necessary
parameters for the graphics to be generated automatically. After each query
test run is complete, the graphic is regenerated, so that the user can see the
current results before the whole benchmark is complete. This file is called
`gnuplot_script.txt` and is located into `{RESULTS_OUTPUT_DIR}`. The user
can edit it to fine-tune the parameters or headings after the test is complete
so that one could get the look and feel he/she wants for the final result.

###### Script output

###### Benchmark output

In the directory set by the parameter {RESULTS_OUTPUT_DIR} (example:
`/benchmark/dbt3/results/myisam_test_2011-12-08_191427/`) there are the
following files/directories:

* A directory for each test, named as the parameter {KEYWORD} from the test
 configuration (example: `mariadb-5-3-2`)
* cpu_info.txt — the output
 of "`/bin/cat /proc/cpuinfo`" OS command
* uname.txt — the output of "`uname -a`" OS
 command
* results.dat — the results of each query
 execution in one file. This file will be used as a datafile for the gnuplot
 script. It also contains the ratio between the current test and the first
 one.
* gnuplot_script.txt — the Gnuplot script that
 renders the graphics.
* graphics.jpeg — the output graphics
* A benchmark configuration file
 (example: `myisam_test_mariadb_5_3_mysql_5_5_mysql_5_6.conf`) copied
 from `mariadb-tools/dbt3_benchmark/tests/`
* A results database configuration file (example: `results_db.conf`)
 copied from `mariadb-tools/dbt3_benchmark/tests/results_db_conf/`
* A test configuration file (example: `test_myisam.conf`) copied
 from `mariadb-tools/dbt3_benchmark/tests/test_conf/`

###### Test output

In the subdirectory for the particular test, set by the parameter {KEYWORD}
(example:
`/benchmark/dbt3/results/myisam_test_2011-12-08_191427/mariadb-5-3-2/`),
there are the following files:

* pre_test_os_resutls.txt - the output of the OS commands (if any) executed
 before the first query run for that test
* pre_test_sql_resutls.txt - the output of the SQL commands (if any)
 executed before the first query run for that test
```

```
* post_test_os_resutls.txt - the output of the OS commands (if any)
 executed after the last query run for that test
* post_test_sql_resutls.txt - the output of the SQL commands (if any)
 executed after the last query run for that test
* all_explains.txt - a file containing all the explain queries, their
 startup parameters for the benchmark and the explain result
* The config file (my.cnf) that was passed to mysqld or postgres
 (example: `mariadb_myisam_my.cnf`) copied
 from `mariadb-tools/dbt3_benchmark/config/s$SCALE_FACTOR/`
* The queries configuration file (example: ''queries-mariadb.conf'') copied
 from `mariadb-tools/dbt3_benchmark/tests/queries_conf/`
* The database configuration file (example: ''db_mariadb_5_3_myisam.conf'')
 copied from `mariadb-tools/dbt3_benchmark/tests/db_conf/`

###### Query output

For each query execution there are several files that are outputted by the
automation script. They are all saved under the subdirectory set by the
parameters `{KEYWORD}`:

* Explain result - a file named
 '{query_name}_{number_of_query_run}_results.txt' (example:
 '1_explain.sql_1_results.txt' — first test for 1_explain.sql)
* Pre-run OS commands - OS commands, executed before the actual query run.
 Output is a file named
 'pre_run_os_q_{query_name}_no_{number_of_query_run}_results.txt' (example:
 'pre_run_os_q_1.sql_no_2_results.txt' — second test for query 1.sql)
* Pre-run SQL commands - SQL commands executed before the actual query run.
 Output is a file named
 'pre_run_sql_q_{query_name}_no_{number_of_query_run}_results.txt'.
* Post-run OS commands - OS commands, executed after the actual query run.
 Output is a file named
 'post_run_os_q_{query_name}_no_{number_of_query_run}_results.txt'.
* Post-run SQL commands - SQL commands executed after the actual query run.
 Output is a file named
 'post_run_sql_q_{query_name}_no_{number_of_query_run}_results.txt'.
* CPU utilization statistics:
 '{query_name}_no_{number_of_query_run}_sar_u.txt'
* I/O and transfer rate statistics:
 '{query_name}_no_{number_of_query_run}_sar_b.txt'
* Memory utilization statistics:
 '{query_name}_no_{number_of_query_run}_sar_r.txt'

###### Hooks

The automation script provides hooks that allow the user to add both SQL and OS commands prior and after
each test. Here is a list of all possible hooks:

* Pre-test SQL hook: it is set with the parameter `PRE_TEST_SQL`. Contains
 SQL commands that are run once for the whole test configuration before the
 first run.
 (Example: "`use dbt3; select version(); show variables; show engines; show table status;`")
* Post-test SQL hook: it is set with the parameter `POST_TEST_SQL`. Contains
 SQL commands that are run once for the whole test configuration after the
 last run.
* Pre-test OS hook: it is set with the parameter `PRE_TEST_OS`. Contains OS
 commands that are run once for the whole test configuration before the first
 run.
* Post-test OS hook: it is set with the parameter `POST_TEST_OS`. Contains OS
 commands that are run once for the whole test configuration after the last
 run.
* Pre-run SQL hook: it is set with the parameter `PRE_RUN_SQL`. Contains SQL
 commands that are run prior each query run.
 (Example: "`flush status; set global userstat=on;`")
* Post-run SQL hook: it is set with the parameter `POST_RUN_SQL`. Contains SQL
 commands that are run after each query run.
 (Example: "`show status; select * from information_schema.TABLE_STATISTICS;`")
* Pre-run OS hook: it is set with the parameter `PRE_RUN_OS`. Contains OS
 commands that are run once prior each query run.
```

```
* Post-run OS hook: it is set with the parameter `POST_RUN_OS`. Contains OS
 commands that are run once after each query run.

The results of these commands is stored into the
`{RESULTS_OUTPUT_DIR}/{KEYWORD}` folder (see [#script-output](#script-output))

###### Activities

Here are the main activities that this script does:

1. Parse the configuration files and check the input parameters - if any of the
 required parameters is missing, the script will stop resulting an error.
1. Collect hardware information - collecting information about the hardware of
 the machine that the benchmark is run. Currently it collects `cpuinfo`
 and `uname`. Results of these commands are stored into the results output
 directory set as an input parameter
1. Loop through the passed test configurationsFor each passed in test
 configuration the script does the following:

  1. Start the results database server. The results of the test are stored into
 that database.
  1. Clears the caches on the serverClearing the caches is done with the
 following
 command:
```

sudo /sbin/sysctl vm.drop_caches=3

```
    * NOTE: In order to clear the caches, the user is required to have
 sudo rights and the following line should be added to the `sudoers` file
 (edit it with "`sudo vusudo`"
 command):
```

{your_username} ALL=NOPASSWD:/sbin/sysctl

```
 1. Start the database server
 1. Perform pre-test SQL commands. The results are stored
under `results_output_dir/{KEYWORD}` folder and are
called `pre_test_sql_results.txt`. `{KEYWORD}` is a unique keyword for
the current database configuration.
 1. Perform pre-test OS commands. The results are stored
under `results_output_dir/{KEYWORD}` folder and are
called `pre_test_os_results.txt`.

   * NOTE: If in the test configuration the setting QUERIES_AT_ONCE is
set to 0, then the server is restarted between each query run. Thus the
steps 3.2, 3.3, 3.4 and 3.5 are executed only once right before step 3.6.2.
 1. Read all query configurations and execute the following for each of them:

   1. Check the test configuration parameters. If something is wrong with some
required parameter, the program will exit resulting an error.
   1. Get the server version if that's the first run of the query
   1. Perform pre-run OS commands in shell. The results of these queries are
stored into a file named `pre_run_os_q_{QUERY}_no_{RUN_NO}_results.txt`
under `results_output_dir/{KEYWORD}` where `{QUERY}` is the query name
(ex: `1.sql`), `{RUN_NO}` is the sequential run number
and `{KEYWORD}` is a unique keyword for the particular test
configuration.
   1. Perform pre-run SQL queries. The results of these queries are stored into a
file named `pre_run_sql_q_{QUERY}_no_{RUN_NO}_results.txt`
under `results_output_dir/{KEYWORD}` where `{QUERY}` is the query name
(ex: `1.sql`), `{RUN_NO}` is the sequential run number and `{KEYWORD}` is
a unique keyword for the particular test configuration.
   1. Perform warm-up runs if set into the test configuration file
   1. Perform actual test run and measure time.

     * During this step, a new child process is created in order to measure
the statistics of the OS. Currently the statistics being collected are:

       * CPU utilization statistics. The command for this is:
```

sar -u 0 2>null

```
       * I/O and transfer rate statistics. The command for this is:
```

sar -b 0 2>null

```
       * Memory utilization statistics. The command for this is:
```

sar -r 0 2>null

```
     * These statistics are measured every N seconds, where `N` is set with
 the `OS_STATS_INTERVAL` test configuration parameter.
     * The test run for MariaDB and MySQL has an implemented mechanism for
 cut-off when timeout exceeds. It is controlled with the `TIMEOUT` test
 parameter. Currently for PostgreSQL there is no such functionality and
 should be implemented in future versions.
    1. Execute the "explain" statement for that query.

     * NOTE: Running `EXPLAIN` queries with MySQL prior version 5.6.3 could
 result in long running queries since MySQL has to execute the whole query
 when there are nested selects in it. For MariaDB and PostgreSQL there is
 no such problem. The long-running explain queries are for queries #7, 8,
 9, 13 and 15. For that reason in MySQL prior version 5.6.3 for these
 queries no `EXPLAIN` selects should be executed.
    1. Perform post-run SQL queries
    1. Perform post-run OS commands in shell
    1. Log the results into the results database
    1. A graphics with the current results is generated using Gnuplot
  1. Shutdown the database server.
  1. Perform post-test SQL commands. The results are stored
 under `results_output_dir/{KEYWORD}` folder and are
 called `post_test_sql_results.txt`.
  1. Perform post-test OS commands. The results are stored
 under `results_output_dir/{KEYWORD}` folder and are
 called `post_test_os_results.txt`.
  1. Stop the results database server

###### Script calling examples

* Example call for MyISAM test for scale factor 30 and timeout 10 minutes:
```

perl launcher.pl
--project-home=/path/to/project/home/
--results-output-dir=/path/to/project/home/results/myisam_test
--datadir=/path/to/project/home/db_data/
--test=/path/to/project/home/mariadb-tools/dbt3_benchmark/tests/myisam_test_mariadb_5_3_mysql_5_5_mysql_5_6.conf
--queries-home=/path/to/project/home/gen_query/
--scale-factor=30
--TIMEOUT=600

```
...where `/path/to/project/home` is where the `mariadb-tools` project is
located. This will replace all occurrences of the string "$PROJECT_HOME" in the
configuration files (example: "`TMPDIR = $PROJECT_HOME/temp/`" will become
"`TMPDIR = /path/to/project/home/temp/`").

`--TIMEOUT` overrides the timeout setting into the test
configuration file to 10 minutes.

* Example for InnoDB test for scale factor 30 with 2 hours timeout per query
 and 3 runs for each query:
```

perl launcher.pl
--project-home=/path/to/project/home/
--results-output-dir=/path/to/project/home/results/innodb_test
--datadir=/path/to/project/home/db_data/
--test=/path/to/project/home/mariadb-tools/dbt3_benchmark/tests/innodb_test_mariadb_5_3_mysql_5_5_mysql_5_6.conf
--queries-home=/path/to/project/home/gen_query/
--scale-factor=30

--TIMEOUT=7200
--NUM_TESTS=3

```
* If a newer version of [MariaDB 5.5](https://mariadb.com/docs/release-notes/community-server/old-
releases/5.5/changes-improvements-in-mariadb-5-5) is available:

  * copy or edit the DMBS server configuration
 file `mariadb-tools/dbt3_benchmark/tests/db_conf/db_mariadb_5_5_myisam.conf`
 and change the parameter `DBMS_HOME` to the new binary distribution. You
 can also edit `KEYWORD` and `GRAPH_HEADING`

* If you want to add additional test in the MyISAM benchmark for [MariaDB 5.3]
(https://mariadb.com/docs/release-notes/community-server/old-releases/5.3/changes-improvements-in-
mariadb-5-3),
 but with another defaults-file (my.cnf):

  * copy or edit the DMBS server configuration
 file `mariadb-tools/dbt3_benchmark/tests/db_conf/db_mariadb_5_3_myisam.conf`
 and change the parameter CONFIG_FILE to the new my.cnf
  * copy or edit the test configuration
 file `mariadb-tools/dbt3_benchmark/tests/myisam_test_mariadb_5_3_mysql_5_5_mysql_5_6.conf`
 and add the new configuration settings:
```

[mariadb_5_3_new_configuration]
QUERIES_CONFIG = $PROJECT_HOME/mariadb-tools/dbt3_benchmark/tests/queries_conf/queries-mariadb.conf
DB_CONFIG = $PROJECT_HOME/mariadb-tools/dbt3_benchmark/tests/db_conf/db_mariadb_5_3_myisam_new_configuration.conf

```
* If you want to add additional startup parameters for query 6 for MariaDB for
 example:

  * copy or edit the
 file `mariadb-tools/dbt3_benchmark/tests/queries_conf/queries-mariadb.conf`
 and add a parameter
 "`STARTUP_PARAMS=--optimizer_switch='mrr=on' --mrr_buffer_size=96M`"
 for example for the section for query 6.
  * copy or edit the test configuration
 file `mariadb-tools/dbt3_benchmark/tests/myisam_test_mariadb_5_3_mysql_5_5_mysql_5_6.conf`
 to include the new queries configuration file

###### Results

###### MyISAM test

DBT3 benchmark for the following configuration:

* [MariaDB 5.3.2](https://mariadb.com/docs/release-notes/community-server/old-releases/5.3/5.3.2) Beta +
MyISAM
* [MariaDB 5.5.18](https://mariadb.com/kb/en/mariadb-5518-release-notes/) + MyISAM
* MySQL 5.5.19 + MyISAM
* MySQL 5.6.4 + MyISAM

Results page: [DBT3 benchmark results MyISAM](#pg-long-running-tests-benchmarks-dbt3-benchmark-results-
myisam-5ade69bd)

###### InnoDB test

DBT3 benchmark for the following configuration:

* [MariaDB 5.3.2](https://mariadb.com/docs/release-notes/community-server/old-releases/5.3/5.3.2) Beta +
XtraDB
* MariaDB 5.5.18 + XtraDB
* MySQL 5.5.19 + InnoDB
* MySQL 5.6.4 + InnoDB

Results page: [DBT3 benchmark results InnoDB](#pg-long-running-tests-benchmarks-dbt3-benchmark-results-
innodb-42ee177d)

###### PostgreSQL test

DBT3 benchmark for the following configuration:

* [MariaDB 5.3.2](https://mariadb.com/docs/release-notes/community-server/old-releases/5.3/5.3.2) Beta +
XtraDB
* MySQL 5.6.4 + InnoDB
* PostgreSQL

Results page: (TODO)

<sub>_This page is licensed: CC BY-SA / Gnu FDL_</sub>
```

DBT3 Benchmark Results InnoDB

Introduction

This page shows the results for benchmarking the following configuration:

- MariaDB 5.3.1 Beta + XtraDB with all optimizations (optimizater_switch) set to `ON`
- MariaDB 5.3.1 Beta + XtraDB with Igor's suggested optimizations
- MySQL 5.5.13 + InnoDB
- MySQL 5.6.2 + InnoDB

The test is performed using the automation script under `/mariadb-tools/dbt3_benchmark`.

Details about this automation script could be found in DBT3 automation scripts

Hardware

The tests were performed on pitbull.askmonty.org. It has the following parameters:

- CPU: Two socket X hexacore Intel Xeon X5660 = 12 CPUs with hyperthreading on: 24 virtual CPUs
- Memory: 23GB
- Logical disk: HDD 500.1 GB as software RAID1
  - device size with M = 1000*1000: 500107 MBytes (500 GB)
  - cache/buffer size = 16384 KBytes
  - Nominal Media Rotation Rate: 7200
- Operating System: Ubuntu 10.10 (x86_64)

Scale factor 10

This test is performed with the following parameters:

- Scale factor: 10
- Query timeout: 600 sec.
- Cluster size: 3
- Total DB size on disk: about 24GB

Steps to reproduce

Follow the instructions in DBT3 automation scripts to prepare the environment for the test.

Before you run the test, reassure that the settings into the test configuration files match your prepared environment. For more details on the test configuration, please, refer to the test configuration parameters.

For that test you should set the SCALE_FACTOR parameter to 10 for the following files before the test:

- mariadb_innodb_igor_s1.pm
- mariadb_innodb_s1.pm
- mysql_5_5_15_innodb_s1.pm
- mysql_5_6_2_innodb_s1.pm

NOTE: In future versions the scale factor will be passed in as an input parameter to `launcher.pl` instead of being test configuration parameter.

After the environment is prepared, the following command should be executed in the shell:

```
perl launcher.pl --test={PROJECT_HOME}/mariadb-tools/dbt3_benchmark/tests/mariadb_myisam_s1.pm \
 --test={PROJECT_HOME}/mariadb-tools/dbt3_benchmark/tests/mysql_5_5_15_myisam_s1.pm \
 --test={PROJECT_HOME}/mariadb-tools/dbt3_benchmark/tests/mysql_5_6_2_myisam_s1.pm
 --results-output-dir=path/to/results/output/dir
```

Results

Here is the graphics of the results:

NOTE: Queries that are cut off by the graphics have timed out the period of
600 seconds for that test.

Here are the actual results:

Configuration	1.sql	2.sql	3.sql	4.sql	5.sql	6.sql	7.sql	8.sql	9.sql	10.sql	11.sql	12.sql	13.sql	14.sql	15.sql	16.sql	17.sql	18.sql	19.sql	20.sql	21.sql	22.sql
MariaDB 5.3.1 + XtraDB with all optimizations to ON	165	n/a	4243.333	n/a	n/a	114.333	n/a	n/a	n/a	n/a	536	173	n/a	n/a	n/a	52	452	n/a	n/a	n/a	n/a	8
MariaDB 5.3.1 + XtraDB with Igor's suggestions for optimization	163.667	n/a	n/a	n/a	n/a	114	n/a	n/a	n/a	n/a	538	280.667	n/a	257	n/a	60	456	n/a	n/a	n/a	n/a	8
MySQL 5.5.15 + InnoDB	104	n/a	n/a	n/a	n/a	103	n/a	n/a	n/a	n/a	534.667	177	n/a	n/a	n/a	n/a	476	n/a	n/a	n/a	n/a	6
MySQL 5.6.2 + InnoDB	103	n/a	n/a	n/a	n/a	104	n/a	n/a	n/a	n/a	531	168	n/a	n/a	n/a	55	460.667	n/a	n/a	n/a	n/a	6

The archived folder with all the results and details for that benchmark can be downloaded from:Image:Res_myisam_timeout_120_s10_2011-09-15_190613.zip

Comments

From the graphics we can see that for the first query MySQL has performed better than MariaDB with about 37%.

For the third query on the other hand MariaDB with all optimizations set to ON is the only one query that returned results before the timeout exceeded. This means that it has at least 30% better performance. Also there is some option that could optimize Igor's set of settings even more for that query. For the particular numbers, the same test should be performed with longer timeouts.

For query #6 it turns out that both MySQL 5.5.15 and 5.6.2 are faster than MariaDB with about 10%.

For query #12 Igor's settings could be readjusted, so that the query execution time could fall with 38%.

Igor's settings turned out to be the only one that could finish query #14 before the timeout exceeded.

From query #16 we can see that MySQL have made a great performance improvement from version 5.5.15 to 5.6.2 to make the query finish at least ten times faster.

For all the other queries the results are either statistically the same, or the queries have timed out for all configurations and the test should be repeated with longer timeout limit.

Summary

Most of the queries have timed out for the given period of 10 minutes per query and until a new test with longer timeout is performed, no correct comparison summary could be made.

This page is licensed: CC BY-SA / Gnu FDL

DBT3 Benchmark Results MyISAM

Introduction

This page shows the results for benchmarking the following configuration:

- MariaDB 5.3.2 + MyISAM
- MariaDB 5.5.18 + MyISAM

- MySQL 5.5.19 + MyISAM
- MySQL 5.6.4 + MyISAM

The test is performed using the automation script `/mariadb-tools/dbt3_benchmark/launcher.pl`.

Details about this automation script can be found on the DBT3 automation scripts page.

Hardware

The tests were performed on our `facebook-maria1` machine. It has the following parameters:

- CPU: 16 Intel® Xeon® CPU L5520 @ 2.27GHz
- Memory: Limited to 16 GB out of 72 by adding 'mem=16G' parameter to /boot/grub/menu.lst
- Logical disk: HDD 2 TB
- Operating system:

Scale factor 30

This test was performed with the following parameters:

- Scale factor: 30
- Query timeout: 2 hours
- Number of tests per query: 1
- Total DB size on disk: about 50GB
- Available memory: 16 GB

NOTE: The available memory is controlled by a parameter `mem=16G` added to the file `/boot/grub/menu.lst`

Steps to reproduce

Follow the instructions in DBT3 automation scripts to prepare the environment for the test.

Before you run the test, ensure that the settings in the test configuration files match your prepared environment. For more details on the test configuration, please, refer to the Test configuration parameters.

After the environment is prepared, the following command should be executed in the shell:

```
perl launcher.pl \
--results-output-dir=/home/mariadb/benchmark/dbt3/results/myisam_test \
--project-home=/home/mariadb/benchmark/dbt3/ \
--datadir=/home/mariadb/benchmark/dbt3/db_data/ \
--test=./tests/myisam_test_mariadb_5_3_mysql_5_5_mysql_5_6.conf \
--queries-home=/home/mariadb/benchmark/dbt3/gen_query/ --scale-factor=30 \
--TIMEOUT=7200
```

Compared configurations

The following configurations have been compared in this test:

Case 1: MariaDB 5.3.2 + MyISAM

Here are the common options that the mysqld server was started with:

```
net_read_timeout = 300
net_write_timeout = 600
key_buffer_size = 3G
skip-external-locking
key_buffer = 16M
max_allowed_packet = 16M
table_open_cache = 1024
thread_cache = 512
sort_buffer_size = 512K
net_buffer_length = 8K
read_buffer_size = 256K
read_rnd_buffer_size = 512K
myisam_sort_buffer_size = 8M
max_connections = 256
query_cache_size = 0
query_cache_type = 0
sql-mode = NO_ENGINE_SUBSTITUTION

#Per-test optimizations
optimizer_switch='index_merge=on'
optimizer_switch='index_merge_union=on'
optimizer_switch='index_merge_sort_union=on'
optimizer_switch='index_merge_intersection=on'
optimizer_switch='index_merge_sort_intersection=off'
optimizer_switch='index_condition_pushdown=on'
optimizer_switch='derived_merge=on'
optimizer_switch='derived_with_keys=on'
optimizer_switch='firstmatch=off'
optimizer_switch='loosescan=off'
optimizer_switch='materialization=on'
optimizer_switch='in_to_exists=on'
optimizer_switch='semijoin=on'
optimizer_switch='partial_match_rowid_merge=on'
optimizer_switch='partial_match_table_scan=on'
optimizer_switch='subquery_cache=off'
optimizer_switch='mrr=on'
optimizer_switch='mrr_cost_based=off'
optimizer_switch='mrr_sort_keys=on'
optimizer_switch='outer_join_with_cache=on'
optimizer_switch='semijoin_with_cache=off'
optimizer_switch='join_cache_incremental=on'
optimizer_switch='join_cache_hashed=on'
optimizer_switch='join_cache_bka=on'
optimizer_switch='optimize_join_buffer_size=on'
optimizer_switch='table_elimination=on'

join_buffer_space_limit = 3072M
join_buffer_size = 1536M
join_cache_level = 6
mrr_buffer_size = 96M
tmp_table_size = 96M
max_heap_table_size = 96M
```

Case 2: MariaDB 5.5.18 + MyISAM

Uses the same configuration file as MariaDB 5.3.2 in Case 1.

Case 3: MySQL 5.5.19 + MyISAM

Here are the common options that the mysqld server was started with:

```
net_read_timeout = 300
net_write_timeout = 600
key_buffer_size = 3G
skip-external-locking
key_buffer = 16M
max_allowed_packet = 16M
table_open_cache = 1024
thread_cache = 512
sort_buffer_size = 512K
net_buffer_length = 8K
read_buffer_size = 256K
myisam_sort_buffer_size = 8M
max_connections = 256
query_cache_size = 0
query_cache_type = 0
sql-mode = NO_ENGINE_SUBSTITUTION

join_buffer_size = 1536M
tmp_table_size = 96M
max_heap_table_size = 96M
read_rnd_buffer_size = 96M
```

Case 4: MySQL 5.6.4 + MyISAM

Here are the common options that the mysqld server was started with:

```
net_read_timeout = 300
net_write_timeout = 600
key_buffer_size = 3G
skip-external-locking
key_buffer = 16M
max_allowed_packet = 16M
table_open_cache = 1024
thread_cache = 512
sort_buffer_size = 512K
net_buffer_length = 8K
read_buffer_size = 256K
myisam_sort_buffer_size = 8M
max_connections = 256
query_cache_size = 0
query_cache_type = 0
sql-mode = NO_ENGINE_SUBSTITUTION

optimizer_switch='mrr=on'
optimizer_switch='mrr_cost_based=off'
optimizer_switch='batched_key_access=on'
optimizer_switch='index_condition_pushdown=on'

join_buffer_size = 1536M
tmp_table_size = 96M
max_heap_table_size = 96M
read_rnd_buffer_size = 96M
```

The server has been restarted between each query run and the caches have been cleared between each query run.

Results (without q20)

Here is the graphics of the results:

(Smaller bars are better)

NOTE: Queries that are cut off by the graphics have timed out the period of 2 hours.

Here are the actual results in seconds (smaller is better):

Configuration	1.sql	2.sql	2-opt.sql	3.sql	4.sql	5.sql	6.sql	7.sql	8.sql	9.sql	10.sql	11.sql	12.sql	13.sql	14.sql	15.sql	16.sql	17.sql	17-opt1.sql	17-opt2.sql	18.sql	18-opt.sql	19.sql	19-opt1.sql	19-opt2.sql	20.sql	21.sql	22.sql	Version	Query and explain details
Configuration	MariaDB 5.3.2 + MyISAM	Ratio	MariaDB 5.5.18 + MyISAM	Ratio	MySQL 5.5.19 + MyISAM	Ratio	MySQL 5.6.4 + MyISAM	Ratio																						
1.sql	261	1.00	308	1.18	259	0.99	277	1.06																						
2.sql	47	1.00	48	1.02	499	10.62	49	1.04																						
2-opt.sql	46	1.00	48	1.04	-	-	-	-																						
3.sql	243	1.00	246	1.01	>7200	-	1360	5.60																						
4.sql	137	1.00	135	0.99	4117	30.05	137	1.00																						
5.sql	181	1.00	187	1.03	6164	34.06	1254	6.93																						

Configuration	1.sql	2.sql	2-opt.sql	3.sql	4.sql	5.sql	6.sql	7.sql	8.sql	9.sql	10.sql	11.sql	12.sql	13.sql	14.sql	15.sql	16.sql	17.sql	17-opt1.sql	17-opt2.sql	18.sql	18-opt.sql	19.sql	19-opt1.sql	19-opt2.sql	20.sql	21.sql	22.sql	Version	Query and explain details
6.sql	198	1.000	205	1.004	>7200	-	194	0.98																						
7.sql	779	1.000	896	1.15	814	1.04	777	1.00																						
8.sql	270	1.000	287	1.006	749	2.77	1512	5.60																						
9.sql	252	1.000	254	1.001	>7200	-	298	1.18																						
10.sql	782	1.000	854	1.009	>7200	-	1881	2.41																						
11.sql	45	1.000	36	0.80	357	7.93	49	1.09																						
12.sql	211	1.000	217	1.003	>7200	-	213	1.01																						
13.sql	251	1.000	236	0.94	1590	6.33	244	0.97																						
14.sql	88	1.000	91	1.003	1590	18.07	94	1.07																						

Configuration	1.sql	2.sql	2-opt.sql	3.sql	4.sql	5.sql	6.sql	7.sql	8.sql	9.sql	10.sql	11.sql	12.sql	13.sql	14.sql	15.sql	16.sql	17.sql	17-opt1.sql	17-opt2.sql	18.sql	18-opt.sql	19.sql	19-opt1.sql	19-opt2.sql	20.sql	21.sql	22.sql	Version	Query and explain details
15.sql	162	1.000	164	1.001	4580	28.27	165	1.02																						
16.sql	154	1.000	152	0.99	174	1.13	173	1.12																						
17.sql	1493	1.000	1495	1.000	865	0.58	794	0.53																						
17-opt1.sql	795	1.000	794	1.000	862	1.08	794	1.000																						
17-opt2.sql	1482	1.000	1458	0.98	2167	1.46	1937	1.31																						
18.sql	971	1.000	931	0.96	>7200	-	>7200	-																						
18-opt.sql	121	1.000	125	1.003	-	-	-	-																						
19.sql	212	1.000	212	1.000	2004	9.45	61	0.29																						

Configuration	1.sql	2.sql	2-opt.sql	3.sql	4.sql	5.sql	6.sql	7.sql	8.sql	9.sql	10.sql	11.sql	12.sql	13.sql	14.sql	15.sql	16.sql	17.sql	17-opt1.sql	17-opt2.sql	18.sql	18-opt.sql	19.sql	19-opt1.sql	19-opt2.sql	20.sql	21.sql	22.sql	Version	Query and explain details
19-opt1.sql	59	1.000	59	1.000	1999	33.88	61	1.03																						
19-opt2.sql	260	1.000	216	0.83	443	1.70	236	0.91																						
20.sql	-	-	-	-	-	-	-	-																						
21.sql	173	1.000	179	1.03	>7200	-	183	1.06																						
22.sql	13	1.000	14	1.08	10	0.77	13	1.00																						
Version	5.3.2-MariaDB-beta		5.5.18-MariaDB		5.5.19		5.6.4-m7																							

Configuration	1.sql	2.sql	2-opt.sql	3.sql	4.sql	5.sql	6.sql	7.sql	8.sql	9.sql	10.sql	11.sql	12.sql	13.sql	14.sql	15.sql	16.sql	17.sql	17-opt1.sql	17-opt2.sql	18.sql	18-opt.sql	19.sql	19-opt1.sql	19-opt2.sql	20.sql	21.sql	22.sql	Version	Query and explain details
Query and explain details	Explain details		Explain details		Explain details		Explain details																							

NOTE: The columns named "*Ratio*" are calculated values of the ratio between the current value compared to the value in the first test configuration. The formula for it is `(current_value/value_in_first_row)`. For example if MariaDB 5.3.2 (the first column) handles a query for 100 seconds and MySQL 5.6.4 (the last configuration) handles the same query for 120 seconds, the ratio will be `120/100 = 1.20`. This means that it takes MySQL 5.6.4 20% more time to handle the same query.

The archived folder with all the results and details for that benchmark can be downloaded from here:MyISAM s30 on facebook-maria1

Notes

Queries 2-opt.sql and 18-opt.sql are tested only for MariaDB 5.3.2 and MariaDB 5.5.18

- Additional startup parameters for 2_opt:

```
--optimizer_switch='mrr_sort_keys=off'
```

- Additional startup parameters for 18_opt:

```
--optimizer_switch='semijoin=off' --optimizer_switch='index_condition_pushdown=on'
```

- Additional modifications for 17-opt1:

```
SELECT
SUM(l_extendedprice) / 7.0 AS avg_yearly
FROM
 part STRAIGHT_JOIN lineitem
WHERE
p_partkey = l_partkey
...
```

- Additional modifications for 17-opt2:

```
SELECT
    SUM(l_extendedprice) / 7.0 AS avg_yearly
FROM
    lineitem STRAIGHT_JOIN part
WHERE
    p_partkey = l_partkey
...
```

- Additional modifications for 19-opt1:

```
SELECT
    SUM(l_extendedprice* (1 - l_discount)) AS revenue
FROM
    part STRAIGHT_JOIN lineitem
WHERE
    (
        p_partkey = l_partkey
...
```

- Additional modifications for 19-opt2:

```
SELECT
    SUM(l_extendedprice* (1 - l_discount)) AS revenue
FROM
    lineitem STRAIGHT_JOIN part
WHERE
    (
        p_partkey = l_partkey
...
```

Benchmark for q20

This benchmarked only q20 with the same settings as described above for the other queries. The only difference is the timeout that was used: 30000 seconds
(8 hours and 20 min).

Compared cases

The benchmark for q20 compares the following cases:

- q20.sql - the original query is run with the IN-TO-EXISTS strategy for all servers.
  The following optimizer switches were used for MariaDB:

```
--optimizer_switch='in_to_exists=on,materialization=off,semijoin=off';
```

- q20-opt0.sql - the original query is changed so that the same join order is chosen as for the two subsequent variants that test materialization where this order is optimal.
  The join order is:

```
SELECT s_name, s_address
 FROM supplier, nation
 WHERE s_suppkey IN (SELECT DISTINCT (ps_suppkey)
 FROM '''part straight_join partsupp'''
 WHERE ps_partkey = p_partkey ...
```

- Since the IN-TO-EXISTS strategy is essentially the same for both MariaDB and MySQL, this query was tested for MySQL only.
- q20-opt1.sql - modifies the original query in two ways:
  - enforces the MATERIALIZATION strategy, and
  - enforces an optimal JOIN order via straight_join as follows:

```
SELECT s_name, s_address
 FROM supplier, nation
 WHERE s_suppkey IN (SELECT DISTINCT (ps_suppkey)
 FROM '''part straight_join partsupp'''
 WHERE ps_partkey = p_partkey ...
```

q20-opt1.sql uses the following optimizer switches for MariaDB:

```
--optimizer_switch='in_to_exists=off,materialization=on,semijoin=off';
```

- q20-opt2.sql - the same as q20-opt1.sql but allows the optimizer to choose the subquery strategy via the following switch:

```
--optimizer_switch='in_to_exists=on,materialization=on,semijoin=on';
```

- This switch results in the choice of SJ-MATERIALIZATION.

NOTE: For MySQL there are no such *optimizer-switch* parameters, and the tests were started without any additional startup parameters. The default algorithm in MySQL is *in_to_exists*.

Results for q20

Here is the graphics of the results of the benchmarked q20:
(Smaller bars are better)

NOTE: Queries that are cut off by the graphics have timed out the period of 30000 seconds.

Here are the actual results in seconds (smaller is better):

Configuration	20.sql	20-opt0.sql	20-opt1.sql	20-opt2.sql	Version	Query and explain details
MariaDB 5.3.2 + MyISAM	20070	-	5560	5615	5.3.2-MariaDB-beta	Explain details
MariaDB 5.5.18 + MyISAM	19922	-	5529	5572	5.5.18-MariaDB	Explain details
MySQL 5.5.19 + MyISAM	17832	>30000	-	-	5.5.19	Explain details
MYSQL 5.6.4 + MyISAM	19845	>30000	-	-	5.6.4-m7	Explain details

This page is licensed: CC BY-SA / Gnu FDL

DBT3 Example Preparation Time

This page contains database preparation and creation times that were discovered while working on the DBT3 Automation Scripts.

Database creation times

Setup	Time	Size on Disk
Postgre s30 on pitbull	758 min = 12 h 38 min.	75 GB
MariaDB + MyISAM s10 on 4CPU 4GB RAM developer machine	96 m 22s = 1h 36m 22s	17 GB
MariaDB + MyISAM s100 on facebook-maria2:	424m 4s = 7h 4m 4s.	162 GB
MariaDB + InnoDB s100 on facebook-maria2:	577m 42s = 9h 37m 42s.	
MariaDB 5.3.2 + InnoDB s30 on facebook-maria1	122m39.753s = 2h 2m 40s.	71 GB
MariaDB 5.5.18 + InnoDB s30 on facebook-maria1	150m46.218s = 2h 30m 46s.	71 GB
MySQL 5.6.4 + InnoDB s30 on facebook-maria1:	140m18.272s = 2h 20m 18s.	71 GB
MariaDB + MyISAM s30 on pitbull with pre_create_PK	> 3h 30m (with limited RAM to 11 GB)	
MariaDB + MyISAM s30 on facebook-maria1:	99m 8s = 1h 39m 8s.	49 GB
MySQL 5.6.4 + MyISAM s30 on facebook-maria1	114m 17s.	49 GB

Dataload creation time

Setup	Time	Size on Disk
Scale 10 on 4CPU 4GB RAM developer machine	8 min 44 sec.	10.4 GB
Scale 30 on pitbull.askmonty.org	20 min 28 sec.	32 GB (with limited RAM to 11 GB)
Scale 100 on facebook-maria2	78 min 27.601s.	106 GB
Scale 30 on facebook-maria1	23 min 40 sec.	

This page is licensed: CC BY-SA / Gnu FDL

MariaDB 5.3 - Asynchronous I/O on Windows with InnoDB

Consider 2 pseudo-code implementation of event handling loop handling IO completion on Windows.

- Using Windows events

```
void io_thread() {
    HANDLE handles = new HANDLE[32];
    ...
    for (;;) {
       DWORD index = WaitForMultipleObjects(handles,32, FALSE);
       DWORD num_bytes;
       // Find file and overlapped structure for the index,
       GetOverlappedResult(file, overlapped, &num_bytes, TRUE);
       // handle io represented by overlapped
   }
```

- I/O Completion port based

```
void io_thread() {
    for (;;) {
       DWORD num_bytes;
       ULONG_PTR key;
       OVERLAPPED *overlapped;
       if (GetQueuedCompletionStatus(io_completion_port, &num_bytes,
            &key, &overlapped, INFINITE)) {
          // handle io represented by overlapped
      }
   }
```

Which one is more efficient ? The right answer is - I/O completion port based. This is because:

1. the number of outstanding events a thread can handle is not restricted by a constant like in the WaitForMultipleObject() case.
2. if there several io_handler() threads running, they load-balance since every I/O can be “dequeued” by GetQueuedCompletionStatus in any io handler thread. With WaitForMultipleObjects(), the thread that will dequeue the I/O result is predetermined for each I/O.

InnoDB has used asynchronous file I/O on Windows since the dawn of time, probably since NT3.1 . On some reason unknown to me (I can only speculate that Microsoft documentation was not good enough back then), InnoDB has always used method with events, and this lead to relatively complicated designs - if you’re seeing “segment” mentioning in os0file.c or fil0fil.c , this is mostly due to the fact that number of events WaitForMultipleObjects() can handle is fixed.

I changed async IO handling for XtraDB in MariaDB5.3 to use completion ports, rather than wait_multiple technique. The results of a synthetic benchmark are good.

The test that I run was sysbench 0.4 “update_no_key”

```
4       16      64      256     1024
mariadb-5.2             17812   22378   23436   7882    6043
mariadb-5.2-fix         19217   24302   25499   25986   25925
mysql-5.5.13            12961   20445   16393   14407   5343
```

I do understand, sysbench it does not resemble anything that real-life load, and I’m ready to admit cheating with durability for this specific benchmark, but this is an equal-opportunity cheating, all 3 versions ran with the same parameters.

What do I refer to as durability cheating:

1. using innodb_flush_log_at_trx_commit=0 , which, for me , is ok for many scenarios
2. “Switch off Windows disk flushing” setting, which has the effect of not flushing data in the disk controller (file system caching is not used here anyway). This setting is only recommended for battery backed disks, my own desktop does not have it, of course.

However, if I have not done the above, then I would be measuring the latency of a FlushFileBuffers() in my benchmark, which was not what I wanted. I wanted to stress the asynchronous IO.

And here is the obligatory graph:

Notes

This is taken from an original Facebook note from Vladislav Vaintroub, and it can be found: note.php?note_id=238687382817625

It is also worth noting a note from Vlad about the graph: “The graph is given for 5.2, because I developed that patch for 5.2. I pushed it into 5.3 though :)”

This page is licensed: CC BY-SA / Gnu FDL

MariaDB 5.3/MySQL 5.5 Windows performance patches

I just backported Windows performance patches I’ve done for 5.5 back to MariaDB 5.3. There will be a bit more in Maria that in MySQL 5.5, but more on this later.

First, I feel Windows performance improvements in 5.5 were never adequately described, so here is the redux. For those familiar with Windows systems programming, MySQL code used to offer of low-hanging performance fruits. I picked some of them those back in my days in MySQL/Sun. The result benchmark curve became really nice: look at Calvin’s blog entry.

If graphs in this blog looks familiar to you, it is because it was often used by Oracle marketing as proof of big-O's positive influence on MySQL code :)

There were 3 Windows performance related patches. I comment on the bugs history a little bit, too.

- Bug#24509. The fix removed the limit of 2048 open MyISAM files, and as a nice side-effect allowed for much bigger table cache. When mysqld starts, it checks the maximum open files value, and corrects the value of table cache, if max_open_files is low or max_connection is high. This is what also happened during benchmarks. If you look at the read-only benchmark graph in the Calvin's blog above, you'll notice a drop around 64 concurrent users. No wonder, mysql server recalculated table cache size, setting it to the absolute minimum, that is to 64.

The fix was to create an own sort-of C runtime library on top of pure Win32, which is capable of handling more than 2048 open files (16K default). Some other things are also done nicer than in Microsofts C runtime, e.g there are no locks, and there is an acceptable pread()/pwrite() implementation. The main advantage as I said is being able to have a large table cache - for this, rewriting C runtime is likely an overkill, but I did not come up with anything better.

- Bug#52102. This bug was fixing a lot of questionable places in InnoDB that were written probably back in the NT3.1 days .

First it is importantto understand how innodb "mutex" structure is acquired. Details on it are hairy, mutex is a composite structure which has a real os mutex(known under Windows as CRITICAL_SECTION) plus innodb event (known under Windows as event). There are a couple of variations on the implementation - mutex can be an interlocked (aka atomic for my Unix friends) variable, under Unix event is represented as condition variable.

Acquisition is done in 2 steps - first, trylock on os mutex is performed , possibly several times with in a loop, if unsuccessful, event is reserved in a global table of events known as "sync array", event is entering a waiting state. mutex unlock would wakeup the waiters if there are any. Do not ask me why the implementation is so complicated, it is so :) Maybe, this design helps to find deadlocks.

Variation of this implementation - instead of trylock on mutex, there might be a compare_exchange instruction on interlocked(atomic) variable.

Back to Windows, the implementation of the above exposed a couple of interesting self-compensating bugs.

1. First, I fixed os_mutex_trylock() to be what it really means . The implementation was EnterCriticalSection, which is "try very hard", and actually acquire the lock. A more conscious trylock would be TryEnterCriticalSection. When I fixed that, contrary to my expectation, this made mysqld really slow. When trylock() failed, innodb started to enter code paths it has never seen before. for example, reserving space in the mentioned "sync array". Access to sync array is protected by so-called "slow lock" and this showed up very often in the profiler. The next step was fixing the "slow lock"
2. "slow Innodb mutex" was implemented as kernel object aka Windows mutex (for my Unix friends this is sort of SysV semaphore). It can be used to synchronize processes but is an absolute overkill for synchronization of threads inside the same process. It was a "really slow mutex". Changing this to CRITICAL_SECTION made it faster however…
3. When all of the above was fixed, found out that Windows events (mentioned events) did not really scale well in many-threads scenarios. On newer Windows (Vista+), there is a CONDITION_VARIABLE that is documented to scale better, and measuring also showed that it scaled really well. So I used condition variables when possible, which is ironic, because InnoDB events were really modeled after Windows events.
4. Re-enabled implementation of fast mutexes as atomic variables. Prior to the patch, precompiler flags to enable atomics were commented out with "Windows atomics do not work well" in CMakeLists.txt. Great comment, given that unlike software developers, atomic instructions have no preferences for the OSes they are being used on :)

So, the story about "atomics did not work well on Windows" was a cumulative effect of different things.

Prior to that patch . Once atomics were enabled, implementation of fast mutexes did not use CRITICAL_SECTION, but compare_exchange instruction. Ingenious "trylock_veryhard" as we have seen at the step 1. above is not used

anymore, instead it is a quite correct “try” lock . Once try_lock() began to fail with many concurrent threads, overhead of sync array guard implemented as Windows kernel object that we have seen in 2. became apparent, and less-then inefficient Windows events mentioned in 3. finished that picture.

- Bug#56585

This patch was merely to compensate for negative effects of the 5.5 metadata lock on MyISAM benchmarks, and fix was using native Vista performance primitives. The patch per se is not interesting, and repeats a lot of what was done for Innodb. What was great, was a discussion prior to the patch between myself, Davi, Dmitry on different implementations of reader writer locks, including 2 homebacked ones, and one by Vance Morrison.

Without doubt, the discussions around that was a highlight in my very short stint at Oracle. Also, if you want to get a MySQL-classic-style code review with 17 things to fix, of which at least 10 would be marked with “Coding Style” (yes, both words capitalized) , try to get Dmitry Lenev as a reviewer, he’s great - this is the proof 118295 Anyway, the patch improves MyISAM throughput by 10-20% , which I think is quite ok. Somehow those percents were subsequently eaten by MDL though :)

Notes

Taken from a note on Facebook: note.php?note_id=238505812835782 by Vladislav Vaintroub.

This page is licensed: CC BY-SA / Gnu FDL

mariadb-tools

All of the helper and wrapper scripts we use for benchmarks and tests are available in the mariadb.org-tools project on GitHub.

This page is licensed: CC BY-SA / Gnu FDL

Performance of MEMORY Tables

Between MariaDB 5.5.21 and 5.5.22 some work was done on how the hash index for a MEMORY table is created. This results in better performance when inserting rows into a memory table.

The following benchmark compares MariaDB 5.5.21 and 5.5.25. Compiled with identical settings on the same machine. The operation was loading 50 million rows into a `MEMORY` table with LOAD DATA INFILE.

Two different tables were tested: one with an indexed INT column and one with an indexed CHAR(20) column. The data files were pre-generated and located on a SSD. In order to make the effect visible, the cpu speed was set to minimum (core I5 @ 800Mhz)

Result:

Table Type	MariaDB Version	rows per second	Percent
INT	5.5.21	411022	100%
	5.5.25	510016	124%
CHAR(20)	5.5.21	259399	100%
	5.5.25	411535	159%

This is how the benchmark was run:

```
MariaDB [test]> tee 5.5.21.txt
MariaDB [test]> set @instance="5.5.21";
MariaDB [test]> source bench.sql
```

The script used to generate the data files:

```
#!/usr/bin/perl -w

$ROWS=50*1024*1024;

open F, ">/tmp/hash1.txt" or die;
for ($i=0; $i<$ROWS; $i++) {
    printf F "%d\n", int(rand($ROWS));
}
close F or die;

open F, ">/tmp/hash2.txt" or die;
for ($i=0; $i<$ROWS; $i++) {
    $s="";
    for (1..20) { $s .= chr(ord('a')+int(rand(26))); }
    print F $s, "\n";
}
close F or die;
```

The benchmark SQL script `bench.sql`:

```
use test;

-- need big heap tables
set max_heap_table_size=4*1024*1024*1024;

-- table to hold test results
create table if not exists results (
  id serial,
  operation char(32),
  opsize bigint unsigned,
  started datetime,
  ended datetime,
  instance char(20)
);

-- dummy run with second data file
drop table if exists t1;
create table t1 (c1 char(20), index (c1)) engine memory;
load data infile "/tmp/hash2.txt" into table t1;
drop table t1;

-- do total of 5 runs for each table

-- run #1
create table t1 (c1 int, index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash1.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
values ("load into INT table", @rows, @t1, @t2, @instance);
drop table t1;

create table t1 (c1 char(20), index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash2.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
values ("load into CHAR(20) table", @rows, @t1, @t2, @instance);
drop table t1;

-- run #2
create table t1 (c1 int, index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash1.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
values ("load into INT table", @rows, @t1, @t2, @instance);
drop table t1;

create table t1 (c1 char(20), index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash2.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
values ("load into CHAR(20) table", @rows, @t1, @t2, @instance);
drop table t1;

-- run #3
create table t1 (c1 int, index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash1.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
```

```
values ("load into INT table", @rows, @t1, @t2, @instance);
drop table t1;

create table t1 (c1 char(20), index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash2.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
values ("load into CHAR(20) table", @rows, @t1, @t2, @instance);
drop table t1;

-- run #4
create table t1 (c1 int, index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash1.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
values ("load into INT table", @rows, @t1, @t2, @instance);
drop table t1;

create table t1 (c1 char(20), index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash2.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
values ("load into CHAR(20) table", @rows, @t1, @t2, @instance);
drop table t1;

-- run #5
create table t1 (c1 int, index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash1.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
values ("load into INT table", @rows, @t1, @t2, @instance);
show table status like 't1';
drop table t1;

create table t1 (c1 char(20), index (c1)) engine memory;
select @t1:=now();
load data infile "/tmp/hash2.txt" into table t1;
select @t2:=now();
select @rows:=count(*) from t1;
insert into results (operation, opsize, started, ended, instance)
values ("load into CHAR(20) table", @rows, @t1, @t2, @instance);
show table status like 't1';
drop table t1;

-- list all results
select operation, instance, unix_timestamp(ended)-unix_timestamp(started) as duration,
 opsize/(unix_timestamp(ended)-unix_timestamp(started)) as ops_per_sec
from results order by operation, instance, started;

-- list average results
select operation, instance, avg(opsize/(unix_timestamp(ended)-unix_timestamp(started))) as
avg_ops_per_sec
from results group by operation, instance;
```

This page is licensed: CC BY-SA / Gnu FDL

Recommended Settings for Benchmarks

Running benchmarks requires a lot of different settings. In this article we collect our best known settings and recommendations.

Hardware and BIOS Settings

We have had good experiences with Intel's hyperthreading on newer Xeon CPUs. Please turn on hyperthreading in your BIOS.

NUMA

The NUMA architecture attaches resources (most important: memory) to individual NUMA nodes (typically: NUMA node = cpu socket). This results in a performance penalty when a cpu core from one NUMA node accesses memory from another NUMA node.

The NUMA topology can be checked with the *numactl* command:

```
~ $numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 12 13 14 15 16 17
node 0 size: 12278 MB
node 0 free: 11624 MB
node 1 cpus: 6 7 8 9 10 11 18 19 20 21 22 23
node 1 size: 12288 MB
node 1 free: 11778 MB
node distances:
node   0   1
  0:  10  21
  1:  21  10
```

The Linux kernel uses ACPI tables from BIOS to detect if the hardware is NUMA. On NUMA hardware extra optimizations kick in:

- if a task has been scheduled on a certain NUMA node, the scheduler tries to put it on the same node again in the future
- if a task running on a certain NUMA node allocates memory, the kernel tries hard to map physical memory from the same NUMA node

This results in all kinds of weird behavior when you run one big process (mysqld) that consumes most of the memory. In such cases it is recommended to either turn off NUMA (BIOS or kernel command line) or prefix such problem processes with *numactl –interleave all*. You can enable this by running mysqld_safe with the `--numa-interleave` option.

More details can be found here.

Linux Kernel Settings

See configuring Linux for MariaDB.

InnoDB Settings

innodb_buffer_pool_size to about 80% of RAM or leaving <5G RAM free (on large RAM systems). Less if lots of connections are used.

innodb_log_file_size to be larger than the amount of writes in the test run or sufficient to cover several minutes of the test run at least.

MyISAM Settings

General Settings

threads_cache_size should be the same as max_connections (unless using thread pools).

This page is licensed: CC BY-SA / Gnu FDL

RQG Performance Comparisons

Performance testing

The `performance/perfrun.pl` executes each query against a set of tw servers and reports the outcome.

```
perl performance/perfrun.pl --input-directory=... --dsn1=... --dsn2=... --output-file=... --filter=...
```

- `--input-directory` contains the queries to be run, one query per file. Alternative sources for queries will be made available in the future;
- `--dsn1` and `--dsn2` specify the locations of the two servers being compared in the Perl DBI URL format. If the queries are not fully qualified, the DSNs must contain the database name. The username and the password are also provided via the DSN. For example:

```
--dsn1=dbi:mysql:host=127.0.0.1:port=19300:user=root:database=test \
--dsn2=dbi:mysql:host=127.0.0.1:port=19302:user=root:database=test
```

- `--output-file=...` specifies a file where matching queries will be dumped in a form suitable for follow-up filtering with `perfreport`
- if no `--filter` is specified, all queries from the input will be reported;

Performance reporting

The `performance/perfreport.pl` script takes an output file created by `perfrun` and prints out its contents, possibly applying a filter in the process:

```
perl performance/perfreport.pl --input-file=... --filter=...
```

If no `--filter` is specified, all queries present in the input file are printed.

Filters

The queries reported by the system can be filtered out using perl-based filtering expressions. A filter can be applied during data collection, at which point it determines what information is stored in the output file and at

If you do not specify a filter, it is assumed that you want to output all queries.

The following variables can participate in filter expressions:

- Variables from `SHOW SESSION STATUS`, except:
  - variables that are not reset at the start of the query, which includes `Com_*`, `Uptime`, `Opened_files` and the like;
  - variables that relate to the operation of SSL encryption or the query cache;
- Variables from `SHOW GLOBAL STATUS LIKE 'Innodb_%'`

For each MySQL status variable, 4 Perl variables are provided
— the value of the MySQL variable from each server, their absolute difference and their ratio. For example, for the `Innodb_rows_read` MySQL variable, you can use `Innodb_rows_read1`, `Innodb_rows_read2`, `Innodb_rows_read_delta` and `Innodb_rows_read_ratio` in your Perl filter expressions.

In addition to the MySQL status variables, the framework provides the following additional variables:

- `$Execution_time{1|2|delta|ratio}` reports the time the query took to run from start to end in seconds. The value is truncated at milliseconds in order to prevent exorbitant performance ratios from being reported on very fast queries;
- `$Temperature` can either be `cold` or `warm` depending on whether the first execution of the query is being processed, or the second.
- `$Query` contains the text of the query, in order to enable filtering such as `$Query !~ m{l_receiptDATE}`

For example, the following command-line option:

```
--filter='($Execution_time1 > 0.1 || $Execution_time2 > 0.1) && $Temperature eq "warm"'
```

Will only process or report queries that took more than 0.1 sec to execute on a warm server.

On-screen Reporting Format

The ASCII on-screen reporting format is as follows:

```
Query: SELECT  l_linenumber FROM lineitem  WHERE l_suppkey  IN ( 10 , 1 ) AND l_shipdate NOT IN ( '1993-
06-06' , '1998-02-04' , '1992-01-08' ) AND l_linenumber NOT IN ( 8 , 7 ) AND l_quantity <> 3 AND (
l_orderkey NOT IN ( 1298 , 93 ) OR l_suppkey = 10 ) AND ( l_suppkey BETWEEN 4 AND 10 + 4 OR l_linenumber
= 2 AND l_commitDATE BETWEEN '1993-06-27' AND '1993-09-05' AND l_linenumber BETWEEN 3 AND 9 + 9 );
Cache: warm
                                  5.3.0-MariaDB-     5.2.6-MariaDB-          Delta              Ratio
-----------------------------------------------------------------------------------------------------
Execution time                             0.011s            0.004s           -0.007s            0.36
Innodb_buffer_pool_read_requests          2171              1836              -335               0.85
Handler_read_rnd                           583                 0              -583
Handler_read_next                          583               602                19               1.03
Innodb_rows_read                          1166               602              -564               0.52
```

Only variables whose values are different between the two servers are reported.
In this particular example, the query ran slower on `5.3.0-MariaDB` with warm cache and caused twice as many `Innodb_rows_read`.

On-disk Data Storage Format

The on-disk storage format is `Data::Dumper` objects, wrapped in `<![CDATA[ ... ]]>` tags, without the file being a full-blown XML. The serialized representation is created by `GenTest::QueryPerformanceDelta::serialize()` and is read by using `eval()` in `performance/perfreport.pl`

See also:

- RQG Documentation
- RQG Extensions for MariaDB Features
- Optimizer Quality
- QA Tools
- Worklog Quality Checklist Template

This page is licensed: CC BY-SA / Gnu FDL

run-sql-bench.pl

`run-sql-bench.pl` is a perl script for automating runs of sql-bench
(You can find sql-bench in the MariaDB source code.)

`run-sql-bench.pl` can be found in the mariadb-tools project on Launchpad. Once you have a copy of mariadb-tools, you'll find the script, and its configuration directories, in the `sql-bench` directory. For the purposes of this article, wherever you located your local branch of `mariadb-tools` will be called `${BASE_DIR}`.

The `run-sql-bench.pl` script is located at '`${BASE_DIR}/sql-bench/run-sql-bench.pl`'.

Example configuration scripts used for different runs can be found in the various subdirectories of `${BASE_DIR}/sql-bench/`.

To run the `run-sql-bench.pl` script, do the following:

1. Branch a MariaDB or MySQL tree
2. Optionally do some code changes in that tree and commit your changes
3. Edit `run-sql-bench.pl` to set internal options, especially the "`my $path`" variable.
4. Create a `${BASE_DIR}/sql-bench/conf/host.cnf` file for your system. An easy way to do this is by duplicating one of the example `host.cnf` files:

```
cp -avi ${BASE_DIR}/sql-bench/conf/pitbull.cnf ${BASE_DIR}/sql-bench/conf/${HOSTNAME}.cnf
```

Edit the file to customize it for your system.

1. Create `${HOSTNAME}.cnf` files under the `${BASE_DIR}/sql-bench/basic`, `${BASE_DIR}/sql-bench/debug`, `${BASE_DIR}/sql-bench/debug-full`, and `${BASE_DIR}/sql-bench/O2` directories, depending on which automated tests you want to run. Use the example files as a base and customize them for your system.
2. Run the `run-sql-bench.pl` script:

```
cd ${BASE_DIR}/sql-bench/; ./run-sql-bench.pl --repository=[/path/to/bzr/repository] --sql-bench-options=
[additional sql-bench-options] --debug=[yes|no]
```

- `--repository` is the MariaDB tree to use and compile, the script will also look here for sql-bench.
- `--sql-bench-options` is mostly used in testing and debugging cases where we want to have short run times. For instance, using `--small-test` or `--small-table`.
- You can separate several sql-bench options with spaces like so:

```
--sql-bench-options="--small-test --small-table"
```

The results

Results are stored at the location specified by the `$sql_bench_results` variable in the `${BASE_DIR}/sql-bench/conf/hostname.cnf` file for your host.
Results are organized in sub directories with the following schema:

```
sql-bench-results-dir/${HOSTNAME}/YYYY-MM-DD
```

Future plans

- Crash and error detection and reporting.
- One should be able to specify a test name for each file (`run-all-tests --suffix='_xxxx'`)

This page is licensed: CC BY-SA / Gnu FDL

Segmented Key Cache Performance

Testing method for segmented key cache performance

We used SysBench v0.5 from Launchpad to test the segmented key cache performance for the MyISAM storage engine of MariaDB 5.2.2-gamma.

As wrapper scripts for automated running of SysBench we used the `sysbench/` directory from MariaDB Tools.

To test that splitting the key cache's global mutex into several mutex helps under multi user load, we wrote a new SysBench test called `select_random_points.lua`. We used one big table and selected random points with increasing number of concurrent users.

Main testing outcomes

We see up to 250% performance gain depending on the amount of concurrent users.

Detailed testing outcomes

On our machine pitbull

On pitbull with –random-points=10

In relative numbers:

```
Threads                1      4      8      16      32      64      128
(32/off)             -3%    53%    122%    155%    226%    269%    237%
(64/off)             -6%    55%    130%    162%    234%    270%    253%

select_random_points.lua --random-points=10
```

On pitbull with –random-points=50

In relative numbers:

```
Threads                1      4      8      16      32      64      128
(32/off)             -3%    53%    113%    154%    232%    254%    231%
(64/off)             -1%    55%    121%    161%    235%    268%    244%

select_random_points.lua --random-points=50
```

On pitbull with –random-points=100

In relative numbers:

```
Threads                1      4      8      16      32      64      128
(32/off)             -3%    54%    121%    160%    209%    246%    219%
(64/off)             -6%    56%    129%    167%    219%    260%    241%

select_random_points.lua --random-points=100
```

Detailed numbers of all runs on pitbull

You can find the absolute and relative numbers in our OpenOffice.org spread sheet here: SysBench v0.5 select_random_points on pitbull

On our machine perro

On perro with –random-points=10

In relative numbers:

```
Threads                1      4      8      16      32      64      128
(32/off)              1%     2%     17%     45%     73%     70%     71%
(64/off)            -0.3%    6%     19%     46%     72%     74%     80%

select_random_points.lua --random-points=10
```

On perro with –random-points=50

In relative numbers:

```
Threads                1      4      8      16      32      64      128
(32/off)              1%    10%     26%     69%    105%    122%    114%
(64/off)             -1%     8%     27%     75%    111%    120%    131%

select_random_points.lua --random-points=50
```

On perro with –random-points=100

In relative numbers:

```
Threads                1      4      8      16      32      64      128
(32/off)            -0.2%    1%     22%     73%    114%    114%    126%
(64/off)            -0.1%    4%     22%     75%    112%    125%    135%

select_random_points.lua --random-points=100
```

Detailed numbers of all runs on perro

You can find the absolute and relative numbers in our OpenOffice.org spread sheet here: SysBench v0.5 select_random_points on perro

Table and query used

Table definition:

```
CREATE TABLE sbtest (
  id  INT UNSIGNED NOT NULL AUTO_INCREMENT,
  k   INT UNSIGNED NOT NULL DEFAULT '0',
  c   CHAR(120) NOT NULL DEFAULT '',
  pad CHAR(60) NOT NULL DEFAULT '',
  PRIMARY KEY (id),
  KEY k (k)
) ENGINE=MyISAM
```

Query used:

```
SELECT id, k, c, pad
    FROM sbtest
    WHERE k IN (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
```

The `?` parameters were replaced by random numbers when running the SysBench test. We used 10, 50, and 100 random points in our tests.

We inserted 20 million rows using random data, which gave us a data and index file size of:

```
3.6G    sbtest.MYD
313M    sbtest.MYI
```

We chose our key buffer size to be big enough to hold the index file.

Testing environment

MariaDB sources

We used MariaDB 5.2.2-gamma with following revision from our launchpad repository Revision #2878

```
revno: 2878
committer: Sergei Golubchik <sergii@pisem.net>
branch nick: 5.2
timestamp: Tue 2010-10-26 07:37:44 +0200
message:
  fixes for windows
```

Compiling MariaDB

We compiled MariaDB using this line:

```
BUILD/compile-amd64-max
```

MariaDB runtime options

We used following configuration for running MariaDB

```
MYSQLD_OPTIONS="--no-defaults \
  --datadir=$DATA_DIR \
  --language=./sql/share/english \
  --log-error \
  --key_buffer_size=512M \
  --max_connections=256 \
  --query_cache_size=0 \
  --query_cache_type=0 \
  --skip-grant-tables \
  --socket=$MY_SOCKET \
  --table_open_cache=512 \
  --thread_cache=512 \
  --key_cache_segments=0 \ # 0 | 32 | 64
  --tmpdir=$TEMP_DIR"
```

SysBench v0.5 select_random_points.lua options

We run the SysBench v0.5 select_random_points.lua test with following options:

```
# 20 million rows.
TABLE_SIZE=20000000

SYSBENCH_OPTIONS="--oltp-table-size=$TABLE_SIZE \
  --max-requests=0 \
  --mysql-table-engine=MyISAM \
  --mysql-user=root \
  --mysql-engine-trx=no \
  --myisam-max-rows=50000000 \
  --rand-seed=303"
```

We tested with increasing number of concurrent users with a warm up time of 8 minutes and a run time of 20 minutes:

```
NUM_THREADS="1 4 8 16 32 64 128"
...
  --num-threads=$THREADS
```

We also tested an increasing number of random points:

```
# Default option is --random-points=10.
SYSBENCH_TESTS[0]="select_random_points.lua"
SYSBENCH_TESTS[1]="select_random_points.lua --random-points=50"
SYSBENCH_TESTS[2]="select_random_points.lua --random-points=100"
```

Kernel parameters

IO scheduler

For optimal IO performance running a database we are using the noop scheduler. You can check your scheduler setting with:

```
cat /sys/block/${DEVICE}/queue/scheduler
```

For instance, it should look like this output:

```
cat /sys/block/sda/queue/scheduler
[noop] deadline cfq
```

You can find detailed notes about Linux schedulers here: Linux schedulers in TPCC like benchmark.

Open file limits

Having a lot of concurrent connections can hit the open file limit on your system. On most Linux systems the open file limit is at 1024, which can be not enough. Please set your open file limit higher by editing

```
$EDITOR /etc/security/limits.conf
```

and adding a line like

```
#ftp            hard    nproc           0
#@student       -       maxlogins       4
*               -       nofile          16384

# End of file
```

Your `ulimit -a` output should look like this afterwards:

```
ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 15975
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) 1744200
open files                      (-n) 16384
```

Machines used for testing

perro

```
# OS: openSUSE 11.1 (x86_64)
# Platform: x86_64
# CPU: Quad-core Intel @ 3.20GHz: 4 CPUs
# RAM: 2GB
# Disk(s): 2 x ST31000528AS S-ATA as software RAID 0
```

pitbull

```
# OS: Ubuntu 10.10
# Platform: x86_64
# CPU: Two-socket x hexa-core Intel Xeon X5660 @ 2.80GHz. With hyperthreading: 24CPUs
# RAM: 28GB
# Disk(s): 1 x ST3500320NS S-ATA
```

This page is licensed: CC BY-SA / Gnu FDL

sysbench Benchmark Setup

For our automated MariaDB/MySQL sysbench benchmark tests, we use sysbench from `lp:sysbench`. This page describes the basic parameters and configuration we use.

You can find the automation wrapper scripts we use for running sysbench in lp:mariadb-tools

Current general parameters

```
table_open_cache = 512
thread_cache = 512
query_cache_size = 0
query_cache_type = 0
```

Current InnoDB parameters

```
innodb_data_home_dir = /data/mysql/
innodb_data_file_path = ibdata1:128M:autoextend
innodb_log_group_home_dir = /data/mysql/
innodb_buffer_pool_size = 1024M
innodb_additional_mem_pool_size = 32M
innodb_log_file_size = 256M
innodb_log_buffer_size = 16M
innodb_flush_log_at_trx_commit = 1
innodb_lock_wait_timeout = 50
innodb_doublewrite = 0
innodb_flush_method = O_DIRECT
innodb_thread_concurrency = 0
innodb_max_dirty_pages_pct = 80
```

Compile

Install MariaDB or MySQL to /usr/local and make a symlink to /usr/local/mysql.
Please use non-debug version! On Mac OS X we currently need automake1.10.

```
./autogen.sh
./configure \
  --without-drizzle \
  --with-mysql-includes=/usr/local/mysql/include/mysql \
  --with-mysql-libs=/usr/local/mysql/lib/mysql
make
optionally: make install
```

Start and prepare database to use

```
mysqladmin -uroot drop sbtest
mysqladmin -uroot create sbtest
```

Tests

We use the latest sysbench with Lua scripting support. Therefore the test names differ from sysbench <= 0.4. To get reasonable results we use a run time of 5 minutes.

We run the tests with 1, 4, 8, 16, 32, 64, and 128 threads.

```
NUM_THREADS="1 4 8 16 32 64 128"
SYSBENCH_TESTS="delete.lua \
  insert.lua \
  oltp_complex_ro.lua \
  oltp_complex_rw.lua \
  oltp_simple.lua \
  select.lua \
  update_index.lua \
  update_non_index.lua"

NUM_THREADS=1
TEST_DIR=${HOME}/work/monty_program/sysbench/sysbench/tests/db

./sysbench \
  --test=${TEST_DIR}/oltp_simple.lua \
  --oltp-table-size=2000000 \
  --max-time=300 \
  --max-requests=0 \
  --mysql-table-engine=InnoDB \
  --mysql-user=root \
  --mysql-engine-trx=yes \
  --num-threads=$NUM_THREADS \
  prepare

./sysbench \
  --test=${TEST_DIR}/oltp_simple.lua \
  --oltp-table-size=2000000 \
  --max-time=300 \
  --max-requests=0 \
  --mysql-table-engine=InnoDB \
  --mysql-user=root \
  --mysql-engine-trx=yes \
  --num-threads=$NUM_THREADS \
  run
```

Custom added tests

We created a couple of custom tests for SysBench:

- select_random_ranges.lua
- select_random_points.lua

Both of these have been added to the latest SysBench v0.5 repository.

This page is licensed: CC BY-SA / Gnu FDL

InnoDB Upgrade Tests

- 10.0.32 Release Upgrade Tests
- 10.0.34 Release Upgrade Tests
- 10.0.36 Release Upgrade Tests
- 10.1.22 Pre-release Upgrade Tests
- 10.1.25 Release Upgrade Tests
- 10.1.26 Release Upgrade Tests
- 10.1.30 Release Upgrade Tests
- 10.1.32 Release Upgrade Tests
- 10.1.36 Release Upgrade Tests
- 10.2.12 Release Upgrade Tests
- 10.2.13 Release Upgrade Tests
- 10.2.14 Release Upgrade Tests
- 10.2.17 Release Upgrade Tests
- 10.2.18 Release Upgrade Tests
- 10.2.5 Pre-release Upgrade Tests
- 10.2.7 Release Upgrade Tests
- 10.3.10 Release Upgrade Tests
- 10.3.3-beta Release Upgrade Tests
- 10.3.4-beta Release Upgrade Tests
- 10.3.5-gamma Release Upgrade Tests
- 10.3.6-gamma Release Upgrade Tests
- 10.3.9 Release Upgrade Tests

This page is licensed: CC BY-SA / Gnu FDL

10.0.32 Release Upgrade Tests

Upgrade from 10.0

Tested revision

d85d6c9c416aab85f1ded344a5b870d4cc56f9f6

Test date

2017-08-03 19:25:53

Summary (PASS)

All tests passed

Details

#	type	pages ize	OLD version	file format	encry pted	compre ssed		NEW version	file format	encry pted	compre ssed	readо nly	res ult	not es
1	cras h	4	10.0.28 (inbuilt)	Antelo pe	-	-	= >	10.0.32 (inbuilt)	Antelo pe	-	-	-	OK	

#	type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
2	crash	4	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
3	crash	8	10.0.28 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	
4	crash	8	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
5	crash	16	10.0.28 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	
6	crash	16	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
7	normal	4	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
8	normal	4	10.0.28 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	
9	normal	8	10.0.28 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	
10	normal	8	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
11	normal	16	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
12	normal	16	10.0.28 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	

Upgrade from MySQL 5.6

Tested revision

d85d6c9c416aab85f1ded344a5b870d4cc56f9f6

Test date

2017-08-03 19:45:21

Summary (PASS)

All tests passed

Details

#	type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
1	normal	16	5.6.35 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	
2	normal	16	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
3	normal	4	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
4	normal	4	5.6.35 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	

#	type	pages ize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
5	normal	8	5.6.35 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	
6	normal	8	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	

Crash Recovery

Tested revision

d85d6c9c416aab85f1ded344a5b870d4cc56f9f6

Test date

2017-08-03 19:55:23

Summary (PASS)

All tests passed

Details

#	type	pages ize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
1	crash	16	10.0.32 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	
2	crash	16	10.0.32 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
3	crash	4	10.0.32 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	
4	crash	4	10.0.32 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	
5	crash	8	10.0.32 (inbuilt)	Antelope	-	-	=>	10.0.32 (inbuilt)	Antelope	-	-	-	OK	
6	crash	8	10.0.32 (inbuilt)	Barracuda	-	-	=>	10.0.32 (inbuilt)	Barracuda	-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.0.34 Release Upgrade Tests

Tested revision

d01dbe66a8bf9cb6031f95159c49100f9299a768

Test date

2018-02-01 07:13:41

Summary

All tests passed

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
undo-recovery	4	10.0.34 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo-recovery	8	10.0.34 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo-recovery	16	10.0.34 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
recovery	16	10.0.34 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
recovery	4	10.0.34 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
recovery	8	10.0.34 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash	4	10.0.33 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash	8	10.0.33 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash	16	10.0.33 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
normal	16	10.0.33 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
normal	4	10.0.33 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
normal	8	10.0.33 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo	16	10.0.33 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo	4	10.0.33 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo	8	10.0.33 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash	4	10.0.14 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash	8	10.0.14 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash	16	10.0.14 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
undo	8	10.0.18 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash	4	5.6.39 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash	8	5.6.39 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash	16	5.6.39 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo	4	5.6.39 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
undo	8	5.6.39 (inbuilt)		-	-	=>	10.0.34 (inbuilt)		-	-	-	OK	
crash-downgrade	4	10.0.34 (inbuilt)		-	-	=>	10.0.33 (inbuilt)		-	-	-	OK	
crash-downgrade	8	10.0.34 (inbuilt)		-	-	=>	10.0.33 (inbuilt)		-	-	-	OK	
crash-downgrade	16	10.0.34 (inbuilt)		-	-	=>	10.0.33 (inbuilt)		-	-	-	OK	
downgrade	16	10.0.34 (inbuilt)		-	-	=>	10.0.33 (inbuilt)		-	-	-	OK	
downgrade	4	10.0.34 (inbuilt)		-	-	=>	10.0.33 (inbuilt)		-	-	-	OK	
downgrade	8	10.0.34 (inbuilt)		-	-	=>	10.0.33 (inbuilt)		-	-	-	OK	
crash-downgrade	4	10.0.34 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
crash-downgrade	8	10.0.34 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
crash-downgrade	16	10.0.34 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
downgrade	16	10.0.34 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
downgrade	4	10.0.34 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
downgrade	8	10.0.34 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.0.36 Release Upgrade Tests

Tested revision

e023f9a4d5a620b54d7f7132567150d80b630692

Test date

2018-08-05 19:08:18

Summary

Two tests failed due to problems in previous versions (unrelated to the release)

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	4	10.0.36 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo-recovery	8	10.0.36 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo-recovery	16	10.0.36 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
recovery	16	10.0.36 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
recovery	4	10.0.36 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
recovery	8	10.0.36 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
crash	4	10.0.35 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
crash	8	10.0.35 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
crash	16	10.0.35 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
normal	16	10.0.35 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
normal	4	10.0.35 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
normal	8	10.0.35 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo	16	10.0.35 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo	4	10.0.35 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo	8	10.0.35 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
crash	4	10.0.14 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
crash	8	10.0.14 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	16	10.0.14 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
crash	4	5.6.41 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
crash	8	5.6.41 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
crash	16	5.6.41 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
normal	16	5.6.41 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
normal	4	5.6.41 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
normal	8	5.6.41 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo	16	5.6.41 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	5.6.41 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
undo	8	5.6.41 (inbuilt)		-	-	=>	10.0.36 (inbuilt)		-	-	-	OK	
crash-downgrade	4	10.0.36 (inbuilt)		-	-	=>	10.0.35 (inbuilt)		-	-	-	OK	
crash-downgrade	8	10.0.36 (inbuilt)		-	-	=>	10.0.35 (inbuilt)		-	-	-	OK	
crash-downgrade	16	10.0.36 (inbuilt)		-	-	=>	10.0.35 (inbuilt)		-	-	-	OK	
downgrade	16	10.0.36 (inbuilt)		-	-	=>	10.0.35 (inbuilt)		-	-	-	OK	
downgrade	4	10.0.36 (inbuilt)		-	-	=>	10.0.35 (inbuilt)		-	-	-	OK	
downgrade	8	10.0.36 (inbuilt)		-	-	=>	10.0.35 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash-downgrade	4	10.0.36 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
crash-downgrade	8	10.0.36 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
crash-downgrade	16	10.0.36 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
downgrade	16	10.0.36 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
downgrade	4	10.0.36 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	
downgrade	8	10.0.36 (inbuilt)		-	-	=>	10.0.14 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.1.22 Pre-release Upgrade Tests

Upgrade from 10.0

Tested revision

dbd1d7ea8e96a2b4cff89ec889494700d634b3a3

Test date

2017-03-07 13:42:18

Summary (PASS)

All tests passed

Details

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1	crash	16	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
2	crash	16	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
3	crash	4	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
4	crash	4	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
5	crash	8	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
6	crash	8	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
7	crash	16	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
8	crash	16	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
9	crash	4	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
10	crash	4	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
11	crash	8	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
12	crash	8	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
13	crash	8	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
14	crash	8	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
15	crash	16	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
16	crash	16	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
17	crash	4	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
18	crash	4	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
19	crash	4	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
20	crash	4	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
21	crash	16	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
22	crash	16	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
23	crash	8	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
24	crash	8	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
25	crash	8	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
26	crash	8	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
27	crash	16	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
28	crash	16	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
29	crash	4	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
30	crash	4	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
31	crash	8	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
32	crash	8	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
33	crash	16	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
34	crash	16	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
35	crash	4	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
36	crash	4	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
37	crash	8	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
38	crash	8	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
39	crash	4	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
40	crash	4	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
41	crash	16	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
42	crash	16	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
43	crash	16	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
44	crash	16	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
45	crash	8	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
46	crash	8	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
47	crash	4	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
48	crash	4	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
49	normal	4	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
50	normal	4	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
51	normal	8	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
52	normal	8	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
53	normal	16	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
54	normal	16	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
55	normal	4	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
56	normal	4	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
57	normal	16	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
58	normal	16	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
59	normal	8	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
60	normal	8	10.0.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
61	normal	8	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
62	normal	8	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
63	normal	4	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
64	normal	4	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
65	normal	16	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
66	normal	16	10.0.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
67	normal	16	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
68	normal	16	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
69	normal	4	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
70	normal	4	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
71	normal	8	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
72	normal	8	10.0.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
73	normal	4	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
74	normal	4	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
75	normal	8	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
76	normal	8	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
77	normal	16	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
78	normal	16	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
79	normal	4	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
80	normal	4	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
81	normal	16	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
82	normal	16	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
83	normal	8	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
84	normal	8	10.0.29 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
85	normal	8	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
86	normal	8	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
87	normal	16	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
88	normal	16	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
89	normal	4	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
90	normal	4	10.0.29 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
91	normal	4	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
92	normal	4	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
93	normal	16	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
94	normal	16	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
95	normal	8	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
96	normal	8	10.0.29 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	

Upgrade from 10.1

Tested revision

adc91387e3add6d9c850b7c2a44760deaceb3638

Test date

2017-03-05 18:09:10

Summary (FAIL)

34 failures out of 1280 combinations, all associated with MDEV-9422 (crash-upgrade from 10.1.10 with encryption enabled)

Details

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1	crash	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
2	crash	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
3	crash	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
4	crash	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
5	crash	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
6	crash	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
7	crash	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
8	crash	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
9	crash	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
10	crash	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
11	crash	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
12	crash	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
13	crash	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
14	crash	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
15	crash	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
16	crash	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
17	crash	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
18	crash	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
19	crash	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
20	crash	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
21	crash	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
22	crash	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
23	crash	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
24	crash	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
25	crash	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
26	crash	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
27	crash	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
28	crash	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
29	crash	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
30	crash	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
31	crash	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
32	crash	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
33	crash	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
34	crash	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
35	crash	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
36	crash	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
37	crash	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
38	crash	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
39	crash	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
40	crash	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
41	crash	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
42	crash	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
43	crash	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
44	crash	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
45	crash	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
46	crash	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
47	crash	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
48	crash	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
49	crash	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
50	crash	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
51	crash	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
52	crash	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
53	crash	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
54	crash	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
55	crash	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
56	crash	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
57	crash	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
58	crash	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
59	crash	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
60	crash	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
61	crash	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
62	crash	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
63	crash	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
64	crash	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
65	crash	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
66	crash	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
67	crash	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
68	crash	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
69	crash	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
70	crash	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
71	crash	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
72	crash	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
73	crash	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
74	crash	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
75	crash	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
76	crash	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
77	crash	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
78	crash	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
79	crash	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
80	crash	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
81	crash	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
82	crash	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
83	crash	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
84	crash	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
85	crash	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
86	crash	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
87	crash	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
88	crash	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
89	crash	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
90	crash	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
91	crash	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
92	crash	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
93	crash	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
94	crash	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
95	crash	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
96	crash	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
97	crash	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
98	crash	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
99	crash	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
100	crash	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
101	crash	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
102	crash	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
103	crash	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
104	crash	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
105	crash	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
106	crash	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
107	crash	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
108	crash	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
109	crash	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
110	crash	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
111	crash	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
112	crash	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
113	crash	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
114	crash	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
115	crash	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
116	crash	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
117	crash	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
118	crash	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
119	crash	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
120	crash	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
121	crash	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
122	crash	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
123	crash	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
124	crash	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
125	crash	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
126	crash	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
127	crash	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
128	crash	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
129	crash	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
130	crash	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
131	crash	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
132	crash	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
133	crash	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
134	crash	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
135	crash	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
136	crash	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
137	crash	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
138	crash	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
139	crash	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
140	crash	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
141	crash	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
142	crash	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
143	crash	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
144	crash	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
145	crash	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
146	crash	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
147	crash	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
148	crash	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
149	crash	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
150	crash	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
151	crash	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
152	crash	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
153	crash	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
154	crash	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
155	crash	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
156	crash	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
157	crash	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
158	crash	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
159	crash	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
160	crash	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
161	crash	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
162	crash	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	FAIL	MDEV-9422
163	crash	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
164	crash	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
165	crash	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
166	crash	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	FAIL	MDEV-9422
167	crash	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
168	crash	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	FAIL	MDEV-9422
169	crash	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
170	crash	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	FAIL	MDEV-9422
171	crash	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
172	crash	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	FAIL	MDEV-9422
173	crash	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	FAIL	MDEV-9422
174	crash	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
175	crash	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	FAIL	MDEV-9422
176	crash	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
177	crash	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
178	crash	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
179	crash	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
180	crash	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
181	crash	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
182	crash	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
183	crash	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
184	crash	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
185	crash	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
186	crash	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
187	crash	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
188	crash	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
189	crash	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
190	crash	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
191	crash	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
192	crash	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
193	crash	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
194	crash	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
195	crash	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
196	crash	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
197	crash	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
198	crash	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
199	crash	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
200	crash	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
201	crash	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
202	crash	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
203	crash	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
204	crash	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
205	crash	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
206	crash	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
207	crash	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
208	crash	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
209	crash	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
210	crash	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
211	crash	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
212	crash	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
213	crash	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
214	crash	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
215	crash	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
216	crash	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
217	crash	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
218	crash	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
219	crash	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
220	crash	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
221	crash	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
222	crash	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
223	crash	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
224	crash	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
225	crash	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
226	crash	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
227	crash	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
228	crash	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
229	crash	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
230	crash	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
231	crash	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
232	crash	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
233	crash	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
234	crash	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
235	crash	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
236	crash	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
237	crash	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
238	crash	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
239	crash	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
240	crash	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
241	crash	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	FAIL	MDEV-9422
242	crash	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	FAIL	MDEV-9422
243	crash	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
244	crash	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
245	crash	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	FAIL	MDEV-9422
246	crash	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
247	crash	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
248	crash	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	FAIL	MDEV-9422
249	crash	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	FAIL	MDEV-9422

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
250	crash	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
251	crash	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	FAIL	MDEV-9422
252	crash	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
253	crash	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
254	crash	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	FAIL	MDEV-9422
255	crash	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
256	crash	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	FAIL	MDEV-9422
257	crash	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
258	crash	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
259	crash	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
260	crash	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
261	crash	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
262	crash	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
263	crash	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
264	crash	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
265	crash	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
266	crash	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
267	crash	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
268	crash	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
269	crash	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
270	crash	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
271	crash	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
272	crash	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	

#	type	pages ize	OLD version	encryp ted	compres sed		NEW version	encryp ted	compres sed	reado nly	res ult	notes
273	cras h	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
274	cras h	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
275	cras h	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
276	cras h	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
277	cras h	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
278	cras h	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
279	cras h	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
280	cras h	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
281	cras h	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
282	cras h	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
283	cras h	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
284	cras h	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
285	cras h	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
286	cras h	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
287	cras h	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
288	cras h	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
289	cras h	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
290	cras h	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
291	cras h	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
292	cras h	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
293	cras h	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
294	cras h	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
295	cras h	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
296	crash	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
297	crash	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
298	crash	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
299	crash	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
300	crash	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
301	crash	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
302	crash	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
303	crash	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
304	crash	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
305	crash	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
306	crash	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
307	crash	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
308	crash	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
309	crash	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
310	crash	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
311	crash	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
312	crash	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
313	crash	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
314	crash	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
315	crash	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
316	crash	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
317	crash	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
318	crash	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
319	crash	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
320	crash	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
321	crash	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	FAIL	MDEV-9422
322	crash	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
323	crash	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
324	crash	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
325	crash	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
326	crash	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
327	crash	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
328	crash	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
329	crash	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
330	crash	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
331	crash	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
332	crash	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
333	crash	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
334	crash	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	FAIL	MDEV-9422
335	crash	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
336	crash	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
337	crash	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
338	crash	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
339	crash	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
340	crash	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
341	crash	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
342	crash	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
343	crash	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
344	crash	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
345	crash	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
346	crash	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
347	crash	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
348	crash	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
349	crash	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
350	crash	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
351	crash	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
352	crash	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
353	crash	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
354	crash	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
355	crash	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
356	crash	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
357	crash	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
358	crash	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
359	crash	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
360	crash	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
361	crash	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
362	crash	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
363	crash	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
364	crash	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
365	crash	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
366	crash	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
367	crash	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
368	crash	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
369	crash	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
370	crash	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
371	crash	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
372	crash	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
373	crash	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
374	crash	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
375	crash	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
376	crash	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	FAIL	MDEV-9422
377	crash	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
378	crash	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
379	crash	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
380	crash	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
381	crash	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
382	crash	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
383	crash	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
384	crash	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
385	crash	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
386	crash	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	FAIL	MDEV-9422
387	crash	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	FAIL	MDEV-9422

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
388	crash	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
389	crash	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
390	crash	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	FAIL	MDEV-9422
391	crash	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
392	crash	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	FAIL	MDEV-9422
393	crash	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	FAIL	MDEV-9422
394	crash	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	FAIL	MDEV-9422
395	crash	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
396	crash	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
397	crash	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	FAIL	MDEV-9422
398	crash	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
399	crash	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
400	crash	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
401	crash	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
402	crash	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	FAIL	MDEV-9422
403	crash	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	FAIL	MDEV-9422
404	crash	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
405	crash	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	FAIL	MDEV-9422
406	crash	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
407	crash	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	FAIL	MDEV-9422
408	crash	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
409	crash	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	FAIL	MDEV-9422
410	crash	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	FAIL	MDEV-9422

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
411	crash	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
412	crash	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
413	crash	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	FAIL	MDEV-9422
414	crash	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
415	crash	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
416	crash	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	FAIL	MDEV-9422
417	crash	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
418	crash	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
419	crash	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
420	crash	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
421	crash	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
422	crash	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
423	crash	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
424	crash	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
425	crash	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
426	crash	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
427	crash	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
428	crash	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
429	crash	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
430	crash	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
431	crash	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
432	crash	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
433	crash	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
434	crash	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
435	crash	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
436	crash	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
437	crash	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
438	crash	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
439	crash	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
440	crash	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
441	crash	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
442	crash	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
443	crash	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
444	crash	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
445	crash	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
446	crash	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
447	crash	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
448	crash	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
449	crash	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
450	crash	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
451	crash	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
452	crash	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
453	crash	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
454	crash	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	FAIL	MDEV-9422
455	crash	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
456	crash	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
457	crash	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
458	crash	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
459	crash	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
460	crash	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
461	crash	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
462	crash	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
463	crash	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
464	crash	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
465	crash	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
466	crash	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
467	crash	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
468	crash	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
469	crash	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
470	crash	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
471	crash	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
472	crash	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
473	crash	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
474	crash	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
475	crash	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
476	crash	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
477	crash	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
478	crash	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
479	crash	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
480	crash	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
481	crash	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
482	crash	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
483	crash	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
484	crash	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
485	crash	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
486	crash	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
487	crash	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
488	crash	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
489	crash	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
490	crash	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
491	crash	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
492	crash	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
493	crash	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
494	crash	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
495	crash	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
496	crash	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
497	crash	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
498	crash	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
499	crash	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
500	crash	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
501	crash	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
502	crash	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
503	crash	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
504	crash	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
505	crash	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
506	crash	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
507	crash	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
508	crash	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
509	crash	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
510	crash	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
511	crash	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
512	crash	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
513	crash	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
514	crash	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
515	crash	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
516	crash	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
517	crash	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
518	crash	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
519	crash	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
520	crash	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
521	crash	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
522	crash	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
523	crash	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
524	crash	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
525	crash	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
526	crash	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
527	crash	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
528	crash	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
529	crash	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
530	crash	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
531	crash	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
532	crash	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
533	crash	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
534	crash	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
535	crash	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
536	crash	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
537	crash	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
538	crash	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
539	crash	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
540	crash	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
541	crash	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
542	crash	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
543	crash	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
544	crash	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
545	crash	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
546	crash	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
547	crash	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
548	crash	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
549	crash	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
550	crash	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
551	crash	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
552	crash	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
553	crash	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
554	crash	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
555	crash	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
556	crash	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
557	crash	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
558	crash	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
559	crash	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
560	crash	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
561	crash	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
562	crash	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
563	crash	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
564	crash	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
565	crash	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
566	crash	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
567	crash	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
568	crash	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
569	crash	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
570	crash	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
571	crash	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
572	crash	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
573	crash	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
574	crash	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
575	crash	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
576	crash	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
577	crash	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
578	crash	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
579	crash	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
580	crash	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
581	crash	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
582	crash	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
583	crash	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
584	crash	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
585	crash	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
586	crash	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
587	crash	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
588	crash	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
589	crash	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
590	crash	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
591	crash	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
592	crash	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
593	crash	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
594	crash	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
595	crash	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
596	crash	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
597	crash	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
598	crash	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
599	crash	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
600	crash	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
601	crash	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
602	crash	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
603	crash	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
604	crash	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
605	crash	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
606	crash	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
607	crash	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
608	crash	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
609	crash	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
610	crash	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
611	crash	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
612	crash	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
613	crash	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
614	crash	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
615	crash	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
616	crash	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
617	crash	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
618	crash	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
619	crash	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
620	crash	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
621	crash	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
622	crash	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
623	crash	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
624	crash	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
625	crash	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
626	crash	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
627	crash	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
628	crash	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
629	crash	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
630	crash	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
631	crash	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
632	crash	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
633	crash	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
634	crash	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
635	crash	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
636	crash	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
637	crash	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
638	crash	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
639	crash	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
640	crash	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
641	normal	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
642	normal	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
643	normal	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
644	normal	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
645	normal	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
646	normal	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
647	normal	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
648	normal	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
649	normal	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
650	normal	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
651	normal	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
652	normal	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
653	normal	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
654	normal	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
655	normal	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
656	normal	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
657	normal	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
658	normal	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
659	normal	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
660	normal	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
661	normal	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
662	normal	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
663	normal	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
664	normal	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
665	normal	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
666	normal	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
667	normal	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
668	normal	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
669	normal	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
670	normal	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
671	normal	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
672	normal	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
673	normal	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
674	normal	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
675	normal	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
676	normal	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
677	normal	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
678	normal	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
679	normal	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
680	normal	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
681	normal	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
682	normal	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
683	normal	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
684	normal	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
685	normal	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
686	normal	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
687	normal	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
688	normal	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
689	normal	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
690	normal	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
691	normal	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
692	normal	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
693	normal	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
694	normal	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
695	normal	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
696	normal	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
697	normal	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
698	normal	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
699	normal	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
700	normal	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
701	normal	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
702	normal	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
703	normal	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
704	normal	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
705	normal	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
706	normal	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
707	normal	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
708	normal	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
709	normal	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
710	normal	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
711	normal	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
712	normal	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
713	normal	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
714	normal	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
715	normal	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
716	normal	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
717	normal	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
718	normal	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
719	normal	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
720	normal	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
721	normal	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
722	normal	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
723	normal	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
724	normal	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
725	normal	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
726	normal	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
727	normal	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
728	normal	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
729	normal	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
730	normal	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
731	normal	64	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
732	normal	64	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
733	normal	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
734	normal	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
735	normal	64	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
736	normal	64	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
737	normal	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
738	normal	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
739	normal	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
740	normal	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
741	normal	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
742	normal	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
743	normal	4	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
744	normal	4	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
745	normal	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
746	normal	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
747	normal	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
748	normal	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
749	normal	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
750	normal	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
751	normal	4	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
752	normal	4	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
753	normal	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
754	normal	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
755	normal	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
756	normal	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
757	normal	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
758	normal	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
759	normal	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
760	normal	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
761	normal	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
762	normal	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
763	normal	8	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
764	normal	8	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
765	normal	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
766	normal	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
767	normal	8	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
768	normal	8	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
769	normal	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
770	normal	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
771	normal	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
772	normal	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
773	normal	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
774	normal	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
775	normal	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
776	normal	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
777	normal	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
778	normal	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
779	normal	16	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
780	normal	16	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
781	normal	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
782	normal	16	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
783	normal	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
784	normal	16	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
785	normal	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
786	normal	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
787	normal	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
788	normal	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
789	normal	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
790	normal	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
791	normal	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
792	normal	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
793	normal	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
794	normal	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
795	normal	32	10.1.10 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
796	normal	32	10.1.10 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
797	normal	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
798	normal	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
799	normal	32	10.1.10 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
800	normal	32	10.1.10 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
801	normal	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	

#	type	pages ize	OLD version	encryp ted	compres sed		NEW version	encryp ted	compres sed	reado nly	res ult	notes
802	norm al	4	10.1.10 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
803	norm al	4	10.1.10 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
804	norm al	4	10.1.10 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
805	norm al	4	10.1.10 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
806	norm al	4	10.1.10 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
807	norm al	4	10.1.10 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
808	norm al	4	10.1.10 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
809	norm al	4	10.1.10 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
810	norm al	4	10.1.10 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
811	norm al	4	10.1.10 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
812	norm al	4	10.1.10 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
813	norm al	4	10.1.10 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
814	norm al	4	10.1.10 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
815	norm al	4	10.1.10 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
816	norm al	4	10.1.10 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
817	norm al	8	10.1.10 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
818	norm al	8	10.1.10 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
819	norm al	8	10.1.10 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
820	norm al	8	10.1.10 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
821	norm al	8	10.1.10 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
822	norm al	8	10.1.10 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
823	norm al	8	10.1.10 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
824	norm al	8	10.1.10 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	

#	type	pages ize	OLD version	encryp ted	compres sed		NEW version	encryp ted	compres sed	reado nly	res ult	notes
825	norm al	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
826	norm al	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
827	norm al	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
828	norm al	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
829	norm al	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
830	norm al	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
831	norm al	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
832	norm al	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
833	norm al	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
834	norm al	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
835	norm al	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
836	norm al	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
837	norm al	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
838	norm al	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
839	norm al	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
840	norm al	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
841	norm al	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
842	norm al	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
843	norm al	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
844	norm al	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
845	norm al	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
846	norm al	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
847	norm al	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	

#	type	pages ize	OLD version	encryp ted	compres sed		NEW version	encryp ted	compres sed	reado nly	res ult	notes
848	norm al	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
849	norm al	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
850	norm al	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
851	norm al	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
852	norm al	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
853	norm al	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
854	norm al	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
855	norm al	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
856	norm al	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
857	norm al	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
858	norm al	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
859	norm al	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
860	norm al	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
861	norm al	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
862	norm al	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
863	norm al	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
864	norm al	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
865	norm al	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
866	norm al	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
867	norm al	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
868	norm al	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
869	norm al	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
870	norm al	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	

#	type	pages ize	OLD version	encryp ted	compres sed		NEW version	encryp ted	compres sed	reado nly	res ult	notes
871	norm al	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
872	norm al	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
873	norm al	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
874	norm al	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
875	norm al	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
876	norm al	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
877	norm al	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
878	norm al	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
879	norm al	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
880	norm al	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
881	norm al	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
882	norm al	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
883	norm al	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
884	norm al	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
885	norm al	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
886	norm al	8	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
887	norm al	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
888	norm al	8	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
889	norm al	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
890	norm al	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
891	norm al	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
892	norm al	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
893	norm al	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
894	normal	8	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
895	normal	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
896	normal	8	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
897	normal	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
898	normal	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
899	normal	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
900	normal	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
901	normal	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
902	normal	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
903	normal	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
904	normal	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
905	normal	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
906	normal	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
907	normal	32	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
908	normal	32	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
909	normal	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
910	normal	32	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
911	normal	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
912	normal	32	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
913	normal	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
914	normal	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
915	normal	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
916	normal	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
917	normal	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
918	normal	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
919	normal	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
920	normal	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
921	normal	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
922	normal	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
923	normal	64	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
924	normal	64	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
925	normal	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
926	normal	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
927	normal	64	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
928	normal	64	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
929	normal	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
930	normal	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
931	normal	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
932	normal	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
933	normal	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
934	normal	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
935	normal	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
936	normal	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
937	normal	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
938	normal	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
939	normal	16	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
940	normal	16	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
941	normal	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
942	normal	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
943	normal	16	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
944	normal	16	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
945	normal	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
946	normal	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
947	normal	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
948	normal	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
949	normal	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
950	normal	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
951	normal	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
952	normal	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
953	normal	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
954	normal	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
955	normal	4	10.1.10 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
956	normal	4	10.1.10 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
957	normal	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
958	normal	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
959	normal	4	10.1.10 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
960	normal	4	10.1.10 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
961	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
962	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
963	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
964	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
965	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
966	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
967	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
968	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
969	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
970	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
971	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
972	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
973	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
974	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
975	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
976	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
977	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
978	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
979	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
980	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
981	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
982	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
983	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
984	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
985	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
986	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
987	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
988	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
989	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
990	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
991	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
992	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
993	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
994	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
995	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
996	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
997	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
998	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
999	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1000	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1001	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1002	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1003	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1004	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1005	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1006	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1007	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1008	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1009	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1010	normal	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1011	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1012	normal	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1013	normal	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1014	normal	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1015	normal	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1016	normal	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1017	normal	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1018	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1019	normal	64	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1020	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1021	normal	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1022	normal	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1023	normal	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1024	normal	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1025	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1026	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1027	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1028	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1029	normal	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1030	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1031	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	

#	type	pages ize	OLD version	encryp ted	compres sed		NEW version	encryp ted	compres sed	reado nly	res ult	notes
103 2	norm al	32	10.1.20 (InnoDB plugin)	-	zlib	= >	10.1.22 (inbuilt)	on	zlib	-	OK	
103 3	norm al	32	10.1.20 (InnoDB plugin)	-	-	= >	10.1.22 (inbuilt)	on	-	-	OK	
103 4	norm al	32	10.1.20 (InnoDB plugin)	on	-	= >	10.1.22 (inbuilt)	-	-	-	OK	
103 5	norm al	32	10.1.20 (InnoDB plugin)	-	-	= >	10.1.22 (inbuilt)	-	-	-	OK	
103 6	norm al	32	10.1.20 (InnoDB plugin)	on	-	= >	10.1.22 (inbuilt)	on	-	-	OK	
103 7	norm al	32	10.1.20 (InnoDB plugin)	on	zlib	= >	10.1.22 (inbuilt)	-	-	-	OK	
103 8	norm al	32	10.1.20 (InnoDB plugin)	-	zlib	= >	10.1.22 (inbuilt)	-	-	-	OK	
103 9	norm al	32	10.1.20 (InnoDB plugin)	on	zlib	= >	10.1.22 (inbuilt)	on	-	-	OK	
104 0	norm al	32	10.1.20 (InnoDB plugin)	-	zlib	= >	10.1.22 (inbuilt)	on	-	-	OK	
104 1	norm al	64	10.1.20 (InnoDB plugin)	on	zlib	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
104 2	norm al	64	10.1.20 (InnoDB plugin)	-	zlib	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
104 3	norm al	64	10.1.20 (InnoDB plugin)	-	zlib	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
104 4	norm al	64	10.1.20 (InnoDB plugin)	on	zlib	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
104 5	norm al	64	10.1.20 (InnoDB plugin)	on	zlib	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
104 6	norm al	64	10.1.20 (InnoDB plugin)	-	zlib	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
104 7	norm al	64	10.1.20 (InnoDB plugin)	on	zlib	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
104 8	norm al	64	10.1.20 (InnoDB plugin)	-	zlib	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
104 9	norm al	64	10.1.20 (InnoDB plugin)	on	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
105 0	norm al	64	10.1.20 (InnoDB plugin)	on	-	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
105 1	norm al	64	10.1.20 (InnoDB plugin)	-	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
105 2	norm al	64	10.1.20 (InnoDB plugin)	-	-	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
105 3	norm al	64	10.1.20 (InnoDB plugin)	-	-	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
105 4	norm al	64	10.1.20 (InnoDB plugin)	-	-	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1055	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1056	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1057	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1058	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1059	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1060	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1061	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1062	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1063	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1064	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1065	normal	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1066	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1067	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1068	normal	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1069	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1070	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1071	normal	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1072	normal	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1073	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1074	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1075	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1076	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1077	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1078	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1079	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1080	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1081	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1082	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1083	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1084	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1085	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1086	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1087	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1088	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1089	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1090	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1091	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1092	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1093	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1094	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1095	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1096	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1097	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1098	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1099	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1100	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1101	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1102	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1103	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1104	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1105	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1106	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1107	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1108	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1109	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1110	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1111	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1112	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1113	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1114	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1115	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1116	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1117	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1118	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1119	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1120	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1121	normal	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1122	normal	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1123	normal	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1124	normal	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1125	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1126	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1127	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1128	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1129	normal	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1130	normal	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1131	normal	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1132	normal	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1133	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1134	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1135	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1136	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1137	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1138	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1139	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1140	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1141	normal	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1142	normal	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1143	normal	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1144	normal	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1145	normal	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1146	normal	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	

#	type	pages ize	OLD version	encryp ted	compres sed		NEW version	encryp ted	compres sed	reado nly	res ult	notes
114 7	norm al	64	10.1.20 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
114 8	norm al	64	10.1.20 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
114 9	norm al	64	10.1.20 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
115 0	norm al	64	10.1.20 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
115 1	norm al	64	10.1.20 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
115 2	norm al	64	10.1.20 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
115 3	norm al	32	10.1.20 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
115 4	norm al	32	10.1.20 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
115 5	norm al	32	10.1.20 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
115 6	norm al	32	10.1.20 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
115 7	norm al	32	10.1.20 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
115 8	norm al	32	10.1.20 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
115 9	norm al	32	10.1.20 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	on	-	-	OK	
116 0	norm al	32	10.1.20 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	-	-	-	OK	
116 1	norm al	32	10.1.20 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
116 2	norm al	32	10.1.20 (inbuilt)	on	zlib	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
116 3	norm al	32	10.1.20 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
116 4	norm al	32	10.1.20 (inbuilt)	-	zlib	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
116 5	norm al	32	10.1.20 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
116 6	norm al	32	10.1.20 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
116 7	norm al	32	10.1.20 (inbuilt)	-	-	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
116 8	norm al	32	10.1.20 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
116 9	norm al	4	10.1.20 (inbuilt)	on	-	= >	10.1.22 (InnoDB plugin)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1170	normal	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1171	normal	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1172	normal	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1173	normal	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1174	normal	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1175	normal	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1176	normal	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1177	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1178	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1179	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1180	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1181	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1182	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1183	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1184	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1185	normal	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1186	normal	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1187	normal	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1188	normal	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1189	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1190	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	
1191	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	zlib	-	OK	
1192	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1193	normal	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1194	normal	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1195	normal	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1196	normal	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1197	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1198	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1199	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
1200	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
1201	normal	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1202	normal	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1203	normal	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1204	normal	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1205	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1206	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1207	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1208	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1209	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1210	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1211	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1212	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1213	normal	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1214	normal	8	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1215	normal	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1216	normal	8	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1217	normal	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1218	normal	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1219	normal	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1220	normal	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1221	normal	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1222	normal	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1223	normal	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1224	normal	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1225	normal	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1226	normal	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1227	normal	32	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1228	normal	32	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1229	normal	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1230	normal	32	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1231	normal	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1232	normal	32	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1233	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1234	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1235	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1236	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1237	normal	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1238	normal	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1239	normal	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1240	normal	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1241	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1242	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1243	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1244	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1245	normal	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1246	normal	4	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1247	normal	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1248	normal	4	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1249	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1250	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1251	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1252	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1253	normal	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1254	normal	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1255	normal	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1256	normal	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1257	normal	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1258	normal	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1259	normal	16	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1260	normal	16	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1261	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1262	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1263	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1264	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1265	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1266	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1267	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1268	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1269	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1270	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	on	-	-	OK	
1271	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1272	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.1.22 (inbuilt)	-	-	-	OK	
1273	normal	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1274	normal	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
1275	normal	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1276	normal	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
1277	normal	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1278	normal	64	10.1.20 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	
1279	normal	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	-	zlib	-	OK	
1280	normal	64	10.1.20 (inbuilt)	on	-	=>	10.1.22 (inbuilt)	on	zlib	-	OK	

Upgrade from MySQL 5.6

Tested revision

dbd1d7ea8e96a2b4cff89ec889494700d634b3a3

Test date

2017-03-07 18:37:29

Summary (PASS)

All tests passed

Details

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1	crash	16	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
2	crash	16	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
3	crash	4	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
4	crash	4	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
5	crash	8	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
6	crash	8	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
7	crash	4	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
8	crash	4	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
9	crash	16	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
10	crash	16	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
11	crash	8	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
12	crash	8	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
13	normal	4	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
14	normal	4	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
15	normal	16	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
16	normal	16	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
17	normal	8	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	-	-	-	OK	
18	normal	8	5.6.35 (inbuilt)	-	-	=>	10.1.22 (InnoDB plugin)	on	-	-	OK	
19	normal	8	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
20	normal	8	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
21	normal	4	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
22	normal	4	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	
23	normal	16	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	-	-	-	OK	
24	normal	16	5.6.35 (inbuilt)	-	-	=>	10.1.22 (inbuilt)	on	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.1.25 Release Upgrade Tests

Upgrade from 10.0

Tested revision

da7604a2943f633df3b193e26ee20110bae9fa7a

Test date

2017-07-04 18:46:27

Summary (PASS)

All tests passed

Details

#	type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
1	crash	4	10.0.28 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
2	crash	4	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
3	crash	4	10.0.28 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
4	crash	4	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
5	crash	8	10.0.28 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
6	crash	8	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
7	crash	8	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
8	crash	8	10.0.28 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
9	crash	16	10.0.28 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
10	crash	16	10.0.28 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
11	crash	16	10.0.28 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	

#	type	pages ize	OLD version	file format	encry pted	compre ssed		NEW version	file format	encry pted	compre ssed	readon ly	res ult	not es
12	cras h	16	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	on	-	-	OK	
13	nor mal	4	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	on	-	-	OK	
14	nor mal	4	10.0.28 (inbuilt)	Antelo pe	-	-	=>	10.1.25 (inbuilt)	Antelo pe	on	-	-	OK	
15	nor mal	4	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	-	-	-	OK	
16	nor mal	4	10.0.28 (inbuilt)	Antelo pe	-	-	=>	10.1.25 (inbuilt)	Antelo pe	-	-	-	OK	
17	nor mal	8	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	-	-	-	OK	
18	nor mal	8	10.0.28 (inbuilt)	Antelo pe	-	-	=>	10.1.25 (inbuilt)	Antelo pe	-	-	-	OK	
19	nor mal	8	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	on	-	-	OK	
20	nor mal	8	10.0.28 (inbuilt)	Antelo pe	-	-	=>	10.1.25 (inbuilt)	Antelo pe	on	-	-	OK	
21	nor mal	16	10.0.28 (inbuilt)	Antelo pe	-	-	=>	10.1.25 (inbuilt)	Antelo pe	on	-	-	OK	
22	nor mal	16	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	on	-	-	OK	
23	nor mal	16	10.0.28 (inbuilt)	Antelo pe	-	-	=>	10.1.25 (inbuilt)	Antelo pe	-	-	-	OK	
24	nor mal	16	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	-	-	-	OK	

Upgrade from 10.1

Tested revision

da7604a2943f633df3b193e26ee20110bae9fa7a

Test date

2017-07-04 19:37:09

Summary (FAIL)

MDEV-13112, MDEV-13113, one unknown failure

Details

#	type	page size	OLD version	file forma t	encry pted	compre ssed		NEW version	file forma t	encry pted	compre ssed	read only	res ult	notes
1	cras h	64	10.1.20 (inbuilt)	Antelo pe	-	zlib	=>	10.1.25 (inbuilt)	Antelo pe	-	zlib	-	OK	
2	cras h	64	10.1.20 (inbuilt)	Barrac uda	-	zlib	=>	10.1.25 (inbuilt)	Barrac uda	-	zlib	-	OK	

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
3	crash	64	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	FAIL	MDEV-13112(31)
4	crash	64	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
5	crash	64	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
6	crash	64	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
7	crash	64	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
8	crash	64	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
9	crash	8	10.1.20 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
10	crash	8	10.1.20 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
11	crash	8	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
12	crash	8	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	FAIL	MDEV-13112(14)
13	crash	8	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
14	crash	8	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
15	crash	8	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
16	crash	8	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
17	crash	16	10.1.20 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	FAIL	MDEV-13113
18	crash	16	10.1.20 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
19	crash	16	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	FAIL	MDEV-13113
20	crash	16	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
21	crash	16	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
22	crash	16	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	FAIL	MDEV-13113
23	crash	16	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
24	crash	16	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	FAIL	MDEV-13113
25	crash	32	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
26	crash	32	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
27	crash	32	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
28	crash	32	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
29	crash	32	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	FAIL	MDEV-13112(23)
30	crash	32	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
31	crash	32	10.1.20 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
32	crash	32	10.1.20 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
33	crash	4	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	FAIL	MDEV-13112(7)
34	crash	4	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
35	crash	4	10.1.20 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
36	crash	4	10.1.20 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
37	crash	4	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
38	crash	4	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
39	crash	4	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
40	crash	4	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
41	normal	8	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
42	normal	8	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
43	normal	8	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
44	normal	8	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
45	normal	8	10.1.20 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
46	normal	8	10.1.20 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
47	normal	8	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	OK	
48	normal	8	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
49	normal	64	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
50	normal	64	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	OK	
51	normal	64	10.1.20 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
52	normal	64	10.1.20 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
53	normal	64	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
54	normal	64	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
55	normal	64	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
56	normal	64	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
57	normal	32	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
58	normal	32	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	OK	
59	normal	32	10.1.20 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
60	normal	32	10.1.20 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
61	normal	32	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
62	normal	32	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
63	normal	32	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
64	normal	32	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
65	normal	16	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
66	normal	16	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	OK	
67	normal	16	10.1.20 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
68	normal	16	10.1.20 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
69	normal	16	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
70	normal	16	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
71	normal	16	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
72	normal	16	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
73	normal	4	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
74	normal	4	10.1.20 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
75	normal	4	10.1.20 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
76	normal	4	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
77	normal	4	10.1.20 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
78	normal	4	10.1.20 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	OK	
79	normal	4	10.1.20 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
80	normal	4	10.1.20 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	

Upgrade from MySQL 5.6

Tested revision

da7604a2943f633df3b193e26ee20110bae9fa7a

Test date

2017-07-04 22:22:10

Summary (PASS)

All tests passed

Details

#	type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
1	normal	16	5.6.35 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
2	normal	16	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
3	normal	16	5.6.35 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
4	normal	16	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
5	normal	4	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
6	normal	4	5.6.35 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	

#	type	pages ize	OLD version	file format	encry pted	compre ssed		NEW version	file format	encry pted	compre ssed	readon ly	res ult	not es
7	nor mal	4	5.6.35 (inbuilt)	Antelop e	-	-	=>	10.1.25 (inbuilt)	Antelop e	on	-	-	OK	
8	nor mal	4	5.6.35 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	on	-	-	OK	
9	nor mal	8	5.6.35 (inbuilt)	Antelop e	-	-	=>	10.1.25 (inbuilt)	Antelop e	-	-	-	OK	
1 0	nor mal	8	5.6.35 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	-	-	-	OK	
1 1	nor mal	8	5.6.35 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	on	-	-	OK	
1 2	nor mal	8	5.6.35 (inbuilt)	Antelop e	-	-	=>	10.1.25 (inbuilt)	Antelop e	on	-	-	OK	

Crash recovery

Tested revision

da7604a2943f633df3b193e26ee20110bae9fa7a

Test date

2017-07-04 22:47:13

Summary (FAIL)

MDEV-13112

Details

#	typ e	page size	OLD version	file format	encry pted	compre ssed		NEW version	file format	encry pted	compre ssed	read only	res ult	notes
1	cra sh	16	10.1.25 (inbuilt)	Antelo pe	on	-	=>	10.1.25 (inbuilt)	Antelo pe	on	-	-	OK	
2	cra sh	16	10.1.25 (inbuilt)	Barrac uda	on	-	=>	10.1.25 (inbuilt)	Barrac uda	on	-	-	OK	
3	cra sh	16	10.1.25 (inbuilt)	Antelo pe	-	-	=>	10.1.25 (inbuilt)	Antelo pe	-	-	-	OK	
4	cra sh	16	10.1.25 (inbuilt)	Barrac uda	-	-	=>	10.1.25 (inbuilt)	Barrac uda	-	-	-	OK	
5	cra sh	16	10.1.25 (inbuilt)	Antelo pe	on	zlib	=>	10.1.25 (inbuilt)	Antelo pe	on	zlib	-	OK	
6	cra sh	16	10.1.25 (inbuilt)	Barrac uda	on	zlib	=>	10.1.25 (inbuilt)	Barrac uda	on	zlib	-	FAI L	MDEV-13112(39)
7	cra sh	16	10.1.25 (inbuilt)	Antelo pe	-	zlib	=>	10.1.25 (inbuilt)	Antelo pe	-	zlib	-	OK	
8	cra sh	16	10.1.25 (inbuilt)	Barrac uda	-	zlib	=>	10.1.25 (inbuilt)	Barrac uda	-	zlib	-	OK	
9	cra sh	4	10.1.25 (inbuilt)	Antelo pe	on	-	=>	10.1.25 (inbuilt)	Antelo pe	on	-	-	OK	

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
10	crash	4	10.1.25 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
11	crash	4	10.1.25 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
12	crash	4	10.1.25 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
13	crash	4	10.1.25 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	FAIL	MDEV-13112(54)
14	crash	4	10.1.25 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
15	crash	4	10.1.25 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
16	crash	4	10.1.25 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
17	crash	32	10.1.25 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
18	crash	32	10.1.25 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	FAIL	MDEV-13112(36)
19	crash	32	10.1.25 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
20	crash	32	10.1.25 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
21	crash	32	10.1.25 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
22	crash	32	10.1.25 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
23	crash	32	10.1.25 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
24	crash	32	10.1.25 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
25	crash	64	10.1.25 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	
26	crash	64	10.1.25 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
27	crash	64	10.1.25 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
28	crash	64	10.1.25 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
29	crash	64	10.1.25 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	FAIL	MDEV-13112(21)
30	crash	64	10.1.25 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
31	crash	64	10.1.25 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
32	crash	64	10.1.25 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
33	crash	8	10.1.25 (inbuilt)	Barracuda	on	zlib	=>	10.1.25 (inbuilt)	Barracuda	on	zlib	-	FAIL	MDEV-13112(97)
34	crash	8	10.1.25 (inbuilt)	Antelope	on	zlib	=>	10.1.25 (inbuilt)	Antelope	on	zlib	-	OK	
35	crash	8	10.1.25 (inbuilt)	Antelope	-	zlib	=>	10.1.25 (inbuilt)	Antelope	-	zlib	-	OK	
36	crash	8	10.1.25 (inbuilt)	Barracuda	-	zlib	=>	10.1.25 (inbuilt)	Barracuda	-	zlib	-	OK	
37	crash	8	10.1.25 (inbuilt)	Antelope	on	-	=>	10.1.25 (inbuilt)	Antelope	on	-	-	OK	
38	crash	8	10.1.25 (inbuilt)	Barracuda	on	-	=>	10.1.25 (inbuilt)	Barracuda	on	-	-	OK	
39	crash	8	10.1.25 (inbuilt)	Barracuda	-	-	=>	10.1.25 (inbuilt)	Barracuda	-	-	-	OK	
40	crash	8	10.1.25 (inbuilt)	Antelope	-	-	=>	10.1.25 (inbuilt)	Antelope	-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.1.26 Release Upgrade Tests

10.1.30 Release Upgrade Tests

Tested revision

461cf3e5a3c2d346d75b1407b285f8daf9d01f67

Test date

2017-12-23

Summary

Some tests failed with known bug MDEV-13112. Undo upgrade from 10.0.18 failed because the old server hang on shutdown (a known problem with old servers). Crash upgrade from 10.1.10 (4K, zlib, XtraDB, Barracuda) failed (MDEV-14759).

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(52)
recovery	4	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(59)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	32	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(20)
recovery	8	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(71)
recovery	16	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	16	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	4	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	32	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	64	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	8	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	16	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(17)
crash	4	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(43)
crash	32	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(36)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	32	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(47)
crash	8	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(60)
crash	16	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.29 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.29 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.29 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.29 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(33)
crash	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(19)
crash	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(44)
crash	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(29)
crash	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(66)
crash	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.30 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.30 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.30 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.30 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
normal	4	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
normal	8	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
normal	16	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
normal	4	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
normal	8	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
crash	16	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
crash	4	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
crash	8	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
crash	16	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
crash	4	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
crash	8	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
undo	16	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
undo	4	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
undo	8	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
undo	16	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
undo	4	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
undo	8	10.0.33 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
crash	16	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
crash	4	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
crash	8	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
crash	16	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
crash	4	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
crash	8	10.0.14 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	16	10.0.18 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
normal	16	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
normal	4	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
normal	8	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
normal	16	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
normal	4	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
normal	8	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
crash	16	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	4	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
crash	8	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
crash	16	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
crash	4	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
crash	8	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
undo	16	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
undo	4	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
undo	8	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		on	-	-	OK	
undo	16	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
undo	4	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	
undo	8	5.6.38 (inbuilt)		-	-	=>	10.1.30 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.1.32 Release Upgrade Tests

Tested revision

4d83b01537e3f1eb257a8e9a6416938e03adea59

Test date

2018-04-17 01:44:10

Summary

Known bug MDEV-13112. A few upgrades from MySQL and old MariaDB fail because the old versions hang on shutdown.

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(70)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	4	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(60)
recovery	32	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(28)
recovery	32	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(38)
recovery	8	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(76)
recovery	16	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	64	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	4	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	32	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	64	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	8	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	16	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	8	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	16	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(54)
crash	4	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(41)
crash	32	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(50)
crash	32	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(35)
crash	8	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(43)
crash	16	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(23)
crash	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(52)
crash	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(33)
crash	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(21)
crash	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13112(49)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.1.32 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.1.32 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.1.32 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.1.32 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
normal	8	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
normal	16	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
normal	16	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
normal	8	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
crash	4	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
crash	8	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
crash	16	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
crash	16	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
crash	4	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
crash	8	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	16	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	4	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	16	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
crash	4	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
crash	8	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
crash	16	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
crash	16	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
crash	4	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
crash	8	10.0.14 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
normal	4	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
crash	4	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
crash	8	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
crash	16	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
crash	16	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
crash	4	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
crash	8	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	4	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	8	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		on	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	OK	
undo	8	5.6.39 (inbuilt)		-	-	=>	10.1.32 (inbuilt)		-	-	-	FAIL	TEST_FAILURE

This page is licensed: CC BY-SA / Gnu FDL

10.1.36 Release Upgrade Tests

Tested revision

38e5dc0f772daecca1d2681885d3d85414eb6826

Test date

2018-10-03 11:54:43

Summary

Tests passed

Details

type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
recovery	16	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	4	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	8	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	16	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	

type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
undo-recovery	64	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	4	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	32	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	64	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	8	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	16	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	

type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
normal	64	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.1.36 (inbuilt)	Barracuda	on	-	-	OK	

type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
crash	8	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.1.36 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.1.36 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.1.36 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		-	-	-	OK	
normal	8	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		-	-	-	OK	

type	pages ize	OLD version	file format	encry pted	compre ssed		NEW version	file format	encry pted	compre ssed	reado nly	res ult	not es
normal	16	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		-	-	-	OK	
normal	16	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		on	-	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		on	-	-	OK	
normal	8	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		on	-	-	OK	
crash	4	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		-	-	-	OK	
crash	8	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		-	-	-	OK	
crash	16	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		-	-	-	OK	
crash	16	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		on	-	-	OK	
crash	4	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		on	-	-	OK	
crash	8	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		on	-	-	OK	
undo	16	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		-	-	-	OK	
undo	4	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		-	-	-	OK	
undo	8	10.0.36 (inbuilt)		-	-	=>	10.1.36 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.2.12 Release Upgrade Tests

Tested revision

d361401bc26f49480daec8b0307afff6edae2ecc

Test date

2018-01-07 07:34:24

Summary

Some tests failed due to MDEV-13103, MDEV-13094 as usual. Crash recovery with encryption also fails due to MDEV-13103.

Details

type	page size	OLD version	file forma t	encry pted	compr essed		NEW version	file forma t	encry pted	compr essed	read only	res ult	notes
recover y	16	10.2.12 (inbuilt)	Barra cuda	on	-	=>	10.2.12 (inbuilt)	Barra cuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	4	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	8	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	16	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	64	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	4	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	32	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	64	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	8	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	16	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	8	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	16	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.12 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.12 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.12 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.12 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	8	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	16	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	16	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
normal	4	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	16	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	4	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	8	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	16	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
undo	4	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
undo	8	10.0.33 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	64	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	8	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	16	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	32	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	4	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
normal	8	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
normal	16	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
normal	64	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
normal	32	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
crash	64	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
crash	8	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
crash	16	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
crash	32	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
crash	4	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
crash	4	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
crash	8	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
crash	16	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
crash	64	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
crash	32	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	16	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	4	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	32	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	64	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	8	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	16	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
undo	4	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
undo	64	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
undo	8	5.7.20 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	4	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	8	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	16	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
normal	16	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
normal	4	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
normal	8	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	16	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	4	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	8	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		on	-	-	OK	
undo	16	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
undo	4	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	
undo	8	5.6.38 (inbuilt)		-	-	=>	10.2.12 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.2.13 Release Upgrade Tests

Tested revision

00f0c039d2f4213ccf0a0202349ecb162a799989

Test date

2018-02-15 22:17:40

Summary

Some tests failed due to MDEV-13103, MDEV-13094 as usual.

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	4	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	8	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	16	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	4	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	4	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	32	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	64	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	8	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	16	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	32	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.13 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.13 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.13 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.13 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	8	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	16	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	8	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	16	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	4	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	16	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	4	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	64	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	8	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	32	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	4	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	4	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	8	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	16	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	64	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	32	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
crash	64	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
crash	8	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
crash	16	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
crash	32	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
crash	4	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
crash	4	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
crash	8	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
crash	16	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
crash	64	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
crash	32	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	16	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	4	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	32	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	64	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	8	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	4	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	32	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	64	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	8	5.7.21 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	4	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	8	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		on	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	4	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	
undo	8	5.6.39 (inbuilt)		-	-	=>	10.2.13 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.2.14 Release Upgrade Tests

Tested revision

b3cdafcb93da2e57d49f26f0846dc957458ee72c

Test date

2018-04-17 05:25:43

Summary

Known bugs MDEV-13103, MDEV-13094. A few upgrades from MySQL and old MariaDB fail because the old versions hang on shutdown.

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	4	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	8	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	16	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	16	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	4	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	32	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	64	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	8	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	16	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	4	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.14 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.14 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.14 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.14 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	16	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	16	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	8	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	16	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	4	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	16	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
undo	4	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	8	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	16	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	32	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	4	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	4	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	8	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	16	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	64	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	32	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
crash	64	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
crash	8	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
crash	16	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
crash	32	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
crash	4	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
crash	4	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
crash	8	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
crash	16	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
crash	64	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
crash	32	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	16	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	4	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	32	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	8	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	16	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
undo	4	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
undo	32	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
undo	64	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
undo	8	5.7.21 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	4	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	8	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		on	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	OK	
undo	8	5.6.39 (inbuilt)		-	-	=>	10.2.14 (inbuilt)		-	-	-	FAIL	TEST_FAILURE

This page is licensed: CC BY-SA / Gnu FDL

10.2.17 Release Upgrade Tests

Tested revision

5553d3f1f691a374124686946bd1f1cc4ef9fb45

Test date

2018-08-30 01:21:02

Summary

Known bug MDEV-13094. A few upgrades from MySQL and old MariaDB fail because the old versions hang on shutdown.

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	4	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	8	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	16	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	8	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	4	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	32	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	64	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	8	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	16	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	4	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.16 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.16 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.16 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.16 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.17 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.17 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.17 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.17 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	16	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	16	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	8	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	16	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	4	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	8	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	16	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
undo	4	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
undo	8	10.0.36 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	FAIL	TEST_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	8	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	16	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	32	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	4	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	4	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	8	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	16	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	64	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	32	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
crash	64	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
crash	8	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
crash	16	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
crash	32	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
crash	4	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
crash	4	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
crash	8	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
crash	16	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
crash	64	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
crash	32	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	16	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	4	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	32	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	8	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	16	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
undo	4	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
undo	32	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
undo	64	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
undo	8	5.7.23 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	4	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	8	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	16	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
normal	16	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	4	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
normal	8	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	16	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	4	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	OK	
undo	8	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	16	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	OK	
undo	8	5.6.41 (inbuilt)		-	-	=>	10.2.17 (inbuilt)		-	-	-	FAIL	TEST_FAILURE

This page is licensed: CC BY-SA / Gnu FDL

10.2.18 Release Upgrade Tests

Tested revision

2dfb4a8abe3af501f8a6780ed782a2eee5e6f6d5

Test date

2018-10-03 13:37:05

Summary

Few unrelated failures due to bugs in previous versions

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	4	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	8	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	16	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	16	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	4	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	32	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	64	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	8	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	16	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	64	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.2.18 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.2.18 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.2.18 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.2.18 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	8	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	16	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	16	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
normal	8	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	16	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	4	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	8	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	16	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
undo	8	10.0.36 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
normal	64	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	8	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	16	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	32	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	4	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	4	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
normal	8	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
normal	16	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
normal	64	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
crash	64	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
crash	8	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
crash	16	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
crash	32	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
crash	4	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
crash	4	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
crash	8	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
crash	16	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
crash	64	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
crash	32	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	16	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	4	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	32	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	64	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	8	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	16	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
undo	4	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
undo	32	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
undo	64	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
undo	8	5.7.23 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	4	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	8	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
normal	16	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
normal	4	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
normal	8	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	16	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	4	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	8	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		on	-	-	OK	
undo	16	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	OK	
undo	8	5.6.41 (inbuilt)		-	-	=>	10.2.18 (inbuilt)		-	-	-	FAIL	TEST_FAILURE

This page is licensed: CC BY-SA / Gnu FDL

10.2.5 Pre-release Upgrade Tests

Upgrade from 10.0

Tested revision

b56262f69677fdb158b3d19dd8848e5802b2dd27

Test date

2017-03-27 02:05:09

Summary (PASS)

All tests passed

Details

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1	normal	4	10.0.29 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
2	normal	4	10.0.29 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
3	normal	8	10.0.29 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
4	normal	8	10.0.29 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
5	normal	16	10.0.29 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
6	normal	16	10.0.29 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
7	normal	4	10.0.29 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
8	normal	4	10.0.29 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
9	normal	8	10.0.29 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
10	normal	8	10.0.29 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
11	normal	16	10.0.29 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
12	normal	16	10.0.29 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	

Upgrade from 10.1

Tested revision

b56262f69677fdb158b3d19dd8848e5802b2dd27

Test date

2017-03-27 02:31:34

Summary (FAIL)

One test failed: MDEV-12388

Details

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
2	normal	64	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
3	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
4	normal	64	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
5	normal	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
6	normal	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
7	normal	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
8	normal	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
9	normal	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
10	normal	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
11	normal	64	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
12	normal	64	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
13	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
14	normal	64	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
15	normal	64	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
16	normal	64	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
17	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
18	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
19	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
20	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
21	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
22	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
23	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
24	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
25	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
26	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
27	normal	8	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
28	normal	8	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
29	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
30	normal	8	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
31	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
32	normal	8	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
33	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
34	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
35	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
36	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
37	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
38	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
39	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
40	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
41	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
42	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
43	normal	16	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
44	normal	16	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
45	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
46	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
47	normal	16	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
48	normal	16	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
49	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
50	normal	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
51	normal	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
52	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
53	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
54	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
55	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
56	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
57	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
58	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
59	normal	32	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
60	normal	32	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
61	normal	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
62	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
63	normal	32	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
64	normal	32	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
65	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
66	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
67	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
68	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
69	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
70	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
71	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	FAIL	MDEV-12388
72	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
73	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
74	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
75	normal	4	10.1.20 (InnoDB plugin)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
76	normal	4	10.1.20 (InnoDB plugin)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
77	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
78	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
79	normal	4	10.1.20 (InnoDB plugin)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
80	normal	4	10.1.20 (InnoDB plugin)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
81	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
82	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
83	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
84	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
85	normal	8	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
86	normal	8	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
87	normal	8	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
88	normal	8	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
89	normal	8	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
90	normal	8	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
91	normal	8	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
92	normal	8	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
93	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
94	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
95	normal	8	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
96	normal	8	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
97	normal	64	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
98	normal	64	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
99	normal	64	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
100	normal	64	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
101	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
102	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
103	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
104	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
105	normal	64	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
106	normal	64	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
107	normal	64	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
108	normal	64	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
109	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
110	normal	64	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
111	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
112	normal	64	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
113	normal	16	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
114	normal	16	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
115	normal	16	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
116	normal	16	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
117	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
118	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
119	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
120	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
121	normal	16	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
122	normal	16	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
123	normal	16	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
124	normal	16	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
125	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
126	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
127	normal	16	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
128	normal	16	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
129	normal	4	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
130	normal	4	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
131	normal	4	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
132	normal	4	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
133	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
134	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
135	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
136	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
137	normal	4	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
138	normal	4	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
139	normal	4	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
140	normal	4	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
141	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
142	normal	4	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
143	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
144	normal	4	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
145	normal	32	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
146	normal	32	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	-	-	OK	
147	normal	32	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
148	normal	32	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	-	-	OK	
149	normal	32	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
150	normal	32	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
151	normal	32	10.1.20 (inbuilt)	on	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
152	normal	32	10.1.20 (inbuilt)	-	zlib	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
153	normal	32	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
154	normal	32	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
155	normal	32	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
156	normal	32	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
157	normal	32	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
158	normal	32	10.1.20 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	
159	normal	32	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	-	zlib	-	OK	
160	normal	32	10.1.20 (inbuilt)	on	-	=>	10.2.5 (inbuilt)	on	zlib	-	OK	

Upgrade from MySQL 5.6

Tested revision

b56262f69677fdb158b3d19dd8848e5802b2dd27

Test date

2017-03-28 14:00:57

Summary (PASS)

All tests passed

Details

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1	normal	16	5.6.35 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
2	normal	16	5.6.35 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
3	normal	4	5.6.35 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
4	normal	4	5.6.35 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
5	normal	8	5.6.35 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
6	normal	8	5.6.35 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	

Upgrade from MySQL 5.7

Tested revision

b56262f69677fdb158b3d19dd8848e5802b2dd27

Test date

2017-03-28 23:04:16

Summary (PASS)

All tests passed

Details

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
1	normal	8	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
2	normal	8	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
3	normal	64	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
4	normal	64	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
5	normal	16	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
6	normal	16	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
7	normal	32	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
8	normal	32	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
9	normal	4	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	

#	type	pagesize	OLD version	encrypted	compressed		NEW version	encrypted	compressed	readonly	result	notes
10	normal	4	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
11	crash	64	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
12	crash	64	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
13	crash	16	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
14	crash	16	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
15	crash	8	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
16	crash	8	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
17	crash	32	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
18	crash	32	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
19	crash	4	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
20	crash	4	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
21	normal	32	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
22	normal	32	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
23	normal	4	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
24	normal	4	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
25	normal	16	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
26	normal	16	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
27	normal	64	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	
28	normal	64	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
29	normal	8	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	-	-	-	OK	
30	normal	8	5.7.17 (inbuilt)	-	-	=>	10.2.5 (inbuilt)	on	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.2.7 Release Upgrade Tests

Upgrade from 10.0

Tested revision

5ff2db7f67401511b30dbd3fc69a1ea87d7e8cc4

Test date

2017-07-10 07:11:04

Summary (PASS)

All tests passed

Details

#	type	pages ize	OLD version	file format	encry pted	compre ssed		NEW version	file format	encry pted	compre ssed	readon ly	res ult	not es
1	nor mal	16	10.0.28 (inbuilt)	Antelop e	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
2	nor mal	16	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
3	nor mal	16	10.0.28 (inbuilt)	Antelop e	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
4	nor mal	16	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
5	nor mal	4	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
6	nor mal	4	10.0.28 (inbuilt)	Antelop e	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
7	nor mal	4	10.0.28 (inbuilt)	Antelop e	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
8	nor mal	4	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
9	nor mal	8	10.0.28 (inbuilt)	Antelop e	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
10	nor mal	8	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
11	nor mal	8	10.0.28 (inbuilt)	Barrac uda	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
12	nor mal	8	10.0.28 (inbuilt)	Antelop e	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	

Upgrade from 10.1

Tested revision

5ff2db7f67401511b30dbd3fc69a1ea87d7e8cc4

Test date

2017-07-10 07:29:38

Summary (PASS)

All tests passed

Details

#	type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
1	normal	64	10.1.24 (inbuilt)	Antelope	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
2	normal	64	10.1.24 (inbuilt)	Barracuda	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
3	normal	64	10.1.24 (inbuilt)	Barracuda	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
4	normal	64	10.1.24 (inbuilt)	Antelope	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
5	normal	64	10.1.24 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
6	normal	64	10.1.24 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
7	normal	64	10.1.24 (inbuilt)	Antelope	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
8	normal	64	10.1.24 (inbuilt)	Barracuda	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
9	normal	8	10.1.24 (inbuilt)	Antelope	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
10	normal	8	10.1.24 (inbuilt)	Barracuda	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
11	normal	8	10.1.24 (inbuilt)	Antelope	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
12	normal	8	10.1.24 (inbuilt)	Barracuda	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
13	normal	8	10.1.24 (inbuilt)	Barracuda	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
14	normal	8	10.1.24 (inbuilt)	Antelope	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
15	normal	8	10.1.24 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
16	normal	8	10.1.24 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
17	normal	16	10.1.24 (inbuilt)	Barracuda	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
18	normal	16	10.1.24 (inbuilt)	Antelope	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
19	normal	16	10.1.24 (inbuilt)	Barracuda	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
20	normal	16	10.1.24 (inbuilt)	Antelope	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
21	normal	16	10.1.24 (inbuilt)	Antelope	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
22	normal	16	10.1.24 (inbuilt)	Barracuda	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
23	normal	16	10.1.24 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
24	normal	16	10.1.24 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
25	normal	32	10.1.24 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
26	normal	32	10.1.24 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
27	normal	32	10.1.24 (inbuilt)	Barracuda	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
28	normal	32	10.1.24 (inbuilt)	Antelope	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
29	normal	32	10.1.24 (inbuilt)	Barracuda	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
30	normal	32	10.1.24 (inbuilt)	Antelope	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
31	normal	32	10.1.24 (inbuilt)	Barracuda	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
32	normal	32	10.1.24 (inbuilt)	Antelope	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
33	normal	4	10.1.24 (inbuilt)	Barracuda	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
34	normal	4	10.1.24 (inbuilt)	Antelope	on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
35	normal	4	10.1.24 (inbuilt)	Barracuda	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
36	normal	4	10.1.24 (inbuilt)	Antelope	-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
37	normal	4	10.1.24 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
38	normal	4	10.1.24 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
39	normal	4	10.1.24 (inbuilt)	Antelope	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
40	normal	4	10.1.24 (inbuilt)	Barracuda	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
41	normal	8	10.1.20 (inbuilt)	Antelope	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
42	normal	8	10.1.20 (inbuilt)	Barracuda	on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
43	normal	8	10.1.20 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
44	normal	8	10.1.20 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	

#	type	pages ize	OLD version	file format	encry pted	compre ssed		NEW version	file format	encry pted	compre ssed	reado nly	res ult	not es
4 5	nor mal	8	10.1.20 (inbuilt)	Barrac uda	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	
4 6	nor mal	8	10.1.20 (inbuilt)	Antelop e	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	
4 7	nor mal	8	10.1.20 (inbuilt)	Barrac uda	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
4 8	nor mal	8	10.1.20 (inbuilt)	Antelop e	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
4 9	nor mal	64	10.1.20 (inbuilt)	Antelop e	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
5 0	nor mal	64	10.1.20 (inbuilt)	Barrac uda	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
5 1	nor mal	64	10.1.20 (inbuilt)	Barrac uda	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	
5 2	nor mal	64	10.1.20 (inbuilt)	Antelop e	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	
5 3	nor mal	64	10.1.20 (inbuilt)	Barrac uda	on	-	= >	10.2.7 (inbuilt)		on	-	-	OK	
5 4	nor mal	64	10.1.20 (inbuilt)	Antelop e	on	-	= >	10.2.7 (inbuilt)		on	-	-	OK	
5 5	nor mal	64	10.1.20 (inbuilt)	Antelop e	-	-	= >	10.2.7 (inbuilt)		-	-	-	OK	
5 6	nor mal	64	10.1.20 (inbuilt)	Barrac uda	-	-	= >	10.2.7 (inbuilt)		-	-	-	OK	
5 7	nor mal	32	10.1.20 (inbuilt)	Antelop e	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
5 8	nor mal	32	10.1.20 (inbuilt)	Barrac uda	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
5 9	nor mal	32	10.1.20 (inbuilt)	Antelop e	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	
6 0	nor mal	32	10.1.20 (inbuilt)	Barrac uda	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	
6 1	nor mal	32	10.1.20 (inbuilt)	Barrac uda	on	-	= >	10.2.7 (inbuilt)		on	-	-	OK	
6 2	nor mal	32	10.1.20 (inbuilt)	Antelop e	on	-	= >	10.2.7 (inbuilt)		on	-	-	OK	
6 3	nor mal	32	10.1.20 (inbuilt)	Antelop e	-	-	= >	10.2.7 (inbuilt)		-	-	-	OK	
6 4	nor mal	32	10.1.20 (inbuilt)	Barrac uda	-	-	= >	10.2.7 (inbuilt)		-	-	-	OK	
6 5	nor mal	16	10.1.20 (inbuilt)	Antelop e	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
6 6	nor mal	16	10.1.20 (inbuilt)	Barrac uda	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
6 7	nor mal	16	10.1.20 (inbuilt)	Antelop e	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	

#	type	pages ize	OLD version	file format	encry pted	compre ssed		NEW version	file format	encry pted	compre ssed	readon ly	res ult	not es
6 8	nor mal	16	10.1.20 (inbuilt)	Barrac uda	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	
6 9	nor mal	16	10.1.20 (inbuilt)	Barrac uda	-	-	= >	10.2.7 (inbuilt)		-	-	-	OK	
7 0	nor mal	16	10.1.20 (inbuilt)	Antelop e	-	-	= >	10.2.7 (inbuilt)		-	-	-	OK	
7 1	nor mal	16	10.1.20 (inbuilt)	Antelop e	on	-	= >	10.2.7 (inbuilt)		on	-	-	OK	
7 2	nor mal	16	10.1.20 (inbuilt)	Barrac uda	on	-	= >	10.2.7 (inbuilt)		on	-	-	OK	
7 3	nor mal	4	10.1.20 (inbuilt)	Barrac uda	on	-	= >	10.2.7 (inbuilt)		on	-	-	OK	
7 4	nor mal	4	10.1.20 (inbuilt)	Antelop e	on	-	= >	10.2.7 (inbuilt)		on	-	-	OK	
7 5	nor mal	4	10.1.20 (inbuilt)	Antelop e	-	-	= >	10.2.7 (inbuilt)		-	-	-	OK	
7 6	nor mal	4	10.1.20 (inbuilt)	Barrac uda	-	-	= >	10.2.7 (inbuilt)		-	-	-	OK	
7 7	nor mal	4	10.1.20 (inbuilt)	Antelop e	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
7 8	nor mal	4	10.1.20 (inbuilt)	Barrac uda	on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	OK	
7 9	nor mal	4	10.1.20 (inbuilt)	Antelop e	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	
8 0	nor mal	4	10.1.20 (inbuilt)	Barrac uda	-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	OK	

Upgrade from 10.2

Tested revision

5ff2db7f67401511b30dbd3fc69a1ea87d7e8cc4

Test date

2017-07-10 09:34:15

Summary (FAIL)

MDEV-13094, MDEV-13103

Details

#	typ e	page size	OLD version	file forma t	encry pted	compr essed		NEW version	file forma t	encry pted	compr essed	read only	res ult	notes
1	cras h	64	10.2.6 (inbuilt)		-	zlib	= >	10.2.7 (inbuilt)		-	zlib	-	FAI L	KNOWN_BUGS MDEV-13094(1)
2	cras h	64	10.2.6 (inbuilt)		on	zlib	= >	10.2.7 (inbuilt)		on	zlib	-	FAI L	KNOWN_BUGS MDEV-13103(1)

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
3	crash	64	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
4	crash	64	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
5	crash	8	10.2.6 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
6	crash	8	10.2.6 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
7	crash	8	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
8	crash	8	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
9	crash	16	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
10	crash	16	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
11	crash	16	10.2.6 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(2)
12	crash	16	10.2.6 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
13	crash	32	10.2.6 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
14	crash	32	10.2.6 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
15	crash	32	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
16	crash	32	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
17	crash	4	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
18	crash	4	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
19	crash	4	10.2.6 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
20	crash	4	10.2.6 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(2)
21	normal	32	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
22	normal	32	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
23	normal	32	10.2.6 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
24	normal	32	10.2.6 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
25	normal	4	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
26	normal	4	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
27	normal	4	10.2.6 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(2)
28	normal	4	10.2.6 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
29	normal	16	10.2.6 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
30	normal	16	10.2.6 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(2)
31	normal	16	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
32	normal	16	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
33	normal	64	10.2.6 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
34	normal	64	10.2.6 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
35	normal	64	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
36	normal	64	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
37	normal	8	10.2.6 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	OK	
38	normal	8	10.2.6 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	OK	
39	normal	8	10.2.6 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
40	normal	8	10.2.6 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	

Upgrade from MySQL 5.6

Tested revision

5ff2db7f67401511b30dbd3fc69a1ea87d7e8cc4

Test date

2017-07-10 10:34:00

Summary (PASS)

All tests passed

Details

#	type	pages ize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
1	normal	16	5.6.35 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
2	normal	16	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
3	normal	16	5.6.35 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
4	normal	16	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
5	normal	4	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
6	normal	4	5.6.35 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
7	normal	4	5.6.35 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
8	normal	4	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
9	normal	8	5.6.35 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
10	normal	8	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
11	normal	8	5.6.35 (inbuilt)	Barracuda	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
12	normal	8	5.6.35 (inbuilt)	Antelope	-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	

Upgrade from MySQL 5.7

Tested revision

5ff2db7f67401511b30dbd3fc69a1ea87d7e8cc4

Test date

2017-07-10 10:52:42

Summary (PASS)

All tests passed

Details

#	type	pages ize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
1	crash	64	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
2	crash	64	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
3	crash	8	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	

#	type	pagesize	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	readonly	result	notes
4	crash	8	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
5	crash	16	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
6	crash	16	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
7	crash	32	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
8	crash	32	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
9	crash	4	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
10	crash	4	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
11	normal	16	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
12	normal	16	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
13	normal	32	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
14	normal	32	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
15	normal	64	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
16	normal	64	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
17	normal	8	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
18	normal	8	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	
19	normal	4	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	OK	
20	normal	4	5.7.17 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		on	-	-	OK	

Crash recovery

Tested revision

5ff2db7f67401511b30dbd3fc69a1ea87d7e8cc4

Test date

2017-07-10 11:23:26

Summary (FAIL)

MDEV-13094, MDEV-13103

Details

#	type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
1	crash	16	10.2.7 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
2	crash	16	10.2.7 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(3)
3	crash	16	10.2.7 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
4	crash	16	10.2.7 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
5	crash	4	10.2.7 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
6	crash	4	10.2.7 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(3)
7	crash	4	10.2.7 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
8	crash	4	10.2.7 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
9	crash	32	10.2.7 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
10	crash	32	10.2.7 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
11	crash	32	10.2.7 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
12	crash	32	10.2.7 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
13	crash	64	10.2.7 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
14	crash	64	10.2.7 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
15	crash	64	10.2.7 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
16	crash	64	10.2.7 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(2)
17	crash	8	10.2.7 (inbuilt)		-	zlib	=>	10.2.7 (inbuilt)		-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(2)
18	crash	8	10.2.7 (inbuilt)		on	zlib	=>	10.2.7 (inbuilt)		on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
19	crash	8	10.2.7 (inbuilt)		-	-	=>	10.2.7 (inbuilt)		-	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)
20	crash	8	10.2.7 (inbuilt)		on	-	=>	10.2.7 (inbuilt)		on	-	-	FAIL	KNOWN_BUGS MDEV-13094(2)

This page is licensed: CC BY-SA / Gnu FDL

10.3.10 Release Upgrade Tests

Tested revision

bad2f1569da57c4a81cc84ec2f4a79924df9c8d6

Test date

2018-10-04 23:04:05

Summary

Known bug MDEV-13094. A few upgrades from MySQL and old MariaDB fail because the old versions hang on shutdown

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	4	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	8	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	16	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	64	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.3.10 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.3.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	4	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	32	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	64	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	8	10.3.10 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	16	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.3.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	64	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	4	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.18 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.18 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.18 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.18 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.36 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.36 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.36 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.36 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.10 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.10 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.10 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.10 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	8	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	16	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	16	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	8	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	4	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	8	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	16	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
undo	4	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
undo	8	10.0.36 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
normal	64	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	8	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	16	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	32	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	4	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	8	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	16	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	64	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	32	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	16	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	4	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	32	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	64	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	8	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	16	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
undo	4	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
undo	32	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
undo	64	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
undo	8	5.7.23 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	4	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	8	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	16	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	
normal	16	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	4	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
normal	8	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	16	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	
undo	4	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	16	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	8	5.6.41 (inbuilt)		-	-	=>	10.3.10 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.3.3-beta Release Upgrade Tests

Tested revision

52c40b406d5a68bf774a75dc6fa9ef1c166b22b1

Test date

2017-12-24

Summary

Some tests failed due to the known bug MDEV-13103. Crash upgrade from 10.3.2 fails due to an incompatible change (removal of MLOG_UNDO* record types), allowed for Alpha=>Beta transition.

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	4	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	8	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	8	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	16	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	64	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	4	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	32	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	64	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	8	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	16	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	16	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
crash	4	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	4	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
crash	32	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
crash	32	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
crash	8	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	8	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
crash	16	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	16	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	4	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	4	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	32	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	32	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	8	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	8	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	16	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.3.2 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.3.2 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.3.2 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.3.2 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.11 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.11 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.11 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.11 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.3 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.3 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.3 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.3 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	8	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	16	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	16	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
normal	4	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
normal	8	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	16	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	4	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	8	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	16	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	4	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	8	10.0.33 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	64	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	8	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	16	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	32	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	4	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	4	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
normal	8	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
normal	16	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
normal	64	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
normal	32	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	16	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	4	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	32	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	64	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	8	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	4	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	32	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	64	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	8	5.7.20 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	4	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	8	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	16	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
normal	16	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
normal	4	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
normal	8	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	16	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	4	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	8	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		on	-	-	OK	
undo	16	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	4	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	
undo	8	5.6.38 (inbuilt)		-	-	=>	10.3.3 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.3.4-beta Release Upgrade Tests

Tested revision

8f102b584d4e8f02da924f9be094014136eb453f

Test date

2018-01-19 17:39:10

Summary

All major upgrades fail with MDEV-14990 (non-InnoDB issue, general server). Also known bugs MDEV-13103, MDEV-13094

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	4	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	8	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	16	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	16	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	4	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	32	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	64	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	8	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	16	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	4	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.3.3 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.3.3 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.3.3 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.3.3 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	4	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	8	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	16	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	4	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	8	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	16	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	16	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	4	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	64	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	8	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	16	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	4	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	4	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	32	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	32	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	8	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	8	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	16	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	32	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	64	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.2.12 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	32	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	64	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.2.12 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	4	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	32	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	64	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	8	10.2.12 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	16	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	4	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	32	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	64	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	8	10.2.12 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	16	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	4	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	8	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	16	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	4	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	8	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	16	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	32	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	64	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.1.30 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	32	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	64	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.1.30 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	4	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	32	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	64	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	8	10.1.30 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	16	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	4	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	32	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	64	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	8	10.1.30 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.4 (inbuilt)	Barracuda	on	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.4 (inbuilt)	Barracuda	-	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.4 (inbuilt)	Barracuda	on	zlib	-	FAIL	UPGRADE_FAILURE
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.4 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
normal	4	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.0.33 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	64	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	8	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	16	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	32	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	4	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	4	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
normal	8	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
normal	16	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
normal	64	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	16	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	4	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	32	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	64	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	8	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	16	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
undo	4	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
undo	32	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
undo	64	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
undo	8	5.7.21 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	4	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	8	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	16	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
normal	16	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
normal	4	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
normal	8	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	16	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	4	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	8	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		on	-	-	FAIL	UPGRADE_FAILURE
undo	16	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
undo	4	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE
undo	8	5.6.39 (inbuilt)		-	-	=>	10.3.4 (inbuilt)		-	-	-	FAIL	UPGRADE_FAILURE

This page is licensed: CC BY-SA / Gnu FDL

10.3.5-gamma Release Upgrade Tests

Tested revision

583eb96c2492adb87e88a014b24eb0724fb00257

Test date

2018-03-11 12:40:20

Summary

Known bugs MDEV-13103, MDEV-13094. A few upgrades from MySQL and old MariaDB fail because the old versions hang on shutdown

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	4	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	8	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	16	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	32	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	4	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	32	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	64	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	8	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	TEST_FAILURE
undo-recovery	16	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	64	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.3.4 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.3.4 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.3.4 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.3.4 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	4	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.13 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.13 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.31 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.5 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.5 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.5 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.5 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	8	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	16	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	16	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	8	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	4	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	16	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
undo	4	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
normal	64	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	8	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	16	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	32	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	4	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	8	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	16	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	64	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	32	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	16	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	4	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	32	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	64	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	8	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	16	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
undo	4	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
undo	32	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
undo	64	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
undo	8	5.7.21 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	4	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	FAIL	TEST_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		on	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	
undo	8	5.6.39 (inbuilt)		-	-	=>	10.3.5 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.3.6-gamma Release Upgrade Tests

Tested revision

560743198604caf677c543db9719cef871df09ce

Test date

2018-04-17 10:50:21

Summary

One new (previously unknown) bug MDEV-15912. Known bugs MDEV-13103, MDEV-13094. One failure identified as a deadlock of yet unknown origin, could be a side-effect of known bugs. A few upgrades from MySQL and old MariaDB fail because the old versions hang on shutdown.

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	4	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
recovery	4	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	32	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	8	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	8	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
recovery	16	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.3.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	4	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	64	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.3.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	4	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	32	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	64	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	8	10.3.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
undo-recovery	16	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.3.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	8	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.3.5 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.3.5 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.3.5 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.3.5 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	8	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.14 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.14 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.14 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.14 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	FAIL	KNOWN_BUGS MDEV-13094(1)
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	KNOWN_BUGS MDEV-13094(1)
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	KNOWN_BUGS MDEV-13103(1)
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.32 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.32 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	64	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	FAIL	SERVER_DEADLOCKED
undo	8	10.1.32 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.32 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.6 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.6 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.6 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	FAIL	UPGRADE_FAILURE
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.6 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	8	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	16	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	16	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	4	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	8	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	16	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	4	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	16	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
undo	4	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
undo	8	10.0.34 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	FAIL	TEST_FAILURE

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	64	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	8	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	16	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	32	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	4	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	4	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	8	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	16	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	64	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	32	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	16	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	4	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	32	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	64	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	8	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	16	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
undo	4	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
undo	32	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
undo	64	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
undo	8	5.7.21 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	
normal	16	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	4	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
normal	8	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	16	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	4	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	OK	
undo	8	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	16	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	8	5.6.39 (inbuilt)		-	-	=>	10.3.6 (inbuilt)		-	-	-	OK	

This page is licensed: CC BY-SA / Gnu FDL

10.3.9 Release Upgrade Tests

Tested revision

ca26f91bcaa21933147974c823852a2e1c2e2bd7

Test date

2018-08-30 07:09:44

Summary

Known bug MDEV-13094. A few upgrades from MySQL and old MariaDB fail because the old versions hang on shutdown.

Details

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	16	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
recovery	16	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	4	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
recovery	4	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	32	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
recovery	64	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	8	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
recovery	8	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
recovery	16	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
recovery	16	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	4	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
recovery	4	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	32	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
recovery	64	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
recovery	8	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
recovery	8	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	16	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	4	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	32	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	64	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	8	10.3.9 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo-recovery	16	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo-recovery	4	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	32	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	64	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	8	10.3.9 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo-recovery	16	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	4	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	32	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	64	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	8	10.3.9 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo-recovery	16	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	4	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	32	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	64	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo-recovery	8	10.3.9 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	4	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.3.8 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.3.8 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.3.8 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.3.8 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.17 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.17 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.17 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.17 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
crash	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
crash	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	16	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.2.6 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	8	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
undo	4	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.35 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	16	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	4	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	32	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	64	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
normal	8	10.1.13 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	16	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	4	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	32	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
normal	64	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	8	10.1.10 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	10.1.10 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	-	=>	10.3.9 (inbuilt)	Barracuda	on	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	-	=>	10.3.9 (inbuilt)	Barracuda	-	-	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	on	zlib	=>	10.3.9 (inbuilt)	Barracuda	on	zlib	-	OK	
undo	16	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	4	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	32	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	64	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
undo	8	10.1.22 (inbuilt)	Barracuda	-	zlib	=>	10.3.9 (inbuilt)	Barracuda	-	zlib	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	8	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	16	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	16	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
normal	4	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
normal	8	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	16	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	4	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	8	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	16	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
undo	4	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
undo	8	10.0.36 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	16	10.0.14 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
normal	4	10.0.14 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
normal	8	10.0.14 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	16	10.0.18 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	4	10.0.18 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	8	10.0.18 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
normal	64	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	8	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	16	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	32	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	4	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	4	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
normal	8	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
normal	16	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
normal	64	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
normal	32	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	16	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	4	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	32	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	64	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	8	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	16	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
undo	4	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
undo	32	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
undo	64	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
undo	8	5.7.23 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	4	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	8	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	16	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
normal	16	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	

type	page size	OLD version	file format	encrypted	compressed		NEW version	file format	encrypted	compressed	read only	result	notes
normal	4	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
normal	8	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	16	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	4	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	FAIL	TEST_FAILURE
undo	8	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		on	-	-	OK	
undo	16	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	OK	
undo	4	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	FAIL	TEST_FAILURE
undo	8	5.6.41 (inbuilt)		-	-	=>	10.3.9 (inbuilt)		-	-	-	FAIL	TEST_FAILURE

This page is licensed: CC BY-SA / Gnu FDL

QA Datasets

See DBT-3 Dataset.

Information on the standardized datasets and database dumps used for DBT-3 Dataset tests.

See DBT-3 Queries.

Information on the standardized datasets and database dumps used for DBT-3 Queries tests.

See OpenStreetMap Dataset.

Information on the standardized datasets and database dumps used for OpenStreetMap tests.

See osmdb06.sql.

Information on the standardized datasets and database dumps used for OpenStreetMap.

This page is licensed: CC BY-SA / Gnu FDL

DBT-3 Dataset

This page describes our setup for DBT-3 tests. A very cogent resource on the DBT3 Benchmark is also available. See also dbt-3-queries

Get and compile DBT3

- Get DBT3
- Make sure that you have pg_ctl and createdb of PostgreSQL installed. On Ubuntu look for postgresql and postgresql-client and make sure that pg_ctl is in your PATH, for instance:

```
PATH=/usr/lib/postgresql/8.4/bin:$PATH
```

- Compile DBT3

```
gunzip -c dbt3-1.9.tar.gz | tar xvf -
cd dbt3-1.9
./configure
make
cd src/dbgen
make
```

- Generate data:

```
./dbgen -s30
```

DDL

Substitute `${ENGINE}` with the storage engine you want to use:

```
CREATE TABLE supplier (
  s_suppkey  INTEGER PRIMARY KEY,
  s_name CHAR(25),
  s_address VARCHAR(40),
  s_nationkey INTEGER,
  s_phone CHAR(15),
  s_acctbal REAL,
  s_comment VARCHAR(101)) Engine ${ENGINE};

CREATE TABLE part (
  p_partkey INTEGER PRIMARY KEY,
  p_name VARCHAR(55),
  p_mfgr CHAR(25),
  p_brand CHAR(10),
  p_type VARCHAR(25),
  p_size INTEGER,
  p_container CHAR(10),
  p_retailprice REAL,
  p_comment VARCHAR(23)) Engine ${ENGINE};

CREATE TABLE partsupp (
  ps_partkey INTEGER,
  ps_suppkey INTEGER,
  ps_availqty INTEGER,
  ps_supplycost REAL,
  ps_comment VARCHAR(199),
  PRIMARY KEY (ps_partkey, ps_suppkey)) Engine ${ENGINE};

CREATE TABLE customer (
  c_custkey INTEGER PRIMARY KEY,
  c_name VARCHAR(25),
  c_address VARCHAR(40),
  c_nationkey INTEGER,
  c_phone CHAR(15),
  c_acctbal REAL,
  c_mktsegment CHAR(10),
  c_comment VARCHAR(117))  Engine ${ENGINE};

CREATE TABLE orders (
  o_orderkey INTEGER PRIMARY KEY,
  o_custkey INTEGER,
  o_orderstatus CHAR(1),
  o_totalprice REAL,
  o_orderDATE DATE,
  o_orderpriority CHAR(15),
  o_clerk CHAR(15),
  o_shippriority INTEGER,
  o_comment VARCHAR(79))  Engine ${ENGINE};

CREATE TABLE lineitem (
  l_orderkey INTEGER,
  l_partkey INTEGER,
  l_suppkey INTEGER,
  l_linenumber INTEGER,
  l_quantity REAL,
  l_extendedprice REAL,
  l_discount REAL,
  l_tax REAL,
  l_returnflag CHAR(1),
  l_linestatus CHAR(1),
  l_shipDATE DATE,
  l_commitDATE DATE,
  l_receiptDATE DATE,
  l_shipinstruct CHAR(25),
  l_shipmode CHAR(10),
  l_comment VARCHAR(44),
  PRIMARY KEY (l_orderkey, l_linenumber)) Engine ${ENGINE};
```

```
CREATE TABLE nation (
  n_nationkey INTEGER PRIMARY KEY,
  n_name CHAR(25),
  n_regionkey INTEGER,
  n_comment VARCHAR(152)) Engine ${ENGINE};

CREATE TABLE region (
  r_regionkey INTEGER PRIMARY KEY,
  r_name CHAR(25),
  r_comment VARCHAR(152)) Engine ${ENGINE};

CREATE TABLE time_statistics (
  task_name VARCHAR(40),
  s_time TIMESTAMP,
  e_time TIMESTAMP,
  int_time INTEGER)  Engine ${ENGINE};
```

Load data

Substitute `${DATA_DIR}` with the path to your generated data.

```
LOAD DATA LOCAL INFILE '${DATA_DIR}/supplier.tbl' into table supplier fields terminated by '|';
  LOAD DATA LOCAL INFILE '${DATA_DIR}/part.tbl' into table part fields terminated by '|';
  LOAD DATA LOCAL INFILE '${DATA_DIR}/partsupp.tbl' into table partsupp fields terminated by '|';
  LOAD DATA LOCAL INFILE '${DATA_DIR}/customer.tbl' into table customer fields terminated by '|';
  LOAD DATA LOCAL INFILE '${DATA_DIR}/orders.tbl' into table orders fields terminated by '|';
  LOAD DATA LOCAL INFILE '${DATA_DIR}/lineitem.tbl' into table lineitem fields terminated by '|';
  LOAD DATA LOCAL INFILE '${DATA_DIR}/nation.tbl' into table nation fields terminated by '|';
  LOAD DATA LOCAL INFILE '${DATA_DIR}/region.tbl' into table region fields terminated by '|';
```

Indexes we need

```
ALTER TABLE lineitem
  ADD INDEX i_l_shipdate(l_shipdate),
  ADD INDEX i_l_suppkey_partkey (l_partkey, l_suppkey),
  ADD INDEX i_l_partkey (l_partkey),
  ADD INDEX i_l_suppkey (l_suppkey),
  ADD INDEX i_l_receiptdate (l_receiptdate),
  ADD INDEX i_l_orderkey (l_orderkey),
  ADD INDEX i_l_orderkey_quantity (l_orderkey, l_quantity),
  ADD INDEX i_l_commitdate (l_commitdate);

CREATE INDEX i_c_nationkey ON customer (c_nationkey);

ALTER TABLE orders
  ADD INDEX i_o_orderdate (o_orderdate),
  ADD INDEX i_o_custkey (o_custkey);

CREATE INDEX i_s_nationkey ON supplier (s_nationkey);

ALTER TABLE partsupp
  ADD INDEX i_ps_partkey (ps_partkey),
  ADD INDEX i_ps_suppkey (ps_suppkey);

CREATE INDEX i_n_regionkey ON nation (n_regionkey);
```

Analyze tables

```
ANALYZE TABLE supplier;
ANALYZE TABLE part;
ANALYZE TABLE partsupp;
ANALYZE TABLE customer;
ANALYZE TABLE orders;
ANALYZE TABLE lineitem;
ANALYZE TABLE nation;
ANALYZE TABLE region;
```

This page is licensed: CC BY-SA / Gnu FDL

DBT-3 Queries

Q1

See MDEV-4309 (just speeding up temptable-based GROUP BY execution). Optimizer seems to make a good choice here.

```
SELECT
        l_returnflag,
        l_linestatus,
        SUM(l_quantity) AS sum_qty,
        SUM(l_extendedprice) AS sum_base_price,
        SUM(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
        SUM(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
        AVG(l_quantity) AS avg_qty,
        AVG(l_extendedprice) AS avg_price,
        AVG(l_discount) AS avg_disc,
        COUNT(*) AS count_order
FROM
        lineitem
WHERE
        l_shipdate <= date_sub('1998-12-01', INTERVAL 63 DAY)
GROUP BY
        l_returnflag,
        l_linestatus
ORDER BY
        l_returnflag,
        l_linestatus
```

Q4

See MDEV-6015.

Applicable optimizations:

- subquery cache brings no benefit because subquery refers to outer_table.pk, which is different for each row
- EXISTS-to-IN is applicable
  - After that, BKA brings slight speedup

Comments on query plan choice

- It seems, we're using the best possible query plan here.

```
SELECT
    o_orderpriority,
    COUNT(*) AS order_count
FROM
    orders
WHERE
    o_orderdate >= '1995-06-06'
    AND o_orderdate < date_add('1995-06-06', INTERVAL 3 MONTH)
    AND EXISTS (
        SELECT
            *
        FROM
            lineitem
        WHERE
            l_orderkey = o_orderkey
            AND l_commitdate < l_receiptdate
    )
GROUP BY
    o_orderpriority
ORDER BY
    o_orderpriority;
```

.

This page is licensed: CC BY-SA / Gnu FDL

OpenStreetMap Dataset

This page describes how to use the OpenStreetMap dataset in testing.

Database Schema

The database schema is available here. To import:

```
mysqladmin create osm
cat osmdb06.sql | mysql osm
```

By default, this schema uses a mixture of InnoDB and MyISAM tables. To convert all tables to Aria:

```
sed -i -e 's/InnoDB/Aria/gi' osmdb06.sql
sed -i -e 's/MyISAM/Aria/gi' osmdb06.sql
```

30 tables are created.

Data

The data is provided in the form of XML files (.OSM files) that require the Java-based Osmosis tool to load into MariaDB. The tool is available from dev.openstreetmap.org. Version 0.36 is known to work.

Various .OSM files are available, including the entire world (>200Gb unzipped) and individual countries.

Data is loaded with the following command-line (in the example, we're using the bulgaria.osm file, replace with the file you choose):

```
chmod +x bin/osmosis
bin/osmosis --read-xml file=bulgaria.osm --write-apidb dbType="mysql" host="localhost:port"
validateSchemaVersion=no database="osm" user="root" password="<password-goes-here>"
```

Data is inserted into 19 tables, as follows:

```
MariaDB [(none)]> use information_schema;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [information_schema]> select TABLE_NAME, TABLE_ROWS from TABLES
    -> where TABLE_ROWS > 0
    -> AND
    -> TABLE_SCHEMA='osm'
    -> ORDER BY TABLE_ROWS DESC;
+--------------------------+------------+
| TABLE_NAME               | TABLE_ROWS |
+--------------------------+------------+
| current_way_nodes        |    1559099 |
| way_nodes                |    1559099 |
| current_nodes            |    1477247 |
| nodes                    |    1477247 |
| node_tags                |     311751 |
| way_tags                 |     287585 |
| ways                     |     100007 |
| current_ways             |     100007 |
| changeset_tags           |      18738 |
| current_relation_members |      14560 |
| relation_members         |      14560 |
| changesets               |       9369 |
| relation_tags            |       3948 |
| current_relations        |        937 |
| relations                |        937 |
| users                    |        537 |
+--------------------------+------------+
16 rows in set (0.00 sec)
```

This page is licensed: CC BY-SA / Gnu FDL

osmdb06.sql

Below is the schema described in the OpenStreetMap Dataset Use article. To use, copy everything in the box below into a file called ‘`osmdb06.sql`’, then continue with the instructions.

```
-- phpMyAdmin SQL Dump
-- version 2.11.9.3
-- http://www.phpmyadmin.net
--
-- Хост: mysql.leonenko.info
-- Час стварэньня: 16 Сак 2009, 15:12
-- Вэрсія сэрвэра: 5.0.67
-- Вэрсія PHP: 5.2.6

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- База дадзеных: `osmapper_belarus`
--

-- --------------------------------------------------------

--
-- Структура табліцы `changesets`
--

DROP TABLE IF EXISTS `changesets`;
CREATE TABLE IF NOT EXISTS `changesets` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `user_id` BIGINT(20) NOT NULL,
  `created_at` DATETIME NOT NULL,
  `min_lat` INT(11) DEFAULT NULL,
  `max_lat` INT(11) DEFAULT NULL,
  `min_lon` INT(11) DEFAULT NULL,
  `max_lon` INT(11) DEFAULT NULL,
  `closed_at` DATETIME NOT NULL,
  `num_changes` INT(11) NOT NULL DEFAULT '0',
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=20103 ;

-- --------------------------------------------------------

--
-- Структура табліцы `changeset_tags`
--

DROP TABLE IF EXISTS `changeset_tags`;
CREATE TABLE IF NOT EXISTS `changeset_tags` (
  `id` BIGINT(64) NOT NULL,
  `k` VARCHAR(255) NOT NULL DEFAULT '',
  `v` VARCHAR(255) NOT NULL DEFAULT '',
  KEY `changeset_tags_id_idx` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `current_nodes`
--

DROP TABLE IF EXISTS `current_nodes`;
CREATE TABLE IF NOT EXISTS `current_nodes` (
  `id` BIGINT(64) NOT NULL AUTO_INCREMENT,
  `latitude` INT(11) NOT NULL,
  `longitude` INT(11) NOT NULL,
  `changeset_id` BIGINT(20) NOT NULL,
  `visible` TINYINT(1) NOT NULL,
  `timestamp` DATETIME NOT NULL,
```

```
  `tile` INT(10) UNSIGNED NOT NULL,
  `version` BIGINT(20) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `current_nodes_timestamp_idx` (`timestamp`),
  KEY `current_nodes_tile_idx` (`tile`),
  KEY `changeset_id` (`changeset_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=348816842 ;

-- --------------------------------------------------------

--
-- Структура табліцы `current_node_tags`
--

DROP TABLE IF EXISTS `current_node_tags`;
CREATE TABLE IF NOT EXISTS `current_node_tags` (
  `id` BIGINT(64) NOT NULL,
  `k` VARCHAR(255) NOT NULL DEFAULT '',
  `v` VARCHAR(255) NOT NULL DEFAULT '',
  PRIMARY KEY  (`id`,`k`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `current_relations`
--

DROP TABLE IF EXISTS `current_relations`;
CREATE TABLE IF NOT EXISTS `current_relations` (
  `id` BIGINT(64) NOT NULL AUTO_INCREMENT,
  `changeset_id` BIGINT(20) NOT NULL,
  `timestamp` DATETIME NOT NULL,
  `visible` TINYINT(1) NOT NULL,
  `version` BIGINT(20) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `current_relations_timestamp_idx` (`timestamp`),
  KEY `changeset_id` (`changeset_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=80283 ;

-- --------------------------------------------------------

--
-- Структура табліцы `current_relation_members`
--

DROP TABLE IF EXISTS `current_relation_members`;
CREATE TABLE IF NOT EXISTS `current_relation_members` (
  `id` BIGINT(64) NOT NULL,
  `member_type` ENUM('node','way','relation') NOT NULL DEFAULT 'node',
  `member_id` BIGINT(11) NOT NULL,
  `member_role` VARCHAR(255) NOT NULL DEFAULT '',
  `sequence_id` INT(11) NOT NULL DEFAULT '0',
  PRIMARY KEY  (`id`,`member_type`,`member_id`,`member_role`,`sequence_id`),
  KEY `current_relation_members_member_idx` (`member_type`,`member_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `current_relation_tags`
--

DROP TABLE IF EXISTS `current_relation_tags`;
CREATE TABLE IF NOT EXISTS `current_relation_tags` (
  `id` BIGINT(64) NOT NULL,
  `k` VARCHAR(255) NOT NULL DEFAULT '',
  `v` VARCHAR(255) NOT NULL DEFAULT '',
  PRIMARY KEY  (`id`,`k`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `current_ways`
--

DROP TABLE IF EXISTS `current_ways`;
CREATE TABLE IF NOT EXISTS `current_ways` (
  `id` BIGINT(64) NOT NULL AUTO_INCREMENT,
  `changeset_id` BIGINT(20) NOT NULL,
  `timestamp` DATETIME NOT NULL,
  `visible` TINYINT(1) NOT NULL,
  `version` BIGINT(20) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `current_ways_timestamp_idx` (`timestamp`),
  KEY `changeset_id` (`changeset_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=31336923 ;

-- --------------------------------------------------------

--
-- Структура табліцы `current_way_nodes`
--

DROP TABLE IF EXISTS `current_way_nodes`;
CREATE TABLE IF NOT EXISTS `current_way_nodes` (
  `id` BIGINT(64) NOT NULL,
  `node_id` BIGINT(64) NOT NULL,
  `sequence_id` BIGINT(11) NOT NULL,
  PRIMARY KEY  (`id`,`sequence_id`),
  KEY `current_way_nodes_node_idx` (`node_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `current_way_tags`
--

DROP TABLE IF EXISTS `current_way_tags`;
CREATE TABLE IF NOT EXISTS `current_way_tags` (
  `id` BIGINT(64) NOT NULL,
  `k` VARCHAR(255) NOT NULL DEFAULT '',
  `v` VARCHAR(255) NOT NULL DEFAULT '',
  PRIMARY KEY  (`id`,`k`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `diary_comments`
--

DROP TABLE IF EXISTS `diary_comments`;
CREATE TABLE IF NOT EXISTS `diary_comments` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `diary_entry_id` BIGINT(20) NOT NULL,
  `user_id` BIGINT(20) NOT NULL,
  `body` text NOT NULL,
  `created_at` DATETIME NOT NULL,
  `updated_at` DATETIME NOT NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `diary_comments_entry_id_idx` (`diary_entry_id`,`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

-- --------------------------------------------------------
```

```
--
-- Структура табліцы `diary_entries`
--

DROP TABLE IF EXISTS `diary_entries`;
CREATE TABLE IF NOT EXISTS `diary_entries` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `user_id` BIGINT(20) NOT NULL,
  `title` VARCHAR(255) NOT NULL,
  `body` text NOT NULL,
  `created_at` DATETIME NOT NULL,
  `updated_at` DATETIME NOT NULL,
  `latitude` DOUBLE DEFAULT NULL,
  `longitude` DOUBLE DEFAULT NULL,
  `language` VARCHAR(3) DEFAULT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Структура табліцы `friends`
--

DROP TABLE IF EXISTS `friends`;
CREATE TABLE IF NOT EXISTS `friends` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `user_id` BIGINT(20) NOT NULL,
  `friend_user_id` BIGINT(20) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `user_id_idx` (`friend_user_id`),
  KEY `friends_user_id_idx` (`user_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Структура табліцы `gps_points`
--

DROP TABLE IF EXISTS `gps_points`;
CREATE TABLE IF NOT EXISTS `gps_points` (
  `altitude` FLOAT DEFAULT NULL,
  `trackid` INT(11) NOT NULL,
  `latitude` INT(11) NOT NULL,
  `longitude` INT(11) NOT NULL,
  `gpx_id` BIGINT(64) NOT NULL,
  `timestamp` DATETIME DEFAULT NULL,
  `tile` INT(10) UNSIGNED NOT NULL,
  KEY `points_gpxid_idx` (`gpx_id`),
  KEY `points_tile_idx` (`tile`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `gpx_files`
--

DROP TABLE IF EXISTS `gpx_files`;
CREATE TABLE IF NOT EXISTS `gpx_files` (
  `id` BIGINT(64) NOT NULL AUTO_INCREMENT,
  `user_id` BIGINT(20) NOT NULL,
  `visible` TINYINT(1) NOT NULL DEFAULT '1',
  `name` VARCHAR(255) NOT NULL DEFAULT '',
  `size` BIGINT(20) DEFAULT NULL,
  `latitude` DOUBLE DEFAULT NULL,
  `longitude` DOUBLE DEFAULT NULL,
  `timestamp` DATETIME NOT NULL,
```

```
  `public` TINYINT(1) NOT NULL DEFAULT '1',
  `description` VARCHAR(255) NOT NULL DEFAULT '',
  `inserted` TINYINT(1) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `gpx_files_timestamp_idx` (`timestamp`),
  KEY `gpx_files_visible_public_idx` (`visible`,`public`),
  KEY `gpx_files_user_id_idx` (`user_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Структура табліцы `gpx_file_tags`
--

DROP TABLE IF EXISTS `gpx_file_tags`;
CREATE TABLE IF NOT EXISTS `gpx_file_tags` (
  `gpx_id` BIGINT(64) NOT NULL DEFAULT '0',
  `tag` VARCHAR(255) NOT NULL,
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  PRIMARY KEY  (`id`),
  KEY `gpx_file_tags_gpxid_idx` (`gpx_id`),
  KEY `gpx_file_tags_tag_idx` (`tag`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Структура табліцы `messages`
--

DROP TABLE IF EXISTS `messages`;
CREATE TABLE IF NOT EXISTS `messages` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `from_user_id` BIGINT(20) NOT NULL,
  `title` VARCHAR(255) NOT NULL,
  `body` text NOT NULL,
  `sent_on` DATETIME NOT NULL,
  `message_read` TINYINT(1) NOT NULL DEFAULT '0',
  `to_user_id` BIGINT(20) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `messages_to_user_id_idx` (`to_user_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Структура табліцы `nodes`
--

DROP TABLE IF EXISTS `nodes`;
CREATE TABLE IF NOT EXISTS `nodes` (
  `id` BIGINT(64) NOT NULL,
  `latitude` INT(11) NOT NULL,
  `longitude` INT(11) NOT NULL,
  `changeset_id` BIGINT(20) NOT NULL,
  `visible` TINYINT(1) NOT NULL,
  `timestamp` DATETIME NOT NULL,
  `tile` INT(10) UNSIGNED NOT NULL,
  `version` BIGINT(20) NOT NULL,
  PRIMARY KEY  (`id`,`version`),
  KEY `nodes_timestamp_idx` (`timestamp`),
  KEY `nodes_tile_idx` (`tile`),
  KEY `changeset_id` (`changeset_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
```

```
-- Структура табліцы `node_tags`
--

DROP TABLE IF EXISTS `node_tags`;
CREATE TABLE IF NOT EXISTS `node_tags` (
  `id` BIGINT(64) NOT NULL,
  `version` BIGINT(20) NOT NULL,
  `k` VARCHAR(255) NOT NULL DEFAULT '',
  `v` VARCHAR(255) NOT NULL DEFAULT '',
  PRIMARY KEY  (`id`,`version`,`k`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `relations`
--

DROP TABLE IF EXISTS `relations`;
CREATE TABLE IF NOT EXISTS `relations` (
  `id` BIGINT(64) NOT NULL DEFAULT '0',
  `changeset_id` BIGINT(20) NOT NULL,
  `timestamp` DATETIME NOT NULL,
  `version` BIGINT(20) NOT NULL,
  `visible` TINYINT(1) NOT NULL DEFAULT '1',
  PRIMARY KEY  (`id`,`version`),
  KEY `relations_timestamp_idx` (`timestamp`),
  KEY `changeset_id` (`changeset_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `relation_members`
--

DROP TABLE IF EXISTS `relation_members`;
CREATE TABLE IF NOT EXISTS `relation_members` (
  `id` BIGINT(64) NOT NULL DEFAULT '0',
  `member_type` ENUM('node','way','relation') NOT NULL DEFAULT 'node',
  `member_id` BIGINT(11) NOT NULL,
  `member_role` VARCHAR(255) NOT NULL DEFAULT '',
  `version` BIGINT(20) NOT NULL DEFAULT '0',
  `sequence_id` INT(11) NOT NULL DEFAULT '0',
  PRIMARY KEY  (`id`,`version`,`member_type`,`member_id`,`member_role`,`sequence_id`),
  KEY `relation_members_member_idx` (`member_type`,`member_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `relation_tags`
--

DROP TABLE IF EXISTS `relation_tags`;
CREATE TABLE IF NOT EXISTS `relation_tags` (
  `id` BIGINT(64) NOT NULL DEFAULT '0',
  `k` VARCHAR(255) NOT NULL DEFAULT '',
  `v` VARCHAR(255) NOT NULL DEFAULT '',
  `version` BIGINT(20) NOT NULL,
  PRIMARY KEY  (`id`,`version`,`k`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `schema_migrations`
--
```

```
DROP TABLE IF EXISTS `schema_migrations`;
CREATE TABLE IF NOT EXISTS `schema_migrations` (
  `version` VARCHAR(255) NOT NULL,
  UNIQUE KEY `unique_schema_migrations` (`version`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `sessions`
--

DROP TABLE IF EXISTS `sessions`;
CREATE TABLE IF NOT EXISTS `sessions` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `session_id` VARCHAR(255) DEFAULT NULL,
  `data` text,
  `created_at` DATETIME DEFAULT NULL,
  `updated_at` DATETIME DEFAULT NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `sessions_session_id_idx` (`session_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;

-- --------------------------------------------------------

--
-- Структура табліцы `users`
--

DROP TABLE IF EXISTS `users`;
CREATE TABLE IF NOT EXISTS `users` (
  `email` VARCHAR(255) NOT NULL,
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `active` INT(11) NOT NULL DEFAULT '0',
  `pass_crypt` VARCHAR(255) NOT NULL,
  `creation_time` DATETIME NOT NULL,
  `display_name` VARCHAR(255) NOT NULL DEFAULT '',
  `data_public` TINYINT(1) NOT NULL DEFAULT '0',
  `description` text NOT NULL,
  `home_lat` DOUBLE DEFAULT NULL,
  `home_lon` DOUBLE DEFAULT NULL,
  `home_zoom` SMALLINT(6) DEFAULT '3',
  `nearby` INT(11) DEFAULT '50',
  `pass_salt` VARCHAR(255) DEFAULT NULL,
  `image` text,
  `administrator` TINYINT(1) NOT NULL DEFAULT '0',
  `email_valid` TINYINT(1) NOT NULL DEFAULT '0',
  `new_email` VARCHAR(255) DEFAULT NULL,
  `visible` TINYINT(1) NOT NULL DEFAULT '1',
  `creation_ip` VARCHAR(255) DEFAULT NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `users_email_idx` (`email`),
  UNIQUE KEY `users_display_name_idx` (`display_name`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=101442 ;

-- --------------------------------------------------------

--
-- Структура табліцы `user_preferences`
--

DROP TABLE IF EXISTS `user_preferences`;
CREATE TABLE IF NOT EXISTS `user_preferences` (
  `user_id` BIGINT(20) NOT NULL,
  `k` VARCHAR(255) NOT NULL,
  `v` VARCHAR(255) NOT NULL,
  PRIMARY KEY  (`user_id`,`k`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- --------------------------------------------------------

--
-- Структура табліцы `user_tokens`
--

DROP TABLE IF EXISTS `user_tokens`;
CREATE TABLE IF NOT EXISTS `user_tokens` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `user_id` BIGINT(20) NOT NULL,
  `token` VARCHAR(255) NOT NULL,
  `expiry` DATETIME NOT NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `user_tokens_token_idx` (`token`),
  KEY `user_tokens_user_id_idx` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Структура табліцы `ways`
--

DROP TABLE IF EXISTS `ways`;
CREATE TABLE IF NOT EXISTS `ways` (
  `id` BIGINT(64) NOT NULL DEFAULT '0',
  `changeset_id` BIGINT(20) NOT NULL,
  `timestamp` DATETIME NOT NULL,
  `version` BIGINT(20) NOT NULL,
  `visible` TINYINT(1) NOT NULL DEFAULT '1',
  PRIMARY KEY  (`id`,`version`),
  KEY `ways_timestamp_idx` (`timestamp`),
  KEY `changeset_id` (`changeset_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `way_nodes`
--

DROP TABLE IF EXISTS `way_nodes`;
CREATE TABLE IF NOT EXISTS `way_nodes` (
  `id` BIGINT(64) NOT NULL,
  `node_id` BIGINT(64) NOT NULL,
  `version` BIGINT(20) NOT NULL,
  `sequence_id` BIGINT(11) NOT NULL,
  PRIMARY KEY  (`id`,`version`,`sequence_id`),
  KEY `way_nodes_node_idx` (`node_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура табліцы `way_tags`
--

DROP TABLE IF EXISTS `way_tags`;
CREATE TABLE IF NOT EXISTS `way_tags` (
  `id` BIGINT(64) NOT NULL DEFAULT '0',
  `k` VARCHAR(255) NOT NULL,
  `v` VARCHAR(255) NOT NULL,
  `version` BIGINT(20) NOT NULL,
  PRIMARY KEY  (`id`,`version`,`k`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Абмежаваньні для экспартаваных табліц
--
```

```
--
-- Абмежаваньні для табліцы `current_nodes`
--
ALTER TABLE `current_nodes`
  ADD CONSTRAINT `current_nodes_ibfk_1` FOREIGN KEY (`changeset_id`) REFERENCES `changesets` (`id`);

--
-- Абмежаваньні для табліцы `current_node_tags`
--
ALTER TABLE `current_node_tags`
  ADD CONSTRAINT `current_node_tags_ibfk_1` FOREIGN KEY (`id`) REFERENCES `current_nodes` (`id`);

--
-- Абмежаваньні для табліцы `current_relations`
--
ALTER TABLE `current_relations`
  ADD CONSTRAINT `current_relations_ibfk_1` FOREIGN KEY (`changeset_id`) REFERENCES `changesets` (`id`);

--
-- Абмежаваньні для табліцы `current_relation_members`
--
ALTER TABLE `current_relation_members`
  ADD CONSTRAINT `current_relation_members_ibfk_1` FOREIGN KEY (`id`) REFERENCES `current_relations`
(`id`);

--
-- Абмежаваньні для табліцы `current_relation_tags`
--
ALTER TABLE `current_relation_tags`
  ADD CONSTRAINT `current_relation_tags_ibfk_1` FOREIGN KEY (`id`) REFERENCES `current_relations` (`id`);

--
-- Абмежаваньні для табліцы `current_ways`
--
ALTER TABLE `current_ways`
  ADD CONSTRAINT `current_ways_ibfk_1` FOREIGN KEY (`changeset_id`) REFERENCES `changesets` (`id`);

--
-- Абмежаваньні для табліцы `current_way_nodes`
--
ALTER TABLE `current_way_nodes`
  ADD CONSTRAINT `current_way_nodes_ibfk_2` FOREIGN KEY (`node_id`) REFERENCES `current_nodes` (`id`),
  ADD CONSTRAINT `current_way_nodes_ibfk_1` FOREIGN KEY (`id`) REFERENCES `current_ways` (`id`);

--
-- Абмежаваньні для табліцы `current_way_tags`
--
ALTER TABLE `current_way_tags`
  ADD CONSTRAINT `current_way_tags_ibfk_1` FOREIGN KEY (`id`) REFERENCES `current_ways` (`id`);

--
-- Абмежаваньні для табліцы `nodes`
--
ALTER TABLE `nodes`
  ADD CONSTRAINT `nodes_ibfk_1` FOREIGN KEY (`changeset_id`) REFERENCES `changesets` (`id`);

--
-- Абмежаваньні для табліцы `node_tags`
--
ALTER TABLE `node_tags`
  ADD CONSTRAINT `node_tags_ibfk_1` FOREIGN KEY (`id`, `version`) REFERENCES `nodes` (`id`, `version`);

--
-- Абмежаваньні для табліцы `relations`
--
ALTER TABLE `relations`
  ADD CONSTRAINT `relations_ibfk_1` FOREIGN KEY (`changeset_id`) REFERENCES `changesets` (`id`);
```

```
--
-- Абмежаваньні для табліцы `relation_members`
--
ALTER TABLE `relation_members`
  ADD CONSTRAINT `relation_members_ibfk_1` FOREIGN KEY (`id`, `version`) REFERENCES `relations` (`id`,
`version`);

--
-- Абмежаваньні для табліцы `relation_tags`
--
ALTER TABLE `relation_tags`
  ADD CONSTRAINT `relation_tags_ibfk_1` FOREIGN KEY (`id`, `version`) REFERENCES `relations` (`id`,
`version`);

--
-- Абмежаваньні для табліцы `ways`
--
ALTER TABLE `ways`
  ADD CONSTRAINT `ways_ibfk_1` FOREIGN KEY (`changeset_id`) REFERENCES `changesets` (`id`);

--
-- Абмежаваньні для табліцы `way_nodes`
--
ALTER TABLE `way_nodes`
  ADD CONSTRAINT `way_nodes_ibfk_1` FOREIGN KEY (`id`, `version`) REFERENCES `ways` (`id`, `version`);

--
-- Абмежаваньні для табліцы `way_tags`
--
ALTER TABLE `way_tags`
  ADD CONSTRAINT `way_tags_ibfk_1` FOREIGN KEY (`id`, `version`) REFERENCES `ways` (`id`, `version`);
```

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Development Tools

> **Note:**
>
> This section contains background information for MariaDB developers.

See Jira - Status and Transition Definitions.

Explains the various lifecycle statuses and workflow transitions used for issue tracking in MariaDB's Jira instance.Details how Jira is utilized for roadmap scheduling, sprint planning, and managing the overall progress of development tasks.

See Jira - Project Planning and Tracking.

Details how Jira is utilized for roadmap scheduling, sprint planning, and managing the overall progress of development tasks.

See Profiling with Linux perf tool.

A technical guide on using the Linux `perf` utility to analyze server performance and identify code bottlenecks.

See Worklog.

Documentation regarding the MariaDB Worklog system, used to define, track, and review the progress of new feature implementations.

See Buildbot.

Overview of the continuous integration system used to automate the building and testing of MariaDB across various platforms.

See Using Git With MariaDB.

Outlines the best practices and mandatory procedures for using Git to manage source code contributions and version control.

This page is licensed: CC BY-SA / Gnu FDL

Jira - Status and Transition Definitions

Here is the MariaDB Server (MDEV) Jira project workflow.

OPEN

- Status assigned to newly created Jira issues
- Indicates that the issue has been reported and awaits further action
- Transitions
  - Needs feedback → NEEDS FEEDBACK
  - Start progress → IN PROGRESS
  - Close Issue → CLOSED

CONFIRMED

- Status applied to validate the existence of a bug type of issue
- Implies that the bug can be consistently reproduced and is scheduled for development
- Transitions
  - Stop Progress → STALLED
  - Request testing → IN TESTING
  - Needs feedback → NEEDS FEEDBACK
  - Close Issue → CLOSED
  - Request Review → IN REVIEW

NEEDS FEEDBACK

- Status utilised when progress is contingent on receiving essential feedback
- Feedback requests should be clearly communicated to the designated individual
- Issue can be closed after staying in a month in NEED FEEDBACK with no activity
- Transitions
  - Provide feedback → OPEN
  - No Feedback → CLOSED

IN PROGRESS

- Status employed when the assignee is actively working on the issue
- Denotes that the issue is being actively addressed and moving forward
- Transitions
  - Stop Progress → STALLED
    - From the end of 2023, we are moving the issues “IN PROGRESS”, which didn’t get any time to get meaningful updates from the assignee for 6 weeks in this status, to STALLED.
  - Request testing → IN TESTING
  - Needs feedback → NEEDS FEEDBACK
  - Close Issue → CLOSED
  - Request Review → IN REVIEW

IN REVIEW

- Status used to request a peer review of an issue, typically involving code evaluation
- Seeking input from another assignee to ensure quality and adherence to standards
- Transitions
  - Review done → STALLED
  - Request testing → IN TESTING
  - Close Issue → CLOSED

IN TESTING

- Status employed to initiate testing of completed work, often involving a quality engineer
- Ensuring that the implemented solution meets functional and performance requirements
- Transitions
  - Stop Testing → STALLED
  - Close Issue → CLOSED

STALLED

- Status indicating that progress on the issue has been temporarily halted
- Distinguished from the Open status by the presence of prior work on the stalled issue
- Transitions
  - Unconfirm issue → OPEN
  - Request testing → IN TESTING
  - Needs Feedback → NEEDS FEEDBACK
  - Start Progress → IN PROGRESS
  - Close Issue → CLOSED
  - Request Review → IN REVIEW

CLOSED

- Status assigned when work on the issue has been finalized
- Signifies that the issue has been successfully resolved and no further action is expected
- In the event that the issue requires reopening, it will revert to the Stalled status
- The resolution is another field that will explain how it has been resolved.
- Transitions
  - Reopen Issue → STALLED

This page is licensed: CC BY-SA / Gnu FDL

Jira - Project Planning and Tracking

Jira is the tool that is used for bug reporting, project planning and tracking in MariaDB development. It replaced the previous tool called WorkLog. Jira is also where you can find the MariaDB roadmap.

The MariaDB Jira is located at jira.mariadb.org

Everyone is welcome to sign up for a free account and submit issues, post new tasks or ideas, or to comment on existing issues. The sign up can be found next to the log in boxes on the front page or directly through this link.

After filling out the sign up form you can log in.

Links to Jira Tasks

You can easily create links to Jira tasks in the documentation using their “MDEV” number. For example, the text ‘`mdev-191`’, when entered into the documentation becomes: MDEV-191.

See Also

- Jira MDEV workflow - statuses and transitions

This page is licensed: CC BY-SA / Gnu FDL

Profiling with Linux perf tool

Linux `perf` tool can be used to do non-intrusive profiling.

Frequency Based Sampling

This mechanism can be used to answer the question, what is MariaDB doing on my CPU.

Recording a sample

Perf records at a high frequency, so only a short recording is sufficient to answer the question. If this is too short adjust the frequency at which perf does its recordings.

```
sudo perf record -p ${pidof mysqld} -g -o sample.perf -- sleep 5
```

The `-g` option here records the calling stack. Because seeing time in a mutex function isn't particularly interesting without knowing which mutex it is.

Viewing a sample

To view a recording, you need the debug symbols for your executable. See this page on getting the debug symbols available.

Changing the ownership of the recording means you can run perf report without sudo.

This shows where in the process MariaDB is spending most of its time at the top level. MariaDB uses threads per user connection so this will usually show a significant time in a `handle_connection` function. There are background threads that also run, so this can quickly show if its connection related time or a background thread.

```
sudo chown $USER: sample.perf
perf report -i sample.perf -g
```

To see which low level functions are consuming the most time, `--no-children` means that each function listed include only the time that is being spend it this function and excluding the other functions it calls.

```
perf report -i sample.perf -g --no-children
```

Expanding out the function shows the complete call stack again. Multiple functions may have called the function you are looking at so there may be a different frequency breakdown.

A more complete example of performance analysis using perf is on this Percona community blog article.

Dynamic Tracepoints

Adding dynamic tracepoints

One can add tracepoints at function entry/exit (and other locations too):

```
sudo perf probe -x /path/to/ha_rocksdb.so  --add rocksdb_prepare
sudo perf probe -x /path/to/ha_rocksdb.so  --add rocksdb_prepare%return
```

Viewing the tracepoints

```
sudo perf probe -l
```

Running the profiler

Something like:

```
perf record -e 'probe_ha_rocksdb:*' -a -- sleep 60
```

Note: `-a` means system-wide.

There's also `-p $PID` option

```
perf record -e 'probe_ha_rocksdb:*' -p $(pidof mysqld) -- sleep 60
```

Examining the trace

```
perf script
```

This page is licensed: CC BY-SA / Gnu FDL

Worklog

Worklog has been replaced, please refer to JIRA - project planning and tracking for further information.

Worklog is the tool used to track all development of MariaDB.

The MariaDB Worklog is open to everyone at index.pl (a free account on the developer wiki is required to suggest new tasks and add comments, votes, donations etc.). The account signup page for the wiki is here.

If you find something in the worklog that you really would like to have done, you can commit to donate some money to the developer when this is done. (Search after "Make offer" on the worklog item you would like to sponsor).

If there is something in worklog which you would like to develop, you can contact us at 'maria-developers (at) lists.launchpad.com' or 'community (at) askmonty.org'. If you deliver a working solutions that is accepted into the MariaDB source tree, you will get 60 % of the so far committed money. The rest of the money is kept by MariaDB Corporation Ab for help managing the project, code reviews, bug fixes, testing, maintenance, updates and merges to future MariaDB versions.

You can also add a link in the MariaDB worklog to tasks to the MySQL worklog. Just refer to the MySQL task as #WL

Source Code for Worklog

The source code for the Worklog application is hosted on Launchpad. The license is GPL.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot

As an example, you can see what's happening with 11.3 here.

See About Buildbot.

See Branch Protection Using Buildbot.

See Buildbot Database Schema.

See Buildbot Development.

See Buildbot ToDo.

See Package Testing with Buildbot and KVM.

See Buildbot runvm.

See Buildbot Setup.

This page is licensed: CC BY-SA / Gnu FDL

About Buildbot

Overview

The goal of MariaDB Foundation Buildbot is to ensure that the MariaDB Server is being thoroughly tested on all supported platforms and environments. We are currently running 100 different configurations for the following platforms:

- x64 and x86
- aarch64
- ppc64le
- s390x

and operating systems:

- Debian 9, 10, 11 and Sid
- Ubuntu 16.04, 18.04, 20.04 and 21.04
- Fedora 33 and 34
- CentOS 7 and 8
- RHEL 7 and 8
- SLES 12 and 15
- OpenSUSE 15 and 42
- Windows
- AIX-7.2

Moreover, we run other ecosystem tests for:

- PHP
- DBdeployer
- MySqlJS
- PyMySQL

Packages built by buildbot can be downloaded from here.

What is Buildbot?

The MariaDB Foundation uses Buildbot, a continuous integration and testing framework to test and create MariaDB Server packages. It is hosted onand ensures that each push to the MariaDB Server GitHub repository is properly tested.

Who uses Buildbot?

Buildbot should be used by each MariaDB developer to ensure that the new changes that are made are properly tested on all supported platforms and environments. In order to enforce this, Buildbot is used for branch protection. However, even though branch protection is enabled, **only a selected few** builders are part of it. So, it is the

developer’s responsibility to monitor all the builders and make sure that everything is fine before making the final push to the main MariaDB branch.

Buildbot keywords

- Changes/Repository - Any change that occurs in the source code (commit)
- Build Master - The main process that runs on a dedicated machine. It checks for changes in the source code and is in charge of scheduling builds
- Build - The actual configuration that is tested. It consists of a sequence of steps that define the actual test (e.g. get source code, compile, run tests, etc)
- Buildbot Worker - The process which waits for commands from the Build Master in order to run a build

How does the Buildbot work?

As it comes to the Buildbot Master, we use a multi-master configuration. This means that we have multiple running master processes. So, we have a dedicated master for the user interface and several other that deal with looking for changes and scheduling builds.

Each time a push is made to the MariaDB Server Repository, it is detected by the buildbot master which schedules all the builds. Each build defines a different test configuration. We mainly use Docker Latent Workers which means that for each build, the master starts a Docker container on a remote machine. The container is configured to run the buildbot-worker process on startup. This process can now receive instructions from the master. In this way, by using latent workers there isn’t a buildbot-worker process continuously running on the worker machine. Instead, for each build a separate container is started.

Below, you can find a detailed step by step overview of what happens after a push is made to the MariaDB Server repository:

- Step 1: Detect a new change in the MariaDB Server repository
  - Trigger source tarball creation
- Step 2: Tarball creation
  - Clone the repository and create a source tarball corresponding to the latest changes
  - Trigger bintar builds
- Step 3: Bintar builds
  - Fetch the source tarball previously created
  - Compile
  - Test (mysql-test-run)
  - Save bintar
  - Trigger package creation builds
  - Trigger ecosystem builds
- Step 4.1: Package creation
  - Fetch source
  - Create packages
  - Save packages
  - Trigger installation builds
- Step 4.2: Ecosystem tests
- Step 5: Installation builds
  - Fetch packages
  - Test if they install successfully
  - Trigger upgrade builds

- Step 6: Upgrade builds
  - Test if the previously MariaDB Server version can be successfully upgraded to the current one

The information below refers to the old Buildbot (), and not the new Buildbot (). The information is old, outdated, or otherwise currently incorrect.

Overview

The current state of the MariaDB trees with respect to build or test failures is always available from the Buildbot setup page.

- MariaDB-5.5 waterfall status page.
- MariaDB-10.0 waterfall status page.
- MariaDB-10.1 waterfall status page.

The BuildBot setup polls the Launchpad trees every 5 minutes for changes. Whenever a new push is found in one of our trees, the new code is compiled and run through the test suite.

If all platforms are green after this, everything is good. If not, it means there is a problem with the push, and someone needs to look into it ASAP. If it was your push, then the someone who needs to look at it is you!

BuildBot is a generic, GPL'ed program providing a continuous integration test framework. For more information on BuildBot, see the the BuildBot project homepage.

Volunteering to Run a Build Slave

Many of our build hosts are run by community members, and we are always looking for additional volunteers to help us cover additional platforms or build options in BuildBot.

If you are able to provide a spare machine for this purpose, your help is greatly appreciated! This is a good way to get involved without having to spend a lot of time on it. Get started by writing an email to 'developers (at) lists.mariadb.org' with an offer to run a BuildBot slave.

Setting up the Slave BuildBot

See buildbot-setup.

Pausing mysql-test-run.pl

Sometimes you need to work when your computer is busy running tests for buildbot. We've added a new feature to the mysql-test-run.pl script which allows you to stop it temporarily so you can use your computer and then restart the tests when you're ready.

To do this, define the environment variable "MTR_STOP_FILE". Whenever the file specified by this environment variable exists, the mysql-test-run.pl script will stop as soon as it is able to do so (i.e. it won't stop immediately). When the file is removed, the mysql-test-run.pl script will continue from where it left off.

If you plan on using this feature you should also set the "MTR_STOP_KEEP_ALIVE" environment variable with a value of 120. This will make the script print messages to buildbot every 2 minutes which will prevent a timeout.

Database with Test Results

Buildbot saves the results of test runs in a database, to be used for enhanced reporting on web pages without need to change the Buildbot code, and for data mining when investigating test failures.

The database schema is documented under Buildbot Database Schema.
The schema is likely to evolve as we gradually add more kinds of information.

For now, the data is not externally available. But the plan is to set up a slave database to replicate the data, and provide access (eg.
remote database accounts) to members of the community with interesting ideas about how to present or mine this data, or who are just curious to play with it. If anyone has an interest in this, or wants to volunteer a slave host for this purpose, please send a developers@lists.mariadb.org.
The more people show interest in this, the faster it is likely to happen!

Reports

We are developing new reports fed off the test results in the database. These reports will be located here. The first report is the Cross Reference report. This report allows all test failures to be searched.

Buildbot Maintenance

Here is some information on how our Buildbot installation is set up and maintained:

- The configuration file is included in the Tools for MariaDB repository.
- The building and testing of binary packages is documented on the package-testing-with-buildbot-and-kvm page.
- We developed a small tool, runvm, which is used to do some of the builds inside a virtual machine, mostly to test builds of binary packages.
- The BuildBot Development page describes how we developed some of the enhancements to BuildBot that we use and have contributed upstream.

See Also

- The Buildbot ToDo page

This page is licensed: CC BY-SA / Gnu FDL

Branch Protection Using Buildbot

MariaDB uses branch protection to ensure that pushes to the MariaDB Server git repository cannot be made without first passing a series of tests. This page aims to describe what a developer should do in order to ensure that their changes get pushed to the main repository.

The current protected branches are the main branches **10.X**/**11.X** and release branches **bb-10.X-release**/**bb-11.X-release***

Protected Branches Flow

1. Rebase your work on top of the latest main branch. `git rebase 10.11`
2. Push to bb-* branch
3. Wait for all the checks to run
4. If all checks are successful you can push to the main branch to be the same as bb-* branch
5. If not (failed checks), check the build status, make changes accordingly and repeat this flow

Protected Branches Builders

The following are the **only** required builders at this point:

1. amd64-debian-11-debug-ps-embedded
2. amd64-debian-12
3. amd64-debian-12-debug-embedded

4. amd64-debian-11-msan
5. amd64-fedora-40
6. amd64-last-N-failed
7. amd64-ubuntu-2004-clang
8. amd64-ubuntu-2004-debug
9. amd64-ubuntu-2204-debug-ps
10. amd64-windows (?)

The protected branches builders run a subset of tests in order to avoid sporadic failures. The list is defined by:

```
--
suite=main,spider,spider/bg,spider/bugfix,spider/feature,spider/regression/e1121,spider/regression/e11212
2 --skip-
test="^stack_crash$|^float$|^derived_split_innodb$|^mysql_client_test$|^kill$|^processlist_not_embedded$|
^sp-big$
```

From github protected_branches_mtr_additional_args.

All other checks are not required in order to make the push. However, please take a look at all the builders and look for failures.

Disabled Protected Branches

Viewing Build Status

The build status can be seen directly from GitHub near the commit message. Below, you can find an example (please note the **yellow dot** near the commit message):

If you click on the **yellow dot**, you can find the list of checks performed and their status, as shown below:

By clicking on **Details** you will be redirected to the buildbot page showing all the build details.

Alternatively, you can look for the builds on the buildbot Grid View. This allows filtering by branch, so to only see changes for one particular branch you can use grid?branch=10.5.

Status and Action Items

- Yellow dot - specifies that one or more of checks are not yet completed.
  - Action item: Wait for all the checks to finish
- Green tick - all checks were successful
  - Action item: No specific action is needed. The push can go through!
- Red X - one or more checks have failed
  - Action item: Look at the status of failing builds and fix potential issues

Note: Only buildbot and not external CI tests (Travis, AppVeyor, etc) are currently in the protected branches criteria. Please take note of other failures, and if they are acceptable and explainable, then merge.

Re-trigger Checks

In some cases it might be useful to re-trigger one or more checks. This can easily be done from the buildbot.mariadb.org interface. The steps are:

1. Login to buildbot.mariadb.org using the GitHub credentials: Hit the “Anonymous” button from the upper right corner and then “Login with GitHub” as shown below:

-

1. Open the build details of the build you want to re-trigger

2. Hit the “Rebuild” button from the upper right corner

Note: If you want to re-trigger all checks, the easiest approach is to make a new push. Alternatively, you can follow the above steps for all the checks

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Database Schema

This page describes the database schema used by Buildbot to save results from test runs.

The idea is to be able to use this data from outside of Buildbot for things like additional web pages presenting test results, or search/data mining facilities for searching for test failures.

Accessing the database

The plan is to make remote database connections available to community members. For this, we need to set up a slave host replicating the master Buildbot database (which would in any case be good to isolate the running Buildbot from possibly high load from queries).

However, for now the database access is only available locally on the machine (hasky) running the buildbot master.

Schema

The most current information about the schema used is available in the file `buildbot/process/mtrlogobserver.py` in the Buildbot sources. As the code evolves and more kinds of information is made available in the database, the schema might be extended, but the schema description in the source code should always be up-to-date.

The `test_run` table

This table has one row for every test run that Buildbot does. Thus, each row corresponds to one cell in the [waterfall Waterfall display]. The format of the table is as follows:

```
CREATE TABLE test_run(
    id INT PRIMARY KEY AUTO_INCREMENT,
    branch VARCHAR(100),
    revision VARCHAR(32) NOT NULL,
    platform VARCHAR(100) NOT NULL,
    dt TIMESTAMP NOT NULL,
    bbnum INT NOT NULL,
    typ VARCHAR(32) NOT NULL,
    info VARCHAR(255),
    KEY (branch, revision),
    KEY (dt),
    KEY (platform, bbnum)
) ENGINE=innodb
```

- id: Primary key, just an auto_increment id.
- branch: This is the name of the bzr branch of the test run.
- revision: The Bzr revision number tested.
- platform: The name of the builder that ran the test.
- dt: Date when the buildbot run was started.
- bbnum: The Buildbot ‘”build number”’ which together with `platform` uniquely identifies a build within Buildbot.
- typ: Concise abbreviation describing the kind of test. For example `pr` for –ps-protocol with row based replication, or `nm` for normal run with mixed-mode replication.
- info: Short textual description of the kind of test run.

The `test_failure` table

This table has one row for every test failure encountered:

```
CREATE TABLE test_failure(
    test_run_id INT NOT NULL,
    test_name VARCHAR(100) NOT NULL,
    test_variant VARCHAR(16) NOT NULL,
    info_text VARCHAR(255),
    failure_text TEXT,
    PRIMARY KEY (test_run_id, test_name, test_variant)
) ENGINE=innodb
```

- test_run_id: This identifies the test run in which the test failure occurred (eg. it is a foreign key to `id` in table `test_run`).
- test_name: The name of the test that failed, eg. `main.information_schema`.
- test_variant: Some tests are run multiple times in different variants. Ie. many replication tests are run under both statement-based, mixed-mode, and row-based replication. The variant will be ‘stmt’, ‘mix’, or ‘row’ accordingly. For tests that do not have multiple variants, the value will be the empty string (ie. not a NULL value).
- info_text: This is a short description that mysql-test-run.pl sometimes gives for some kinds of test failures (for example “timeout”).
- failure_text: This is the entire output from mysql-test-run.pl concerning this test failure. It usually contains the diff against the result file, a stacktrace for a crash, etc. This is useful to run `LIKE` queries against when searching for test failures similar to one being investigated.

The `test_warnings` table

This table holds information about test problems that were detected after a test case ran, during server restart (typically by finding an error or warning message in the server error log files). A typical example of this is a memory leak or a crash during server shutdown.

Such a failure can not be attributed to a specific test case, as it could be caused by any of the tests run against the server since last restart, or could even be a general problem not caused by any test case. Instead, for each occurrence, this table provides a list of names of the tests that were run by the server prior to detecting the error or warning.

```
CREATE TABLE test_warnings(
    test_run_id INT NOT NULL,
    list_id INT NOT NULL,
    list_idx INT NOT NULL,
    test_name VARCHAR(100) NOT NULL,
    PRIMARY KEY (test_run_id, list_id, list_idx)
) ENGINE=innodb
```

- test_run_id: Identifies the corresponding row in table `test_run`.
- list_id: This is a counter for occurrences of warnings within each test run (ie. it starts over from 0 again for each different value of `test_run_id`).
- list_idx: This is a counter for each test name (ie. it starts over from 0 again for each different value of `test_run_id` ‘’and’’ `list_id`).
- test_name: The name of the test run by the server prior to seeing the warning.

Sample queries

Show all platforms that failed for a particular revision of a particular branch:

```
SELECT platform
  FROM test_run r
WHERE branch = 'mysql-6.0-testing2'
  AND revision = '2819'
  AND (EXISTS (SELECT * FROM test_failure f WHERE f.test_run_id = r.id)
    OR EXISTS (SELECT * FROM test_warnings w WHERE w.test_run_id = r.id));
```

Find failures similar to a given failure being investigated:

```
SELECT branch, revision, platform, test_name, test_variant, failure_text
  FROM  test_failure f
  INNER JOIN test_run r ON (f.test_run_id = r.id)
  WHERE failure_text LIKE "%--protocol=TCP' failed%";
```

Check which branches a specific kind of failure has occurred in:

```
SELECT branch, COUNT(*)
  FROM test_failure f
  INNER JOIN test_run r ON (f.test_run_id = r.id)
  WHERE failure_text LIKE "%--protocol=TCP' failed%"
  GROUP BY branch;
```

Find all test runs where a given test was run against a server that later had warnings in the error log, and also count the number of occurrences of this event in each run:

```
SELECT branch, revision, platform, COUNT(*)
  FROM test_warnings w
  INNER JOIN test_run r ON (w.test_run_id = r.id)
  WHERE test_name = 'rpl.rpl_plugin_load'
  GROUP BY r.id;
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Development

See https://mariadb.org/the-onion-framework-a-new-approach-to-buildbot-configuration/

This page is licensed: CC BY-SA / Gnu FDL

Buildbot ToDo

High-priority, High-ROI items

- Ability on the web pages to mark certain builds of high priority, so they will be processed before others as build slaves become available. It must be possible to mark pending builds so not only individually, but also by branch, as well as by (branch,revision). The feature should probably interact in a meaningful way with the existing “nextBuild” ability to prioritise builds (see “Prioritizing Builds” in the Buildbot documentation). Eg.
  it could set appropriate properties somewhere that would be available to the nextBuild function.
- We need a way to kill all running builds, and remove all pending builds, for a given combination of branch and revision. It is possible to manually cancel a pending or running build, but this becomes extremely inconvenient when there are many builders.
  - Comments from buildbot mailing list: “There is already a button (hidden on the change page) for stopping all builds associated to a change.
    Unfortunately, the only way to get to this page seems to be through the waterfall.” “Note that currently, the button on the change page only cancels still-pending builds, it does not stop builds in progress.”
- Fix problem when pushing multiple bzr revisions at one time:

    - Buildbot should wait a few moments before starting a build, and then start building the latest (currently it seems to build the first revision immediately, and even often seems to not build the latest revision at all, getting confused about the order)
    - When pushing a change that changes revision numbers, the grid display can get really confused, would be nice to fix this somehow.
- Any failure other than compile/test failure should mean a re-run. Failures like buildslave reboot or loss of connection or timeout of bazaar operation should be interpreted as if the build has never happened, and the system should re-try building.
- In the bzr source step, retry the checkout a couple of times if it fails (to be more robust against temporary Launchpad problems, which we see a number of). Could maybe be done as a general Buildbot optional feature for all revision control systems supported, or maybe specific for the bzr depending on what can work.
- Get PBXT test suite fixed in `--valgrind`; once that is done include it in the list of main suites so that it is tested on all platforms (and remove the special extra pbxt test steps in some Buildbot builds, as they will no longer be needed).
- Load statistics. We need it to be able to see if adding another tree will starve the buildbot:
    - Percentage of busy time for available slaves
    - Median/Max response time (How much time do I have to wait after I've pushed for test run to complete)
- Fix that mysql-test-run uses `--skip-ssl` by default.
- Fix that `mysql-test-run.pl --mem` doesn't properly clean up `/dev/shm/var*` (so that we can use `--mem` and `--parallel` option to speed up the more powerful machines).
- For triggered builds (ie. one builder makes a source tarball which another builder consumes), make it so that the pushed changesets are also transferred, so the blamelist on the build page works (currently the blame list is empty).
- Upgrade to Buildbot 0.8 on the master. Then use the facility to use a MariaDB server as backend for both speedup (hopefully) and more flexible querying of buildbot history.

Lower priority items

- It should be easier to setup for a new user. Ideally, there should be one tarball that one can download, unpack and run. Within the tarball there should be a script and/or config files which allow to add the buildslave into automatic startup. The following should not be there:
    - Dependence of buildslave on a load of python libraries. Ship them.
    - Same for bzr
    - Same for mysqld itself. Dl/make/make-install everything we need in the buildbot directory.

If the above is not done, we'll never get much participation from old or unusual machines. The owners of such machines do not have the latest libraries, installing them requires some manual effort, which simply sets the bar too high.

- Check archivist observation/bug that build slave `uname` info disappears when slave is disconnected.
- Self refresh at a time interval (ajax maybe to reduce transfer) so users can know when to fire up a slave.
- In the waterfall page, second line from top is "current activity". When displaying a particular branch ("?branch=5.1"), the "current activity" for hosts that are currently building another branch could also display the name of the branch (as a link to waterfall page for that branch) to avoid confusion that it is active on the current branch.
- Run with `--valgrind-mysqld` instead of `--valgrind`. Little point in Valgrinding mysqltest when we in any case ignore any errors in that program.
- Show pending builds for a slave on the 'builders' page, eg. hardy-x86-rtai

- Show dates in local time of the client. One way is with
  Javascript: a client-side script (would be nice if some dates were still shown when javascript is not available).
- Check if InnoDB is fixed sufficiently to have no Valgrind leaks with `innodb_use_sys_malloc`; if so remove hack in `mysql-test-run.pl` to disable this in valgrind case.
- Fix problem that illegal regexp causes build exception (example:the `sol-sparc-32` compile log).
  The problem is that Buildbot grabs regexps from the suppression file in the
  MariaDB tree and tries to compile them; if this throws exception, then the exception should be caught (and possibly warned about), not crash the whole build.

Improve binary package build and test

- Add `apt-get source mariadb-server` checking in buildbot (for package testing).
- Add testing of bintar package with real start of server mysqld_safe and also test on different distro/version than what was built (eg hardy<->jaunty).
- In package build, sign .debs with dummy key, to more accurately test the real build process.
- Switch to build the bintar packages on centos 5, to work with older glibc versions? Or maybe even Debian 4 which is older I think? But do the install test in Buildbot also on other/newer distros (should probably test on several in each build).
- Add a test step for .deb that tests upgrading from an earlier version of
  MariaDB (currently we only check upgrade from MySQL).
- Add a 'follow log' feature to runvm (which does `ssh guest tail -f log > log`). Use this in the Buildbot packaging step that runs mysql-test-run.pl, so that we can get mysqld.X.err.Y server logs available like for the non-kvm builds.
- For runvm: add the options "-o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no" to the ssh command used to login to the guest vm, so we do not get a login failure due to different host keys in different guests.

Staging trees

Idea is to set it up so that each developer/group has a stating tree. Any push to this tree will first get a full round of testing in Buildbot. If all results are green after this, it will automatically be pushed to the main tree. If another push gets in first, it will automatically merge the new stuff and re-try a full Buildbot test. If there is a problem (test failure or merge conflict), it will send mail with details.

- Get something working, simple initially.
- Staging tree per-captain
- `mysqltest --require` not-staging to speed up Valgrind

This page is licensed: CC BY-SA / Gnu FDL

Package Testing with Buildbot and KVM

Buildbot Testing of Binary MariaDB Packages

This part of the Buildbot setup uses KVM virtual machines to build packages on a wide range of Linux distributions and test the different packages in various ways.

The builds run on a single host, which has an Intel Core i7 860 @ 2.80GHz CPU, 8 GByte of RAM, and a 500GByte disk. The build host is installed with Ubuntu 9.04 Server 64-bit.
The current box is capable of running up to 3 builds in parallel. The entire set of 19 builds take around 3.5 hours to complete.

The build host has no virtual machines that are kept running when idle. Instead each build boots up and shuts down virtual machines on the fly as needed, using the runvm tool.

The details of the setup (installation) of the virtual machines are here.

Builds

(Please check the Buildbot configuration for details, in case this information becomes out of date).

At the time of writing, the builds done are these:

- Source tarball build (on a 32-bit Ubuntu 9.04 VM)
- Binary tarballs, i386 and amd64, on Ubuntu 8.04 VMs.
- Ubuntu, i386 and amd64: 8.04, 8.10, 9.04, 9.10, 10.04 (alpha).
- Debian, i386 and amd64, Debian 4 and Debian 5.
- Centos 5, i386 and amd64.

The builds use the packaging scripts from OurDelta.

In all builds (except for source tarball), the binary package is built, and subsequently attempted installed. The build is done on a virtual machine with compilers and other necessary build tools installed.
The virtual machines are cloned from a reference image before each build (using the –base-image option to runvm). This way, the original images are not modified in any way, so there is no risk of “pollution” from previous builds, and no need to carefully clean up after a build or test.

The source tarball build is done in a virtual machine that is not reset after every build. This is to preserve the bzr shared repository setup inside the virtual machine. This way it is only necessary in each build to download new changes from Launchpad, which saves significant amounts of time (and Launchpad bandwidth). Since this particular build only works inside the source checkout directory, the risk of cross-build pollution is minimal.

The source tarball build is the only one that is triggered by bzr commits. After it has successfully built a source tarball, the tarball is uploaded to the Buildbot master, and all of the other builders are triggered to start building. Each will download the source tarball from the master and build from that rather than from a bzr checkout (this is the correct way, as we want to ensure that users building from the source tarball themselves get the same results as the packages built by us).

Since we use the OurDelta scripts, which use a “bakery” for the build scripts similar to a source tarball, there is a bakery tarball similarly generated from bzr and uploaded to the master.

Because of this triggering, it is not possible to manually force a Buildbot run on “latest bzr version” on the non-tarball builders. Instead to force a build, it is necessary to set the build properties for source tarball and bakery manually to the files one wants to tests. But it is often easier just to pick an existing build (with correct tarball) and use the “resubmit build” feature instead. Another option is to force build on the tarball build only; it will then in turn trigger all of the other builds.

Basic Install Test

On all builds (except source tarball), after the build a test is performed where the resulting package is installed in a virtual machine and basic testing is done.

The install test is done in a separate virtual machine from the one in which it was built, with no build tools and a very minimal install (to check that we have no hidden dependencies). The install test is currently *very* basic, basically just create a table and insert a row (it would be a nice ToDo to extend this testing to a more complete “smoke test”).

Note that for .deb (Debian and Ubuntu), we set up a small fake local apt repository, so that we can properly test that ‘’apt-get install’’ is able to pull in any extra dependencies without anything special needed on the part of the user.

Upgrade Tests for .deb Packages

On .deb builders (Debian and Ubuntu), we do three additional upgrade tests.

Two of them install the newly built MariaDB packages on top of an already installed MySQL package (the default MySQL package on that particular distro version), as well as on top of an earlier version of the MariaDB packages. For this, separate virtual machine images are used, based on the one used in the install test, but in addition with MySQL/MariaDB pre-installed.

The third test is performed on a clean image without any pre-installed MySQL or MariaDB packages. The test installs the latest available release of the same major version from MariaDB public repo, and then runs dist-upgrade using the local repo as a source of the new packages. The main goal of this test is to check that regular system upgrades performed via dist-upgrade work as expected in regard to MariaDB server.

In all tests, some very basic test data is also created in the installation, so we can test that replacing the old server installation with our new MariaDB package does not nuke the user's existing data! The test is also useful to check that the upgrade runs correctly with respect to dependencies etc, something that is quite complex to get right with the .deb packaging.

Upgrade Tests for .rpm Packages

On .rpm builders (CentOS, Fedora, RHEL, SLES, openSUSE) we have numerous upgrade tests, although only a small portion of them is run for every branch.

All tests upgrade an already existing installation to the new MariaDB packages. The variables are:

- what kind of existing installation is being upgraded: MariaDB server, MariaDB Galera, MySQL, Percona server, or mysql/mariadb installations provided by distributions;
- which command is used for upgrade: it can either be performed via upgrade/update (depending on the package manager), or install, or dist-upgrade;
- which packages are installed and upgraded: it can be a minimal set, server + client (plus whatever dependencies they bring), or a full set of packages.

In some cases, the tests can also set additional command-line options or environment variables for the package manager. It is normally done when the default installation does not work, and the package manager suggests using a different set of parameters.

Each test uses a clean VM image, installs an initial set of packages, and then uses one of the commands to upgrade the installation. It further checks that the server is upgraded, restarted and is reachable.

mysql-test-run.pl Test

On the .deb builders (Debian and Ubuntu), we additionally do a full mysql-test-run.pl test run. This is done using the package *mariadb-test*.

Note that currently, due to Bug #491349, on newer Ubuntus the default apparmor profile prevents running mysql-test-run.pl. Until this is fixed, the tests need to uninstall apparmor before running the test suite.

See Also

- Buildbot Setup for Virtual Machines

This page is licensed: CC BY-SA / Gnu FDL

Buildbot runvm

One type of build we do in BuildBot is to build and test MariaDB binary packages for the platforms we release on. We build and test packages for Debian

(4 and 5), Ubuntu (8.04 to 10.04), Centos 5, and generic Linux; amd64 and i386 architectures. This testing is done with virtual machines run in KVM.

To better control the startup and shutdown of the virtual machines we use a small wrapper around KVM we developed called **runvm**. The purpose of this tool is to make it easy to boot up a virtual machine, run some build commands inside it, and shut it down cleanly.

This wrapper encapsulates the steps needed to boot up a virtual machine, run a series of commands inside it, and shut it down gracefully afterwards. Some special care is taken in the script to ensure that the virtual machine is always shut down after use (gracefully if possible), even in case of various failures or the loss of the parent process or controlling TTY. And if a conflicting virtual machine somehow manages to escape shutdown, runvm automatically attempts to terminate it before starting a new one. This extra robustness is important for fully automated testing as in our Buildbot setup, to ensure that the system can run unattended for longer periods of time.

Essentially, instead of a normal Buildbot session which would do something like this on the slave:

```
./configure && make
```

We instead use *runvm* to do the same inside a virtual machine running as a
KVM guest with the build slave as host:

```
runvm image.qcow2 "./configure" "make"
```

See the **runvm Usage Examples** or **runvm –help** sections below for more detailed examples, but this is the basic idea.

runvm Usage Examples

Usage Example One

Here is an example command you could use to run a build inside a virtual machine using runvm:

```
runvm --port=2222 ubuntu-hardy-i386.qcow2 \
  "= scp -P 2222 mariadb-5.1.41-rc.tar.gz localhost:" \
  "tar zxf mariadb-5.1.41-rc.tar.gz" \
  "cd mariadb-5.1.41-rc && ./configure" \
  "cd mariadb-5.1.41-rc && make"
```

In this example, `ubuntu-hardy-amd64.qcow2` is a KVM image already installed with compilers and set up for password-less ssh access (using public key authentication). Port 2222 on the host side is forwarded to the ssh service (port 22) on the guest side (so by specifying different `--port` options it is easy to run multiple `runvm` invocations in parallel;
in our Buildbot setup we run 3 builds in parallel this way).

Note the use of the `scp` command, prefixed with an equals sign
“`=`” Commands prefixed in this way are run on the host side rather than the guest side; this is a convenient way to copy data in or results out of the virtual machine while the `runvm` session is running.

Using `runvm` in this way we are able to easily and flexibly manage a large number of virtual machines for automated builds with very little overhead and complexity. In fact we have around 70 distinct virtual machines! The only resource they take is a little disk space (around 37 GByte). And the virtual machines images are also simple to set up, requiring only a minimal install; no need to set up networking bridges or IP addresses, or to install a Buildbot client. All the complex logic runs on the host system, which only needs to be installed once.

By keeping the virtual images simple, builds and tests run in a minimal environment, which is useful to detect any missing dependencies or other problems that do not show themselves on normal developer machines with a full

desktop install (we even do install testing on a separate virtual machine from the one used to build, with compilers etc. not installed on the one used to test installation).

Usage Example Two

A further refinement of **example one** (above) is to create a new temporary virtual machine image before each step as a copy of a reference image, run the build, and throw away the temporary image after the build. This avoids any possibility of a previous build influencing a following build in any way (and thus also simplifies the build setup, as we can install stuff freely without any need to do cleanup). It also avoids having to fix a broken image, like needing to manually run `fsck` after a crash or similar issue. We use this technique for most of our binary package builds in Buildbot.

To use this copy-and-discard technique with runvm, the –base-image option is useful:

```
runvm --port=2222 --base-image=ubuntu-hardy-i386.qcow2 tmp.qcow2 \
  "= scp -P 2222 mariadb-5.1.41-rc.tar.gz localhost:" \
  "tar zxf mariadb-5.1.41-rc.tar.gz" \
  "cd mariadb-5.1.41-rc && ./configure" \
  "cd mariadb-5.1.41-rc && make"
```

This will run the build in a temporary copy `tmp.qcow2` of the reference image `ubuntu-hardy-i386.qcow2`, without modifying the reference image in any way. This uses the copy-on-write feature of the qcow2 image format (see `qemu-img(1)`), so it even takes only very little time (fraction of a second) and minimal space (only changed blocks are written to the new image).

Additional Usage Examples

The above two examples show basically how the package testing in our Buildbot setup is done. There are some further details of course, like more options for the build commands and extra care to get logfiles out to debug problems; the full details are available in our Buildbot configuration file.
But the basic principle is just a number of `runvm` commands like the examples above.

Getting runvm

The `runvm` tool is available under GPL on Lauchpad in the project Tools for MariaDB. In the bzr repository it is found as buildbot/runvm. 0If someone finds it useful or has suggestions for improvements, please drop us a line on the maria-developers mailing list.

runvm –help

Since it might be useful, here is the output from **runvm –help** (check the latest version of the tool for up-to-date output):

```
Usage: ./runvm <options> image.qcow2 [command ...]

Boot the given KVM virtual machine image and wait for it to come up.
Run the list of commands one at a time, aborting on receiving an error.
When all commands are run (or one of them failed), shutdown the virtual
machine and exit.

Commands are by default run inside the virtual machine using ssh(1). By
prefixing a command with an equals sign '=', it will instead be run on the
host system (for example to copy files into or out of the virtual machine
using scp(1)).

Some care is taken to ensure that the virtual machine is shutdown
gracefully and not left running even in case the controlling tty is
closed or the parent process killed. If a previous virtual machine is
already running on a conflicting port, an attempt is made to shut it
down first. For this purpose, a PID file is created in $HOME/.runvm/

Available options:

  -p, --port=N        Forward this port on the host side to the ssh port (port
                      22) on the guest side. Must be different for each runvm
                      instance running in parallel to avoid conflicts. The
                      default is 2222.
                      To copy files in/out of the guest use a command prefixed
                      with '=' calling scp(1) with the -P option using the port
                      specified here, like this:
                          runvm img.qcow2 "=scp -P 2222 file.txt localhost:"
  -u, --user=USER     Name of the account to ssh into in the guest. Defaults to
                      the name of the user invoking runvm on the host.
  -m, --memory=N      Amount of memory (in megabytes) to allocate to the guest.
                      Defaults to 2047.
  --smp=N             Number of CPU cores to allocate to the guest.
                      Defaults to 2.
  -c, --cpu=NAME      Type of CPU to emulate for KVM, see qemu(1) for details.
                      For example:
                          --cpu=qemu64      For 64-bit amd64 emulation
                          --cpu=qemu32      For 32-bit x86 emulation
                          --cpu=qemu32,-nx  32-bit and disable "no-execute"
                      The default is qemu32,-nx
  --netdev=NAME       Network device to emulate. The 'virtio' device has good
                      performance but may not have driver support in all
                      operating systems, if so another can be specified.
                      The default is virtio.
  --kvm=OPT           Pass additional option OPT to kvm. Specify multiple times
                      to pass more than one option. For example
                          runvm --kvm=-cdrom --kvm=mycd.iso img.qcow2 ...
  --initial-sleep=SECS
                      Wait this many seconds before starting to poll the guest
                      ssh port for it to be up. Default 15.
  --startup-timeout=SECS
                      Wait at most this many seconds for the guest OS to respond
                      to ssh. If this time is exceeded assume it has failed to
                      boot correctly. Default 300.
  --shutdown-timeout=SECS
                      Wait at most this many seconds for the guest OS to
                      shutdown gracefully after sending a shutdown command. If
                      this time is exceeded, assume the guest has failed to
                      shutdown gracefully and kill it forcibly. Default 120.
  --kvm-retries=N     If the guest fails to come up, retry the boot this many
                      times before giving up. This helps if the virtual machine
                      sometimes crashes during boot. Default 3.
  -l, --logfile=FILE  File to redirect the output from kvm into. This includes
                      any (error) messages from kvm, and also includes anything
                      the guest writes to the kvm emulated serial port (it can
                      be useful to set the guest to send boot loader and kernel
                      messages to the serial console and log them with this
                      option). Default is to not log this output anywhere.
```

```
-b, --base-image=IMG
                    Instead of booting an existing image, create a new
                    copy-on-write image based on IMG. This uses the -b option
                    of qemu-img(1). IMG is not modified in any way. This way,
                    the booted image can be discarded after use, so that each
                    use of IMG is using the same reference image with no risk
                    of "pollution" between different invocations.
                    Note that this DELETES any existing image of the same
                    name as the one specified on the command line to boot! It
                    will be replaced with the image created as a copy of IMG,
                    with any modifications done during the runvm session.
```

This page is based on a blog post by Kristian Nielsen, the primary developer of `runvm`.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup

See Buildbot Setup Notes.

See Buildbot Setup for macOS.

See Buildbot Setup for Windows.

See Buildbot Setup for BSD.

See Buildbot Setup for Solaris Sparc.

See Buildbot Setup for Solaris x86.

See Buildbot Setup for Ubuntu-Debian.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup Notes

General setup instructions are available in the BuildBot manual, specifically in the section on Setting up a build slave.

In addition to installing BuildBot on the slave host, it is also necessary to install all the tools needed to branch MariaDB from Launchpad and compile it. It is a good idea to first manually branch the code from Launchpad and successfully build it, as otherwise a lot of time may be needed fixing things one at a time as new builds are started and fail in one way or the other.

Unfortunately, bzr is memory hungry, so at least 1 Gigabyte of memory is recommended (you may be able to squeeze through with less, but bzr is a real memory hog). A few Gigabytes of disk space are also needed to hold the build directory.

Here are some detailed instructions for various systems:

- Buildbot Setup for Ubuntu-Debian
- Buildbot Setup for MacOSX
- Buildbot Setup for Solaris
- Buildbot Setup for Windows

See the Buildbot TODO for plans and ideas on improving Buildbot.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for macOS

Setting up a Buildbot Slave on macOS

Install buildbot-slave using macports or buildbot from fink (if you use old version of `buildbot` / `buildbot-slave`, you should use the `buildbot` command instead of `buildslave-2.6` in the following instructions).

Add user buildbot. Make sure that you do not have a buildbot user and group on your system:

```
# Check for group with id 101
id -g -nr 101

# Check for user with id 101
id -u -nr 101
```

If you do not have the group and user buildbot, then create the group and the user buildbot:

```
# Create group buildbot with group id 101
GROUP="buildbot"
dscl . create /groups/$GROUP
dscl . create /groups/$GROUP name $GROUP
dscl . create /groups/$GROUP passwd "*"
dscl . create /groups/$GROUP gid 101

# Create user buildbot with id 101
BUILDSLAVE_HOME=/var/lib/buildslave
USER=buildbot
mkdir -p $BUILDSLAVE_HOME
dscl . -create /Users/$USER
dscl . -create /Users/$USER RealName "Buildbot slave"
dscl . -create /Users/$USER NFSHomeDirectory $BUILDSLAVE_HOME
dscl . -create /Users/$USER UserShell /bin/bash
dscl . -create /Users/$USER UniqueID 101
dscl . -create /Users/$USER PrimaryGroupID 101
chown 101:101 $BUILDSLAVE_HOME
```

To hide the user use:

```
defaults write /Library/Preferences/com.apple.loginwindow HiddenUsersList -array-add buildbot
```

Prepare the environment:

```
sudo -i -u buildbot
buildslave-2.6 create-slave --usepty=0 maria-slave hasky.askmonty.org:9989 <slavename> <passwd>
bzr init-repo maria-slave/<slavedirectory>
$EDITOR maria-slave/info/admin
$EDITOR maria-slave/info/host
logout
```

Buildbot can be started/stopped manually with these commands (it's a good idea to start and stop it to see if it is set up correctly). If you installed buildbot from fink, please make sure that the buildbot user is using the environment settings. Your .profile should contain following line:

```
sudo - buildbot
more .profile
test -r /sw/bin/init.sh && . /sw/bin/init.sh
```

```
sudo -i -u buildbot
buildslave-2.6 start maria-slave
buildslave-2.6 stop maria-slave
logout
```

In order to make buildbot start on system boot, you'll need to create a `/Library/LaunchDaemons/net.sourceforge.buildbot.plist` file with the following contents (modified example from buildbot wiki):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
    <dict>
    <key>StandardOutPath</key>
    <string>twistd.log</string>
    <key>StandardErrorPath</key>
    <string>twistd-err.log</string>
    <key>EnvironmentVariables</key>
    <dict>
        <key>PATH</key>
        <string>/opt/local/bin:/sbin:/usr/sbin:/bin:/usr/bin</string>
        <key>PYTHONPATH</key>
        <string>/opt/local/lib/python2.5/site-packages</string>
    </dict>
    <key>GroupName</key>
    <string>daemon</string>
    <key>KeepAlive</key>
    <dict>
        <key>SuccessfulExit</key>
        <false/>
    </dict>
    <key>Label</key>
    <string>net.sourceforge.buildbot.slave.test</string>
    <key>ProgramArguments</key>
    <array>
        <string>/opt/local/bin/buildslave-2.6</string>
        <string>start</string>
        <string>--nodaemon</string>
        <string>maria-slave</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>UserName</key>
    <string>buildbot</string>
    <key>WorkingDirectory</key>
    <string>/var/lib/buildslave/</string>
    </dict>
</plist>
```

If you installed buildbot from fink, you can edit and copy the `plist` file:

```
$EDITOR /sw/share/doc/buildbot-py26/contrib/os-x/net.sourceforge.buildbot.slave.plist
```

Your `plist` file should similar to this one after editing:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-
1.0.dtd
">
<plist version="1.0">
<dict>
         <key>Label</key>
         <string>net.sourceforge.buildbot.slave</string>

         <!-- Change this to the user you want to run buildbot as -->
         <key>UserName</key>
         <string>buildbot</string>

         <!-- Change this to your buildbot working directory -->
         <key>WorkingDirectory</key>
         <string>/Volumes/MiniHD2/ServiceData/buildslave/maria-slave</string>

         <key>EnvironmentVariables</key>
         <dict>
                 <key>PATH</key>
                 <string>/sw/bin:/sw/sbin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin</string>
         </dict>

         <key>ProgramArguments</key>
         <array>
                 <string>/sw/bin/twistd</string>
                 <string>--nodaemon</string>
                 <string>--python=buildbot.tac</string>
                 <string>--logfile=buildbot.log</string>
                 <string>--prefix=slave</string>
         </array>

         <key>KeepAlive</key>
         <dict>
                 <key>SuccessfulExit</key>
                 <false/>
         </dict>

         <key>RunAtLoad</key>
         <true/>

</dict>
</plist>
```

```
sudo cp /sw/share/doc/buildbot-py26/contrib/os-x/net.sourceforge.buildbot.slave.plist
/Library/LaunchDaemons/
```

Warning:

You have to start your build slave via `launchd`, otherwise you will run into several problems. For further details, please refer to Using Launchd.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Windows

This is the recipe for setting up a MariaDB Buildbot slave on Windows:

1. Prepare the development environment
2. Install Python, 32 bit. Twisted does not work on 64 bit and builtbot hasn't been tested properly with Python version 3.
   Note: As of June 2016, there is no fresh Twistd for 32-bit Python, so a 64-bit version has to be installed. Installed 2.7.11 64-bit, it seems to work.

3. Install pywin32. Make sure the version matches your Python version perfectly, and get the .exe file, not the zip file.
   Note: As of June 2016, used `pywin32-220.win-amd64-py2.7.exe`, it seems to work.
4. Install Twisted
   Note: As of June 2016, used `Twisted 16.2.0 for Python 2.7 64 bits`
5. Install buildbot: Get the zip file and unpack it. In an administrator shell, cd to the buildbot dir and run "python setup.py install". After that, the unpacked buildbot directory is no longer needed.
   Note: As of June 2016, used `buildbot 0.8.12`.

When this has been done, you run these commands:

```
cd <somewhere>
mkdir buildbot
cd buildbot
bzr init-repo .
C:\buildbot\buildbot-slave-0.8.3\build\scripts-2.7\buildslave.bat create-slave --usepty=0 <slavedir>
hasky.askmonty.org:9989 <slavename> <passwd>
```

Adjust paths in the last command according to the location of your unpacked buildbot.

You get and from one of the MariaDB Corporation or MariaDB Foundation people handling the buildbot system. can be whatever you want - is a good choice.

Please edit the files \info\admin and \info\host with your information. The host information should include your Windows version, 32 or 64 bit, and your Visual Studio version.

You can test the buildbot slave with this command:

```
C:\buildbot\buildbot-slave-0.8.3\build\scripts-2.7\buildslave.bat start <somewhere>\buildbot\<slavedir>
```

When buildbot starts, you should configure it as a service instead of manually. See the instructions here. It's under the section "Windows Buildbot service setup". This document also has the generic Windows Buildbot installation documentation.

Why Buildbot Should Run as a Service

When buildbot is running in user session, and application that is started by buildbot crashes, you'll get a crash popup. There are popups that are generated by post-mortem debugging and there are popups that are generated by Windows Error reporting. It is hard (or impossible for those who did not try this before) to get rid of all popups. Thus, do run buildbot as service. There are no popups in services.

Common Windows Buildbot Failures

Test hangs after trying to call cdb to print a backtrace.

MTR attempts to call `cdb` via the Perl backticks operator, when mysqld.exe crashes. On the first run, cdb is downloading public Windows symbols from msdl.microsoft.com/download/symbols. The symbols are cached in C:\cdb_symbols, and following runs will be faster, however crashdump analysis on very first crash will typically take some time.

If you find this bothering, start the test with `--mysqld=--gdb` which will cause no crashdump files to be created and thus will prevent `cdb` from being called.

Buildbot exception `The process cannot access the file because it is being used by another process`

This is due to the fact that buildbot does not clean up any processes left over from a previous run, and those processes may hold locks on files that are needed for a new build to start. Current workaround is to use windows job

objects that allow to terminate entire process trees. We use special “process launcher” utility called “`dojob` . This will also require changing buildbot configuration for the builder.

Download dojob.cpp, and compile it with

```
cl dojob.cpp
```

Then, put dojob.exe into a directory in the PATH environment variable.
Then, change buildbot configuration to use “dojob” for every command, as a replacement for “cmd /c”, e.g

```
factory.addStep(Compile(
        name = "cmake",
        command=["dojob", WithProperties("cd c:\\buildbot\\%(buildername)s\\build && cmake .")]
));
```

Buildbot exception `ShellCommand.failed: command failed: SIGKILL failed to kill process`

(Seen very seldom?). This usually happens after retrying multiple failing tests multiple times. It appears that MTR’s `--retries=` option is not safe to use on Windows. The solution is to run the test with no `--retries` .

Buildbot exception `Connection to the other side was lost in a non-clean fashion.`

This is a sympthom of intermittent network issues, which cause Buildbot to abort the current build altogether. The following workarounds are possible:

- In your `buildbot.tac` file, specify a higher `keepalive` value, such as 60000.
  Note: For the current versions (as of June 2016), the opposite solution worked: keepalive = 60. KeepAliveTime in the Windows registry has been set to 60000 as suggested below, but there is no proof it made any difference.
- If you are running the Windows host inside VMWare, substitute the `e1000` network adapter with `vmxnet3`
- If your build host is behind a firewall, makes sure that the firewall does not time out any idle connections quickly. Configure at least a 24-hour timeout.
- If your build host is behind a firewall, consider disabling the built-in Windows Firewall in order to avoid one potential point of failure.
- Modify the Windows `KeepAliveTime` registry setting to a lower value, such as 60000 (equal to 60 seconds). For more information, see TechNet
- Make sure the buildbot master does not experience prolonged bouts of 100% CPU activity, as this may prevent keepalives from working. If the buildbot master `twisted.log` says that data is frequently being loaded from on-disk pickles, increase the `buildCacheSize` in the master configuration file to be more than the number of builds per builder that the log file reports are being loaded.

Alternative Windows Buildbot Setup

> **Note:**
>
> This is an experimental setup.

In case the default Windows buildbot setup is not sufficiently reliable due to “connection lost in a non-clean fashion” errors, the following setup can be used instead. It runs the buildbot daemon on a linux host while doing the builds on Windows, thus working around Twisted issues on Windows.

Note that the procedure below _significantly_ degrades the overall security of your Windows host. It is strongly recommended that a properly-firewalled, standalone virtual machine is used.

- Set the Windows server to auto-login with the user the builds will run as. This will prevent any issues relating to running things as a service and the privilege considerations associated with services;

- Install FreeSSHd on Windows. The OpenSSHd/Cygwin combo is not likely to work.
  - Do not run FreeSSHd it as a service, but rather as a startup console application for the user you are running the builds under. This allows FreeSSHd UI to properly operate;
  - Generate a private/public key pair using PuTTYGen and place the public part in a properly named file in the key directory specified in the FreeSSHd GUI
  - Using the FreeSSHd GUI, create a new user and set it to use key authentication;
  - Set the SFTP home path to the directory where builds will take place.
- Disable UAC (User Access Control) (or set to lowest level) by using the Users Control Panel. Add your buildbot user to the Administrators group.
- Export the private key from PuTTYGen and copy it to a Linux host.
- Install buildbot on the Linux host
- The buildbot configuration file will then look something like:

```
f_win2008r2_i386_release.addStep(ShellCommand(
        command=["ssh", "-i", "/home/buildbot/keys/id_rsa", "buildbot@win2008r2-build", "cmd", "/C",
"whoami"]
));
```

The following buildbot considerations apply under this setup:

- buildbot will not perform a proper cleanup of the build directory on the Windows host before each build (it will only clean up things on the linux side). So, a directory cleanup via rmdir must be placed as an explicit first step in the build sequence;
- Bzr() can not be used to check out BZR trees, as it expects that the bzr checkout command will run locally. Instead, an explicit call to bzr checkout via a SSH command must be used;

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for BSD

Here are the steps I did when installing and configuring a buildbot slave on a PC-BSD 9 box.

Add buildbot user:

```
sudo adduser
  buildbot
  /bin/sh
```

Python was already installed.

Bazaar was already installed.

NTP was already installed.

Install Zope3:

```
cd /usr/ports/www/zope3
sudo make install clean
  # accepted default options
cd /usr/ports/devel/py-zope.interface
sudo make install clean
```

Install Twisted:

```
cd /usr/ports/devel/py-twisted
sudo make install clean
  # accepted default options
```

Install `ccache`:

```
cd /usr/ports/devel/ccache
sudo make install clean
  # accepted default options
```

Run a test compile of MariaDB:

```
cd
cd src/maria/build
BUILD/compile-pentium64-max
  # test compile appeared to work
```

Install buildbot:

```
cd /usr/ports/devel/buildbot
sudo make install clean
  # accepted default options
```

Create the buildbot slave. On the build master, add new entry to `/etc/buildbot/maria-master-private.cfg`:

```
slave-name=bsd9
```

Remember the `${slave-name}` and `${password}` configured above, they're used in the next step.

Back on `bsd9`, do this:

```
sudo su - buildbot
buildslave create-slave --usepty=0 /home/buildbot/maria-slave \
hasky.askmonty.org:9989 ${slave-name} ${password}

echo '${contact-email-address}' > /home/buildbot/maria-slave/info/admin
echo 'A host running PC-BSD 9.' > /home/buildbot/maria-slave/info/host

bzr init-repo maria-slave/bsd9

exit
```

Start the build slave:

```
sudo su - buildbot
buildslave start maria-slave
```

Create the archive directory:

```
sudo su - buildbot
mkdir archive
exit
sudo ln -s /home/buildbot/archive /archive
```

Install Apache:

```
cd /usr/ports/www/apache22
sudo make install clean
  # accepted default options
```

Configure Apache:

```
sudo su -s
echo 'apache22_enable="YES"' >> /etc/rc.conf
echo 'alias /archive "/archive"\
<Directory "/archive">\
  Options All Multiviews\
  AllowOverride All\
  Order allow,deny\
  Allow from all\
</Directory>' >> /usr/local/etc/apache22/httpd.conf

sudo /usr/local/etc/rc.d/apache22 start
```

Install `md5sum`:

```
md5sum already installed at /compat/linux/usr/bin/md5sum
edited /home/buildbot/.profile and added that dir to the path
  # That didn't work, so did the following:
cd /usr/local/bin/
sudo ln -s /compat/linux/usr/bin/md5sum md5sum
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Solaris Sparc

Setting up a BuildBot slave on Solaris

NOTE #1:_It would probably make sense to create a separate zone for the buildbot on your Solaris 10 system. That is left up to you!_

NOTE #2:_You might need to install the Zope Interface package for some of the Python pieces to work properly. Do this after installing Python 2.5. You can find information on the Zope Interface on their_ *site.*

Solaris 10 (SPARC)

Unless you want to spend time optimizing the underlying Solaris 10 (SPARC) installation, it is strongly suggested that you choose the “Entire” Software
Group during the installation process. Once you have your Solaris 10 (SPARC) system on the network and able to access the Internet you can use the following suggestions to setup your buildbot:

1. By default Solaris 10 comes packages with Python 2.6.x. For compatibility reasons, you need to install Python 2.5.x. The author found it easiest to use Sunfreeware (ftp://ftp.sunfreeware.com/pub/freeware/sparc/5.10/):

```
$ cd /tmp
$ ftp ftp.sunfreeware.com
Name: anonymous
Password: (your email address)
ftp> bin
ftp> cd pub/freeware/sparc/5.10/
ftp> get python-2.5.4-sol10-sparc-local.gz
ftp> quit
$ gunzip python-2.5.4-sol10-sparc-local.gz
$ pkgadd -d python-2.5.4-sol10-sparc-local
```

If you are unfamiliar with howto install packages from Sunfreeware please read their howto.

1. Setup your environment:

- Create a user:

```
$ useradd -d /export/home/buildbot -m buildbot
```

It is imperative that you not use Bash as the buildbot user's shell.
It could have been a dependency issue but I literally spent several days trying to solve why the buildbot wasn't checking out code with bzr only to discover that using the default Solaris shell fixed the problem. If someone comes up with a solution please let us know on the mailing list.

- Adjust the global profile (you could be more granular here but since I'm setting up a dedicated system I wasn't):

```
$ vi /etc/profile
LD_LIBRARY_PATH=/opt/csw/lib:/usr/local/lib:/usr/sfw/lib:$LD_LIBRARY_PATH

# Add required libraries

PYTHONPATH=/usr/local/lib/python2.5/site-packages:$PYTHONPATH

# Makes Python 2.5 the default

PATH=/usr/local/bin:/usr/bin:/usr/sbin:/etc:/usr/sfw/bin:$PATH

# Puts "local" packages in your path

export LOGNAME PATH PYTHONPATH LD_LIBRARY_PATH
```

1. Install the latest Twisted:

```
$ wget tmrc.mit.edu/mirror/twisted/Twisted/9.0/Twisted-9.0.0.tar.bz2
$ bunzip2 Twisted-9.0.0.tar.bz2
$ tar -xf Twisted-9.0.0.tar
$ cd Twisted-9.0.0
$ python setup.py build install
```

1. Install required packages from Sunfreeware:

```
automake, autoconf, gcc, m4, md5, openssl, libsigsegv, Tcl, Tk, perl,
libtool, sed, libgcc, gcc, libintl, libiconv, zlib, binutils, groff, cmake
```

1. Install the Bazaar DVCS:

- You can find a package on Google Code.
- Install the latest Bazaar for Solaris 10. At the time of this writing it was 1.14.1: bzr-1.14.1-sol10-sparc-local.gz .

1. Volunteer your buildbot to the Maria team. Follow the directions listed in the Volunteering to run a build slave section of the About Buildbot page.
2. Create the buildbot as the buildbot user:

```
buildbot create-slave --usepty=0 /export/home/buildbot/maria-slave
hasky.askmonty.org:9989 ${buildbotname} <passwd>
```

Replace ${buildbotname} with the buildbot name you received from the MariaDB developers. Replace with the password they gave you.
You can adjust these and other parameters anytime within the maria-slave/buildbot.tac file at anytime in the future.

1. Create a shared Bazaar repository in the buildbot build directory:

```
HOME=/export/home/buildbot; bzr init-repo maria-slave/${buildbotname}
```

Replace ${buildbotname} like you did in Step #7.

1. Attach the buildbot to the master (this assumes you are not logged in as buildbot):

```
sudo su - buildbot -c "/usr/bin/buildbot start /export/home/buildbot/maria-slave"
```

OPTIONAL: You can create a proper service by following (and modifying as necessary) these directions

1. Please ask on the #maria IRC channel on irc.freenode.net if you have problems or questions setting up your buildbot. Or ask on the on the maria-developers mailing list.
2. Check the status of your buildbot through the BuildBot buildslaves page.

This howto was contributed by Adam Dutko. The original version is here.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Solaris x86

The following steps were used to create a Solaris 10 x86 BuildSlave.

I started with a default install of Solaris 10.

First I added a new user with:

```
groupadd sudo
useradd -d /export/home/${username} -m -s /bin/bash -g staff -G sudo ${username}
passwd ${username}
```

I then logged in as the new user and setup an ssh key.

Now to install software

Prior to actually installing the software, I adjusted the global profile so that the /usr/local/ dirs were included in the various PATHs:

```
vi /etc/profile

# Add the following lines:

LD_LIBRARY_PATH=/opt/csw/lib:/usr/local/lib:/usr/sfw/lib:$LD_LIBRARY_PATH # Add required libraries
PYTHONPATH=/usr/local/lib/python2.5/site-packages:$PYTHONPATH
PATH=/usr/local/bin:/usr/bin:/usr/sbin:/etc:/usr/sfw/bin:$PATH # Puts "local" packages in your path
export LOGNAME PATH PYTHONPATH LD_LIBRARY_PATH
```

The extra software, I downloaded from sunfreeware:

```
cd /tmp
ftp ftp.sunfreeware.com
anonymous
none

bin
cd pub/freeware/intel/10/
mget python-2.5.1-sol10-x86-local.gz sudo-1.7.4p4-sol10-x86-local.gz libintl-3.4.0-sol10-x86-local.gz
libgcc-3.4.6-sol10-x86-local.gz libiconv-1.13.1-sol10-x86-local.gz

mget automake-1.9-sol10-intel-local.gz autogen-5.9.8-sol10-x86-local.gz autoconf-2.68-sol10-x86-local.gz
gcc-4.5.1-sol10-x86-local.gz

mget m4-1.4.15-sol10-x86-local.gz libtool-2.4-sol10-x86-local.gz gmp-4.2.1-sol10-x86-local.gz

mget md5-6142000-sol10-intel-local.gz openssl-1.0.0c-sol10-x86-local.gz libsigsegv-2.9-sol10-x86-local.gz
tcl-8.5.9-sol10-x86-local.gz tk-8.5.9-sol10-x86-local.gz perl-5.12.2-sol10-x86-local.gz

mget libtool-2.4-sol10-x86-local.gz sed-4.2.1-sol10-x86-local.gz zlib-1.2.5-sol10-x86-local.gz binutils-
2.21-sol10-x86-local.gz groff-1.21-sol10-x86-local.gz bzip2-1.0.6-sol10-x86-local.gz

mget make-3.82-sol10-x86-local.gz sed-4.2.1-sol10-x86-local.gz gdb-6.8-sol10-x86-local.gz coreutils-8.9-
sol10-x86-local.gz cmake-2.6.0-sol10-x86-local.gz

quit
```

With all of the software downloaded, I next setup and configured sudo and python:

```
su
gunzip -v python-2.5.1-sol10-x86-local.gz
pkgadd -d python-2.5.1-sol10-x86-local

gunzip -v libintl-3.4.0-sol10-x86-local.gz libgcc-3.4.6-sol10-x86-local.gz libiconv-1.13.1-sol10-x86-
local.gz sudo-1.7.4p4-sol10-x86-local.gz

pkgadd -d libintl-3.4.0-sol10-x86-local
pkgadd -d libgcc-3.4.6-sol10-x86-local
pkgadd -d libiconv-1.13.1-sol10-x86-local
pkgadd -d sudo-1.7.4p4-sol10-x86-local

mkdir -p /usr/local/var/lib/
/usr/local/sbin/visudo
```

With sudo now working, I logged out and then back in. I then installed the other packages:

```
cd /tmp

gunzip -v *.gz

sudo pkgadd -d autoconf-2.68-sol10-x86-local
sudo pkgadd -d autogen-5.9.8-sol10-x86-local
sudo pkgadd -d automake-1.9-sol10-intel-local
sudo pkgadd -d binutils-2.21-sol10-x86-local
sudo pkgadd -d gcc-4.5.1-sol10-x86-local
sudo pkgadd -d groff-1.21-sol10-x86-local
sudo pkgadd -d libsigsegv-2.9-sol10-x86-local
sudo pkgadd -d make-3.82-sol10-x86-local
sudo pkgadd -d m4-1.4.15-sol10-x86-local
sudo pkgadd -d md5-6142000-sol10-intel-local
sudo pkgadd -d openssl-1.0.0c-sol10-x86-local
sudo pkgadd -d perl-5.12.2-sol10-x86-local
sudo pkgadd -d tcl-8.5.9-sol10-x86-local
sudo pkgadd -d tk-8.5.9-sol10-x86-local
sudo pkgadd -d zlib-1.2.5-sol10-x86-local
sudo pkgadd -d bzip2-1.0.6-sol10-x86-local
sudo pkgadd -d libtool-2.4-sol10-x86-local
sudo pkgadd -d sed-4.2.1-sol10-x86-local
sudo pkgadd -d gdb-6.8-sol10-x86-local
sudo pkgadd -d coreutils-8.9-sol10-x86-local
sudo pkgadd -d gmp-4.2.1-sol10-x86-local
sudo pkgadd -d cmake-2.6.0-sol10-x86-local
```

With those packages installed it was time to install the pieces of software which don't have pre-built packages:

Install Zope Interface:

```
cd /tmp
wget http://www.zope.org/Products/ZopeInterface/3.3.0/zope.interface-3.3.0.tar.gz
gunzip -v zope.interface-3.3.0.tar.gz
gtar -xf zope.interface-3.3.0.tar
cd zope.interface-3.3.0/
python setup.py build
sudo python setup.py install
```

Install the latest Twisted framework:

```
cd /tmp
wget http://tmrc.mit.edu/mirror/twisted/Twisted/10.2/Twisted-10.2.0.tar.bz2
bunzip2 Twisted-10.2.0.tar.bz2
gtar -xf Twisted-10.2.0.tar
cd Twisted-10.2.0
sudo python setup.py install
```

Install Bazaar:

```
cd /tmp
wget http://launchpad.net/bzr/2.2/2.2.2/+download/bzr-2.2.2.tar.gz
gunzip -v bzr-2.2.2.tar.gz
gtar -xf bzr-2.2.2.tar
cd bzr-2.2.2
sudo python setup.py install
```

Install ccache:

```
cd /tmp
wget http://samba.org/ftp/ccache/ccache-3.1.4.tar.gz
gunzip ccache-3.1.4.tar.gz
gtar xvf ccache-3.1.4.tar
cd ccache-3.1.4
./configure --prefix /usr
make
sudo make install
```

Configure and start NTP:

```
sudo cp /etc/inet/ntp.server /etc/inet/ntp.conf
sudo vi /etc/inet/ntp.conf

#
# Comment out the following lines:
#server 127.127.XType.0
#fudge 127.127.XType.0 stratum 0
#broadcast 224.0.1.1 ttl 4
#
# Add in the following lines:

server 0.us.pool.ntp.org
server 1.us.pool.ntp.org
server 2.us.pool.ntp.org
server 3.us.pool.ntp.org

# save the file and quit back the command prompt

sudo touch /var/ntp/ntp.drift
sudo ntpdate 0.us.pool.ntp.org
sudo svcadm enable svc:/network/ntp
```

Check out and make a test build of MariaDB:

```
cd
mkdir src
cd src/
bzr branch lp:maria trunk
cd trunk/
BUILD/compile-solaris-amd64
```

Add a user for buildbot:

```
sudo useradd -d /export/home/buildbot -m buildbot
```

Install Buildbot:

```
cd /tmp
wget http://buildbot.googlecode.com/files/buildbot-slave-0.8.3.tar.gz
gunzip -v buildbot-slave-0.8.3.tar.gz
gtar -xf buildbot-slave-0.8.3.tar
cd buildbot-slave-0.8.3/
sudo python setup.py install
```

Create the buildbot as the buildbot user:

On the build master, add new entry to /etc/buildbot/maria-master-private.cfg

Remember the ${slave-name} and ${password} configured above, they’re used in the next step.

Back on the solaris machine:

```
sudo su - buildbot
buildslave create-slave --usepty=0 /export/home/buildbot/maria-slave \
hasky.askmonty.org:9989 ${slavename} ${password}

echo '${contact-email-address}' > /export/home/buildbot/maria-slave/info/admin
echo 'A host running Solaris 10 x86.' > /export/home/buildbot/maria-slave/info/host

exit
```

Now start the slave:

```
sudo su - buildbot
buildslave start maria-slave
```

That's the basic process.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Ubuntu-Debian

Setting up a Buildbot Slave on Ubuntu and Debian

One great way to contribute to MariaDB development is to run a buildbot builder. These builders are used for running automated builds and tests of MariaDB. The instructions on this page should help you get a builder set up on Ubuntu and Debian.

Setting up the MariaDB Build Environment

For Ubuntu and Debian, a quick way to install much of what you need is:

```
sudo apt-get build-dep mariadb-server
```

If you're running a version of Debian or Ubuntu that doesn't have MariaDB, then do the following:

```
sudo apt-get build-dep mysql-server
```

After running one (or both) of the above, run the following to catch things that they may have missed:

```
sudo apt-get install devscripts fakeroot doxygen texlive-latex-base ghostscript libevent-dev libssl-dev
zlib1g-dev libpam0g-dev libreadline-gplv2-dev autoconf automake automake1.11 dpatch ghostscript-x
libfontenc1 libjpeg62 libltdl-dev libltdl7 libmail-sendmail-perl libxfont1 lmodern texlive-latex-base-doc
ttf-dejavu ttf-dejavu-extra libaio-dev xfonts-encodings xfonts-utils libxml2-dev unixodbc-dev bzr scons
check libboost-all-dev openssl epm libjudy-dev libjemalloc-dev libcrack2-dev git libkrb5-dev libcurl4-
openssl-dev thrift-compiler libsystemd-dev dh-systemd libssl1.0.2 openjdk-8-jdk uuid-dev libnuma-dev gdb
libarchive-dev libasio-dev dh-exec
```

After setting up the build environment do a test build to confirm that things are working. First get the source code using the **git** instructions on the Getting the MariaDB Source Code page, then follow the steps on the Generic Build Instructions page for building MariaDB using **cmake**. If your build succeeds, you're ready to move on to the next step of installing and configuring buildbot.

Do not hesitate to ask for help on the maria-developers mailing list or on IRC.

Buildbot Installation and Setup

Using APT

The easiest way to install buildbot on Ubuntu and Debian is to install the buildbot-slave package, like so:

```
sudo apt-get install buildbot-slave
```

Using Pip

Another way to install buildbot is using the Python **pip** package manager. Pip can be installed with:

```
sudo apt-get install python-pip
```

Next install twisted and the buildbot-slave package using pip:

```
sudo pip install twisted==11.0.0
sudo pip install buildbot-slave==0.8.9
```

Creating the Buildbot Builder

After the buildbot-slave package is installed (either via apt or pip), you need to create the builder using the `buildslave create-slave` command. As part of this command you will need to specify a name for your buildslave and a password. Both need to be given to the MariaDB Buildbot maintainers so that they can add your builder to the build pool. Ask on the maria-developers mailing list or on IRC for who these people are.

An example command for creating the slave is:

```
sudo buildslave create-slave /var/lib/buildbot/slaves/maria buildbot.askmonty.org slavename password
```

If you installed buildbot using pip, the convention is to create a buildbot user and then, as that user, create the buildslave in the home directory like so:

```
sudo buildslave create-slave ~/maria-slave buildbot.askmonty.org slavename password
```

Put some appropriate info in info/admin and info/host files that are created, this will display on the information screen about your builder. See here for example: bb01

Submit your builder information to the MariaDB Buildbot admins. Also let them know if your machine can run multiple builds at the same time (and how many). After adding your builder's information to the main buildbot configuration, all that's left is for you to do is to start your builder.

Starting and Stopping the Builder

If you installed your builder using apt, then you can start and stop it with:

```
sudo /etc/init.d/buildslave start
sudo /etc/init.d/buildslave stop
```

If you installed your build slave using pip, then do the following as the buildbot user in their home directory:

```
buildslave start maria-slave
buildslave stop maria-slave
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for VM Host

This page documents the general setup process for a server that is acting as virtual machine host, like those documented in the Buildbot Setup for Virtual Machines section.

Provision hardware with the most recent Ubuntu LTS release. Add host to DNS. Apply updates (replace `<host>` with hostname):

```
ssh <host>.mariadb.net
sudo apt-get update
sudo apt-get dist-upgrade
```

Install some favorite packages (these aren't necessarily required, but I like them)

```
sudo apt-get install tree renameutils vim-nox
```

A buildbot admin needs to add the new host to the allowed list of rsync clients on the VM master (whichever host is the official host of VM files) The VM master changes periodically, so check to make sure you have the correct one:

```
vi /etc/rsyncd.conf
```

Make a `/kvm/` dir and rsync it with the VM master above. The dir often resides at `/home/kvm/` (or wherever the storage drive is) and is then linked to `/kvm/`. The VMs rely on the `/kvm/vms/` path, so the `/kvm/` location is required:

```
vm_master="hostname"      # ask for this from a buildbot admin
mkdir /home/kvm
ln -sv /home/kvm /kvm
rsync --dry-run --delete --exclude=deprecated \\
  --exclude=iso --exclude=lost+found -avPL ${vm_master}::kvm/ /home/kvm/
# assuming the dry-run looks good, start the "real" rsync in a screen session
screen
rsync --delete --exclude=deprecated --exclude=iso --exclude=lost+found -avPL ${vm_master}::kvm/
/home/kvm/
```

Detach from screen session with `Ctrl+a d`. Configure vim.basic as the default editor (optional):

```
update-alternatives --config editor
```

Install buildbot-slave, bzr, and kvm

```
sudo apt-get install bzr git buildbot-slave qemu kvm
sudo apt-get install libsdl2-2.0-0
```

Add a default user, and then add the user to the appropriate groups:

```
username="mydefaultusername"
sudo adduser ${username}
for group in sudo tty kvm;do
  sudo adduser ${username} ${group}
done
```

Log out, then back in as the default user and change the password. Set up the `/.ssh/authorized_keys` file so you can login that way. Create other standard users and set up their ssh keys (optional).

Turn off password login:

> **Important:**
>
> Make sure to have your ssh key setup before doing this, and disallow all root logins and password logins (it is safer to only allow logins using ssh keys with regular users).

```
sudo perl -i -pe "s/#PasswordAuthentication yes/PasswordAuthentication no/" /etc/ssh/sshd_config
sudo perl -i -pe "s/PermitRootLogin yes/PermitRootLogin no/" /etc/ssh/sshd_config
sudo /etc/init.d/ssh restart
```

Check out `mariadb-tools`:

```
mkdir ~/src
cd ~/src/
bzr branch lp:mariadb-tools
```

Put `runvm` in the right place:

```
sudo cp -v ~/src/mariadb-tools/buildbot/runvm /usr/local/bin/
ls -l /usr/local/bin/
```

Add the buildbot user to the `kvm` and `tty` groups:

```
sudo adduser buildbot kvm
sudo adduser buildbot tty
```

A buildbot admin needs to add this builder to the `maria-master-private.cfg` file on the `${buildmaster}` and also add it to the `c['slaves']` array in `maria-master.cfg` then create the buildslave using the hostname and whatever `${password}` was agreed upon by you and the buildbot admin:

```
sudo buildslave create-slave /var/lib/buildbot/slaves/maria buildbot.askmonty.org ${host} ${password}
```

Add the following to `/etc/default/buildslave` (replace `${hostname}` with the name of the host):

```
HOME=/var/lib/buildbot
SLAVE_ENABLED[1]=1
SLAVE_NAME[1]="${hostname} maria slave"
SLAVE_USER[1]="buildbot"
SLAVE_BASEDIR[1]="$HOME/slaves/maria"
SLAVE_OPTIONS[1]=""
SLAVE_PREFIXCMD[1]=""
```

Edit the admin and host files and add contact information and details on the builder:

```
sudo vi /var/lib/buildbot/slaves/maria/info/*
```

Copy over the buildbot .ssh dir from terrier:

```
scp terrier.askmonty.org:buildbot-ssh.tar.gz .

cd /var/lib/buildbot
sudo tar -zxvf ~/buildbot-ssh.tar.gz
sudo chown -Rv buildbot: .ssh
sudo chmod -v 700 .ssh
sudo chmod -Rv go-r .ssh
```

Edit `/etc/passwd` and change the buildbot user's shell from `/bin/false` to `/bin/bash` , then su to the buildbot user and copy in the `/etc/skel` files:

```
sudo su - buildbot
cp -v /etc/skel/.bash* .
cp -v /etc/skel/.profile .
exit
```

Change ownership of the `buildbot/slaves` dir to `buildbot:buildbot` :

```
sudo chown -Rv buildbot:buildbot ~buildbot/slaves
```

Move the `/var/lib/buildbot` directory to `/home` (or whatever location you want to use to store things) and then link it back:

```
sudo mv -vi /var/lib/buildbot /home/;cd /var/lib/;sudo ln -sv /home/buildbot ./
```

Update `/etc/default/locale` and change it to: `LANG=en_US.UTF-8` :

```
sudo vi /etc/default/locale
sudo locale-gen
```

Monitor the rsync, wait for it to finish. Once the rsync is finished, test the runvm script:

```
sudo su - buildbot
for i in '/kvm/vms/vm-xenial-amd64-serial.qcow2 6666 qemu64' '/kvm/vms/vm-xenial-i386-serial.qcow2 6666
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/testtest/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils unixodbc libxml2
libjudydebian1" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

Remove the “testtest” VMs we created above

```
rm -v /kvm/vms/*testtest*
```

Start the buildslave:

```
sudo /etc/init.d/buildslave start

tail -f ~buildbot/slaves/maria/twistd.log
```

ssh to `${buildmaster}` and add this new host to `kvm_slaves` in the `maria-master.cfg` file:

```
sudo vi /etc/buildbot/maria-master.cfg
```

Still on `${buildmaster}`, test and then reload buildbot:

```
cd /etc/buildbot
sudo -u buildbot PYTHONPATH=/usr/local/buildbot/lib/python python -c 'exec open("maria-master.cfg", "r")'
sudo /etc/init.d/buildmaster reload
sudo tail -f /var/lib/buildbot/maria/twistd.log
```

This page is licensed: CC BY-SA / Gnu FDL

Setup of a Buildbot Build Slave

These build instructions should in general be platform agnostic. It is based on a post by Adam M. Dutko on maria-discuss. See the list archive.

To setup a build slave, the basic outline is:

1. Install Python (2.4 or later)
2. Install Twisted (twistedmatrix.com 8.0.x or later) - need Core, Mail, Web and Words (possibly Conch too)
3. Install ZopeInterface (ZopeInterface)
4. Install all the necessary compiler/build components.
5. Install Buildbot
6. Install Git and make sure you can clone from the main project

7. Get a username/login and password from the buildmaster (dbart on Libera.Chat)
8. Verify your buildbot can talk to the buildbot master instance and can run builds.

The following links will also help:

1.
2. buildbot-setup-buildbot-setup-notes
3. about-buildbot

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines

This section documents the exact steps used to set up each virtual machine used in the MariaDB Buildbot for building packages and testing. There are lots of boring details, but the information here should be useful for reference, and as a help when installing new virtual images for additional distros and versions.

The articles here document how the virtual machine images were actually set up, not the most efficient way to recreate all of them. Thus, some historical artifacts remain, reflecting the order they were initially added and lessons learned along the way.

For example, the steps for the earliest VMs are somewhat overly enthusiastic about using the `qemu-img create -b` option to make one vm a copy-on-write copy on top of another image. This makes it harder to modify the images afterwards, since if the base image is modified, any copy-on-write vm on top of it needs to be re-created.

The descriptions do not go into painful detail about the installation procedure.
Generally the .iso for the distro was downloaded, and the default, non-graphical installer started. Installing Linux these days is quite easy, and generally just defaults were picked. With respect to harddisk partitioning, generally an 8Gb image was configured, of which 1/2 Gb was reserved for swap and the rest for a single root partition “/”.

For details on setting up a host server to run these VMs, see Buildbot Setup for VM host.

See Buildbot Setup for Virtual Machines - FreeBSD 9.2.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - General Principles.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Additional Steps.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Debian.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Red Hat.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - FreeBSD 9.2

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-freebsd92-amd64-serial.qcow2 15G
qemu-img create -f qcow2 /kvm/vms/vm-freebsd92-i386-serial.qcow2 15G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 2048 -hda /kvm/vms/vm-freebsd92-amd64-serial.qcow2 -cdrom /kvm/iso/freebsd/9.2/FreeBSD-9.2-
RELEASE-amd64-dvd1.iso -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net
user,hostfwd=tcp:127.0.0.1:2283-:22
kvm -m 2048 -hda /kvm/vms/vm-freebsd92-i386-serial.qcow2 -cdrom /kvm/iso/freebsd/9.2/FreeBSD-9.2-RELEASE-
i386-dvd1.iso -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2284-:22
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote_host} localhost
```

Replace `${remote_host}` with the host the vm is running on.

> **Note:**
>
> When you activate the install, vncviewer will disconnect after a bit with a complaint about the rect being too large. This is fine. The installer has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- Keymap Selection: Do not set non-default key mapping
- Hostname: to freebsd-92-amd64 or freebsd-92-i386
- Distribution Select: deselect “games”, leave “ports” selected
- Partitioning: Guided and use the entire disk
  - Accept the default partitioning (Finish then Commit)
- Archive Extraction: …wait while installer installs the OS…
- Network Configuration: choose `vtnet0`
  - configure ipv4 w/ DHCP
  - do not configure ipv6
- System Configuration: leave sshd selected, don’t select the others
- Dumpdev Configuration: do not enable crash dumps
- add a buildbot user
  - add user to ‘wheel’ group
- Final Configuration: Apply configuration and exit
- Manual Configuration: yes to opening a shell
  - `echo 'console="comconsole"' >> /boot/loader.conf`
  - Edit /etc/ttys and change off to on and dialup to vt100 for the ttyu0 entry.
  - `shutdown -p now`

Now that the VM is installed, it’s time to configure it.
If you have the memory you can do the following simultaneously:

```
kvm -m 2048 -hda /kvm/vms/vm-freebsd92-amd64-serial.qcow2 -boot c -smp 2 -cpu qemu64 -net
nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2283-:22 -nographic
kvm -m 2048 -hda /kvm/vms/vm-freebsd92-i386-serial.qcow2 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio
-net user,hostfwd=tcp:127.0.0.1:2284-:22 -nographic

ssh -p 2283 localhost
ssh -p 2284 localhost

su -

cd /usr/ports/security/sudo; make install clean # accept defaults
visudo # uncomment the '# %wheel ALL=(ALL) NOPASSWD: ALL' line
exit

cd /usr/ports/ports-mgmt/portmaster; sudo make install clean # select both autocomplete options

ssh hasky.askmonty.org
exit
sudo su buildbot -
ssh hasky.askmonty.org
exit
exit
exit

scp -P 2283 /kvm/vms/authorized_keys localhost:.ssh/
scp -P 2283 /kvm/vms/authorized_keys buildbot@localhost:.ssh/

scp -P 2284 /kvm/vms/authorized_keys localhost:.ssh/
scp -P 2284 /kvm/vms/authorized_keys buildbot@localhost:.ssh/

ssh -p 2283 localhost
ssh -p 2284 localhost

sudo portsnap fetch extract
sudo portsnap fetch update
sudo portmaster -Da

sudo shutdown -p now
```

VMs for building BSD bintars

```
for i in '/kvm/vms/vm-freebsd92-amd64-serial.qcow2 2283 qemu64' '/kvm/vms/vm-freebsd92-i386-serial.qcow2
2284 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "cd /usr/ports/devel/libevent;     sudo make install clean BATCH=yes" \
    "cd /usr/ports/devel/boost-all;    sudo make install clean BATCH=yes" \
    "cd /usr/ports/devel/cmake;        sudo make install clean BATCH=yes" \
    "cd /usr/ports/devel/thrift/;      sudo make install clean BATCH=yes" \
    "cd /usr/ports/devel/bzr/;         sudo make install clean BATCH=yes" \
    "cd /usr/ports/devel/bison/;       sudo make install clean BATCH=yes" \
    "cd /usr/ports/databases/unixODBC; sudo make install clean BATCH=yes" \
    "cd /usr/ports/devel/doxygen;      sudo make install clean BATCH=yes"; \
done
```

VMs for install testing.

```
qemu-img create -b /kvm/vms/vm-freebsd92-amd64-serial.qcow2 -f qcow2 /kvm/vms/vm-freebsd92-amd64-
install.qcow2
qemu-img create -b /kvm/vms/vm-freebsd92-i386-serial.qcow2 -f qcow2 /kvm/vms/vm-freebsd92-i386-
install.qcow2

for i in '/kvm/vms/vm-freebsd92-amd64-serial.qcow2 2283 qemu64' '/kvm/vms/vm-freebsd92-i386-serial.qcow2
2284 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo portsnap cron update"; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-freebsd92-amd64-serial.qcow2 2283 qemu64' '/kvm/vms/vm-freebsd92-i386-serial.qcow2
2284 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade/')" \
    "cd /usr/ports/databases/mysql55-server;    sudo make install clean BATCH=yes" \
    "sudo /usr/local/etc/rc.d/mysql-server onestart" \
    'mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into
t values (1); select * from t"' ;\
done
```

VMs for MariaDB upgrade testing

```
for i in '/kvm/vms/vm-freebsd92-amd64-serial.qcow2 2283 qemu64' '/kvm/vms/vm-freebsd92-i386-serial.qcow2
2284 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    "cd /usr/ports/databases/mariadb55-server;    sudo make install clean BATCH=yes" \
    "sudo /usr/local/etc/rc.d/mysql-server onestart" \
    'mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into
t values (1); select * from t"'; \
done
```

Add Key to known_hosts

Do the following on each kvm host server (terrier, terrier2, i7, etc…) to add the VMs to known_hosts.

```
# freebsd92-amd64
cp -avi /kvm/vms/vm-freebsd92-amd64-install.qcow2 /kvm/vms/vm-freebsd92-amd64-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-freebsd92-amd64-test.qcow2 -redir tcp:2283::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2283 buildbot@localhost sudo shutdown -p now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-freebsd92-amd64-test.qcow2

# freebsd92-i386
cp -avi /kvm/vms/vm-freebsd92-i386-install.qcow2 /kvm/vms/vm-freebsd92-i386-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-freebsd92-i386-test.qcow2 -redir tcp:2284::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2284 buildbot@localhost sudo shutdown -p now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-freebsd92-i386-test.qcow2
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - General Principles

The installations are kept minimal, picking mostly default options. This helps ensure our stuff works on default installs, and also saves effort during the creation of the virtual machines.

Since most of the logic happens in the Buildbot slave on a host machine, and inside the Buildbot master, there is no need to install Buildbot or other similar complex stuff inside the VMs.

ssh

The virtual machines are configured with ssh access. An account called
'buildbot' is created, with passwordless login (using ssh public key authentication), and with passwordless sudo access (needed for automated scripting). The public key for user buildbot is installed in every VM, and the private key is added to the account running the Buildbot master on the host machine.

To generate the ssh keys:

```
ssh-keygen -t dsa
```

Leave the passphrase empty. Run this as the user running the Buildbot slave on the KVM host machine. The resulting /.ssh/id_dsa.pub will be needed below for each virtual machine install.

Serial port

The vms are configured to use the (emulated) serial port as the console. When running KVM on the host, the console then maps to the stdin/stdout. This is useful to get kernel log messages available to easier debug any problems. The bootloader Grub is similarly configured to use the serial port.

The vms are also set up to give a login prompt on the serial port (running getty). In retrospect this has proven not really needed, as if needing to manually log in and investigate things, it is often as easy to just run KVM in graphics mode. It may be useful in cases where the host machine is remote though (in this case graphics mode does still work but can be a bit slow).

To configure for serial console, the following two files are referenced below, and should be prepared in advance:

`ttyS0`

```
# ttyS0 - getty
#
# This service maintains a getty on ttyS0 from the point the system is
# started until it is shut down again.

start on stopped rc2
start on stopped rc3
start on stopped rc4
start on stopped rc5

stop on runlevel 0
stop on runlevel 1
stop on runlevel 6

respawn
exec /sbin/getty 115200 ttyS0
```

`ttyS0.conf`

```
# ttyS0 - getty
#
# This service maintains a getty on ttyS0 from the point the system is
# started until it is shut down again.

start on stopped rc RUNLEVEL=[2345]
stop on runlevel [!2345]

respawn
exec /sbin/getty -L 115200 ttyS0 vt102
```

Unattended MariaDB install on Debian/Ubuntu

my.seed

The MariaDB (and MySQL) package on Debian and Ubuntu prompts the user about the root password to set. We want to test this important step, but we do not want the prompt of course. To do this we use the following configuration file for defaults for debconf. This file is needed in the steps below under the name "my.seed" (be careful to preserve it exactly as here, including trailing space and tab characters!)

```
mariadb-server-5.1  mysql-server/root_password_again    password    rootpass
    mariadb-server-5.1  mysql-server/root_password  password    rootpass
    mariadb-server-5.1  mysql-server/error_setting_password error
    mariadb-server-5.1  mysql-server-5.1/nis_warning    note
    mariadb-server-5.1  mysql-server-5.1/really_downgrade   boolean false
    mariadb-server-5.1  mysql-server-5.1/start_on_boot  boolean true
    mariadb-server-5.1  mysql-server-5.1/postrm_remove_databases    boolean false
    mariadb-server-5.1  mysql-server/password_mismatch  error
    mariadb-server-5.1  mysql-server/no_upgrade_when_using_ndb  error
```

For some background see here.
The file my.seed can be generated from an existing install using debconf-get-selections.

sources.append

For Debian/Ubuntu, we add a local repository to the list of apt source to be able to test `apt-get install`. For this, the following file "sources.append" must be prepared:

```
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
```

Emulation options

The default network card emulated by KVM has poor performance. To solve this we instead use the "virtio" network device, using the KVM options "-net nic,model=virtio -net user". Except on Debian 4, which has an old kernel without support for virtio. Background info here.

A detail is that some 32-bit vms crashed during boot with default options. This was fixed by using the kvm "-cpu qemu32,-nx" option. Some background information here.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Additional Steps

See Install cmake on Build VMs.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Installing Correct Libraries for PAM and readline.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Installing the Boost Library Needed for the OQGraph Storage Engine.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Installing VM Images for Testing .deb Upgrade Between Versions.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Resizing a Virtual Machine Image.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Update Debian 4 Mirrors for Buildbot VMs.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

This page is licensed: CC BY-SA / Gnu FDL

Install cmake on Build VMs

Requires cmake. Install cmake on all -build VMs (and other Unix-like machines) with:

```
wget http://www.cmake.org/files/v2.8/cmake-2.8.8.tar.gz
tar -zxvf cmake-2.8.8.tar.gz
cd cmake-2.8.8;./configure
make
sudo make install
```

This page is licensed: CC BY-SA / Gnu FDL

Installing Correct Libraries for PAM and readline

Some additional/alternative libraries needs to be installed to handle readline and PAM correctly.

The newer libreadline is GPLv3 and so not compatible with the MariaDB/MySQL
GPLv2 license. The PAM libraries are needed for the PAM plugin.

On the Centos and RHEL -build VMs, install the pam-devel package:

```
sudo yum install pam-devel
```

On all the Debian/Ubuntu -build virtual machines, install libpam0g-dev:

```
sudo apt-get install libpam0g-dev
```

On debian6/maverick/natty, install libreadline5-dev (replacing any libreadline6-dev already there):

```
sudo apt-get install libreadline5-dev
```

On oneiric (and any newer, eg. Debian 7 or Ubuntu 12.04), the package is called libreadline-gplv2-dev:

```
sudo apt-get install libreadline-gplv2-dev
```

This page is licensed: CC BY-SA / Gnu FDL

Installing the Boost Library Needed for the OQGraph Storage Engine

The OQGraph storage engine needs a newer version of Boost that what is available on (most) distributions. The version installed is 1.42.0, available from.

The boost library is installed in all the builder virtual machine images with the following single command:

```
for i in "vm-hardy-amd64-build qemu64" "vm-hardy-i386-build qemu32,-nx" \
        "vm-intrepid-amd64-build qemu64" "vm-intrepid-i386-build qemu32,-nx" \
        "vm-karmic-amd64-build qemu64" "vm-karmic-i386-build qemu32,-nx" \
        "vm-jaunty-amd64-build qemu64" "vm-jaunty-i386-deb-build qemu32,-nx" \
        "vm-lucid-amd64-build qemu64" "vm-lucid-i386-build qemu32,-nx" \
        "vm-maverick-amd64-build qemu64" "vm-maverick-i386-build qemu32,-nx" \
        "vm-natty-amd64-build qemu64" "vm-natty-i386-build qemu64" \
        "vm-oneiric-amd64-build qemu64" "vm-oneiric-i386-build qemu64" \
        "vm-debian5-amd64-build qemu64" "vm-debian5-i386-build qemu32,-nx" \
        "vm-debian4-amd64-build qemu64 --netdev=e1000" "vm-debian4-i386-build qemu32,-nx --netdev=e1000"
\
        "vm-centos5-i386-build qemu32,-nx" "vm-centos5-amd64-build qemu64" \
        "vm-hardy-amd64-build qemu64" "vm-hardy-i386-build qemu32,-nx" \
        "vm-jaunty-i386-deb-tarbake qemu32,-nx" ; do \
  set $i; \
  runvm -m 512 --smp=1 --port=2200 --user=buildbot --cpu=$2 $3 $1.qcow2 \
    "= scp -P 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/boost_1_42_0.tar.gz
buildbot@localhost:/dev/shm/" \
    "sudo mkdir -p /usr/local/src /usr/local/include" \
    "cd /usr/local/src && sudo tar zxf /dev/shm/boost_1_42_0.tar.gz" \
    "cd /usr/local/include && sudo ln -s ../src/boost_1_42_0/boost ." ; \
done
```

Upgrade Boost to 1.49

To upgrade boost on the VMs to 1.49 I performed the following steps:

I initially tried to upgrade the VMs using a script like the one at the top of the page which was used to install boost 1.42, but I ran into issues with getting it to work on all of the VMs (it worked on some, but not on others). So I ended up using the steps below.

1. Copy the VM (keeping the original as a backup, in case something goes wrong):

```
oldvm="vm-debian6-i386-build.qcow2"
newvm="vm-debian6-i386-build.upd.qcow2"
cp -avi ${oldvm} ${newvm}
```

1. Start the VM:

```
kvm -m 1024 -hda /kvm/vms/${newvm} -redir 'tcp:22775::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=virtio -net user -nographic
```

1. Copy the boost tar.gz file over to the VM:

```
scp -i /kvm/vms/ssh-keys/id_dsa -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -P 22775
/kvm/boost_1_49_0.tar.gz buildbot@localhost:/dev/shm/
```

1. ssh into the VM:

```
ssh -i /kvm/vms/ssh-keys/id_dsa -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -p 22775
buildbot@localhost
```

1. once inside the VM, perform the following steps:

```
cd /usr/local/src
sudo tar zxf /dev/shm/boost_1_49_0.tar.gz
cd /usr/local/include/
sudo rm boost
sudo ln -s ../src/boost_1_49_0/boost .
sudo /sbin/shutdown -h now
```

This page is licensed: CC BY-SA / Gnu FDL

Installing VM Images for Testing .deb Upgrade Between Versions

This step creates virtual machine images used to do an important additional upgrade test for .debs. Each virtual machine is pre-installed with an older version of MariaDB, and the test tries upgrading this version to the newly build one (to check that dependencies etc. work out correctly).

This step may not be easily possible to replicate exactly for new or updated images. For example, when we add a new platform/distro, we will not have the same old version of MariaDB available for installation on the new planform. Or if needing to re-create an image in the future, the original old .debs used before may no longer be available. However, this is not a big problem, as we can just use whatever version of MariaDB is available. In fact, while we cannot reasonably test every possible upgrade combination between MariaDB versions, it is still useful to test upgrades from different versions on different platforms, to increase coverage a bit.

The bulk of the images were installed with the following two loops, using packages from the OurDelta repository for MariaDB 5.1.42, which was the one available from there at the time of installation.

```
for i in "vm-hardy-amd64-install qemu64 hardy" "vm-hardy-i386-install qemu32,-nx hardy" \
          "vm-intrepid-amd64-install qemu64 intrepid" "vm-intrepid-i386-install qemu32,-nx intrepid" \
          "vm-karmic-amd64-install qemu64 karmic" "vm-karmic-i386-install qemu32,-nx karmic" \
          "vm-jaunty-amd64-install qemu64 jaunty" "vm-jaunty-i386-deb-install qemu32,-nx jaunty" \
          "vm-debian5-amd64-install qemu64 lenny" "vm-debian5-i386-install qemu32,-nx lenny" ; do \
  set $i; \
  runvm -b  $1.qcow2 -m 512 --smp=1 --port=2200 --user=buildbot --cpu=$2 $(echo $1 | sed -e 's/-install/-
upgrade2/').qcow2 \
    "wget -O- http://ourdelta.org/deb/ourdelta.gpg | sudo apt-key add -" \
    "sudo sh -c \"echo 'deb http://mirror.ourdelta.org/deb $3 mariadb-ourdelta' >
/etc/apt/sources.list.d/ourdelta.list\"" \
    "sudo apt-get update || true" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mariadb-server mariadb-test libmariadbclient-
dev || true" \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    "sudo rm /etc/apt/sources.list.d/ourdelta.list" \
    "sudo apt-get update || true" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y || true" ; \
done
```

```
for i in "vm-debian4-amd64-install qemu64 etch /kvm/debian-40r8-amd64-netinst.iso" "vm-debian4-i386-
install qemu32,-nx etch /kvm/debian-40r8-i386-netinst.iso" ; do \
  set $i; \
  runvm -b  $1.qcow2 -m 512 --smp=1 --port=2200 --user=buildbot --cpu=$2 --netdev=e1000 --kvm=-cdrom --
kvm=$4 $(echo $1 | sed -e 's/-install/-upgrade2/').qcow2 \
    "wget -O- http://ourdelta.org/deb/ourdelta.gpg | sudo apt-key add -" \
    "sudo sh -c \"echo 'deb http://mirror.ourdelta.org/deb $3 mariadb-ourdelta' >
/etc/apt/sources.list.d/ourdelta.list\"" \
    "sudo apt-get update || true" \
    "sudo sh -c 'DEBIAN_FRONTEND=noninteractive apt-get install -y mariadb-server mariadb-test
libmariadbclient-dev' || true" \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    "sudo rm /etc/apt/sources.list.d/ourdelta.list" \
    "sudo apt-get update || true" \
    "sudo sh -c 'DEBIAN_FRONTEND=noninteractive apt-get upgrade -y' || true" ; \
done
```

The Ubuntu 10.04 “lucid” images were installed manually (as no packages for lucid were available from OurDelta at the time of installation):

Create and boot 64-bit lucid upgrade image:

```
qemu-img create -b vm-lucid-amd64-install.qcow2 -f qcow2 vm-lucid-amd64-upgrade2.qcow2
kvm -m 512 -hda vm-lucid-amd64-upgrade2.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=virtio -net user -nographic
```

Create directory inside the image:

```
mkdir buildbot
```

Copy in packages to be installed from outside:

```
scp -P 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -r /archive/pack/mariadb-5.1-
knielsen/build-277/kvm-deb-lucid-amd64/debs buildbot@localhost:buildbot/
```

Install the MariaDB packages, remove package dir, and upgrade to latest security fixes:

```
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
rm -Rf buildbot
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
```

Now the same, but for 32-bit lucid:

```
qemu-img create -b vm-lucid-i386-install.qcow2 -f qcow2 vm-lucid-i386-upgrade2.qcow2
kvm -m 512 -hda vm-lucid-i386-upgrade2.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu32,-nx -net
nic,model=virtio -net user -nographic
```

Create directory inside the image:

```
mkdir buildbot
```

Copy in packages to be installed from outside:

```
scp -P 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -r /archive/pack/mariadb-5.1-
knielsen/build-277/kvm-deb-lucid-x86/debs buildbot@localhost:buildbot/
```

Install the MariaDB packages, remove package dir, and upgrade to latest security fixes:

```
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
rm -Rf buildbot
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
```

The Ubuntu 10.10 “maverick” images were likewise installed manually:

Create and boot 64-bit maverick upgrade image:

```
qemu-img create -b vm-maverick-amd64-install.qcow2 -f qcow2 vm-maverick-amd64-upgrade2.qcow2
kvm -m 512 -hda vm-maverick-amd64-upgrade2.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=virtio -net user -nographic

ssh -p 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost mkdir buildbot
scp -P 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -r /archive/pack/mariadb-5.1-
knielsen/build-619/kvm-deb-maverick-amd64/debs buildbot@localhost:buildbot/
ssh -p 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost

sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
rm -Rf buildbot
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
```

Same for 32-bit maverick:

```
qemu-img create -b vm-maverick-i386-install.qcow2 -f qcow2 vm-maverick-i386-upgrade2.qcow2
kvm -m 512 -hda vm-maverick-i386-upgrade2.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu32,-nx -net
nic,model=virtio -net user -nographic

ssh -p 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost mkdir buildbot
scp -P 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -r /archive/pack/mariadb-5.1-
knielsen/build-619/kvm-deb-maverick-x86/debs buildbot@localhost:buildbot/
ssh -p 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost

sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
rm -Rf buildbot
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
```

For Ubuntu 11.04 “natty”, mariadb packages were installed from the repository for the previous version.

64-bit Ubuntu natty:

```
qemu-img create -b vm-natty-amd64-install.qcow2 -f qcow2 vm-natty-amd64-upgrade2.qcow2
kvm -m 512 -hda vm-natty-amd64-upgrade2.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=virtio -net user -nographic
ssh -p 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost

sudo sh -c 'cat > /etc/apt/sources.list.d/tmp.list'
deb http://ftp.osuosl.org/pub/mariadb/repo/5.1/ubuntu maverick main
deb-src http://ftp.osuosl.org/pub/mariadb/repo/5.1/ubuntu maverick main

sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
sudo rm /etc/apt/sources.list.d/tmp.list
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
```

32-bit Ubuntu natty:

```
qemu-img create -b vm-natty-i386-install.qcow2 -f qcow2 vm-natty-i386-upgrade2.qcow2
kvm -m 512 -hda vm-natty-i386-upgrade2.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=virtio -net user -nographic
ssh -p 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost

sudo sh -c 'cat > /etc/apt/sources.list.d/tmp.list'
deb http://ftp.osuosl.org/pub/mariadb/repo/5.1/ubuntu maverick main
deb-src http://ftp.osuosl.org/pub/mariadb/repo/5.1/ubuntu maverick main

sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
sudo rm /etc/apt/sources.list.d/tmp.list
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
```

For Ubuntu 11.10 “oneiric”. The steps performed are listed at: Buildbot Setup for Virtual Machines - Ubuntu 11.10 “oneiric” (in the VMs for Upgrade Testing section)

This page is licensed: CC BY-SA / Gnu FDL

Resizing a Virtual Machine Image

Some KVM images end up not having enough space on them. In such cases, it is preferable to increase the size of the VM rather than to just delete an image and rebuild it from scratch. The steps outlined below document what was done to increase the size of the Red Hat 5 x86 VM and should be able to be easily adapted to other VMs, should they need the same treatment in the future.

1. Make a copy of the VM to work on (we don’t want to change the original):

```
cp -avi vm-rhel5-x86-build.qcow2 vm-rhel5-x86-build-new.qcow2
```

1. Using the `qemu-img` command, resize the image:

```
qemu-img info vm-rhel5-x86-build-new.qcow2
qemu-img resize vm-rhel5-x86-build-new.qcow2 +10G
qemu-img info vm-rhel5-x86-build-new.qcow2
rsync -avP vm-rhel5-x86-build-new.qcow2 terrier:/kvm/vms/
```

Not all versions of `qemu-img` can resize VMs.

1. Boot the newly resized image with gparted:

```
vm=vm-rhel5-x86-build-new.qcow2
kvm -m 2048 -hda /kvm/vms/${vm} -cdrom /kvm/iso/gparted-live-0.14.1-6-i486.iso -boot d -smp 2 -cpu qemu64
-net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:22666-:22
```

1. Connect to the VM using VNC from your local machine:

```
vncviewer -via <vmhost> localhost
```

Midway through booting you’ll have to reconnect

1. Use gparted to either expand the existing primary partition or, especially on VMs with LVM, add a new partition (since GParted can’t change LVM partitions). Exit when finished and shutdown the VM.
2. Boot the VM again, this time without a VNC server:

```
kvm -m 2048 -hda /kvm/vms/${vm} -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net
user,hostfwd=tcp:127.0.0.1:22666-:22 -nographic
```

1. login to the VM:

```
ssh -t -p 22666 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
buildbot@localhost
```

1. if expanded an existing partition: verify the new size
2. else if you created a new partition:

- mount the new partition to tmp location

```
sudo mount /dev/hda3 /mnt
```

- rsync contents of /home/ to the new partition

```
sudo rsync -avP /home/ /mnt/
```

- edit fstab to mount new partition to /home

```
sudo vi /etc/fstab
```

- mv `/home` to `/home-old`, create `/home` dir, mount `/home`

```
sudo mv -vi /home /home-old;sudo mkdir -v /home;sudo mount /home
```

- (optional) unmount `/mnt`

```
sudo umount /mnt
```

- reboot and verify that things look good

```
sudo /sbin/shutdown -h now
```

- if things do look good (new drive mounted OK, accounts work, etc…), delete `/home-old`

1. Move the old VM to `-old` and the `-new` VM to what the old VM used to be named

```
sudo mv -vi /kvm/vms/vm-rhel5-x86-build.qcow2 vm-rhel5-x86-build-old.qcow2; sudo mv -vi /kvm/vms/vm-
rhel5-x86-build-new.qcow2 /kvm/vms/vm-rhel5-x86-build.qcow2
```

1. on other VM hosts, make a copy of the old file then rsync over the updated files (the copy helps speed up the rsync):

```
sudo cp -avi /kvm/vms/vm-rhel5-x86-build.qcow2 /kvm/vms/vm-rhel5-x86-build-old.qcow2
sudo rsync -avP terrier.askmonty.org::kvm/vms/vm-rhel5-x86-build* /kvm/vms/
```

1. Test the new VM with a build to make sure it works

This page is licensed: CC BY-SA / Gnu FDL

Update Debian 4 Mirrors for Buildbot VMs

Debian 4 has become so old that the apt repository has been moved out of the main Debian 4 mirror servers, and into the archive of old versions. This needs to be fixed by pointing the Debian 4 images to a different mirror:

64-bit:

```
kvm -m 512 -hda vm-debian4-amd64-install.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=e1000 -net user -nographic

sudo vi /etc/apt/sources.list
# replace http://ftp.dk.debian.org/debian/ with http://ftp.de.debian.org/archive/debian/
```

32-bit:

```
kvm -m 512 -hda vm-debian4-i386-install.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=e1000 -net user -nographic

sudo vi /etc/apt/sources.list
# replace http://ftp.dk.debian.org/debian/ with http://ftp.de.debian.org/archive/debian/
```

After that, it is necessary to re-do from scratch the -update and -update2 debian4 images (as these are built on top of the -install images).

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Debian

See Buildbot Setup for Virtual Machines - Debian 4 amd64.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Debian 4 i386.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Debian 5 amd64.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Debian 5 i386.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Debian 6 amd64.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Debian 6 i386.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Debian 4 amd64

```
cd /kvm/vms
qemu-img create -f qcow2 vm-debian4-amd64-serial.qcow2 8G
kvm -m 2047 -hda /kvm/vms/vm-debian4-amd64-serial.qcow2 -cdrom /kvm/debian-40r8-amd64-netinst.iso -redir
'tcp:2240::22' -boot d -smp 2 -cpu qemu64 -net nic,model=e1000 -net user
```

Serial console and account setup

From base install, setup for serial port, and setup accounts for passwordless ssh login and sudo:

```
kvm -m 2047 -hda /kvm/vms/vm-debian4-amd64-serial.qcow2 -cdrom /kvm/debian-40r8-amd64-netinst.iso -redir
'tcp:2240::22' -boot c -smp 2 -cpu qemu64 -net nic,model=e1000 -net user

su
apt-get install sudo openssh-server
VISUAL=vi visudo
# Add at the end:t %sudo ALL=NOPASSWD: ALL

# Add account <USER> to group sudo
# Copy in public ssh key.

# Add in /etc/inittab:
S0:2345:respawn:/sbin/getty -L ttyS0 19200 vt100
```

Add to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line:

```
console=tty0 console=ttyS0,115200n8

# Add user buildbot, with disabled password. Add as sudo, and add ssh key.
sudo /usr/sbin/adduser --disabled-password buildbot
sudo /usr/sbin/adduser buildbot sudo
sudo su - buildbot
mkdir .ssh
# Add all necessary keys.
cat >.ssh/authorized_keys
chmod -R go-rwx .ssh
```

VM for build

```
qemu-img create -b vm-debian4-amd64-serial.qcow2 -f qcow2 vm-debian4-amd64-build.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian4-amd64-build.qcow2 -cdrom /kvm/debian-40r8-amd64-netinst.iso -redir
'tcp:2240::22' -boot c -smp 2 -cpu qemu64 -net nic,model=e1000 -net user -nographic

sudo apt-get build-dep mysql-server-5.0
# Some latex packages fail to install because they complain that the
# source is more than 5 years old! I solved by setting back the clock a
# couple of years temporarily ...
sudo apt-get install devscripts doxygen texlive-latex-base gs lsb-release fakeroot libevent-dev libssl-
dev zlib1g-dev libreadline5-dev
```

VM for install testing

```
qemu-img create -b vm-debian4-amd64-serial.qcow2 -f qcow2 vm-debian4-amd64-install.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian4-amd64-install.qcow2 -cdrom /kvm/debian-40r8-amd64-netinst.iso -redir
'tcp:2240::22' -boot c -smp 2 -cpu qemu64 -net nic,model=e1000 -net user -nographic
```

See the General Principles article for how to make the '`my.seed`' file.

```
# No packages mostly!
sudo apt-get install debconf-utils

cat >>/etc/apt/sources.list <<END
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
END

sudo debconf-set-selections /tmp/my.seed
```

VM for upgrade testing

```
qemu-img create -b vm-debian4-amd64-install.qcow2 -f qcow2 vm-debian4-amd64-upgrade.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian4-amd64-upgrade.qcow2 -cdrom /kvm/debian-40r8-amd64-netinst.iso -redir
'tcp:2240::22' -boot c -smp 2 -cpu qemu64 -net nic,model=e1000 -net user -nographic

sudo apt-get install mysql-server-5.0
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Debian 4 i386

Create the VM:

```
cd /kvm/vms
qemu-img create -f qcow2 vm-debian4-i386-serial.qcow2 8G
kvm -m 2047 -hda /kvm/vms/vm-debian4-i386-serial.qcow2 -cdrom /kvm/debian-40r8-i386-netinst.iso -redir
'tcp:2241::22' -boot d -smp 2 -cpu qemu32,-nx -net nic,model=e1000 -net user
```

Serial console and account setup

From base install, setup for serial port, and setup accounts for passwordless ssh login and sudo:

```
kvm -m 2047 -hda /kvm/vms/vm-debian4-i386-serial.qcow2 -cdrom /kvm/debian-40r8-i386-netinst.iso -redir
'tcp:2241::22' -boot c -smp 2 -cpu qemu32,-nx -net nic,model=e1000 -net user

su
apt-get install sudo openssh-server
VISUAL=vi visudo

# Add at the end:t %sudo ALL=NOPASSWD: ALL

# Add account <USER> to group sudo
# Copy in public ssh key.

# Add in /etc/inittab:
S0:2345:respawn:/sbin/getty -L ttyS0 19200 vt100
```

Add to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line:

```
console=tty0 console=ttyS0,115200n8
```

Do these steps:

```
# Add user buildbot, with disabled password. Add as sudo, and add ssh key.
sudo /usr/sbin/adduser --disabled-password buildbot
sudo /usr/sbin/adduser buildbot sudo
sudo su - buildbot
mkdir .ssh
# Add all necessary keys.
cat >.ssh/authorized_keys
chmod -R go-rwx .ssh
```

VM for build

```
qemu-img create -b vm-debian4-i386-serial.qcow2 -f qcow2 vm-debian4-i386-build.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian4-i386-build.qcow2 -cdrom /kvm/debian-40r8-i386-netinst.iso -redir
'tcp:2241::22' -boot c -smp 2 -cpu qemu32,-nx -net nic,model=e1000 -net user -nographic

sudo apt-get build-dep mysql-server-5.0
# Some latex packages fail to install because they complain that the
# source is more than 5 years old! I solved by setting back the clock a
# couple of years temporarily ...
sudo apt-get install devscripts doxygen texlive-latex-base gs lsb-release fakeroot libevent-dev libssl-
dev zlib1g-dev libreadline5-dev
```

VM for install testing

```
qemu-img create -b vm-debian4-i386-serial.qcow2 -f qcow2 vm-debian4-i386-install.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian4-i386-install.qcow2 -cdrom /kvm/debian-40r8-i386-netinst.iso -redir
'tcp:2241::22' -boot c -smp 2 -cpu qemu32,-nx -net nic,model=e1000 -net user -nographic
```

See the General Principles article for how to make the '`my.seed`' file.

```
# No packages mostly!
sudo apt-get install debconf-utils

cat >>/etc/apt/sources.list <<END
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
END

sudo debconf-set-selections /tmp/my.seed
```

VM for upgrade testing

```
qemu-img create -b vm-debian4-i386-install.qcow2 -f qcow2 vm-debian4-i386-upgrade.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian4-i386-upgrade.qcow2 -cdrom /kvm/debian-40r8-i386-netinst.iso -redir
'tcp:2241::22' -boot c -smp 2 -cpu qemu64 -net nic,model=e1000 -net user -nographic

sudo apt-get install mysql-server-5.0
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Debian 5 amd64

Download netinst CD image debian-503-amd64-netinst.iso

```
cd /kvm/vms
qemu-img create -f qcow2 vm-debian5-amd64-serial.qcow2 8G
kvm -m 2047 -hda /kvm/vms/vm-debian5-amd64-serial.qcow2 -cdrom /kvm/debian-503-amd64-netinst.iso -redir
'tcp:2234::22' -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Serial console and account setup

From base install, setup for serial port, and setup accounts for passwordless ssh login and sudo:

```
kvm -m 2047 -hda /kvm/vms/vm-debian5-amd64-serial.qcow2 -redir 'tcp:2234::22' -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user

su
apt-get install sudo openssh-server
visudo
# uncomment %sudo ALL=NOPASSWD: ALL
# add user account to group sudo `addgroup <USER> sudo`.
# Copy in public ssh key.

# Add in /etc/inittab:
S0:2345:respawn:/sbin/getty -L ttyS0 19200 vt100
```

Add to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line (after removing `quiet splash'):

```
console=tty0 console=ttyS0,115200n8
```

Add user buildbot, with disabled password. Add as sudo, and add ssh key.

```
sudo adduser --disabled-password buildbot
sudo adduser buildbot sudo
sudo su - buildbot
mkdir .ssh
# Add all necessary keys.
cat >.ssh/authorized_keys
chmod -R go-rwx .ssh
```

VM for building .deb

```
qemu-img create -b vm-debian5-amd64-serial.qcow2 -f qcow2 vm-debian5-amd64-build.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian5-amd64-build.qcow2 -redir 'tcp:2234::22' -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user -nographic

sudo apt-get build-dep mysql-server-5.0
sudo apt-get install devscripts hardening-wrapper doxygen texlive-latex-base ghostscript libevent-dev
libssl-dev zlib1g-dev libreadline5-dev
```

VM for install testing

```
qemu-img create -b vm-debian5-amd64-serial.qcow2 -f qcow2 vm-debian5-amd64-install.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian5-amd64-install.qcow2 -redir 'tcp:2234::22' -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user -nographic

# No packages mostly!
sudo apt-get install debconf-utils

cat >>/etc/apt/sources.list <<END
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
END

sudo debconf-set-selections /tmp/my.seed
```

See the General Principles article for how to make the '`my.seed`' file.

VM for upgrade testing

```
qemu-img create -b vm-debian5-amd64-install.qcow2 -f qcow2 vm-debian5-amd64-upgrade.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian5-amd64-upgrade.qcow2 -redir 'tcp:2234::22' -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user -nographic

sudo apt-get install mysql-server-5.0
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Debian 5 i386

Base install

Download netinst CD image debian-503-i386-netinst.iso and install:

```
cd /kvm
qemu-img create -f qcow2 vms/vm-debian5-i386-base.qcow2 8G
kvm -m 2047 -hda /kvm/vms/vm-debian5-i386-base.qcow2 -cdrom /kvm/debian-503-i386-netinst.iso -redir
'tcp:2226::22' -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
```

Serial port console

From base install, setup for serial port, and setup accounts for passwordless ssh login and sudo:

```
qemu-img create -b vm-debian5-i386-base.qcow2 -f qcow2 vm-debian5-i386-serial.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian5-i386-serial.qcow2 -redir 'tcp:2226::22' -boot c -smp 2 -cpu qemu32,-
nx -net nic,model=virtio -net user -nographic
su
apt-get install sudo openssh-server emacs22-nox
visudo
# uncomment %sudo ALL=NOPASSWD: ALL
# add user account to group sudo.
# Copy in public ssh key.

# Add in /etc/inittab:
S0:2345:respawn:/sbin/agetty -h -L ttyS0 19200 vt100
```

Add to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line (after removing `quiet splash'):

```
console=tty0 console=ttyS0,115200n8
```

Add user buildbot, with disabled password. Add as sudo, and add ssh key.

```
sudo adduser --disabled-password buildbot
sudo adduser buildbot sudo
sudo su - buildbot
mkdir .ssh
# Paste all necessary keys.
cat >.ssh/authorized_keys
chmod -R go-rwx .ssh
```

Image for .deb build

```
qemu-img create -b vm-debian5-i386-serial.qcow2 -f qcow2 vm-debian5-i386-build.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian5-i386-build.qcow2 -redir 'tcp:2226::22' -boot c -smp 2 -cpu qemu32,-
nx -net nic,model=virtio -net user -nographic

sudo apt-get build-dep mysql-server-5.0
sudo apt-get install devscripts hardening-wrapper doxygen texlive-latex-base ghostscript libevent-dev
libssl-dev zlib1g-dev libreadline5-dev
```

Image for install testing

```
qemu-img create -b vm-debian5-i386-serial.qcow2 -f qcow2 vm-debian5-i386-install.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian5-i386-install.qcow2 -redir 'tcp:2226::22' -boot c -smp 2 -cpu
qemu32,-nx -net nic,model=virtio -net user -nographic

# No packages mostly!
sudo apt-get install debconf-utils

cat >>/etc/apt/sources.list <<END
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
END

sudo debconf-set-selections /tmp/my.seed
```

See the General Principles article how to obtain the `my.seed` file.

Image for upgrade testing

```
qemu-img create -b vm-debian5-i386-install.qcow2 -f qcow2 vm-debian5-i386-upgrade.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian5-i386-upgrade.qcow2 -redir 'tcp:2226::22' -boot c -smp 2 -cpu
qemu32,-nx -net nic,model=virtio -net user -nographic

sudo apt-get install mysql-server-5.0
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Debian 6 amd64

Create the VM:

```
cd /kvm/vms
qemu-img create -f qcow2 vm-debian6-amd64-serial.qcow2 8G
kvm -m 2047 -hda /kvm/vms/vm-debian6-amd64-serial.qcow2 -cdrom /kvm/debian-6a1-amd64-netinst.iso -redir
'tcp:2244::22' -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Serial console and account setup

From base install, setup for serial port, and setup accounts for passwordless ssh login and sudo:

```
kvm -m 2047 -hda /kvm/vms/vm-debian6-amd64-serial.qcow2 -redir 'tcp:2244::22' -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user

su
apt-get install sudo openssh-server
visudo

# Add: %sudo ALL=NOPASSWD: ALL
# add user account to group sudo `addgroup <USER> sudo`.
# Copy in public ssh key.

# Add in /etc/inittab:
S0:2345:respawn:/sbin/getty -L ttyS0 19200 vt100
```

Editing /etc/default/grub:

```
sudo vi /etc/default/grub
# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
```

Add user buildbot, with disabled password. Add as sudo, and add ssh key.

```
sudo adduser --disabled-password buildbot
sudo adduser buildbot sudo
sudo su - buildbot
mkdir .ssh
# Add all necessary keys.
cat >.ssh/authorized_keys
chmod -R go-rwx .ssh
```

VM for building .deb

```
qemu-img create -b vm-debian6-amd64-serial.qcow2 -f qcow2 vm-debian6-amd64-build.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian6-amd64-build.qcow2 -redir 'tcp:2244::22' -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user -nographic

sudo apt-get build-dep mysql-server-5.1
sudo apt-get install devscripts hardening-wrapper doxygen texlive-latex-base ghostscript libevent-dev
libssl-dev zlib1g-dev libreadline-dev
```

VM for install testing

```
qemu-img create -b vm-debian6-amd64-serial.qcow2 -f qcow2 vm-debian6-amd64-install.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian6-amd64-install.qcow2 -redir 'tcp:2244::22' -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user -nographic
```

See the General Principles article for how to make the '`my.seed`' file.

```
# No packages mostly!
sudo apt-get install debconf-utils

cat >>/etc/apt/sources.list <<END
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
END

sudo debconf-set-selections /tmp/my.seed
```

VM for upgrade testing

```
qemu-img create -b vm-debian6-amd64-install.qcow2 -f qcow2 vm-debian6-amd64-upgrade.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian6-amd64-upgrade.qcow2 -redir 'tcp:2244::22' -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user -nographic

sudo apt-get install mysql-server-5.1
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Debian 6 i386

Base Install

```
cd /kvm/vms
qemu-img create -f qcow2 vm-debian6-i386-serial.qcow2 8G
kvm -m 2047 -hda /kvm/vms/vm-debian6-i386-serial.qcow2 -cdrom /kvm/debian-6a1-i386-netinst.iso -redir
'tcp:2245::22' -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Serial console and account setup

From base install, setup for serial port, and setup accounts for passwordless ssh login and sudo:

```
kvm -m 2047 -hda /kvm/vms/vm-debian6-i386-serial.qcow2 -redir 'tcp:2245::22' -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user

su
apt-get install sudo openssh-server
visudo
# Add: %sudo ALL=NOPASSWD: ALL
# add user account to group sudo `addgroup <USER> sudo`.
# Copy in public ssh key.

# Add in /etc/inittab:
S0:2345:respawn:/sbin/getty -L ttyS0 19200 vt100
```

Editing /etc/default/grub:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
```

Add user buildbot, with disabled password. Add as sudo, and add ssh key.

```
sudo adduser --disabled-password buildbot
sudo adduser buildbot sudo
sudo su - buildbot
mkdir .ssh
# Add all necessary keys.
cat >.ssh/authorized_keys
chmod -R go-rwx .ssh
```

VM for building .deb

```
qemu-img create -b vm-debian6-i386-serial.qcow2 -f qcow2 vm-debian6-i386-build.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian6-i386-build.qcow2 -redir 'tcp:2245::22' -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user -nographic

sudo apt-get build-dep mysql-server-5.1
sudo apt-get install devscripts hardening-wrapper doxygen texlive-latex-base ghostscript libevent-dev
libssl-dev zlib1g-dev libreadline-dev
```

VM for install testing

```
qemu-img create -b vm-debian6-i386-serial.qcow2 -f qcow2 vm-debian6-i386-install.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian6-i386-install.qcow2 -redir 'tcp:2245::22' -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user -nographic
```

See above for how to obtain my.seed

```
# No packages mostly!
sudo apt-get install debconf-utils

cat >>/etc/apt/sources.list <<END
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
END

sudo debconf-set-selections /tmp/my.seed
```

VM for upgrade testing

```
qemu-img create -b vm-debian6-i386-install.qcow2 -f qcow2 vm-debian6-i386-upgrade.qcow2
kvm -m 2047 -hda /kvm/vms/vm-debian6-i386-upgrade.qcow2 -redir 'tcp:2245::22' -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user -nographic

sudo apt-get install mysql-server-5.1
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Red Hat

See Buildbot Setup for Virtual Machines - Centos 5 amd64.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - CentOS 5 i386.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - CentOS 6.2.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Fedora 16.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Fedora 17.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Fedora 18.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Fedora 19.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Fedora 20.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Red Hat 6 x86.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Centos 5 amd64

Base install

```
cd /kvm
wget http://ftp.klid.dk/ftp/centos/5.3/isos/x86_64/CentOS-5.3-x86_64-bin-DVD.iso
qemu-img create -f qcow2 vms/vm-centos5-amd64-serial.qcow2 8G
kvm -m 2047 -hda /kvm/vms/vm-centos5-amd64-serial.qcow2 -cdrom CentOS-5.3-x86_64-bin-DVD.iso -redir
'tcp:2237::22' -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Serial port and account setup

```
kvm -m 2047 -hda /kvm/vms/vm-centos5-amd64-serial.qcow2 -cdrom CentOS-5.3-i386-bin-DVD.iso -redir
'tcp:2237::22' -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
```

Add to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line (after removing `quiet splash'):

```
console=tty0 console=ttyS0,115200n8
```

Run these commands:

```
cat >>/etc/inittab <<END

# Serial console.
S0:2345:respawn:/sbin/agetty -h -L ttyS0 19200 vt100
END
```

```
useradd buildbot

# Password is disabled by default in Centos5.

usermod -a -G wheel buildbot
visudo

# Uncomment the line "%wheel        ALL=(ALL)       NOPASSWD: ALL"
# Comment out this line:
# Defaults    requiretty

# Put in public ssh key for own account and host buildbot account.
# Note that Centos5 seems to require .ssh/authorized_keys chmod go-rwx.

su - buildbot
mkdir .ssh
chmod go-rwx .ssh
cat >.ssh/authorized_keys
chmod go-rwx .ssh/authorized_keys
```

Image for rpm build

```
qemu-img create -b vm-centos5-amd64-serial.qcow2 -f qcow2 vm-centos5-amd64-build.qcow2
kvm -m 2047 -hda /kvm/vms/vm-centos5-amd64-build.qcow2 -cdrom /kvm/CentOS-5.3-x86_64-bin-DVD.iso -redir
'tcp:2237::22' -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
```

Install compilers etc:

```
sudo yum groupinstall "Development Tools"
sudo yum install gperf readline-devel ncurses-devel libaio-devel openssl-devel zlib-devel perl perl\
(DBI\)
```

Download 5.0 rpm for shared-compat:

```
sudo mkdir -p /srv/shared/yum/CentOS/5/x86_64/RPMS/
cd /srv/shared/yum/CentOS/5/x86_64/RPMS/
sudo wget http://mirror.ourdelta.org/yum/CentOS/5/x86_64/RPMS/MySQL-OurDelta-shared-5.0.87.d10-
65.el5.x86_64.rpm
```

Image for install/test

```
qemu-img create -b vm-centos5-amd64-serial.qcow2 -f qcow2 vm-centos5-amd64-install.qcow2
kvm -m 2047 -hda /kvm/vms/vm-centos5-amd64-install.qcow2 -cdrom /kvm/CentOS-5.3-x86_64-bin-DVD.iso -redir
'tcp:2237::22' -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
```

Install extra dependencies:

```
sudo yum install perl perl\(DBI\)
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - CentOS 5 i386

Base install

```
cd /kvm
qemu-img create -f qcow2 vms/vm-centos5-i386-base.qcow2 8G
# ISO (dvd) install:
kvm -m 2047 -hda /kvm/vms/vm-centos5-i386-base.qcow2 -cdrom CentOS-5.3-i386-bin-DVD.iso -redir
'tcp:2225::22' -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
```

Configure for serial console

```
(cd vms && qemu-img create -b vm-centos5-i386-base.qcow2 -f qcow2 vm-centos5-i386-serial.qcow2)
kvm -m 2047 -hda /kvm/vms/vm-centos5-i386-serial.qcow2 -cdrom CentOS-5.3-i386-bin-DVD.iso -redir
'tcp:2225::22' -boot c -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user -nographic
```

Add to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line (after removing `quiet splash'):

```
console=tty0 console=ttyS0,115200n8
```

Add login prompt on serial console:

```
cat >>/etc/inittab <<END

# Serial console.
S0:2345:respawn:/sbin/agetty -h -L ttyS0 19200 vt100
END
```

Create account.

```
useradd buildbot
# Password is disabled by default in Centos5.
usermod -a -G wheel buildbot
visudo
# Uncomment the line "%wheel        ALL=(ALL)       NOPASSWD: ALL"
# Comment out this line:
# Defaults    requiretty

# Put in public ssh key for own account and host buildbot account.
# Note that Centos5 seems to require .ssh/authorized_keys chmod go-rwx.
su - buildbot
mkdir .ssh
chmod go-rwx .ssh
cat >.ssh/authorized_keys
# Paste the id_dsa.pub key, see above.
chmod go-rwx .ssh/authorized_keys
```

Image for rpm build

```
qemu-img create -b vm-centos5-i386-serial.qcow2 -f qcow2 vm-centos5-i386-build.qcow2
kvm -m 2047 -hda /kvm/vms/vm-centos5-i386-build.qcow2 -cdrom /kvm/CentOS-5.3-i386-bin-DVD.iso -redir
'tcp:2225::22' -boot c -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user -nographic
```

Install compilers etc:

```
sudo yum groupinstall "Development Tools"
sudo yum install gperf readline-devel ncurses-devel zlib-devel libaio-devel openssl-devel perl perl\
(DBI\)
```

Download 5.0 rpm for shared-compat:

```
sudo mkdir -p /srv/shared/yum/CentOS/5/i386/RPMS/
cd /srv/shared/yum/CentOS/5/i386/RPMS/
sudo wget http://mirror.ourdelta.org/yum/CentOS/5/i386/RPMS/MySQL-OurDelta-shared-5.0.87.d10-
65.el5.i386.rpm
```

Image for install/test

```
qemu-img create -b vm-centos5-i386-serial.qcow2 -f qcow2 vm-centos5-i386-install.qcow2
kvm -m 2047 -hda /kvm/vms/vm-centos5-i386-install.qcow2 -cdrom /kvm/CentOS-5.3-i386-bin-DVD.iso -redir
'tcp:2225::22' -boot c -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user -nographic
```

Install extra dependencies:

```
sudo yum install perl perl\(DBI\)
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - CentOS 6.2

Base install

```
qemu-img create -f qcow2 /kvm/vms/vm-centos6-amd64-serial.qcow2 8G
qemu-img create -f qcow2 /kvm/vms/vm-centos6-i386-serial.qcow2 8G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 1024 -hda /kvm/vms/vm-centos6-amd64-serial.qcow2 -cdrom /kvm/iso/centos/CentOS-6.2-x86_64-bin-
DVD1.iso -redir tcp:22255::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
kvm -m 1024 -hda /kvm/vms/vm-centos6-i386-serial.qcow2 -cdrom /kvm/iso/centos/CentOS-6.2-i386-bin-
DVD1.iso -redir tcp:22256::22 -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote-host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. The CentOS installer has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- The Installer will throw up a “Storage Device Warning”, choose “Yes, discard any data”
- Set the hostname to centos6-amd64 (or centos6-i386)
- Click the “Configure Network” button on the Hostname screen.
  - Edit System eth0 to “connect automatically”
  - Apply and then close the “Network Connections” window
- Set Timezone to Europe/Helsinki (keep “System clock uses UTC” checked)
- When partitioning disks, choose “Use All Space”
  - do not check the “Encrypt system” checkbox
  - do check the “Review and modify partitioning layout” checkbox
  - Delete the LVM stuff and leaving the sda1 partition alone, repartition the physical volume as follows

Device	Size(MB)	Mount Point	Type	Format
sda2	5672	/	ext4	yes
sda3	(max allowable)	(n/a)	swap	yes

- Minimal install
- Customize Later

When the install is finished, you will be prompted to reboot. Go ahead and do so, but it will fail. Kill the VM (after the reboot fails) and start it up again:

```
kvm -m 1024 -hda /kvm/vms/vm-centos6-amd64-serial.qcow2 -redir tcp:22255::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user

kvm -m 1024 -hda /kvm/vms/vm-centos6-i386-serial.qcow2 -redir tcp:22256::22 -boot c -smp 2 -cpu qemu32,-
nx -net nic,model=virtio -net user
```

You may connect via VNC as before, but ssh is probably preferred. Login as root.

Now that the VM is installed, it's time to configure it.

```
ssh -p 22255 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
root@localhost
ssh -p 22256 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
root@localhost
```

After logging in as root, create a local user

```
adduser ${username}
usermod -a -G wheel ${username}
passwd ${username}
```

Enable password-less sudo and serial console:

```
visudo
# Uncomment the line "%wheel        ALL=(ALL)       NOPASSWD: ALL"
# Comment out this line:
# Defaults    requiretty
```

Still logged in as root, add to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line (after removing 'quiet'):

```
console=tty0 console=ttyS0,115200n8
```

Add login prompt on serial console:

```
cat >>/etc/inittab <<END

# Serial console.
S0:2345:respawn:/sbin/agetty -h -L ttyS0 19200 vt100
END
```

Logout as root, and then, from the VM host server:

Install proper ssh:

```
ssh -t -p 22255 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "sudo yum install openssh-server openssh-clients"
ssh -t -p 22256 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "sudo yum install openssh-server openssh-clients"
```

Create a .ssh folder:

```
ssh -t -p 22255 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
ssh -t -p 22256 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
```

Copy over the authorized keys file:

```
scp -P 22255 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
scp -P 22256 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
```

Set permissions on the .ssh folder correctly:

```
ssh -t -p 22255 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
ssh -t -p 22256 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
```

Create the buildbot user:

```
ssh -p 22255 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
ssh -p 22256 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
```

Upload the ttyS0.conf file and put it where it goes:

```
scp -P 22255 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0.conf
buildbot@localhost:
scp -P 22256 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0.conf
buildbot@localhost:

ssh -p 22255 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo cp -vi
ttyS0.conf /etc/init/; rm -v ttyS0.conf;'
ssh -p 22256 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo cp -vi
ttyS0.conf /etc/init/; rm -v ttyS0.conf;'
```

Update the VM:

```
ssh -p 22255 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
ssh -p 22256 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
```

Once logged in:

```
sudo yum update
```

After updating, shut down the VM:

```
sudo shutdown -h now
```

VMs for building .rpms

```
for i in '/kvm/vms/vm-centos6-amd64-serial.qcow2 22255 qemu64' '/kvm/vms/vm-centos6-i386-serial.qcow2
22256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "sudo yum -y groupinstall 'Development Tools'" \
    "sudo yum -y install wget tree gperf readline-devel ncurses-devel zlib-devel pam-devel libaio-devel
openssl-devel libxml2 libxml2-devel unixODBC-devel bzr perl perl\(DBI\)" \
    "sudo yum -y remove systemtap-sdt-dev" \
    "bzr co --lightweight lp:mariadb-native-client" \
    "sudo mkdir -vp /usr/src/redhat/SOURCES /usr/src/redhat/SPECS /usr/src/redhat/RPMS
/usr/src/redhat/SRPMS" \
    "wget http://www.cmake.org/files/v2.8/cmake-2.8.8.tar.gz;tar -zxvf cmake-2.8.8.tar.gz;cd cmake-
2.8.8;./configure;make;sudo make install"; \
done
```

Also:

- Installing the Boost library needed for the OQGraph storage engine

VMs for install testing.

`MariaDB.local.repo` points at a local directory on the VM. `MariaDB.repo` points at the real MariaDB YUM repository.

```
for i in '/kvm/vms/vm-centos6-amd64-serial.qcow2 22255 qemu64' '/kvm/vms/vm-centos6-i386-serial.qcow2
22256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo yum -y update" \
    "sudo yum -y install patch libaio perl perl-Time-HiRes perl-DBI libtool-ltdl unixODBC" \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    "sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/"; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-centos6-amd64-serial.qcow2 22255 qemu64' '/kvm/vms/vm-centos6-i386-serial.qcow2
22256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade/')" \
    "sudo yum -y update" \
    'sudo yum -y install mysql-server libtool-ltdl unixODBC' \
    'sudo /etc/init.d/mysqld start' \
    'mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into
t values (1); select * from t"' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    "sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/"; \
done
```

The MariaDB upgrade testing VMs were not built. There is currently an error with installing MariaDB from the YUM repo.

VMs for MariaDB upgrade testing

```
for i in '/kvm/vms/vm-centos6-amd64-serial.qcow2 22255 qemu64' '/kvm/vms/vm-centos6-i386-serial.qcow2
22256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    'sudo yum -y update' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.repo
buildbot@localhost:/tmp/" \
    'sudo rpm --verbose --import http://downloads.mariadb.org/repo/RPM-GPG-KEY-MariaDB' \
    'sudo mv -vi /tmp/MariaDB.repo /etc/yum.repos.d/' \
    'sudo yum -y remove mysql-libs' \
    'sudo yum -y install MariaDB-server MariaDB-client MariaDB-test' \
    'sudo yum -y install cronie cronie-anacron crontabs.noarch postfix' \
    'sudo /etc/init.d/mysqld start' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    'sudo rm -v /etc/yum.repos.d/MariaDB.repo' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    'sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/'; \
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Fedora 16

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-fedora16-i386-serial.qcow2 10G
qemu-img create -f qcow2 /kvm/vms/vm-fedora16-amd64-serial.qcow2 10G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 1024 -hda /kvm/vms/vm-fedora16-i386-serial.qcow2 -cdrom /kvm/iso/fedora/Fedora-16-i386-DVD.iso -
redir tcp:2263::22 -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
kvm -m 1024 -hda /kvm/vms/vm-fedora16-amd64-serial.qcow2 -cdrom /kvm/iso/fedora/Fedora-16-x86_64-DVD.iso
-redir tcp:2264::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote-host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. The Fedora installer has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- The Installer will throw up a “Storage Device Warning”, choose “Yes, discard any data”
- Set the hostname to fedora16-amd64 (or fedora16-i386)
- Click the “Configure Network” button on the Hostname screen.
  - Edit System eth0 to “connect automatically”
  - Apply and then close the “Network Connections” window
- When partitioning disks, choose “Use All Space”
  - uncheck the “Use LVM” checkbox
  - do not check the “Encrypt system” checkbox

- Minimal install
- Customize Later

When the install is finished, you will be prompted to reboot. Go ahead and do so, but it will fail. Kill the VM (after the reboot fails) and start it up again:

```
kvm -m 1024 -hda /kvm/vms/vm-fedora16-i386-serial.qcow2 -redir tcp:2263::22 -boot c -smp 2 -cpu qemu32,-
nx -net nic,model=virtio -net user
kvm -m 1024 -hda /kvm/vms/vm-fedora16-amd64-serial.qcow2 -redir tcp:2264::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user
```

Until the extra user is installed you must connect via VNC as before. SSH is preferred, so that's what we'll do first. Login as root.

```
ssh -p 2263 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
root@localhost
ssh -p 2264 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
root@localhost
```

After logging in as root, install proper ssh and then create a local user:

```
/sbin/chkconfig --level 35 network on
ifup eth0
yum install openssh-server openssh-clients
adduser ${username}
usermod -a -G wheel ${username}
passwd ${username}
```

Enable password-less sudo and serial console:

```
visudo
# Uncomment the line "%wheel        ALL=(ALL)       NOPASSWD: ALL"
# Comment out this line:
# Defaults    requiretty
```

Still logged in as root, add to /boot/grub/menu.lst:

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

grub2-mkconfig -o /boot/grub2/grub.cfg
```

Logout as root, and then, from the VM host server:

Create a .ssh folder:

```
ssh -t -p 2263 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
ssh -t -p 2264 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
```

Copy over the authorized keys file:

```
scp -P 2263 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
scp -P 2264 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
```

Set permissions on the .ssh folder correctly:

```
ssh -t -p 2263 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
ssh -t -p 2264 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
```

Create the buildbot user:

```
ssh -p 2263 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
ssh -p 2264 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
```

su to the local buildbot user and ssh to the vm to put the key in known_hosts:

For i386:

```
sudo su - buildbot
ssh -p 2263 buildbot@localhost
# exit, then exit again
```

For amd64:

```
sudo su - buildbot
ssh -p 2264 buildbot@localhost
# exit, then exit again
```

Upload the ttyS0 file and put it where it goes:

```
scp -P 2263 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:
scp -P 2264 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:

ssh -p 2263 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mv -vi
ttyS0 /etc/event.d/;'
ssh -p 2264 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mv -vi
ttyS0 /etc/event.d/;'
```

Update the VM:

```
ssh -p 2263 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
ssh -p 2264 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
```

Once logged in:

```
sudo yum update
```

After updating, shut down the VM:

```
sudo shutdown -h now
```

VMs for building .rpms

```
for i in '/kvm/vms/vm-fedora16-amd64-serial.qcow2 2264 qemu64' '/kvm/vms/vm-fedora16-i386-serial.qcow2
2263 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "sudo yum -y groupinstall 'Development Tools'" \
    "sudo yum -y install wget tree gperf readline-devel ncurses-devel zlib-devel pam-devel libaio-devel
openssl-devel perl perl\(DBI\)" \
    "sudo yum -y remove systemtap-sdt-dev" \
    "sudo mkdir -vp /usr/src/redhat/SOURCES /usr/src/redhat/SPECS /usr/src/redhat/RPMS
/usr/src/redhat/SRPMS" \
    "wget http://www.cmake.org/files/v2.8/cmake-2.8.8.tar.gz;tar -zxvf cmake-2.8.8.tar.gz;cd cmake-
2.8.8;./configure;make;sudo make install"; \
done
```

Also:

- Installing the Boost library needed for the OQGraph storage engine

VMs for install testing.

`MariaDB.local.repo` points at a local directory on the VM. `MariaDB.repo` points at the real MariaDB YUM repository.

```
for i in '/kvm/vms/vm-fedora16-amd64-serial.qcow2 2264 qemu64' '/kvm/vms/vm-fedora16-i386-serial.qcow2
2263 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo yum -y update" \
    "sudo yum -y install libaio perl perl-Time-HiRes perl-DBI" \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    "sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/"; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-fedora16-amd64-serial.qcow2 2264 qemu64' '/kvm/vms/vm-fedora16-i386-serial.qcow2
2263 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade/')" \
    "sudo yum -y update" \
    'sudo yum -y install mysql-server' \
    'sudo systemctl enable mysqld.service' \
    'sudo systemctl start mysqld.service' \
    'mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into
t values (1); select * from t"' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    "sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/"; \
done
```

The MariaDB upgrade testing VMs were not built. Once we have MariaDB Fedora 16 RPMs, then I will attempt building this VM. For now, the placeholder text below is copied from the Buildbot Setup for Virtual Machines - CentOS 6.2 page.

VMs for MariaDB upgrade testing

```
for i in '/kvm/vms/vm-fedora16-amd64-serial.qcow2 2264 qemu64' '/kvm/vms/vm-fedora16-i386-serial.qcow2
2263 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    'sudo yum -y update' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.repo
buildbot@localhost:/tmp/" \
    'sudo rpm --verbose --import http://downloads.mariadb.org/repo/RPM-GPG-KEY-MariaDB' \
    'sudo mv -vi /tmp/MariaDB.repo /etc/yum.repos.d/' \
    'sudo yum -y remove mysql-libs' \
    'sudo yum -y install MariaDB-server MariaDB-client MariaDB-test' \
    'sudo yum -y install cronie cronie-anacron crontabs.noarch postfix' \
    'sudo /etc/init.d/mysqld start' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    'sudo rm -v /etc/yum.repos.d/MariaDB.repo' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    'sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/'; \
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Fedora 17

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-fedora17-i386-serial.qcow2 10G
qemu-img create -f qcow2 /kvm/vms/vm-fedora17-amd64-serial.qcow2 10G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 1024 -hda /kvm/vms/vm-fedora17-i386-serial.qcow2 -cdrom /kvm/iso/fedora/Fedora-17-i386-DVD.iso -
redir tcp:2265::22 -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
kvm -m 1024 -hda /kvm/vms/vm-fedora17-amd64-serial.qcow2 -cdrom /kvm/iso/fedora/Fedora-17-x86_64-DVD.iso
-redir tcp:2266::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote-host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. The Fedora installer has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- The Installer will throw up a “Storage Device Warning”, choose “Yes, discard any data”
- Set the hostname to fedora17-amd64 (or fedora17-i386)
- Click the “Configure Network” button on the Hostname screen.
  - Edit System eth0 to “connect automatically”
  - Apply and then close the “Network Connections” window
- When partitioning disks, choose “Use All Space”
  - uncheck the “Use LVM” checkbox
  - do not check the “Encrypt system” checkbox

- Minimal install
- Customize Later

When the install is finished, you will be prompted to reboot. Go ahead and do so, but it will fail. Kill the VM (after the reboot fails) and start it up again:

```
kvm -m 1024 -hda /kvm/vms/vm-fedora17-i386-serial.qcow2 -redir tcp:2265::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user
kvm -m 1024 -hda /kvm/vms/vm-fedora17-amd64-serial.qcow2 -redir tcp:2266::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user
```

Until the extra user is installed you must connect via VNC as before. SSH is preferred, so that's what we'll do first. Login as root.

```
ssh -p 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
root@localhost
ssh -p 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
root@localhost
```

After logging in as root, install proper ssh and then create a local user:

```
/sbin/chkconfig --level 35 network on
ifup eth0
yum install openssh-server openssh-clients
adduser ${username}
usermod -a -G wheel ${username}
passwd ${username}
```

Enable password-less sudo and serial console:

```
visudo
# Uncomment the line "%wheel        ALL=(ALL)       NOPASSWD: ALL"
# Comment out this line:
# Defaults    requiretty
```

Still logged in as root, add to /boot/grub/menu.lst:

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

grub2-mkconfig -o /boot/grub2/grub.cfg
```

Logout as root, and then, from the VM host server:

Create a .ssh folder:

```
ssh -t -p 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
ssh -t -p 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
```

Copy over the authorized keys file:

```
scp -P 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
scp -P 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
```

Set permissions on the .ssh folder correctly:

```
ssh -t -p 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
ssh -t -p 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
```

Create the buildbot user:

```
ssh -p 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
ssh -p 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
```

su to the local buildbot user and ssh to the vm to put the key in known_hosts:

For i386:

```
sudo su - buildbot
ssh -p 2265 buildbot@localhost
# exit, then exit again
```

For amd64:

```
sudo su - buildbot
ssh -p 2266 buildbot@localhost
# exit, then exit again
```

Upload the ttyS0 file and put it where it goes:

```
scp -P 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:
scp -P 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:

ssh -p 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mkdir -v
/etc/event.d;sudo mv -vi ttyS0 /etc/event.d/;'
ssh -p 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mkdir -v
/etc/event.d;sudo mv -vi ttyS0 /etc/event.d/;'
```

Update the VM:

```
ssh -p 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
ssh -p 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
```

Once logged in:

```
sudo yum update
```

After updating, shut down the VM:

```
sudo shutdown -h now
```

VMs for building .rpms

On the VM host server:

```
qemu-img create -b /kvm/vms/vm-fedora17-i386-serial.qcow2 -f qcow2 /kvm/vms/vm-fedora17-i386-build.qcow2
qemu-img create -b /kvm/vms/vm-fedora17-amd64-serial.qcow2 -f qcow2 /kvm/vms/vm-fedora17-amd64-
build.qcow2
```

Boot the VMs with:

```
kvm -m 1024 -hda /kvm/vms/vm-fedora17-i386-build.qcow2 -redir tcp:2265::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user -nographic
kvm -m 1024 -hda /kvm/vms/vm-fedora17-amd64-build.qcow2 -redir tcp:2266::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user -nographic
```

Copy over the boost tar file:

```
scp -P 2265 -i /kvm/vms/ssh-keys/id_dsa -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null
/kvm/boost_1_49_0.tar.gz buildbot@localhost:/dev/shm/
scp -P 2266 -i /kvm/vms/ssh-keys/id_dsa -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null
/kvm/boost_1_49_0.tar.gz buildbot@localhost:/dev/shm/
```

Login to the VMs:

```
ssh -p 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
ssh -p 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
```

Once logged in:

```
sudo yum -y groupinstall 'Development Tools'
sudo yum -y install tar wget tree gperf readline-devel ncurses-devel zlib-devel pam-devel libaio-devel
openssl-devel perl perl\(DBI\)
sudo yum -y remove systemtap-sdt-dev
sudo mkdir -vp /usr/src/redhat/SOURCES /usr/src/redhat/SPECS /usr/src/redhat/RPMS /usr/src/redhat/SRPMS

# cmake
wget http://www.cmake.org/files/v2.8/cmake-2.8.8.tar.gz;tar -zxvf cmake-2.8.8.tar.gz;cd cmake-
2.8.8;./configure;make;sudo make install

# boost
cd /usr/local/src;sudo tar zxf /dev/shm/boost_1_49_0.tar.gz;cd /usr/local/include/;sudo ln -s
../src/boost_1_49_0/boost .

# shutdown the VM
sudo shutdown -h now
```

VMs for install testing.

On the VM host server:

```
qemu-img create -b /kvm/vms/vm-fedora17-i386-serial.qcow2 -f qcow2 /kvm/vms/vm-fedora17-i386-
install.qcow2
qemu-img create -b /kvm/vms/vm-fedora17-amd64-serial.qcow2 -f qcow2 /kvm/vms/vm-fedora17-amd64-
install.qcow2
```

Boot the VMs with:

```
kvm -m 1024 -hda /kvm/vms/vm-fedora17-i386-install.qcow2 -redir tcp:2265::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user -nographic
kvm -m 1024 -hda /kvm/vms/vm-fedora17-amd64-install.qcow2 -redir tcp:2266::22 -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user -nographic
```

Login to the VMs:

```
ssh -p 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
ssh -p 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
```

Once logged in:

```
sudo yum -y update
sudo yum -y install patch tar libaio perl perl-Time-HiRes perl-DBI

# shutdown the VM
sudo shutdown -h now
```

VMs for MySQL upgrade testing

On the VM host server:

```
qemu-img create -b /kvm/vms/vm-fedora17-i386-serial.qcow2 -f qcow2 /kvm/vms/vm-fedora17-i386-
upgrade.qcow2
qemu-img create -b /kvm/vms/vm-fedora17-amd64-serial.qcow2 -f qcow2 /kvm/vms/vm-fedora17-amd64-
upgrade.qcow2
```

Boot the VMs with:

```
kvm -m 1024 -hda /kvm/vms/vm-fedora17-i386-upgrade.qcow2 -redir tcp:2265::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user -nographic
kvm -m 1024 -hda /kvm/vms/vm-fedora17-amd64-upgrade.qcow2 -redir tcp:2266::22 -boot c -smp 2 -cpu qemu64
-net nic,model=virtio -net user -nographic
```

Login to the VMs:

```
ssh -p 2265 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
ssh -p 2266 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
```

Once logged in:

```
sudo yum -y update
sudo yum -y install tar mysql-server
sudo systemctl enable mysqld.service
sudo systemctl start mysqld.service
mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into t
values (1); select * from t"

# shutdown the VM
sudo shutdown -h now
```

The MariaDB upgrade testing VMs were not built. Once we have MariaDB Fedora 17 RPMs, then I will attempt building this VM. For now, the placeholder text below is copied from the Buildbot Setup for Virtual Machines - CentOS 6.2 page.

VMs for MariaDB upgrade testing

```
for i in '/kvm/vms/vm-fedora17-amd64-serial.qcow2 2266 qemu64' '/kvm/vms/vm-fedora17-i386-serial.qcow2
2265 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    'sudo yum -y update' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.repo
buildbot@localhost:/tmp/" \
    'sudo rpm --verbose --import http://downloads.mariadb.org/repo/RPM-GPG-KEY-MariaDB' \
    'sudo mv -vi /tmp/MariaDB.repo /etc/yum.repos.d/' \
    'sudo yum -y remove mysql-libs' \
    'sudo yum -y install MariaDB-server MariaDB-client MariaDB-test' \
    'sudo yum -y install cronie cronie-anacron crontabs.noarch postfix' \
    'sudo /etc/init.d/mysqld start' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    'sudo rm -v /etc/yum.repos.d/MariaDB.repo' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    'sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/'; \
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Fedora 18

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-fedora18-i386-serial.qcow2 10G
qemu-img create -f qcow2 /kvm/vms/vm-fedora18-amd64-serial.qcow2 10G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 2048 -hda /kvm/vms/vm-fedora18-i386-serial.qcow2 -cdrom /kvm/iso/fedora/Fedora-18-i386-DVD.iso -
redir tcp:2277::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
kvm -m 2048 -hda /kvm/vms/vm-fedora18-amd64-serial.qcow2 -cdrom /kvm/iso/fedora/Fedora-18-x86_64-DVD.iso
-redir tcp:2278::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote-host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. The Fedora installer has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- Under “Network Configuration” set the hostnames to fedora18-amd64 and fedora18-i386
- Change “Software Selection” to “Minimal Install” (default is “Gnome Desktop”)
- For “Installation Destination” select the disk then click continue.
- do not check the encryption checkbox
- On “Installation Options” screen, expand the “Partition scheme configuration” box and select a “Partition type” of “Standard Partition”
- While installing, set the root password

When the install is finished, you will be prompted to reboot. Go ahead and do so, but it will fail. Kill the VM (after the reboot fails) and start it up again:

```
kvm -m 2048 -hda /kvm/vms/vm-fedora18-i386-serial.qcow2 -redir tcp:2277::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user
kvm -m 2048 -hda /kvm/vms/vm-fedora18-amd64-serial.qcow2 -redir tcp:2278::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user
```

Until a non-root user is installed you must connect as root. SSH is preferred, so that's what we'll do first. Login as root.

```
ssh -p 2277 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
root@localhost
ssh -p 2278 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
root@localhost
```

After logging in as root, install proper ssh and then create a local user:

```
adduser ${username}
usermod -a -G wheel ${username}
passwd ${username}
```

Enable password-less sudo and serial console:

```
visudo
# Uncomment the line "%wheel        ALL=(ALL)       NOPASSWD: ALL"
# Comment out this line:
# Defaults    requiretty
```

Still logged in as root, add to /boot/grub/menu.lst:

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

grub2-mkconfig -o /boot/grub2/grub.cfg
```

Logout as root, and then, from the VM host server:

Create a .ssh folder:

```
ssh -t -p 2277 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
ssh -t -p 2278 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
```

Copy over the authorized keys file:

```
scp -P 2277 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
scp -P 2278 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
```

Set permissions on the .ssh folder correctly:

```
ssh -t -p 2277 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
ssh -t -p 2278 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
```

Create the buildbot user:

```
ssh -p 2277 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
ssh -p 2278 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
```

su to the local buildbot user and ssh to the vm to put the key in known_hosts:

For i386:

```
sudo su - buildbot
ssh -p 2277 buildbot@localhost
# exit, then exit again
```

For amd64:

```
sudo su - buildbot
ssh -p 2278 buildbot@localhost
# exit, then exit again
```

Upload the ttyS0 file and put it where it goes:

```
scp -P 2277 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:
scp -P 2278 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:

ssh -p 2277 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mkdir -v
/etc/event.d;sudo mv -vi ttyS0 /etc/event.d/;'
ssh -p 2278 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mkdir -v
/etc/event.d;sudo mv -vi ttyS0 /etc/event.d/;'
```

Update the VM:

```
ssh -p 2277 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
ssh -p 2278 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
```

Once logged in:

```
sudo yum update
```

After updating, shut down the VM:

```
sudo shutdown -h now
```

VMs for building .rpms

```
for i in '/kvm/vms/vm-fedora18-i386-serial.qcow2 2277 qemu64' '/kvm/vms/vm-fedora18-amd64-serial.qcow2
2278 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
  "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/boost_1_49_0.tar.gz
buildbot@localhost:/dev/shm/" \
  "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/thrift-0.9.0.tar.gz
buildbot@localhost:/dev/shm/" \
  "sudo yum -y groupinstall 'Development Tools'" \
  "sudo yum -y install cmake tar wget tree gperf readline-devel ncurses-devel zlib-devel pam-devel
libaio-devel openssl-devel libxml2 libxml2-devel unixODBC-devel bzr perl perl\(DBI\)" \
  "sudo yum -y remove systemtap-sdt-dev" \
  "bzr co --lightweight lp:mariadb-native-client" \
  "sudo mkdir -vp /usr/src/redhat/SOURCES /usr/src/redhat/SPECS /usr/src/redhat/RPMS
/usr/src/redhat/SRPMS" \
  "cd /usr/local/src;sudo tar zxf /dev/shm/thrift-0.9.0.tar.gz;pwd;ls" \
  "cd /usr/local/src/thrift-0.9.0;echo;pwd;sudo ./configure --prefix=/usr --enable-shared=no --enable-
static=yes CXXFLAGS=-fPIC CFLAGS=-fPIC && echo && echo 'now making' && echo && sleep 3 && sudo make &&
echo && echo 'now installing' && echo && sleep 3 && sudo make install" \
  "cd /usr/local/src;sudo tar zxf /dev/shm/boost_1_49_0.tar.gz;cd /usr/local/include/;sudo ln -vs
../src/boost_1_49_0/boost ." ; \
done
```

VMs for install testing.

```
for i in '/kvm/vms/vm-fedora18-i386-serial.qcow2 2277 qemu64' '/kvm/vms/vm-fedora18-amd64-serial.qcow2
2278 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo yum -y update" \
    "sudo yum -y install patch tar libaio perl perl-Time-HiRes perl-DBI" ; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-fedora18-i386-serial.qcow2 2277 qemu64' '/kvm/vms/vm-fedora18-amd64-serial.qcow2
2278 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade/')" \
    "sudo yum -y update" \
    "sudo yum -y install patch tar mysql-server libtool-ltdl unixODBC" \
    "sudo systemctl enable mysqld.service" \
    "sudo systemctl start mysqld.service" \
    'mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into
t values (1); select * from t"' ; \
done
```

VMs for MariaDB upgrade testing

```
for i in '/kvm/vms/vm-fedora18-amd64-serial.qcow2 2278 qemu64' '/kvm/vms/vm-fedora18-i386-serial.qcow2
2277 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    'sudo yum -y update' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB-${2}.repo
buildbot@localhost:/tmp/MariaDB.repo" \
    'sudo rpm --verbose --import https://yum.mariadb.org/RPM-GPG-KEY-MariaDB' \
    'sudo mv -vi /tmp/MariaDB.repo /etc/yum.repos.d/' \
    'sudo yum -y remove mysql-libs' \
    'sudo yum -y install MariaDB-server MariaDB-client MariaDB-test libtool-ltdl unixODBC' \
    'sudo yum -y install cronie cronie-anacron crontabs.noarch postfix patch tar' \
    'sudo /etc/init.d/mysqld start' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    'sudo rm -v /etc/yum.repos.d/MariaDB.repo' \
    'sudo yum -y update' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    'sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/'; \
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Fedora 19

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-fedora19-i386-serial.qcow2 20G
qemu-img create -f qcow2 /kvm/vms/vm-fedora19-amd64-serial.qcow2 20G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 2048 -hda /kvm/vms/vm-fedora19-i386-serial.qcow2 -cdrom /ds413/iso/fedora/Fedora-19-i386-DVD.iso -
redir tcp:2289::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
kvm -m 2048 -hda /kvm/vms/vm-fedora19-amd64-serial.qcow2 -cdrom /ds413/iso/fedora/Fedora-19-x86_64-
DVD.iso -redir tcp:2290::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote-host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. The Fedora installer has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- Under “Network Configuration” set the hostnames to fedora19-amd64 and fedora19-i386
- Change “Software Selection” to “Minimal Install” (default is “Gnome Desktop”)
- For “Installation Destination” select the disk then click continue.
- do not check the encryption checkbox
- On “Installation Options” screen, expand the “Partition scheme configuration” box and select a “Partition type” of “Standard Partition”
- While installing, set the root password and an initial user

When the install is finished, you will be prompted to reboot. Go ahead and do so, but it will fail. Kill the VM (after the reboot fails) and start it up again:

```
kvm -m 2048 -hda /kvm/vms/vm-fedora19-i386-serial.qcow2 -redir tcp:2289::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user
kvm -m 2048 -hda /kvm/vms/vm-fedora19-amd64-serial.qcow2 -redir tcp:2290::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user
```

Log in using the initial user created during the install:

```
ssh -p 2289 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost
ssh -p 2290 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost
```

After logging in, add the initial user to the “wheel” group.

```
usermod -a -G wheel ${username}
```

Enable password-less sudo for the “wheel” group and serial console sudo:

```
visudo
# Uncomment the line "%wheel        ALL=(ALL)       NOPASSWD: ALL"
# Comment out this line:
# Defaults    requiretty
```

Edit /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

grub2-mkconfig -o /boot/grub2/grub.cfg
```

Logout as root, and then, from the VM host server:

Create a .ssh folder:

```
ssh -t -p 2289 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
ssh -t -p 2290 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
```

Copy over the authorized keys file:

```
scp -P 2289 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
scp -P 2290 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
```

Set permissions on the .ssh folder correctly:

```
ssh -t -p 2289 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
ssh -t -p 2290 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -R go-rwx .ssh"
```

Create the buildbot user:

```
ssh -p 2289 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
ssh -p 2290 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -R go-rwx .ssh;
sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
```

su to the local buildbot user and ssh to the vm to put the key in known_hosts:

For i386:

```
sudo su - buildbot
ssh -p 2289 buildbot@localhost
# exit, then exit again
```

For amd64:

```
sudo su - buildbot
ssh -p 2290 buildbot@localhost
# exit, then exit again
```

Upload the ttyS0 file and put it where it goes:

```
scp -P 2289 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:
scp -P 2290 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:

ssh -p 2289 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mkdir -v
/etc/event.d;sudo mv -vi ttyS0 /etc/event.d/;'
ssh -p 2290 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mkdir -v
/etc/event.d;sudo mv -vi ttyS0 /etc/event.d/;'
```

Update the VM:

```
ssh -p 2289 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
ssh -p 2290 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
```

Once logged in:

```
sudo yum update
```

Change selinux policy to `permissive`:

```
sudo vi /etc/selinux/config
# change:
SELINUX=enforcing
# to:
SELINUX=permissive
```

After updating, shut down the VM:

```
sudo shutdown -h now
```

VMs for building .rpms

```
for i in '/kvm/vms/vm-fedora19-i386-serial.qcow2 2289 qemu64' '/kvm/vms/vm-fedora19-amd64-serial.qcow2
2290 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
  "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/thrift-0.9.0.tar.gz
buildbot@localhost:/dev/shm/" \
  "sudo yum -y groupinstall 'Development Tools'" \
  "sudo yum -y install yum-utils" \
  "sudo yum-builddep -y mariadb" \
  "sudo yum -y install automake libtool flex pkgconfig gcc-c++ libevent-devel python-devel ruby-devel
rpm-build" \
  "sudo yum -y install cmake tar wget tree gperf readline-devel ncurses-devel zlib-devel pam-devel
libaio-devel openssl-devel" \
  "sudo yum -y install libxml2 libxml2-devel bison bison-devel boost-devel unixODBC-devel bzr perl perl\
(DBI\)" \
  "sudo yum -y remove systemtap-sdt-dev" \
  "bzr co --lightweight lp:mariadb-native-client" \
  "sudo mkdir -vp /usr/src/redhat/SOURCES /usr/src/redhat/SPECS /usr/src/redhat/RPMS
/usr/src/redhat/SRPMS" \
  "cd /usr/local/src;sudo tar zxf /dev/shm/thrift-0.9.0.tar.gz;pwd;ls" \
  "cd /usr/local/src/thrift-0.9.0;echo;pwd;sudo ./configure --prefix=/usr --enable-shared=no --enable-
static=yes CXXFLAGS=-fPIC CFLAGS=-fPIC && echo && echo 'now making' && echo && sleep 3 && sudo make &&
echo && echo 'now installing' && echo && sleep 3 && sudo make install" ; \
done
```

VMs for install testing.

```
for i in '/kvm/vms/vm-fedora19-i386-serial.qcow2 2289 qemu64' '/kvm/vms/vm-fedora19-amd64-serial.qcow2
2290 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo yum -y update" \
    "sudo yum -y install patch tar libaio perl perl-Time-HiRes perl-DBI unixODBC" ; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-fedora19-i386-serial.qcow2 2289 qemu64' '/kvm/vms/vm-fedora19-amd64-serial.qcow2
2290 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade/')" \
    "sudo yum -y update" \
    "sudo yum -y install patch tar mysql-server libtool-ltdl unixODBC" \
    "sudo systemctl enable mysqld.service" \
    "sudo systemctl start mysqld.service" \
    'mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into
t values (1); select * from t"' ; \
done
```

VMs for MariaDB upgrade testing

```
for i in '/kvm/vms/vm-fedora19-amd64-serial.qcow2 2290 qemu64' '/kvm/vms/vm-fedora19-i386-serial.qcow2
2289 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    'sudo yum -y update' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB-${2}.repo
buildbot@localhost:/tmp/MariaDB.repo" \
    'sudo rpm --verbose --import https://yum.mariadb.org/RPM-GPG-KEY-MariaDB' \
    'sudo mv -vi /tmp/MariaDB.repo /etc/yum.repos.d/' \
    'sudo yum -y remove mysql-libs' \
    'sudo yum -y install MariaDB-server MariaDB-client MariaDB-test libtool-ltdl unixODBC' \
    'sudo yum -y install cronie cronie-anacron crontabs.noarch patch tar' \
    'sudo /etc/init.d/mysql start' \
    'mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into
t values (1); select * from t"' \
    'sudo rm -v /etc/yum.repos.d/MariaDB.repo' \
    'sudo yum -y update' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    'sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/'; \
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Fedora 20

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-fedora20-i386-serial.qcow2 20G
qemu-img create -f qcow2 /kvm/vms/vm-fedora20-amd64-serial.qcow2 20G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 2048 -hda /kvm/vms/vm-fedora20-i386-serial.qcow2 -cdrom /ds413/iso/fedora/Fedora-20-i386-DVD.iso -
redir tcp:2291::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
kvm -m 2048 -hda /kvm/vms/vm-fedora20-amd64-serial.qcow2 -cdrom /ds413/iso/fedora/Fedora-20-x86_64-
DVD.iso -redir tcp:2292::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote-host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. The Fedora installer has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- Set the language to English(US)
- Set the timezone to Etc/Greenwich Mean Time timezone
- Change “Software Selection” to “Minimal Install” (default is “Gnome Desktop”)
- Under “Network Configuration” set the hostnames to fedora20-amd64 and fedora20-i386
- For “Installation Destination” select the disk then click continue.
  - On “Installation Options” screen, select the “Partition scheme” drop-down menu and select “Standard Partition”. We do not want LVM.
  - do not check the encryption checkbox

- Select the “Begin installation” button to start the install
- While installing, set the root password and an initial user.
  - Be sure the initial user is an administrator

When the install is finished, you will be prompted to reboot. Go ahead and do so, but it will fail. Kill the VM after the reboot fails and start it up again with:

```
kvm -m 2048 -hda /kvm/vms/vm-fedora20-i386-serial.qcow2 -redir tcp:2291::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user
kvm -m 2048 -hda /kvm/vms/vm-fedora20-amd64-serial.qcow2 -redir tcp:2292::22 -boot c -smp 2 -cpu qemu64 -
net nic,model=virtio -net user
```

Log in using the initial user created during the install:

```
ssh -p 2291 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost
ssh -p 2292 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost
```

After logging in, add the initial user to the “wheel” group.

```
sudo usermod -a -G wheel ${username}
```

Enable password-less sudo for the “wheel” group and serial console sudo:

```
sudo visudo
# Uncomment the line "%wheel        ALL=(ALL)       NOPASSWD: ALL"
# Comment out this line:
# Defaults    requiretty
```

Edit /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

Logout and then, from the VM host server:

Create a .ssh folder:

```
ssh -t -p 2291 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
ssh -t -p 2292 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "mkdir -v .ssh"
```

Copy over the authorized keys file:

```
scp -P 2291 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
scp -P 2292 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/authorized_keys
localhost:.ssh/
```

Set permissions on the .ssh folder correctly:

```
ssh -t -p 2291 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -vR go-rwx .ssh"
ssh -t -p 2292 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ~/.ssh/buildbot.id_dsa
localhost "chmod -vR go-rwx .ssh"
```

Create the buildbot user:

```
ssh -t -p 2291 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -vR go-rwx
.ssh; sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
ssh -t -p 2292 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no localhost 'chmod -vR go-rwx
.ssh; sudo adduser buildbot; sudo usermod -a -G wheel buildbot; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
```

su to the local buildbot user and ssh to the vm to put the key in known_hosts:

For i386:

```
sudo su - buildbot
ssh -p 2291 buildbot@localhost
# exit, then exit again
```

For amd64:

```
sudo su - buildbot
ssh -p 2292 buildbot@localhost
# exit, then exit again
```

Upload the ttyS0 file and put it where it goes:

```
scp -P 2291 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:
scp -P 2292 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/ttyS0
buildbot@localhost:

ssh -t -p 2291 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mkdir
-v /etc/event.d;sudo mv -vi ttyS0 /etc/event.d/;'
ssh -t -p 2292 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo mkdir
-v /etc/event.d;sudo mv -vi ttyS0 /etc/event.d/;'
```

Update the VM:

```
ssh -t -p 2291 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
ssh -t -p 2292 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost
```

Once logged in:

```
sudo yum -y update
```

Change selinux policy to `permissive`:

```
sudo sed -i 's/SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config
```

After updating, exit, then shut down the VM:

```
exit
```

```
ssh -t -p 2291 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo
shutdown -h now;exit'
ssh -t -p 2292 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost 'sudo
shutdown -h now;exit'
```

VMs for building .rpms

```
for i in '/kvm/vms/vm-fedora20-i386-serial.qcow2 2291 qemu64' '/kvm/vms/vm-fedora20-amd64-serial.qcow2
2292 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_build_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed
-e 's/serial/build/')" \
  "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/thrift-0.9.0.tar.gz
buildbot@localhost:/dev/shm/" \
  "sudo yum -y groupinstall 'Development Tools'" \
  "sudo yum -y install yum-utils" \
  "sudo yum-builddep -y mariadb" \
  "sudo yum -y install automake libtool flex pkgconfig gcc-c++ libevent-devel python-devel ruby-devel
rpm-build Judy-devel" \
  "sudo yum -y install cmake tar wget tree gperf readline-devel ncurses-devel zlib-devel pam-devel
libaio-devel openssl-devel" \
  "sudo yum -y install libxml2 libxml2-devel bison bison-devel boost-devel unixODBC-devel bzr perl perl\
(DBI\)" \
  "sudo yum -y remove systemtap-sdt-dev" \
  "bzr co --lightweight lp:mariadb-native-client" \
  "sudo mkdir -vp /usr/src/redhat/SOURCES /usr/src/redhat/SPECS /usr/src/redhat/RPMS
/usr/src/redhat/SRPMS" \
  "cd /usr/local/src;sudo tar zxf /dev/shm/thrift-0.9.0.tar.gz;pwd;ls" \
  "cd /usr/local/src/thrift-0.9.0;echo;pwd;sudo ./configure --prefix=/usr --enable-shared=no --enable-
static=yes CXXFLAGS=-fPIC CFLAGS=-fPIC && echo && echo 'now making' && echo && sleep 3 && sudo make &&
echo && echo 'now installing' && echo && sleep 3 && sudo make install" ; \
done
```

VMs for install testing.

```
for i in '/kvm/vms/vm-fedora20-i386-serial.qcow2 2291 qemu64' '/kvm/vms/vm-fedora20-amd64-serial.qcow2
2292 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_install_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 |
sed -e 's/serial/install/')" \
    "sudo yum -y update" \
    "sudo yum -y install patch tar libaio perl perl-Time-HiRes perl-DBI unixODBC Judy" ; \
done
```

VMs for MariaDB upgrade testing (Fedora repo)

```
for i in '/kvm/vms/vm-fedora20-i386-serial.qcow2 2291 qemu64' '/kvm/vms/vm-fedora20-amd64-serial.qcow2
2292 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_upgrade_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 |
sed -e 's/serial/upgrade/')" \
    "sudo yum -y update" \
    "sudo yum -y install patch tar mariadb-server libtool-ltdl unixODBC Judy" \
    "sudo systemctl enable mariadb.service" \
    "sudo systemctl start mariadb.service" \
    'mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into
t values (1); select * from t"' ; \
done
```

VMs for MariaDB upgrade testing (MariaDB repo)

```
for i in '/kvm/vms/vm-fedora20-amd64-serial.qcow2 2292 qemu64' '/kvm/vms/vm-fedora20-i386-serial.qcow2
2291 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_upgrade2_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 |
sed -e 's/serial/upgrade2/')" \
    'sudo yum -y update' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB-${2}.repo
buildbot@localhost:/tmp/MariaDB.repo" \
    'sudo rpm --verbose --import https://yum.mariadb.org/RPM-GPG-KEY-MariaDB' \
    'sudo mv -vi /tmp/MariaDB.repo /etc/yum.repos.d/' \
    'sudo yum -y remove mysql-libs' \
    'sudo yum -y install MariaDB-server MariaDB-client MariaDB-test libtool-ltdl unixODBC Judy' \
    'sudo yum -y install cronie cronie-anacron crontabs.noarch patch tar' \
    'sudo /etc/init.d/mysql start' \
    'mysql -uroot -e "create database mytest; use mytest; create table t(a int primary key); insert into
t values (1); select * from t"' \
    'sudo rm -v /etc/yum.repos.d/MariaDB.repo' \
    'sudo yum -y update' \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/vms/MariaDB.local.repo
buildbot@localhost:/tmp/" \
    'sudo mv -vi /tmp/MariaDB.local.repo /etc/yum.repos.d/'; \
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Red Hat 6 x86

The following steps were used to create a Red Hat 6 x86 build slave.

Initial Setup

```
cd vms
qemu-img create -f qcow2 vm-rhel6-x86-base.qcow2 8G
kvm -m 1024 -hda vm-rhel6-x86-base.qcow2 -cdrom ../iso/red-hat/rhel-server-6.0-i386-dvd.iso -redir
'tcp:22275::22' -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
```

When the VM boots. Go through the prompts.

Re-initialize the drive, when prompted.

Set the Hostname to “rhel6-x86”.

Configure Network, set eth0 to “Connect Automatically”

Set the root password.

Set partitioning type to: Use All Space

Don’t encrypt the partitions (just adds overhead).

Select “Write Changes to Disk” on the popup that appears.

Set the software set to “Virtual Host”.

After clicking next on the install-type page, the installation will finally start.

After installation completes, click reboot. Then shutdown the VM.

Serial Console Setup

```
cd vms
qemu-img create -b vm-rhel6-x86-base.qcow2 -f qcow2 vm-rhel6-x86-serial.qcow2
kvm -m 1024 -hda vm-rhel6-x86-serial.qcow2 -redir 'tcp:22275::22' -boot c -smp 2 -cpu qemu32,-nx -net
nic,model=virtio -net user
```

Login as root.

Add to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line (after removing ‘quiet’):

```
console=tty0 console=ttyS0,115200n8
```

Add login prompt on serial console:

```
cat >>/etc/inittab <<END

# Serial console.
S0:2345:respawn:/sbin/agetty -h -L ttyS0 19200 vt100
END
```

Create buildbot account

With the network up and running, it’s time to add a user so that we don’t have to login as root all the time.

```
useradd buildbot
# Password is disabled by default in Centos5.
usermod -a -G wheel buildbot
visudo
# Uncomment the line "%wheel        ALL=(ALL)       NOPASSWD: ALL"
# Comment out this line:
# Defaults    requiretty

# Put in public ssh key for own account and host buildbot account.
# Note that Centos5 seems to require .ssh/authorized_keys chmod go-rwx.
su - buildbot
mkdir .ssh
chmod go-rwx .ssh

# scp the key from the vm host over to .ssh/authorized_keys

chmod go-rwx .ssh/authorized_keys
```

Now logout and then ssh to the VM as the buildbot user. On my local box I added the following to my /.ssh/config file to make logging in easier:

```
Host rhel6-x86
HostName localhost
  User buildbot
  IdentityFile ~/.ssh/id_rsa
  Port 2225
```

With the above in place I can simply type:

```
ssh rhel6-x86
```

…to connect to the vm.

RHN and Updates

Register the system with RHN:

```
sudo rhn_register
```

Choose defaults when registering. After the process is complete:

```
sudo yum update
```

The first time you update you'll be prompted to import some GPG keys from Red
Hat. The updating process may take a while, depending on the number of updates and the speed of your Internet connection.

After updating shutdown so we can make more copies.

```
sudo shutdown -h now
```

Image for RPM Build

```
qemu-img create -b vm-rhel6-x86-serial.qcow2 -f qcow2 vm-rhel6-x86-build.qcow2
kvm -m 1024 -hda vm-rhel6-x86-build.qcow2 -redir 'tcp:22275::22' -boot c -smp 2 -cpu qemu32,-nx -net
nic,model=virtio -net user -nographic
```

Wait for the system to boot.

Install compilers etc:

```
sudo yum groupinstall "Development Tools"
sudo yum install libaio-devel openssl-devel
```

If the "Development Tools" group is not available, the following lines will install the packages from it:

```
# Mandatory Packages:
sudo yum install autoconf automake binutils bison flex gcc gcc-c++ gettext
sudo yum install libtool make patch pkgconfig redhat-rpm-config rpm-build

# Default Packages:
sudo yum install byacc cscope ctags cvs diffstat doxygen elfutils gcc-gfortran
sudo yum install git indent intltool patchutils rcs subversion swig systemtap

# Optional Packages:
sudo yum install ElectricFence ant babel bzr chrpath cmake compat-gcc-34
sudo yum install compat-gcc-34-c++ compat-gcc-34-g77 dejagnu expect gcc-gnat
sudo yum install gcc-java gcc-objc gcc-objc++ imake jpackage-utils libstdc++-docs
sudo yum install mercurial mod_dav_svn nasm perltidy python-docs rpmdevtools
sudo yum install rpmlint systemtap-sdt-devel systemtap-server
```

Other packages to install:

```
sudo yum install gperf readline-devel ncurses-devel zlib-devel perl perl\(DBI\)
```

Create rpm directories and download 5.0 rpm for shared-compat:

```
sudo mkdir -vp /usr/src/redhat/SOURCES /usr/src/redhat/SPECS /usr/src/redhat/RPMS /usr/src/redhat/SRPMS
sudo mkdir -vp /srv/shared/yum/CentOS/5/i386/RPMS/
cd /srv/shared/yum/CentOS/5/i386/RPMS/
sudo wget http://mirror.ourdelta.org/yum/CentOS/5/i386/RPMS/MySQL-OurDelta-shared-5.0.87.d10-
65.el5.i386.rpm
```

Image for install/test

```
qemu-img create -b vm-rhel6-x86-serial.qcow2 -f qcow2 vm-rhel6-x86-install.qcow2
kvm -m 1024 -hda vm-rhel6-x86-install.qcow2 -redir 'tcp:22275::22' -boot c -smp 2 -cpu qemu32,-nx -net
nic,model=virtio -net user -nographic
```

Install extra dependencies:

```
sudo yum install perl perl\(DBI\)
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu

See Buildbot Setup for Virtual Machines - Ubuntu 10.04 (alpha), i386 and amd64.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 10.10 “maverick”.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 11.04 “natty”.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 11.10 “oneiric”.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 12.04 “precise”.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 12.10 “quantal”.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 13.04 “raring”.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 13.10 “saucy”.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 14.04 “trusty”.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 8.04, 8.10, and 9.10.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 9.04 amd64.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

See Buildbot Setup for Virtual Machines - Ubuntu 9.04 i386.

Explains how to provision and configure virtual machine instances specifically for Buildbot testing.

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 10.04 (alpha), i386 and amd64

Base Install

```
qemu-img create -f qcow2 vm-lucid-amd64-serial.qcow2 8G
kvm -m 1024 -hda vm-lucid-amd64-serial.qcow2 -cdrom /kvm/lucid-server-amd64.iso -redir tcp:2238::22 -boot
d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-lucid-amd64-serial.qcow2 -cdrom /kvm/lucid-server-amd64.iso -redir tcp:2238::22 -boot
c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
ssh -p 2238 localhost
# edit /boot/grub/menu.lst and visudo, see below
ssh -t -p 2238 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2238 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2238 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2238 ttyS0.conf buildbot@localhost:
ssh -p 2238 buildbot@localhost 'sudo cp ttyS0.conf /etc/init/; rm ttyS0.conf; sudo shutdown -h now'
```

```
qemu-img create -f qcow2 vm-lucid-i386-serial.qcow2 8G
kvm -m 1024 -hda vm-lucid-i386-serial.qcow2 -cdrom /kvm/lucid-server-i386.iso -redir tcp:2239::22 -boot d
-smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-lucid-i386-serial.qcow2 -cdrom /kvm/lucid-server-i386.iso -redir tcp:2239::22 -boot c
-smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user -nographic
ssh -p 2239 localhost
# edit /boot/grub/menu.lst and visudo, see below
ssh -t -p 2239 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2239 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2239 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2239 ttyS0.conf buildbot@localhost:
ssh -p 2239 buildbot@localhost 'sudo cp ttyS0.conf /etc/init/; rm ttyS0.conf; sudo shutdown -h now'
```

Enabling passwordless sudo:

```
sudo VISUAL=vi visudo
# uncomment `%sudo ALL=NOPASSWD: ALL' line in `visudo`, and move to end.
```

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
```

VMs for building .debs

```
for i in 'vm-lucid-amd64-serial.qcow2 2238 qemu64' 'vm-lucid-i386-serial.qcow2 2239 qemu32,-nx' ; do \
  set $i; \
  runvm --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.1" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libreadline5-dev" ; \
done
```

VMs for install testing.

See above for how to obtain my.seed and sources.append.

```
cat >sources.append <<END
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
END

for i in 'vm-lucid-amd64-serial.qcow2 2238 qemu64' 'vm-lucid-i386-serial.qcow2 2239 qemu32,-nx' ; do \
  set $i; \
  runvm --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y debconf-utils" \
    "= scp -P $2 my.seed sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for upgrade testing

```
for i in 'vm-lucid-amd64-install.qcow2 2238 qemu64' 'vm-lucid-i386-install.qcow2 2239 qemu32,-nx' ; do \
  set $i; \
  runvm --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/install/upgrade/')" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.1' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ;\
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 10.10 “maverick”

Base Install

```
qemu-img create -f qcow2 vm-maverick-amd64-serial.qcow2 8G
kvm -m 1024 -hda vm-maverick-amd64-serial.qcow2 -cdrom /kvm/ubuntu-10.10-server-amd64.iso -redir
tcp:2246::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-maverick-amd64-serial.qcow2 -cdrom /kvm/ubuntu-10.10-server-amd64.iso -redir
tcp:2246::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
ssh -p 2246 localhost
# edit /boot/grub/menu.lst and visudo, see below
ssh -t -p 2246 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2246 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2246 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2246 ttyS0.conf buildbot@localhost:
ssh -p 2246 buildbot@localhost 'sudo cp ttyS0.conf /etc/init/; rm ttyS0.conf; sudo shutdown -h now'
```

```
qemu-img create -f qcow2 vm-maverick-i386-serial.qcow2 8G
kvm -m 1024 -hda vm-maverick-i386-serial.qcow2 -cdrom /kvm/ubuntu-10.10-server-i386.iso -redir
tcp:2247::22 -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-maverick-i386-serial.qcow2 -cdrom /kvm/ubuntu-10.10-server-i386.iso -redir
tcp:2247::22 -boot c -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user -nographic
ssh -p 2247 localhost
# edit /boot/grub/menu.lst and visudo, see below
ssh -t -p 2247 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2247 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2247 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2247 ttyS0.conf buildbot@localhost:
ssh -p 2247 buildbot@localhost 'sudo cp ttyS0.conf /etc/init/; rm ttyS0.conf; sudo shutdown -h now'
```

Enabling passwordless sudo:

```
sudo VISUAL=vi visudo
# Add line at end: `%sudo ALL=NOPASSWD: ALL'
```

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
```

VMs for building .debs

```
for i in 'vm-maverick-amd64-serial.qcow2 2246 qemu64' 'vm-maverick-i386-serial.qcow2 2247 qemu32,-nx' ;
do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.1" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libreadline6-dev" ; \
done
```

VMs for install testing.

See the General Principles article for how to make the '`my.seed`' and '`sources.append`' files.

```
cat >sources.append <<END
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
END
```

```
for i in 'vm-maverick-amd64-serial.qcow2 2246 qemu64' 'vm-maverick-i386-serial.qcow2 2247 qemu32,-nx' ;
do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y debconf-utils" \
    "= scp -P $2 my.seed sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for upgrade testing

```
for i in 'vm-maverick-amd64-install.qcow2 2246 qemu64' 'vm-maverick-i386-install.qcow2 2247 qemu32,-nx' ;
do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/install/upgrade/')" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.1' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ;\
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 11.04 “natty”

Base Install

```
qemu-img create -f qcow2 vm-natty-amd64-serial.qcow2 8G
kvm -m 1024 -hda vm-natty-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu-11.04-server-amd64.iso -redir
tcp:2255::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user

# Install, picking default options mostly, only adding openssh server.

kvm -m 1024 -hda vm-natty-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu-11.04-server-amd64.iso -redir
tcp:2255::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
ssh -p 2255 localhost

# edit /boot/grub/menu.lst and visudo, see below

ssh -t -p 2255 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2255 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\ny' | ssh -p 2255 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2255 ttyS0.conf buildbot@localhost:
ssh -p 2255 buildbot@localhost 'sudo cp ttyS0.conf /etc/init/; rm ttyS0.conf; sudo shutdown -h now'
```

```
qemu-img create -f qcow2 vm-natty-i386-serial.qcow2 8G
kvm -m 1024 -hda vm-natty-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu-11.04-server-i386.iso -redir
tcp:2256::22 -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user

# Install, picking default options mostly, only adding openssh server.

kvm -m 1024 -hda vm-natty-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu-11.04-server-i386.iso -redir
tcp:2256::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
ssh -p 2256 localhost

# edit /boot/grub/menu.lst and visudo, see below

ssh -t -p 2256 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2256 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2256 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2256 ttyS0.conf buildbot@localhost:
ssh -p 2256 buildbot@localhost 'sudo cp ttyS0.conf /etc/init/; rm ttyS0.conf; sudo shutdown -h now'
```

Enabling passwordless sudo:

```
sudo VISUAL=vi visudo
# Add line at end: `%sudo ALL=NOPASSWD: ALL'
```

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
```

VMs for building .debs

```
for i in 'vm-natty-amd64-serial.qcow2 2255 qemu64' 'vm-natty-i386-serial.qcow2 2256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.1" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libreadline6-dev" ; \
done
```

VMs for install testing.

See above for how to obtain my.seed and sources.append.

```
for i in 'vm-natty-amd64-serial.qcow2 2255 qemu64' 'vm-natty-i386-serial.qcow2 2256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y debconf-utils" \
    "= scp -P $2 my.seed sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for upgrade testing

```
for i in 'vm-natty-amd64-install.qcow2 2255 qemu64' 'vm-natty-i386-install.qcow2 2256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/install/upgrade/')" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.1' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ;\
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 11.10 “oneiric”

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-oneiric-amd64-serial.qcow2 8G
qemu-img create -f qcow2 /kvm/vms/vm-oneiric-i386-serial.qcow2 8G

# Start each VM booting from the server install iso one at a time and perform
# the following install steps:
kvm -m 1024 -hda /kvm/vms/vm-oneiric-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu-11.10-server-amd64.iso -
redir tcp:2255::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
kvm -m 1024 -hda /kvm/vms/vm-oneiric-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu-11.10-server-i386.iso -
redir tcp:2256::22 -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user

# Once running you can connect to the VNC server from your local host with:
#
#    vncviewer -via ${remote-host} localhost
#
# (replace ${remote-host} with the host the vm is running on)

# Install, picking default options mostly, with the following notes:
# * Set the hostname to ubuntu-oneiric
# * When partitioning disks, choose "Guided - use entire disk" (we do not want LVM)
# * No automatic updates
# * Choose software to install: OpenSSH server

# Now that the VM is installed, it's time to configure it.
# If you have the memory you can do the following simultaneously:
kvm -m 1024 -hda /kvm/vms/vm-oneiric-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu-11.10-server-amd64.iso -
redir tcp:2255::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
kvm -m 1024 -hda /kvm/vms/vm-oneiric-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu-11.10-server-i386.iso -
redir tcp:2256::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic

ssh -p 2255 localhost
# edit /boot/grub/menu.lst and visudo, see below

ssh -p 2256 localhost
# edit /boot/grub/menu.lst and visudo, see below

ssh -t -p 2255 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"
ssh -t -p 2256 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"

scp -P 2255 /kvm/vms/authorized_keys localhost:.ssh/
scp -P 2256 /kvm/vms/authorized_keys localhost:.ssh/

echo $'Buildbot\n\n\n\n\ny' | ssh -p 2255 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2256 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'

scp -P 2255 /kvm/vms/ttyS0.conf buildbot@localhost:
scp -P 2256 /kvm/vms/ttyS0.conf buildbot@localhost:

ssh -p 2255 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h
now'
ssh -p 2256 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h
now'
```

Enabling passwordless sudo:

```
sudo VISUAL=vi visudo
# Add line at end: `%sudo ALL=NOPASSWD: ALL'
```

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
```

VMs for building .debs

```
for i in '/kvm/vms/vm-oneiric-amd64-serial.qcow2 2255 qemu64' '/kvm/vms/vm-oneiric-i386-serial.qcow2 2256
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.1" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libreadline6-dev" ; \
done
```

Install cmake (required for MariaDB 5.5) with:

```
kvm -m 1024 -hda ${vm} -redir tcp:22565::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -
nographic

ssh -o IdentityFile=.ssh/buildbot.id_dsa -p 22565 buildbot@localhost

wget http://www.cmake.org/files/v2.8/cmake-2.8.6.tar.gz
tar -zxvf cmake-2.8.6.tar.gz
cd cmake-2.8.6;./configure
make
sudo make install

sudo /sbin/shutdown -h now
```

Also:

- Installing the Boost library needed for the OQGraph storage engine

VMs for install testing.

See above for how to obtain my.seed and sources.append.

```
for i in '/kvm/vms/vm-oneiric-amd64-serial.qcow2 2255 qemu64' '/kvm/vms/vm-oneiric-i386-serial.qcow2 2256
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y debconf-utils" \
    "= scp -P $2 /kvm/vms/my.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-oneiric-amd64-install.qcow2 2255 qemu64' '/kvm/vms/vm-oneiric-i386-install.qcow2
2256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/install/upgrade/')" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.1' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ;\
done
```

VMs for MariaDB upgrade testing

The steps below are based on the Natty steps on Installing VM images for testing .deb upgrade between versions.

64-bit Ubuntu oneiric:

```
qemu-img create -b vm-oneiric-amd64-install.qcow2 -f qcow2 vm-oneiric-amd64-upgrade2.qcow2
kvm -m 512 -hda vm-oneiric-amd64-upgrade2.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=virtio -net user -nographic
ssh -p 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost

sudo sh -c 'cat > /etc/apt/sources.list.d/tmp.list'
deb http://ftp.osuosl.org/pub/mariadb/repo/5.2/ubuntu natty main
deb-src http://ftp.osuosl.org/pub/mariadb/repo/5.2/ubuntu natty main

sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 1BB943DB
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
sudo rm /etc/apparmor.d/usr.sbin.mysqld
sudo /etc/init.d/apparmor restart
sudo apt-get -f install
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
sudo rm /etc/apt/sources.list.d/tmp.list
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
sudo shutdown -h now
```

32-bit Ubuntu oneiric:

```
qemu-img create -b vm-oneiric-i386-install.qcow2 -f qcow2 vm-oneiric-i386-upgrade2.qcow2
kvm -m 512 -hda vm-oneiric-i386-upgrade2.qcow2 -redir 'tcp:2200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=virtio -net user -nographic
ssh -p 2200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost

sudo sh -c 'cat > /etc/apt/sources.list.d/tmp.list'
deb http://ftp.osuosl.org/pub/mariadb/repo/5.2/ubuntu natty main
deb-src http://ftp.osuosl.org/pub/mariadb/repo/5.2/ubuntu natty main

sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 1BB943DB
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
sudo rm /etc/apparmor.d/usr.sbin.mysqld
sudo /etc/init.d/apparmor restart
sudo apt-get -f install
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
sudo rm /etc/apt/sources.list.d/tmp.list
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
sudo shutdown -h now
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 12.04 “precise”

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-precise-amd64-serial.qcow2 8G
qemu-img create -f qcow2 /kvm/vms/vm-precise-i386-serial.qcow2 8G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 1024 -hda /kvm/vms/vm-precise-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-12.04-server-
amd64.iso -redir tcp:22255::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
kvm -m 1024 -hda /kvm/vms/vm-precise-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-12.04-server-
i386.iso -redir tcp:22256::22 -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote-host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. Ubuntu has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- Set the hostname to ubuntu-precise
- When partitioning disks, choose “Guided - use entire disk” (we do not want LVM)
- No automatic updates
- Choose software to install: OpenSSH server

Now that the VM is installed, it’s time to configure it.
If you have the memory you can do the following simultaneously:

```
kvm -m 1024 -hda /kvm/vms/vm-precise-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-12.04-server-amd64.iso -redir tcp:22255::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
kvm -m 1024 -hda /kvm/vms/vm-precise-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-12.04-server-i386.iso -redir tcp:22256::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic

ssh -p 22255 localhost
# edit /boot/grub/menu.lst and visudo, see below

ssh -p 22256 localhost
# edit /boot/grub/menu.lst and visudo, see below

ssh -t -p 22255 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"
ssh -t -p 22256 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"

scp -P 22255 /kvm/vms/authorized_keys localhost:.ssh/
scp -P 22256 /kvm/vms/authorized_keys localhost:.ssh/

echo $'Buildbot\n\n\n\n\ny' | ssh -p 22255 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp -vi .ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-rwx ~buildbot/.ssh'
echo $'Buildbot\n\n\n\n\ny' | ssh -p 22256 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp -vi .ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-rwx ~buildbot/.ssh'

scp -P 22255 /kvm/vms/ttyS0.conf buildbot@localhost:
scp -P 22256 /kvm/vms/ttyS0.conf buildbot@localhost:

ssh -p 22255 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h now'
ssh -p 22256 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h now'
```

Enabling passwordless sudo:

```
sudo VISUAL=vi visudo
# Add line at end: `%sudo ALL=NOPASSWD: ALL'
```

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
```

VMs for building .debs

```
for i in '/kvm/vms/vm-precise-amd64-serial.qcow2 22255 qemu64' '/kvm/vms/vm-precise-i386-serial.qcow2 22256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e 's/serial/build/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.5" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libpam0g-dev libreadline-gplv2-dev autoconf automake automake1.9 defoma dpatch ghostscript-x libfontenc1 libjpeg62 libltdl-dev libltdl7 libmail-sendmail-perl libxfont1 lmodern psfontmgr texlive-latex-base-doc ttf-dejavu ttf-dejavu-extra libaio-dev xfonts-encodings xfonts-utils" ; \
done
```

Also:

- Installing the Boost library needed for the OQGraph storage engine

VMs for install testing.

See Buildbot Setup for Virtual Machines - General Principles for how to obtain `my.seed` and `sources.append`.

```
for i in '/kvm/vms/vm-precise-amd64-serial.qcow2 22255 qemu64' '/kvm/vms/vm-precise-i386-serial.qcow2
22256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils" \
    "= scp -P $2 /kvm/vms/my.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-precise-amd64-install.qcow2 22255 qemu64' '/kvm/vms/vm-precise-i386-install.qcow2
22256 qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/install/upgrade/')" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y libaio1 mysql-server-5.5' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ;\
done
```

VMs for MariaDB upgrade testing

The steps below are based on the Natty steps on Installing VM images for testing .deb upgrade between versions.

64-bit Ubuntu precise:

```
qemu-img create -b vm-precise-amd64-install.qcow2 -f qcow2 vm-precise-amd64-upgrade2.qcow2
kvm -m 512 -hda vm-precise-amd64-upgrade2.qcow2 -redir 'tcp:22200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=virtio -net user -nographic
ssh -p 22200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost

sudo sh -c 'cat > /etc/apt/sources.list.d/tmp.list'
deb http://ftp.osuosl.org/pub/mariadb/repo/5.5/ubuntu oneiric main
deb-src http://ftp.osuosl.org/pub/mariadb/repo/5.5/ubuntu oneiric main

sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 1BB943DB
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
sudo apt-get -f install
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
sudo rm /etc/apt/sources.list.d/tmp.list
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y libaio1
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
sudo shutdown -h now
```

32-bit Ubuntu precise:

```
qemu-img create -b vm-precise-i386-install.qcow2 -f qcow2 vm-precise-i386-upgrade2.qcow2
kvm -m 512 -hda vm-precise-i386-upgrade2.qcow2 -redir 'tcp:22200::22' -boot c -smp 1 -cpu qemu64 -net
nic,model=virtio -net user -nographic
ssh -p 22200 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no buildbot@localhost

sudo sh -c 'cat > /etc/apt/sources.list.d/tmp.list'
deb http://ftp.osuosl.org/pub/mariadb/repo/5.5/ubuntu oneiric main
deb-src http://ftp.osuosl.org/pub/mariadb/repo/5.5/ubuntu oneiric main

sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 1BB943DB
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y --allow-unauthenticated mariadb-server mariadb-
test libmariadbclient-dev
sudo apt-get -f install
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
sudo rm /etc/apt/sources.list.d/tmp.list
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y libaio1
sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y
sudo shutdown -h now
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 12.10 “quantal”

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-quantal-amd64-serial.qcow2 10G
qemu-img create -f qcow2 /kvm/vms/vm-quantal-i386-serial.qcow2 10G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 2048 -hda /kvm/vms/vm-quantal-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-12.10-server-
amd64.iso -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2275-:22
kvm -m 2048 -hda /kvm/vms/vm-quantal-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-12.10-server-
i386.iso -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2276-:22
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote-host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. Ubuntu has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- Set the hostname to ubuntu-quantal-amd64 or ubuntu-quantal-i386
- When partitioning disks, choose “Guided - use entire disk” (we do not want LVM)
- No automatic updates
- Choose software to install: OpenSSH server

Now that the VM is installed, it’s time to configure it.
If you have the memory you can do the following simultaneously:

```
kvm -m 2048 -hda /kvm/vms/vm-quantal-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-12.10-server-amd64.iso -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2275-:22 -nographic
kvm -m 2048 -hda /kvm/vms/vm-quantal-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-12.10-server-i386.iso -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2276-:22 -nographic

ssh -p 2275 localhost
# edit /boot/grub/menu.lst and visudo, see below

ssh -p 2276 localhost
# edit /boot/grub/menu.lst and visudo, see below

ssh -t -p 2275 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"
ssh -t -p 2276 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"

scp -P 2275 /kvm/vms/authorized_keys localhost:.ssh/
scp -P 2276 /kvm/vms/authorized_keys localhost:.ssh/

echo $'Buildbot\n\n\n\n\ny' | ssh -p 2275 localhost 'chmod -vR go-rwx .ssh; sudo adduser --disabled-password buildbot; sudo addgroup buildbot sudo; sudo mkdir -v ~buildbot/.ssh; sudo cp -vi .ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-rwx ~buildbot/.ssh'
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2276 localhost 'chmod -vR go-rwx .ssh; sudo adduser --disabled-password buildbot; sudo addgroup buildbot sudo; sudo mkdir -v ~buildbot/.ssh; sudo cp -vi .ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-rwx ~buildbot/.ssh'

scp -P 2275 /kvm/vms/ttyS0.conf buildbot@localhost:
scp -P 2276 /kvm/vms/ttyS0.conf buildbot@localhost:

ssh -p 2275 buildbot@localhost 'sudo apt-get update && sudo apt-get -y dist-upgrade;'
ssh -p 2276 buildbot@localhost 'sudo apt-get update && sudo apt-get -y dist-upgrade;'

ssh -p 2275 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h now'
ssh -p 2276 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h now'
```

Enabling passwordless sudo:

```
sudo VISUAL=vi visudo
# Add line at end: `%sudo ALL=NOPASSWD: ALL'
```

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
# exit back to the host server
```

VMs for building .debs

```
for i in '/kvm/vms/vm-quantal-amd64-serial.qcow2 2275 qemu64' '/kvm/vms/vm-quantal-i386-serial.qcow2 2276
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/boost_1_49_0.tar.gz
buildbot@localhost:/dev/shm/" \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/thrift-0.9.0.tar.gz
buildbot@localhost:/dev/shm/" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.5" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libpam0g-dev libreadline-gplv2-dev
autoconf automake automake1.9 defoma dpatch ghostscript-x libfontenc1 libjpeg62 libltdl-dev libltdl7
libmail-sendmail-perl libxfont1 lmodern psfontmgr texlive-latex-base-doc ttf-dejavu ttf-dejavu-extra
libaio-dev xfonts-encodings xfonts-utils libxml2-dev unixodbc-dev" \
    "cd /usr/local/src;sudo tar zxf /dev/shm/thrift-0.9.0.tar.gz;pwd;ls" \
    "cd /usr/local/src/thrift-0.9.0;echo;pwd;sudo ./configure --prefix=/usr --enable-shared=no --enable-
static=yes CXXFLAGS=-fPIC CFLAGS=-fPIC && echo && echo 'now making' && echo && sleep 3 && sudo make &&
echo && echo 'now installing' && echo && sleep 3 && sudo make install" \
    "cd /usr/local/src;sudo tar zxf /dev/shm/boost_1_49_0.tar.gz;cd /usr/local/include/;sudo ln -vs
../src/boost_1_49_0/boost ." ; \
done
```

VMs for install testing.

See Buildbot Setup for Virtual Machines - General Principles for how to obtain `my.seed` and `sources.append`.

```
for i in '/kvm/vms/vm-quantal-amd64-serial.qcow2 2275 qemu64' '/kvm/vms/vm-quantal-i386-serial.qcow2 2276
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-quantal-amd64-serial.qcow2 2275 qemu64' '/kvm/vms/vm-quantal-i386-serial.qcow2 2276
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.5' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ;\
done
```

VMs for MariaDB upgrade testing

The steps below are based on the Natty steps on Installing VM images for testing .deb upgrade between versions.

```
for i in '/kvm/vms/vm-quantal-amd64-serial.qcow2 2275 qemu64' '/kvm/vms/vm-quantal-i386-serial.qcow2 2276
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "= scp -P $2 /kvm/vms/mariadb-quantal.list buildbot@localhost:/tmp/tmp.list" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    'sudo mv -vi /tmp/tmp.list /etc/apt/sources.list.d/' \
    'sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 0xcbcb082a1bb943db' \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mariadb-server mariadb-server-5.5 mariadb-
client mariadb-client-5.5 mariadb-test libmariadbclient-dev' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    'sudo rm -v /etc/apt/sources.list.d/tmp.list' \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get update' \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils' \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y'; \
done
```

Add Key to known_hosts

Do the following on each kvm host server (terrier, terrier2, i7, etc…) to add the VMs to known_hosts.

```
# quantal-amd64
cp -avi /kvm/vms/vm-quantal-amd64-install.qcow2 /kvm/vms/vm-quantal-amd64-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-quantal-amd64-test.qcow2 -redir tcp:2275::22 -boot c -smp 2 -cpu qemu64 -net
nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2275 buildbot@localhost sudo shutdown -h now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-quantal-amd64-test.qcow2

# quantal-i386
cp -avi /kvm/vms/vm-quantal-i386-install.qcow2 /kvm/vms/vm-quantal-i386-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-quantal-i386-test.qcow2 -redir tcp:2276::22 -boot c -smp 2 -cpu qemu64 -net
nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2276 buildbot@localhost sudo shutdown -h now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-quantal-i386-test.qcow2
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 13.04 “raring”

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-raring-amd64-serial.qcow2 15G
qemu-img create -f qcow2 /kvm/vms/vm-raring-i386-serial.qcow2 15G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 2048 -hda /kvm/vms/vm-raring-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-13.04-server-
amd64.iso -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2279-:22
kvm -m 2048 -hda /kvm/vms/vm-raring-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-13.04-server-i386.iso
-boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2280-:22
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote-host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. Ubuntu has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- Set the hostname to ubuntu-raring-amd64 or ubuntu-raring-i386
- When partitioning disks, choose “Guided - use entire disk” (we do not want LVM)
- No automatic updates
- Choose software to install: OpenSSH server

Now that the VM is installed, it’s time to configure it.
If you have the memory you can do the following simultaneously:

```
kvm -m 2048 -hda /kvm/vms/vm-raring-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-13.04-server-
amd64.iso -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2279-:22 -
nographic
kvm -m 2048 -hda /kvm/vms/vm-raring-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-13.04-server-i386.iso
-boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2280-:22 -nographic

ssh -p 2279 localhost
# edit /boot/grub/menu.lst and visudo, see below

ssh -p 2280 localhost
# edit /boot/grub/menu.lst and visudo, see below

ssh -t -p 2279 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"
ssh -t -p 2280 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"

scp -P 2279 /kvm/vms/authorized_keys localhost:.ssh/
scp -P 2280 /kvm/vms/authorized_keys localhost:.ssh/

echo $'Buildbot\n\n\n\n\ny' | ssh -p 2279 localhost 'chmod -vR go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir -v ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2280 localhost 'chmod -vR go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir -v ~buildbot/.ssh; sudo cp -vi
.ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-
rwx ~buildbot/.ssh'

scp -P 2279 /kvm/vms/ttyS0.conf buildbot@localhost:
scp -P 2280 /kvm/vms/ttyS0.conf buildbot@localhost:

ssh -p 2279 buildbot@localhost 'sudo apt-get update && sudo apt-get -y dist-upgrade;'
ssh -p 2280 buildbot@localhost 'sudo apt-get update && sudo apt-get -y dist-upgrade;'

ssh -p 2279 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h
now'
ssh -p 2280 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h
now'
```

Enabling passwordless sudo:

```
sudo VISUAL=vi visudo
# Add line at end: `%sudo ALL=NOPASSWD: ALL'
```

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
GRUB_HIDDEN_TIMEOUT_QUIET=false
GRUB_TIMEOUT=0
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
# exit back to the host server
```

VMs for building .debs

```
for i in '/kvm/vms/vm-raring-amd64-serial.qcow2 2279 qemu64' '/kvm/vms/vm-raring-i386-serial.qcow2 2280
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/boost_1_49_0.tar.gz
buildbot@localhost:/dev/shm/" \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/thrift-0.9.0.tar.gz
buildbot@localhost:/dev/shm/" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.5" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libpam0g-dev libreadline-gplv2-dev
autoconf automake automake1.9 dpatch ghostscript-x libfontenc1 libjpeg62 libltdl-dev libltdl7 libmail-
sendmail-perl libxfont1 lmodern texlive-latex-base-doc ttf-dejavu ttf-dejavu-extra libaio-dev xfonts-
encodings xfonts-utils libxml2-dev unixodbc-dev bzr" \
    "bzr co --lightweight lp:mariadb-native-client" \
    "cd /usr/local/src;sudo tar zxf /dev/shm/thrift-0.9.0.tar.gz;pwd;ls" \
    "cd /usr/local/src/thrift-0.9.0;echo;pwd;sudo ./configure --prefix=/usr --enable-shared=no --enable-
static=yes CXXFLAGS=-fPIC CFLAGS=-fPIC && echo && echo 'now making' && echo && sleep 3 && sudo make &&
echo && echo 'now installing' && echo && sleep 3 && sudo make install" \
    "cd /usr/local/src;sudo tar zxf /dev/shm/boost_1_49_0.tar.gz;cd /usr/local/include/;sudo ln -vs
../src/boost_1_49_0/boost ." ; \
done
```

VMs for install testing.

See Buildbot Setup for Virtual Machines - General Principles for how to obtain `my.seed` and `sources.append`.

```
for i in '/kvm/vms/vm-raring-amd64-serial.qcow2 2279 qemu64' '/kvm/vms/vm-raring-i386-serial.qcow2 2280
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-raring-amd64-serial.qcow2 2279 qemu64' '/kvm/vms/vm-raring-i386-serial.qcow2 2280
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.5' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ;\
done
```

VMs for MariaDB upgrade testing

The steps below are based on the Natty steps on Installing VM images for testing .deb upgrade between versions.

```
for i in '/kvm/vms/vm-raring-amd64-serial.qcow2 2279 qemu64' '/kvm/vms/vm-raring-i386-serial.qcow2 2280
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "= scp -P $2 /kvm/vms/mariadb-raring.list buildbot@localhost:/tmp/tmp.list" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    'sudo mv -vi /tmp/tmp.list /etc/apt/sources.list.d/' \
    'sudo apt-key adv --recv-keys --keyserver pgp.mit.edu 0xcbcb082a1bb943db' \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mariadb-server mariadb-server-5.5 mariadb-
client mariadb-client-5.5 mariadb-test libmariadbclient-dev mysql-common=5.5.30-mariadb1~quantal
libmysqlclient18=5.5.30-mariadb1~quantal' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    'sudo rm -v /etc/apt/sources.list.d/tmp.list' \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get update' \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils' \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y'; \
done
```

Add Key to known_hosts

Do the following on each kvm host server (terrier, terrier2, i7, etc…) to add the VMs to known_hosts.

```
# raring-amd64
cp -avi /kvm/vms/vm-raring-amd64-install.qcow2 /kvm/vms/vm-raring-amd64-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-raring-amd64-test.qcow2 -redir tcp:2279::22 -boot c -smp 2 -cpu qemu64 -net
nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2279 buildbot@localhost sudo shutdown -h now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-raring-amd64-test.qcow2

# raring-i386
cp -avi /kvm/vms/vm-raring-i386-install.qcow2 /kvm/vms/vm-raring-i386-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-raring-i386-test.qcow2 -redir tcp:2280::22 -boot c -smp 2 -cpu qemu64 -net
nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2280 buildbot@localhost sudo shutdown -h now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-raring-i386-test.qcow2
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 13.10 “saucy”

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-saucy-amd64-serial.qcow2 20G
qemu-img create -f qcow2 /kvm/vms/vm-saucy-i386-serial.qcow2 20G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 2048 -hda /kvm/vms/vm-saucy-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-13.10-server-
amd64.iso -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2287-:22
kvm -m 2048 -hda /kvm/vms/vm-saucy-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-13.10-server-i386.iso
-boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2288-:22
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote_host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. Ubuntu has just resized the vnc screen. Simply reconnect.

Install, picking default options mostly, with the following notes:

- Set the hostname to ubuntu-saucy-amd64 or ubuntu-saucy-i386
- do not encrypt the home directory
- When partitioning disks, choose “Guided - use entire disk” (we do not want LVM)
- No automatic updates
- Choose software to install: OpenSSH server

Now that the VM is installed, it’s time to configure it.
If you have the memory you can do the following simultaneously:

```
kvm -m 2048 -hda /kvm/vms/vm-saucy-amd64-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-13.10-server-amd64.iso -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2287-:22 -nographic
kvm -m 2048 -hda /kvm/vms/vm-saucy-i386-serial.qcow2 -cdrom /kvm/iso/ubuntu/ubuntu-13.10-server-i386.iso -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user,hostfwd=tcp:127.0.0.1:2288-:22 -nographic

ssh -p 2287 localhost
sudo update-alternatives --config editor
# edit /boot/grub/menu.lst and visudo, see below

ssh -p 2288 localhost
sudo update-alternatives --config editor
# edit /boot/grub/menu.lst and visudo, see below

ssh -t -p 2287 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"
ssh -t -p 2288 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"

scp -P 2287 /kvm/vms/authorized_keys localhost:.ssh/
scp -P 2288 /kvm/vms/authorized_keys localhost:.ssh/

echo $'Buildbot\n\n\n\n\ny' | ssh -p 2287 localhost 'chmod -vR go-rwx .ssh; sudo adduser --disabled-password buildbot; sudo addgroup buildbot sudo; sudo mkdir -v ~buildbot/.ssh; sudo cp -vi .ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-rwx ~buildbot/.ssh'
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2288 localhost 'chmod -vR go-rwx .ssh; sudo adduser --disabled-password buildbot; sudo addgroup buildbot sudo; sudo mkdir -v ~buildbot/.ssh; sudo cp -vi .ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-rwx ~buildbot/.ssh'

scp -P 2287 /kvm/vms/ttyS0.conf buildbot@localhost:
scp -P 2288 /kvm/vms/ttyS0.conf buildbot@localhost:

ssh -p 2287 buildbot@localhost 'sudo apt-get update && sudo apt-get -y dist-upgrade;'
ssh -p 2288 buildbot@localhost 'sudo apt-get update && sudo apt-get -y dist-upgrade;'

ssh -p 2287 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h now'
ssh -p 2288 buildbot@localhost 'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h now'
```

Enabling passwordless sudo:

```
sudo VISUAL=vi visudo
# Add line at end: `%sudo ALL=NOPASSWD: ALL'
```

Editing /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
GRUB_HIDDEN_TIMEOUT_QUIET=false
GRUB_TIMEOUT=0
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
# exit back to the host server
```

VMs for building .debs

```
for i in '/kvm/vms/vm-saucy-amd64-serial.qcow2 2287 qemu64' '/kvm/vms/vm-saucy-i386-serial.qcow2 2288
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/thrift-0.9.0.tar.gz
buildbot@localhost:/dev/shm/" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.5" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libpam0g-dev libreadline-gplv2-dev
autoconf automake automake1.9 dpatch ghostscript-x libfontenc1 libjpeg62 libltdl-dev libltdl7 libmail-
sendmail-perl libxfont1 lmodern texlive-latex-base-doc ttf-dejavu ttf-dejavu-extra libaio-dev xfonts-
encodings xfonts-utils libxml2-dev unixodbc-dev bzr scons check libboost-all-dev openssl epm" \
    "bzr co --lightweight lp:mariadb-native-client" \
    "cd /usr/local/src;sudo tar zxf /dev/shm/thrift-0.9.0.tar.gz;pwd;ls" \
    "cd /usr/local/src/thrift-0.9.0;echo;pwd;sudo ./configure --prefix=/usr --enable-shared=no --enable-
static=yes CXXFLAGS=-fPIC CFLAGS=-fPIC && echo && echo 'now making' && echo && sleep 3 && sudo make &&
echo && echo 'now installing' && echo && sleep 3 && sudo make install" ; \
done
```

VMs for install testing.

See Buildbot Setup for Virtual Machines - General Principles for how to obtain `my.seed` and `sources.append`.

```
for i in '/kvm/vms/vm-saucy-amd64-serial.qcow2 2287 qemu64' '/kvm/vms/vm-saucy-i386-serial.qcow2 2288
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils unixodbc libxml2"
\
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-saucy-amd64-serial.qcow2 2287 qemu64' '/kvm/vms/vm-saucy-i386-serial.qcow2 2288
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.5' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ;\
done
```

VMs for MariaDB upgrade testing

```
for i in '/kvm/vms/vm-saucy-amd64-serial.qcow2 2287 qemu64' '/kvm/vms/vm-saucy-i386-serial.qcow2 2288
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "= scp -P $2 /kvm/vms/mariadb-saucy.list buildbot@localhost:/tmp/tmp.list" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    'sudo mv -vi /tmp/tmp.list /etc/apt/sources.list.d/' \
    'sudo apt-key adv --recv-keys --keyserver pgp.mit.edu 0xcbcb082a1bb943db' \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mariadb-server' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    'sudo rm -v /etc/apt/sources.list.d/tmp.list' \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get update' \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils' \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y'; \
done
```

Add Key to known_hosts

Do the following on each kvm host server (terrier, terrier2, i7, etc…) to add the VMs to known_hosts.

```
# saucy-amd64
cp -avi /kvm/vms/vm-saucy-amd64-install.qcow2 /kvm/vms/vm-saucy-amd64-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-saucy-amd64-test.qcow2 -redir tcp:2287::22 -boot c -smp 2 -cpu qemu64 -net
nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2287 buildbot@localhost sudo shutdown -h now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-saucy-amd64-test.qcow2

# saucy-i386
cp -avi /kvm/vms/vm-saucy-i386-install.qcow2 /kvm/vms/vm-saucy-i386-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-saucy-i386-test.qcow2 -redir tcp:2288::22 -boot c -smp 2 -cpu qemu64 -net
nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2288 buildbot@localhost sudo shutdown -h now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-saucy-i386-test.qcow2
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 14.04 “trusty”

Base Install

```
qemu-img create -f qcow2 /kvm/vms/vm-trusty-amd64-serial.qcow2 20G
qemu-img create -f qcow2 /kvm/vms/vm-trusty-i386-serial.qcow2 20G
```

Start each VM booting from the server install iso one at a time and perform the following install steps:

```
kvm -m 2048 -hda /kvm/vms/vm-trusty-amd64-serial.qcow2 \
  -cdrom /kvm/iso/ubuntu/trusty-server-amd64.iso \
  -boot d -smp 2 -cpu qemu64 \
  -net nic,model=virtio \
  -net user,hostfwd=tcp:127.0.0.1:2293-:22

kvm -m 2048 -hda /kvm/vms/vm-trusty-i386-serial.qcow2 \
  -cdrom /kvm/iso/ubuntu/trusty-server-i386.iso \
  -boot d -smp 2 -cpu qemu64 \
  -net nic,model=virtio \
  -net user,hostfwd=tcp:127.0.0.1:2294-:22
```

Once running you can connect to the VNC server from your local host with:

```
vncviewer -via ${remote_host} localhost
```

Replace ${remote-host} with the host the vm is running on.

Note: When you activate the install, vncviewer may disconnect with a complaint about the rect being too large. This is fine. Ubuntu has just resized the vnc screen. Simply reconnect.

During the install, pick default options for the most part, with the following notes:

- Set the hostname to `ubuntu-trusty-amd64` or `ubuntu-trusty-i386` (depending on which architecture we're installing)
- do not encrypt the home directory
- When prompted if the timezone is correct, answer "No" and when the list comes up, scroll to the bottom and choose "UTC"
- When partitioning disks, choose "Guided - use entire disk" (we do not want LVM)
- No automatic updates
- Choose software to install: OpenSSH server

Reboot when prompted. It will fail, so just kill the kvm process with `^Ctrl+C`.

Now that the VM is installed, it's time to configure it.
If you have the memory you can do the following simultaneously:

```
qemu-system-x86_64 -m 2048 -hda /kvm/vms/vm-trusty-amd64-serial.qcow2 \
  -boot c -smp 2 -cpu qemu64 \
  -net nic,model=virtio \
  -net user,hostfwd=tcp:127.0.0.1:2293-:22 \
  -nographic

qemu-system-x86_64 -m 2048 -hda /kvm/vms/vm-trusty-i386-serial.qcow2 \
  -boot c -smp 2 -cpu qemu64 \
  -net nic,model=virtio \
  -net user,hostfwd=tcp:127.0.0.1:2294-:22 \
  -nographic
```

- set `vim` as the preferred editor

```
ssh -t -p 2293 localhost sudo update-alternatives --config editor
ssh -t -p 2294 localhost sudo update-alternatives --config editor
```

- Enabling passwordless sudo:

```
sudo VISUAL=vi visudo
# Add line at end: `%sudo ALL=NOPASSWD: ALL'
```

- edit /boot/grub/menu.lst:

```
sudo vi /etc/default/grub

# Add/edit these entries:
GRUB_HIDDEN_TIMEOUT_QUIET=false
GRUB_TIMEOUT=0
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
# exit back to the host server
```

- create the `.ssh` directory and the `sudo` group

```
ssh -t -p 2293 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"
ssh -t -p 2294 localhost "mkdir -v .ssh; sudo addgroup $USER sudo"
```

- copy the `authorized_keys` file up to the VMs

```
scp -P 2293 /kvm/vms/authorized_keys localhost:.ssh/
scp -P 2294 /kvm/vms/authorized_keys localhost:.ssh/
```

- create the buildbot user on the amd64 VM

```
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2293 localhost 'chmod -vR go-rwx .ssh; sudo adduser --disabled-password buildbot; sudo addgroup buildbot sudo; sudo mkdir -v ~buildbot/.ssh; sudo cp -vi .ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-rwx ~buildbot/.ssh'
```

- create the buildbot user on the x86 VM

```
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2294 localhost 'chmod -vR go-rwx .ssh; sudo adduser --disabled-password buildbot; sudo addgroup buildbot sudo; sudo mkdir -v ~buildbot/.ssh; sudo cp -vi .ssh/authorized_keys ~buildbot/.ssh/; sudo chown -vR buildbot:buildbot ~buildbot/.ssh; sudo chmod -vR go-rwx ~buildbot/.ssh'
```

- copy the ttyS0 file to the VMs

```
scp -i -P 2293 /kvm/vms/ttyS0.conf buildbot@localhost:
scp -i -P 2294 /kvm/vms/ttyS0.conf buildbot@localhost:
```

- Apply all OS updates

```
ssh -t -i -p 2293 buildbot@localhost \
  'sudo apt-get update && sudo apt-get -y dist-upgrade;'

ssh -t -i -p 2294 buildbot@localhost \
  'sudo apt-get update && sudo apt-get -y dist-upgrade;'
```

- copy the ttyS0 file to its proper location and shut down the VM

```
ssh -t -i -p 2293 buildbot@localhost \
  'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h now'

ssh -t -i -p 2294 buildbot@localhost \
  'sudo cp -vi ttyS0.conf /etc/init/; rm -v ttyS0.conf; sudo shutdown -h now'
```

VMs for building .debs

```
for i in '/kvm/vms/vm-trusty-amd64-serial.qcow2 2293 qemu64' '/kvm/vms/vm-trusty-i386-serial.qcow2 2294
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "= scp -P $2 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no /kvm/thrift-0.9.0.tar.gz
buildbot@localhost:/dev/shm/" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.5" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libpam0g-dev libreadline-gplv2-dev
autoconf automake automake1.9 dpatch ghostscript-x libfontenc1 libjpeg62 libltdl-dev libltdl7 libmail-
sendmail-perl libxfont1 lmodern texlive-latex-base-doc ttf-dejavu ttf-dejavu-extra libaio-dev xfonts-
encodings xfonts-utils libxml2-dev unixodbc-dev bzr scons check libboost-all-dev openssl epm libjudy-dev
libjemalloc-dev" \
    "bzr co --lightweight lp:mariadb-native-client" \
    "cd /usr/local/src;sudo tar zxf /dev/shm/thrift-0.9.0.tar.gz;pwd;ls" \
    "cd /usr/local/src/thrift-0.9.0;echo;pwd;sudo ./configure --prefix=/usr --enable-shared=no --enable-
static=yes CXXFLAGS=-fPIC CFLAGS=-fPIC && echo && echo 'now making' && echo && sleep 3 && sudo make &&
echo && echo 'now installing' && echo && sleep 3 && sudo make install" ; \
done
```

VMs for install testing.

See Buildbot Setup for Virtual Machines - General Principles for how to obtain `my.seed` and `sources.append`.

```
for i in '/kvm/vms/vm-trusty-amd64-serial.qcow2 2293 qemu64' '/kvm/vms/vm-trusty-i386-serial.qcow2 2294
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils unixodbc libxml2
libjudydebian1 libjemalloc1" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for MySQL upgrade testing

```
for i in '/kvm/vms/vm-trusty-amd64-serial.qcow2 2293 qemu64' '/kvm/vms/vm-trusty-i386-serial.qcow2 2294
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.5 libjemalloc1' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ;\
done
```

VMs for MariaDB upgrade testing

```
for i in '/kvm/vms/vm-trusty-amd64-serial.qcow2 2293 qemu64' '/kvm/vms/vm-trusty-i386-serial.qcow2 2294
qemu64' ; do \
  set $i; \
  runvm --user=buildbot --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/upgrade2/')" \
    "= scp -P $2 /kvm/vms/my55.seed /kvm/vms/sources.append buildbot@localhost:/tmp/" \
    "= scp -P $2 /kvm/vms/mariadb-trusty.list buildbot@localhost:/tmp/tmp.list" \
    "sudo debconf-set-selections /tmp/my55.seed" \
    'sudo mv -vi /tmp/tmp.list /etc/apt/sources.list.d/' \
    'sudo apt-key adv --recv-keys --keyserver pgp.mit.edu 0xcbcb082a1bb943db' \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mariadb-server libjemalloc1' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' \
    'sudo rm -v /etc/apt/sources.list.d/tmp.list' \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get update' \
    "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y patch libaio1 debconf-utils' \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y'; \
done
```

Add Key to known_hosts

Do the following on each kvm host server (terrier, terrier2, i7, etc…) to add the VMs to known_hosts.

```
# trusty-amd64
cp -avi /kvm/vms/vm-trusty-amd64-install.qcow2 /kvm/vms/vm-trusty-amd64-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-trusty-amd64-test.qcow2 -redir tcp:2293::22 -boot c -smp 2 -cpu qemu64 -net
nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2293 buildbot@localhost sudo shutdown -h now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-trusty-amd64-test.qcow2

# trusty-i386
cp -avi /kvm/vms/vm-trusty-i386-install.qcow2 /kvm/vms/vm-trusty-i386-test.qcow2
kvm -m 1024 -hda /kvm/vms/vm-trusty-i386-test.qcow2 -redir tcp:2294::22 -boot c -smp 2 -cpu qemu64 -net
nic,model=virtio -net user -nographic
sudo su - buildbot
ssh -p 2294 buildbot@localhost sudo shutdown -h now
# answer "yes" when prompted
exit # the buildbot user
rm -v /kvm/vms/vm-trusty-i386-test.qcow2
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 8.04, 8.10, and 9.10

These 6 platforms were installed together (i386 and amd64).

Base Installations

Ubuntu 8.04 “Hardy” amd64:

```
qemu-img create -f qcow2 vm-hardy-amd64-serial.qcow2 8G
kvm -m 1024 -hda vm-hardy-amd64-serial.qcow2 -cdrom /kvm/ubuntu-8.04.3-server-amd64.iso -redir
tcp:2228::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-hardy-amd64-serial.qcow2 -cdrom /kvm/ubuntu-8.04.3-server-amd64.iso -redir
tcp:2228::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
ssh -p 2228 localhost
# edit /boot/grub/menu.lst and visudo, see below
sudo /sbin/shutdown -h now
kvm -m 1024 -hda vm-hardy-amd64-serial.qcow2 -cdrom /kvm/ubuntu-8.04.3-server-amd64.iso -redir
tcp:2228::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
ssh -t -p 2228 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2228 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2228 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2228 ttyS0 buildbot@localhost:
ssh -p 2228 buildbot@localhost 'sudo cp ttyS0 /etc/event.d; rm ttyS0; sudo shutdown -h now'
```

Ubuntu 8.04 “Hardy” i386:

```
qemu-img create -f qcow2 vm-hardy-i386-serial.qcow2 8G
kvm -m 1024 -hda vm-hardy-i386-serial.qcow2 -cdrom /kvm/ubuntu-8.04.3-server-i386.iso -redir tcp:2229::22
-boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-hardy-i386-serial.qcow2 -cdrom /kvm/ubuntu-8.04.3-server-i386.iso -redir tcp:2229::22
-boot c -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user -nographic
ssh -p 2229 localhost
# edit /boot/grub/menu.lst and visudo, see below
ssh -t -p 2229 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2229 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2229 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2229 ttyS0 buildbot@localhost:
ssh -p 2229 buildbot@localhost 'sudo cp ttyS0 /etc/event.d; rm ttyS0; sudo shutdown -h now'
```

Ubuntu 8.10 “Intrepid” amd64:

```
qemu-img create -f qcow2 vm-intrepid-amd64-serial.qcow2 8G
kvm -m 1024 -hda vm-intrepid-amd64-serial.qcow2 -cdrom /kvm/ubuntu-8.10-server-amd64.iso -redir
tcp:2230::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-intrepid-amd64-serial.qcow2 -cdrom /kvm/ubuntu-8.10-server-amd64.iso -redir
tcp:2230::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
ssh -p 2230 localhost
# edit /boot/grub/menu.lst and visudo, see below
ssh -t -p 2230 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2230 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2230 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2230 ttyS0 buildbot@localhost:
ssh -p 2230 buildbot@localhost 'sudo cp ttyS0 /etc/event.d; rm ttyS0; sudo shutdown -h now'
```

Ubuntu 8.10 “Intrepid” i386:

```
qemu-img create -f qcow2 vm-intrepid-i386-serial.qcow2 8G
kvm -m 1024 -hda vm-intrepid-i386-serial.qcow2 -cdrom /kvm/ubuntu-8.10-server-i386.iso -redir
tcp:2231::22 -boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-intrepid-i386-serial.qcow2 -cdrom /kvm/ubuntu-8.10-server-i386.iso -redir
tcp:2231::22 -boot c -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user -nographic
ssh -p 2231 localhost
# edit /boot/grub/menu.lst and visudo, see below
ssh -t -p 2231 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2231 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2231 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2231 ttyS0 buildbot@localhost:
ssh -p 2231 buildbot@localhost 'sudo cp ttyS0 /etc/event.d; rm ttyS0; sudo shutdown -h now'
```

Ubuntu 9.10 “Karmic” amd64:

```
qemu-img create -f qcow2 vm-karmic-amd64-serial.qcow2 8G
kvm -m 1024 -hda vm-karmic-amd64-serial.qcow2 -cdrom /kvm/ubuntu-9.10-server-amd64.iso -redir
tcp:2232::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-karmic-amd64-serial.qcow2 -cdrom /kvm/ubuntu-9.10-server-amd64.iso -redir
tcp:2232::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic
ssh -p 2232 localhost
# edit /boot/grub/menu.lst and visudo, see below
ssh -t -p 2232 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2232 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2232 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2232 ttyS0.conf buildbot@localhost:
ssh -p 2232 buildbot@localhost 'sudo cp ttyS0.conf /etc/init/; rm ttyS0.conf; sudo shutdown -h now'
```

Ubuntu 9.10 “Karmic” i386:

```
qemu-img create -f qcow2 vm-karmic-i386-serial.qcow2 8G
kvm -m 1024 -hda vm-karmic-i386-serial.qcow2 -cdrom /kvm/ubuntu-9.10-server-i386.iso -redir tcp:2233::22
-boot d -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user
# Install, picking default options mostly, only adding openssh server.
kvm -m 1024 -hda vm-karmic-i386-serial.qcow2 -cdrom /kvm/ubuntu-9.10-server-i386.iso -redir tcp:2233::22
-boot c -smp 2 -cpu qemu32,-nx -net nic,model=virtio -net user -nographic
ssh -p 2233 localhost
# edit /boot/grub/menu.lst and visudo, see below
ssh -t -p 2233 localhost "mkdir .ssh; sudo addgroup $USER sudo"
scp -P 2233 authorized_keys localhost:.ssh/
echo $'Buildbot\n\n\n\n\ny' | ssh -p 2233 localhost 'chmod -R go-rwx .ssh; sudo adduser --disabled-
password buildbot; sudo addgroup buildbot sudo; sudo mkdir ~buildbot/.ssh; sudo cp .ssh/authorized_keys
~buildbot/.ssh/; sudo chown -R buildbot:buildbot ~buildbot/.ssh; sudo chmod -R go-rwx ~buildbot/.ssh'
scp -P 2233 ttyS0.conf buildbot@localhost:
ssh -p 2233 buildbot@localhost 'sudo cp ttyS0.conf /etc/init/; rm ttyS0.conf; sudo shutdown -h now'
```

For visudo to enable passwordless sudo:

```
sudo VISUAL=vi visudo

# uncomment `%sudo ALL=NOPASSWD: ALL' line in `visudo`, and move to end.
```

For 8.04/8.10/9.04, /boot/grub/menu.lst is edited as follows:

```
sudo vi /boot/grub/menu.lst

# Add this:
  serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
  terminal --timeout=3 serial console

# Also add in menu.lst to kernel line (after removing `quiet splash'):
  console=tty0 console=ttyS0,115200n8
```

For 9.10, /boot/grub/menu.lst is edited as follows:

```
sudo vi /etc/default/grub

# Add/edit these entries:
    GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
    GRUB_TERMINAL="serial"
    GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"

sudo update-grub
```

VMs for building .debs

```
for i in 'vm-hardy-amd64-serial.qcow2 2228 qemu64' 'vm-hardy-i386-serial.qcow2 2229 qemu32,-nx' 'vm-
intrepid-amd64-serial.qcow2 2230 qemu64' 'vm-intrepid-i386-serial.qcow2 2231 qemu32,-nx' ; do \
  set $i; \
  runvm --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get build-dep -y mysql-server-5.0" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libreadline5-dev" ; \
done

for i in 'vm-karmic-amd64-serial.qcow2 2232 qemu64' 'vm-karmic-i386-serial.qcow2 2233 qemu32,-nx' ; do \
  set $i; \
  runvm --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/build/')" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get -y build-dep mysql-server-5.1" \
    "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y devscripts hardening-wrapper fakeroot doxygen
texlive-latex-base ghostscript libevent-dev libssl-dev zlib1g-dev libreadline5-dev" ; \
done
```

VMs for install testing

See the General Principles article for how to make the '`my.seed`' and '`sources.append`' files.

```
for i in 'vm-hardy-amd64-serial.qcow2 2228 qemu64' 'vm-hardy-i386-serial.qcow2 2229 qemu32,-nx' 'vm-
intrepid-amd64-serial.qcow2 2230 qemu64' 'vm-intrepid-i386-serial.qcow2 2231 qemu32,-nx' 'vm-karmic-
amd64-serial.qcow2 2232 qemu64' 'vm-karmic-i386-serial.qcow2 2233 qemu32,-nx' ; do \
  set $i; \
  runvm --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/serial/install/')" \
      "sudo DEBIAN_FRONTEND=noninteractive apt-get update" \
      "sudo DEBIAN_FRONTEND=noninteractive apt-get install -y debconf-utils" \
      "= scp -P $2 my.seed sources.append buildbot@localhost:/tmp/" \
      "sudo debconf-set-selections /tmp/my.seed" \
      "sudo sh -c 'cat /tmp/sources.append >> /etc/apt/sources.list'"; \
done
```

VMs for upgrade testing

```
for i in 'vm-hardy-amd64-install.qcow2 2228 qemu64' 'vm-hardy-i386-install.qcow2 2229 qemu32,-nx' 'vm-
intrepid-amd64-install.qcow2 2230 qemu64' 'vm-intrepid-i386-install.qcow2 2231 qemu32,-nx' ; do \
  set $i; \
  runvm --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/install/upgrade/')" \
  'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.0'
  'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ; \
done

for i in 'vm-karmic-amd64-install.qcow2 2232 qemu64' 'vm-karmic-i386-install.qcow2 2233 qemu32,-nx' ; do
\
  set $i; \
  runvm --logfile=kernel_$2.log --base-image=$1 --port=$2 --cpu=$3 "$(echo $1 | sed -e
's/install/upgrade/')" \
    'sudo DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server-5.1' \
    'mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key);
insert into t values (1); select * from t"' ; \
done
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 9.04 amd64

Base Install

```
qemu-img create -f qcow2 vm-jaunty-amd64-serial.qcow2 8G
kvm -m 1024 -hda vm-jaunty-amd64-serial.qcow2 -cdrom /kvm/ubuntu-9.04-server-amd64.iso -redir
tcp:2227::22 -boot d -smp 2 -cpu qemu64 -net nic,model=virtio -net user
```

Serial port and passwordless login/sudo

```
kvm -m 1024 -hda vm-jaunty-amd64-serial.qcow2 -cdrom /kvm/ubuntu-9.04-server-amd64.iso -redir
tcp:2227::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user

sudo apt-get install emacs22-nox
```

Add to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line (after removing `quiet splash'):

```
console=tty0 console=ttyS0,115200n8
```

Create /etc/event.d/ttyS0:

```
# ttyS0 - getty
#
# This service maintains a getty on ttyS0 from the point the system is
# started until it is shut down again.

start on stopped rc2
start on stopped rc3
start on stopped rc4
start on stopped rc5

stop on runlevel 0
stop on runlevel 1
stop on runlevel 6

respawn
exec /sbin/getty 115200 ttyS0
```

Add account and allow passwordless sudo:

```
sudo adduser --disabled-password buildbot
sudo adduser buildbot sudo
sudo EDITOR=emacs visudo
# uncomment `%sudo ALL=NOPASSWD: ALL' line in `visudo`, and move to end.

sudo su -s /bin/bash - buildbot
mkdir .ssh
cat >.ssh/authorized_keys
# Paste public key for buildbot user on host system.
chmod -R go-rwx .ssh
# Do manual login from host to guest once, to make sure .ssh/known_hosts is updated.
```

VM for .deb building

```
qemu-img create -b vm-jaunty-amd64-serial.qcow2 -f qcow2 vm-jaunty-amd64-build.qcow2
kvm -m 1024 -hda vm-jaunty-amd64-build.qcow2 -cdrom /kvm/ubuntu-9.04-server-amd64.iso -redir tcp:2227::22
-boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic

sudo apt-get build-dep mysql-server-5.1
```

VM for install testing

```
qemu-img create -b vm-jaunty-amd64-serial.qcow2 -f qcow2 vm-jaunty-amd64-install.qcow2
kvm -m 1024 -hda vm-jaunty-amd64-install.qcow2 -cdrom /kvm/ubuntu-9.04-server-amd64.iso -redir
tcp:2227::22 -boot c -smp 2 -cpu qemu64 -net nic,model=virtio -net user -nographic

sudo adduser --system --group mysql
```

This page is licensed: CC BY-SA / Gnu FDL

Buildbot Setup for Virtual Machines - Ubuntu 9.04 i386

This vm is used to build source tarball and for Ubuntu 9.04 32-bit .deb.

First create and install image:

```
qemu-img create -f qcow2 vm-jaunty-i386-base.qcow2 8G
kvm -m 2047 -hda vm-jaunty-i386-base.qcow2 -cdrom ubuntu-9.04-server-i386.iso -redir tcp:2222::22 -boot d
-cpu qemu32,-nx
# Install
```

Create a vm, based on the first one, which is configured for serial port.

```
qemu-img create -b vm-jaunty-i386-base.qcow2 -f qcow2 vm-jaunty-i386-serial.qcow2
kvm -m 2047 -hda vm-jaunty-i386-base.qcow2 -redir tcp:2222::22 -boot c -cpu qemu32,-nx
```

To configure kernel and grub for serial console, add the following to /boot/grub/menu.lst:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=3 serial console
```

also add in menu.lst to kernel line (after removing `quiet splash'):

```
console=tty0 console=ttyS0,115200n8
```

Create /etc/event.d/ttyS0:

```
# ttyS0 - getty
#
# This service maintains a getty on ttyS0 from the point the system is
# started until it is shut down again.

start on stopped rc2
start on stopped rc3
start on stopped rc4
start on stopped rc5

stop on runlevel 0
stop on runlevel 1
stop on runlevel 6

respawn
exec /sbin/getty 115200 ttyS0
```

Add an account:

```
sudo adduser --disabled-password buildbot
```

Copy public ssh key into /.ssh/authorized_keys

Enable passwordless sudo:

```
sudo adduser buildbot sudo
# uncomment `%sudo ALL=NOPASSWD: ALL' line in `visudo`, and move to end.
```

Create a new vm for building source tarballs:

```
qemu-img create -b vm-jaunty-i386-serial.qcow2 -f qcow2 vm-jaunty-i386-deb-tarbake.qcow2
```

Install compilers:

```
sudo apt-get build-dep mysql-5.1-server
sudo apt-get install devscripts hardening-wrapper bzr
```

Copy in an existing bzr shared repository to buildbot/.bzr (or run bzr init-repo and bzr branch –no-tree lp:maria).

Create a new vm for building .debs:

```
qemu-img create -b vm-jaunty-i386-serial.qcow2 -f qcow2 vm-jaunty-i386-deb-build.qcow2
```

Install compilers:

```
sudo apt-get build-dep mysql-5.1-server
sudo apt-get install devscripts hardening-wrapper
```

Create a new VM for testing .deb installs:

```
qemu-img create -b vm-jaunty-i386-serial.qcow2 -f qcow2 vm-jaunty-i386-deb-install.qcow2
```

Install tools and local apt repository.

```
sudo apt-get install debconf-utils
cat >>/etc/apt/sources.list <<END
deb file:///home/buildbot/buildbot/debs binary/
deb-src file:///home/buildbot/buildbot/debs source/
END
```

Setup default package config for debconf to enable unattended install. Copy in my.seed (see above) to vm.

```
sudo debconf-set-selections /tmp/my.seed
```

Create a new VM for testing upgrades:

```
qemu-img create -b vm-jaunty-i386-deb-install.qcow2 -f qcow2 vm-jaunty-i386-deb-upgrade.qcow2
```

Prepare initial MySQL install with some test data.

```
sudo apt-get install mysql-server-5.1
mysql -uroot -prootpass -e "create database mytest; use mytest; create table t(a int primary key); insert
into t values (1); select * from t"
```

This page is licensed: CC BY-SA / Gnu FDL

Using Git With MariaDB

See Using Git.

See Configuring Git to Send Commit Notices.

This page is licensed: CC BY-SA / Gnu FDL

Using Git

Getting the Source

If you just want to get the latest source and don't require the ability to push changes to the MariaDB repository, you can use the following command to check out the latest branch (replace `10.5` with `latest` or the latest branch version):

```
git clone -b 10.5 https://github.com/MariaDB/server.git
```

Setup of git for MariaDB

- Set your name with git

```
git config --global user.name "Ivan Ivanov"
git config --global user.email "ivan@mariadb.com"
```

- Go to ssh and add your public SSH key (GitHub Help).
- Clone the repository

```
git clone git@github.com:MariaDB/server.git
cd server
git checkout 10.5
```

- Config repository pull and alias for fast forward:

```
git config pull.ff only
git config --global alias.ff "merge --ff-only"
```

Commit Comments

In git commit messages are normally formatted like this:

```
subject

body
more body
...
```

That is, the first line is considered a *subject*, much like email subject. Many git commands and pages on GitHub only show the commit subject, not the complete comment. After the subject goes an empty line, then the detailed description of the comment. Structure your commit comments this way, and don't forget the empty line.

Branches

In Git, each repository has only one branch that is currently checked out.

```
git branch
```

List Existing Branches

To see which branches exists locally and remotely:

```
git branch --all
```

To Move to Work on a Specific Branch

```
git checkout branch-name
```

Note that if the output from `git branch --all` is `remotes/origin/XXX` you should just use `XXX` as branch name.

Making a Local Copy of a Branch (Like bzr clone)

```
branch clone old_directory new_directory
cd new_directory
git remote set-url origin git@github.com:MariaDB/server.git
git pull
```

Remove Last Commit From a Branch

```
git reset --hard HEAD^
```

Fetch a Branch Someone Has Done a Rebase on

If you get the following error on pull:

```
shell>  git pull
X11 forwarding request failed on channel 0
fatal: Not possible to fast-forward, aborting.
```

Instead of removing your copy and then clone, you can do:

```
git reset --hard origin/##branch-name##
```

Other Things About Branches

- Note: branches whose names start with `bb-` are automatically put into the buildbot.

Commit Emails

In the MariaDB project, it is a good idea (and a long tradition since MySQL Ab) to have all your commits sent to a commits@mariadb.org mailing list. It allows others to follow the progress, comment, etc.

A script and instructions on how to setup commit triggers in Git are here:. Jira task for commit trigger was MDEV-6278.

Attributing Code to Someone Else

If you add code on behalf of someone else, please attribute the code to the original author:

- Run `git citool` and move changed files to staged.
- Don't `commit`, abort instead
- run `git commit --author="Original author name <email_address>"`

The above is needed as `git citool` can't handle the `--author` option.

Applying a Pull Request

At the end of the pull request page there is a button "Merge pull request" and next to it a link "command line instructions". Click the link, you'll see something like

Step 1: From your project repository, check out a new branch and test the changes.

```
git checkout -b mariadb-server-joeuser-cool-feature 10.3
git pull https://github.com/joeuser/mariadb-server cool-feature
```

Step 2: Merge the changes and update on GitHub.

```
git checkout 10.3
git merge --no-ff mariadb-server-joeuser-cool-feature
git push origin 10.3
```

Note where to pull from.

Now, checkout the branch you want to merge it to, say, `bb-10.3-stage`, and do the following

```
git fetch https://github.com/joeuser/mariadb-server cool-feature
git checkout FETCH_HEAD
git rebase @{-1}
```

Now's the time to compile the code, test it, fix, if necessary. Issue these commands:

```
git checkout @{-1}
git ff @{-1}
```

If you want to do small changes to the pull request, do it in a separate commit, after `git rebase @{-1}` above. If you want to do *big* changes to the pull request, perhaps you shouldn't merge it in the first place, but ask the contributor to fix it?

Examples

Diff For Last Commit

```
git show
```

Applying New Code From Main Tree to a Branch

You are working on a branch (`NEW_FEATURE`) and would like to have into that branch all changes from the main branch (`10.1`).

```
git checkout 10.1
git pull
git checkout NEW_FEATURE
git rebase 10.1
```

Applying a Bugfix in the Main Branch

You've just fixed and committed a bug in the main 10.1 branch and want to merge it with the latest 10.1. Often a rebase is better in this case. Assuming your current checked out branch is 10.1:

```
git fetch origin
git rebase origin/10.1
```

This will work even if you have done multiple commits in your local 10.1 tree.

Dealing With Conflicts When One Tries to Push

What to do when you have fixed a bug in the main tree but notices that someone has changed the tree since you pulled last time. This approach ensures that your patch is done in one block and not spread out over several change sets.

```
git clone 10.1
cd 10.1
< fix a bug here>
git citool
git push
# ^ and the above fails, because someone has pushed to 10.1 in between
git branch tmp
# ^ copy our work to branch named 'tmp'
get checkout 10.1
git reset --hard HEAD^
# ^ remove our work from '10.1' local branch'
git pull
# ^ get changes from remote
git checkout tmp
git rebase 10.1
# ^ switch to 'tmp' and try to rebase 'tmp' branch on top of 10.1 branch.
#   here you will be asked to merge if necessary
git checkout 10.1
git pull --ff . tmp
# ^ switch back to the '10.1' branch, and pull from 'tmp' branch.
git branch -D tmp
#^ this removes the tmp. branch
git push
```

Finding in Which MariaDB Version a Commit Exists

```
sh> git tag --contain e65f667bb60244610512efd7491fc77eccceb9db
mariadb-10.0.30
mariadb-10.1.22
mariadb-10.1.23
mariadb-10.2.5
mariadb-10.3.0
mariadb-galera-10.0.30
```

See Also

- git-for-bzr-users.pdf
- Sergei's "Using GIT" presentation. Malaga Meeting, January 2015 "Sergei's"Using GIT" presentation. Malaga Meeting, January 2015"

This page is licensed: CC BY-SA / Gnu FDL

Configuring Git to Send Commit Notices

Commit emails for MariaDB are sent to commits@lists.mariadb.org. You can find the archive here.

To allow others to see what you are working on in your MariaDB tree:

1. subscribe to the email list
2. configure git to send your commits to commits@mariab.org.

Download the post-commit git trigger script. Configure as:

```
git config --global hooks.postcommitrecipients "commits@mariadb.org"
git config --global hooks.postcommitbranches "*"
```

Also you might want to check the README for the post-commit trigger.

The post-commit git trigger uses *sendmail* for sending emails. Some platforms don't have *sendmail* and then you'll need to modify to make use of something that is supported.

Also, the post-commit trigger is just one approach. You can also use git-email on at least Debian and Fedora to send commit emails to the MariaDB commits email list.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Internals

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See Merging into MariaDB.

Serves as the primary category overview explaining the processes and strategies used to merge various source trees into the MariaDB codebase.

See MariaDB Source Code Internals.

Serves as the main landing page for articles detailing the low-level source code architecture and internal logic of the MariaDB server.

See Using MariaDB with Your Programs (API).

This main page serves as a landing hub for developers looking to integrate MariaDB functionality into external applications using internal APIs.

This page is licensed: CC BY-SA / Gnu FDL

Merging into MariaDB

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See Creating a New Merge Tree.

See Merging with a Merge Tree.

This page is licensed: CC BY-SA / Gnu FDL

Creating a New Merge Tree

> **Note:**
>
> This page contains background information for MariaDB developers.

A **merge tree** is a branch created specifically to simplify merges of third-party packages into MariaDB. It contains nothing but pristine upstream sources, laid out at the same paths they occupy in the MariaDB tree. That gives a clear separation between upstream changes and MariaDB changes, so that in most cases Git can merge new upstream releases automatically.

MariaDB merge trees live in the mergetrees repository. Each third-party package has its own branch, named `merge-<package>` — for example `merge-pcre`, `merge-zlib`, or `merge-innodb-5.7`. The branches are independent of each other and share no history.

Create a merge tree once per package. Afterwards, updating it for each new upstream release is a separate and much shorter procedure — see Merging with a Merge Tree.

Prerequisites

The package should already be present in the MariaDB tree, together with any MariaDB changes to it. You also need to know which upstream release corresponds to the copy currently in the tree, because the merge tree must start from exactly that version. Starting from a different version makes the first real merge produce spurious conflicts.

Creating the Branch

Clone the `mergetrees` repository and create an orphan branch for the package. An orphan branch starts with no history, which is what you want — the merge tree is not a variation of any other merge tree.

```
git clone https://github.com/MariaDB/mergetrees
cd mergetrees
git checkout --orphan merge-pcre
git rm -rf .
```

Adding the Upstream Sources

Download the upstream source tarball of the same version that is already in the MariaDB tree, and unpack it so that the files sit at **the same paths they have in the MariaDB tree**. This layout is what lets Git match the files up during a merge, so it is the step that matters most.

```
tar xf ~/pcre-8.34.tar.bz2
mv pcre-8.34 pcre
```

Then commit, using the upstream version as the commit message, and push the branch:

```
git add .
git commit -m 'pcre-8.34'
git push --set-upstream origin merge-pcre
```

Recording the Merge Base

The new branch now has to be connected to the MariaDB tree, so that later merges bring across only the upstream changes since this point. Do that with a **null merge** — a merge that records the ancestry without changing a single file.

In your MariaDB tree, add the merge tree repository as a remote:

```
git remote add merge https://github.com/MariaDB/mergetrees
git fetch merge
```

Then merge the new branch with the `ours` strategy, which keeps the MariaDB tree exactly as it is:

```
git merge --allow-unrelated-histories -s ours merge/merge-pcre
```

> **Note:**
>
> `-s ours` is what makes this a null merge: the resulting commit has two parents but its content is identical to the MariaDB tree before the merge. `--allow-unrelated-histories` is required because the merge tree branch is an orphan.

Verify that nothing changed, then push:

```
git diff HEAD^1 --stat
```

The command produces no output if the merge was correctly a null merge. Use a commit message that identifies the package and the version, following the existing convention:

```
pcre-8.34 mergetree initial merge
```

The merge tree is now ready. From here on, use Merging with a Merge Tree to pull in each new upstream release.

This page is licensed: CC BY-SA / Gnu FDL

Merging with a Merge Tree

> **Note:**
>
> This page contains background information for MariaDB developers.

If you have a merge tree, you merge into MariaDB as follows:

1. MariaDB merge trees are in the mergetrees repository. Add it as a new remote:

```
git remote add merge https://github.com/MariaDB/mergetrees
```

1. Check out the branch you want to update and merge, for example:

```
git checkout merge-innodb-5.6
```

1. delete everything in the branch
2. download the latest released source tarball, unpack it, copy files into the repository:

- for InnoDB-5.6: use the content of the `storage/innobase/` of the latest MySQL 5.6 source release tarball.
- for performance schema 5.6: use `storage/perfschema`, `include/mysql/psi`, `mysql-test/suite/perfschema`, and `mysql-test/suite/perfschema_stress` from the latest MySQL 5.6 source release tarball.
- for SphinxSE: use `mysqlse/` subdirectory from the latest Sphinx source release tarball.
- for XtraDB: use the content of the `storage/innobase/` of the latest Percona-Server source release tarball (5.5 or 5.6 as appropriate).
- for pcre: simply unpack the latest pcre release source tarball into the repository, rename `pcre-X-XX/` to `pcre`.

1. Now `git add .`, `git commit` (use the tarball version as a comment), `git push`
2. merge this branch into MariaDB
3. Sometimes after a merge, some changes may be needed:

- for performance schema 5.6: update `storage/perfschema/ha_perfschema.cc`, plugin version under `maria_declare_plugin`.
- for InnoDB-5.6: update `storage/innobase/include/univ.i`, setting `INNODB_VERSION_MAJOR`, `INNODB_VERSION_MINOR`, `INNODB_VERSION_BUGFIX` to whatever MySQL version you were merging from.
- for XtraDB-5.5: update `storage/xtradb/include/univ.i`, setting `PERCONA_INNODB_VERSION`, `INNODB_VERSION_STR` to whatever Percona-Server version you were merging from.
- for XtraDB-5.6: update `storage/xtradb/include/univ.i`, setting `PERCONA_INNODB_VERSION`, `INNODB_VERSION_MAJOR`, `INNODB_VERSION_MINOR`, `INNODB_VERSION_BUGFIX` to whatever Percona-Server version you were merging from.

This page is licensed: CC BY-SA / Gnu FDL

MariaDB Source Code Internals

> **Note:**
>
> This page contains background information for MariaDB developers.

See Stored Procedure Internals.

Explores the internal implementation and execution flow of stored procedures, including how they are parsed and stored by the server.

See Connection Memory Usage.

Provides a technical breakdown of how the server allocates and manages memory for each client connection and thread.

This page is licensed: CC BY-SA / Gnu FDL

Stored Procedure Internals

> **Note:**
>
> This page contains background information for MariaDB developers.

Implementation Specification for Stored Procedures

How Parsing and Execution of Queries Work

In order to execute a query, the function `sql_parse.cc:mysql_parse()` is called, which in turn calls the parser (`yyparse()`) with an updated Lex structure as the result. `mysql_parse()` then calls `mysql_execute_command()` which dispatches on the command code (in Lex) to the corresponding code for executing that particular query.

There are three structures involved in the execution of a query which are of interest to the stored procedure implementation:

- Lex (mentioned above) is the "compiled" query, that is the output from the parser and what is then interpreted to do the actual work.
  It contains an enum value (sql_command) which is the query type, and all the data collected by the parser needed for the execution (table names, fields, values, etc).
- THD is the "run-time" state of a connection, containing all that is needed for a particular client connection, and, among other things, the
  Lex structure currently being executed.
- `Item_*`: During parsing, all data is translated into "items", objects of the subclasses of "Item", such as `Item_int`, `Item_real`, `Item_string`, etc., for basic datatypes, and also various more specialized Item types for expressions to be evaluated (`Item_func` objects).

How to Fit Stored Procedures into this Scheme

Overview of the Classes and Files for Stored Procedures

(More detailed APIs at the end of this page)

class sp_head (sp_head.{cc,h})

This contains, among other things, an array of "instructions" and the method for executing the procedure.

class sp_pcontext (sp_pcontext.{cc,h}

This is the parse context for the procedure. It's primarily used during parsing to keep track of local parameters, variables and labels, but it's also used at CALL time to find the parameters mode (IN, OUT or INOUT) and type when setting up the runtime context.

class sp_instr (sp_head.{cc,h})

This is the base class for "instructions", that is, what is generated by the parser. It turns out that we only need a minimum of 5 different sub classes:

- sp_instr_stmt
  Execute a statement. This is the "call-out" any normal SQL statement, like a SELECT, INSERT etc. It contains the Lex structure for the statement in question.
- sp_instr_set
  Set the value of a local variable (or parameter)
- sp_instr_jump
  An unconditional jump.
- sp_instr_jump_if_not
  Jump if condition is not true. It turns out that the negative test is most convenient when generating the code for the flow control constructs.
- sp_instr_freturn
  Return a value from a FUNCTION and exit.
  For condition HANDLERs some special instructions are also needed, see that section below.

class sp_rcontext (sp_rcontext.h)

This is the runtime context in the THD structure.
It contains an array of items, the parameters and local variables for the currently executing stored procedure.
This means that variable value lookup is in runtime is constant time, a simple index operation.

class Item_splocal (Item.{cc,h})

This is a subclass of Item. Its sole purpose is to hide the fact that the real Item is actually in the current frame (runtime context).
It contains the frame offset and defers all methods to the real Item in the frame. This is what the parser generates for local variables.

Utility Functions (sp.{cc,h})

This contains functions for creating, dropping and finding a stored procedure in the mysql.proc table (or the internal cache).

Parsing CREATE PROCEDURE

When parsing a CREATE PROCEDURE the parser first initializes the `sphead` and `spcont` (runtime context) fields in the Lex.
The sql_command code for the result of parsing a is `SQLCOM_CREATE_PROCEDURE`.

The parsing of the parameter list and body is relatively straightforward:

- Parameters:
  name, type and mode (IN/OUT/INOUT) is pushed to `spcont`
- Declared local variables:
  Same as parameters (mode is then IN)
- Local Variable references:
  If an identifier is found in `spcont`, an `Item_splocal` is created with the variable's frame index, otherwise an `Item_field` or `Item_ref` is created (as before).
- Statements:
  The Lex in THD is replaced by a new Lex structure and the statement, is parsed as usual. A `sp_instr_stmt` is created, containing the new
  Lex, and added to the instructions in `sphead`.
  Afterwards, the procedure's Lex is restored in THD.
- SET var:
  Setting a local variable generates a `sp_instr_set` instruction, containing the variable's frame offset, the expression (an Item), and the type.
- Flow control:
  Flow control constructs such as IF, WHILE, etc, generate a conditional and unconditional jumps in the "obvious" way, but a few notes may be required:
- Forward jumps: When jumping forward, the exact destination is not known at the time of the creation of the jump instruction. The
  1. sphead `therefore contains a list of instruction-label pairs for each forward reference. When the position later is known, the instructions in the list are updated with the correct location.`
- Loop constructs have optional labels. If a loop doesn't have a label, an anonymous label is generated to simplify the parsing.
- There are two types of CASE. The "simple" case is implemented with an anonymous variable bound to the value to be tested.

A Simple Example

Parsing the procedure:

```
CREATE PROCEDURE a(s CHAR(16))
      BEGIN
        DECLARE x INT;
        SET x = 3;
        WHILE x > 0 DO
          SET x = x-1;
          INSERT INTO db.tab VALUES (x, s);
        END WHILE;
      END
```

would generate the following structures:

```
______
      thd: |      |     _________
           | lex -+--->|         |                     ___________________
           |______|    | spcont -+------------------->| "s",in,char(16):0 |
                       | sphead -+------              |("x",in,int     :1)|
                       |_________|      |              |___________________|
                                        |
                                  ____V_____________________
                                 | m_name: "a"            |
                                 | m_defstr: "create ..."|
                                 | m_instr: ...           |
                                 |________________________|
```

Note that the contents of the `spcont` is changing during the parsing, at all times reflecting the state of the would-be runtime frame.
The `m_instr` is an array of instructions:

```
Pos.  Instruction
       0    sp_instr_set(1, '3')
       1    sp_instr_jump_if_not(5, 'x>0')
       2    sp_instr_set(1, 'x-1')
       3    sp_instr_stmt('insert into ...')
       4    sp_instr_jump(1)
       5    <end>
```

Here, '3', 'x>0', etc, represent the Items or Lex for the respective expressions or statements.

Parsing CREATE FUNCTION

Creating a function is essentially the same thing as for a PROCEDURE, with the addition that a FUNCTION has a return type and a RETURN statement, but no OUT or INOUT parameters.

The main difference during parsing is that we store the result type in the sp_head. However, there are big differences when it comes to invoking a FUNCTION. (See below.)

Storing, Caching, Dropping

As seen above, the entire definition string, including the "CREATE PROCEDURE" (or "FUNCTION") is kept. The procedure definition string is stored in the table mysql.proc with the name and type as the key, the type being one of the enum ("procedure","function").

A PROCEDURE is just stored in the mysql.proc table. A FUNCTION has an additional requirement. They will be called in expressions with the same syntax as UDFs, so UDFs and stored FUNCTIONs share the namespace. Thus, we must make sure that we do not have UDFs and FUNCTIONs with the same name (even if they are stored in different places).

This means that we can reparse the procedure as many time as we want.
The first time, the resulting Lex is used to store the procedure in the database (using the function sp.c:sp_create_procedure()).

The simplest way would be to just leave it at that, and re-read the procedure from the database each time it is called. (And in fact, that's the way the earliest implementation will work.)
However, this is not very efficient, and we can do better. The full implementation should work like this:

1. Upon creation time, parse and store the procedure. Note that we still need to parse it to catch syntax errors, but we can't check if called procedures exists for instance.
2. Upon first CALL, read from the database, parse it, and cache the resulting Lex in memory. This time we can do more error checking.
3. Upon subsequent CALLs, use the cached Lex.

Note that this implies that the Lex structure with its sphead must be reentrant, that is, reusable and shareable between different threads and calls. The runtime state for a procedure is kept in the sp_rcontext in THD.

The mechanisms of storing, finding, and dropping procedures are encapsulated in the files sp.{cc,h}.

CALLing a Procedure

A CALL is parsed just like any statement. The resulting Lex has the sql_command SQLCOM_CALL, the procedure's name and the parameters are pushed to the Lex' value_list.

sql_parse.cc:mysql_execute_command() then uses sp.cc:sp_find() to get the sp_head for the procedure (which may have been read from the database or fetched from the in-memory cache) and calls the sp_head's method execute().
Note: It's important that substatements called by the procedure do not do send_ok(). Fortunately, there is a flag in THD->net to disable this during CALLs. If a substatement fails, it will however send an error back to the client, so the CALL mechanism must return immediately and without sending an error.

The sp_head::execute() method works as follows:

1. Keep a pointer to the old runtime context in THD (if any)
2. Create a new runtime context. The information about the required size is in sp_head's parse time context.
3. Push each parameter (from the CALL's Lex->value_list) to the new context. If it's an OUT or INOUT parameter, the parameter's offset in the caller's frame is set in the new context as well.
4. For each instruction, call its execute() method.
   The result is a pointer to the next instruction to execute (or NULL) if an error occurred.
5. On success, set the new values of the OUT and INOUT parameters in the caller's frame.

USE database

Before executing the instruction we also keeps the current default database (if any). If this was changed during execution (i.e. a USE statement has been executed), we restore the current database to the original.

This is the most useful way to handle USE in procedures. If we didn't, the caller would find himself in a different database after calling a function, which can be confusing.
Restoring the database also gives full freedom to the procedure writer:

- It's possible to write "general" procedures that are independent of the actual database name.
- It's possible to write procedures that work on a particular database by calling USE, without having to use fully qualified table names everywhere (which doesn't help if you want to call other, "general", procedures anyway).

Evaluating Items

There are three occasions where we need to evaluate an expression:

- When setting a variable
- When calling a procedure
- When testing an expression for a branch (in IF, WHILE, etc)

The semantics in stored procedures is “call-by-value”, so we have to evaluate any “func” Items at the point of the CALL or SET, otherwise we would get a kind of “lazy” evaluation with unexpected results with respect to OUT parameters for instance.
For this the support function, `sp_head.cc:eval_func_item()` is needed.

Calling a FUNCTION

Functions don’t have an explicit call keyword like procedures. Instead, they appear in expressions with the conventional syntax “fun(arg, …)”.
The problem is that we already have User Defined Functions (UDFs) which are called the same way. A UDF is detected by the lexical analyzer (not the parser!), in the `find_keyword()` function, and returns a `UDF_*_FUNC` or `UDA_*_SUM` token with the `udf_func` object as the yylval.

So, stored functions must be handled in a similar way, and as a consequence, UDFs and functions must not have the same name.

Detecting and Parsing a FUNCTION Invocation

The existence of UDFs are checked during the lexical analysis (in sql_lex.cc:find_keyword()). This has the drawback that they must exist before they are referred to, which was ok before SPs existed, but then it becomes a problem.
The first implementation of SP FUNCTIONs will work the same way, but this should be fixed a.s.a.p. (This will required some reworking of the way UDFs are handled, which is why it’s not done from the start.)
For the time being, a FUNCTION is detected the same way, and returns the token SP_FUNC. During the parsing we only check for the *existence* of the function, we don’t parse it, since wa can’t call the parser recursively.

When encountering a SP_FUNC with parameters in the expression parser, an instance of the new Item_func_sp class is created. Unlike UDFs, we don’t have different classes for different return types, since we at this point don’t know the type.

Collecting FUNCTIONs to invoke

A FUNCTION differs from a PROCEDURE in one important aspect: Whereas a PROCEDURE is CALLed as statement by itself, a FUNCTION is invoked “on-the-fly” during the execution of *another* statement.
This makes things a lot more complicated compared to CALL:

- We can’t read and parse the FUNCTION from the mysql.proc table at the point of invocation; the server requires that all tables used are opened and locked at the beginning of the query execution.
  One “obvious” solution would be to simply push “mysql.proc” to the list of tables used by the query, but this implies a “join” with this table if the query is a select, so it doesn’t work (and we can’t exclude this table easily; since a privileged used might in fact want to search the proc table).
  Another solution would of course be to allow the opening and closing of the mysql.proc table during a query execution, but this it not possible at the present.

So, the solution is to collect the names of the referred FUNCTIONs during parsing in the lex.
Then, before doing anything else in `mysql_execute_command()`, read all functions from the database an keep them in the THD, where the function `sp_find_function()` can find them during the execution.
Note: Even with an in-memory cache, we must still make sure that the functions are indeed read and cached at this point.
The code that read and cache functions from the database must also be invoked recursively for each read FUNCTION to make sure we have *all* the functions we need.

Parsing DROP PROCEDURE/FUNCTION

The procedure name is pushed to Lex->value_list.
The sql_command code for the result of parsing a is `SQLCOM_DROP_PROCEDURE` / `SQLCOM_DROP_FUNCTION`.

Dropping is done by simply getting the procedure with the sp_find() function and calling `sp_drop()` (both in `sp.{cc,h}`).

DROP PROCEDURE/DROP FUNCTION also supports the non-standard "IF EXISTS", analogous to other DROP statements in MariaDB.

Condition and Handlers

Condition names are lexical entities and are kept in the parser context just like variables. But, condition are just "aliases" for SQLSTATE strings, or mysqld error codes (which is a non-standard extension in MySQL), and are only used during parsing.

Handlers comes in three types, CONTINUE, EXIT and UNDO. The latter is like an EXIT handler with an implicit rollback, and is currently not implemented.
The EXIT handler jumps to the end of its BEGIN-END block when finished.
The CONTINUE handler returns to the statement following that which invoked the handler.

The handlers in effect at any point is part of each thread's runtime state, so we need to push and pop handlers in the sp_rcontext during execution. We use special instructions for this:

- sp_instr_hpush_jump
  Push a handler. The instruction contains the necessary information, like which conditions we handle and the location of the handler.
  The jump takes us to the location after the handler code.
- sp_instr_hpop
  Pop the handlers of the current frame (which we are just leaving).

It might seems strange to jump past the handlers like that, but there's no extra cost in doing this, and for technical reasons it's easiest for the parser to generate the handler instructions when they occur in the source.

When an error occurs, one of the error routines is called and an error message is normally sent back to the client immediately.
Catching a condition must be done in these error routines (there are quite a few) to prevent them from doing this. We do this by calling a method in the THD's sp_rcontext (if there is one). If a handler is found, this is recorded in the context and the routine returns without sending the error message.
The execution loop (sp_head::execute()) checks for this after each statement and invokes the handler that has been found. If several errors or warnings occurs during one statement, only the first is caught, the rest are ignored.

Invoking and returning from a handler is trivial in the EXIT case.
We simply jump to it, and it will have an sp_instr_jump as its last instruction.

Calling and returning from a CONTINUE handler poses some special problems. Since we need to return to the point after its invocation, we push the return location on a stack in the sp_rcontext (this is done by the execution loop). The handler then ends with a special instruction, sp_instr_hreturn, which returns to this location.

CONTINUE handlers have one additional problem: They are parsed at the lexical level where they occur, so variable offsets will assume that it's actually called at that level. However, a handler might be invoked from a sub-block where additional local variables have been declared, which will then share the location of any local variables in the handler itself. So, when calling a CONTINUE handler, we need to save any local variables above the handler's frame offset, and restore them upon return. (This is not a problem for EXIT handlers, since they will leave the block anyway.)
This is taken care of by the execution loop and the sp_instr_hreturn instruction.

Examples

EXIT handler:

```
begin
        declare x int default 0;

        begin
          declare exit handler for 'XXXXX' set x = 1;

          (statement1);
          (statement2);
        end;
        (statement3);
      end
```

```
Pos.  Instruction
       0    sp_instr_set(0, '0')
       1    sp_instr_hpush_jump(4, 1)           # location and frame size
       2    sp_instr_set(0, '1')
       3    sp_instr_jump(6)
       4    sp_instr_stmt('statement1')
       5    sp_instr_stmt('statement2')
       6    sp_instr_hpop(1)
       7    sp_instr_stmt('statement3')
```

CONTINUE handler:

```
CREATE PROCEDURE hndlr1(val INT)
      BEGIN
        DECLARE x INT DEFAULT 0;
        DECLARE foo CONDITION FOR 1146;
        DECLARE CONTINUE HANDLER FOR foo SET x = 1;

        INSERT INTO t3 VALUES ("hndlr1", val);     # Non-existing table?
        IF x>0 THEN
          INSERT INTO t1 VALUES ("hndlr1", val);   # This instead then
        END IF;
      END|
```

```
Pos.  Instruction
       0    sp_instr_set(1, '0')
       1    sp_instr_hpush_jump(4, 2)
       2    sp_instr_set(1, '1')
       3    sp_instr_hreturn(2)                 # frame size
       4    sp_instr_stmt('insert ... t3 ...')
       5    sp_instr_jump_if_not(7, 'x>0')
       6    sp_instr_stmt('insert ... t1 ...')
       7    sp_instr_hpop(2)
```

Cursors

For stored procedures to be really useful, you want to have cursors.
MySQL doesn't yet have "real" cursor support (with API and ODBC support, allowing updating, arbitrary scrolling, etc), but a simple asensitive, non-scrolling, read-only cursor can be implemented in SPs using the class Protocol_cursor.
This class intecepts the creation and sending of results sets and instead stores it in-memory, as MYSQL_FIELDS and MYSQL_ROWS (as in the client API).

To support this, we need the usual name binding support in sp_pcontext (similar to variables and conditions) to keep track on declared cursor names, and a corresponding run-time mechanism in sp_rcontext.
Cursors are lexically scoped like everything with a body or BEGIN/END block, so they are pushed and popped as

usual (see conditions and variables above).
The basic operations on a cursor are OPEN, FETCH and CLOSE, which will each have a corresponding instruction. In addition, we need instructions to push a new cursor (this will encapsulate the LEX of the SELECT statement of the cursor), and a pop instruction:

- sp_instr_cpush
  Push a cursor to the sp_rcontext. This instruction contains the LEX for the select statement
- sp_instr_cpop
  Pop a number of cursors from the sp_rcontext.
- sp_instr_copen
  Open a cursor: This will execute the select and get the result set in a separate memroot.
- sp_instr_cfetch
  Fetch the next row from the in-memory result set. The instruction contains a list of the variables (frame offsets) to set.
- sp_instr_cclose
  Free the result set.

A cursor is a separate class, sp_cursor (defined in sp_rcontex.h) which encapsulates the basic operations used by the above instructions.
This class contains the LEX, Protocol_cursor object, and its memroot, as well as the cursor's current state.
Compiling and executing is fairly straight-forward. sp_instr_copen is a subclass of sp_instr_stmt and uses its mechanism to execute a substatement.

Example

```
begin
        declare x int;
        declare c cursor for select a from t1;

        open c;
        fetch c into x;
        close c;
      end
```

```
Pos.  Instruction
       0    sp_instr_cpush('select a from ...')
       1    sp_instr_copen(0)                  # The 0'th cursor
       2    sp_instr_cfetch(0)                 # Contains the variable list
       3    sp_instr_cclose(0)
       4    sp_instr_cpop(1)
```

The SP cache

There are two ways to cache SPs:

1. one global cache, share by all threads/connections,
2. one cache per thread.

There are pros and cons with both methods:

1.

- Pros: Save memory, each SP only read from table once,
- Cons: Needs locking (= serialization at access), requires thread-safe data structures,

1.

- Pros: Fast, no locking required (almost), limited thread-safe requirement,

- Cons: Uses more memory, each SP read from table once per thread.

Unfortunately, we cannot use alternative 1 for the time being, as most of the data structures to be cached (lex and items) are not reentrant and thread-safe. (Things are modified at execution, we have THD pointers stored everywhere, etc.)
This leaves us with alternative 2, one cache per thread; or actually two, since we keep FUNCTIONs and PROCEDUREs in separate caches.
This is not that terrible; the only case when it will perform significantly worse than a global cache is when we have an application where new threads are connecting, calling a procedure, and disconnecting, over and over again.

The cache implementation itself is simple and straightforward, a hashtable wrapped in a class and a C API (see APIs below).

There is however one issue with multiple caches: dropping and altering procedures. Normally, this should be a very rare event in a running system; it's typically something you do during development and testing, so it's not unthinkable that we would simply ignore the issue and let any threads running with a cached version of an SP keep doing so until its disconnected.
But assuming we want to keep the caches consistent with respect to drop and alter, it can be done:

1. A global counter is needed, initialized to 0 at start.
2. At each DROP or ALTER, increase the counter by one.
3. Each cache has its own copy of the counter, copied at the last read.
4. When looking up a name in the cache, first check if the global counter is larger than the local copy.
   If so, clear the cache and return "not found", and update the local counter; otherwise, lookup as usual.

This minimizes the cost to a single brief lock for the access of an integer when operating normally. Only in the event of an actual drop or alter, is the cache cleared. This may seem to be drastic, but since we assume that this is a rare event, it's not a problem.
It would of course be possible to have a much more fine-grained solution, keeping track of each SP, but the overhead of doing so is not worth the effort.

Class and Function APIs

This is an outline of the key types. Some types and other details in the actual files have been omitted for readability.

The parser context: sp_pcontext.h

```
typedef enum
      {
        sp_param_in,
        sp_param_out,
        sp_param_inout
      } sp_param_mode_t;

      typedef struct
      {
        LEX_STRING name;
        enum enum_field_types type;
        sp_param_mode_t mode;
        uint offset;                    // Offset in current frame
        my_bool isset;
      } sp_pvar_t;

      typedef struct sp_cond_type
      {
        enum { number, state, warning, notfound, exception } type;
        char sqlstate[6];
        uint mysqlerr;
      } sp_cond_type_t;

      class sp_pcontext
      {
        sp_pcontext();

        // Return the maximum frame size
        uint max_framesize();

        // Return the current frame size
        uint current_framesize();

        // Return the number of parameters
        uint params();

        // Set the number of parameters to the current frame size
        void set_params();

        // Set type of the variable at offset 'i' in the frame
        void set_type(uint i, enum enum_field_types type);

        // Mark the i:th variable to "set" (i.e. having a value) with
        // 'val' true.
        void set_isset(uint i, my_bool val);

        // Push the variable 'name' to the frame.
        void push_var(LEX_STRING *name,
                      enum enum_field_types type, sp_param_mode_t mode);

        // Pop 'num' variables from the frame.
        void pop_var(uint num = 1);

        // Find variable by name
        sp_pvar_t *find_pvar(LEX_STRING *name);

        // Find variable by index
        sp_pvar_t *find_pvar(uint i);

        // Push label 'name' of instruction index 'ip' to the label context
        sp_label_t *push_label(char *name, uint ip);

        // Find label 'name' in the context
        sp_label_t *find_label(char *name);

        // Return the last pushed label
        sp_label_t *last_label();
```

```
        // Return and remove the last pushed label.
        sp_label_t *pop_label();

        // Push a condition to the context
        void push_cond(LEX_STRING *name, sp_cond_type_t *val);

        // Pop a 'num' condition from the context
        void pop_cond(uint num);

        // Find a condition in the context
        sp_cond_type_t *find_cond(LEX_STRING *name);

        // Increase the handler count
        void add_handler();

        // Returns the handler count
        uint handlers();

    // Push a cursor
        void push_cursor(LEX_STRING *name);

    // Find a cursor
    my_bool find_cursor(LEX_STRING *name, uint *poff);

    // Pop 'num' cursors
    void pop_cursor(uint num);

    // Return the number of cursors
    uint cursors();
      }
```

Run-time context (call frame): sp_rcontext.h:

```
#define SP_HANDLER_NONE      0
    #define SP_HANDLER_EXIT      1
    #define SP_HANDLER_CONTINUE  2
    #define SP_HANDLER_UNDO      3

    typedef struct
    {
      struct sp_cond_type *cond;
      uint handler;             // Location of handler
      int type;
      uint foffset;             // Frame offset for the handlers declare level
    } sp_handler_t;

    class sp_rcontext
    {
      // 'fsize' is the max size of the context, 'hmax' the number of handlers,
      // 'cmax' the number of cursors
      sp_rcontext(uint fsize, uint hmax, , uint cmax);

      // Push value (parameter) 'i' to the frame
      void push_item(Item *i);

      // Set slot 'idx' to value 'i'
      void set_item(uint idx, Item *i);

      // Return the item in slot 'idx'
      Item *get_item(uint idx);

      // Set the "out" index 'oidx' for slot 'idx. If it's an IN slot,
      // use 'oidx' -1.
      void set_oindex(uint idx, int oidx);

      // Return the "out" index for slot 'idx'
      int get_oindex(uint idx);

      // Set the FUNCTION result
      void set_result(Item *i);

      // Get the FUNCTION result
      Item *get_result();

      // Push handler at location 'h' for condition 'cond'. 'f' is the
      // current variable frame size.
      void push_handler(sp_cond_type_t *cond, uint h, int type, uint f);

      // Pop 'count' handlers
      void pop_handlers(uint count);

      // Find a handler for this error. This sets the state for a found
      // handler in the context. If called repeatedly without clearing,
      // only the first call's state is kept.
      int find_handler(uint sql_errno);

      // Returns 1 if a handler has been found, with '*ip' and '*fp' set
      // to the handler location and frame size respectively.
      int found_handler(uint *ip, uint *fp);

      // Clear the found handler state.
      void clear_handler();

      // Push a return address for a CONTINUE handler
      void push_hstack(uint ip);

      // Pop the CONTINUE handler return stack
      uint pop_hstack();

      // Save variables from frame index 'fp' and up.
      void save_variables(uint fp);
```

```
    // Restore saved variables from to frame index 'fp' and up.
    void restore_variables(uint fp);

    // Push a cursor for the statement (lex)
    void push_cursor(LEX *lex);

    // Pop 'count' cursors
    void pop_cursors(uint count);

    // Pop all cursors
    void pop_all_cursors();

    // Get the 'i'th cursor
    sp_cursor *get_cursor(uint i);

  }
```

The procedure: sp_head.h:

```
#define TYPE_ENUM_FUNCTION  1
      #define TYPE_ENUM_PROCEDURE 2

      class sp_head
      {
        int m_type;             // TYPE_ENUM_FUNCTION or TYPE_ENUM_PROCEDURE

        sp_head();

        void init(LEX_STRING *name, LEX *lex, LEX_STRING *comment, char suid);

        // Store this procedure in the database. This is a wrapper around
        // the function sp_create_procedure().
        int create(THD *);

        // Invoke a FUNCTION
        int
        execute_function(THD *thd, Item **args, uint argcount, Item **resp);

        // CALL a PROCEDURE
        int
        execute_procedure(THD *thd, List<Item> *args);

        // Add the instruction to this procedure.
        void add_instr(sp_instr *);

        // Returns the number of instructions.
        uint instructions();

        // Returns the last instruction
        sp_instr *last_instruction();

        // Resets lex in 'thd' and keeps a copy of the old one.
        void reset_lex(THD *);

        // Restores lex in 'thd' from our copy, but keeps some status from the
        // one in 'thd', like ptr, tables, fields, etc.
        void restore_lex(THD *);

        // Put the instruction on the backpatch list, associated with
        // the label.
        void push_backpatch(sp_instr *, struct sp_label *);

        // Update all instruction with this label in the backpatch list to
        // the current position.
        void backpatch(struct sp_label *);

        // Returns the SP name (with optional length in '*lenp').
        char *name(uint *lenp = 0);

        // Returns the result type for a function
        Item_result result();

        // Sets various attributes
        void sp_set_info(char *creator, uint creatorlen,
                         longlong created, longlong modified,
                         bool suid, char *comment, uint commentlen);
      }
```

Instructions

The base class

```
class sp_instr
        {
          // 'ip' is the index of this instruction
          sp_instr(uint ip);

          // Execute this instruction.
          // '*nextp' will be set to the index of the next instruction
          // to execute. (For most instruction this will be the
          // instruction following this one.)
          // Returns 0 on success, non-zero if some error occurred.
          virtual int execute(THD *, uint *nextp)
        }
<<code>>

===== Statement instruction
<<code>>
        class sp_instr_stmt : public sp_instr
        {
          sp_instr_stmt(uint ip);

          int execute(THD *, uint *nextp);

          // Set the statement's Lex
          void set_lex(LEX *);

          // Return the statement's Lex
          LEX *get_lex();
        }
```

SET instruction

```
class sp_instr_set : public sp_instr
        {
          // 'offset' is the variable's frame offset, 'val' the value,
          // and 'type' the variable type.
          sp_instr_set(uint ip,
                       uint offset, Item *val, enum enum_field_types type);

          int execute(THD *, uint *nextp);
        }
```

Unconditional jump

```
class sp_instr_jump : public sp_instr
        {
          // No destination, must be set.
          sp_instr_jump(uint ip);

          // 'dest' is the destination instruction index.
          sp_instr_jump(uint ip, uint dest);

          int execute(THD *, uint *nextp);

          // Set the destination instruction 'dest'.
          void set_destination(uint dest);
        }
```

Conditional jump

```
class sp_instr_jump_if_not : public sp_instr_jump
        {
          // Jump if 'i' evaluates to false. Destination not set yet.
          sp_instr_jump_if_not(uint ip, Item *i);

          // Jump to 'dest' if 'i' evaluates to false.
          sp_instr_jump_if_not(uint ip, Item *i, uint dest)

          int execute(THD *, uint *nextp);
        }
```

Return a function value

```
class sp_instr_freturn : public sp_instr
        {
          // Return the value 'val'
          sp_instr_freturn(uint ip, Item *val, enum enum_field_types type);

          int execute(THD *thd, uint *nextp);
        }
```

Push a handler and jump

```
class sp_instr_hpush_jump : public sp_instr_jump
        {
          // Push handler of type 'htype', with current frame size 'fp'
          sp_instr_hpush_jump(uint ip, int htype, uint fp);

          int execute(THD *thd, uint *nextp);

          // Add condition for this handler
          void add_condition(struct sp_cond_type *cond);
        }
```

Pops handlers

```
class sp_instr_hpop : public sp_instr
        {
          // Pop 'count' handlers
          sp_instr_hpop(uint ip, uint count);

          int execute(THD *thd, uint *nextp);
        }
```

Return from a CONTINUE handler

```
class sp_instr_hreturn : public sp_instr
        {
          // Return from handler, and restore variables to 'fp'.
          sp_instr_hreturn(uint ip, uint fp);

          int execute(THD *thd, uint *nextp);
        }
```

Push a CURSOR

```
class sp_instr_cpush : public sp_instr_stmt
    {
          // Push a cursor for statement 'lex'
      sp_instr_cpush(uint ip, LEX *lex)

      int execute(THD *thd, uint *nextp);
        }
```

Pop CURSORs

```
class sp_instr_cpop : public sp_instr_stmt
    {
          // Pop 'count' cursors
      sp_instr_cpop(uint ip, uint count)

      int execute(THD *thd, uint *nextp);
        }
```

Open a CURSOR

```
class sp_instr_copen : public sp_instr_stmt
    {
          // Open the 'c'th cursor
      sp_instr_copen(uint ip, uint c);

      int execute(THD *thd, uint *nextp);
        }
```

Close a CURSOR

```
class sp_instr_cclose : public sp_instr
    {
          // Close the 'c'th cursor
      sp_instr_cclose(uint ip, uint c);

      int execute(THD *thd, uint *nextp);
        }
```

Fetch a row with CURSOR

```
class sp_instr_cfetch : public sp_instr
    {
          // Fetch next with the 'c'th cursor
      sp_instr_cfetch(uint ip, uint c);

      int execute(THD *thd, uint *nextp);

      // Add a target variable for the fetch
      void add_to_varlist(struct sp_pvar *var);
        }
```

Utility functions: sp.h

```
#define SP_OK                0
      #define SP_KEY_NOT_FOUND     -1
      #define SP_OPEN_TABLE_FAILED -2
      #define SP_WRITE_ROW_FAILED  -3
      #define SP_DELETE_ROW_FAILED -4
      #define SP_GET_FIELD_FAILED  -5
      #define SP_PARSE_ERROR       -6

      // Finds a stored procedure given its name. Returns NULL if not found.
      sp_head *sp_find_procedure(THD *, LEX_STRING *name);

      // Store the procedure 'name' in the database. 'def' is the complete
      // definition string ("create procedure ...").
      int sp_create_procedure(THD *,
                              char *name, uint namelen,
                              char *def, uint deflen,
                              char *comment, uint commentlen, bool suid);

      // Drop the procedure 'name' from the database.
      int sp_drop_procedure(THD *, char *name, uint namelen);

      // Finds a stored function given its name. Returns NULL if not found.
      sp_head *sp_find_function(THD *, LEX_STRING *name);

      // Store the function 'name' in the database. 'def' is the complete
      // definition string ("create function ...").
      int sp_create_function(THD *,
                             char *name, uint namelen,
                             char *def, uint deflen,
                             char *comment, uint commentlen, bool suid);

      // Drop the function 'name' from the database.
      int sp_drop_function(THD *, char *name, uint namelen);
```

The cache: sp_cache.h

```
/* Initialize the SP caching once at startup */
      void sp_cache_init();

      /* Clear the cache *cp and set *cp to NULL */
      void sp_cache_clear(sp_cache **cp);

      /* Insert an SP to cache. If **cp points to NULL, it's set to a
         new cache */
      void sp_cache_insert(sp_cache **cp, sp_head *sp);

      /* Lookup an SP in cache */
      sp_head *sp_cache_lookup(sp_cache **cp, char *name, uint namelen);

      /* Remove an SP from cache */
      void sp_cache_remove(sp_cache **cp, sp_head *sp);
```

The mysql.proc schema

This is the mysql.proc table used in MariaDB 10.4:

```
CREATE TABLE `proc` (
  `db` char(64) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
  `name` char(64) NOT NULL DEFAULT '',
  `type` enum('FUNCTION','PROCEDURE','PACKAGE','PACKAGE BODY') NOT NULL,
  `specific_name` char(64) NOT NULL DEFAULT '',
  `language` enum('SQL') NOT NULL DEFAULT 'SQL',
  `sql_data_access` enum('CONTAINS_SQL','NO_SQL','READS_SQL_DATA','MODIFIES_SQL_DATA') NOT NULL DEFAULT
'CONTAINS_SQL',
  `is_deterministic` enum('YES','NO') NOT NULL DEFAULT 'NO',
  `security_type` enum('INVOKER','DEFINER') NOT NULL DEFAULT 'DEFINER',
  `param_list` blob NOT NULL,
  `returns` longblob NOT NULL,
  `body` longblob NOT NULL,
  `definer` char(141) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
  `created` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `modified` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `sql_mode`
set('REAL_AS_FLOAT','PIPES_AS_CONCAT','ANSI_QUOTES','IGNORE_SPACE','IGNORE_BAD_TABLE_OPTIONS','ONLY_FULL_
GROUP_BY','NO_UNSIGNED_SUBTRACTION','NO_DIR_IN_CREATE','POSTGRESQL','ORACLE','MSSQL','DB2','MAXDB','NO_KE
Y_OPTIONS','NO_TABLE_OPTIONS','NO_FIELD_OPTIONS','MYSQL323','MYSQL40','ANSI','NO_AUTO_VALUE_ON_ZERO','NO_
BACKSLASH_ESCAPES','STRICT_TRANS_TABLES','STRICT_ALL_TABLES','NO_ZERO_IN_DATE','NO_ZERO_DATE','INVALID_DA
TES','ERROR_FOR_DIVISION_BY_ZERO','TRADITIONAL','NO_AUTO_CREATE_USER','HIGH_NOT_PRECEDENCE','NO_ENGINE_SU
BSTITUTION','PAD_CHAR_TO_FULL_LENGTH','EMPTY_STRING_IS_NULL','SIMULTANEOUS_ASSIGNMENT') NOT NULL DEFAULT
'',
  `comment` text CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `character_set_client` char(32) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `collation_connection` char(32) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `db_collation` char(32) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `body_utf8` longblob DEFAULT NULL,
  `aggregate` enum('NONE','GROUP') NOT NULL DEFAULT 'NONE',
  PRIMARY KEY (`db`,`name`,`type`)
) ENGINE=Aria DEFAULT CHARSET=utf8 PAGE_CHECKSUM=1 TRANSACTIONAL=1 COMMENT='Stored Procedures'
```

This page is licensed: CC BY-SA / Gnu FDL

Connection Memory Usage

> **Note:**
>
> This page contains background information for MariaDB developers.

When creating a connection, a THD object is created for that connection. This contains all connection information and also caches to speed up queries and avoid frequent malloc() calls.

When creating a new connection, the following malloc() calls are done for the THD:

The following information is the state in MariaDB 10.6.1 when compiled without debugging.

Local Thread Memory

This is part of `select memory_used from information_schema.processlist`.

Amount allocated	Where allocated	Description
26646	THD::THD	Allocation of THD object
256	Statement_map::Statement_map(), my_hash_init(key_memory_prepared_statement_map, &st_hash	Prepared statements
256	my_hash_init(key_memory_prepared_statement_map, &names_hash	Names of used prepared statements
128	wsrep_wfc(), Opt_trace_context(), dynamic_array()	
1024	Diagnostics_area::init(),init_sql_alloc(PSI_INSTRUMENT_ME, &m_warn_root	

Amount allocated	Where allocated	Description
120	Session_sysvars_tracker, global_system_variables.session_track_system_variables	Tracking of changed session variables
280	THD::THD,my_hash_init(key_memory_user_var_entry,&user_vars	
280	THD::THD,my_hash_init(PSI_INSTRUMENT_ME, &sequences	Cache of used sequences
1048	THD::THD, m_token_array= my_malloc(PSI_INSTRUMENT_ME, max_digest_length	
16416	CONNECT::create_thd(), my_net_init(), net_allocate_new_packet()	This is for reading data from the connected user
16416	check_connection(), thd->packet.alloc()	This is for sending data to connected user

Objects Stored in THD->memroot During Connect

Amount allocated	Where allocated	Description
72	send_server_handshake_packet, mpvio->cached_server_packet.pkt=	
64	parse_client_handshake_packet, thd->copy_with_error(…db,db_len)	
32	parse_client_handshake_packet, sctx->user=	
368	ACL_USER::copy(), root=	Allocation of ACL_USER object
56	ACL_USER::copy(), dst->user= safe_lexcstrdup_root(root, user)	
56	ACL_USER::copy()	Allocation of other connect attributes
56	ACL_USER::copy()	
64	ACL_USER::copy()	
64	ACL_USER::copy()	
32	mysql_change_db()	Store current db in THD
48	dbname_cache->insert(db_name)	Store db name in db name cache
40	mysql_change_db(), my_register_filename(db.opt)	Store filename db.opt
8216	load_db_opt(), init_io_cache()	Disk cache for reading db.opt
1112	load_db_opts(), put_dbopts()	Cache default database parameters

State at First Call to mysql_execute_command

```
(gdb) p thd->status_var.local_memory_used
$24 = 75496
(gdb) p thd->status_var.global_memory_used
$25 = 17544
(gdb) p thd->variables.query_prealloc_size
$30 = 24576
(gdb) p thd->variables.trans_prealloc_size
$37 = 4096
```

This page is licensed: CC BY-SA / Gnu FDL

Using MariaDB with Your Programs (API)

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See Progress Reporting.

See libmysqld.

See Non-Blocking Client Library.

This page is licensed: CC BY-SA / Gnu FDL

Progress Reporting

> **Note:**
>
> This page contains background information for MariaDB developers.

MariaDB supports progress reporting for some long running commands.

What is Progress Reporting?

Progress reporting means that:

- There is a `Progress` column in SHOW PROCESSLIST which shows the total progress (0-100%)
- INFORMATION_SCHEMA.PROCESSLIST has three columns which allow you to see in which process stage we are and how much of that stage is completed:
  - `STAGE`
  - `MAX_STAGE`
  - `PROGRESS` (within current stage).
- The client receives progress messages which it can display to the user to indicate how long the command will take.

We have separate progress reporting for stages because different stages take different amounts of time.

Supported Commands

Currently, the following commands can send progress report messages to the client:

- ALTER TABLE
- CREATE INDEX
- DROP INDEX
- LOAD DATA INFILE (not `LOAD DATA LOCAL INFILE`, as in that case we don't know the size of the file).

Some Aria storage engine operations also support progress messages:

- CHECK TABLE
- REPAIR TABLE
- ANALYZE TABLE
- OPTIMIZE TABLE

Limitations

Although the above commands support progress reporting, there are some limitations to what progress is reported. To be specific, when executing one of these commands against an InnoDB table with ALGORITHM=INPLACE (which is the default in MariaDB 10.0+), progress is only reported during the merge sort phase while reconstructing indexes.

Enabling and Disabling Progress Reporting

`mysqld` (the MariaDB server) automatically sends progress report messages to clients that support the new protocol, using the value of the progress_report_time variable. They are sent every max(`global.progress_report_time` , `progress_report_time`) seconds (by default 5). You can disable the sending of progress report messages to the client by setting either the local variable (affects only the current connection) or the global variable (affects all connections) to `0` .

If the extra column in `SHOW PROCESSLIST` gives you a compatibility problem, you can disable it by starting `mysqld` with the `--old` flag.

Clients Which Support Progress Reporting

- The mariadb command line client
- The `mytop` that comes with MariaDB has a `'%'` column which shows the progress.

Progress Reporting in the mysql Command Line Client

Progress reporting is enabled by default in the mariadb client. You can disable it with `--disable-progress-reports` . It is automatically disabled in batch mode.

When enabled, for every supported command you get a progress report like:

```
ALTER TABLE my_mail ENGINE=aria;
Stage: 1 of 2 'copy to tmp table'  5.37% of stage done
```

This is updated every progress_report_time seconds (the default is 5). If the global `progress_report_time` is higher, this will be used. You can also disable error reporting by setting the variable to `0` .

How to Add Support for Progress Reporting to a Client

You need to use the MariaDB 5.3 or later client library. You can check that the library supports progress reporting by doing:

```
#ifdef CLIENT_PROGRESS
```

To enable progress reporting to the client you need to add `CLIENT_PROGRESS` to the `connect_flag` in `mysql_real_connect()` :

```
mysql_real_connect(mysql, host, user, password,
                   database, opt_mysql_port, opt_mysql_unix_port,
                   connect_flag | CLIENT_PROGRESS);
```

Then you need to provide a callback function for progress reports:

```
static void report_progress(const MYSQL *mysql, uint stage, uint max_stage,
                            double progress, const char *proc_info,
                            uint proc_info_length);

mysql_options(&mysql, MYSQL_PROGRESS_CALLBACK, (void*) report_progress);
```

The above `report_progress` function will be called for each progress message.

This is the implementation used by `mysql.cc` :

```
uint last_progress_report_length;

static void report_progress(const MYSQL *mysql, uint stage, uint max_stage,
                            double progress, const char *proc_info,
                            uint proc_info_length)
{
  uint length= printf("Stage: %d of %d '%.*s' %6.3g%% of stage done",
                      stage, max_stage, proc_info_length, proc_info,
                      progress);
  if (length < last_progress_report_length)
    printf("%*s", last_progress_report_length - length, "");
  putc('\r', stdout);
  fflush(stdout);
  last_progress_report_length= length;
}
```

If you want only one number for the total progress, you can calculate it with:

```
double total_progress=
 ((stage -1) / (double) max_stage * 100.00 + progress / max_stage);
```

Note: `proc_info` is totally independent of stage. You can have many different `proc_info` values within a stage. The idea behind `proc_info` is to give the user more information about what the server is doing.

How to Add Support for Progress Reporting to a Storage Engine

The functions to use for progress reporting are:

```
void thd_progress_init(MYSQL_THD thd, unsigned int max_stage);
```

Initialize progress reporting with stages. This is mainly used for commands that are totally executed within the engine, like `CHECK TABLE`.
You should not use this for operations that could be called by, for example, `ALTER TABLE` as this has already called the function.

`max_stage` is the number of stages your storage engine will have.

```
void thd_progress_report(MYSQL_THD thd, unsigned long long progress,
                         unsigned long long max_progress);
```

The above is used for reporting progress.

- `progress` is how much of the file/rows/keys you have gone through.
- `max_progress` is the max number of rows you will go through.

You can call this with varying numbers, but normally the ratio `progress/max_progress` should be increasing.

This function can be called even if you are not using stages, for example when enabling keys as part of `ALTER TABLE` or `ADD INDEX`.

```
void thd_progress_next_stage(MYSQL_THD thd);
```

To go to the next stage in a multi-stage process initiated by `thd_progress_init()`:

```
void thd_progress_end(MYSQL_THD thd);
```

End progress reporting; Sets 'Progress' back to 0 in `SHOW PROCESSLIST`.

```
const char *thd_proc_info(thd, 'stage name');
```

This sets the name of the current status/stage that is displayed in `SHOW PROCESSLIST` and in the client. It's recommended that you call this between stages and thus before `thd_progress_report()` and `thd_progress_next_stage()`.

This functions returns the last used `proc_info`. It's recommended that you restore `proc_info` to its original value when you are done processing.

Note: `thd_proc_info()` is totally independent of stage. You can have many different `proc_info` values within a stage to give the user more information about what is going on.

Examples to Look at in the MariaDB Source:

- `client/mysql.cc` for an example of how to use reporting.
- `libmysql/client.c:cli_safe_read()` to see how progress packets are handled in client
- `sql/protocol.cc::net_send_progress_packet()` for how progress packets are handled in server.

Format of Progress Packets

The progress packet is sent as an error packet with error number `65535`.

It contains the following data (in addition to the error header):

Option	Number of bytes	Other info
1	1	Number of strings. For future
Stage	1	Stage from 1 - Max_stage
Max_stage	1	Max number of stages
Progress	3	Progress in % * 1000
Status_length	1-2	Packet length of string in net_field_length() format
Status	Status_length	Status / Stage name

See Also

- What is MariaDB 5.3

This page is licensed: CC BY-SA / Gnu FDL

libmysqld

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See Embedded MariaDB Interface.

See The mariadb-test and mariadb-test-embedded Programs.

This page is licensed: CC BY-SA / Gnu FDL

Embedded MariaDB Interface

> **Note:**
>
> This page contains background information for MariaDB developers.

The embedded MariaDB server, `libmysqld` has the identical interface as the C client library `libmysqclient`.

The normal usage of the embedded server is to use the normal `mysql.h` include file in your application and link with `libmysqld` instead of `libmysqlclient`.

The intention is that one should be able to move from a server/client version of MariaDB to a single server version of MariaDB by just changing which library you link with.

This means that the embedded C client API only changes when the normal C API changes, usually only between major releases.

The only major change required in your application if you are going to use the embedded server is that you have to call the following functions from your application:

```
int mysql_library_init(int argc, char **argv, char **groups)
void mysql_library_end(void);
```

This is also safe to do when using the standard C library.

Notes

- libmysqld.so has many more exported symbols than the C library to allow one to expose and use more parts of MariaDB. In normal applications one should not use them, as they may change between every release.
- The embedded server library and related test binaries are no longer part of binary tarball release archives.

See Also

- mysql_library_init
- mariadb client with MariaDB embedded

This page is licensed: CC BY-SA / Gnu FDL

The mariadb-test and mariadb-test-embedded Programs

> **Note:**
>
> This page contains background information for MariaDB developers.

The `mariadb-test` program runs a test case against a MariaDB or MySQL server and optionally compares the output with a result file. This program reads input written in a special test language. Typically, you invoke `mariadb-test` via mariadb-test-run.pl rather than invoking it directly.

`mariadb-test_embedded` is similar but is built with support for the `libmariadbd` embedded server.

Features of mariadb-test:

- Can send SQL statements to the server for execution
- Can execute external shell commands
- Can test whether the result from an SQL statement or shell command is as expected
- Can connect to one or more standalone mariadbd servers and switch between connections
- Can connect to an embedded server (`libmariadbd`), if MariaDB is compiled with support for `libmariadbd`. (In this case, the executable is named `mariadb-test_embedded` rather than `mariadb-test`.)

By default, `mariadb-test` reads the test case on the standard input. To run mariadb-test this way, you normally invoke it like this:

```
shell> mariadb-test **[options] [db_name]** < //test_file//
```

You can also name the test case file with a `--test-file=file_name` option.

The exit value from `mariadb-test` is `0` for success, `1` for failure, and `62` if it skips the test case (for example, if after checking some preconditions it decides not to run the test).

Options

`mariadb-test` supports the following options:

Option	Description
–help, -?	Display a help message and exit.
–basedir=dir, -b dir	The base directory for tests.
–character-sets-dir=path	The directory where character sets are installed.
–compress, -C	Compress all information sent between the client and the server if both support compression.
–connect-timeout=N	This can be used to set the `MYSQL_OPT_CONNECT_TIMEOUT` parameter of `mysql_options` to change the number of seconds before an unsuccessful connection attempt times out.
–continue-on-error	Continue test even if we got an error. This is mostly useful when testing a storage engine to see what from a test file it can execute, or to find all syntax errors in a newly created big test file
–cursor-protocol	Use cursors for prepared statements.
–database=db_name, -D db_name	The default database to use.
–debug[=debug_options], -#[debug_options]	Write a debugging log if MariaDB is built with debugging support. The default `debug_options` value is `d:t:S:i:O,/tmp/mysqltest.trace` on Unix, and `d:t:i:O,\mysqld.trace` on Windows.
–debug-check	Print some debugging information when the program exits.
–debug-info	Print debugging information and memory and CPU usage statistics when the program exits.
–host=host_name, -h host_name	Connect to the server on the given host.
–logdir=dir_name	The directory to use for log files.

Option	Description
–mark-progress	Write the line number and elapsed time to test_file.progress.
–max-connect-retries=num	The maximum number of connection attempts when connecting to server.
–max-connections=num	The maximum number of simultaneous server connections per client (that is, per test). If not set, the maximum is 128. Minimum allowed limit is 8, maximum is 5120.
–no-defaults	Do not read default options from any option files. If used, this must be the first option.
–non-blocking-api	Use the non-blocking client API for communication.
–overlay-dir=name	Overlay directory.
–password[=password], -p[password]	The password to use when connecting to the server. If you use the short option form (`-p`), you cannot have a space between the option and the password. If you omit the password value following the `--password` or `-p` option on the command line, you are prompted for one.
plugin-dir	Directory for client-side plugins.
–port=port_num, -P port_num	The TCP/IP port number to use for the connection, or `0` for default to, in order of preference, `my.cnf` , `$MYSQL_TCP_PORT` , `/etc/services` , built-in default (`3306`).
–prologue=name	Include the contents of the given file before processing the contents of the test file. The included file should have the same format as other `mariadb-test` test files. This option has the same effect as putting a `--source`` *file_name* command as the first line of the test file.
–protocol=name	The protocol to use for connection (`tcp` , `socket` , `pipe` , `memory`).
–ps-protocol	Use the prepared-statement protocol for communication.
–quiet	Suppress all normal output. This is a synonym for `--silent` .
–record, -r	Record the output that results from running the test file into the file named by the `--result-file` option, if that option is given. It is an error to use this option without also using `--result-file` .
–result-file=file_name, -R file_name	This option specifies the file for test case expected results. `--result-file` , together with `--record` , determines how mariadb-test treats the test actual and expected results for a test case:If the test produces no results, mariadb-test exits with an error message to that effect, unless `--result-file` is given and the named file is an empty file.Otherwise, if `--result-file` is not given, `mariadb-test` sends test results to the standard output. With `--result-file` but not `--record` , `mariadb-test` reads the expected results from the given file and compares them with the actual results. If the results do not match, mariadb-test writes a reject file in the same directory as the result file, outputs a diff of the two files, and exits with an error. With both `--result-file` and `--record` , `mariadb-test` updates the given file by writing the actual test results to it.

Option	Description
–result-format-version=#	Version of the result file format to use.
–server-arg=value, -A value	Pass the argument as an argument to the embedded server. For example, `--server-arg=--tmpdir=/tmp` or `--server-arg=--core`. Up to 64 arguments can be given.
–server-file=file_name, -F file_name	Read arguments for the embedded server from the given file. The file should contain one argument per line.
–shared-memory-base-name	Shared-memory name to use for Windows connections using shared memory to a local server (started with the `--shared-memory` option). Case-sensitive.
–silent, -s	Suppress all normal output.
–sleep=num, -T num	Cause all sleep commands in the test case file to sleep num seconds. This option does not affect real_sleep commands. An option value of `0` can be used, which effectively disables sleep commands in the test case.
–socket=path, -S path	The socket file to use when connecting to localhost (which is the default host).
–sp-protocol	Execute DML statements within a stored procedure. For every DML statement, mariadb-test creates and invokes a stored procedure that executes the statement rather than executing the statement directly.
–ssl	Enable TLS for secure connection (automatically enabled with other flags). Disable with `--skip-ssl`.
–ssl-ca=name	CA file in PEM format (check OpenSSL docs, implies `--ssl`).
–ssl-capath=name	CA directory (check OpenSSL docs, implies `--ssl`).
–ssl-cert=name	X509 cert in PEM format (implies `--ssl`).
–ssl-cipher=name	SSL cipher to use (implies `--ssl`).
–ssl-key=name	X509 key in PEM format (implies `--ssl`).
–ssl-crl=name	Certificate revocation list (implies `--ssl`).

Option	Description
–ssl-crlpath=name	Certificate revocation list path (implies `--ssl`).
–ssl-verify-server-cert	Verify server's "Common Name" in its cert against hostname used when connecting. This option is disabled by default.
–suite-dir=name	Suite directory.
–tail-lines=nn	Specify how many lines of the result to include in the output if the test fails because an SQL statement fails. The default is `0`, meaning no lines of result printed.
–test-file=file_name, -x file_name	Read test input from this file. The default is to read from the standard input.
–timer-file=file_name, -m file_name	If given, the number of microseconds spent running the test will be written to this file. This is used by mariadb-test-run.pl for its reporting.
–tmpdir=dir_name, -t dir_name	The temporary directory where socket files are created.
–user=user_name, -u user_name	The user name to use when connecting to the server.
–verbose, -v	Verbose mode. Print out more information about what the program does.
–version, -V	Display version information and exit.
–view-protocol	Every `SELECT` statement is wrapped inside a view.
–wait-longer-for-timeouts	Wait longer for timeouts. Useful when running under valgrind.

See Also

- New Features for mysqltest in MariaDB

This page is licensed: GPLv2

Non-Blocking Client Library

> **Note:**
>
> This section contains background information, mostly aimed at engineers developing MariaDB features.

See Non-Blocking Operation in the Client Library.

See Non-Blocking API Reference.

See Using the Non-Blocking Library.

This page is licensed: CC BY-SA / Gnu FDL

Non-Blocking Operation in the Client Library

> **Note:**
>
> This page contains background information for MariaDB developers.

MariaDB supports *non-blocking* operations in the client library. This allows an application to start a query or other operation against the database, and then continue to do other work (in the same thread) while the request is sent over the network, the query is processed in the server, and the result travels back. As parts of the result become ready, the application can — at its leisure — call back into the library to continue processing, repeating this until the operation is completed.

Non-blocking operation is implemented entirely within the client library. This means no special server support is necessary and non-blocking operation works with any version of the MariaDB or MySQL server, the same as the normal blocking API. It also means that it is not possible to have two queries running at the same time on the same connection (this is a protocol limitation). But a single thread can have any number of non-blocking queries running at the same time, each using its own MYSQL connection object.

Non-blocking operation is useful when an application needs to run a number of independent queries in parallel at the same time, to speed up operation compared to running them sequentially one after the other. This could be multiple queries against a single server (to better utilize multiple CPU cores and/or a high-capacity I/O system on the server), or it could be queries against multiple servers (e.g. SHOW STATUS against all running servers for monitoring, or a map/reduce-like operation against a big sharded database).

Non-blocking operation is also very useful in applications that are already written in a non-blocking style, for example using a framework like libevent, or, for example, a GUI-application using an event loop. Using the non-blocking client library allows the integrations of database queries into such applications, without the risk of long-running queries “hanging” the user interface or stalling the event loop, and without having to manually spawn separate threads to run the queries and re-synchronize with the threads to get the results back.

In this context, “blocking” means the situation where communication on the network socket to the server has to wait while processing the query. Waiting can be necessary because the server has not yet had time to process the query, or because the data needs to travel over the network from the server, or even because the first part of a large request needs to be sent out on the network before local socket buffers can accept the last part. Whenever such a wait is necessary, control returns to the application. The application will then run `select()` or `poll()` (or something similar) to detect when any wait condition is satisfied, and then call back into the library to continue processing.

An example program is available in the MariaDB source tree:

```
tests/async_queries.c
```

It uses `libevent` to run a set of queries in parallel from within a single thread / event loop. This is a good example of how to integrate non-blocking query processing into an event-based framework.

The non-blocking API in the client library is entirely optional. The new library is completely ABI- and source-compatible with existing applications. Also, applications not using non-blocking operations are not affected, nor is there any significant performance penalty for having support for non-blocking operations in the library for applications which do not use them.

The library internally uses co-routines, and requires a co-routine implementation to work. Native implementations are included for i386, amd64, and (since Connector/C version 3.3.12) aarch64 architectures. For other architectures, a fallback to `ucontext` is automatically used if available. An alternate fallback `boost::context` can also be used instead of `ucontext` by building with `-DWITH_BOOST_CONTEXT=ON` (`boost::context` is not used by default). If no co-routine implementation is available, the non-blocking operations are disabled and do not work.

This page is licensed: CC BY-SA / Gnu FDL

Non-Blocking API Reference

> **Note:**
>
> This page contains background information for MariaDB developers.

Here is a list of all functions in the non-blocking client API and their parameters. Apart from operating in a non-blocking way, they all work exactly the same as their blocking counterparts, so their exact semantics can be obtained from the documentation of the normal client API.

The API also contains the following three functions which are used to get the socket `fd` and `timeout` values when waiting for events to occur:

```
my_socket mysql_get_socket(const MYSQL *mysql)
```

Return the descriptor of the socket used for the connection.

```
unsigned int STDCALL mysql_get_timeout_value(const MYSQL *mysql)
```

This should only be called when a `_start()` or `_cont()` function returns a value with the `MYSQL_WAIT_TIMEOUT` flag set. In this case, it returns the value, in seconds, after which a timeout has occurred and the application should call the appropriate `_cont()` function passing `MYSQL_WAIT_TIMEOUT` as the event that occurred.

This is used to handle connection and read timeouts.

```
unsigned int STDCALL mysql_get_timeout_value_ms(const MYSQL *mysql)
```

Like `mysql_get_timeout_value()`, this should only be called when a `_start()` or `_cont()` function returns a value with the `MYSQL_WAIT_TIMEOUT` flag set. In this case, it returns the value, in milliseconds, after which a timeout has occurred and the application should call the appropriate `_cont()` function passing `MYSQL_WAIT_TIMEOUT` as the event that occurred.

The difference to `mysql_get_timeout_value()` is that this provides millisecond resolution for timeouts, rather than just whole seconds. Internal timeouts can be in milliseconds.

At the end is a list of all functions from the normal API which can be used safely in a non-blocking program, since they never need to block.

```
int mysql_real_connect_start(MYSQL **ret, MYSQL *mysql, const char *host,
                         const char *user, const char *passwd, const char *db,
                         unsigned int port, const char *unix_socket,
                         unsigned long client_flags)
```

```
int mysql_real_connect_cont(MYSQL **ret, MYSQL *mysql, int ready_status)
```

`mysql_real_connect_start()` initiates a non-blocking connection request to a server.

When `mysql_real_connect_start()` or `mysql_real_connect_cont()` returns zero, a copy of the passed '`mysql`' argument is stored in `*ret`.

```
int mysql_real_query_start(int *ret, MYSQL *mysql, const char *stmt_str,
unsigned long length)
int mysql_real_query_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_fetch_row_start(MYSQL_ROW *ret, MYSQL_RES *result)
int mysql_fetch_row_cont(MYSQL_ROW *ret, MYSQL_RES *result, int ready_status)
```

Initiate fetch of another row from a `SELECT` query.

If the `MYSQL_RES` was obtained from `mysql_use_result()`, then this function allows stream processing, where initial rows are returned to the application while the server is still sending subsequent rows. When no more data is available on the socket, `mysql_fetch_row_start()` or `mysql_fetch_row_cont()` return `MYSQL_WAIT_READ` (or possibly `MYSQL_WAIT_WRITE` if using TLS and TLS re-negotiation is needed; also `MYSQL_WAIT_TIMEOUT` may be set if read timeout is enabled). When data becomes available, more rows can be fetched with `mysql_fetch_row_cont()`.

If the `MYSQL_RES` was obtained from `mysql_store_result()` or `mysql_store_result_start()` / `mysql_store_result_cont()`, this function cannot block — `mysql_fetch_row_start()` always returns `0` (and if desired, `plain mysql_fetch_row()` may be used instead with equivalent effect).

```
int mysql_set_character_set_start(int *ret, MYSQL *mysql, const char *csname)
int mysql_set_character_set_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
mysql_select_db_start(int *ret, MYSQL *mysql, const char *db)
int mysql_select_db_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_send_query_start(int *ret, MYSQL *mysql, const char *q, unsigned long length)
int mysql_send_query_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_store_result_start(MYSQL_RES **ret, MYSQL *mysql)
int mysql_store_result_cont(MYSQL_RES **ret, MYSQL *mysql, int ready_status)
```

```
int mysql_free_result_start(MYSQL_RES *result)
int mysql_free_result_cont(MYSQL_RES *result, int ready_status)
```

This function can need to wait if not all rows were fetched before it was called (then it needs to consume any pending rows sent from the server so they do not interfere with any subsequent queries sent).

If all rows were already fetched, then this function does not need to wait. `mysql_free_result_start()` will return zero (or if so desired, plain `mysql_free_result()` may be used instead).

Note that `mysql_free_result()` returns no value, so there is no extra '`ret`' parameter for `mysql_free_result_start()` or `mysql_free_result_cont()`.

```
int mysql_close_start(MYSQL *sock)
int mysql_close_cont(MYSQL *sock, int ready_status)
```

`mysql_close()` sends a `COM_QUIT` request to the server, though it does not wait for any reply.

Thus theoretically it can block (if the socket buffer is full), though in practice it is probably unlikely to occur frequently.

The non-blocking version of `mysql_close()` is provided for completeness; for many applications using the normal `mysql_close()` is probably sufficient (and may be simpler).

Note that `mysql_close()` returns no value, so there is no extra '`ret`' parameter for `mysql_close_start()` or `mysql_close_cont()`.

```
int mysql_change_user_start(my_bool *ret, MYSQL *mysql, const char *user, const
                            char *passwd, const char *db)
int mysql_change_user_cont(my_bool *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_query_start(int *ret, MYSQL *mysql, const char *q)
int mysql_query_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_shutdown_start(int *ret, MYSQL *mysql, enum mysql_enum_shutdown_level
                         shutdown_level)
int mysql_shutdown_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_dump_debug_info_start(int *ret, MYSQL *mysql)
int mysql_dump_debug_info_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_refresh_start(int *ret, MYSQL *mysql, unsigned int refresh_options)
int mysql_refresh_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_kill_start(int *ret, MYSQL *mysql, unsigned long pid)
int mysql_kill_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_set_server_option_start(int *ret, MYSQL *mysql,
                              enum enum_mysql_set_option option)
int mysql_set_server_option_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_ping_start(int *ret, MYSQL *mysql)
int mysql_ping_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_stat_start(const char **ret, MYSQL *mysql)
int mysql_stat_cont(const char **ret, MYSQL *mysql, int ready_status)
```

```
int mysql_list_dbs_start(MYSQL_RES **ret, MYSQL *mysql, const char *wild)
int mysql_list_dbs_cont(MYSQL_RES **ret, MYSQL *mysql, int ready_status)
```

```
int mysql_list_tables_start(MYSQL_RES **ret, MYSQL *mysql, const char *wild)
int mysql_list_tables_cont(MYSQL_RES **ret, MYSQL *mysql, int ready_status)
```

```
int mysql_list_processes_start(MYSQL_RES **ret, MYSQL *mysql)
int mysql_list_processes_cont(MYSQL_RES **ret, MYSQL *mysql, int ready_status)
```

```
int mysql_list_fields_start(MYSQL_RES **ret, MYSQL *mysql, const char *table,
                        const char *wild)
int mysql_list_fields_cont(MYSQL_RES **ret, MYSQL *mysql, int ready_status)
```

```
int mysql_read_query_result_start(my_bool *ret, MYSQL *mysql)
int mysql_read_query_result_cont(my_bool *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_stmt_prepare_start(int *ret, MYSQL_STMT *stmt, const char *query,
                         unsigned long length)
int mysql_stmt_prepare_cont(int *ret, MYSQL_STMT *stmt, int ready_status)
```

```
int mysql_stmt_execute_start(int *ret, MYSQL_STMT *stmt)
int mysql_stmt_execute_cont(int *ret, MYSQL_STMT *stmt, int ready_status)
```

```
int mysql_stmt_fetch_start(int *ret, MYSQL_STMT *stmt)
int mysql_stmt_fetch_cont(int *ret, MYSQL_STMT *stmt, int ready_status)
```

```
int mysql_stmt_store_result_start(int *ret, MYSQL_STMT *stmt)
int mysql_stmt_store_result_cont(int *ret, MYSQL_STMT *stmt, int ready_status)
```

```
int mysql_stmt_close_start(my_bool *ret, MYSQL_STMT *stmt)
int mysql_stmt_close_cont(my_bool *ret, MYSQL_STMT *stmt, int ready_status)
```

```
int mysql_stmt_reset_start(my_bool *ret, MYSQL_STMT *stmt)
int mysql_stmt_reset_cont(my_bool *ret, MYSQL_STMT *stmt, int ready_status)
```

```
int mysql_stmt_free_result_start(my_bool *ret, MYSQL_STMT *stmt)
int mysql_stmt_free_result_cont(my_bool *ret, MYSQL_STMT *stmt, int ready_status)
```

```
int mysql_stmt_send_long_data_start(my_bool *ret, MYSQL_STMT *stmt,
                                unsigned int param_number,
                                const char *data, unsigned long length)
int mysql_stmt_send_long_data_cont(my_bool *ret, MYSQL_STMT *stmt, int ready_status)
```

```
int mysql_commit_start(my_bool *ret, MYSQL *mysql)
int mysql_commit_cont(my_bool *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_rollback_start(my_bool *ret, MYSQL *mysql)
int mysql_rollback_cont(my_bool *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_autocommit_start(my_bool *ret, MYSQL *mysql, my_bool auto_mode)
int mysql_autocommit_cont(my_bool *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_next_result_start(int *ret, MYSQL *mysql)
int mysql_next_result_cont(int *ret, MYSQL *mysql, int ready_status)
```

```
int mysql_stmt_next_result_start(int *ret, MYSQL_STMT *stmt)
int mysql_stmt_next_result_cont(int *ret, MYSQL_STMT *stmt, int ready_status)
```

Client API Functions That Never Block

The following client API functions never need to do I/O and thus can never block. Therefore, they can be used as normal in programs using non-blocking operations; no need to call any special `_start()` variant. (Even if a `_start()` variant was available, it would always return zero, so no `_cont()` call would ever be needed).

- mysql_num_rows()
- mysql_num_fields()
- mysql_eof()
- mysql_fetch_field_direct()
- mysql_fetch_fields()

- mysql_row_tell
- mysql_field_tell()
- mysql_field_count()
- mysql_affected_rows()
- mysql_insert_id()
- mysql_errno()
- mysql_error()
- mysql_sqlstate()
- mysql_warning_count()
- mysql_info()
- mysql_thread_id()
- mysql_character_set_name()
- mysql_init()
- mysql_ssl_set()
- mysql_get_ssl_cipher()
- mysql_use_result()
- mysql_get_character_set_info()
- mysql_set_local_infile_handler()
- mysql_set_local_infile_default()
- mysql_get_server_info()
- mysql_get_server_name()
- mysql_get_client_info()
- mysql_get_client_version()
- mysql_get_host_info()
- mysql_get_server_version()
- mysql_get_proto_info()
- mysql_options()
- mysql_data_seek()
- mysql_row_seek()
- mysql_field_seek()
- mysql_fetch_lengths()
- mysql_fetch_field()
- mysql_escape_string()
- mysql_hex_string()
- mysql_real_escape_string()
- mysql_debug()
- myodbc_remove_escape()
- mysql_thread_safe()
- mysql_embedded()
- mariadb_connection()
- mysql_stmt_init()
- mysql_stmt_fetch_column()
- mysql_stmt_param_count()
- mysql_stmt_attr_set()
- mysql_stmt_attr_get()

- mysql_stmt_bind_param()
- mysql_stmt_bind_result()
- mysql_stmt_result_metadata()
- mysql_stmt_param_metadata()
- mysql_stmt_errno()
- mysql_stmt_error()
- mysql_stmt_sqlstate()
- mysql_stmt_row_seek()
- mysql_stmt_row_tell()
- mysql_stmt_data_seek()
- mysql_stmt_num_rows()
- mysql_stmt_affected_rows()
- mysql_stmt_insert_id()
- mysql_stmt_field_count()
- mysql_more_results()
- mysql_get_socket()
- mysql_get_timeout_value
- mysql_get_timeout_value_ms

This page is licensed: CC BY-SA / Gnu FDL

Using the Non-Blocking Library

> **Note:**
>
> This page contains background information for MariaDB developers.

Overview

The MariaDB non-blocking client API is modeled after the normal blocking library calls. This makes it easy to learn and remember. It makes it easier to translate code from using the blocking API to using the non-blocking API (or vice versa). And it also makes it simple to mix blocking and non-blocking calls in the same code path.

For every library call that may block on socket I/O, such as '`int mysql_real_query(MYSQL, query, query_length)`', two additional non-blocking calls are introduced:

```
int mysql_real_query_start(&status, MYSQL, query, query_length)
int mysql_real_query_cont(&status, MYSQL, wait_status)
```

To do non-blocking operation, an application first calls `mysql_real_query_start()` instead of `mysql_real_query()`, passing the same parameters.

If `mysql_real_query_start()` returns zero, then the operation completed without blocking, and 'status' is set to the value that would normally be returned from `mysql_real_query()`.

Else, the return value from `mysql_real_query_start()` is a bitmask of events that the library is waiting on. This can be `MYSQL_WAIT_READ`, `MYSQL_WAIT_WRITE`, or `MYSQL_WAIT_EXCEPT`, corresponding to the similar flags for `select()` or `poll()`; and it can include `MYSQL_WAIT_TIMEOUT` when waiting for a timeout to occur (e.g. a connection timeout).

In this case, the application continues other processing and eventually checks for the appropriate condition(s) to occur on the socket (or for timeout). When this occurs, the application can resume the operation by calling `mysql_real_query_cont()`, passing in `'wait_status'` a bitmask of the events which actually occurred.

Just like `mysql_real_query_start()`, `mysql_real_query_cont()` returns zero when done, or a bitmask of events it needs to wait on. Thus the application continues to repeatedly call `mysql_real_query_cont()`, intermixed with other processing of its choice; until zero is returned, after which the result of the operation is stored in `'status'`.

Some calls, like `mysql_option()`, do not do any socket I/O, and so can never block. For these, there are no separate `_start()` or `_cont()` calls. See the "Non-blocking API reference" page for a full list of what functions can and can not block.

The checking for events on the socket / timeout can be done with `select()` or `poll()` or a similar mechanism. Though often it will be done using a higher-level framework (such as libevent), which supplies facilities for registering and acting on such conditions.

The descriptor of the socket on which to check for events can be obtained by calling `mysql_get_socket()`. The duration of any timeout can be obtained from `mysql_get_timeout_value()`.

Here is a trivial (but full) example of running a query with the non-blocking API. The example is found in the MariaDB source tree as `client/async_example.c`. A larger, more realistic example using `libevent` is found as `tests/async_queries.c` in the source code:

```
static void run_query(const char *host, const char *user, const char *password) {
  int err, status;
  MYSQL mysql, *ret;
  MYSQL_RES *res;
  MYSQL_ROW row;

  mysql_init(&mysql);
  mysql_options(&mysql, MYSQL_OPT_NONBLOCK, 0);

  status = mysql_real_connect_start(&ret, &mysql, host, user, password, NULL, 0, NULL, 0);
  while (status) {
    status = wait_for_mysql(&mysql, status);
    status = mysql_real_connect_cont(&ret, &mysql, status);
  }

  if (!ret)
    fatal(&mysql, "Failed to mysql_real_connect()");

  status = mysql_real_query_start(&err, &mysql, SL("SHOW STATUS"));
  while (status) {
    status = wait_for_mysql(&mysql, status);
    status = mysql_real_query_cont(&err, &mysql, status);
  }
  if (err)
    fatal(&mysql, "mysql_real_query() returns error");

  /* This method cannot block. */
  res= mysql_use_result(&mysql);
  if (!res)
    fatal(&mysql, "mysql_use_result() returns error");

  for (;;) {
    status= mysql_fetch_row_start(&row, res);
    while (status) {
      status= wait_for_mysql(&mysql, status);
      status= mysql_fetch_row_cont(&row, res, status);
    }
    if (!row)
      break;
    printf("%s: %s\n", row[0], row[1]);
  }
  if (mysql_errno(&mysql))
    fatal(&mysql, "Got error while retrieving rows");
  mysql_free_result(res);
  mysql_close(&mysql);
}

/* Helper function to do the waiting for events on the socket. */
static int wait_for_mysql(MYSQL *mysql, int status) {
  struct pollfd pfd;
  int timeout, res;

  pfd.fd = mysql_get_socket(mysql);
  pfd.events =
    (status & MYSQL_WAIT_READ ? POLLIN : 0) |
    (status & MYSQL_WAIT_WRITE ? POLLOUT : 0) |
    (status & MYSQL_WAIT_EXCEPT ? POLLPRI : 0);
  if (status & MYSQL_WAIT_TIMEOUT)
    timeout = 1000*mysql_get_timeout_value(mysql);
  else
    timeout = -1;
  res = poll(&pfd, 1, timeout);
  if (res == 0)
    return MYSQL_WAIT_TIMEOUT;
  else if (res < 0)
    return MYSQL_WAIT_TIMEOUT;
  else {
    int status = 0;
```

```
    if (pfd.revents & POLLIN) status |= MYSQL_WAIT_READ;
    if (pfd.revents & POLLOUT) status |= MYSQL_WAIT_WRITE;
    if (pfd.revents & POLLPRI) status |= MYSQL_WAIT_EXCEPT;
    return status;
  }
}
```

Setting MYSQL_OPT_NONBLOCK

Before using any non-blocking operation, it is necessary to enable it first by setting the `MYSQL_OPT_NONBLOCK` option:

```
mysql_options(&mysql, MYSQL_OPT_NONBLOCK, 0);
```

This call can be made at any time — typically it is done at the start, before `mysql_real_connect()`, but it can be done at any time to start using non-blocking operations.

If a non-blocking operation is attempted without setting the `MYSQL_OPT_NONBLOCK` option, the program will typically crash with a `NULL` pointer exception.

The argument for `MYSQL_OPT_NONBLOCK` is the size of the stack used to save the state of a non-blocking operation while it is waiting for I/O and the application is doing other processing. Normally, applications will not have to change this, and it can be passed as zero to use the default value.

Mixing Blocking and Non-Blocking Operation

It is possible to freely mix blocking and non-blocking calls on the same `MYSQL` connection.

Thus, an application can do a normal blocking `mysql_real_connect()` and subsequently do a non-blocking `mysql_real_query_start()`. Or vice versa, do a non-blocking `mysql_real_connect_start()`, and later do a blocking `mysql_real_query()` on the resulting connection.

Mixing can be useful to allow code to use the simpler blocking API in parts of the program where waiting is not a problem. For example establishing the connection(s) at program startup, or doing small quick queries between large, long-running ones.

The only restriction is that any previous non-blocking operation must have finished before starting a new blocking (or non-blocking) operation, see the next section for more.

Terminating a Non-Blocking Operation Early

When a non-blocking operation is started with `mysql_real_query_start()` or another `_start()` function, it must be allowed to finish before starting a new operation. Thus, the application must continue calling `mysql_real_query_cont()` until zero is returned, indicating that the operation is completed. It is not allowed to leave one operation “hanging” in the middle of processing and then start a new one on top of it.

It is, however, permissible to terminate the connection completely with `mysql_close()` in the middle of processing a non-blocking call. A new connection must then be initiated with `mysql_real_connect` before new queries can be run, either with a new `MYSQL` object or reusing the old one.

In the future, we may implement an abort facility to force an on-going operation to terminate as quickly as possible (but it will still be necessary to call `mysql_real_query_cont()` one last time after abort, allowing it to clean up the operation and return immediately with an appropriate error code).

Restrictions

DNS

When `mysql_real_connect_start()` is passed a hostname (as opposed to a local unix socket or an IP address, it may need to look up the hostname in DNS, depending on local host configuration (e.g. if the name is not in `/etc/hosts` or cached). Such DNS lookups do **not** happen in a non-blocking way. This means that `mysql_real_connect_start()` will not return control to the application while waiting for the DNS response. Thus the application may "hang" for some time if DNS is slow or non-functional.

If this is a problem, the application can pass an IP address to `mysql_real_connect_start()` instead of a hostname, which avoids the problem.
The IP address can be obtained by the application with whatever non-blocking DNS lookup operation is available to it from the operating system or event framework used. Alternatively, a simple solution may be to just add the hostname to the local host lookup file (`/etc/hosts` on Posix/Unix/Linux machines).

Windows Named Pipes and Shared Memory Connections

There is no support in the non-blocking API for connections using Windows named pipes or shared memory.

Named pipes and shared memory can still be used, using either the blocking or the non-blocking API. However, operations that need to wait on I/O on the named pipe do not return control to the application; instead they "hang", waiting for the operation to complete, just like the normal blocking API calls.

This page is licensed: CC BY-SA / Gnu FDL

---

1. Client errors are errors issued by the client, rather than the server. A typical client that issues such errors is the MariaDB command-line client, mariadb.↩
2. Operating system↩
3. FLUSH TABLES WITH READ LOCK↩
4. regular expression↩
5. regular expression↩
6. Key Management System↩
7. Forced Unit Access (FUA) is a feature in storage systems that ensures data is written directly to stable, non-volatile storage before a write command is acknowledged as complete.
   It is implemented as a flag in I/O commands, such as SCSI and SATA commands, and is used to bypass the device's write cache, guaranteeing that data is physically written to the storage medium.
   When the FUA bit is set to 1, the storage device must complete the write operation to the physical media before returning a "good" status, ensuring data durability even in the event of a power failure.↩
8. Direct Memory Access; see https://en.wikipedia.org/wiki/Direct_memory_access for details.↩
9. NUMA (Non-Uniform Memory Access): A hardware architecture where a processor accesses its own local memory faster than non-local memory, requiring database optimization for efficiency.↩
10. RLIMIT_AS stands for Resource Limit: Address Space. It is one of the "ulimits" (user limits) that a system administrator can set to prevent a single process from going rogue.↩
11. ACID compliance refers to a set of properties—Atomicity, Consistency, Isolation, and Durability—that ensure database transactions are processed reliably and maintain data integrity.
    Atomicity guarantees that a transaction is treated as a single, indivisible unit, where all operations succeed or the entire transaction is rolled back.
    Consistency ensures that a transaction brings the database from one valid state to another, adhering to all defined rules and constraints.
    Isolation mandates that concurrent transactions do not interfere with each other, preserving data accuracy during simultaneous operations.
    Durability ensures that once a transaction is committed, its changes are permanently stored and survive system failures such as crashes or power outages.
    These properties are essential for mission-critical applications in industries like banking, healthcare, and e-commerce, where data accuracy

and reliability are paramount.
While ACID compliance enhances data integrity and user confidence, it can impact performance, particularly under high load, due to the overhead of maintaining strict consistency and concurrency control.
Some modern systems, such as data warehouses, may relax isolation requirements to improve read performance, though they still typically maintain atomicity, consistency, and durability.↩

12. NUMA (Non-Uniform Memory Access): A hardware architecture where a processor accesses its own local memory faster than non-local memory, requiring database optimization for efficiency.↩
13. NUMA (Non-Uniform Memory Access): A hardware architecture where a processor accesses its own local memory faster than non-local memory, requiring database optimization for efficiency.↩
14. DML (Data Manipulation Language): The subset of SQL commands used to add, modify, retrieve, or delete data within existing database tables.↩
15. Database Management System. Popular DBMSs include MariaDB, MySQL, Oracle, PostgreSQL, and MongoDB.↩
16. DDL (Data Definition Language): The subset of SQL commands used to create, modify, or destroy the structure of database objects (like tables, indexes, and databases) rather than the data itself.↩
17. **DDL (Data Definition Language)** is the part of SQL used to **define and change the structure of database objects**.

    It covers operations like:

    - Creating and removing objects: `CREATE`, `DROP`
    - Changing object definitions: `ALTER`
    - Clearing data while keeping the table structure: `TRUNCATE`

    DDL affects **schema metadata** (tables, columns, indexes, constraints, views), not day-to-day row updates like `INSERT` / `UPDATE` / `DELETE`.↩
18. **DBA (Database Administrator)** is the person (or role) responsible for **operating and maintaining databases**.

    Typical DBA work includes **user access**, **backups and recovery**, **performance tuning**, **monitoring**, and **schema changes**.↩
19. In databases, change data capture (CDC) is a set of software design patterns used to determine and track the data that has changed (the “deltas”) so that action can be taken using the changed data. The result is a delta-driven dataset.↩
20. DML (Data Manipulation Language): The subset of SQL commands used to add, modify, retrieve, or delete data within existing database tables.↩
21. DDL (Data Definition Language): The subset of SQL commands used to create, modify, or destroy the structure of database objects (like tables, indexes, and databases) rather than the data itself.↩
22. DDL (Data Definition Language): The subset of SQL commands used to create, modify, or destroy the structure of database objects (like tables, indexes, and databases) rather than the data itself.↩
23. DML (Data Manipulation Language): The subset of SQL commands used to add, modify, retrieve, or delete data within existing database tables.↩
24. DML (Data Manipulation Language): The subset of SQL commands used to add, modify, retrieve, or delete data within existing database tables.↩
25. I/O almost always refers to Disk I/O. Reading from and writing to storage is significantly slower than processing data in RAM. Therefore, excessive logging or poorly indexed queries increase “I/O Wait,” which is often the primary bottleneck for database performance.↩
26. TLS stands for Transport Layer Security. It is a cryptographic protocol designed to provide communications security over a computer network. You likely interact with it every day, as it is the “S” in HTTPS.

    What does it actually do?

    TLS provides three main pillars of security:

    1. Encryption: Hides the data being transferred from third parties.
    2. Authentication: Ensures that the parties exchanging information are who they claim to be.
    3. Integrity: Verifies that the data has not been forged or tampered with during transit.

    ↩
27. Man-in-the-Middle (MitM): A cyberattack where an unauthorized third party secretly intercepts, and potentially alters, the communication between two parties (such as a client and a server) who believe they are talking directly to each other.↩
28. Key Management Service↩

29. A Storage Area Network (SAN) is a dedicated high-speed network that connects storage devices to servers, enabling the storage to appear as if it is directly attached to the server's operating system.↩
30. Engine-Independent Table Statistics↩
31. **Nonce:** Short for "number used once", a nonce is a unique, single-use security token that prevents replay attacks and unauthorized request forgeries by ensuring each transaction or session is distinct and valid.↩
32. **Initialization Vector (IV):** A unique, random starting value used in encryption to ensure that identical data blocks produce different ciphertext, preventing pattern recognition and enhancing overall cryptographic security.↩
33. DML (Data Manipulation Language): The subset of SQL commands used to add, modify, retrieve, or delete data within existing database tables.↩
34. cgroup is short for control group. It's Linux kernel feature that organizes processes into groups to monitor and limit resource usage, such as CPU, memory, and disk I/O, for system stability.↩
35. The OOM (Out of Memory) Killer is a safety mechanism within the Linux kernel designed to prevent a system crash when RAM is completely exhausted.↩
36. **NUMA (Non-Uniform Memory Access):** A hardware architecture where a processor accesses its own local memory faster than non-local memory, requiring database optimization for efficiency.↩
37. HNSW (Hierarchical Navigable Small World): A multi-layered, graph-based algorithm used for lightning-fast vector searches by finding "nearest neighbors."↩
38.
39. Data Modification Language, a part of the SQL language.↩
40. Data Modification Language, a part of the SQL language.↩
41. Data Definition Language, a part of the SQL language.↩
42. CTE (Common Table Expression): A temporary, named result set that exists only for the duration of a single `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement, used to make complex queries more readable.↩
43. CTE (Common Table Expression): A temporary, named result set that exists only for the duration of a single `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement, used to make complex queries more readable.↩
44. Brute Force (BF) conflict resolution mechanism:
    - Transaction Conflicts: In a multi-master environment where multiple nodes can process transactions simultaneously, a conflict can occur if two transactions try to modify the same data at the same time.
    - Brute Force (BF) Abort: When a conflict is detected, one of the conflicting transactions (the "victim") may be "killed" or aborted to allow the other transaction (the "aggressor") to proceed and maintain data consistency across the cluster. This is referred to as a "Brute Force" approach to conflict resolution.
    - Information Schema Table: The "Information schema table displaying history of BF aborts" is a system table (like `wsrep_local_bf_aborts` in Galera Cluster) that tracks the count or history of these brute force transaction terminations. This is a crucial metric for monitoring cluster health and identifying problematic transactions or high-contention ar

    ↩
45.
46. DML (Data Manipulation Language): The subset of SQL commands used to add, modify, retrieve, or delete data within existing database tables.↩
47. Random Query Generator↩
48. Random Query Generator↩
49. WorkLogs↩
50. WorkLog↩
51. Information Schema↩